A Framework for Task Sequencing for Redundant Robotic Remote Laser Processing Equipment Based on Redundancy Space Sampling

Villumsen, Sigurd Lazic; Kristiansen, Morten

Published in:
Procedia Manufacturing

DOI (link to publication from Publisher):
10.1016/j.promfg.2017.07.320

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) 1826 – 1836

Procedia MANUFACTURING

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

A Framework for Task Sequencing for Redundant Robotic Remote Laser Processing Equipment Based on Redundancy Space Sampling

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

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

Abstract

This paper presents a framework for task sequencing of redundant remote laser processing equipment. The task sequencing algorithm is based on generating sets of robot configuration samples on all processing contours. The sequencing is done by using a solver for the equality constrained generalized traveling salesman problem (E- GTSP) to find the shortest path between these sets, while taking process constraints, robot redundancy and cluster connectivity into consideration. The algorithm was implemented in Matlab with interfaces to the robot simulation program V-rep and the GLKH E-GTSP solver. A test was carried out with 38 contours on a work piece with a set of angled plateaus.

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

1. Introduction and state of the art

The introduction of high quality, high power fiber guided laser sources, such as fiber lasers and disk lasers have started a development from traditional laser processing towards remote laser processing [1]. This transition is already seen in the automotive industry for remote laser welding (RLW). This is because RLW has the potential to reduce cycle times significantly [2] when compared to e.g. resistance spot welding. Mechanical systems for remote laser processing are however often complicated and are often reported to consist of a laser scanner processing head with 3 degrees of freedom (DOF) mounted on an industrial robot with 6 DOF [1,3]. This trend towards more

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

doi:10.1016/j.promfg.2017.07.320

advanced processing machinery has ensured that the programming of such systems needs to move from on-line programming to CAD/CAM based technologies for path generation and task sequencing. Such systems have already been proven to reduce the cycle times even further for remote laser welding RLW [4–7].

In [8] an automated remote laser welding system for conventional optics is presented. By using augmented-reality to display a user interface directly on a work piece, a set of weld seams can be generated. By using a simple traveling salesman problem (TSP) solver an initial sequence is generated. This sequence is then improved by modifying the laser incident angle. The TSP solver is based on Cartesian space distance, and the modification algorithm is based on the position of the cutting head in Cartesian space. In [5,9–11] a planning system is presented for both laser remote ablation cutting (RAC) and laser welding. The system works by identifying regions around each task that enables the cutting head to reach the cutting kerf. These are circular regions denoted scan circles. A suitable robot path is then found within these scan circles and a corresponding scanner head path is then found. The scan circles are connected by straight lines and paths are connected by using the traveling salesman algorithm. The algorithm enables the use of processes that requires multiple passes such as the RAC 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. 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 defining clusters of robot configurations that allow for welding of the seam in a task. One cluster is defined for each task and a GTSP algorithm connects these clusters by the shortest path. This algorithm utilizes the redundancy of the robot and scanner mirrors to minimize the processing time. The main drawback of the GTSP approach is however that it requires the start and end configuration of a task to be the same. An integrated approach to rough cut path planning and task sequencing for remote laser welding with scanner mirrors is presented in [7,13,14]. Access volumes are defined around entry and endpoints of weld seams with the shape of truncated cones. These truncated cones are defined by the allowable incident angle of the laser beam and by the allowable focus range. By using a tabu search algorithm combined with a path planning heuristic the framework computes close-to-optimal scanner path for each candidate task sequence. Redundancy is handled by fixing a joint to a ”mid-range” value.

In this paper a framework for sequencing of remote laser processing tasks will be presented. The framework will find a task sequence with associated robot and scanner cutting head configurations that minimizes the cycle time of a given work piece. Unlike the current planners, the minimization strategy is developed to take the redundancy of the mechanical system, open and closed processing contours and the process constraints of the laser process itself into account. Furthermore, the cycle time is evaluated by using a heuristic for robot reconfiguration time instead of Cartesian distance.

2. Description of Sequencing Problem

Before the proposed algorithm is presented the following section will give a more detailed description of the sequencing problem. A laser process that needs to be conducted on a work piece will, in the following, be denoted as a task: $\mathbf{t}$. The term task should be seen as a common term for welding, cutting and other continuous laser processes. If there are N tasks $\mathbf{t}_1 ... \mathbf{t}_N$ the sequence of tasks S can be defined by $S = (\boldsymbol{t}_{P(1)} ... \boldsymbol{t}_{P(N)})$. Where P is a vector containing a permutation of the numbers 1 to N. Then, if the processing time of a task is denoted by d_{t_n}, then the total processing time $d_{Process}(P)$ for a given sequence P can be calculated for all tasks by: $d_{Process}(P) = \sum_{i=1}^{N} d_{t_i}$. Furthermore, if two tasks $\mathbf{t}_a$ and $\mathbf{t}_b$ needs to be processed in sequence, then the mechanical system needs to move the laser beam from $\mathbf{t}_a$ to $\mathbf{t}_b$. If the time of such a reconfiguration is denoted $d_{t_b}^{t_a}$ then the reconfiguration time $d_{Rec}(P)$ involved in processing all tasks in a sequence P can be calculated by: $d_{Rec} = \sum_{i=P(1)}^{P(N-1)} \left(d_{t_i}^{t_{i+1}} \right) + d_{t_{P(N)}}^{t_{P(1)}}$. The total processing time d_{Total} for the work piece can then be calculated as: $d_{Total} = d_{Process} + d_{Rec}(P)$. If a sequence P^* yields the lowest cycle time it will in the following be denoted as the optimum sequence. P^* can be described as the minimizing sequence $P^* = \underset{P}{\arg\min}\, d_{Total}(P)$.

