Manipulator motion planning for high-speed robotic laser cutting

Alexandre Dolgui, Anatol Pashkevich

▶ **To cite this version:**

Alexandre Dolgui, Anatol Pashkevich. Manipulator motion planning for high-speed robotic laser cutting. International Journal of Production Research, 2009, 47 (20), pp.5691-5715. 10.1080/00207540802070967 . hal-00513036

HAL Id: hal-00513036
https://hal.science/hal-00513036v1

Submitted on 1 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

For Peer Review Only

Manipulator motion planning for high-speed robotic laser cutting

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2008-IJPR-0234
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	18-Mar-2008
Complete List of Authors:	Dolgui, Alexandre; Ecole des Mines de St Etienne, Division for Indl Engineering and Computer Sciences Pashkevich, Anatol; Belarusian State University of Informatics and Radioelectronics, Robotic Laboratory
Keywords:	PATH PLANNING, OPTIMIZATION, SIMULATION
Keywords (user):	PATH PLANNING, OPTIMIZATION

scholarONE™
Manuscript Central

Manipulator motion planning for high-speed robotic laser cutting

ALEXANDRE DOLGUI[1] and ANATOL PASHKEVICH[1,2]

[1]*Centre for Industrial Engineering and Computer Science*
Ecole de Mines de Saint Etienne,
158, Cours Fauriel, 42023 Saint Etienne, France
e-mail: *dolgui@emse.fr*

[2]*Department of Automatics and Production Systems*
Ecole des Mines de Nantes,
4 rue Alfred-Kastler, Nantes 44307,France
e-mail: anatol.pashkevich@emn.fr

For Peer Review Only

Keywords: Robot path planning; Laser beam cutting; Multiple-criteria optimisation

Corresponding author:
Prof. Alexandre Dolgui,
Centre for Industrial Engineering and Computer Science
Ecole de Mines de Saint Etienne, 158, cours Fauriel, 42023 Saint Etienne, France
Tel. : +33 (0)4.77.42.01.66, Fax : +33 (0)4.77.42.66.66
E-mail : dolgui@emse.fr

Manipulator motion planning for high-speed robotic laser cutting

Alexandre DOLGUI[1] and Anatol PASHKEVICH[1,2]

[1]*Division for Industrial Engineering and Computer Sciences*
Ecole de Mines de Saint Etienne, 158, Cours Fauriel, 42023 Saint Etienne, France
e-mail: *dolgui@emse.fr*

[2] *Department of Automatics and Production Systems*
Ecole des Mines de Nantes
4 rue Alfred-Kastler, Nantes 44307,France, e-mail: anatol.pashkevich@emn.fr

For Peer Review Only

Abstract: Recent advances in laser technology, and especially the essential increase of the cutting speed, motivate amending the existing robot path methods, which do not allow the complete utilisation of the actuator capabilities as well as neglect some particularities in the mechanical design of the wrist of the manipulator arm. This research addresses the optimisation of the 6-axes robot motions for continuous contour tracking while considering the redundancy caused by the tool axial symmetry. The particular contribution of the paper is in the area of multi-objective path planning using graph-based search space representation. In contrast to previous works, the developed optimisation technique is based on the dynamic programming and explicitly incorporates the verification of the velocity/acceleration constrains. This allows the designer to define interactively their importance with respect to the path-smoothness objectives. In addition, this optimisation technique takes into account the capacity of some manipulator wrist axes for unlimited rotation in order to produce more economical motions. The efficiency of the developed algorithms has been carefully investigated via computer simulation. The presented results are implemented in a commercial software package and verified for real-life applications in the automotive industry.

Keywords: Robot path planning; Laser beam cutting; Multiple-criteria optimisation.

1. Introduction

Laser cutting processes have become increasingly important in a wide range of industrial applications due to their many advantages over other cutting methods. The key features of this technology include high speed, narrow kerf width, small thermal distortions and very low roughness on the edges that allow non-contact processing of complex 3D shapes without additional finishing operations. To ensure high productivity and flexibility, the laser is usually manipulated by either a CNC machine or a 5/6-axis industrial robot, which is programmed off-line using available commercial CAD/CAM systems (Nof, 1999; Gropp et al., 1995; Defaux, 2004; Ion, 2005). However, recent advances in the laser cutting technologies motivate enhancements of existing programming methods and encourage developments of dedicated optimisation algorithms that are in the focus of this paper.

The most important aspect of this problem is related to essential increase of the technologically admissible limit for the cutting speed. At present, depending on the laser configuration, and also on a type and thickness of the processing material, the cutting speed can reach more than 50 m/min. This is comparable with the kinematic capabilities of modern industrial robots, which are used to manipulate the laser beam (Steen, 2003; Rooks, 2004; Schlueter, 2005). For instance, for a typical 6-axis anthropomorphic robot, the forearm axis speeds range from 200 to 300 deg/sec and the wrist axes may be actuated with speed from 300 to 500 deg/sec. However, the maximum Cartesian velocity of the tool is usually lower than 1 m/s and significantly varies over the workspace, imposing critical constrains for the implementation of the high-speed laser cutting process. Hence, in this area, the problem of optimal motion planning for a multi-axe manipulator becomes crucial. However, as follows from our experience, direct application of existing algorithms may produce unfeasible results with non-smooth velocity/acceleration profiles (in spite of a quite satisfactory performance for the lower desired cutting speed). For example, as reported in (Ghany et al., 2006), some existing path planning methods may produce cutting head vibrations at high speeds.

For Peer Review Only

Another motivating factor is caused by significant changes in the mechanical design of the robotic manipulators, which are not taken into account by the existing path optimisation techniques. In particular, one of the recent developments in the field of laser cutting, the Robocut system from Robotic Production Technology (www.rpt.net), employs an anthropomorphic manipulator with a standard architecture for the forearm and an innovative design of the wrist with infinite rotation of the 4th and 5th joints. This special design (without external cables to the robotic wrist) offers essential advantages, since the cycle time losses for the reverse rotations of axes after cutting can be avoided. This solution increases speeds for the wrist axes (up to 900 deg/sec), which allows achieving the Cartesian speed of 2.2m/s and acceleration of 1.4G. However, this is out of the scope of the known motion planning methods.

One of specific features of this problem is caused by a kinematic redundancy. It is clear that, from kinematical point of view, laser beam manipulation requires only five degrees of freedom. Accordingly, former robotic cutting systems employed 5-axis manipulators that possess simpler wrist mechanics but rather require sophisticated algorithms for the on-line solution of the inverse kinematics (Pashkevich, 1997). Most of the modern cutting robotic systems are based on the 6-axis robots. Obviously, this additional axe simplifies robot control and programming, while posing another problem: optimal utilisation of the additional motion capabilities.

This paper focuses on the enhancement of the path planning algorithms for the laser cutting robots by imposing additional constraints on the smoothness of the trajectory and taking into account both the manipulator kinematic redundancy and the ability of some axes for unlimited rotations. The remainder of the paper is organised as follows. Section 2 presents analysis of the previous works. Section 3 is devoted to the problem statement and describes the task model, problem constraints, design variables and objectives. Section 4 includes the main theoretical results and contributions, presenting a search space graph model, path optimisation algorithm, and techniques for the generation of the robot control code. Section 5 contains simulation results and

For Peer Review Only

their analysis. In Section 6, an industrial implementation is reported. Finally, Section 7 summarizes the main contributions and suggests some prospective research directions.

2. Related works

Since the debut of robotic laser cutting, most of the related research focused on robot off-line programming (Backes et al., 1995; Kaierle, 1999). The main reason is that, for this technology, the conventional "manual teaching" method proved to be very tedious and time-consuming, requiring a temporary exclusion of the robot from the manufacturing process and performing the operator-controlled movement of the tool along the cutting path, which has to be properly marked beforehand. In contrast, off-line programming can generate the control code by means of computer graphics, away from the factory floor. Thus, the down time may be significantly reduced, enabling very small batch sizes to become economically feasible (Sendler, 1994; Wittenberg, 1995; Mitsi et al., 2005).

At the moment, there are a number of commercial robotic off-line programming/simulation systems on the market. Some of the more commonly used are em-Workplace (RobCAD) from Tecnomatix Technologies (www.tecnomatix.com), IGRIP from Delmia (http://www.delmia.com), CimStation from Silma (www.silma.com), and Workspace (www.workspace5.com). They offer 3D graphical simulation environments with visual programming capabilities and a wide range of convenient design tools, such as robot selection, robot placement, motion simulation, cycle time calculation, and collision-free path planning. Some of these systems include process-specific modules for laser cutting. In particular, the "Robcad/Cut&Seal" module enables the semiautomatic creation of cutting contours and their conversion into linear, circular and spline motions. A similar software tool is also available from Alma (www.alma.fr) -- "act/cut3D", which possesses some additional convenient features, including defining the head orientation for bevelled cutting, imposing "technological" tolerances, and interactive modification of the tool orientation for each point of the programme.

For Peer Review Only

However there still exists a considerable gap between the capabilities of the commercial robotic CAD systems and the requirements of a particular application. Up to now, the robot programs for many 3D cutting applications (especially for high-speed cutting) are constructed interactively and then are verified using simulation/visualization tools incorporated in the corresponding CAD system. The ultimate goal is the generation of reliable programs automatically from designs and drawings, similar to CNC-machine programming methods (Sun and Tsai, 1994, Bohez et al., 2000; Xu et al., 2002, Kim et al., 2003; Anotaipaiboon and Makhanov, 2005; Wang and Xie, 2005, Bi and Lang, 2007).

One of the first contributions in the off-line programming for the laser cutting robots was done by M. Geiger and his co-workers (University of Erlangen-Nuremberg, Germany). They developed technology-oriented approaches, which simultaneously consider the part geometry, process parameters and some properties of the machine tool (Geiger and Gropp, 1992; Geiger and Kolléra, 1994; Bauer, 1996). However, their techniques may be applied to the non-redundant kinematic structures only and based on the five-axis robots.

For manipulators with six degrees of freedom (which are redundant with respect to 3D cutting requirements), the motion planning problem was firstly addressed by Abe, Shibata and Tanie (Abe et al., 1994; Shibata et al., 1997). These authors proposed a genetic algorithm (GA) that optimises the cutting tool orientation using the evaluation function extracted from the experience of skilled operators. Informally, the operator preferences were defined as follows: “Motion of three basic (1st to 3rd) links with low response should be least so as to minimize movement of centre axis of the end-effector.” Formally, the local optimisation criteria was defined as the weighted squared sum of all axis velocities (from 1st to 6th) together with the velocity of the reference point for the 5th axis. Furthermore, for the overall optimisation with a global criterion, all local criteria were summed over the path. As follows from the presented results, the authors succeeded in the generation of manipulator motions for a relatively slow cutting speed (2m/min). However, there are a number of open questions with the application of this technique to current real-life problems (choosing of

For Peer Review Only

weighting factors, taking into account manipulator velocity/acceleration constraints and joint limits, as well as the convergence of algorithms for high-dimensional input data exported from CAD, etc.). Some recent techniques that employ the general end-effector constraints concept (Yao and Gupta, 2007) also suffer from this drawback.

