# Multiobjective optimization of robot motion for laser cutting applications

Anatol Pashkevich, Alexandre Dolgui, Oleg Chumakov

# Multiobjective optimization of robot motion for laser cutting applications

ANATOLY P. PASHKEVICH, ALEXANDRE B. DOLGUI and OLEG A. CHUMAKOV

**Abstract.** This paper focuses on the enhancement of automatic robot programming techniques for laser cutting applications. Such technology has already gained essential industrial acceptance, but its application for small lot production is limited by the tedious and time-consuming process of robot programming. Currently, even sophisticated graphical simulation systems do not allow optimization of robot motion using multiple criteria, nor does it take into account redundancy caused by the tool axial symmetry. The particular contribution of this paper lies in the area of multiobjective optimization of robot motions via graph representation of the search space and dynamic programming procedures. It presents algorithms that allow generation of smooth manipulator trajectories within acceptable time, simultaneously considering kinematics, collision and singularities constraints of the robotic system, as well as the limitations of the robot control units. The efficiency of the 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.

## 1. Introduction

In the last few decades, laser machining has gained essential industrial acceptance as an alternative to mechanical processing and is widely used in various fields of manufacturing. It has significant advantages over traditional production methods due to its high process quality combined with high speed, high precision and potential flexibility (Geiger and Otto 2000). In the automotive industry, for instance, the panels of the small series models are made in two steps. First, sheet metal parts are deep drawn, and then they are cut into their final shape. Using a laser, this can be done fast and precisely, since the required contour processing depends on the program of the robotic system only.

However, manual teaching of robotic laser system is very tedious and time-consuming. It requires temporary exclusion of the robot from the manufacturing process and specific preparation of processing components, which have to be properly marked. For complex processing contours, the time required for marking is substantially larger than the time for the robot teaching itself (Bauer 1996). In some cases, an expensive coordinate measurement machine has to be used for marking.

In contrast to manual teaching, off-line programming generates the control code by means of computer graphics, based on a virtual scene, and away from the factory floor. As the result, the down time of a robot may be reduced by 80–85%, enabling very small batch sizes to become economically feasible (Sendler 1994). This approach also allows interactive program debugging through line-by-line visualization of what is happening on the screen.

At the moment, there are a number of offline robot programming systems and robotic simulation packages on the market. Some of the most common are RobCAD (Tecnomatix Technologies), IGRIP (Deneb Robotics), CimStation (Silma) and Workspace (Robot Simulations). These implement a number of good graphical and path-planning methods; however, there still exists a considerable gap between their capabilities and the requirements of a particular technology. Currently, the robot programs for some cutting applications are constructed interactively. The ultimate goal is the automatic generation of reliable programs from designs and drawings, similar to CNC-machine programming methods. For this reason, this is still an area of active development.

For laser cutting applications, both 2D and 3D offline programming systems are also available on the market. The main contribution in this area has been

*Authors:* A. P. Pashkevich and A. B. Dolgui, Systems Optimisation Laboratory, University of Technology of Troyes, 12, rue Marie Curie B.P. 2060, Troyes, France. E-mail: dolgui@utt.fr. O. A. Chumakov, Robotic Laboratory, Belarusian State University of Informatics and Radioelectronics, 6 P. Brovka St., Minsk, 220600, Belarus.

from Geiger and his co-workers (University of Erlangen-Nuremberg, Germany). They have developed technology oriented techniques, which simultaneously consider the part geometry, process parameters and some properties of the machine tool (Geiger and Kolléra 1994, Bauer and Backes 1995, Backes *et al.* 1996, Otto *et al.* 1997). However, the existing techniques may be applied only to non-redundant kinematic structures, which are based on gantries with five-axis robots.

This paper focuses on enhancement of the 3D offline programming techniques for six-axis robots that possess inherited redundancy with respect to the cutting. In contrast to the known methods, the proposed approach takes into account this redundancy in combination with kinematic, collision and singularities constraints of the robotic system, as well as the limitations of industrial control units. It relies on simultaneous optimization of multiple criteria for all joint coordinates and allows generation of smooth manipulator trajectories within acceptable time for industrial applications.

The remainder of this paper is organized as follows. Section 2 is devoted to a formal statement of the considered problem and describes performance measures, which compose the vector criteria. In section 3, the search space is converted into a directed graph and the problem is reformulated in terms of combinatorial optimization theory. Section 4 includes the main results and presents optimization algorithms for both separates performance measures and their combinations. Section 5 contains simulation results and their analysis. In section 6, industrial implementation is presented and, finally, section 7 summarizes the main contributions of this paper.

## 2. Problem statement

In the general case, the offline programming system should convert workpiece description information into a robot control code program taking into account both the manipulator capability and technological constraints (heat-affected zone, feature straightness, kerf width, etc). However, the core for the program generation tool is a set of optimization routines that focus on robot motion planning and control.

### 2.1. *General optimization problem*

Let us assume that input data for the motion planning system are presented by two vector functions

$$\{\boldsymbol{p}(t), \boldsymbol{n}(t): \quad |\boldsymbol{n}(t)| = 1;\ t \in [0; T]\} \tag{1}$$

where $t$ is a scalar argument (time); $\boldsymbol{p}(t) \in \boldsymbol{R}^3$ defines the $x,y,z$-coordinates of the tool tip, and $\boldsymbol{n}(t) \in \boldsymbol{R}^3$ is the unit vector of the tool axis orientation, which must be normal to the processing surface (figure 1). These data can be directly extracted from the graphical model of the workpiece by, for example, defining the processing contour as an 'augmented line'.

To describe spatial location of the robotic tool, let us introduce another unit vector

$$\boldsymbol{a}(t) = \dot{\boldsymbol{p}}(t)/|\dot{\boldsymbol{p}}(t)|, \tag{2}$$

which is tangential to the workpiece surface and points to the tool motion direction (here 'dot' denotes the time derivative). Assuming that the vectors $\boldsymbol{a}(t)$ and $\boldsymbol{n}(t)$ are mutually orthogonal, i.e.

$$\dot{\mathbf{p}}(t)^T \mathbf{a}(t) = 0, \tag{3}$$

at each point of the processing, a contour may be associated with the coordinate frame in which the $x$-axis is directed along the path, the $z$-axis is directed along the cutting tool, and the $y$-axis completes them to obtained the right-hand oriented triple (see figure 1). The corresponding matrix of homogenous transformation is defined as:

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

where ' × ' denotes the vector product.