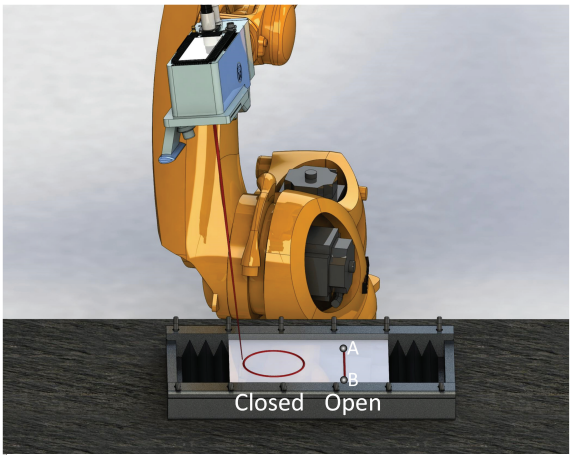

Fig. 1. A KUKA Quantec KR 120 R2500 pro mounted with an Arges remote welding elephant head. On a closed contour processing can start at any point. On an open contour processing can start either in point A or B.

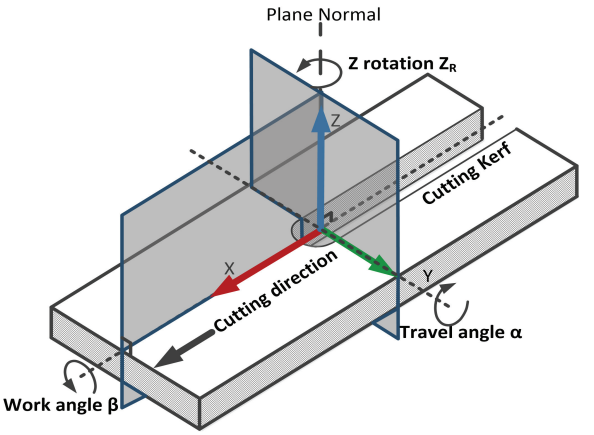

Fig. 2. Shows the frame assignment in the cutting kerf. Notice the three degrees of rotation around the fixed coordinate frames.

However, as $d_{Process}$ only depends on the processing speed and the length of the contours and not the task sequence the task of minimizing d_{Total}(P) is equivalent to minimizing $d_{Rec}(P)$.

$$P^* = \underset{P}{arg\,min}\, d_{Rec}(P) \tag{1}$$

This minimization depends on several factors, the geometric properties of the tasks, the limitations of the laser process itself and the mechanical positioning system. In the following these factors will be discussed.

2.1. Geometric task properties

It will in the following be assumed that all tasks have an associated contour $\boldsymbol{c}$ that defines the geometric region that needs to be processed. A contour $\boldsymbol{c_n}(\sigma_n)$ can be described by : $\boldsymbol{c_n}(\sigma_n) \epsilon SE(3) \forall \sigma \in [0,1]$, Where σ_n is a normalized path variable such that $\sigma_n = 0$ denotes the entry point of the contour, the point where the robot starts processing, and $\sigma_n = 1$ denotes the exit point of the contour, the point where the robot stops processing. If $\boldsymbol{c_n}(\sigma_n = 0) = \boldsymbol{c_n}(\sigma_n = 1)$ then the contour is categorized as being closed and if $\boldsymbol{c_n}(\sigma_n = 0) \neq \boldsymbol{c_n}(\sigma_n = 1)$ the contour is defined as being open. Fig. 1 shows two examples of this. When considering open contours, it can be seen that the process can start in point A or B and end in B or A respectively. For closed contours the entry point $\boldsymbol{c_n}(\sigma = 0)$ can be chosen freely over the entire contour. The exit point is coincident with the entry point as the robot needs finish the process in the same place where it started for a closed contour. To be able to analyses the system it is necessary to define a method for assigning frames in the cutting kerf. This is done as seen on Fig. 2. From this figure it is seen that the z axis is defined along the normal vector of the work piece, the x axis points in the cutting direction. Finally, the y axis is assigned to yield a right hand coordinate system. Three angles are defined: A work angle β, a travel angle α and a rotation around the z axis, Z_R.

2.2. Process limitations

The quality and stability of the conducted process depends on many parameters such as velocity, laser power and beam diameter. However, when considering scheduling, the allowable incident angle of the laser beam is of very high importance as it directly determines to what extent the beam can be moved by angling the laser cutting head or to what extent a laser scanner head can deflect the beam. As the deflecting mirrors on scanning cutting heads are extremely fast when compared to conventional axes, large allowable incident angles will therefore entail fast

repositioning times. The maximum deviation for these angles depend on the process, for RFC it is in [15] found to be ±6° when considering α and β. For remote laser welding it has been reported that inclination angles between 0◦ and 20° has no significant impact on the weld quality [16]. There is no Z_R limit if it is assumed that a round beam is used. This means that all points on the contour C can be described by a location given in x, y, z coordinates, and an angle interval α, β, Z_R within which high quality processing can be conducted. If the dimensionality of point p on a contour $\boldsymbol{c_n}(\sigma_n)$ is denoted M, $p \epsilon \mathcal{R}^M$, then M is 3.