The above mentioned approach has been enhanced in our previous paper (Pashkevich et al., 2004), which presents a detailed study of multiple optimisation objectives and their incorporation in the global criterion. Compared to other works, our motion planning algorithm possesses essential advantages, since it is based on a graph representation of the search space and on the dynamic programming. The latter ensures a much higher computation speed than the GA, which is very important for industrial applications. This algorithm was successfully implemented on a factory floor and later verified in the automotive industry. Nevertheless, recent advances in laser technology and the significant increase of the technological limit on the cutting speed inspire further improvement of our method.

In the area of robotic motion planning, it is also worth mentioning some other works that are not directly related to cutting, but contain some of theoretical background used in this paper. In robotic literature, a lot of research focuses on the time-optimal (or energy-optimal) motion planning with obstacle avoidance for (i) point-to-point and (ii) specified-path formulations. These researches originated from pioneering papers of Bobrow et al. (1985), Shin and Mckay (1985), Pfeiffer and Johanni (1987), and Shiller and Dubowsky (1991). In most of these publications, redundancy is used to optimise secondary objectives, such as manipulability measures or distance to singularities. Comprehensive reviews on this topic can be found in (Chettibi, 2006, Antonelli et al., 2007).

Another related research area is the robotic motion sequencing (scheduling) where most work concentrates on specific non-trivial cases of the travelling-salesman problem (TSP). For robotic applications, the scheduling problem with a minimum-time objective was first addressed by Rubinovitz and Wysk (1988). Later, the TSP-based method was enhanced by Kim et al. (2005), who suggested a number of heuristics and a problem-oriented genetic algorithm for robotic

For Peer Review Only

welding. For other robotic applications, task scheduling concentrates mainly on mechanical assembly and processing, as well as grasping-type tasks (Yuan and Gu 1999, Ben-Arieh et al. 2003). Some recent results are presented in (Bagchi et al., 2006; Dolgui and Pashkevich, 2006). It is obvious that the TSP-based formulations can not be directly used for robotic cutting, where both the processing contour (i.e. the motion sequence) and the tool speed are specified. However, there are some useful modifications of discrete dynamic programming (Jouaneh et al., 1990; Lee, 1995), which are used in this paper.

Another potential approach to the considered problem is based on the standard redundancy resolution techniques, which usually is founded on the generalised inverse of the manipulator Jacobian or the priority task decomposition (Whitney, 1969; Klein and Huang, 1983; Yoshikawa, 1996; Doty et al., 1996). Here, the basic idea is to introduce additional objectives/constraints to obtain a well-defined problem and then solve it by conventional methods. However, in robotic cutting, applying the kinematic-based redundancy resolution methods may engender chaotic joint motion and high accelerations, which leads to erratic behaviour with unpredictable arm configurations (Duarte and Machado, 2000). In addition, these methods are applied independently (locally) to different points of the path and, consequently, they often fail to get a global optimal solution (Hwang et al., 1994).

Summarising the analysis of the related works, most of existing path planning techniques operate with a single objective, such as the path length or travel time. There are few papers which deals directly with multi-objective path planning, and they are mainly for mobile robot navigation (Fujimura, 1996) or surface manufacturing (Chen et al., 2003). To our knowledge, only our previous paper (Pashkevich et al., 2004) addresses this issue to robotic cutting. Finally, current technological advances motivate further enhancements of corresponding path planning techniques which are in the focus of the paper.

For Peer Review Only

3. Problem statement

Generally, a basic problem in automated robot programming for laser cutting is to find optimal manipulator motions from given CAD models of the (i) processing part, (ii) cutting tool, (iii) robot and (iv) work cell environment in order to satisfy specified design objectives and constraints (Chedmail et al., 1998). Based on these data, the trajectory planner generates an optimal tool path in both the Cartesian and joint coordinate space. This path is further validated by a simulation module. Finally, it is translated into a robot control code. The core for the program generation tool is a set of optimisation routines that are addressed in this section.

3.1. *Cutting task model*

Let us assume that the desired Cartesian path, along which the cutting tool is to be moved, is imported from a CAD system and described by two vector functions as follows:

$$\boldsymbol{C} = \left\{ \boldsymbol{p}(t),\, \boldsymbol{n}(t) : \left|\boldsymbol{n}(t)\right| = 1;\;\; t = k\Delta t \in [0,T];\; k = 0,1,2,\ldots n \right\} \qquad (1)$$

where t is a scalar argument (time); $\boldsymbol{p}(t) \in \boldsymbol{R}^3$ defines the Cartesian coordinates of the tool tip, and $\boldsymbol{n}(t) \in \boldsymbol{R}^3$ is the unit vector of the tool axis direction, which must be normal to the part surface (figure 1). These data can be directly extracted from the graphical model of the part, by defining the processing contour as an "*augmented line*". for example. The path is assumed to be closed and time-uniformly sampled into the sequence of nodes $\{\boldsymbol{p}_k, \boldsymbol{n}_k \mid k = 0,\ldots n\}$, where the first and the last coincide ($\boldsymbol{p}_0 = \boldsymbol{p}_n$; $\boldsymbol{n}_0 = \boldsymbol{n}_n$), and the time-interval length is Δt .

[Insert figure 1 about here]

To describe the tool spatial location, let us also define the Cartesian displacement along the path $\Delta \boldsymbol{p}_k = \boldsymbol{p}_{k+1} - \boldsymbol{p}_k;\;\; k = 0,1,2,\ldots n-1$ and introduce a unit direction vector $\boldsymbol{a}_k = \Delta \boldsymbol{p}_k \,/\, \| \Delta \boldsymbol{p}_k \|$, which is tangent to the part surface and to the direction of the points for the tool motion. For the last node, let us define this vector as $\boldsymbol{a}_n = \boldsymbol{a}_0$. Then, assuming that the vectors $\boldsymbol{a}_k$ and $\boldsymbol{n}_k$ are mutually orthogonal, each node may be associated with the Cartesian frame in which the x-axis is directed

For Peer Review Only

along the path, the z-axis is directed along the cutting tool, and the y-axis is computed in such way that these three axes form a right-handed coordinate frame. The corresponding homogenous transformation matrix is

$$\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{a}_k & \boldsymbol{n}_k \times \boldsymbol{a}_k & \boldsymbol{n}_k & \boldsymbol{p}_k \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times 4} \tag{2}$$

where "×" denotes the vector product. It should be noted that, generally, the CAD-imported vectors $\boldsymbol{n}_k$ and the computed vectors $\boldsymbol{n}_k$ may be slightly non-orthogonal. In this case, it is necessary to adjust $\{\boldsymbol{a}_k\}$ using the classical Gram–Schmidt orthogonalisation:

$$\boldsymbol{a}_k \leftarrow \boldsymbol{a}_k - (\boldsymbol{n}_k^T \boldsymbol{a}_k)\cdot \boldsymbol{n}_k\,; \quad \boldsymbol{a}_k \leftarrow \boldsymbol{a}_k \,/\, \|\boldsymbol{a}_k\|\,.$$

The introduced frame sequence $\{\boldsymbol{H}_k \mid k = 0,...n\}$ is used as a pivot for defining the complete pose of the robotic tool, which is usually determined by six independent parameters (three Cartesian coordinates and three Euler angles). However, since the cutting tool is axially symmetric, the frames $\boldsymbol{H}_k$ can be rotated around corresponding z_k-axes without any influence on the technological process. This one-dimensional redundancy leads to an infinite set of admissible tool locations described by the matrix product

$$\boldsymbol{L}_k(\gamma_k) = \boldsymbol{R}_z(\gamma_k)\cdot \boldsymbol{H}_k\,, \quad \gamma_k \in (-\pi, \pi]\,, \tag{3}$$

where γ_k is an arbitrary scalar parameter and $\boldsymbol{R}_z(\gamma)$ is the standard z-axis rotation matrix.

Another source of redundancy is related to the manipulator posture μ (or the configuration index), which is required for the unique mapping from the task space to the joint coordinate space. So, in total, the robotic task is described by a sequence of locations (3), while the design parameters are represented by the sequence $\{\gamma_k,\ \mu_k \mid k = 0,...n\}$. At this step, the design problem can be formulated in the terms of non-linear programming; however this straightforward approach is not prudent because of high search space in this problem.

For Peer Review Only

3.2. *Constraints*

While planning the robot motions for laser cutting and other curve-tracking applications, a designer must take into account three types of constraints: task, robot kinematic and collision constraints (Hwang et al., 1994). *Task constrains* are expressed in the terms of the tool position/orientation required to perform the assigned task and, for the considered problem, they are described by expressions (3). Robot *kinematics constraints* are caused by the manipulator geometry and expressed as workspace limits as well as limits on the range the manipulator joint coordinates. And *collision constraints* arise from the need to avoid collisions between the robot links, workpiece and workcell components.

To define the kinematic constrains more precisely, let us briefly review some results from robot kinematics. For the anthropomorphic serial manipulators that are usually employed with laser cutting, the direct kinematic model, which defines the tool location $\boldsymbol{L}$ corresponding to the joint coordinate vector $\boldsymbol{q}$, may be always expressed in a closed form, as

$$\boldsymbol{L}(\boldsymbol{q}) = \boldsymbol{T}_{tool} \cdot \left(\prod_{i=1}^{6} {}^{i}\boldsymbol{T}_{i-1}(q_i) \right) \cdot \boldsymbol{T}_{base} \ , \qquad (4)$$

where ${}^{i\text{-}1}\boldsymbol{T}_i$ is the transformation matrix from the (i-1)th to the ith link, q_i is the corresponding joint coordinate, the matrix $\boldsymbol{T}_{tool}$ defines the tool tip location with respect to the manipulator mounting flange, and the matrix $\boldsymbol{T}_{base}$ defines the robot base location in the world coordinate system. Particular expressions for the homogenous matrices ${}^{i\text{-}1}\boldsymbol{T}_i$ for various manipulators can be found in common reference books (Ceccarelli, 2004; Spong et al., 2006), usually they are expressed using the Denavit-Hartenberg (DH) notation:

$${}^{i-1}\boldsymbol{T}_i(q_i) = \begin{bmatrix} cos(q_i) & -\sin(q_i)\cos(\alpha_i) & \sin(q_i)\sin(\alpha_i) & a_i\cos(q_i) \\ \sin(q_i) & \cos(q_i)\cos(\alpha_i) & -cos(q_i)\sin(\alpha_i) & a_i\sin(q_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times 4}$$

For Peer Review Only

where a_i, d_i, α_i are the DH-parameters associated with *i*th link and *i*th joint (the offset distance, link length, and twist angle, respectively).

However, the inverse transformation (from the task coordinates space to the manipulator joint space) is generally non-unique and may not be expressed in a closed form. In practice, robot designers prefer special types of manipulator architecture, which admit closed-form solutions. Otherwise, there exist numerical techniques to deal with this problem (Manocha and Canny, 1994; Pashkevich, 1997; Husty et al., 2007). Within our study, and independent of the solution type (analytical or numerical), the inverse kinematical transformation will be presented in an abstract form as $\boldsymbol{q} = \boldsymbol{f}_c^{-1}(\boldsymbol{L}, \mu)$, where $\mu \in \mathrm{M}$ is the configuration index that determines the manipulator posture, and the set M contains all combinations of admissible postures (*shoulder right/left, elbow up/down, wrist plus/minus*). For most of 6–axis industrial manipulators, the number of different configurations is equal to 8, but generally this can rise up to 16 (Manseur and Doty, 1989).

