PRM Based Motion Planning for Sequencing of Remote Laser Processing Tasks

Villumsen, Sigurd Lazic; Kristiansen, Morten

Published in:
Procedia Manufacturing

DOI (link to publication from Publisher):
10.1016/j.promfg.2017.07.109

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 11 (2017) 300 – 310

Procedia MANUFACTURING

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy

PRM Based Motion Planning for Sequencing of Remote Laser Processing Tasks

Sigurd Lazic Villumsen[a,*] Morten Kristiansen[a]

[a]*Aalborg Universitet, Fibigerstræde 16, Aalborg 9220, Denmark*

Abstract

The mechanical system used for remote laser processing can contain as much as 9 degrees of freedom (DOF). In this paper, a sample based motion planning algorithm for such remote laser processing equipment is presented. By construction robot configurations through a sampling strategy redundancy is inherently taken into account and the path is ensured to comply with laser processing constraints. A test showed that the algorithm was capable of finding 1277/1280 possible paths in 2000 iterations for a 9 DOF mechanical system. These 1277 paths were represented in matrix form which can be used for sequencing of laser processing tasks.

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing

Keywords: Remote laser cutting; Remote laser welding; Production planning; Scheduling ; Redundancy; Robotics; Motion planning; Path planning

1. Introduction and state of the art

In the last decade the field of remote laser processing has received a great deal of attention from the scientific community [1,2]. This is especially evident for remote laser welding (RLW) which has the potential to reduce cycle times significantly when compared to e.g. traditional resistance spot welding [3]. One of the benefits of remote laser

* Corresponding author. Tel.: +45 30130852.
E-mail address: sv@m-tech.aau.dk

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing
doi:10.1016/j.promfg.2017.07.109

processing is that the beam can be moved over the work piece by embedding galvanometer driven mirrors in the processing head [4].

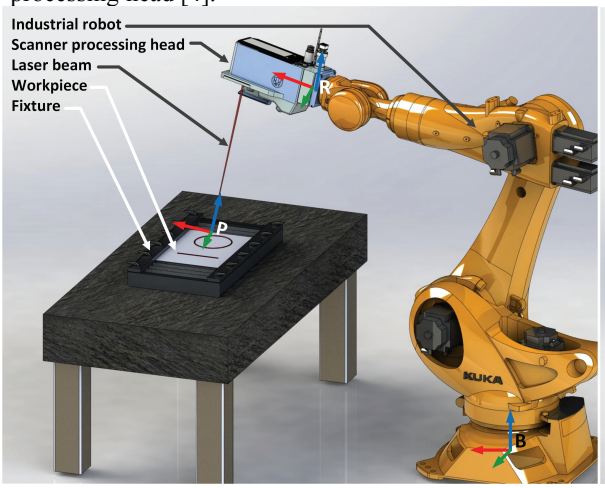

Fig. 1. A KUKA Quantec KR 120 R2500 pro mounted with an Arges remote welding elephant head. Notice the position of the three frames, the base frame B, the robot frame R and the processing frame P.

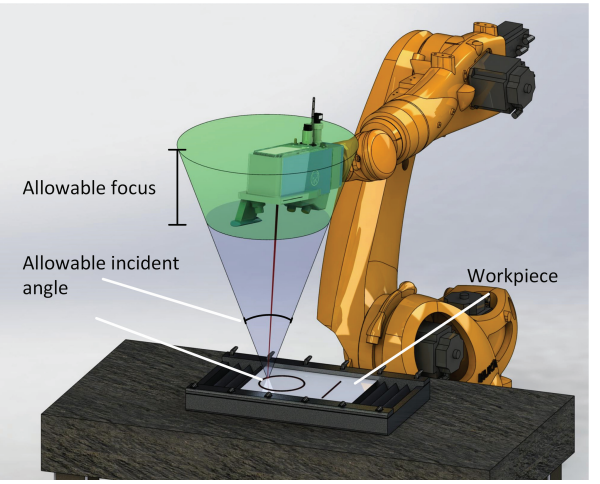

Fig. 2. The region in which the scanner cutting processing head can be situated and still yield a stable process.

Such actuated processing heads are generally referred to as scanner heads as they are capable of scanning the beam over the work piece with very high velocities. In the scientific literature such scanner heads are often mounted on industrial robots to combine the reach and flexibility of the robot with the capabilities of the scanner head. Such a setup is depicted in Fig. 1. The mechanical complexity of the system makes efficient on-line programming difficult. This means that an off-line programming approach needs to be utilized when generating the robot and scanner head programs. Such off- line programming approaches have already been presented in the scientific literature for remote laser welding (RLW) and remote laser cutting (RLC).

An on-line programming system for RLW with conventional optics is presented in [5]. By using a simple solver for the traveling salesman problem (TSP) an initial path is generated. This initial path is then improved and modified by changing the laser incident angle. The Cartesian distance is used as the cost function minimized by the TSP solver.