2.3. Mechanical process equipment and robot redundancy

If the laser processing machinery is considered a fixed base manipulator with N joints that needs to traverse the contours $\boldsymbol{c_n}(\sigma_1) - \boldsymbol{c_n}(\sigma_n)$ a joint space trajectory $\boldsymbol{q_n}$ can be defined for each $\boldsymbol{c_n}(\sigma_n)$.

$$\boldsymbol{c_n}(\sigma_n) = f(\boldsymbol{q_n}(\sigma_n)), \forall \sigma \in [0,1] \tag{2}$$

Where $f(q)$ is the forward kinematic equations of the manipulator. $\boldsymbol{q_n}(\sigma_n)$ is the joint space trajectory of the manipulator. If the joint variables associated with the industrial robot are labelled $\theta_1 - \theta_6$ then the robot joint 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 a 3 DOF cutting head is assumed and the joint variables associated with it are denoted φ_1 (Scanner mirror 1), φ_2 (Scanner mirror 2) and δ (Focus offset) then the cutting head joint vector q_C is defined to be: $q_C = [\varphi_1 \quad \varphi_2 \quad \delta]^T$. By combining these vectors a complete joint vector q is obtained: $q = \begin{bmatrix} q_I \\ q_C \end{bmatrix}$. If the dimensionality of the joint vector q is denoted L, $q \epsilon \mathcal{R}^L$. From this it is seen that the dimensionality, $L = 9$. With a joint vector containing 9 joint variables and a task space defined by only three variables, it is seen that the system possesses 6 degrees of redundancy. This redundancy means that all positions (x, y, z) on a contour C can be realized by an infinity of robot configurations.

2.4. Path sampling

An approach to sampling robot configurations that will make the laser beam touch the processing contour is presented in [17,18]. Here it is proposed that the joint vector q is decomposed into a base vector q_b and a redundancy vector q_r . By specifying values for the redundancy vector and using the inverse kinematics of the robot to calculate the values of q_b, a joint vector sample can be generated for a given point on the end effector path. This division will however not ensure process stability as the incident angle requirement is not taken into account. To generate compliant samples directly it has been chosen to define a homogeneous transformation matrix T that can be used to change the orientation of the kerf transformation. The transformation will rotate the frame coordinate system by a random quantity to generate α, β and Z_R angles as seen in Fig. 2. By sampling β, α and ZR within the limits of a given laser process all samples can be generated in compliance with the maximum angle deviation of the process. By combining this with the cutting head joint vector q_C a sampling vector encompassing all 6 degrees of redundancy is defined: $q_{PS} = [\beta \quad \alpha \quad Z_R \quad \varphi_1 \quad \varphi_2 \quad \delta]^T$. With this sampling strategy a robot configuration can be generated by sampling values for q_{PS} for a given frame. This can then be modified to yield a sample on the entire contour by appending the path variable σ: $q_S = [\sigma \quad \beta \quad \alpha \quad Z_R \quad \varphi_1 \quad \varphi_2 \quad \delta]^T$. By applying this sampling strategy repeatedly to e.g. point A and B in Fig. 1 two point clouds of possible robot configurations can be generated. All configurations will direct the laser beam to the corresponding point and be in compliance with the constraints on incident angle. In the same way, all possible points along a closed contour could be sampled to yield a large cluster. This is depicted in Fig. 3.

This point cloud representation does however implicitly indicate that solutions to the reconfiguration time between two tasks $d_{t_b}^{t_a}$ is also no longer unique if the reconfiguration time between two tasks depend on the configuration of the robot. Essentially, if the robot is redundant in terms of the task, the infinity of configurations directly translates to an infinity of reconfiguration times. By assuming that the robot conducts joint moves and that the joints operate with their maximum velocity over the entire movement the reconfiguration time can be

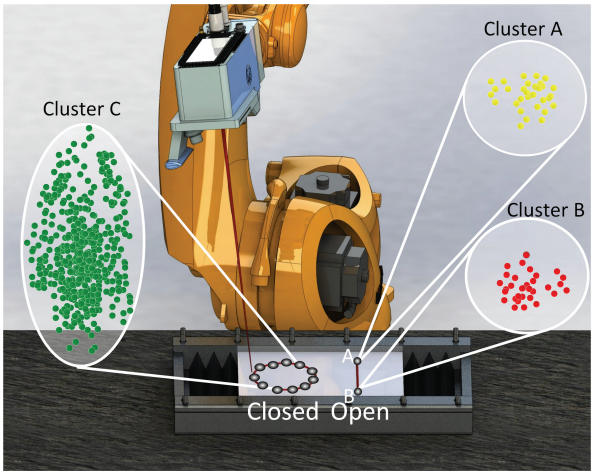

Fig. 3. By sampling the closed and open con- tour from Fig. 1 a set of joint space clusters can be generated. Notice that two clusters are generated for open contours.