It should be stressed that typical robot controllers do not allow changing the manipulator configuration while mowing between successive nodes and the use of the Cartesian space on-line interpolation (*LIN* and *CIR* commands). But for cutting applications, this specific kinematic constraint can be released by temporarily switching to the joint-space interpolation for these specific path segments (*PTP* command). So, in contrast to our previous work (Pashkevich et al., 2004), here we do not use separate manifolds for each value of μ. This also causes some modifications at the final stage, when the optimal path is translated into a robot control code.

Using the inverse kinematic transformation, the sequence of the admissible tool locations (5) can be mapped onto the joint coordinate space as follows:

$$C_Q = \left\{ \boldsymbol{Q}_k(\gamma_k, \mu_k) = \boldsymbol{f}_c^{-1}(\boldsymbol{R}_z(\gamma_k) \cdot \boldsymbol{H}_k, \mu) \;\middle|\; \gamma_k \in (-\pi, \pi], \quad \mu_k \in \mathrm{M}, \quad k = 1 \ldots n \right\} \tag{5}$$

considering both the workspace dimension constraints (i.e. inverse kinematics existence) and the joint limits $q_i^{min} < q_i < q_i^{max}$, $i \in I$, $I_q = \{1, \ldots 6\}$. For computational convenience, the constraint violation case is denoted as $\boldsymbol{Q}_k(\gamma_k) = \varnothing$ (this notation is easily implemented in data structures

For Peer Review Only

employed in optimisation algorithms). Concerning the joint limits, it worth mentioning that the latest laser-cutting manipulators allow unlimited rotation of the 4th and 6th axes, so in this case $I_q = \{1, 2, 3, 5\}$.

The collision constraints are managed in a similar way, i.e. their violation leads to $\boldsymbol{Q}_k(\gamma_k) = \varnothing$ and corresponding tool locations are inevitably excluded from a feasible set. The collision detection functions are standard routines of industrial robotic CAD packages, together with the direct/inverse kinematics of the robotic manipulators.

From a practical point of view, the desired path planning algorithm should produce "*smooth motions at reasonable speeds and at reasonable accelerations*". Hence, many of aforementioned authors impose constraints on the *joint torques*. However, addressing the torque constraints requires a rather accurate dynamic model, which is problematic in real-life industrial projects. A practical solution consists of transforming torque constraints into acceleration ones using the manipulator specification data provided by the manufacturer.

Within the frames of the discretised path presentation (1), the joint velocity/acceleration constraints may be expressed via the finite-difference approximation as:

$$\left| q_{i,k} - q_{i,k-1} \right| < \eta_v \cdot \Delta q_i^{(v)} \ , \tag{6}$$

$$\left| q_{i,k} - 2q_{i,k-1} + q_{i,k-2} \right| < \eta_a \cdot \Delta q_i^{(a)} \ , \tag{7}$$

where $\Delta q_i^{(v)} = \dot{q}_i^{max} \Delta t$; $\Delta q_i^{(a)} = \ddot{q}_i^{max} \Delta t^2$; the notations $\Delta \dot{q}_i^{max}$ and $\ddot{q}_i^{max}$ define the maximum *i*th joint velocity and acceleration; and η_v and η_a are the scaling factors to be adjusted by the designer. Hence, the complete set of constraints arising from the technical nature of the problem are summarised in inequality $\boldsymbol{Q}_k(\gamma_k) \neq \varnothing$ and in the expressions (6), (7), which ensure the path existence and its admissible curvature in the joint coordinate space.

For Peer Review Only

3.3. *Design objectives*

In the qualitative terms, the desired manipulator motion should be as smooth as possible while satisfying the contour-tracking and actuator-dependent constraints (7) - (9). This means that the qualitative performance measures should be based on some type of the "*smoothness" and "economy"* measures applied to all manipulator joints (Edan and Nof, 1997; Shoval et al., 1998).

For the considered problem, which is based on the discrete path presentation, the degree of smoothness can be evaluated by the following criteria:

- *total displacement*

$$J_i^{(s)}(\boldsymbol{\gamma}, \boldsymbol{\mu}) = \sum_{k=1}^{n} \left| q_{i,k}(\gamma_k, \mu_k) - q_{i,k-1}(\gamma_{k-1}, \mu_{k-1}) \right|, \tag{8}$$

- *maximum increment* (*speed*)

$$J_i^{(v)}(\boldsymbol{\gamma}, \boldsymbol{\mu}) = \max_k \left| q_{i,k}(\gamma_k, \mu_k) - q_{i,k-1}(\gamma_{k-1}, \mu_{k-1}) \right|, \tag{9}$$

- *coordinate range*

$$J_i^{(\Delta)}(\boldsymbol{\gamma}, \boldsymbol{\mu}) = \max_k \left[q_{i,k}(\gamma_k, \mu_k) \right] - \min_k \left[q_{i,k}(\gamma_k, \mu_k) \right], \tag{10}$$

where i is the joint number, and $\boldsymbol{\gamma}$, $\boldsymbol{\mu}$ are the vectors composed of the design parameters $\gamma_0, \gamma_1, \ldots \gamma_n$ and $\mu_0, \mu_1, \ldots \mu_n$ respectively.

A geometrical explanation of these performance measures is the following. The *displacement* criterion evaluates the total amount of joint motions (without regard to the direction of the motion). The *increment* characterizes the coordinate variation during the time-interval Δt and is proportional to the speed. Finally, the *range* evaluates the extent of the coordinate variation during the motion.

Intuitively, the minimization of each of these criteria should lead to a smoother generated path. However, as follows from related studies, the objectives (8) – (10) may be in competition with each other, i.e. minimizing one of them may degrade the performance in others. Besides, these assessments are computed for each joint coordinate. Thus, the resulting performance measures form a vector. The designer must choose one of the techniques that are usually used for multi-criteria problems (Steuer, 1985; Cheng and Shih, 1997; Ehrgott , 2005).

For Peer Review Only

In this paper, instead of giving preference to a particular criterion or optimisation technique, all the following options are admitted: (i) defining priority of partial objectives or the primary objective; (ii) applying minimax technique, i.e. the worst-case optimisation; (iii) assigning weights to combine multiple criteria in a linear function. Independent of the chosen option, the corresponding vector-optimisation engine must include the scalar-optimisation routines that are developed in the following section.

4. Path planning algorithm

4.1. *Search space discretisation*

A practical method to obtain an optimal (or feasible) solution under the constraints described in sub-section 3.2 is the sampling of the allowable domain for the redundant parameter γ. Such an approach transforms the continues search space into an acyclic directed graph, with the vertices uniquely representing the matrix of the tool location $\boldsymbol{L}$ and the vector of joint coordinates $\boldsymbol{Q}$.

Let us assume that the parameter $\gamma \in [-\pi, \pi]$ is sampled with the step $\Delta\gamma = 2\pi/m_0$, $m_0 \in Z$ and, for each discrete value of γ and each admissible configuration $\mu \in \mathrm{M}$, the locations (5) are tested for kinematic and collision constraints. Then, the admissible locations, which satisfy the constraints, are included in the search space in such manner that each Cartesian-path node $\{\boldsymbol{p}_k, \boldsymbol{n}_k\}$ produces several elements of the data structure $\{\boldsymbol{L}, \boldsymbol{Q}\}$:

$$\{\boldsymbol{p}_k, \boldsymbol{n}_k\} \rightarrow \bigcup_{j=1}^{m_k} \begin{Bmatrix} \boldsymbol{L}_{k,j} \\ \boldsymbol{Q}_{k,j} \end{Bmatrix} \tag{11}$$

where m_k is the number of the successful locations for the kth node. It should be noted that the "*path-smoothness*" constraints (6), (7) are not tested at this stage yet, since they are associated with several successive locations. The test result can be presented in the plane $t \times (\gamma, \mu)$, where the abscissa (index k) corresponds to the time t, and the ordinate (index j) corresponds to the combination of the redundant parameter γ and the configuration index μ (see figure 2).

[Insert figure 2 about here]

For Peer Review Only

Under such assumptions, the feasible search space can be represented by a directed graph with the vertices

$$V = \bigcup_{k=0}^{n} \bigcup_{j=1}^{m_k} \begin{Bmatrix} \boldsymbol{L}_{k,j} \\ \boldsymbol{Q}_{k,j} \end{Bmatrix} \tag{12}$$

and edges

$$E = \bigcup_{k=1}^{n} \bigcup_{j=1}^{m_k} \bigcup_{l=1}^{m_{k-1}} \left(\begin{Bmatrix} \boldsymbol{L}_{k,j} \\ \boldsymbol{Q}_{k,j} \end{Bmatrix}, \begin{Bmatrix} \boldsymbol{L}_{k-1,l} \\ \boldsymbol{Q}_{k-1,l} \end{Bmatrix} \right), \tag{13}$$

which connect only successive tool locations from the entire set (5). Hence, the considered path-planning task is reduced to the following network optimisation problem.

Design Problem. *For given set of vertexes V and set of edges E, find the "best" path* of *length n*

$$\Pi(j_0, j_1, \dots j_n) = \left\langle \begin{Bmatrix} \boldsymbol{L}_{0,j_0} \\ \boldsymbol{Q}_{0,j_0} \end{Bmatrix} \rightarrow \begin{Bmatrix} \boldsymbol{L}_{1,j_1} \\ \boldsymbol{Q}_{1,j_1} \end{Bmatrix} \rightarrow \dots \begin{Bmatrix} \boldsymbol{L}_{n,j_n} \\ \boldsymbol{Q}_{n,j_n} \end{Bmatrix} \right\rangle \tag{14}$$

with initial and final states

$$V_0 \in \bigcup_{j=1}^{m_0} \begin{Bmatrix} \boldsymbol{L}_{0,j} \\ \boldsymbol{Q}_{0,j} \end{Bmatrix} \quad \text{and} \quad V_n \in \bigcup_{j=1}^{m_n} \begin{Bmatrix} \boldsymbol{L}_{n,j} \\ \boldsymbol{Q}_{n,j} \end{Bmatrix},$$

which minimizes a specified performance index.

It should be stressed that, in this formulation, both the initial and final states are not unique, but the problem can be transformed to the classical problem by adding the virtual start and end nodes. Besides, the initial and the final states may not coincide with each other, as it follows from the technical nature of the problem. It is also worth mentioning that all nodes are solvable for the inverse kinematics and admissible for the collision tests (otherwise they are excluded from the graph (*V*,*E*) at the stage of the graph generation).

To solve this network optimisation problem, it is also necessary to define the *distance matrix*, which assigns certain lengths to the graph edges. Since in the Cartesian space the tool path is sampled uniformly, the distance should be based on a joint space metric. While defining this metric, it is necessary to take into account that the actuator capacities (i.e. maximum speed, acceleration,

For Peer Review Only

etc.) are different for different manipulator axes. So, the displacement components Δq_i corresponding to the joint displacement vector, which are $\Delta \boldsymbol{q} = (\Delta q_1, \dots \Delta q_6)$, should be weighted.