A combined planning system for RLW and RLC is presented in [6–8]. The planning system works by identifying regions around each task that enables the cutting head to reach the cutting kerf. These circular regions are denoted scan circles and are connected by straight lines. Paths are connected by using the traveling salesman algorithm. By utilizing scan circles it is however limited to 2D shapes, and as paths are connected by straight lines optimality cannot be guaranteed. Also the working field of the laser is limited to a circle and the level of redundancy of the robot and scanner system is not utilized to its full extent.

An integrated approach to rough cut path planning and task sequencing for RLW with scanner mirrors is presented in [9–11]. Access volumes in the shape of truncated cones are defined around entry and Exit points of weld seams. These access volumes are constrained by two parameters, the allowable incident angle of the laser beam and by the allowable focus range. Redundancy is handled by fixing a joint to ”mid-range” value.

In [12] an approach to planning of remote laser welding tasks for an articulated robot mounted with scanner mirrors is presented. It is based on robot configuration clusters that allows for welding of the seam in a task. One cluster is defined for each task and a generalized travelling salesman problem (GTSP) solver connects these clusters by the shortest path. Here redundancy is taken into account explicitly, but robot motions are not allowed while processing. Even though the above frameworks show promising results, they all, except [12], treat the problem of task sequencing and robot motion planning in Cartesian space. Also, redundancy with respect to the processing task is not handled explicitly. In [12] redundancy is taken into account, but the robot is not allowed to move while processing the weld seams. To expand the concept of the GTSP approach used in [12] to allow for movements of the

entire mechanical system, it is necessary to determine to what extent it can reconfigure itself while processing a given laser task.

In this paper an algorithm for determining the extent for reconfiguration will be presented. It is based on a probabilistic road map (PRM) algorithm for generating collision free paths between a set of entry and exit configurations for a redundant robot laser cutting machine. The sampling strategy ensures that the end effector path

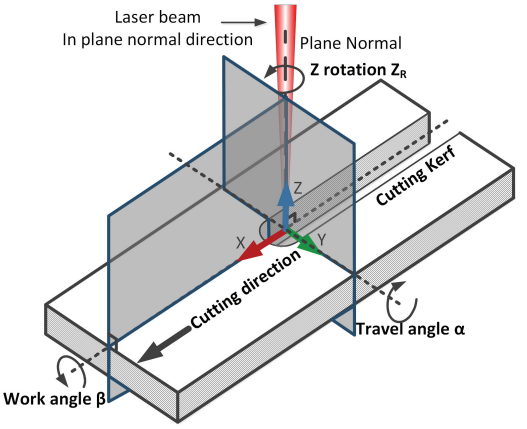

Fig. 3 The frame assignment in the cutting kerf. The Z axis points in the direction of the plane normal, the X axis points in the direction of the cutting direction. The Y axis is constructed by the cross product of the X and Z axis.

complies with process constraints. The resulting paths are concatenated in a matrix form that can be used for task sequencing in a GTSP framework.

2. Problem Formulation and Solution strategy

When a laser process is conducted on a work piece by laser processing equipment, it can be seen as the task of making the laser beam traverse a contour on the surface of the work piece. When a task is processed with a given remote technology the process will often yield the necessary quality within a range of processing parameters. These processing parameters are for example, laser power, processing speed, focus offset and incident angle. When regarding the geometric parameters that effects the process stability, the focus offset and incident angle needs to be considered. If the laser process remains stable for a range of focus offsets and incident angles this means that the laser processing head does not have to be in a specific position to achieve stable cuts or welds. In fact, the processing head can be placed anywhere within a truncated cone (with arced top and bottom) above the work piece. This is depicted in Fig. 2. In [21] the angle of incidence is decomposed into two angles, a work angle β and a travel angle α. Combined with a z rotation: Z_R (Rotation around the beam axis). These angles define three rotational parameters that can describe the angular deviation from a given processing point. These angles are depicted in Fig. 3. The allowable angles depend on the process. For Remote Fusion Cutting (RFC), stability is maintained with α and β angles in the range of $\pm 6°$ [21]. For remote laser welding $\pm 15°$ is often used [10]. Similar data can be found for focus offset, but, as the quality of many laser processes are sensitive to offsets in focus (e.g. RFC and remote ablation cutting (RAC)) the focus will be treated as a fixed parameter in this paper. Furthermore, as most current laser processes use intensity profiles which can be described as round Gaussian distributions, Z_R angle between $-180°$ and $180°$ can be allowed. However, some emerging remote laser processes rely on beam shaping to achieve better melt ejection. As these shaped beams are non-round and non-Gaussian it can only be assumed that they need to be aligned with the cutting direction. This alignment will be defined by the a rotation around the beam axis and thus constraints could be imposed on Z_R for future technologies[1].