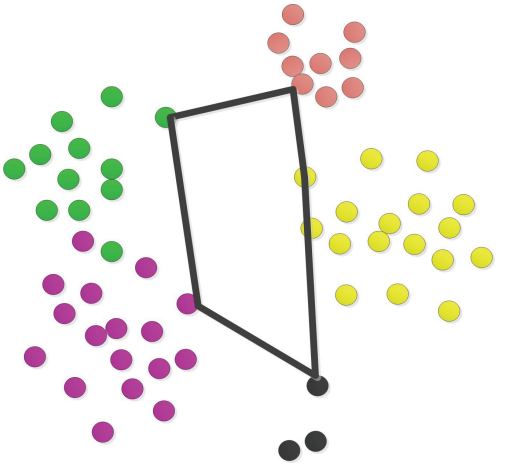

Fig. 4. An E-GTSP problem instance and an example of a solution that ensures that finds the shortest path between the clusters in such a way that each cluster is visited exactly once.

evaluated as the maximum joint angle divided by the maximum joint speed. This opens up for the possibility sequencing based on finding minimum repositioning times from robot configurations.

This means that to minimize equation 1 it is necessary to take the redundancy of the robotic repositioning system into account. This can, with some modifications be done by formulating the minimization problem as an equality constrained traveling salesman problem (E- GTSP). The Generalized traveling salesman (GTSP) can be considered an extension of the traveling salesman problem (TSP) where the set of vertices is partitioned into sets or clusters. The salesman then needs to visit, not every vertex, but every cluster of vertices at least once. The equality constrained generalized traveling salesman problem (E-GTSP) further specifies that each cluster should be visited exactly once. Such a problem can be seen from Fig. 4 where five clusters should be visited. Currently, one of the most powerful solvers for the E-GTSP problem / ATSP problem is the GLKH solver presented in [19]. This solver has been used to find high quality solutions to instances with as much as 17,180 clusters and 85,900 vertices. When considering the case of remote laser processing the set of possible entry and endpoint configurations of a contour can be seen as a point cloud in redundancy space. Each of these point clouds can then be seen as GTSP cluster and each configuration in one of these sets can be seen as a vertex. This entails that for each contour two clusters are formed, one representing the possible set of endpoint configurations and one representing the possible set of entry point configurations. This can be seen from Fig. 5. By using this representation, it is clear that by using a GTSP formulation an approximation to the shortest path between the allowable robot configurations can be obtained. However, algorithms capable of solving large E-GTSP problems generally works by iterative sequence improvements that does not guarantee the optimality of the found solution. Furthermore, as the robot configurations in the clusters are generated by a sample based approach it cannot be guaranteed that the clusters contain optimal robot configuration. This again means that a sample based approach cannot guarantee optimality and the solution will thus only be an approximation of P^*.

3. Proposed sequencing algorithm

To be able to solve the minimization problem described by equation 1 as an E-GTSP the problem is necessary to break it up into several steps. These steps are depicted on Fig. 6. From this figure it is seen that the algorithm is composed of 2 pre-processing steps and 3 algorithm steps. The 2 pre-processing steps will be described briefly in section 3.4. The three algorithm steps will be described below.

3.1. Step 1: Finding optimal entry points

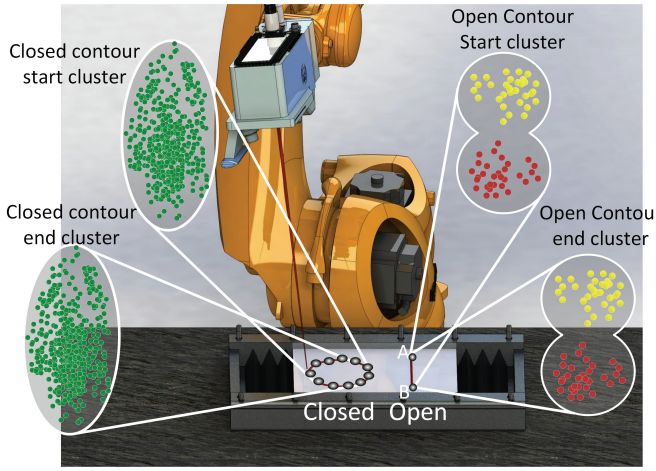

Fig. 5. An image showing the associated joint space clusters with each entry and end point configuration.

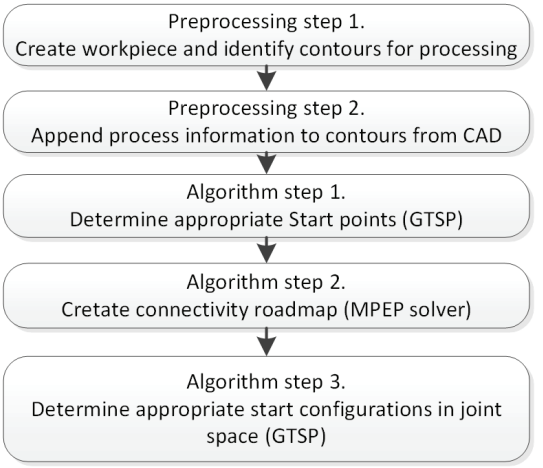

Fig. 6. An overview of the steps used in the generation of robot paths for the remote laser cutting setup.

In the first step of the algorithm the entry/exit points on the closed contours are found. When considering traditional laser cutting this is often done by considering the problem as a GTSP problem where a set of evenly spaced points along the contour are treated as nodes in a GTSP cluster [20]. By generating such clusters for all contours, and by using the Cartesian distance as a cost function an E-GTSP solver can be used to find a set of optimal start points. However, when considering the case of remote laser processing the head of the laser processing system is positioned up to 1 meter above the work piece. If the range of allowable incident angles are disregarded, the path that the robot needs to take should be seen as a projection of the curve in the direction of the plane normal. This entails that it has been chosen to use the same method as for traditional laser processing, however with the modification that the entry point sequencing will be based on the Cartesian distance of the projected contours. When completed this step yields a set of entry points on the contour given in the form of an offset to the path variable σ.