The most appropriate way of assigning the weights, is using the maximum axis speed $\dot{q}_i^{max}$ specified by the manufacturer. In this case $w_i = (\dot{q}_i^{max} \Delta t)^{-1}$ and the product $w_i \cdot \Delta q_i$ shows the degree of actuator capacity utilisation with respect to speed. Finally, combining the evaluations for all axes, the distance in the joint space may be defined using the Euclidean, Manhattan or Chebychev metrics (Deza, 2006) that are usually used because of both the computational convenience and their clear physical meaning (Pashkevich et al., 2004):

$$\rho_E(\Delta \boldsymbol{q}) = \sqrt{\sum_{i=1}^{6} (w_i \, \Delta q_i)^2} \; ; \quad \rho_M(\Delta \boldsymbol{q}) = \sum_{i=1}^{6} | w_i \, \Delta q_i | ; \quad \rho_C(\Delta \boldsymbol{q}) = \max_i | w_i \, \Delta q_i | \tag{15}$$

While applying the above expressions, it is necessary to consider that for cutting applications, the 4th and 6th axes may allow unlimited rotation. In this case, before computing the distance $\rho(.)$, the differences Δq_i must be minimised in accordance with the following expression:

$$\Delta q_i = \min_{p \in Z} \left\{ \Delta q_i + 2\pi p \right\}, \quad i \in \{4, 6\} \tag{16}$$

where p is an integer number. For example, for the axes 1, 2, 3, 5, the movement from $q_i^{(k)} = -120°$ to $q_i^{(k+1)} = +120°$ corresponds to the joint coordinate displacement $\Delta q_i = +240°$. However, for the axes 4 and 6 it is equal to $\Delta q_i = -120°$ and the target joint coordinate is must be changed to $q_i^{(k+1)} = -240°$. In practice, there is an open question: which metrics of $\rho_E(.), \rho_M(.), \rho_C(.)$, should be chosen? So, it is proposed to give some freedom to the designer and to include the choice of the corresponding option in the set of parameters for the optimisation algorithm.

4.2. *Generation of optimal path*

Since there is no combinatorial optimisation technique, which is able to solve this multi-objective problem directly, the vector performance measures (8) - (10) should be converted into an aggregate scalar criterion. At this step, a relevant metric is applied to the sequence of the path

For Peer Review Only

segments whose length is computed using one of the expressions (15). Hence, the desired scalar criteria may be based on the non-weighted Manhattan/Chebychev norm. The weights w_i, which are used for calculating the segment distance $\rho(.)$, are altered between optimisation runs to produce a set of Pareto-optimal solutions.

Corresponding expressions for the objective functions may be written as follows:

$$J^{(s)}(\boldsymbol{\gamma}, \boldsymbol{\mu}) = \sum_{k=1}^{n} \rho_a\left(\boldsymbol{Q}(k, j_k) - \boldsymbol{Q}(k-1, j_{k-1})\right) \tag{17}$$

$$J^{(\Delta)}(\boldsymbol{\gamma}, \boldsymbol{\mu}) = \max_k \rho_a\left(\boldsymbol{Q}(k, j_k) - \boldsymbol{Q}(k-1, j_{k-1})\right) \tag{18}$$

where $\rho_a(.)$, $a \in \{E, M, C\}$ is the distance function, the notation j_k defines the values of the redundant parameters γ, μ at the kth node and, for further convenience, the vectors of the joint coordinates $\boldsymbol{Q}_{k,j}$ are denoted as $\boldsymbol{Q}(k,j)$.

For the additive objective $J^{(s)}$, an optimal path can be found by means of dynamic programming. This is a standard shortest-path problem in an acyclic directed graph. Nevertheless due to the graph specific properties (node clustering), it is not reasonable to apply known algorithms (Dijkstra's , Bellman-Ford, etc.).

This problem is also closely related to the generalised travelling salesman problem (GTSP) employed in the motion and process planning for which there are some efficient optimisation routines, see for example (Ben-Arieh et al., 2003; Dolgui and Pashkevich, 2006).

To describe the proposed algorithm in terms of graph theory, let us introduce the following notations. Let $G = (S_0 \bigcup S_1 \bigcup \ldots S_n, E)$ be an acyclic directed graph with the set of vertices $V = \bigcup_{k=0}^{n} S_k$ partitioned into n+1 mutually exclusive and exhaustive subsets (clusters) $S_k = \{v_{kj}, j = 1,\ldots, m_k\}$ of sizes $| S_k | = m_k$. Here $v_{kj} = (k, j)$ denotes the jth vertex of the kth cluster. It is assumed that each edge $(u, v) \in E$ is evaluated by some travelling distance $\rho(u, v)$. The clusters must be visited in a strictly predefined order $S_0 \to S_1 \to \ldots \to S_n$, passing via exactly one vertex in

For Peer Review Only

each cluster. It should be noted that the above formulation does not admit cluster intersection (i.e. $S_k \bigcap S_l = \varnothing, \ \ \forall\, k \neq l$). This is in agreement with usual engineering practices.

For such a formulation, a vertex may be coded as pair (k, j), an edge is identified by quadruplet $(k_1, j_1, \ k_2, j_2)$, and the solution is defined by the array of j-indices $\{J_{opt}(k), \ k=0,1,\ldots n\}$ of the sequential visiting vertices. Also, following this notation, the cluster sizes are defined in the array $\{J_{max}(k), k=0,1,\ldots,n\}$ and the cost of the edges are denoted as $\rho(k_1, j_1, \ k_2, j_2)$.

A basic expression for the developed algorithm is derived in the following way. Let us consider a reduced order sub-problem with k clusters $S_0, S_1, \ldots S_{k-1}$, and let $d^*_{k-1,\,j}$ be the length of the shortest path $S_0 \to S_1 \to \ldots \to S_{k-1}$ that links a vertex $v_{k-1,\,j} \in S_{k-1}$ to the nearest vertex $u \in S_0$, assuming that all the clusters are visited exactly once. Then, using the dynamic programming, the optimal solutions for the sub-problem with k+1 clusters $S_0, S_1, \ldots S_k$ can be found by choosing the best edge $(u, v_{k,\,j})$, $u \in S_{k-1}$ that links the vertex $v_{k,\,j} \in S_k$ with the cluster S_{k-1}:

$$d^*_{k,\,j} = \min_p \left\{ d^*_{k-1,\,p} + \rho(v_{k-1,\,p}, v_{k,\,j}) \right\}, \quad k = 1,2,\ldots n \tag{19}$$

Therefore, the desired solution for n+1 clusters $S_0, S_1, \ldots S_n$ can be obtained sequentially, starting from $k = 0$ with $d^*_{0,\,j} = 0, \ \forall j$, successively increasing the k-index up to n, computing $d^*_{k,\,j}$, and, finally, choosing the smallest value among $d^*_{n,\,j}, \ j = 1,2,\ldots m_n$. The corresponding sequence of the j-indices $\{ j^*_0, \ j^*_1, \ldots j^*_n \}$, which defines the shortest path, is extracted in reverse order, starting from $k = n$ and $j^*_n = \arg\min_j \{ d^*_{n,\,j} \}$.

An outline of the path planning procedure based on this technique is presented below. The procedure consists of five basic steps where the first two, (1) and (2), implement the recursion (19). These steps deal with creating matrices of optimal distances $M_{dist}(k,j)$ and the corresponding matrix $M_{ind}(k, j)$ of the optimal j-indices of the preceding clusters. In steps (3) and (4), the minimum value in the nth column of the distance matrix $M_{dist}(.)$ is computed, in order to select the best vertex in the

For Peer Review Only

last cluster. Finally, in step (5), the best vertices of the preceding clusters are iteratively extracted from the matrix $M_{ind}(.)$ to generate the optimal path described by the array of the optimal indexes $J_{opt}(.)$. The procedure also uses the array $J_{max}(.)$ containing the upper range for the j-index and the vertex distance function $\rho(.)$ defined previously. The notation d_∞ defines the infinity (or the largest number for computer implementation).

Procedure: Path planning (dynamic programming)

(1) **For** $j = 1$ to $J_{max}(0)$ **do**
  Set $M_{dist}(0, j) := 0$; $M_{ind}(0, j) := 0$
(2) **For** $k = 1$ to n **do**
  For $j = 1$ to $J_{max}(k)$ **do**
    (a) **Set** $d_{min} := d_\infty$
    (b) **For** $p = 1$ to $J_{max}(k\text{-}1)$ **do**
      (α) **Set** $d_{cur} := M_{dist}(k\text{-}1, p) + \rho(k, j, k\text{-}1, p)$
      (β) **If** $k > 0$ & $f_v(k, j, p) \neq 0$
        Set $d_{cur} := d_\infty$
      (γ) **If** $k > 1$ & $f_a(k, j, p, M_{ind}(k\text{-}1, p)) \neq 0$
        Set $d_{cur} := d_\infty$
      (δ) **If** $d_{cur} < d_{min}$ **then**
        Set $d_{min} := d_{cur}$; $j_{opt} := p$
    (c) **Set** $M_{dist}(k, j) := d_{min}$; $M_{ind}(k,j) := j_{opt}$;
(3) **Set** $d_{min} := inf$;
(4) **For** $j := 1$ to $J_{max}(n)$ **do**
  (a) **Set** $d_{cur} := M_{dist}(n, j)$
  (b) **If** $d_{cur} < d_{min}$ **then**
    Set $d_{min} := d_{cur}$; $j_{opt} := j$
(5) **For** $k = 0$ to n **do**
  Set $J_{opt}(n\text{-}k) := j_{opt}$; $j_{opt} := M_{ind}(n\text{-}k, j_{opt})$;

A distinct feature of the developed procedure, which takes into account the problem-specific requirements, is contained in sub-steps (2bβ) and (2bγ) that verify the path-smoothness constraints (6) and (7). Sub-step (2bβ) incorporates the function $f_v(k,j,p)$, which evaluates the velocity constraints (6) for all manipulator axes while the manipulator moves from the location $\boldsymbol{L}_{k\text{-}1,p}$ to the location $\boldsymbol{L}_{k,j}$ (a non-zero value of this function corresponds to violating one of the velocity constraints). Similarly, at the sub-step (2bγ), the function $f_a(k,j,p,l)$ verifies the acceleration constraints (7) for the location sequence $\boldsymbol{L}_{k\text{-}2,l}$, $\boldsymbol{L}_{k\text{-}1,p}$, $\boldsymbol{L}_{k,j}$, where the location $\boldsymbol{L}_{k\text{-}2,l}$ is assumed to be

For Peer Review Only

the optimal predecessor of $\boldsymbol{L}_{k\text{-}1,p}$, i.e. $l = M_{ind}(k\text{-}1,p)$. It is obvious that this is a *penalty-function approach*. The definition of the path-segment of the double length strictly corresponds to the optimality principle incorporated in dynamic programming.

The computational complexity of this algorithm is quite acceptable for practical applications. It is evaluated as $O(n\,m^2)$, where n is the number of clusters and m is the maximum cluster size. This estimation also includes the calculation of the cost components. This calculation is executed for the neighbouring clusters only.

