Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

MANUFACTURING PROCESS PLANNING FOR LASER CUTTING ROBOTIC SYSTEMS

Alexandre Dolgui[1], **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

[2]*Department of Automatics and Production Systems*
Ecole des Mines de Nantes, 4 Alfred-Kastler St., 44307 Nantes, France

Abstract: the paper presents a new computational technique for the manufacturing process planning in laser cutting robotic systems. It focuses on the optimisation of robot motions for continuous contour tracking using the redundancy caused by the tool axial symmetry. In contrast to previous works, the developed technique is based on the dynamic programming and explicitly incorporates verification of the velocity/acceleration constraints. It also takes into account recent advances in robot mechanical design allowing unlimited rotations of some manipulator axes. The technique is implemented in a CAD package and verified in the automotive industry. *Copyright © 2008 IFAC*

Keywords: Computer-aided manufacturing, robotics, redundant manipulators, path planning, dynamic programming, multiobjective optimisation.

1. INTRODUCTION

Recent advances in laser technology motivate amending the existing robot path planning methods, which do not allow the complete utilisation of the actuator capabilities and neglect some particularities in the manipulator mechanical design. At present, the cutting speed is comparable with the kinematic capabilities of industrial robots (Schlueter, 2005), so their performances are becoming a bottleneck in enhancing the cutting cells productivity.

A typical robotic laser cutting system consists of a laser, a beam-delivery-system and a cutting head integrated with a 5- or 6-axis manipulator. One of the recent developments in this field, the Robocut system (www.rpt.net), is based on a 5-axis anthropomorphic manipulator with standard 3-axis forearm architecture and a reduced 2-axis wrist with endlessly rotating 4th and 5th joints. This special design offers essential advantages, since the cycle time losses can be avoided for the reverse rotations. In spite these benefits, the 5-axis robots possess essential disadvantages related to the difficulties to solve the inverse kinematic problem. So, most of the cutting robotic cells are based on the 6-axis robots with the standard 3-axis wrist allowing unlimited rotation of the 4th and 6th axes. Obviously, this redundancy simplifies robot control and programming, and also increases the flexibility of the manufacturing cell, while posing another problem: optimal utilisation of the kinematic redundancy.

For robotic laser cutting, most of the related research focuses on the off-line programming, which allows essentially reduce the system down time and make economically feasible even very small batch sizes (Kaierle, 1999; Mitsi et al., 2005). At the moment, there are a number of commercial off-line programming systems on the market. However, there still exists a considerable gap between their capabilities and requirements of a particular application. And up to now, the robot programs for many 3D cutting applications are constructed interactively. The ultimate goal is the automatic generation of robot programs from CAD drawings, similar to CNC machining.

For manipulators with six degrees of freedom, the motion planning problem was firstly addressed in (Abe et al., 1994; Shibata et al., 1997). These authors proposed a genetic algorithm that optimises the cutting tool orientation using the evaluation function extracted from the experience of skilled operators. However, they succeeded in the generation of manipulator motions for relatively slow cutting speed. Some recent techniques that employ the general end-effector constraints concept (Yao and Gupta, 2007) also suffer from this drawback.

An alternative approach, proposed by the authors of this paper (Pashkevich et al., 2004), is based on the graph-based search space representation and dynamic programming. These yields essential gain in computation speed and allowed successfully apply the technique in industry. Nevertheless, recent advances in technology and the essential increase of the cutting speed motivate further improvements.

This paper focuses on the enhancement of our method by imposing additional constraints on the trajectory smoothness and taking into account the ability of some robot axes for unlimited rotations. Its remainder is organised as follows. Section 2 is devoted to the problem statement, Section 3 presents the main theoretical results, Section 4 contains simulation and implementation issues, and Section 7 summarizes the main contributions.

2. PROBLEM STATEMENT

2.1 Manufacturing 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 is described by two vector functions as follows:

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

here 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. These data can be directly extracted from the graphical model of the part, by defining the processing contour as an "*augmented line*". The path is assumed to be closed and time-uniformly sampled into the sequence of nodes $\{\boldsymbol{p}_k,\ \boldsymbol{n}_k;\ |\ k = 0,...n\}$, 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.
To describe the tool spatial location, let us also define the Cartesian displacement along the path $\Delta\boldsymbol{p}_k = \boldsymbol{p}_{k+1} - \boldsymbol{p}_k$; $k = 0,...n-1$ 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 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 composed of the vectors $\boldsymbol{a}_k$, $\boldsymbol{a}_k \times \boldsymbol{n}_k$, $\boldsymbol{n}_k$, $\boldsymbol{p}_k$ and is denoted as $\boldsymbol{H}_k$.