3.2. Step 2: Sampling and solving the MPEP problem

Now that a set of entry points have been found for each contour the task of determine robot configurations will be considered. This step concerns the task of sampling a set of poses that can be used as entry / exit point candidate configurations. By applying the sampling strategy described in section 2.4, a several configurations are generated for each start and endpoint. However, it is not necessarily possible to reconfigure the robot from a given entry configuration sample to a given exit configuration sample while processing the contour. To identify which exit configurations can be reached from a given entry configuration a connectivity matrix is created as described in [21]. A connectivity matrix R can be seen below.

$$R = \begin{array}{c} \\ q_{En_1} \\ q_{En_2} \\ q_{En_3} \\ \\ q_{En_L} \end{array} \begin{array}{c} \begin{array}{ccccc} q_{Ex_1} & q_{Ex_2} & q_{Ex_3} & & q_{Ex_I} \end{array} \\ \begin{pmatrix} c_(1,1) & c_(1,2) & c_(1,3) & \dots & c_(1,I) \\ c_(2,1) & c_(2,2) & c_(2,3) & \dots & c_(2,I) \\ c_(3,1) & c_(3,2) & c_(3,3) & \dots & c_(3,I) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_(L,1) & c_(L,2) & c_(L,3) & \dots & c_(L,I) \end{pmatrix} \end{array} \tag{3}$$

Where rows represent entry point configurations $q_{En_1} - q_{En_L}$ and columns represents exit point configurations $q_{Ex_1} - q_{Ex_I}$. The entries in R contains the reconfigurations cost between the two configurations while still traversing the contour. A very large integer value (999999) indicates that a given exit point configuration is unreachable from a given entry point configuration. This happens if the path planning algorithm cannot find a solution due to

e.g. velocity constraints or collisions. When completed this step yields a connectivity matrix for each of the processing contours along with a set associated of entry/exit point configurations.

3.3. Step 3: Configuration space sequencing

Now that a set of entry/exit point configurations for each contour have been found along with their connectivity matrix it is possible to determine the sequence of configurations can minimize equation 1. In section 2.4 it was discussed that a set of entry and exit configurations for a contour could be considered as two clusters, an entry cluster and an exit cluster. To ensure that the E-GTSP solver only finds solutions where a contours exit cluster is visited after its entry cluster a modification to the generated E-GTSP graph is necessary. This is done somewhat differently for closed and open contours. For closed contours the entry and exit points is coincident from a Cartesian point of view. In joint space, they are however different. If the cluster containing the entry point configurations is labelled C_{En} and the cluster containing the exit configurations is labelled C_{Ex} then it is clear that the only valid sequence is $C_{En} \rightarrow C_{Ex}$. If the set of all clusters on the work piece except C_{En} and C_{Ex} is labelled " C_1. C_N " then by modifying the weights of the graph it is possible to penalize any solution where C_E is not processed right after C_S.

$$\begin{array}{c} \\ C_1 - C_N \\ C_S \\ C_E \end{array} \begin{array}{c} \begin{array}{ccc} C_1 - C_N & C_S & C_E \end{array} \\ \begin{pmatrix} \sim & \sim & \infty \\ \infty & \sim & R \\ \sim & \infty & \sim \end{pmatrix} \end{array} \tag{4}$$

Where $\sim$ denotes that the cost of the reconfiguration remains unchanged. R denotes the connectivity matrix, but with the addition that all non-infinite entries are truncated to 0. This is done as these robot reconfigurations while conducting laser processing is considered "free" when minimizing equation 1. The entries reading ∞ are used to suppress movements violating the start/endpoint constraint. A cluster in $C_1 - C_N$ would see an infinite cost to C_E as it is an exit cluster. Furthermore, a cluster in $C_1 - C_N$ sees a reconfiguration cost to C_S as it is a start cluster. C_S sees an infinite cost of going to a cluster in $C_1 - C_N$ as it needs to go to its exit cluster first. In practice entries containing ∞ are replaced by a large integer (999999).

For open contours the entry and exit points can no longer be assumed to be coincident neither in Cartesian space nor in joint space. Furthermore, to ensure that processing can be conducted in either direction a more complicated cluster structure needs to be formed. Two sub clusters A and B are defined containing joint samples from the two ends of the open contour (see Fig. 2.4). However, they cannot be labelled as entry or exit clusters as processing can begin in either one. To accommodate this an entry and exit cluster is defined both containing all configurations associated with point A and B. If the entry cluster is labelled C_S and the exit cluster is labelled C_E. Then the sub clusters containing the samples associated with point A can be labelled C_{S_A} and C_{E_A} and the sub cluster containing the samples associated with point B C_{S_B} and C_{E_B}. By applying similar suppression rules as described in the previous section the following matrix is obtained.

$$\begin{array}{c} \\ C_1 - C_N \\ C_S \begin{cases} C_{S_A} \\ C_{S_B} \end{cases} \\ C_E \begin{cases} C_{E_A} \\ C_{E_B} \end{cases} \end{array} \begin{array}{c} \begin{array}{ccccc} & \overbrace{}^{C_S} & & \overbrace{}^{C_E} & \\ C_1 - C_N & C_{S_A} & C_{S_B} & C_{E_A} & C_{E_B} \end{array} \\ \begin{pmatrix} \sim & \sim & \sim & \infty & \infty \\ \infty & \sim & \infty & \infty & R \\ \infty & \infty & \sim & R^T & \infty \\ \sim & \infty & \infty & \sim & \infty \\ \sim & \infty & \infty & \infty & \sim \end{pmatrix} \end{array} \tag{6}$$

This does however entail that the set of configurations doubles for open contours. As cutting mainly involves closed contours this is mainly a drawback when considering e.g. laser welding.