The introduced sliding frame $\boldsymbol{H}(t)$ can be used as a pivot for defining the complete pose (or spatial location) of the robotic tool, which requires six independent parameters (three Cartesian coordinates and three Euler angles) to be described in a manipulator control unit. However, for cutting technology,

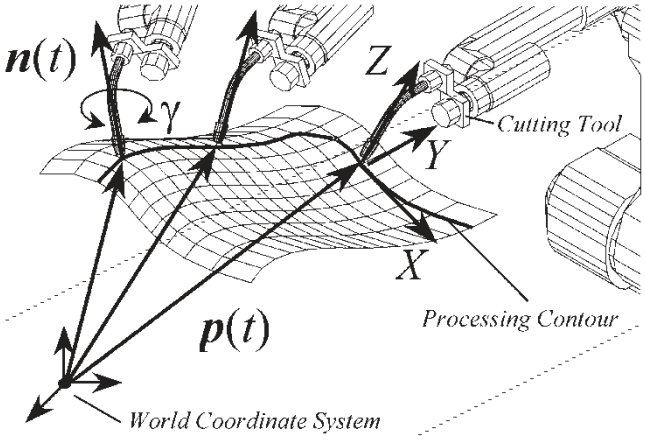


Figure 1. Defining task frames.

five parameters are sufficient because the tool is axially symmetric. Hence, the cutting tool locations $\boldsymbol{L}$ can be defined accurate to the rotation around vector $\boldsymbol{n} = [n_x\ n_y\ n_z]^{\mathrm{T}}$

$$\boldsymbol{L}(t,\gamma) = \begin{bmatrix} \boldsymbol{R}_n\ (\gamma) & {}_{3\times 3} & 0 \\ 0 \quad 0 & 0 & 1 \end{bmatrix} \cdot \boldsymbol{H}(t); \quad t \in [0,\ T]; \gamma \in (-\pi, \pi], \tag{5}$$

where $\gamma \in (-\pi,\pi]$ is the rotation angle and $\boldsymbol{R}_n(\gamma)$ is the corresponding $3 \times 3$ orthogonal rotation matrix:

$$\boldsymbol{R}_n(\gamma) = \begin{bmatrix} n_x^2 V_\gamma + C_\gamma & n_x n_y V_y - n_z S_\gamma & n_x n_z V_\gamma + n_y S_\gamma \\ n_x n_y V_\gamma + n_z S_\gamma & n_y^2 V_\gamma + C_\gamma & n_y n_z V_\gamma - n_x S_\gamma \\ n_x n_z V_\gamma - n_y S_\gamma & n_y n_z V_\gamma + n_x S_\gamma & n_z^2 V_\gamma + C_\gamma \end{bmatrix}$$

and

$$C_\gamma = \cos(\gamma); \quad S_\gamma = \sin(\gamma); \quad V_\gamma = 1 - \cos(\gamma).$$

Therefore, the robotic task description (5) includes one undetermined parameter $\gamma$ (i.e. one redundant degree of freedom), which can be used for optimization purposes. Indeed, the technological tool can be rotated around the laser (or plasma) beam axis without any influence on the quality of processing, provided that this motion does not contradict to robot kinematic and collision constraints. The latter are defined by binary functions $\Psi_k(\boldsymbol{L})$ and $\Psi_c(\boldsymbol{L})$ whose non-zero values correspond to the constraint violation. (These functions are standard routines of industrial robotic CAD packages; they take into account manipulator geometry and link lengths, joint coordinate limits, joint speed limits, workspace geometry, etc.) In addition, to ensure the singularity-free motion of the manipulator, let us define another binary function $\Psi_s(\boldsymbol{L})$ whose zero value defines an admissible distance to singularities. The latter can be expressed as the lower bound of the manipulator manipulability, for example (Yoshikawa 1985). So, the considered problem of robot motion planning can be stated as follows.

**Original design problem**. For a given manipulator task described by parametrized homogeneous matrix-function $\boldsymbol{L}(t,\gamma)$, $t \in [0;T]$, find a scalar function $\gamma(t) \in (-\pi;\pi]$ which defines the continuous sequence of feasible tool locations $\boldsymbol{L}(t,\gamma(t))$ and minimizes (or maximizes) given performance measure

$$J\{\boldsymbol{L}(t,\gamma(t)); \quad t \in [0;\ T]\} \to \min_{\gamma(t)} \tag{6}$$

subject to kinematic, collision and singularities constraints

$$\Psi_k[L(t,\gamma(t))] = 0;\ \Psi_c[L(t,\gamma(t))] = 0;\ \Psi_s[L(t,\gamma(t))] = 0. \tag{7}$$

Geometrical interpretation of this problem may be presented as searching for the best path on the plane that avoids prohibited regions indicating constraint violations. It should be noted that, in spite of the apparent similarity with mobile robot path planning (Latombe 1991), the considered problem essentially differs by objective functions, which are considered in detail in the next section. Besides, in contrast to the mobile robotics, for this problem there are no explicit initial and target points that must be connected by a feasible path (they are defined accurate to the line segment). However, some similarities can be found with motion planning for redundant manipulators (Siciliano 1990) and multi-robot assembly systems (Bonert *et al.* 2000).

### 2.2. *Performance measures*

For the considered task, which needs only five degrees of freedom, the redundant parameter $\gamma$ may be used to smooth the trajectory in joint variable space, in order to avoid sharp turns of the cutting tool (figure 2). This requirement may be formalized in several ways: as minimization of energy, minimization of joint velocities, minimization of joint coordinate range, minimization of joint displacement, etc. However, in each case it is necessary to deal with vector criterion because the dimension of this space is obviously higher than one.

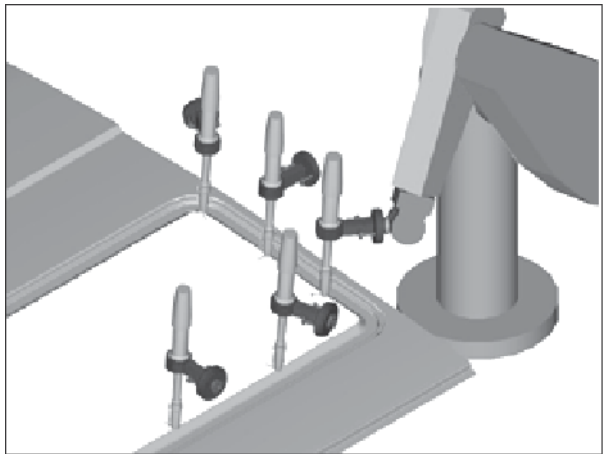

Figure 2. Smooth control of the tool orientation.

For a typical industrial robot, which possesses six degrees of freedom, the mapping from the task space $\{\boldsymbol{L}\}$ to the joint variable space $\{\boldsymbol{Q}\}$ is described by the inverse kinematic function

$$\boldsymbol{Q} = \mathrm{InvKin}(\boldsymbol{L}, M) \tag{8}$$

which is parameterized by the configuration index $M$ that allows us to resolve a non-uniqueness problem. This index $M$ belongs to a finite set (usually eight elements) which corresponds to different robot postures, such as '*elbow-up/ down*', '*arm left/ right*', '*wrist plus/ minus*'.

To ensure continuity of the joint-space trajectory, all of the inverse solutions must have the same configuration index (or belong to the same topological set). For example, if a robot starts with the 'elbow-up' solution, it cannot switch to the elbow-down configuration part way along the trajectory. Therefore, the mapping from the task space to the joint variable space defines several self-motion manifolds (Burdick 1989)

$$\boldsymbol{Q}(t, \gamma, M) = \mathrm{InvKin}[\boldsymbol{L}(t, \gamma), M]; \quad t \in [0, T], \tag{9}$$

that must be considered separately during optimization. In addition, let us define a similar space for the tool orientation angles

$$\Phi(t, \gamma, M) = \mathrm{ToolAng}[\boldsymbol{L}(t, \gamma), M]; \quad t \in [0, T], \tag{10}$$

which has a particular meaning depending on a convention adopted by the robot manufacturer (angles '*a, b, c*' for KUKA robots, '*o, a, t*' for PUMA robots, etc). For similarity purposes, in this paper the orientation angles will be denoted as $\varphi_1$, $\varphi_2$, $\varphi_3$.

