Heuristic Solution for Constrained 7-DOF Motion Planning in 3D Scanning Application

Theodor Borangiu[a], Alexandru Dumitrache[a], Anamaria Dogar[a]

[a] *Centre for Research and Training in Industrial Control Robotics and Materials Engineering University Politehnica of Bucharest, RO*

Abstract

This article proposes a heuristic algorithm for near-optimal handling of the redundancy in robot mechanisms, applied to a 3D scanning platform consisting on a 6-DOF articulated robot arm which moves a laser sensor around an workpiece placed on a rotary table. The proposed method handles both static constraints, like avoiding joint limits, kinematic singularities or collisions, modelled using a configuration map viewed as a grayscale image, and dynamic constraints like velocity and acceleration. The algorithm is compared with Dijkstra algorithm, which gives the optimal solution, but is very slow, and with two other heuristics which are very fast, but give suboptimal solutions.

Key words: motion planning, inverse kinematics, 7-DOF manipulator, 3D laser scanning
PACS: 45.40.Ln, 45.40.Bb

1. Introduction

This work is part of a project whose goal is to develop a 3D laser scanner system by using a 6-DOF vertical articulated robot arm to move a triangulation-based laser probe around the object of interest, which is placed on a rotary table. The manipulator has a spherical working envelope with a radius of 650 mm and the laser probe is able to measure distances from 70 to 250 millimetres with an accuracy of $30\,\mu\mathrm{m}$. An overview of the scanning system is illustrated in Fig. 1. The robotic

Email addresses: borangiu@cimr.pub.ro (Theodor Borangiu), alex@cimr.pub.ro (Alexandru Dumitrache), dogar@cimr.pub.ro (Anamaria Dogar)
URL: www.cimr.pub.ro (Theodor Borangiu)

arm will move around the workpiece being analyzed using computer-generated adaptive scanning paths, which are computed in real-time while the scanning system is discovering the features of the object. The resulting 3D model will be used for reproduction of the scanned parts on a 4-axis CNC milling machine.

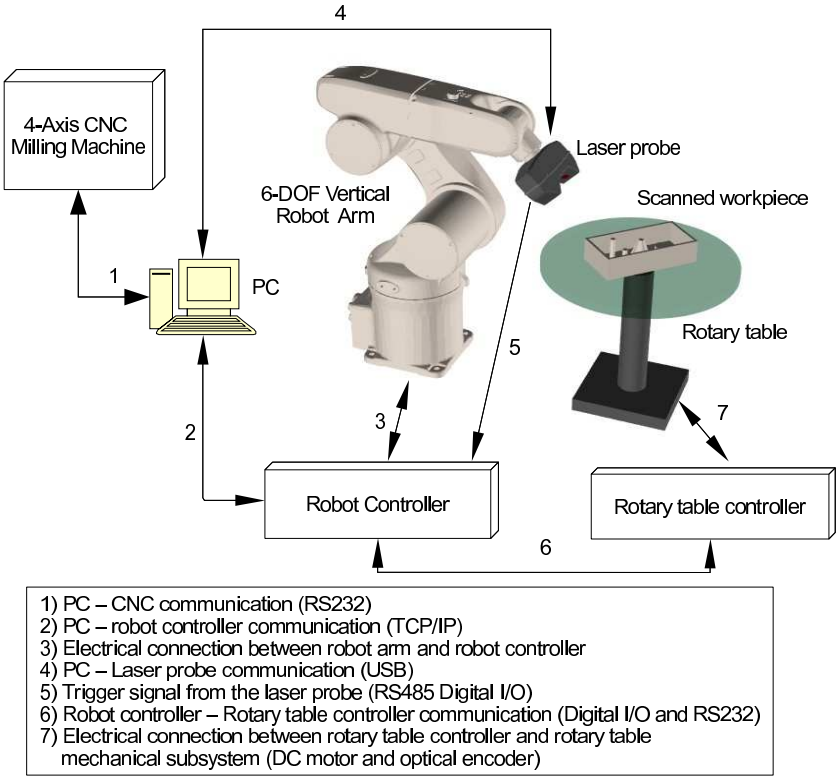


Figure 1: Main components of the scanning system

Current solutions for 3D scanning use either straight line motions in Cartesian space, or simple rotary motions of the device holding the scanned object and/or the laser probe. Moreover, the 3D scanning software designed for coordinate measuring machines (CMMs) is not optimal for robot arms due to singularity issues and nonlinear kinematics. The current paper presents a flexible planning solution which uses kinematic redundancy to specify additional constraints, such as collision and singularity avoidance, keeping the robot far from its joints limits, while maintaining a smooth motion of the mechanism.

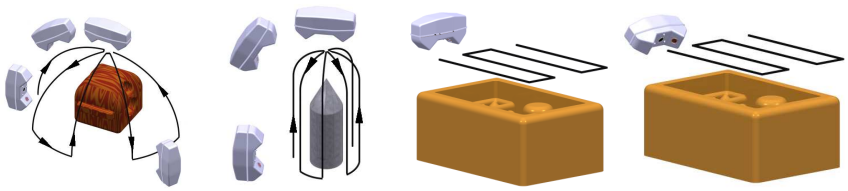
2. Problem description

The scanning trajectories are defined as motions in a reference frame attached to the workpiece. Some examples of proposed scanning trajectory patterns are given in Fig. 2a-d. The trajectories are discretized, i. e. described by a series of closely spaced locations (which include position and orientation), and the path between these locations is approximated by linear interpolation in Cartesian space. The position change between two discrete steps is executed as a straight line motion. For the orientation change, the rotation matrix between two discrete steps is represented in axis-angle format, and the linear interpolation is performed on the rotation angle, keeping the rotation axis fixed. The discretization is only used for planning the motion of the redundant mechanism, and is usually different from the time step used in the feedback loop of the robot and rotary table controllers, which also perform the resampling computations required for following the planned motion.