The 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

$$\boldsymbol{L}_k(\gamma_k) = \boldsymbol{R}_z(\gamma_k)\ \cdot\boldsymbol{H}_k\ ,\ \ \gamma_k \in (-\pi,\ \pi]\ , \qquad (2)$$

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 (2), 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 dimension of the relevant search space.

2.2 Constraints

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). Here, *task constrains* are expressed in the terms of the required position/orientation of the tool and are described by expression (2). Robot *kinematics constraints* are caused by the manipulator geometry. And *collision constraints* arise from the need to avoid collisions between the robot and workcell components.

To define the kinematic constrains more precisely, let us express the tool location $\boldsymbol{L}$ corresponding to the joint coordinate vector $\boldsymbol{q}\in R^6$ as

$$\boldsymbol{L}(\boldsymbol{q}) = \boldsymbol{T}_{tool}\cdot\left(\prod_{i=1}^{6} {}^{i}\boldsymbol{T}_{i-1}(q_i)\right)\cdot\boldsymbol{T}_{base}\ , \qquad (3)$$

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, and the matrix $\boldsymbol{T}_{base}$ defines the robot base location. Particular expressions for the homogenous matrices ${}^{i\text{-}1}\boldsymbol{T}_i$ for various manipulators can be found in common reference books (Spong et al., 2006).

For the inverse transformation $\boldsymbol{q} = \boldsymbol{f}_c^{-1}(\boldsymbol{L},\ \mu)$, let us introduce the configuration index $\mu\in\mathrm{M}$ that determines the manipulator posture, where M contains all combinations of admissible configurations (*shoulder right/left, elbow up/down, wrist plus/minus*). It should be stressed that typical robot controllers do not allow changing the manipulator configuration while moving between successive nodes and using the Cartesian space on-line interpolation. But for cutting applications, this specific kinematic constraint can be released by temporarily switching to the joint-space interpolation. So, in contrast to our previous work (Pashkevich et al., 2004), here we do not use separate manifolds for each value of μ.

Using the inverse kinematic transformation, the sequence of the admissible tool locations can be mapped into the joint coordinate space

$$\boldsymbol{C}_Q = \left\{ \begin{matrix} \boldsymbol{Q}_k(\gamma_k,\mu_k) = \boldsymbol{f}_c^{-1}(\boldsymbol{R}_z(\gamma_k)\ \cdot\boldsymbol{H}_k,\ \mu) \\ \gamma_k \in (-\pi,\ \pi],\ \ \mu_k \in \mathrm{M},\ k\ = 1\ldots n \end{matrix} \right\} \tag{4}$$

taking into account both the workspace dimension limits (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 further convenience, the constraint violation case is denoted as $\boldsymbol{Q}_k(\gamma_k) = \varnothing$. It is also 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 application point of view, the desired path planning algorithm should produce "*smooth motions at reasonable speeds and at reasonable accelerations*". Within the frames of the adopted path presentation (1), the joint velocity/acceleration constraints may be expressed via the finite-difference approximation as:

$$\left| q_{i,k} - q_{i,k-1} \right| < \eta_v \cdot\ \Delta q_i^{(v)}\ , \tag{5}$$

$$\left| q_{i,k} - 2q_{i,k-1} + q_{i,k-2} \right| < \eta_a \cdot\ \Delta q_i^{(a)}\ , \tag{6}$$

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 ith 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 summarized in inequality $\boldsymbol{Q}_k(\gamma_k) \neq \varnothing$ and in the expressions (5), (6), which ensure the path existence and its admissible curvature in the joint coordinate space.

2.3 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. This means that the qualitative performance measures should be based on some type of the "*smoothness/economy"* measures applied to all manipulator joints (Edan and Nof, 1997).