Therefore, for given $M$, the function $\gamma(t)$ defines six joint trajectories $q_k(t)$

$$\gamma(t) \to \{q_1(t), \dots q_6(t)\}, \tag{11}$$

each of which may be evaluated by the following cost functionals:

*Joint coordinate range*

$$J_{\Delta}^{(k)}[\boldsymbol{q}(t)] = \max_t [q_k(t)] - \min_t [q_k(t)]. \tag{12}$$

*Joint coordinate deviation* from a prescribed value $r_k$ (from the centre of joint tolerances, for example)

$$J_d^{(k)}[\boldsymbol{q}(t)] = \max_t |q_k(t) - r_k|. \tag{13}$$

*Joint coordinate displacement*

$$J_s^{(k)}[\boldsymbol{q}(t)] = \int_0^T |\dot{\boldsymbol{q}}_k(t)| \mathrm{d}t. \tag{14}$$

*Joint maximum speed*

$$J_v^{(k)}[\boldsymbol{q}(t)] = \max_t [\dot{\boldsymbol{q}}_k(t)]. \tag{15}$$

It is obvious that mapping from the task space to the tool orientation space, which yields three trajectories

$$\gamma(t) \to \{\varphi_1(t), \dots \varphi_3(t)\}, \tag{16}$$

may also be evaluated applying the same performance measures: tool angle range, deviation, displacement, and maximum speed.

The geometrical meaning of these functionals is the following. The range measure evaluates the width of the smallest tube that contains the corresponding function. The deviation shows the bias of this tube relative to the prescribed value. The displacement characterizes the total amount of joint motion (without regards to the motion direction). Finally, the maximum speed estimates function smoothness.

It should be stressed that such functionals are computed for each joint variable, so the resulting performance measure is a vector:

$$\boldsymbol{J}(Q) = \mathrm{col}[\boldsymbol{J}(q_1(t)), \dots, \boldsymbol{J}(q_6(t))], \tag{17}$$

$$\boldsymbol{J}(\Phi) = \mathrm{col}[\boldsymbol{J}(\varphi_1(t)), \dots, \boldsymbol{J}(\varphi_3(t))]. \tag{18}$$

It also possible to evaluate the joint space trajectory by scalar criteria such as the following.

- *Maximum energy*

$$J_E[\boldsymbol{q}(t)] = \max_t \lfloor \dot{\boldsymbol{q}}(t)^{\mathrm{T}} \boldsymbol{W} \dot{\boldsymbol{q}}(t) \rfloor, \tag{19}$$

  where $\boldsymbol{W}$ is the weighting matrix, and

- *Maximum of inverse manipulability*

$$\boldsymbol{J}_M[\boldsymbol{q}(t) = \max_t \lfloor 1/\det(\boldsymbol{J}(\boldsymbol{q}(t))\boldsymbol{J}(\boldsymbol{q}(t))^{\mathrm{T}}) \rfloor, \tag{20}$$

  where $\boldsymbol{J}$ is the manipulator Jacobian matrix.

### 2.3. *Optimizing multiple objectives*

As follows from the previous section, it is not possible to describe completely the considered design requirements by a single objective. Although in ideal

(and obviously 'utopian') cases, all of the introduced objectives tend to zero, minimizing one of the components may degrade the performance in another. Therefore, the designer must choose one of the techniques that are usually used to balance multiple criteria (Steuer 1986, Cleary 1990, Pamanes and Zeghloul 1991, Chen *et al.* 1995, etc).

In this paper, instead of giving preference to a particular objective or optimization technique, it is proposed to leave the final decision until the design stage, when it may be chosen from the following options:

- defining the priority of partial objectives or the primary objective;
- applying the minimax technique, i.e. the worst-case optimization;
- assigning weights to combine multiple criteria in the linear function (weighted sum approach).

Independent of the chosen technique, the vector-optimization engine must include the scalar-optimization routines that are developed in the following sections.

## 3. Search space representation

A practical method to obtain the optimal solution under complicated constraints (equation (7)) is sampling of the search space (or presenting it by a grid). Such an approach transforms this space into a directed graph, with the states uniquely representing the matrix of the tool location $L$ and the vector of joint coordinates $Q$. A single path in the grid is composed of a sequence of segments connecting two adjacent nodes.

### 3.1. *Graph model*

For the considered problem, the given path (1) may be described by an evenly-distributed sequence of nodes:

$$\{\boldsymbol{p}_i, \boldsymbol{n}_i\}, \quad i = 0 : n \tag{21}$$

where

$$\|\boldsymbol{p}_i - \boldsymbol{p}_{i-1}\| = \Delta S; \quad \forall i = 1 : n$$

and $\Delta S$ is the sampling distance, which ought to be small enough to provide the desired approximation of the contour, but large enough to meet the control unit requirements.

Similarly, the interval of the redundant parameter $\gamma \in (-\pi, \pi]$ may be divided into $m$ segments

$$\gamma \in \{-\pi : 2\pi/m : \pi\} \tag{22}$$

and locations $\boldsymbol{L}[t, \gamma(t)]$ are tested for kinematic, collision and singularities constraints (equation (7)). So, after extraction of only those locations that satisfy the constraints, each node of the path (1) can be mapped into a set of tool locations and corresponding joint coordinates

$$\{\boldsymbol{p}_i, \boldsymbol{n}_i\} \longrightarrow \begin{Bmatrix} L_{i1}, & L_{i2}, & \dots & L_{im} \\ \dots & \dots & \dots & \dots \\ Q_{j1}, & Q_{j2}, & \dots & Q_{jm} \end{Bmatrix} \tag{23}$$

the number of which varies from node to a node.

Therefore, the feasible search space can be represented by a multi-layer directed graph (figure 3) with vertices

$$V = \{\boldsymbol{L}_{if}\} \tag{24}$$

and edges

$$E = \{(\boldsymbol{L}_{if}, \boldsymbol{L}_{kl}) | i = k - 1; \ \vee_i, j, k, l\} \tag{25}$$

where each layer corresponds to a particular configuration index $M$. As the result, the robotic path-planning task is reduced to the following network optimization problem.

**Transformed design problem**. For a given set of vertices $V$ and set of edges $E$, find the 'best' path of length $n$

$$\prod(\gamma_0 \dots \gamma_n) = \langle \boldsymbol{L}_{0j_1} \rightarrow \boldsymbol{L}_{1j_2} \rightarrow \dots \boldsymbol{L}_{nj_n} \rangle \tag{26}$$

with initial state $V_0 \in \{L_{0j}\}$ and final state $V_0 \in \{L_{nj}\}$, which minimizes the specified performance index.

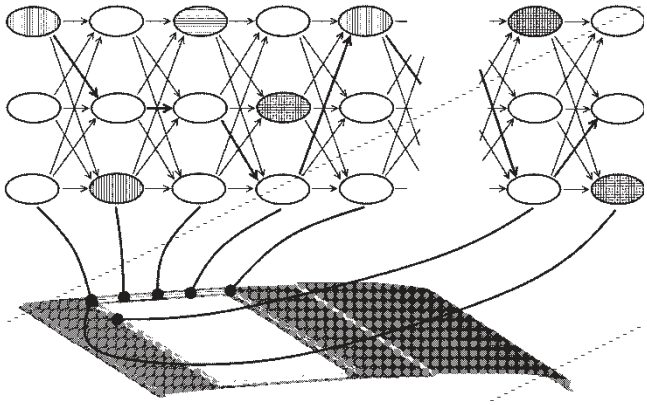

Figure 3. Graph representation of the search space.