A similar algorithm can be applied for the minimax design objective $J^{(\Delta)}$. The only modification needed deals with the sub-step (2bα), which must be re-written in accordance with the following expression:

$$d^*_{k,j} = \min_p \max\left\{d^*_{k-1,p}\,;\ \rho(v_{k-1,p}, v_{k,j})\right\}, \quad k = 1,2,\ldots n \tag{20}$$

that is also based on the dynamic programming principle. Combining two of the path generation options (objectives $J^{(s)}$ and $J^{(\Delta)}$) and altering the distance metrics weights w_i together with the constraint weights η_v and η_a (see expressions (6) and (7)), the designer can generate a collection of the candidate solutions to be included in the Pareto-optimal set.

4.3. *Generation of control programs*

As assumed in previous sections, the sequence $\{\boldsymbol{n}_k, \boldsymbol{p}_k\}$, describing the tool path in the Cartesian space, is extracted from the CAD workpiece model by uniformly sampling the cutting contour. In performing such a conversion, the designer must specify an appropriate distance ΔS between nodes, which should be small enough to ensure desired accuracy of the contour approximation. However, during off-line programming, there exists a lower bound on the time-sampling step, which is determined by the control unit parameters. Let us examine this problem in detail.

In industrial robots, the control code describes the sequence of the path segments (with a specified interpolation type within each: linear/circular for Cartesian space and linear for joint space). Some advanced robot controllers implement also spline interpolation. It is obvious, that for

For Peer Review Only

tracking the cutting contour, the Cartesian space interpolation is used the most, while the joint space interpolation is utilised only for travelling through a singularity.

To reduce the control program size, see for example (Korb and Troch, 2003), the intermediary nodes within the line/circular segments are not specified. Instead, they are generated on-line using the *trapezoid velocity profiles* that typically include three sections (acceleration, uniform motion, and deceleration). Their duration depends on the desired displacement and the velocity/acceleration constraints imposed on both each joint variable and on the Cartesian coordinates. Moreover, in the continuous motion mode (i.e. without stopping at the end of the segment), the successive segments are *joined* in such way that the acceleration section of the succeeding segment coincides with the deceleration section of the preceding one. As a result, the velocity is maintained at the same level for all three time sections.

However, for *short segments*, the trapezium reduces to a triangle. In addition, the travel time for each section is sampled using the controller "clock time" and is rounded up to a greater value. Therefore, for very short distances, the basis of the triangle (time) is fixed while its height (velocity) is adjusted, to ensure the square be equal to a specified displacement. Thus, the joining of very short segments yields unexpected velocity reduction, which is not admissible for the considered technological equipment.

To eliminate this effect, the distance between the successive nodes must be not less than $\Delta S_{min} = v\tau_{min}$, where v is the cutting speed, and $\tau_{\min}$ is the minimal duration of the acceleration/deceleration sections. In practice, the value of ΔS_{min} is high enough to be considered during robot programming. For example, for typical industrial case studies (cutting speed 10 m/min; clock time 0.005 ms; at least 4 time-samples per acceleration section) the minimal distance between the nodes is equal to 3.3 mm. At the same time, in the model (1) the cutting contour is usually sampled with a step less then 1 mm.

To comply with this constraint on the inter-node distance, it is necessary to aggregate short segments into straight and circular portions, which are suitable for on-line implementations. This

For Peer Review Only

problem is solved using our heuristic algorithm, which is based on the sequential extracting of maximum-length sub-sequences with a fixed initial point that can be approximated by a straight or circular line within given tolerances.

At each run, the algorithm deals with three nodes: $\boldsymbol{p}(k_s)$, $\boldsymbol{p}(k_m)$ and $\boldsymbol{p}(k_e)$, where k_s, k_m and k_e are the indices of the start, middle and end nodes of the current segment, respectively. Within the search loop, the index k_e is successively increased from the initial value k_s+2 and the middle index k_m is recomputed as $k_m = \lfloor (k_s + k_e)/2 \rfloor$, where $\lfloor . \rfloor$ denotes the integer part. Then, each current path segment is verified for its capability to the line and circular approximation:

(1) if the angle between the vectors $\boldsymbol{p}(k_m) - \boldsymbol{p}(k_s)$ and $\boldsymbol{p}(k_e) - \boldsymbol{p}(k_m)$ is less than $\alpha_{\max}$, then the segment admits a *straight-line approximation*, and the index k_e is increased.

(2) if the radius of the circle defined by the points $\boldsymbol{p}(k_s)$, $\boldsymbol{p}(k_m)$ and $\boldsymbol{p}(k_e)$ is less than $R_{\max}$, then the segment admits a *circular approximation*, and the index k_e is increased.

(3) otherwise, the segment enlargement is stopped and the last accepted triple (k_s, k_m, k_e) defines the extended segment of the linear or circular type (commands *LIN* and *CIR*).

This loop is then repeated for the rest of the nodes, assigning new values for the indices $k_s = k_e$; $k_e = k_s + 2$, etc. However, before applying this algorithm, the node sequence is divided into the sections with the same configuration index μ, and the subsequent sections are connected by the arcs using joint-space interpolation (command *PTP*). The length of the connecting segment must be also greater or equal to ΔS_{min} , so this segment is successively expanded in both directions.

For each segment, the motion arguments (tool locations in the Cartesian or joint space) are defined by the node $\boldsymbol{L}(k_e)$ for the *LIN*/*PTP* commands, and by the pair $\{\boldsymbol{L}(k_e), \boldsymbol{L}(k_m)\}$ for the *CIR* command. It should be noted, that at this stage there is no need to modify the joint angles for the 4*th* and 6*th* axes, which admit unlimited rotations (this procedure is executed by the controller on-line).

As a result, for the specified tolerances $\alpha_{\max}$ and $R_{\max}$, the optimum path with the uniform short sampling step ΔS is aggregated into longer sections, which are described by the robot control

For Peer Review Only

commands (*LIN*, *CIR* and *PTP* , with required location arguments) and may be reproduced *on-line* by a robot controller with the desired Cartesian velocity.

5. Efficiency of the developed technique

5.1. *Case-study: robot motion planning for 2D cutting*

To demonstrate the efficiency of the proposed technique, first let us apply it to the planar cutting performed by a Scara robot, which possesses a 1 d.o.f. redundancy with respect to this task. The geometric/kinematic parameters of the manipulator are as follows: the link lengths are $l_1 = l_2 = 1000$ mm; the tool offset (horizontal) is $d = 250$ mm; the maximum speed of the axes involved in the planar motion is 120, 120, and 320 deg/sec, respectively. The cutting contour is a square 800 mm × 800 mm with rounded angles (radius 100 mm). Its centre is located at point (1000 mm, 1000 mm) and is surrounded by and an obstacle described in detail in (Pashkevich et al., 2004). After the sampling, the contour is presented as a set of 60 uniformly distributed nodes (this corresponds to the cutting speed 30 m/min, sampling distance 50 mm, and sampling time 0.1 sec).

The direct kinematic model of this manipulator is described by the equations

$$\begin{aligned} x &= l_1\, cos(q_1) + l_2\, cos(q_1 + q_2) + d\, cos(\varphi) \\ y &= l_1\, sin(q_1) + l_2\, sin(q_1 + q_2) + d\, sin(\varphi)\,, \\ \varphi &= q_1 + q_2 + q_3 \end{aligned} \tag{21}$$

where q_1, q_2, q_3 are the joint coordinates, and x, y, φ are the tool location parameters (two Cartesian coordinates and an orientation angle). It should be noted that one of the joint coordinates (vertical translation of Scara manipulator) is not included in the model, since the z-axis motion is not redundant for the considered planar task.

To derive the inverse kinematic model, let us define $x_0 = x - d\, cos(\varphi)$; $y_0 = y - d\, sin(\varphi)$ and sequentially solve the obtained equations for q_2, q_1 and q_3:

$$q_2 = \mu \cdot acos\,\frac{x_0^2 + y_0^2 - l_1^2 - l_2^2}{2\, l_1\, l_2}; \qquad q_1 = atan2\,\frac{x_0\,(l_1 + l_2\, cos\, q_2) + y_0\, l_2\, sin\, q_2}{y_0\,(l_1 + l_2\, cos\, q_2) - x_0\, l_2\, sin\, q_2} \tag{22}$$

$$q_3 = \varphi - q_1 - q_2 \tag{23}$$

For Peer Review Only

where $\mu = sgn(q_2)$ is the configuration index.

Using the inverse model and altering the tool orientation φ with the step of 10^o, a set of 1385 feasible tool locations $\{\boldsymbol{L}_{k,j}\}$ and the corresponding set of joint coordinates $\{\boldsymbol{Q}_{k,j}\}$ have been generated. To detect collisions, both the manipulator and the obstacle were described by a set of line segments. Then, all pairs from these sets were examined for intersections. The obtained manifold of the admissible motions is presented in figure 3, where the forbidden regions (with detected collisions) are shaded. Within the frames of this geometrical interpretation, the problem is to find the best closed path on the manifold surface, which encloses this figure and avoids the forbidden region. Another interpretation of the problem is given in figure 4, where the admissible and forbidden tool locations are shown in the set of the (γ, t)-planes and the goal is to find the best route from the left to the right vertical lines. It should be noted that, in spite of the apparent simplicity of the last interpretation, the optimisation problem is essentially complicated by the non-Euclidian distances.

[Insert figures 3, 4 about here]

For the considered robotic task, a set of solutions was obtained. These solutions are presented in tables 1 & 2 and differ by both the optimisation criteria and the weights: w_i, η_v and η_a. For each partial criterion, there are given the best values of single-objective optimisations (in brackets). As the tables show, for the relaxed velocity/acceleration constraints, tuning of the weights w_i can barely produce acceptable results. Most of the solutions have a tendency for oscillations in the orientation axis, which is undesirable in practice.

In contrast, the explicit imposing of constraints on the velocity and acceleration (and adjusting corresponding weights η_v , η_a) simplifies obtaining smooth solutions with the balanced values of the partial criteria. This result is illustrated in figures 5 and 6, which contain the trajectories generated for the weights $w_i = (\dot{q}_i^{max}\Delta t)^{-1}$. They also demonstrate rather high oscillation of q_3 for the unconstrained case (see figure 5; joint velocity plots) and very smooth trajectories for the

For Peer Review Only

constrained optimisation (similar plots in figure 6). Hence, the proposed approach gives the possibility to eliminate oscillations in the orientation axes.

[Insert Tables 1, 2 about here]

[Insert figures 5, 6 about here]

The final comparison of the obtained results can be done using a multi-axes plot which presents two typical solutions with the values of nine partial design objectives J_{1v}, J_{2v} … $J_{3\Delta}$ (see figure 7). The first solution (a) corresponds to the best result of a known technique, which corresponds to the very tedious adjusting of the weights w_i. The second solution (b) was generated straightforwardly, using the developed method with the standard values $w_i = (\dot{q}_i^{max}\Delta t)^{-1}$ and with a very fast and simple tuning of the algorithm parameters η_v and η_a. As follows from the comparison, the developed technique reduces essentially oscillations of the orientation axis (see axes J_{3s}, J_{3v} and $J_{3\Delta}$ in figure 7, where relevant performance measures are lower by the factors 2…3), while maintaining almost the same values for the rest of the objectives.