By allowing α, β and Z_R deviations it becomes clear that the task space is only defined by the three Cartesian translational parameters x, y and z. With imposed constraints on the orientation variables β, α and Z_R. As described briefly in the introduction a piece of remote laser processing machinery consists of two hardware components, a

robot and a scanning processing head. Typically, the robot has 4-6 degrees of freedom (DOF) and the scanner head has three degrees of freedom (two mirrors for deflection and a linear axis for focus adjustment). If a standard 6 DOF industrial robot is chosen, the resulting system contains 9 DOF. As the task space is only defined by 3 variables this implies that the system contains 6 degrees of redundancy. This again entails that the cone description, see Fig. 2, of the allowable position of the robot end effector seems very simplified. If the joint variables associated with the industrial robot are labelled θ_1-θ_6 an industrial robot configuration vector q_I can be defined as : $q_I = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6]^T$. Furthermore if the joint variables associated with the cutting head are denoted φ_1(Scanner mirror 1), φ_2(Scanner mirror 2) and δ (focus offset) then a cutting head configuration vector q_C can be defined to be: $q_C = [\varphi_1 \quad \varphi_2 \quad \delta]^T$. By combining these vectors, a complete joint vector q is obtained:

$$q = \begin{bmatrix} q_I \\ q_C \end{bmatrix} \tag{1}$$

A point in the space expanded by q will in the following be named a configuration and a point in the task space of the robot will be called a pose. If the laser processing machinery is considered a fixed base manipulator with n joints whose end effector task can be represented by m variables, then the forward kinematics can be expressed as the mapping from configuration space to task space.

$$p = f(q) \tag{2}$$

Where $p \in \mathbb{R}^m$ is the end effector pose and $q \in \mathbb{R}^n$ is the joint vector. With the described mechanical system and the chosen representation of the task space it can be seen that $n = 9$ and $m = 3$. When considering the contour that needs to be processed it can be represented as a set of poses that the robot needs to reach one after another. The series of which can be indexed by a normalized path variable σ such that $\sigma \in [0,1]$. If such a path is denoted $p(\sigma)$it is possible to rewrite equation 2.

$$p(\sigma) = f\big(q(\sigma)\big), \forall \sigma \in [0,1] \tag{3}$$

If the focus point is considered as being the robot end effector the contour that needs to be traversed for laser processing can furthermore be seen being the desired path $p(\sigma)$. It is furthermore assumed that the contour lies within the dexterous task space $\mathcal{T}$ of the manipulator. This can mathematically be defined as:

$$p(\sigma) \in \mathcal{T}, \forall \sigma \in [0,1] \tag{4}$$

If $p(\sigma)$ is sampled in the interval [0,1] the curve can be represented as a sequence of points with orientation that the end effector needs to track : $\{p(\sigma_0) \quad p(\sigma_1) \quad \dots \quad p(\sigma_{s-1}) \quad p(\sigma_s)\}$. Where s is the number of samples and $\sigma_0 = 0$ and $\sigma_s = 1$. In the following $p(\sigma_0)$ will be denoted the entry point of the contour and $p(\sigma_s)$ will be denoted the exit point of the contour as it is here the laser processing begins and ends. Finally, as the robot is redundant in terms of the laser processing task (m<n) two things becomes clear; first, the path $p(\sigma)$ is not unique and second, the entry and exit points can be realized by ∞ joint configurations. To determine valid entry and exit configurations it is necessary to develop a map of the possible reconfigurations from a set of entry configurations to a set of exit configurations while the beam still traverses the contour. Given L entry configurations and I exit configurations the problem of finding a connecting map can be defined as finding a set of paths between these entry and exit configurations that still comply with the task space constraints defined by the contour that needs to be processed. The problem of finding a single path between a given entry configuration i and exit configuration k can be seen as a path planning problem where the Cartesian path is constrained to be on the laser processing contour. The map can then be defined as being an $I \times L$ matrix that contains the cost of reaching an exit configuration from a given entry configuration while traversing $p(\sigma)$. If two configurations cannot be connected a very large integer (999999) indicates no connectivity. If the set of entry configurations are labelled $q_{S_1} - q_{S_L}$ and a set of exit configuration $q_{E_1} - q_{E_I}$ have been found for a contour $p(\sigma)$ then the connectivity matrix R is of the in equation 5:

$$R = \begin{array}{c} \\ q_{S_1} \\ q_{S_2} \\ q_{S_3} \\ \vdots \\ q_{S_L} \end{array} \begin{array}{c} \begin{array}{ccccc} q_{E_1} & q_{E_2} & q_{E_3} & \dots & q_{E_I} \end{array} \\ \begin{pmatrix} c_(1,1) & c_(1,2) & c_(1,3) & \dots & c_(1,I) \\ c_(2,1) & c_(2,2) & c_(2,3) & \dots & c_(2,I) \\ c_(3,1) & c_(3,2) & c_(3,3) & \dots & c_(3,I) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_(L,1) & c_(L,2) & c_(L,3) & \dots & c_(L,I) \end{pmatrix} \end{array} \tag{5}$$

With this definition it is seen that it is necessary to find $I \cdot L$ constrained paths. To obtain such a connectivity matrix the constrained multi query motion planning algorithm needs to be implemented.

2.1. *Solution strategy*

The task of generating a set of joint trajectories that can execute the Cartesian path given by the contour that needs to be processed will in the following be defined as motion planning. The task of motion planning is thus somewhat more restrictive compare to the task of traditional Cartesian path planning where neither time nor the robot joints are taken into account. In the following a solution approach to the multi-query constrained motion planning problem will be presented.