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 one by adding virtual start and end nodes (common for all layers). In addition, all nodes are solvable for inverse kinematics and are admissible for collision and singularity tests (otherwise they are excluded from the graph (*V*,*E*) on the stage of the graph generation).

An alternative formulation of the problem deals with seeking the 'best' sequence

$$\Gamma = (\gamma_0, \gamma_1, \ldots \gamma_n); \quad \gamma_i \in \Gamma_i, \tag{27}$$

such that each element belongs to its own finite set $\Gamma_i$, which is extracted from equation (22) by testing $\boldsymbol{L}_{ij}$ for kinematic, collision and singularity constraints (equation (7)). Because for typical industrial cases $n > 1000$ and the sets $\Gamma_i$ may include up to 50 elements, the exhaustive search is impractical and there is a need to apply computationally efficient procedures. Clearly, the finer the search space representation, the closer is the solution to the true optimum, but the heavier is the computational load. However, there exist lower bounds for the grid resolution, which is determined by robotic controller parameters and is discussed below.

### 3.2. *Path sampling constraints*

As assumed in previous sections, the sequence $\{\boldsymbol{n}_i, \boldsymbol{p}_i\}$ is extracted from the workpiece graphical model where the cutting contour is converted into the 'augmented line'. Performing such a conversion, the designer must specify the appropriate distance between vertices $\Delta S$, to meet two competing goals. On the one hand, the distance should be small enough to ensure the desired accuracy of the path approximation. On the other, there exists a lower bound of the sampling step, which is determined by the parameters of a control unit. Let us investigate this problem in detail.

In industrial robot controllers, the path segments are generated using *trapezoid velocity profiles* that typically include three sections (acceleration, uniform motion, and deceleration). Their duration depends on the desired displacement and velocity/acceleration constraints imposed on each joint variable and on the Cartesian coordinates. Moreover, in continuous motion mode (i.e. without stopping at the path nodes), the 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 both for uniform motion and acceleration/deceleration sections (figure 4(a)).

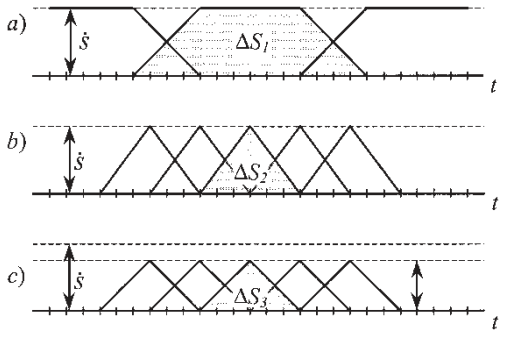


Figure 4. Velocity profiles for different $\Delta S$.

However, for short segments, the trapezium reduces to a triangle (figure 4(b)). In addition, the travel time for each section is sampled using the controller 'clock time' (towards a greater value) and is lower-bounded. Therefore, for short segments, the basis of the triangle is fixed while its height is adjusted to ensure the desired square that must be equal to the displacement. Thus, the joining of very short segments may yield a velocity that is less than the desired one (figure 4(c)).

To estimate the minimal value of $\Delta S$, let us assume that key constraints for the path planning are imposed on the Cartesian coordinates:

$$|\dot{s}(t)| \le \dot{s}_{\max}; \quad |\ddot{s}(t)| \le \ddot{s}_{\max} \tag{28}$$

and the length of the acceleration/deceleration section is equal to the lowest allowable value $\tau_{\min}$. Then, computing the square under the corresponding velocity profile and taking into account the time-sampling, yields

$$\Delta S \le \dot{s}\tau_{\min} \max\left\{1; 1 + \operatorname{int}\left(\frac{\dot{s}_{\max}}{\tau_{\max}\ddot{s}_{\max}}\right)\right\}. \tag{29}$$

Analysis of this expression for typical robots shows that the second term of the 'max' function may be neglected. Therefore, for engineering practice it can be used in a simplified version:

$$\Delta S \le \dot{s}\tau_{\min} \tag{30}$$

Applying this expression to the usual industrial cases (contour speed 0.05 m/s; clock time 0.016 ms; at least four time-samples per acceleration section) gives a rather high lower bound for the path sampling step, which is equal to 3.2 mm. Therefore, this constraint should certainly be taken into account by a CAD operator who normally tries to set the step to be less then 1 mm. To overcome this trouble, it is necessary to combine short contour segments into straight and

circular portions that are efficiently reproduced by a typical robot controller. It should be also noted that the sampling step $\Delta S$ is bonded by the 'clock time' of the Cartesian path planning (i.e. 'upper' control level), while joint-level planning is usually sampled with shorter time periods (up to 1 ms).

## 4. Generation of optimal path

Since the considered performance measures differ by their properties (additive, non-additive, etc), the optimization technique should also be different. In this section, several algorithms are proposed that minimize the performance measures (12)–(20) in acceptable time. To simplify the description of the algorithms, the joint coordinates corresponding to the location $\boldsymbol{L}_{ij}$ are denoted as $q_k(i,j)$, and the trajectories corresponding to the solution vector $\Gamma$ are denoted as $q_k(i,j_{\gamma i})$. The algorithms are equally applicable for optimization in both joint variable space $\boldsymbol{q}$ and tool orientation space $\varphi$, while the description below is given for the first case only.

### 4.1. *Minimization of coordinate deviation*

The optimization problem

$$J_d^{(k)}(\Gamma) = \max_i |q_k(i, j_{\gamma_i}) - r_k| \to \min_\Gamma \tag{31}$$

that minimizes the deviation of the $k$th joint variable with respect to the prescribed value $r_k$ may be solved in a straightforward way, just by selecting for each time instant $t_i$ the value of $\gamma \in \Gamma_i$ that yields a local minimum of the considered difference. It is obvious that such a solution also ensures a global optimum, although in the general case it is not unique. However, using the proposed multiobjective approach, the detected 'critical nodes'

$$(i_r, j_r) = \arg\left( \max_i \min_j |q_k(i,j) - r_k| \right) \tag{32}$$

that correspond to the maximum deviation of $q(t)$, may be converted into constraints, which are taken into account in the next steps, while applying other optimization criteria. Within the proposed formulation, such a transformation is performed by simple reduction of the set $\Gamma_i$ up to one element $\gamma_{jo}$.

It can be proved that a similar approach can also be applied to the minimization of the weighted sum

$$J_d^w(\Gamma) = \sum_k w_k J_d^{(k)}(\Gamma) \to \min_\Gamma, \tag{33}$$

and the 'worst' component

$$J_d^m(\Gamma) = \max_k J_d^{(k)}(\Gamma) \to \min_\Gamma, \tag{34}$$

of the vector performance measure.

### 4.2. *Minimization of the coordinate range*

The optimization problem

$$J_\Delta^{(k)}(\Gamma) = \max_i \lfloor q_k(i, j_{\gamma_i}) \rfloor - \min_i \lfloor q_k(i, j_{\gamma_1}) \rfloor \to \min_\Gamma, \tag{35}$$

which minimizes range of the $k$th joint variable, may be solved by simultaneous application of the previous algorithm and the nonlinear optimization technique. The problem can be reduced to seeking the best-prescribed value $r_k$ that yields a minimum of the corresponding deviation:

$$f^{(k)}(r_k) = \max_i \left[ \min_j |q_k(i,j) - r_k| \right] \to \min_{r_k} \tag{36}$$

In this case, the value $r_k$ is treated as the middle of the coordinate range, so the optimal solution $r_k^o$; gives two 'critical nodes'

$$\{(i_r^+, j_r^+), (i_r^-, j_r^-)\} = \arg\left( \min_{r_k} \max_i \min_j |q_k(i,j) - r_k| \right), \tag{37}$$

that correspond to the upper and the lower levels respectively. Similar to the previous case, the optimal solution is not unique, so the critical nodes may also be converted into constraints for the next optimization steps.