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*

$$J_i^{(s)}(\boldsymbol{\gamma},\boldsymbol{\mu}) = \sum_{k=1}^{n} \left| q_{i,k}(\gamma_k,\mu_k)\text{-}q_{i,k-1}(\gamma_{k-1},\mu_{k-1}) \right|, \tag{7}$$

- *maximum increment* (*speed*)

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

- *coordinate range*

$$J_i^{(\Delta)}(\boldsymbol{\gamma},\boldsymbol{\mu}) = \max_k \left[q_{i,k}(\gamma_k,\mu_k) \right] - \min_k \left[q_{i,k}(\gamma_k,\mu_k) \right], \tag{9}$$

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.

Intuitively, the minimization of each of these criteria should lead to a smoother generated path. However, as follows from our studies, the objectives (7) – (9) may compete with each other. Besides, these indices are computed for each joint coordinate. Thus, the resulting performance measures form a vector and the designer must choose one of the existing multi-criteria optimisation techniques (Cheng and Shih, 1997). However, independent of the chosen technique, the corresponding vector-optimisation engine must include the scalar-optimisation routines that are developed below.

3. PATH PLANNING ALGORITHM

3.1 Search space presentation

To obtain an optimal solution under the above constraints, let us sample the feasible domain for the redundant parameter γ. This transforms the continues search space into an acyclic directed graph, with the vertices uniquely representing the tool location matrix $\boldsymbol{L}$ and the vector of joint coordinates $\boldsymbol{Q}$.

In particular, 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 $\mu\in$M, the locations (2) are tested for the 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 $\{\boldsymbol{L}, \boldsymbol{Q}\}$:

$$\{\mathbf{p}_k, \ \mathbf{n}_k\} \to \bigcup_{j=1}^{m_k} \begin{Bmatrix} \mathbf{L}_{k,j} \\ \mathbf{Q}_{k,j} \end{Bmatrix} \tag{10}$$

where m_k is the number of the successful locations for the kth node. It should be noted that the "*path-smoothness*" constraints (5), (6) are not tested at this stage yet, since they are associated with several successive locations.

Then, the feasible search space can be represented by a directed graph with the vertices

$$V = \bigcup_{k=0}^{n} \bigcup_{j=1}^{m_k} \begin{Bmatrix} \mathbf{L}_{k,j} \\ \mathbf{Q}_{k,j} \end{Bmatrix} \tag{11}$$

and edges

$$E = \bigcup_{k=1}^{n} \bigcup_{j=1}^{m_k} \bigcup_{l=1}^{m_{k-1}} \left(\begin{Bmatrix} \mathbf{L}_{k,j} \\ \mathbf{Q}_{k,j} \end{Bmatrix}, \begin{Bmatrix} \mathbf{L}_{k-1,l} \\ \mathbf{Q}_{k-1,l} \end{Bmatrix} \right), \tag{12}$$

which connect only successive tool locations. Hence, the original robot control problem is reduced to a specific best-path problem for a graph (11), (12), where the performance measure should incorporate the design objectives (7)…(9). Besides, both the initial and final vertices are not unique, but the problem can be transformed to the classical problem by adding the virtual start and end nodes.

Since the Cartesian path is sampled uniformly, the related *distance matrix* should be based on a joint space metric. Also, it is necessary to take into account that the actuator capacities (i.e. maximum speed, acceleration, etc.) are different for different manipulator axes. So, the displacement components Δq_i corresponding to the joint displacement vector $\Delta\boldsymbol{q} = (\Delta q_1, \ldots \Delta q_6)$ should be weighted. The most prudent is assigning the weights $w_i = (\dot{q}_i^{\max}\Delta t)^{-1}$, where $\dot{q}_i^{\max}$ is the maximum axis speed specified by the manufacturer. And finally, the distance in the weighted joint space may be defined using the Euclidean, Manhattan or Chebychev metrics because their clear physical meaning.

While computing these distances, it is also necessary to take into account that the 4th and 6th manipulator axes may allow unlimited rotation. In this case, the differences Δq_i must be pre-processed in accordance with the expression:

$$\Delta q_i = \min_{p\in Z} \ \{ \ \Delta q_i + 2\pi\, p \} \ , \ i \in \{4, \ 6\} \tag{13}$$

where p is an integer number.

3.2 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 (7) - (9) should be converted into an aggregate scalar criterion. At this step, a relevant metric (Manhattan or Chebychev) is applied to the sequence of the path segments, while inside the segments the weights are altered between optimisation runs to produce a set of Pareto-optimal solutions.