2.1.1. Motion planning

As the proposed sample space contains 7 variables it is necessary to ensure that the chosen algorithm is capable of handling large sample spaces. For large task and configuration spaces the task of motion planning is often done by probabilistic methods due to their ability to find solutions even for complex environments [17]. In [13,14] a motion planning problem constrained by a Cartesian path is defined as Motion Planning along End- effector Paths (MPEP). Herein, the joint vector of the robot is divided into a base vector and a redundancy vector. The redundancy vector is sampled and path constraints are complied with by solving the inverse kinematics for the base vector. The algorithms rely on samples drawn from the end effector path at regular intervals which are then checked for collisions. A more general framework for constrained motion planning has been introduced namely the constrained bidirectional rapidly exploring random trees (CBIRRT) [15]. This framework allows for motion on the self-motion manifold, and incorporates sampling methods for constraint manifolds. Both of these presented algorithms are however based on variations of rapidly exploring random trees (RRT). This formulation is however not convenient for multi query problems. Instead a solution similar to that defined in [13,14] will be used, but instead of basing it on RRT it will be based on a probabilistic roadmap approach. Before the motion planning algorithm is described the strategy for generating robot configurations will be discussed.

2.1.2. Generation of robot configurations

If a position on the cutting kerf is described by using only three variables (x, y and z) and three constrained variables(α, β *and* Z_R). The 9 DOF mechanical system contains 6 degrees of redundancy in terms of the laser processing task. To generate solutions an approach would be to specify values for a subset of the parameters 9 joint values and solving the inverse kinematics for the remaining. In [13,14] it is proposed that the joint vector q is decomposed into a base vector q_b and a redundancy vector q_r. To generate a configuration on that places the end effector on the path the redundancy vector is sampled and the inverse kinematics of the manipulator can then be used to solve for q_b. This formulation is however inconvenient due to the strict orientation constraints of the laser processes. This would lead to a large proportion of the sampled configurations to be rejected based on these constraints. Instead it is proposed to sample directly on the constraint manifold by defining a transformation matrix T that can be used to permute the orientation of any kerf transformation. Then by sampling the cutting head joint vector q_C directly a sampling vector q_{PS} for a given point on the cutting kerf can be defined as: $q_{PS} = [\beta \quad \alpha \quad Z_R \quad \phi_1 \quad \phi_2 \quad \delta]^T$. In the same manner a sample vector for generating a configuration directly on the end effector path can be defined by appending the path variable σ.

$$q_P = [\sigma \quad \beta \quad \alpha \quad Z_R \quad \phi_1 \quad \phi_2 \quad \delta]^T \tag{7}$$

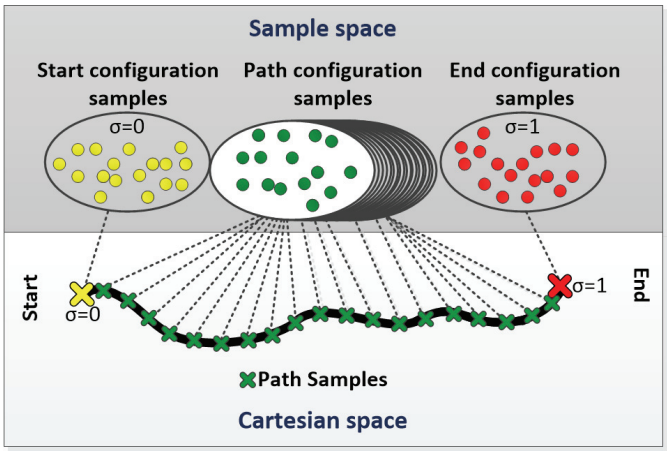

Fig. 4. The sampling concept used for generating the robot configurations. By employing the proposed sampling strategy sampled points on a contour can be converted to robot configurations in compliance with process constraints.

By incorporating such a sampling strategy, a set of valid robot configurations can be constructed for all sampled points on a contour. This is depicted in Fig. 4. By adding the path variable σ to the sample vector all samples contain temporal information as well. This will in the next section be used to transform manipulator joint speed saturations to obstacles in the sampling space. The base vector is then essentially the entire robot vector q_I which is convenient as the problem of solving the inverse kinematics for all joints of a given industrial robot is often trivial compared to solving it for only a subset of the joints. This furthermore allows for non-uniform sampling of the end effector path which can be useful for movements on e.g. the self-motion manifold. Furthermore, by using this formulation a sample space is found which can be used in conjunction with many of the standard sample based robotic motion planning algorithms such as RRT (Rapidly exploring Random Trees) and Probabilistic road maps (PRM). The drawback is however that the inverse kinematics of a 6 axis industrial robot generally contains several solutions. To ensure that a one to one mapping between the sample vector and the robot configuration exists it has been chosen to fix the number of solutions to ones which have the elbow up and wrist not flipped.