The minimization of the weighted sum

$$J_\Delta^w(\Gamma) = \sum_k w_k J_\Delta^{(k)}(\Gamma) \to \min_\Gamma, \tag{38}$$

and the 'worst' component of the vector objective

$$J_\Delta^m(\Gamma) = \max_k J_\Delta^{(k)}(\Gamma) \to \min_\Gamma, \tag{39}$$

are also reduced to the minimization of the nonlinear function of six unknowns $\boldsymbol{r} = (r_1, r_2, r_6)^{\mathrm{T}}$

$$f^w(\boldsymbol{r}) = \sum_k w_k \max_i \min_j |q_k(i,j) - r_k| \to \min_r \tag{40}$$

or

$$f^m(\boldsymbol{r}) = \max_k \max_i \max_j |q_k(i,j) - r_k| \to \min_r. \tag{41}$$

However, because of the non-smooth and polymodal nature of the objective function, conventional nonlinear optimization methods (step descent or gradient search, for instance) cannot be used here. An alternative approach is based on sophisticated random search techniques, e.g. simulating annealing, in particular (Kirkpatrick *et al.* 1983).

### 4.3. *Minimization of coordinate increment*

For the discrete representation of the search space, the coordinate velocity is estimated by the finite difference computed for the successive time instants (increment). Therefore, the related optimization problem is stated as

$$J_v^{(k)}(\Gamma) = \max_i |q_k(i, j_{r_i}) - q_k(i-1, j_{r_{i-1}})| \to \min_\Gamma, \quad (42)$$

and can be solved by means of the dynamic programming. To prove it, let us assume that, at the $p$th stage, there have been found all optimal sequences

$$\Gamma^\circ(p, \chi) = \langle \gamma_0, \gamma_1, \ldots, \gamma_{p-1}, \chi \rangle, \quad (43)$$

with the last element $\chi \in \Gamma_p$ and the corresponding performance measures are denoted as $F_p(\chi)$. Then, for the next stage, the optimal sequence

$$\Gamma^\circ(p+1, \gamma) = \langle \gamma_0, \gamma_1, \ldots, \gamma_{p-1}, \chi, \gamma \rangle, \quad (44)$$

with the last element $\gamma \in \Gamma_{p+1}$ may be found from the following recursion

$$F_{p+1}(\gamma) = \min_{\chi \in \Gamma_p} \max\{F_p(\chi), |q_k(p+1, j_\gamma) - q_k(p, j_\chi)|\}. \quad (45)$$

Therefore, staring from $p-1$ and sequentially increasing the length of the sequence, for each end state the recursion finds both the optimal path and the corresponding performance measure. Therefore, the last step is a simple selection of the best end state from the set $\gamma \in \Gamma_n$.

It is obvious that a similar approach can also be applied to minimization of the weighted sum and the 'worst' component of the corresponding vector performance measure. In addition, because of the common properties of the objectives $J_E$, $J_M$ and $J_v$, which are based on minimax expressions, such a recursion can be used for minimization of the maximum energy or the inverse manipulability as well.

### 4.4. *Minimization of coordinate displacement*

Using the discrete search space, this optimization problem is stated as follows:

$$J_s^{(k)}(\Gamma) = \sum_i |q_k(i, j_{\gamma_i}) - q_k(i-1, j_{r_{i-1}})| \to \min_\Gamma, \quad (46)$$

In contrast to the previous case, it is an additive performance measure that is accumulated along the path. However, it can also be minimized by applying the dynamic programming. Using the notation adopted in the previous section, the corresponding recursion can be written as

$$F_{p+1}(\gamma) = \min_{\chi \in \Gamma_p} \{F_p(\chi), + |q_k(p+1, j_r) - q_k(p, j_x)|\}. \quad (47)$$

So, staring from $p=1$ and sequentially increasing the length of the sequence $\Gamma^\circ(p,\gamma)$, for each end state, the recursion finds both the optimal path and the corresponding performance measure. As in the previous case, the last step deals with the selection of the best end state from the set $\gamma \in \Gamma_n$. It can easily be proved that a similar recursion also yields an optimal solution for the weighted sum and the 'worst' component of the corresponding vector performance measure.

## 5. Simulation results

To demonstrate the proposed technique, let us consider the simple example shown in figure 5. The manipulator consists of three revolute joints and three links of lengths $l_1 = 1.0$, $l_2 = 1.0$ and $l_3 = 0.25$. The cutting contour is defined as a square with the side $d = 0.8$, whose angles are rounded with radius $r = 0.10$. The centre of the contour is located at the point (1.0, 1,0) and is surrounded by an obstacle (see figure 5) with gap $\Delta d = 0.05$. After the sampling, the contour is presented as a set of 60 uniformly distributed nodes (shown as small circles).

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

$$\begin{aligned} x &= l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3); \\ y &= l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) + l_3 \sin(q_1 + q_2 + q_3); \\ \varphi &= q_1 + q_2 + q_3, \end{aligned} \quad (48)$$

where $q_1$, $q_2$, $q_3$ are the joint coordinates, and $x$, $y$, $\varphi$ are the tool location parameters (two Cartesian coordinates and an orientation angle). To derive the inverse kinematic model, let us define

$$\begin{aligned} x' &= x - l_3 \cos(\varphi); \\ y' &= y - l_3 \sin(\varphi); \end{aligned} \quad (49)$$

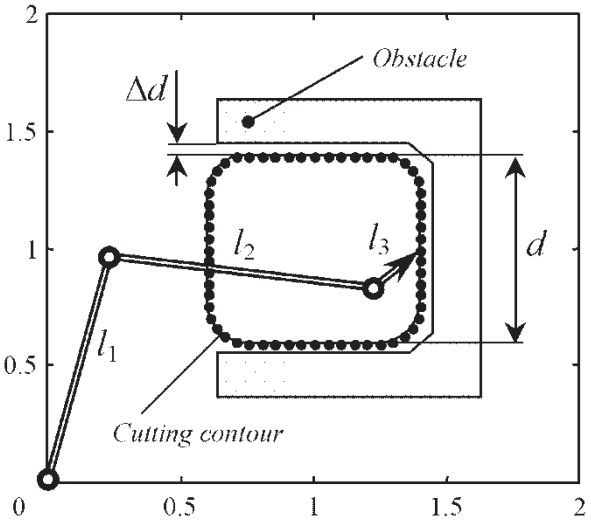


Figure 5. A planar three degree of freedom illustrative example.

and sequentially solve the obtained equations for $q_2$, $q_1$ and $q_3$:

$$\begin{aligned} q_2 &= M\arccos\left((x')^2+(y')^2-l_1^2-l_2^2\right)/2l_1l_2, \\ q_1 &= \operatorname{atan2}\frac{x'(l_1+l_2\cos(q_2))+y'l_2\sin(q_2)}{y'(l_1+l_2\cos(q_2))-x'l_2\sin(q_2)}, \\ q_3 &= \varphi - q_1 - q_2, \end{aligned} \tag{50}$$

where $M=\operatorname{sgn}(q_2)$ is the configuration index.