[Insert figure 7 about here]

5.2. *Comparison with other techniques*

The main advantage of the proposed technique compared to the previous work (Pashkevich et al., 2004) is the ability to generate a smooth path without the tedious adjusting of weights w_i. This useful feature is achieved by modifying the standard dynamic programming routine: including an implicit verification of the velocity/acceleration constraints while creating the backtracking matrix. From this perspective, the new technique is robust with respect to weights assigned by the designer. This advantage has been confirmed by a number of industrial projects where the developed technique has been successfully applied.

Another advantage, which was demonstrated by several 3D cutting case-studies, relates to the algorithm's ability to generate a robot path with large rotations of the manipulator orientation axes (more than 360°). This allows the exploitation of recent advances in the kinematics of cutting

For Peer Review Only

robots, such as the endless rotation of wrist joints. Besides, because the specific generation of the search space (see figure 2) and relevant distance computing for the angles, the feasible solutions may successfully pass through singularities. This allows the changing of the configurations indices while in motion.

The developed technique has also advantages compared to the smooth path generation method recently proposed by Gasparetto and Zanotto (2007). In their paper, the objective function contains a term proportional to the integral of the squared jerk (defined as the derivative of the acceleration) along the trajectory, while the second term is proportional to the total execution time. However, this formulation does not take into account the particularities of the considered application problem (where the execution time is fixed) and may produce local oscillations in the neighbourhood of the singularities caused by the integral-type performance measure used.

Hence, the developed technique yields a significant reduction of the robot programming time while respecting simultaneously both the smoothness objectives and the actuators velocity/acceleration constraints. In the future, this approach will be incorporated in more general design algorithms: (i) optimal path generation for several cutting contours, and also (ii) automatic robot placement in the cutting manufacturing cell. Another prospective application is the redundancy resolution for multi-manipulator robotic systems with path/velocity/acceleration constraints defined in Cartesian space. This corresponds to current industrial needs, since some recent designs for cutting cells include both an industrial robot (cutting tool manipulator) and a positioning table (workpiece manipulator).

6. Industrial implementation

The developed algorithms have been successfully implemented on the manufacturing floor, in ROBOMAX CAD package, which is a powerful system for robotic computer-aided design and off-line programming. It is already used in automotive industry and has been successfully applied for the design of a number of manufacturing lines. The package is completely integrated with

For Peer Review Only

Autodesk™ products and includes some auxiliary tools, which enable the user to design a robotic cell and generate a control program taking into account the particularities of the employed technology.

With respect to laser cutting, the Robomax/Laser subsystems enable to design a workcell layout and optimise robot motion using multi-objective optimisation techniques. At the beginning, using standard routines of the Autodesk Mechanical Desktop (AMD), a mathematical description of the cutting contour is presented in the form of the "*augmented line*". The designer may define either the desired distance between vertices or their total number. In addition, he/she can estimate the minimum number of vertices required keeping the accuracy within tolerances. Relevant software tools allow also to aggregate separate segments in a common cutting contour and perform the unification of sampling distances.

The main design procedure consists of three iteratively repeated steps. The first is the selection of a proper manufacturing environment and location of the manufacturing task within the robot workspace. Here, the designer can create and optimise the layout of the cell interactively, using functions incorporated in the system. The required components (robotic manipulators, workpiece positioners, clamping devices, protective fences, columns, etc.) may be selected from the library. To optimise the position of the workpiece, a set of performance measures is used that considers such important features as the distance to the obstacles, closeness to joint limits, required range of joint motions.

The second step deals with path planning using the algorithms described in the previous sections. The goal is to utilize the kinematics redundancy of the robotic manipulator with respect to the cutting operations. To balance competing objectives, the designer can interactively assign priorities or weights, as well as activate/ deactivate constraints and criteria. In particular, it is possible to choose between the displacement/increment minimisation, define the distance metrics, and to change weights after visual analysis of the obtained path. In addition, there is an interactive

For Peer Review Only

editor allowing the adjustment of some nodes in accordance with other preferences, which are not presented in the formal criteria used in the model.

In the third step, the obtained solution is verified using realistic 3D simulations of the manipulator motions in the manufacturing environment. The ROBOMAX interactive debugger enables the user to evaluate both the total path and its separate segments, corresponding to elementary motions between the nodes. Also, it is possible to inspect joint, velocity and acceleration profiles for all manipulator axes. If the designer is not satisfied with the current solution, he/she can return to the first or the second step to change the design parameters to get a new solution.

After finishing this optimisation cycle, the generated path is converted into a control code in the robot programming language. At this stage, it is also possible to optimise the program, in order to eliminate some nodes (that belong to straight-line or circular segments), or to add additional commands required for control of technological parameters. At the end, the created program is uploaded to the robotic cell controller.

A recent application of the ROBOMAX/Laser is the off-line programming of a robotic cutting station for a minivan production based on KUKA robots and relevant positioning and clamping devices. The station is used for small-batch manufacturing, which requires frequent reprogramming. As follows from our industrial experience, the developed technique provides adequate capabilities for generating robot control codes and crucially reduces the programming and changeover time.

7. Conclusion

The optimal path generation for the redundant manipulators is an important issue for off-line programming of the laser cutting robotic systems. Recent advances in laser technology, and especially essential increase of the cutting speeds, motivate amending the existing methods. At present, these do not completely utilise the actuator capabilities as well as neglect some particularities in the mechanical design of the manipulator wrist.

This work contributes to a multi-objective optimisation of the manipulator motions for the continuous contour tracking with an axis-symmetric technological tool. In contrast to previous

For Peer Review Only

work, this optimisation technique explicitly incorporates the verification of the velocity/acceleration constrains, enabling the designer to interactively define their importance with respect to the path-smoothness objectives. In addition, the proposed approach takes into account the capacity of some wrist axes for unlimited rotation in order to produce more efficient motions.

The developed path planning algorithm is based on dynamic programming with some extensions, which incorporate the constraint checking for each segment of a candidate solution and the penalty assignment if some constraints are violated. To generate a smooth motion, each joint trajectory is evaluated by a set of performance indices such as the coordinate deviation, maximum increment, and total displacement. The search space is converted into a directed acyclic graph and the problem is re-formulated in terms of combinatorial optimisation theory. To evaluate the distance between the successive tool locations, three types of metrics are used, each of which has the weight assigning option for the corresponding joint axis. During the optimisation, the vector objective is converted into a scalar one using the sum or minimax approaches. Then the weights are altered to generate a set of Pareto-optimal solutions. The proposed algorithm has been implemented in an industrial software package and verified on the manufacturing floor in the automotive industry. Its application for off-line programming of robotic laser cutting stations yielded significant reduction of the process planning and changeover time.

Future research will focus on the development of technology-oriented algorithms for optimal robot path planning for several cutting contours. This problem arises from a number of industrial applications where the workpiece contains numerous contours to be cut by a single robot or by several robots. The corresponding path-planning technique requires optimal scheduling of the cutting contours, load balancing among robots, collision avoidance during simultaneous operation, etc. Another prospective research direction is in the automation of the robot placement for cutting manufacturing cells, which requires optimisation of the robot location with respect to the workpiece and also the integration of this technique with the capabilities of the relevant robotic CAD systems.

For Peer Review Only

Acknowledgements. The second author acknowledges the Ecole des Mines de Saint Etienne where he was visiting professor. The authors thank also Chris Yukna for his help in English.

References

ABE, T., SHIBATA, T. and TANIE, K., 1994. Motion planning for 3D cutting by a manipulator with 6 degrees of freedom - optimization by genetic algorithm, *Proceedings of 3rd International Conference on Fuzzy logic Neural Nets and Soft Computing*, 453-454.

ANOTAIPAIBOON, W. and MAKHANOV, S.S., 2005. Tool path generation for five-axis NC machining using adaptive space-filling curves. *International Journal of Production Research,* **43**(8), 1643-1665.

ANTONELLI, G., CHIAVERINI, S., GERIO, G.P., PALLADINO, M. and RENGA, G., 2007. SmartMove4: An industrial implementation of trajectory planning for robots. *Industrial Robot*, **34**(3), pp. 217-224.

BACKES, F., GEIGER, M. and FRANKE, V., 1996. Technology oriented off-line programming for 3D laser material processing. *Technical Paper - Society of Manufacturing Engineers, No SME MS96-145*, 241 – 246.

BAGCHI, T.P., GUPTA, J.N.D. and SRISKANDARAJAH, C., 2006. A review of TSP based approaches for flowshop scheduling. *European Journal of Operational Research*, **169**(3), pp. 816-854.

BAUER, L., 1996. Offline-programming for 3D-laser systems. *Proceedings of SPIE,* Vol 2787, pp.34 – 42.

BEN-ARIEH, D., GUTIN, G., PENN, M., YEO, A. and ZVEROVITCH, A., 2003. Process planning for rotational parts using the generalized travelling salesman problem. *International Journal of Production Research*, **41**(11), 2581-2596.

BI, Z.M. and LANG, S.Y.T., 2007. Automated robotic programming for products with changes. *International Journal of Production Research*, **45**(9), pp. 2105-2118.

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.

BOHEZ, E., MAKHANOV, S.S. and SONTHIPERMPOON, K., 2000. Adaptive nonlinear tool path optimization for five-axis machining. *International Journal of Production Research,* **38**(17 SPEC.), 4329-4343.

CECCARELLI, M., 2004. *Fundamentals of mechanics of robotic manipulation*.(Boston: Kluwer Academic Publishers), 310 pp.

CHEDMAIL P., DOMBRE E. and WENGER P., 1998. *La CAO en Robotique : outils et methodologies*. (Paris : Hermes), 383 pp.

CHEN, H., XI, N. and CHEN, Y., 2003. Multi-objective Optimal Robot Path Planning in Manufacturing. *IEEE International Conference on Intelligent Robots and Systems.* Vol 2, 1167-1172.

For Peer Review Only

CHENG, F.-T. and SHIH, M.-S., 1997. Multiple-goal priority considerations of redundant manipulators. *Robotica,* **15**(6), 675-691.

CHETTIBI, T., 2006. Synthesis of dynamic motions for robotic manipulators with geometric path constraints. *Mechatronics,* **16**(9), 547-563.

DEFAUX, M., 2004. Welding and cutting: Robots take the lead in 3D. *Industrial Robot,* **31**(1), 39-43.

DEZA, E., 2006. Dictionary of distances (Boston, Elsevier), 391 pp.

DOLGUI, A. and PASHKEVICH, A., 2006. Cluster-level operations planning for the out-of-position robotic arc-welding. *International Journal of Production Research,* **44**(4), 675-702.

DOTY, K.L., MELCHIORRI, C. and BONIVENTO, C., 1993. Theory of generalized inverses applied to robotics. *International Journal of Robotics Research,* **12**(1), 1-19.

DUARTE, F.B.M. and MACHADO, J.A.T., 2000. Motion chaos in the pseudoinverse control of redundant robots. *International Workshop on Advanced Motion Control* (*AMC*), 624-629.

EDAN, Y. and NOF, S.Y., 1996. Graphic-based analysis of robot motion economy principles. *Robotics and Computer-Integrated Manufacturing*, **12**(2), pp. 185-193.

EHRGOTT, M., 2005. *Multicriteria Optimization* (Berlin: Springer), 323 pp.

FUJIMURA, K., 1996. Path planning with multiple objectives. *IEEE Robotics and Automation Magazine*, **3**(1), 33-38.