2.1.3. Planning algorithm

With the sampling strategy defined in the previous section the task of constructing the connectivity matrix R can be investigated. As the task of finding R can be considered a multi query problem it has been chosen to base it on PRM. In PRM position samples are repeatedly generated and saved as nodes in a topological graph. When a new sample is generated a connection procedure tries to connect the newly generated node with its nearest neighbors. If the connection is successful (if it doesn't hit an obstacle) the connection is added to the graph as an edge. By using PRM in the Sample space defined by equation 7 paths that comply with process constraints can be generated. However, it is still necessary to modify the standard PRM algorithm to ensure that processing is conducted with equal speed without backtracking over the Cartesian contour. If $G(V, E)$ is a topological graph where V is a set of vertices and E is a set of edges then the process of building $G(V, E)$ can be described by algorithm 1 – BUILD_ROADMAP which is based on the definition from [16] and [17] but with a modified approach the edge adding procedure. The **first** modification is that it has been chosen to insert a limit L to stop the edge adding procedure when a certain number of edges has been added to a node. This entails that in the described procedure, K nearest neighbours are considered, but a maximum of L are added. This ensures a better connectivity of hard to reach nodes and at the same time minimizing the number of total edges which increases speed. The **second** modification is that instead of operating on a single nearest nodes list, it has been chosen to work with two lists of nodes NF and one NB. Where NF is a list of nodes

lying ahead on the end effector path, this entails that all nodes in the list NF will have a larger σ than the newly added node. Similarly, NB contains nodes lying before the newly added node on the end effector path. Now, when considering the edges are added to the graph a node is taken from each list and tested for connectivity. This is done to reduce the risk of a central node being only connected to nodes in one σ direction. In this algorithm the procedures GET_F_NODES and GET_B_NODES returns a sorted list of L nodes ahead or behind the newly added node S respectively. The G.SAME determines if two nodes are identical and the two procedures ADD_EDGE(NF(p),S) and G.ADD_VERTEX(S) adds edges or vertices to the Graph. The GENERATE_SAMPLE_ON_PATH is derived from the procedure described in 3 and explained in pseudo code in algorithm 3. The functions GET_PATH_TRANS (σ) returns the path transformation for a given σ in the format specified by Fig. 3. The function CALC_CUTTING_TRANS ($\varphi_1, \varphi_2, \delta$) calculates the transformation matrix that relates the focal point of the laser to the robot tool flange. The CALC_PERMUTATION(β, α, Z_R) calculates the permutation matrix as described in section 3. Finally, the INVKINE function calculates the inverse kinematics based on the transformation matrix provided.

```
function BUILD_ROADMAP
    i ← 0
    while i < N do
        S ← GENERATE_SAMPLE_ON_PATH
        if S ∈ C_FREE then
            G.ADD_VERTEX(S)
            i++;
            NF ← GET_F_NODES(G,S,K)
            NB ← GET_B_NODES(G,S,K)
            p ← 0
            NAdded ← 0
            while p < K AND NAdded<L do
                if not G.SAME(NF(p),S) and CON-
NECT(NF(p),S) then
                    G.ADD_EDGE(NF(p),S)
                    NAdded++;
                end if
                if not G.SAME(NB(p),S) and CON-
NECT(NB(p),S) then
                    G.ADD_EDGE(NB(p),S)
                    NAdded++;
                end if
                p++;
            end while
        end if
    end while
    G.CONNECT_UNCONNECTED()
end function
```

Algorithm 1. Pseudocode representation of the BUILD_ROADMAP algorithm. It is modified in two ways to increase connectivity

```
function CONNECT(N1,N2)
    S1 ← N1.SampleVector
    S2 ← N2.SampleVector
    t_Path = GET_PATH_TOTAL_TIME()
    t_segment = abs(S1(1) − S2(2)) · t_Path
    NSamples = ceil((t_segment)/SAMPLE_RATE)
    t_sample = t_segment/NSamples
    SList = INTERPOLATE(S1, S, NSamples)
    for i=2 to NSamples do
        q1 ← SAMPLE_2_CONF(SList(i − 1))
        q2 ← SAMPLE_2_CONF(SList(i))
        q̇ = (q1−q2)/t_sample
        if SPEED_SATURATION(q̇) then
            return 0;
        end if
        if COLLISION(q2) then
            return 0
        end if
    end for
    return 1
end function
```

Algorithm 2. Pseudocode describing the implemented CONNECT algorithm. It is based on a standard connect algorithm.

The connection algorithm can be seen in algorithm 2. The function CONNECT_UNCONNECTED () is a heuristic function that tries to connect clusters of nodes that are unconnected. It works by searching the newly generated graph for all node clusters. To ensure a better connectivity two clusters are selected at random. Within each of these two clusters a node is selected and a connection is attempted made with the closest node in the opposite cluster. This pairing is repeated K times. This entire procedure is repeated P times. This ensures a better connectivity of the produced graph. Of importance here is that the linear interpolation often used for node connections (Here obtained by the function INTERPOLATE) is generated in sample space, and it can thus be ensured that all samples

are in compliance with the limitations of the laser process. It should also be noted that as the first entry in the sample vector is the path variable σ it is possible to evaluate joint speed constraints from the sample vector directly. This is done in the function SPEED_SATURATION. Finally, the COLLISION function checks for collisions and the SAMPLE_2_CONF converts the sample vector to a configuration vector in a process very similar to the one specified in algorithm 3. The function GET_PATH _TOTAL_TIME() calculates the total time it takes to traverse the end effector path.