Using the inverse model and altering the tool orientation $\varphi$ with the step of 10°, a set of 1385 feasible tool locations $\{\boldsymbol{L}_{ij}\}$ and a corresponding set of joint coordinates $\{\boldsymbol{Q}_{ij}\}$ 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, until any one was detected. For this example, the approach yields satisfactory results and is not time consuming. In addition, only one configuration index was found, $M=-1$ yields the required solutions for all nodes of the cutting contour.

To investigate the relative importance of the considered performance measures, optimal solutions were first found for a single objective applied to a single coordinate $q_1$, $q_2$ or $q_3$. The following optimization criteria were used: the minimum of the range $J_\Delta$, the minimum if the maximum increment $J_v$, and the minimum of the total displacement $J_s$. The obtained solutions are presented in figures 6–8 where the search trees for the objective $J_s$ are also shown (as a background and demonstration of the search space shape). As follows from these plots, the optimal solutions differ mainly at the beginning and

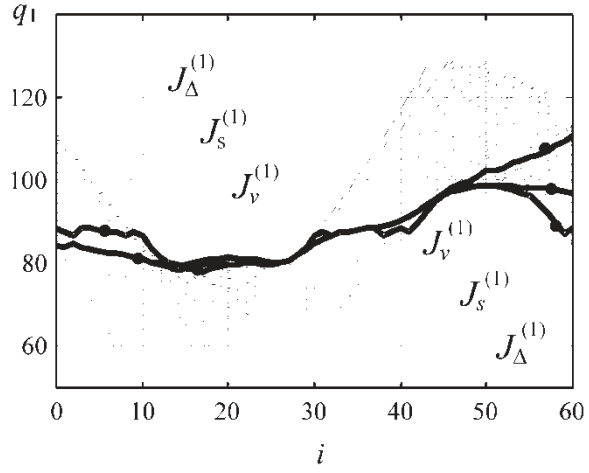


Figure 6. Optimal solutions for the coordinate $q_1$.

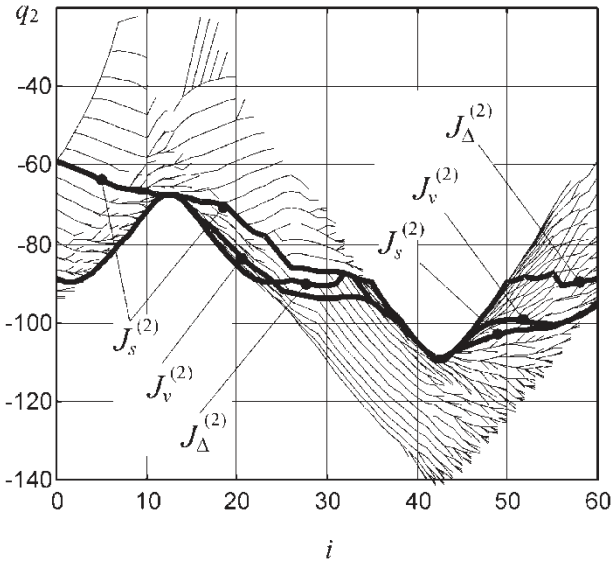


Figure 7. Optimal solutions for the coordinate $q_2$.

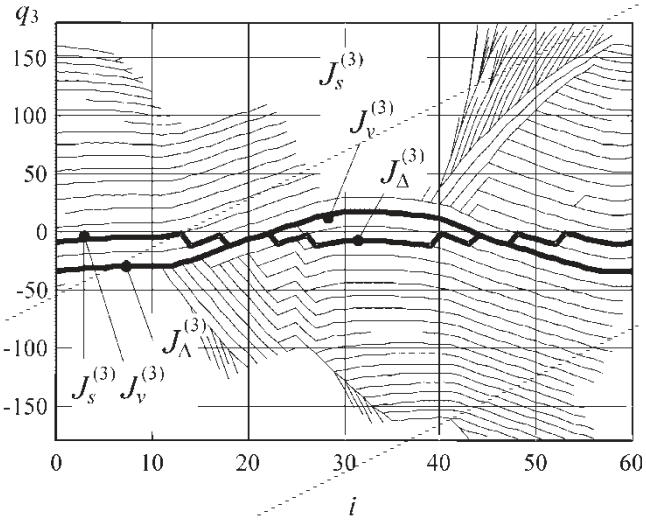


Figure 8. Optimal solutions for the coordinate $q_3$.

the end, while their middle parts almost coincide. In addition, an insufficiently small sampling step causes inclination of these segments, although they should obviously be close to the horizontal to ensure the minimum of $J_s$.

The corresponding numerical values of the considered performance measures are presented in tables 1–3. The minimization of $J_s$ (joint displacement) yields a result that is also satisfactory for other objectives, so it may be chosen as the primary performance measure to present the engineering requirement of a 'smooth' trajectory. However, further analysis shows that minimizing $J_s$ for one joint may lead to a very sharp profile for the remaining ones, especially for the third joint (see figures 9 and 10). Therefore, the competing objectives must be balanced by computing the weighted sum or the 'worst' component of the vector criteria.

The results for the simultaneous optimization of all joint trajectories are presented in table 4. As follows from these results, the weighted sum approach (with equal weights), as well as the 'worst case' minimization, yield roughly the same results, which are also close to

Table 1. Performance measures for optimization of $q_1$ [deg].

| Objective | $J_\Delta^{(1)}$ | $J_v^{(1)}$ | $J_s^{(1)}$ |
|---|---|---|---|
| Minimum of range $J_\Delta^{(1)}$ | 19.09 | 2.69 | 56.56 |
| Minimum of increment $J_v^{(1)}$ | 32.9 | 1.47 | 42.79 |
| Minimum of displacement $J_s^{(1)}$ | 19.09 | 1.47 | 27.70 |

Table 2. Performance measures for optimization of $q_2$ [deg].

| Objective | $J_\Delta^{(2)}$ | $J_v^{(2)}$ | $J_s^{(2)}$ |
|---|---|---|---|
| Minimum of range $J_\Delta^{(2)}$ | 41.32 | 3.31 | 98.78 |
| Minimum of increment $J_v^{(2)}$ | 50.72 | 2.54 | 69.24 |
| Minimum of displacement $J_s^{(2)}$ | 49.76 | 3.19 | 62.92 |

Table 3. Performance measures for optimization of $q_3$ [deg].

| Objective | $J_\Delta^{(3)}$ | $J_v^{(3)}$ | $J_s^{(3)}$ |
|---|---|---|---|
| Minimum of range $J_\Delta^{(3)}$ | 11.50 | 10.69 | 161.8 |
| Minimum of increment $J_v^{(3)}$ | 52.26 | 3.84 | 105.0 |
| Minimum of displacement $J_s^{(3)}$ | 52.26 | 3.84 | 105.0 |

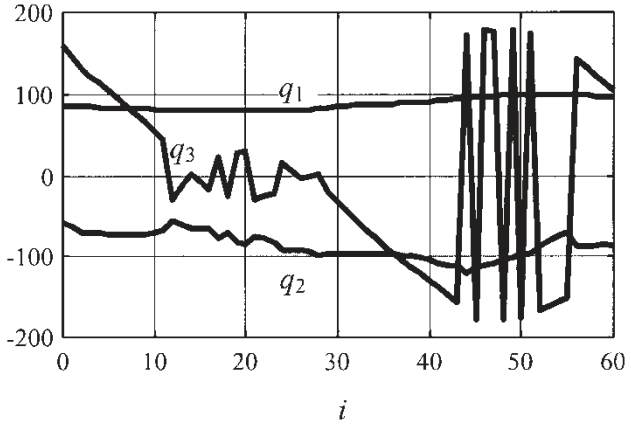


Figure 9. Joint trajectories for minimization of $J_s^{(1)}$.