GASPARETTO, A. and ZANOTTO, V., 2007. A new method for smooth trajectory planning of robot manipulators. *Mechanism and Machine Theory*, **42**(4), pp. 455-471.

GEIGER, M. and GROPP, A., 1992. Laser beam cutting with solid state lasers and industrial robots. *Proc. of the 4th European Conference on Laser Treatment of Materials* (ECLAT), Gottingen, 87-92.

GEIGER, M. and KOLLERA, H., 1994. Tool path optimization for 2D laser cutting. *Production Engineering*: *Annals of German Academic Society for Production Engineering*, **1**(2), 155-158.

GHANY, K.A., RAFEA, H.A. and NEWISHY, M., 2006. Using a Nd:YAG laser and six axes robot to cut zinc-coated steel. *International Journal of Advanced Manufacturing Technology,* **28**(11-12), 1111-1117.

GROPP, A., HUTFLESS, J., SCHUBERTH, S. and GEIGER, M., 1995. Laser beam cutting. *Optical and Quantum Electronics,* **27**(12), 1257-1271.

HUSTY, M.L., PFURNER, M. and SCHROCKER, H.-P., 2007. A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator. *Mechanism and Machine Theory*, 42(1), 66-81.

HWANG, Y.K., CHEN, P.C., MACIEJEWSKI, A.A. and NEIDIGK, D.D., 1994. Global motion planner for curve-tracing robots. *IEEE International Conference on Robotics and Automation*, 662-667

ION, J.C., 2005. *Laser processing of engineering materials: principles, procedure and industrial applications*. (Oxford: Elsevier), 556 pp.

For Peer Review Only

JOUANEH, M.K., DORNFELD, D.A. and TOMIZUKA, M., 1990. Trajectory planning for coordinated motion of a robot and a positioning table -- II: Optimal trajectory specification. *IEEE Transactions on Robotics and Automation,* **6**(6), 746-759.

KAIERLE, S., FUERST, B., KITTEL, J., KREUTZ, E.W. and POPRAWE, R., 1999. Design and manufacturing tools for laser beam processing. *Proceedings of SPIE,* Vol. 3833, 148-159

KIM, H.J., KIM, Y.D. and LEE, D.H., 2005. Scheduling for an arc-welding robot considering heat-caused distortion. *Journal of the Operational Research Society*, **56**(1), 39–50.

KIM, Y.S., WANG, E., HWANG, I.-K. and RHO, H.M., 2003. Integrated machining tool path planning using feature free spaces. *International Journal of Production Research,* **41**(14), 3237-3255.

KLEIN, C. A., HUANG, C.-H., 1983. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-13** (2), 245-250.

KORB, W. and TROCH, I., 2003. Data reduction for manipulator path planning. *Robotica,* **21**(6), 605-614.

LEE, J., 1995. Dynamic programming approach to near minimum-time trajectory planning for two robots. *IEEE Transactions on Robotics and Automation,* **11**(1), 160-164.

MANOCHA, D. and CANNY, J.F., 1994. Efficient inverse kinematics for general 6R manipulators. *IEEE Transactions on Robotics and Automation*, 10(5), 648-657.

MANSEUR, R. and DOTY, K.L., 1989. Robot manipulator with 16 real inverse kinematic solution sets. *International Journal of Robotics Research,* **8**(5), 75-79.

MITSI, S., BOUZAKIS, K.-D., MANSOUR, G., SAGRIS, D. and MALIARIS, G., 2005. Off-line programming of an industrial robot for manufacturing. *International Journal of Advanced Manufacturing Technology,* **26**(3), 262-267.

NOF, S.Y. (1999). *Handbook of Industrial Robotics*. (New York: John Wiley & Sons), 1348 pp.

PASHKEVICH, A., 1997. Real-time inverse kinematics for robots with offset and reduced wrist. *Control Engineering Practice,* **5**(10), 1443-1450.

PASHKEVICH, A.P., DOLGUI, A.B. and CHUMAKOV, O.A., 2004. Multiobjective optimization of robot motion for laser cutting applications. *International Journal of Computer Integrated Manufacturing,* **17**(2), 171-183.

PFEIFFER, F. and JOHANNI, R., 1987. Concept for manipulator trajectory planning. *IEEE Journal of Robotics and Automation*, **RA-3**(2), 115-123.

ROOKS, B., 2004. Laser processing of plastics. *Industrial Robot,* **31**(4), 338-342.

RUBINOVITZ, J. and WYSK, R.A., 1988. Task level off-line programming system for robotic arc welding—an overview. *Journal of Manufacturing Systems*, **7**(4), 293–299.

SCHLUETER, H., 2005. Advances in industrial high power lasers. *Proceedings of SPIE,* Vol. 5777 (PART I), 8-15.

SENDLER. U., 1994. Offline-programmiertes Laserschneiden im Automobilbau, *VDI-Z,* Vol. 136, No. 1/2, 28-30.

For Peer Review Only

SHIBATA, T., ABE, T, TANIE, K. and NOSE, M., 1997. Motion planning by genetic algorithm for a redundant manipulator using a model of criteria of skilled operators. *Information Sciences*, **102**(1-4), 171-186.

SHILLER, Z. and DUBOWSKY, S., 1991. On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, **7**(6), 785-797.

SHIN, K.G. and MCKAY, N.D., 1986. Dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, **AC-31**(6), 491-500.

SHOVAL, S., RUBINOVITZ, J. and NOF, S., 1998. Analysis of robot motion performance and implications to economy principles. *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 3, 2765-2770

SPONG, M.W., HUTCHINSON, S., VIDYASAGAR, M., 2006. *Robot modeling and control*. (Hoboken: John Wiley & Sons), 478 pp.

STEEN, W. M., 2003. *Laser material processing*. (London: Springer), 408 pp.

STEUER, R. E., 1986. *Multiple Criteria Optimization: Theory, Computation and Application*. (New York: John Wiley & Sons), 546 pp.

SUN, R.H., TSAI, Y.C., 1994. A modified analytical model for optimization of NC-tool cutting path. *International Journal of Production Research*, **32** (10), 2335-2344.

WANG, G.G. and XIE, S.Q., 2005. Optimal process planning for a combined punch-and-laser cutting machine using ant colony optimization. *International Journal of Production Research*, **43**(11), 2195-2216.

WHITNEY, D.E., 1969. Resolved motion rate control of manipulators and human prostheses. *IEEE Transaction on Man Machine System*, **MMS-10**(2), pp. 47-53.

WITTENBERG, G., 1995. Developments in offline programming: An overview. *Industrial Robot,* **22**(3), pp. 21-23.

XU, X.J., BRADLEY, C., ZHANG, Y.F., LOH, H.T. and WONG, Y.S., 2002. Tool-path generation for five-axis machining of free-form surfaces based on accessibility analysis. *International Journal of Production Research,* **40**(14), 3253-3274.

YAO, Z. and GUPTA, K., 2007. Path planning with general end-effector constraints. *Robotics and Autonomous Systems*, **55**(4), pp. 316-327.

YOSHIKAWA, T., 1996. Basic optimization methods of redundant manipulators. *Laboratory Robotics and Automation,* **8**(1), 49-60.

YUAN, X. and GU, Y., 1999. An integration of robot programming and sequence planning, in Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 1, pp. 102–107.

For Peer Review Only

Tables

Table 1. Performance measures for the design problem $J^{(s)}(\boldsymbol{\gamma}, \boldsymbol{\mu}) \rightarrow min$ (C-metrics)

Weights	$J_s^{(1)}$ (27.7)	$J_s^{(2)}$ (62.92)	$J_s^{(3)}$ (105.0)	$J_v^{(1)}$ (1.47)	$J_v^{(2)}$ (2.54)	$J_v^{(3)}$ (3.84)	$J_\Delta^{(1)}$ (19.09)	$J_\Delta^{(2)}$ (41.32)	$J_\Delta^{(3)}$ (11.50)
Without velocity/acceleration constraints									
(1, 0, 0)	27.7	222.6	3080.3	2.76	19.43	355.3	19.09	66.09	357.26
(0, 1, 0)	281.5	62.9	1901.8	24.31	3.57	202.5	62.49	49.76	335.59
(0, 0, 1)	107.1	157.2	105.0	3.82	3.86	3.84	52.06	78.58	52.26
$w_i = (\dot{q}_i^{max}\Delta t)^{-1}$	104.0	146.2	158.5	3.66	3.50	11.06	50.92	73.83	48.73
With velocity/acceleration constraints ($\eta_v = 0.5$ & $\eta_a = 0.25$)									
(1, 0, 0)	106.7	116.0	597.6	7.76	6.82	23.64	55.63	55.07	329.7
(0, 1, 0)	91.98	78.16	511.0	5.76	4.83	13.64	54.58	50.47	321.5
(0, 0, 1)	107.1	157.2	105.0	3.82	3.86	3.84	52.06	78.58	52.26
$w_i = (\dot{q}_i^{max}\Delta t)^{-1}$	106.6	153.6	105.7	3.96	3.70	4.00	51.31	76.80	52.72

Table 2. Performance measures for the design problem $J^{(\Delta)}(\boldsymbol{\gamma}, \boldsymbol{\mu}) \rightarrow min$ (C-metrics)

Weights	$J_s^{(1)}$ (27.7)	$J_s^{(2)}$ (62.92)	$J_s^{(3)}$ (105.0)	$J_v^{(1)}$ (1.47)	$J_v^{(2)}$ (2.54)	$J_v^{(3)}$ (3.84)	$J_\Delta^{(1)}$ (19.09)	$J_\Delta^{(2)}$ (41.32)	$J_\Delta^{(3)}$ (11.50)
Without velocity/acceleration constraints									
(1,0,0)	42.8	305.4	2719.1	1.47	29.87	334.2	32.94	58.39	334.2
(0,1,0)	132.1	69.2	1252.0	11.63	2.54	349.7	45.95	50.72	349.69
(0,0,1)	107.06	157.17	105.04	3.82	3.86	3.84	52.06	78.58	52.26
$w_i = (\dot{q}_i^{max}\Delta t)^{-1}$	103.13	145.82	135.80	3.69	3.62	9.71	50.59	74.47	41.59
With velocity/acceleration constraints ($\eta_v = 0.5$ & $\eta_a = 0.25$)									
(1,0,0)	76.84	95.03	441.56	3.33	4.06	12.99	47.26	51.81	309.25
(0,1,0)	69.84	105.88	437.28	2.97	4.06	12.51	42.34	53.27	319.37
(0,0,1)	107.1	157.2	105.0	3.82	3.86	3.84	52.06	78.58	52.26
$w_i = (\dot{q}_i^{max}\Delta t)^{-1}$	107.1	157.2	105.0	3.82	3.86	3.84	52.06	78.58	52.26