When the algorithm has run the sample space paths needs to be extracted. This is done by finding the shortest path from each entry node to each exit node by using Dijkstras shortest path algorithm [18]. To ensure that the found paths are continuously moving forward all edges E in $G(V, E)$ are treated as being unidirectional in the direction going from low σ values to higher σ values. When these paths are extracted the construction of the R matrix becomes trivial, as it is just composed of the cost of the possible paths.

2.1.4. Definition of distance measures

The algorithm presented in the previous section dictates the need to differentiate nodes based on a distance measure. The definition of such a distance measure is however not necessarily trivial. An often used distance metric is the norm of the difference in sample vector for two nodes. If this distance is denoted ds then for two nodes n1 and n2 with sample vectors S 1 and S 2 it can be calculated by: ds = |S 1 − S 2|. The problem with this is that as we have included σ in the sample vector such a metric will be very dependent on the path and not the configuration of the robot. The two sample vectors might however be identical except for different σ values. To remedy this one could consider the two joint vectors q1 and q2 that corresponds to the sample vectors S_1 and S_2. If this is done a distance metric d_q can be defined: $d_q = |q_1 - q_2|$. Here it is ensured that two vertexes are in fact close to each other with respect to the robot configuration. However, if the robot needs to move between the two configurations in a given time the maximum joint speeds needs to be taken into account. If these max speeds are denoted $q_{\mathrm{j\,max}}$ where j is the joint number, then a weight matrix W can be constructed that has $1/q_{j\,max}$ in the diagonal. With this the weighted configuration, vector distance is defined as: $d_{qW} = |W \cdot (q_1 - q_2)|$. Even though this would ensure that the algorithm is less likely to connect nodes that cannot reach each other due to the slow moving axes two configurations might still be sampled from two very different parts of the end effector trajectory. For a closed contour this could be $\sigma_1 = 0$ and $\sigma_2 = 1$ which can be reached by identical robot configurations but as the interpolation between two points is done in sample space over the contour an edge between them will have to traverse the entire end effector path. To reduce the occurrence of the algorithm trying to connect nodes far away from each other on the contour one might consider factoring in the difference in σ values for the two nodes. For two nodes n_1 and n_2 with joint vectors q_1 and q_2 and sigma values σ_1 and σ_2 a distance metric is defined as it can be calculated in the following way: $d_{\sigma qW} = (1 + \omega \cdot abs(\sigma_1 - \sigma_2)) \cdot d_{qW}$. Where ω is a weight determining the magnitude of the effect of differences in σ. The listed distance measures will in the following be tested on an end effector trajectory. The resulting implementations of the algorithm will in the following be denoted $R_{ds}, R_{dq}, R_{d_{qW}}$ and $R_{d_{\sigma qW}}$

2.1.5. Implementation

function GENERATE_SAMPLE_ON_PATH
  $\sigma \leftarrow RAND(0, 1)$
  $\beta \leftarrow RAND(\beta_{min}, \beta_{max})$
  $\alpha \leftarrow RAND(\alpha_{min}, \alpha_{max})$
  $ZR \leftarrow RAND(ZR_{min}, ZR_{max})$
  $\phi_1 \leftarrow RAND(\phi_{1_{min}}, \phi_{1_{max}})$
  $\phi_2 \leftarrow RAND(\phi_{2_{min}}, \phi_{2_{max}})$
  $\delta \leftarrow RAND(\delta_{min}, \delta_{max})$
  $T_{Path} \leftarrow GET\_PATH\_TRANS(\sigma)$
  $T_{Head} \leftarrow CALC\_CUTTING\_TRANS(\phi_1, \phi_2, \delta)$
  $T_{Perm} \leftarrow CALC\_PERMUTATION(\beta, \alpha, ZR)$
  $T_{Path} \leftarrow T_{Path} \cdot T_{Perm} \cdot T_{Head}^{-1}$
  $q_I \leftarrow INVKINE(T_{PATH})$
  return $[q_I, \phi_1, \phi_2, \delta], [\sigma, \beta, \alpha, Z_R, \phi_1, \phi_2, \delta]$
end function

Algorithm 3. Pseudocode representation of the GENERATE_SAMPLE_ON_PATH function. This function is used to sample robot configurations which are guaranteed to be on the path and within the constraints defined by the process.

An implementation of the motion planning algorithm has been developed in MATLAB and the robot simulation software V-REP [19]. Furthermore, the Robotics toolbox from Peter Corke [20] has been used to provide additional functionality. V-REP provides a convenient and flexible simulation environment in which results can be presented. Furthermore, V-REP has a Matlab interface and integrated collision detection. however, the main drawback of this using V-REP is that the communication bandwidth between V-REP and Matlab is quite limited, which entails that collision detection becomes very time consuming when many configurations needs to be checked.