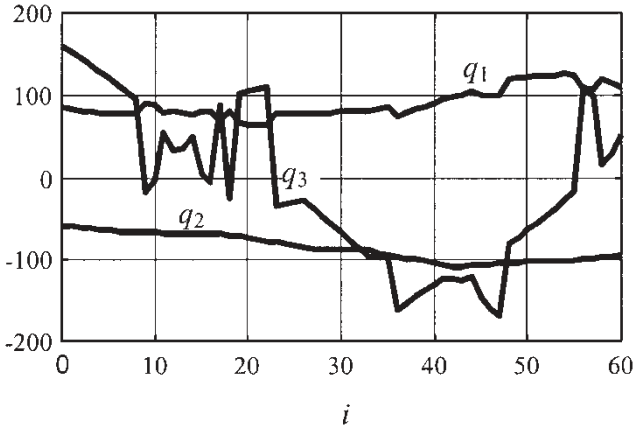


Figure 10. Joint trajectories for minimization of $J_s^{(2)}$.

Table 4. Complete sets of the performance measures for all case studies (in parenthesis, given minimum achievable values).

| Objective | $J_s^{(1)}$ (27.7) | $J_s^{(2)}$ (62.9) | $J_s^{(3)}$ (105.0) | $J_v^{(1)}$ (1.47) | $J_v^{(2)}$ (2.54) | $J_v^{(2)}$ (3.84) | $J_\Delta^{(1)}$ (19.09) | $J_\Delta^{(2)}$ (41.32) | $J_\Delta^{(3)}$ (11.50) |
|---|---|---|---|---|---|---|---|---|---|
| Minimum of weighted displacement $\sum_k w_k J_s^{(k)}$ | 106.6 | 153.6 | 105.7 | 3.96 | 3.70 | 4.00 | 51.31 | 76.80 | 52.72 |
| Minimum of maximum increment $\max_k J_v^{(k)}$ | 107.1 | 157.2 | 105.0 | 3.82 | 3.86 | 3.84 | 52.06 | 78.58 | 52.26 |
| Minimum of displacement $J_s^{(1)}$ | 27.7 | 222.6 | 3080.3 | 2.76 | 19.43 | 355.31 | 19.09 | 66.09 | 357.26 |
| Minimum of displacement $J_s^{(2)}$ | 281.5 | 62.9 | 1901.8 | 24.31 | 3.57 | 202.50 | 62.49 | 49.76 | 335.59 |
| Minimum of displacement $J_s^{(3)}$ | 107.1 | 157.2 | 105.0 | 3.82 | 3.86 | 3.84 | 52.06 | 78.58 | 52.26 |

the result for the minimization of $J_s^{(3)}$. Therefore, in this particular case, the third joint may be considered as a 'key' one and the solution presented in figure 11 may be chosen as the output of the vector optimization process. However, in the general case, it is prudent to generate a set of candidate solutions by altering the weights of the combined performance measure and automatically selecting only Pareto-optimal ones. The obtained set of solutions must be presented to the designer who makes the final decision.

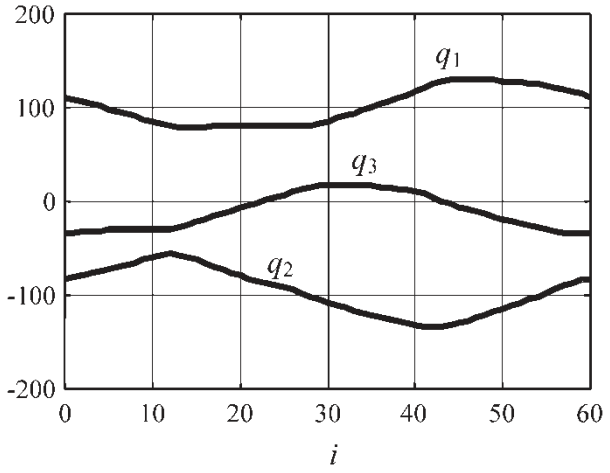


Figure 11. Joint trajectories for minimization of $J_s^{(3)}$.

## 6. Industrial implementation

The algorithms developed here have been successfully implemented on the manufacturing floor, in ROBOMAX CAD package (Buran Co, Russia-USA), which is a powerful integrated system for computer-aided design and offline programming of industrial robotic cells. It is already used in the Russian automotive industry and has been successfully applied to the design of manufacturing lines/cells for Lada cars (AO AutoVAZ, Tolyatti), GAZEL lorries (AO GAZ, Nizhniy Novgorod) and ZIL mini-vans (AMO ZIL, Moscow). The package is completely integrated with Autodesk CAD products and includes a number of auxiliary tools, which enable the user to design a robotic cell and generate a control program, taking into account the particularities of the technology employed.

In application to laser and plasma cutting technology, the Robomax/Laser subsystems (figure 12) allows one to design the workcell layout and optimize the robot motion using multiobjective optimization techniques. As the first step, using standard routines of the Autodesk Mechanical Desktop (AMD), the mathematical description of the cutting contour is presented in the form of the 'augmented line'. The user may define either the desired distance between vertices or their total number. In addition, the user can estimate the