3.4. Implementation

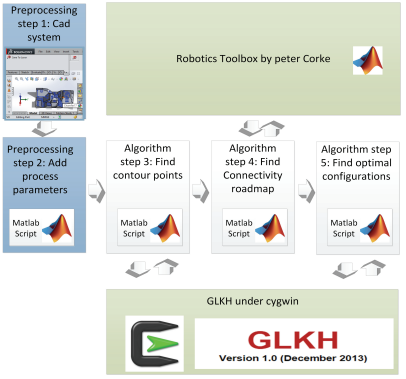

Fig. 7. The structure of the implemented framework. Blue: pre-processing steps. White: Scheduling steps. Green: toolboxes and libraries.

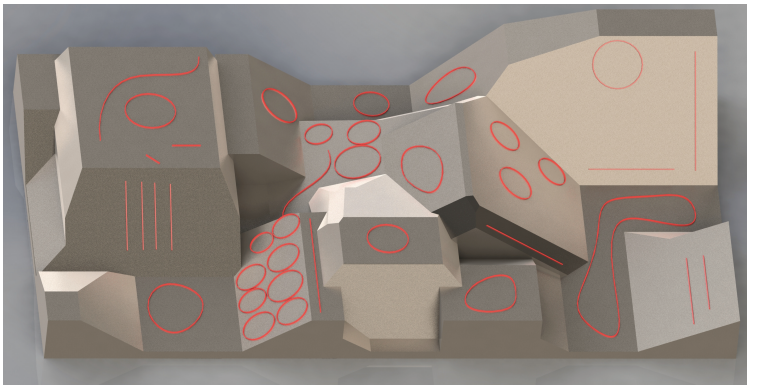

Fig. 8. The chosen test case for the simulation. A total of 38 cuts have been laid out on a series of surfaces with varying angles.

The implementation of the presented algorithm has primarily been conducted in MATLAB and C#. Fig. 7 shows an overview of the implementation. From this figure it is seen that it follows the 5 step procedure described in Fig. 6. When regarding **pre-processing step 1**, it generally concerns the task of generating a set of processing contours on a work piece. To generate these contours a plug-in has been made for solidworks that can convert solidworks sketches into contours as defined in section 2. In **pre-processing step 2** a contour file parser has been implemented in Matlab. The contour files can then be read and processing data, e.g. velocities and maximum allowable angle of incidence can be appended to the contours. In **algorithm step 1 and 3** the main task is the task of minimizing an E-GTSP problem. As the task of implementing an efficient E-GTSP is very time consuming it has been chosen to base the implementation on existing solvers. It has been chosen to use the open source GLKH developed by Keld Helsgaun [19] as it is currently one of the most effective solvers available. Furthermore, it supports files defined by the TSPLIB file format, which entails that the GLKH solver can be replaced by other methods supporting this file format. In **algorithm step 2** a Matlab script samples a set of entry and exit point poses which is then fed to the algorithm described in section 3.2. The implementation of this algorithm is also based on Matlab with an interface to the robot simulation tool V-rep [22] and the Robotics toolbox from Peter Corke [23].

4. Results

In the following section the results obtained by using the proposed algorithm will be presented. The results will be based on the test case seen on Fig. 8. This figure shows 38 contours, samples with a spacing of 1mm, that needs to be sequenced. It has been chosen to process all contours by remote fusion cutting (RFC) with a speed of 0.1 m/s. which is an appropriate processing speed for RFC cutting 0.5mm stainless steel [15]. The total footprint of the work piece is 1400 mm X 800 mm. For this sequence it has been chosen to use the maximum incident angle limits of 6 degrees found in [15]. For the start point sequencing 100 points have been sampled around the edge of the contour in a distance of 500 mm in the direction of the plane normal, as described in section 3.1. Furthermore, 100 points per entry/ exit point has been chosen for the joint space sequencing. The GLKH solver was run for 3000 iterations both for entry point sequencing and configuration sequencing. An optimized sequence was found by following the **three** steps described in section 3.

In the **first** step the entry point of each contour is found. The obtained cost function of the GTSP for Cartesian sampling can be seen from Fig. 9. From this it is seen that the distance decreases as a function of the performed iterations. It has been chosen to stop the sequencing algorithm after 3000 iterations to obtain a good route in a reasonable time. From the shape of the plot the optimization algorithm also seems to have reached a position with slow improvements. The results from the two iterations can be seen from figure 10 where the green line indicates the

obtained path from the entry point sampling strategy. The red contours are the original contours on the surface of the contour. Blue indicates contours projected 500 mm above the surface of the work piece.