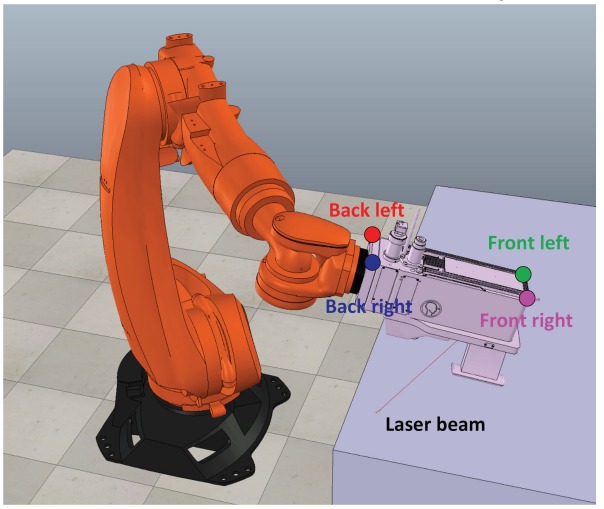

Fig. 5 An image showing the simulation environment V-REP with the industrial robot equipped with scanner mirrors.

Type	Iterations	# Edges	# Con	Avg. cost	Time[s]
R_{d_s}	100	350	215	2,06	236,350
	500	3033	1217	1,58	1287,43
	1000	5618	1216	1,89	2397,03
	2000	11195	1277	1,38	4731,03
	5000	27145	1264	1,37	12900,83
R_{d_q}	100	395	323	1,80	202,090
	500	2865	1038	2,55	1739,82
	1000	5458	1158	2,16	3263,11
	2000	11069	1275	1,42	6097,64
	5000	27613	1261	1,47	14181,42
$R_{d_{qW}}$	100	388	203	1,99	203,880
	500	2674	976	1,92	2024,55
	1000	5702	1257	2,04	3564,89
	2000	11069	1224	1,88	5689,80
	5000	27315	1257	1.69	17297.58
$R_{d_{rqW}}$	100	372	189	2,27	212,200
	500	2796	888	1,88	1813,22
	1000	5659	819	2,55	4198,29
	2000	11326	997	2,04	8390,34

Table 1. Simulation results of the implemented algorithms. The three subscripts denote the used distance metric.

3. Results

A series of simulation trials have been conducted on a contour in the shape of a line as can be seen in Fig. 5. The end-effector needs to traverse this path with a speed of 0.1 m/s, which is a stable speed for conducting RFC on 0.5mm stainless steel [1]. The entry point of the contour lies on the point (x, y, z) [1.1m, -0.85m,1m] and has a plane normal pointing in the z direction. It ends in [1.1m, 0.25m,1m]. To illustrate the movements of the robot cutting head the positions of four front facing corners are plotted. Furthermore, the curve that is traversed by the laser beam will be marked by red. A set of 40 randomly generated entry and exit configurations have been generated. Out of these all entry configurations and 32 end configurations were collision free. This yields a total of 1280 possible paths exists between the entry and exit points. To determine the connectivity, the described algorithm was used with the four distance measures. 6 runs with each distance measure was conducted to determine the necessary amount of iterations. A K value of 15 and an L value of 8 was used for all experiments (15 nearest nodes are considered, but a maximum of 8 are added). The CONNECT_UNCONNECTED() variable P was set to the number of iterations 10 for all experiments. The results from these experiments can be seen from table 1. It should be noted that all runs are independent e.g. the results from a 500 iteration run is not a continuation of the 00 iteration run. From this table it can be seen that generally, as the number of iterations go up, the average distance measure goes down. Furthermore, more connections are found, but the processing time also increases significantly. It is also seen that the distance metrics all finds more than 1250 of the 1280 paths. A plot of three connected path can be seen from Fig. 6. Where A shows the path from entry configuration 13 to exit configuration 30, B the path from entry configuration 23 to exit configuration 11 and C shows the path between entry configuration 1 and exit configuration 3. The curves indicate the position of four of the corners of the cutting head when moving from the entry configuration to the exit configuration. The red line shows the beam path, which follows the reference trajectory. From this it can be seen that the resulting curves all follow the reference trajectory. However, it is also seen that in

some cases, e.g. A the resulting cutting head curve is not very smooth. This is due to fact that only joint velocity limits are taken into account.

4. Conclusion and discussion

In this paper an algorithm was presented which calculates a map in the redundancy space of an industrial robot with an attached scanner head. By including a path variable directly in the sample vector samples were generated on a previously provided end effector path. By using these generated samples, it is possible to evaluate several connected entry and exit configurations. A test was provided for finding connections on a straight line and the presented algorithm was capable of finding 1277 /1280 connections in 2000 iterations. The implementation did