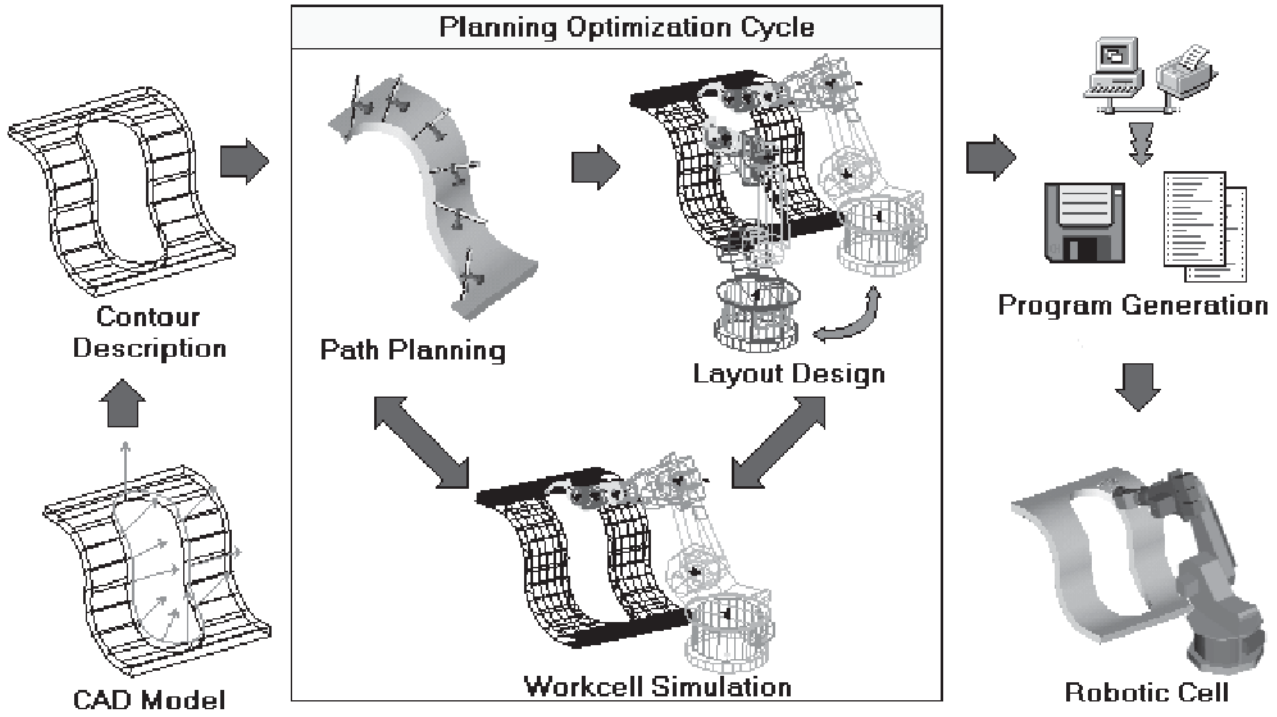


Figure 12. Robotic cell design and programming using Robomax/Laser.

minimum number of vertices required to keep the accuracy within the tolerance setting.

The main design procedure consists of three iteratively repeated steps. The first step is the selection of the proper manufacturing environment and the location of the manufacturing task within the robot workspace. The user can assemble and optimize 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 added from a library. To optimize the position of the workpiece, a set of performance measures is used that take into account such important features as distance to the obstacles, closeness to joint limits, and the required range of joint motion.

The second step deals with path planning using algorithms described in the previous sections. The goal is to utilize the redundancy, which is introduced by the tool rotation around its axis of symmetry. To balance competing objectives, the designer can interactively assign priorities or weights, as well as activate/deactivate some of the criteria. In particular, it is possible to choose between joint/tool angles optimization, and to change the preferable criteria after visual analysis of the obtained path.

For the third step, the obtained solution is verified using realistic simulation of 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.

In addition, it is possible to inspect the velocity and acceleration profiles for each joint, as well as details of the manipulator 3D motion. If the user is not satisfied with the current solution, he or she can return to the first or the second step to change the design parameters and so get a new solution.

After finishing this optimization cycle, the obtained path is converted into the control program in the robotic system language. In this stage, it is also possible to optimize the program, in order to eliminate some nodes (that belong to straight-line segments), or to add any necessary additional commands required for controlling the technological parameters. Finally, the created program is downloaded to the robotic cell controller (see figure 12).

The recent application of ROBOMAX/Laser is the offline programming of a robotic cutting station for AMO ZIL (Moscow). The station includes a KUKA PR161 robot and corresponding positioning and clamping devices. It is used for small-batch manufacturing, which requires frequent reprogramming (almost every day). The industrial experience has shown that the developed technique provides good capabilities for generating optimal programs for robotic cutting of 3D parts of complex shape.

## 7. Conclusions

The developed technique allows the generation of optimal movements of robotic manipulators in 3D space, taking into account the kinematic redundancy and particularities of the laser cutting technology. Incorporating these results in a graphic simulation system leads to the essential reduction of process planning time, enabling even very small batch sizes to become economically feasible for robotic processing.

The particular contribution of this paper deals with the multiobjective optimization of robot motion that is based on simultaneous optimization of performance measures for all joint coordinates. To generate smooth motion, each joint trajectory is evaluated by a set of performance indices such as the coordinate range, deviation, maximum increment, and total displacement. The search space is converted into the directed graph and the problem is re-formulated in terms of combinatorial optimization theory. The optimal solution is obtained via dynamic programming procedures that minimize the weighted sum of the objectives (or the 'worst' of them) and yield the result within an acceptable time for industrial applications. During the optimization, the weights are altered to generate a set of Pareto-optimal solutions.

The proposed algorithms have been implemented in a commercial software package that is already used in the Russian automotive industry. The algorithms will also encourage further research into multiobjective optimization of robot systems for this technological application, such as the optimization of a robotic cell layout and the optimization of a contour processing sequence.

## Acknowledgements

The authors gratefully acknowledge Vladimir Markov and Sergej Marmuzevich (Buran Co, Moscow) for their support and collaboration in the industrial implementation of the work described in this paper.

## References

Backes, F., Geiger, M., and Franke, V., 1996, Technology oriented off-line programming for 3D laser processing. *Technical Papers of the North American Manufacturing Research Institution* (SME), pp. 241–246.

BAUER, L., and BACKES, F., 1995, Offline-programming of 3D laser systems. *Preprints of the Fourth International Workshop on Robotics in Alpe-Adria Region,* Pörtschach, **1,** pp. 47–50.

BAUER, L., 1996, Offline-programming for 3D-laser systems. *European Symposium on Lasers, Optics, and Vision for Productivity in Manufacturing, SPIE Proceedings,* **2787,** 34–42.

BONERT, M., SHU, L. H., and BENHABIB, B., 2000, Motion planning for multi-robot assembly systems. *International Journal of Computer Integrated Manufacturing,* **13**(4), 301–310.

BURDICK, J. W., 1989, On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds. *Proceedings of the IEEE International Conference on Robotics and Automation,* Scottsdale, AZ, pp. 264–270.

CHEN, W., ZHANG, O., YANG, Z., and GRUVER, W. A., 1995, Optimising multiple performance criteria in redundant manipulators by subtask-priority control. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics,* Vancouver, Canada, pp. 2534–2539.

CLEARY, K., 1990, Incorporating multiple criteria in the operation of redundant manipulators. *Proceedings of the IEEE International Conference on Robotics and Automation,* Cincinnati, Ohio, pp. 618–624.

GEIGER, M., and KOLLÉRA, H., 1994, Tool path optimisation for 2D laser cutting. *Production Engineering, Annals of German Academic Society for Production Engineering,* **1**(2), 155–158.

GEIGER, M., and OTTO, A., 2000, *Laser in der Elektronikproduktion and Feinwerktechnik, Tagungsband des 3. Erlanger Seminars LEF 2000* (Bamberg: Meisenbach).

KIRKPATRICK, S., GELATT, C. D., and VECCHI, M. P., 1983, Optimisation by simulated annealing. *Science,* **220**(4598), 671–680.

LATOMBE, J. C., 1991, *Robot Motion Planning* (Dordrecht, Netherlands: Kluwer Academic).

OTTO, A., BACKES, F., and GEIGER, M., 1997, Enhanced process planning for 3D laser beam welding of sheet metal parts. *The Second World Congress on Intelligent Manufacturing Processes and Systems,* Budapest, Hungary, pp. 491–496.

PAMANES, J. A., and ZEGHLOUL, S., 1991, Optimal placement of robotic manipulators using multiple kinematic criteria. *Proceedings of the IEEE International Conference on Robotics and Automation,* Sacramento, CA, pp. 933–938.

SENDLER, U., 1994, Offline-programmiertes Laserschneiden im Automobilbau. *VDI-Z,* **136**(1/2), 28–30.

SICILIANO, B., 1990, Kinematic control of redundant robot manipulators: a tutorial. *Journal of Intelligent Robotic Systems,* **3**(3), 201–212.

STEUER, R., 1986, *Multiple Criteria Optimisation: Theory, Computation and Application* (New York: Wiley).

YOSHIKAWA, T., 1985, Manipulability of robotic mechanisms. *International Journal of Robotics Research,* **4**(2), 3–9.