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

$$J^{(s)}(\boldsymbol{\gamma},\boldsymbol{\mu}) = \sum_{k=1}^{n} \rho_a\left(\boldsymbol{Q}(k,j_k)\text{-}\boldsymbol{Q}(k \ - \ 1,j_{k-1})\right) \tag{14}$$

$$J^{(\Delta)}(\boldsymbol{\gamma},\boldsymbol{\mu}) = \max_k \rho_a\left(\boldsymbol{Q}(k,j_k)\text{-}\boldsymbol{Q}(k \ - \ 1,j_{k-1})\right) \tag{15}$$

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 $\boldsymbol{Q}(k,j)$. So, 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 $\rho(k_1,j_1, k_2,j_2)$.

For the the objectives (14) and (15), an optimal path can be found by means of dynamic programming. A basic expression for the proposed algorithm is derived in the following way. Let us consider a reduced order sub-problem with k clusters S_0, S_1, … 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, … 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}:

$$d^*_{k,j} = \min_p \left\{ d^*_{k-1,p} + \rho\,(v_{k-1,p}, v_{k,j}) \right\} \ , \ k = 1,\ldots n \tag{16}$$

Therefore, the desired solution for n+1 clusters S_0, … 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 $d^*_{n,j}$, $j = 1,2,\ldots m_n$. The corresponding sequence of the j-indices $\{j_0^*, j_1^*, \dots j_n^*\}$, which defines the shortest path, is extracted in the 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 (16). 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 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)

Procedure: Path_planning

```
(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 $J_{max}(k-1)$ do
              (α) Set $d_{cur} := M_{dist}(k-1, p) + \rho(k, j, k-1, p)$
              (β) If $k > 0$ & $f_v(k, j, p) \neq 0$
                    Set $d_{cur} := d_\infty$
              (γ) If $k > 1$ & $f_a(k, j, p, M_{ind}(k-1, p)) \neq 0$
                    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 $J_{opt}(n-k) := j_{opt}$;  $j_{opt} := M_{ind}(n-k, j_{opt})$;
```

A distinct feature of this procedure is contained in sub-steps (2bβ) and (2bγ) that verify the path-smoothness constraints (5) and (6). Sub-step (2bβ) incorporates the function $f_v(k, j, p)$, which evaluates the velocity constraints (5) for all manipulator axes while the manipulator moves from the location $\boldsymbol{L}_{k-1,p}$ 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 $\boldsymbol{L}_{k-2,l}$, $\boldsymbol{L}_{k-1,p}$, $\boldsymbol{L}_{k,j}$, where the location $\boldsymbol{L}_{k-2,l}$ is assumed to be the optimal predecessor of $\boldsymbol{L}_{k-1,p}$, i.e. $l = M_{ind}(k-1, p)$.

A similar algorithm can be applied for the minimax design objective $J^{(\Delta)}$. The only modification needed deals with the sub-step (2bα). 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 (5) and (6)), the designer can generate a collection of the candidate solutions to be included in the Pareto-optimal set.

4. IMPLEMENTATION RESULTS

To evaluate efficiency of the proposed technique, first it was applied to the planar cutting task described in detail in (Pashkevich et al., 2004). For this task, two techniques were applied (the known and the proposed ones), and a set of solutions was obtained that differ by both the optimisation criteria and the weights: w_i, η_v and η_a. As follows from our study, for the previous technique, tuning of the weights w_i can barely produce acceptable results. Most of the solutions have a tendency for undesirable oscillations in the orientation axis. In contrast, the new technique simplifies obtaining smooth solutions with the balanced values of the partial criteria. This result is illustrated in Fig. 1 and 2, which contain the trajectories generated both methods.

The developed algorithms have been also implemented on the manufacturing floor, in ROBOMAX CAD package, which is already used in automotive industry and has been successfully applied for the design of a number of manufacturing lines. With respect to 3D 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.

5. CONCLUSION

This work presents a new computational technique for the manufacturing process planning in laser cutting robotic systems. Its particular contribution is related to the multi-objective optimisation of the manipulator motions for the continuous contour tracking with an axis-symmetric technological tool.