For Peer Review Only

Figures

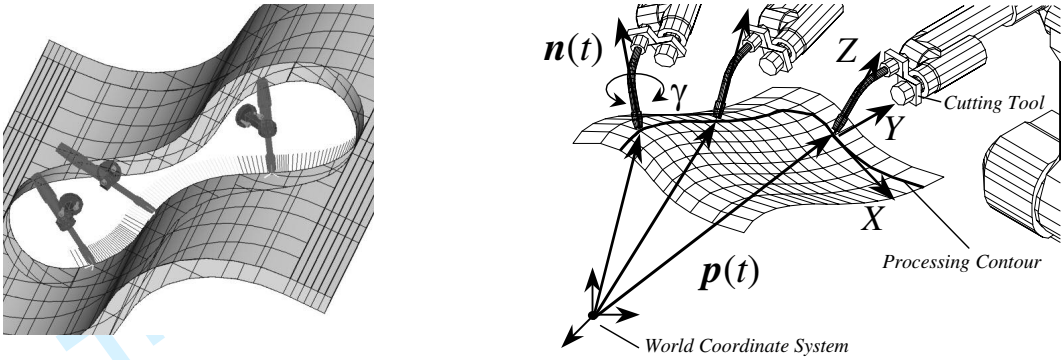

Figure 1. Definition of the task frames

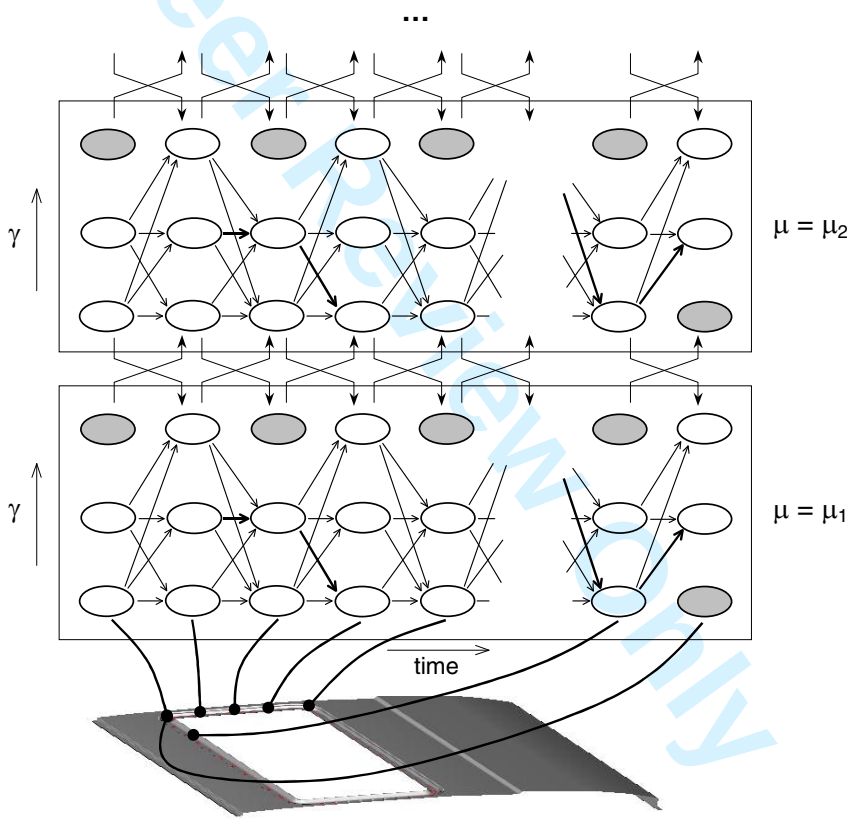

Figure 2. Descretisation of the search space

("O" – admissible locations, "●" – inadmissible locations)

For Peer Review Only

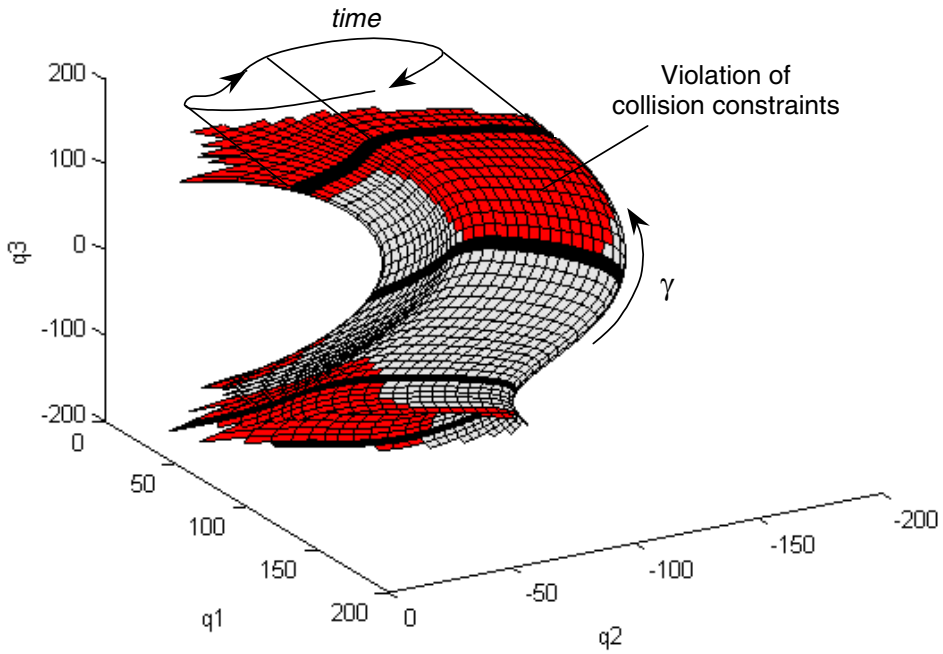

Figure 3. Joint-space manifold of the admissible manipulator motions

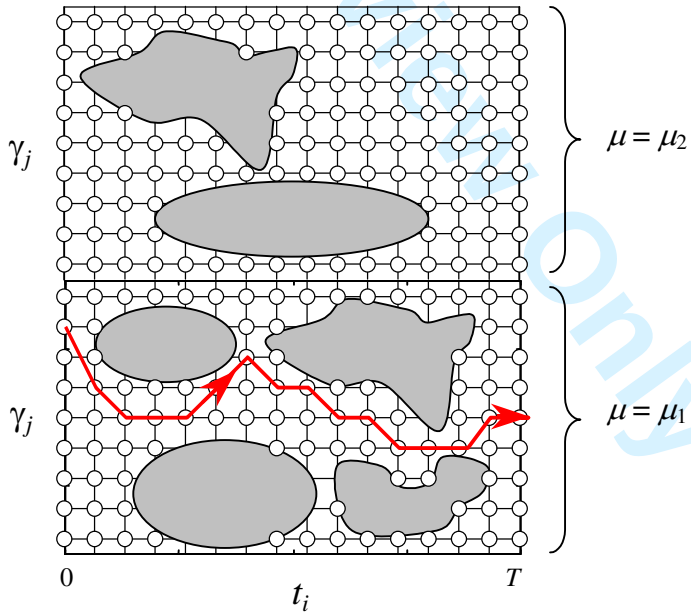

Figure 4. Admissible manipulator motions in the (γ, t)-plane

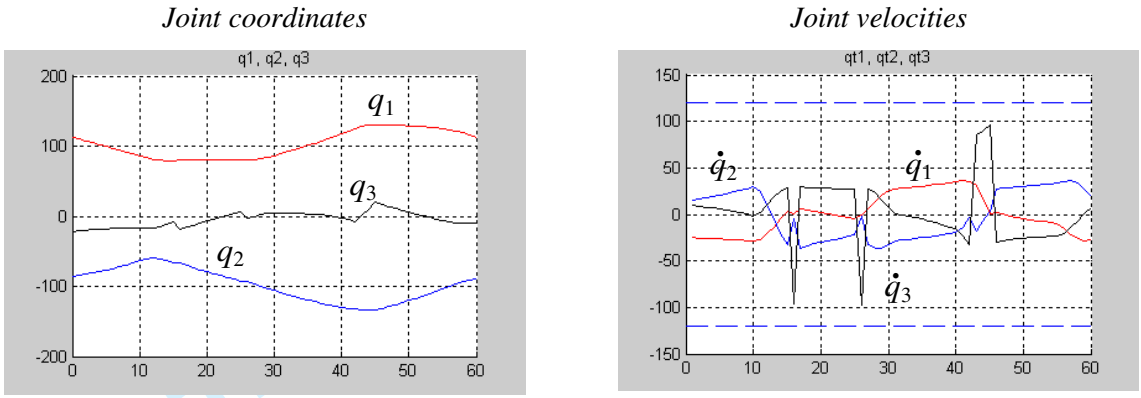

Figure 5. Optimisation without the velocity/acceleration constraints

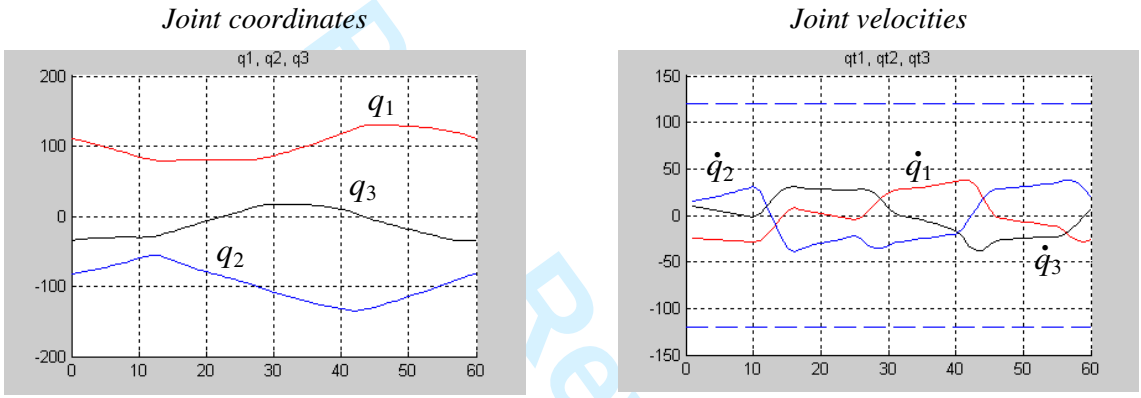

Figure 6. Optimisation with the velocity/acceleration constraints

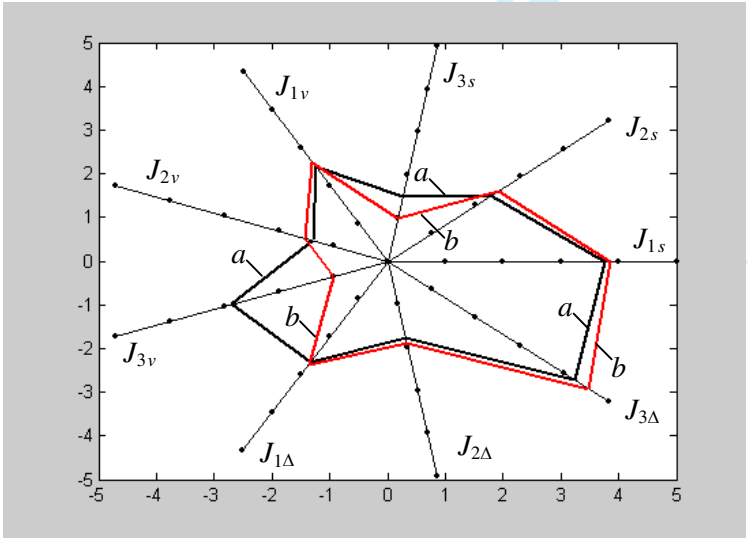

Figure 7. Pareto-optimal solutions:
(a) – without constraints; (b) – with constraints.