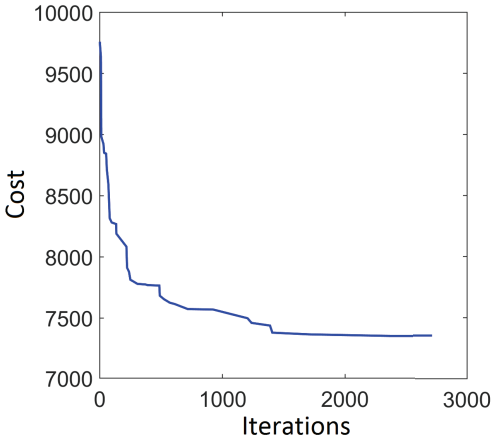

Fig. 9. The tour cost as a function of the iterations when running the generalized traveling salesman solver.

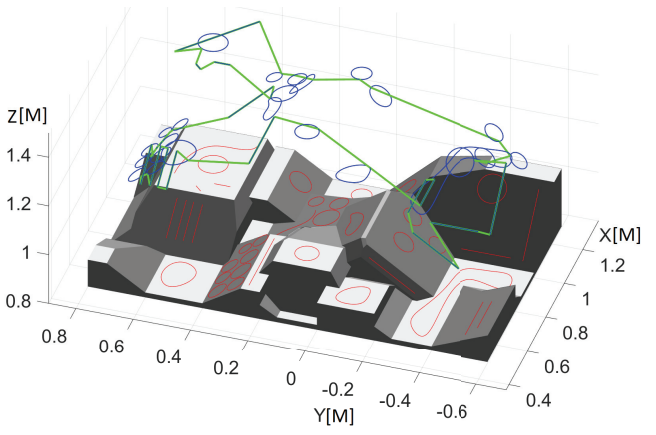

Fig. 10. The obtained results after 3000 iterations. It is seen that a reasonable path is found. It is however difficult to evaluate the optimality of the path due to the magnitude of the solution space

In the **second** step a connectivity map was created for each contour based on the algorithm described in [21]. It has been chosen to run 2000 iterations of the path planning algorithm per contour. The maximum connections per node has been set to 12. Table 1 lists the number of generated paths per node from the 2000 iterations. It is seen that the number of paths range from 86 to 9771 found paths. By creating a combined matrix composed of all the above connections with the modifications described in section 3.3 a TSPLIB file can be created as described in 3.4. As the set of contours contains 24 closed contours and 14 open contour the number of nodes in the searched graph can be calculated to be 10400 (24 open contours with 200 configurations (100 entry configurations and 100 exit configurations) + 14 open contours with 400 configurations (100 Entry configurations and 100 exit configurations which needs to be doubled as described in section 3.3). This entails that the number of nodes is still well below the limit of the GTSP instances that are being treated by the scientific community [19].

In the **third** step the TSPLIB file is then processed by the GLKH solver and a sequence with associated configurations is found. When this is done the paths seen in Fig. 11 is obtained. This figure shows the resulting cutting sequence. The magenta lines indicate a robot intertask reposition and a black line entails a repositioning that is conducted while traversing the contour. This entails that all black lines can be seen as free moves as they are conducted while processing the contours. The ratio between the black and magenta lines can be used to describe how much movement can be conducted on while processing. This ratio can be seen from table 2.

Table 2. Table showing the number of connected paths when finding connected entry and exit points over the 38 contours on the test case.

Metric	Total movement	Intertask movement	Intertask %
Cartesian distance	9.5422[m]	4.2615 [m]	44.6%
Reconfiguration time	13.5657[s]	1.3373 [s]	9.8%

From this table it is seen that when considering the Cartesian distance, the scheduling task is capable of conducting approximately 55% of the total movement while processing the laser processing the contour. It is however also seen that when the metric is changed to reconfiguration time this changes to approximately 90 %. As the optimization has been conducted on the reconfiguration time this is not surprising. The discrepancy between the two metrics generally originates from the scanner head of the robot system. When reconfiguration is conducted by the scanner head and not the robot the reconfiguration time can almost be truncated to 0 for small distances.

5. Conclusion and discussion

A framework for task sequencing has been presented in this paper. The sequencing algorithm worked by minimizing the reconfiguration time between tasks to reduce the processing time of work pieces. By utilizing redundancy, the sequencing algorithm allowed the mechanical system, to reconfigure itself while processing a given task. This reconfiguration ensures that movements between tasks can be minimized. The framework was divided into 5 steps out of which the first two are pre-processing steps which generated the contours for processing. In the third step a state of the art E-GTSP solver is used to find approximations to optimal start points. Then, in step four a motion planning algorithm determines the connectivity of entry / exit point configuration candidates. Finally, in step five the same E-GTSP solver is used for sequencing these entry point candidates. The implemented Matlab algorithm can generate an optimized path for 38 contours in approximately 15 hours. This path uses approximately

#	Paths	#	Paths	#	Paths	#	Paths
1	3237	11	8509	21	8206	31	8051
2	433	12	7302	22	9272	32	311
3	7212	13	9771	23	86	33	955
4	7448	14	402	24	3609	34	2988
5	531	15	2053	25	577	35	7690
6	4453	16	1547	26	2517	36	8058
7	877	17	4489	27	7710	37	4035
8	7789	18	1344	28	1365	38	7440
9	390	19	943	29	1241		
10	7425	20	1289	30	1241		

Table 1. Table showing the number of connected paths when finding connected entry and exit points over the 38 contours on the test case.