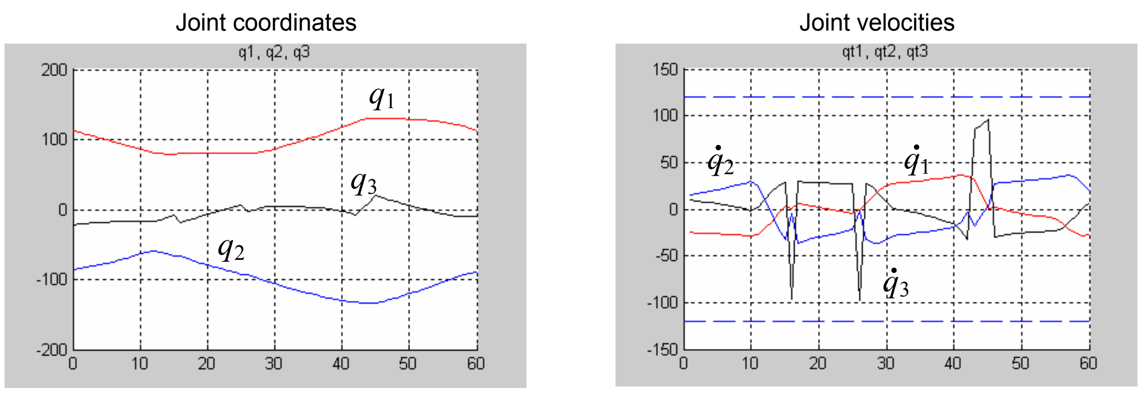

Fig. 1. Optimisation results for the known technique

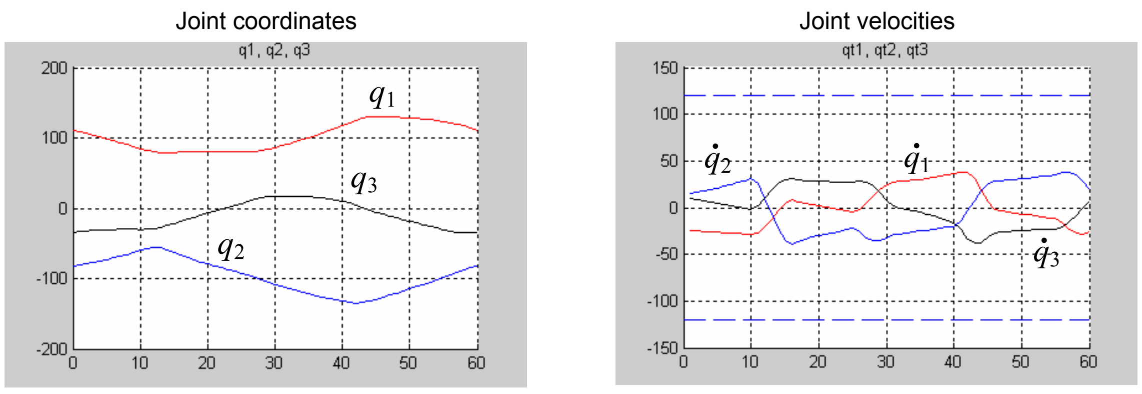

Fig. 2. Optimisation results for the proposed technique

In contrast to previous 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 technique is implemented in a CAD package and verified in the automotive industry.

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 Int. Conference. on Fuzzy Logic, Neural Nets and Soft Computing*, 453-454.

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

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

Hwang, Y.K., Chen, P.C., Maciejewski, A.A. & Neidigk, D.D. (1994). Global motion planner for curve-tracing robots. *IEEE Int. Conference on Robotics and Automation*, 662-667.

Kaierle, S., Fuerst, B., Kittel, J., Kreutz, E.W. & Poprawe, R. (1999). Design and manufacturing tools for laser beam processing. *Proceedings of SPIE*, Vol. 3833, 148-159

Mitsi, S., Bouzakis, K.-D., Mansour, G., Sagris, D. & Maliaris, G. (2005). Off-line programming of an industrial robot for manufacturing. International Journal of Advanced Manufacturing Technology, 26(3), 262-267.

Pashkevich, A., Dolgui, A. & Chumakov, O. (2004). Multiobjective optimization of robot motion for laser cutting applications. *Int. J. of Computer Integrated Manufacturing*, 17(2), 171-183.

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

Shibata, T., Abe, T, Tanie, K. & 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.

Spong, M., Hutchinson, S., Vidyasagar, M. (2006). *Robot modeling and control*, John Wiley & Sons, 478 pp.

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