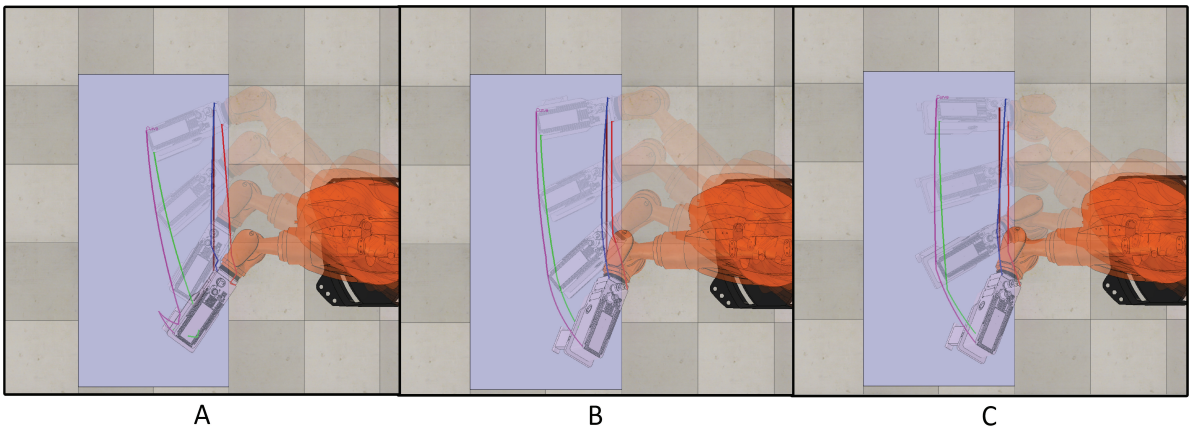

Fig. 6 Shows three possible connections between end point configurations for the robot over the end effector path indicated by the red line

however suffer from the limited communication bandwidth between the implemented collision checker and the Matlab implementation of the described algorithm. This entailed that the running time for these 2000 iterations were more than an hour. The resulting matrix of connections can be used as a basis for task sequencing in a GTSP framework. Even though the chosen sample vector does entail that a continuous position profile can be obtained while traversing the processing contour and that bounds can be put on the joint speeds it cannot guarantee that the resulting path has a continuous velocity profile. To ensure that a continuous velocity profile is obtained the resulting path should be smoothed by using commonly used approaches such as cubic polynomials. The drawback would however be that the resulting path from this operation could move the robot into collisions or singularities. Another approach would be to include the derivate of the sample vector parameters. This would ensure that a continuous velocity is obtained for all nodes. The drawback of this approach would however be that the number of states in the sample vector would double.

References

[1] S. L. Villumsen, M. Kristiansen, Physics Procedia, Volume 83, 2016, pp. 1206-1216, ISSN 1875-3892,
[2] A. Mahrle, M. Luetke, E. Beyer, Proceedings of the Institution of Mechanical Engineering Part C-Journal 224 (2010) pp. 1007–1018.
[3] H. Park, G. Lee, in: Science and Technology, 2005. KORUS 2005. Proceedings. pp. 625–629.
[4] M. F. Zaeh, J. Moesl, J. Musiol, F. Oefele, Physics Procedia 5, Part A (2010) Proceedings of the LANE 2010. pp. 19 – 33.
[5] G. Reinhart, U. Munzert, W. Vogl, CIRP Annals - Manufacturing Technology 57 (2008) pp. 37 – 40.
[6] J. Hatwig, P. Minnerup, M. Zaeh, G. Reinhart, in: Mechatronics and Automation (ICMA), 2012 International Conference on, pp. 1323–1328.
[7] J. Hatwig, G. Reinhart, M. F. Zaeh, Production Engineering. Research and Development (1 July 2010).
[8] J. Hatwig, G. Reinhart, M. F. Zaeh, Production Engineering 4 (2010) 327–332.
[9] A. Kovács : A. Press (Ed.), 23rd International Conference on Automated Planning and Scheduling (ICAPS).
[10] G. Erdős, Z. Kemény, A. Kovács, J. J. Váncza, Procedia CIRP 7 (2013) pp.222–227.

[11] A. Kovács , International Journal of Production Research (2015) 1–15.
[12] J. Stemmann, R. Zunke, in: Operations Research Proceedings, Operations Research Proceedings, Springer Berlin Heidelberg, 2005.
[13] G. Oriolo, M. Ottavi, M. Vendittelli, in: Intelligent Robots and Systems, 2002. IEEE/RSJ, volume 2, IEEE, pp. 1657–1662.
[14] G. Oriolo, C. Mongillo, in: Robotics and Automation, 2005. ICRA 2005., IEEE, pp. 2154–2160.
[15] D. Berenson, S. S. Srinivasa, D. Ferguson, J. J. Kuffner, in: Robotics and Automation, 2009. ICRA'09. IEEE, pp. 625–632.
[16] S. M. LaValle, Planning Algorithms, Cambridge University Press, New York, NY, USA, 2006.
[17] L. E. Kavraki, P. Švestka, J.-C. Latombe, M. H. Overmars, Robotics and Automation, IEEE Transactions on 12 (1996) 566–580.
[18] D. B. Johnson, Journal of the ACM (JACM) 20 (1973) 385–388.
[19] M. F. E. Rohmer, S. P. N. Singh, in: Proc. of The International Conference on Intelligent Robots and Systems (IROS).
[20] P. I. Corke, Robotics, Vision & Control: Fundamental Algorithms in Matlab, Springer, 2011.
[21] S. L. Villumsen, M. Kristiansen, Physics Procedia - NOLAMP 15 78 (2015) 89 – 98.