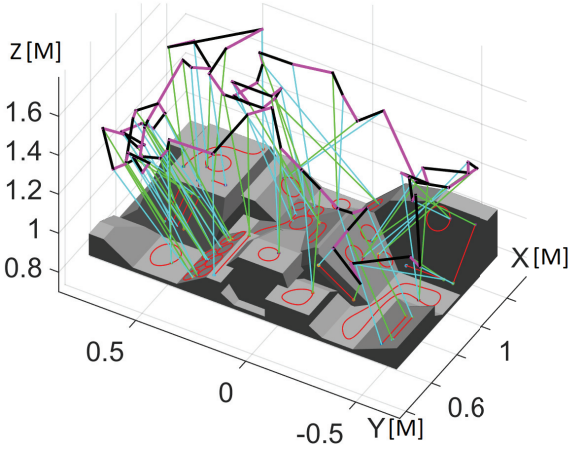

Fig 11. The sequence obtained by the scheduling algorithm. Cyan: markings relate the robot end effector position to an entry point on a contour. Green lines: links an exit point to its corresponding robot end effector position. Magenta lines: indicate a robot intertask reposition and a black line entails a repositioning while processing a contour

45 % of the movement in inter task movements. However, from a reconfiguration time perspective quantity only 9.8 % as the fast scanner mirrors of the remote cutting system are used for a majority of the intertask moves.

The proposed framework for task sequencing is based on a sample based approach to the task of finding appropriate entry and exit points. The main difficulty of the sequencing problem is however the up to 6 degrees of redundancy of the mechanical positioning system. This entails that if the sample space should be uniformly sampled with e.g. 10 samples per axis a total of $2 \cdot 10^6$ samples would be required per closed contour. This number of nodes far exceeds the capabilities of state-of-the-art GTSP solvers and the capabilities of the path planning algorithm on which this task framework relies. By combining the proposed planning framework with a local search algorithm, that searches in the null space of the redundant robot, the obtained solutions could potentially be improved.

References

[1] M. F. Zaeh, J. Moesl, J. Musiol, F. Oefele, Physics Procedia 5, Part A (2010) 19 – 33. Proceedings of the LANE 2010, Part 1.
[2] H. Park, G. Lee, in: Science and Technology, 2005. KORUS 2005. Proceedings. pp. 625–629.
[3] M. F. Zaeh, J. Hatwig, J. Musiol, O. Roesch, G. Reinhart, Robotics (ISR), 6th German Conference on Robotics (ROBOTIK) (2010) pp. 1 –8.
[4] G. Erdős, Z. Kemény, A. Kovács, J. Váncza, Procedia CIRP 7 (2013) pp. 222–227.
[5] J. Hatwig, G. Reinhart, M. F. Zaeh, Production Engineering. Research and Development (1 July 2010).
[6] A. Kovács, in: Proceedings of the 2nd Workshop on Planning and Robotics (PlanRob2014), AAAI Press, Portsmouth, 2014, pp. 172–181.
[7] A. Kovács, in: A. Press (Ed.), 23rd International Conference on Automated Planning and Scheduling (ICAPS).
[8] G. Reinhart, U. Munzert, W. Vogl, CIRP Annals - Manufacturing Technology 57 (2008) pp. 37 – 40
[9] J. Hatwig, P. Minnerup, M. Zaeh, G. Reinhart, in: Mechatronics and Automation (ICMA), 2012 International Conference on, pp. 1323–1328.
[10] J. Musiol, M. Luetke, M. Schweier, J. Hatwig, A. Wetzig, E. Beyer, M. F. Zaeh, in: Proc. SPIE, volume 8239,

[11] J. Hatwig, G. Reinhart, M. F. Zaeh, Production Engineering 4 (2010) 327–332.
[12] J. Stemmann, R. Zunke, Operations Research Proceedings 2005, Springer , pp. 729–734.
[13] G. Erdős, Z. Kemény, A. Kovács, J. J. Váncza, Procedia CIRP 7 (2013) pp. 222–227.
[14] A. Kovács, International Journal of Production Research (2015) pp. 1–15.
[15] S. Villumsen, M. Kristiansen, Physics Procedia - NOLAMP 15 78 (2015) pp. 89 – 98.
[16] F. Oefele, J. Musiol, M. Zaeh, et al., ICALEO 2008, Temecula, USA.
[17] G. Oriolo, M. Ottavi, M. Vendittelli, in: Intelligent Robots and Systems, 2002. IEEE/RSJ pp. 1657–1662.
[18] G. Oriolo, C. Mongillo, in: Robotics and Automation, 2005. ICRA 2005., IEEE, pp. 2154–2160.
[19] K. Helsgaun, Mathematical Programming Computation 7 (2015) pp. 269–287.
[20] R. Dewil, P. Vansteenwegen, D. Cattrysse, M. Laguna, T. Vossen, International Journal of Production Research 53 (2015) pp. 1761–1776.
[21] S. L. Villumsen, M. Kristiansen: FAIM 2017.
[22] M. F. E. Rohmer, S. P. N. Singh, in: Proc. of The International Conference on Intelligent Robots and Systems (IROS).
[23] P. I. Corke, Robotics, Vision & Control: Fundamental Algorithms in Matlab, Springer, 2011.