The scanning trajectories are generated as sequences of motions, described by positions and orientations of the laser probe with respect to the workpiece. Since the robot arm has 6 degrees of freedom, this appears to be sufficient in order to achieve any desired position and orientation. The reason for using the rotary table is that the robotic arm alone is not able to look at the piece from the front side and then from the back side, especially for large workpieces. For the trajectories in Fig. 2 (a) and (b), it is clear that the workpiece has to be rotated on the table in order to complete the motion. For the trajectories in Fig. 2(c) and (d), if the object is small enough, it will be possible to perform the zig-zag path without rotating the table, but for larger objects, an out-of-range condition for the robot arm will occur, and that can be avoided by rotating the table.

A straightforward solution is to incorporate the rotary table motion in the scanning trajectory. This solution may be good for simple scanning patterns like those presented in Fig. 2, where the solution for the rotary table motion is obvious. However, as one of the aims of this work is to develop automated adaptive scanning procedures, by discovering the workpiece's concave regions which are hard to reach (Impoco et al. , 2005), the resulting trajectories will be more complex. Also, the generation of adaptive scanning trajectories is not an easy task, so it was decided to make the task of rotary table movement completely automated.

The aim of this paper is to present an algorithm that determines automatically a suitable trajectory for the rotary table, when a scanning path around the workpiece is known. Input data for this problem consists of the scanning paths, which are a series of locations (positions and orientations), in the workpiece's reference frame.



(a) (b) (c) (d)

Figure 2: Examples of scanning patterns: (a) spherical; (b) cylindrical; (c) zig-zag; (d) zig-zag with tilted probe

The scanning system has to move continuously and synchronously the rotary table and the robot arm, such as the laser probe reaches the programmed locations and take measurements without stopping the system motion. Output data is a sequence of joint values of the robotic arm and the angle of the rotary table, which give the desired location of the laser probe with respect to the workpiece. In other words, the problem presented here is the inverse kinematics (*IK*) for a 7-DOF kinematic chain. In addition, the computed solution has to satisfy the following requirements:

- minimize the accelerations and limit the speed of the rotary table, since a smooth motion is required for being able to hold the workpiece without additional fixtures
- avoid collisions with any obstacles which may be within the manipulator's range, or between the manipulator and the rotary table;
- flexible reach of the robotic arm, while avoiding the proximity of singular configurations which can result in very high joint speeds

3. Problem modelling

3.1. Kinematics model

From a kinematics point of view, the 6-DOF robot arm and the rotary table are modelled using the Denavit-Hartenberg convention (Spong et al. , 2005), as shown in Fig. 3 and Table 1. The rotary table may be considered fixed and the robotic arm rotating around the workpiece, hence the effect of the 7^{th} degree of freedom is applied before the other 6 links in the kinematic chain. The position of the rotary table relative to robot is modelled therefore as a link 0, or link R, where θ_R is the rotary angle variable.

The location of the laser probe with respect to the workpiece is specified as a homogeneous transformation matrix (HTM), which contains information about the Cartesian position and the orientation. The orientation may be specified in yaw-pitch-roll angles or axis-angle formats, but it should be converted to the HTM format. The scanning trajectory is specified as a series of HTMs, each matrix $T_L^{(k)}$ corresponding to a discrete time step k.

The location of the laser probe with respect to the manipulator base is $T_M^{(k)}$, which is the product between the DK solution for the 6-DOF arm, and the tool transform T_{TL} which specifies the location of the laser probe with respect to the the robot arm flange, and is computed using a calibration procedure (Borangiu et al. , 2009).

$$T_M^{(k)} = T_{DK}\left(\theta_1^{(k)} \dots \theta_6^{(k)}\right) \cdot T_{TL} \tag{1}$$

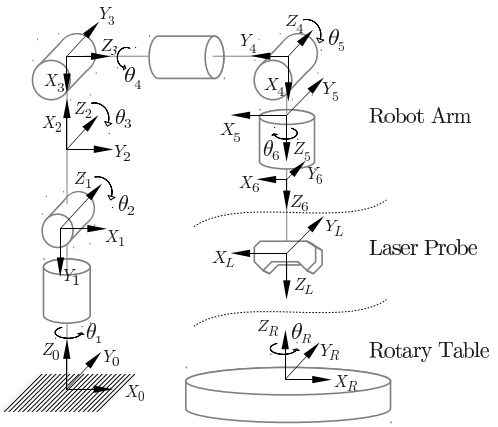


Figure 3: Kinematics model of the scanning system

Link	a_i [mm]	d_i[mm]	α_i [deg]	θ_i [deg]
0/R	-500	-200	0	θ_R
1	75	335	-90	θ_1
2	270	0	0	θ_2
3	-90	0	90	θ_3
4	0	295	-90	θ_4
5	0	0	90	θ_5
6	0	80	0	θ_6

Table 1: Denavit-Hartenberg parameters of the 7-DOF mechanism

The location of the rotary table with respect to the robot base is $T_R^{(k)}\left(\theta_R^{(k)}\right)$, which is a translation indicating the position of the table with respect to manipulator, followed rotation around OZ:

$$T_R^{(k)}\left(\theta_R^{(k)}\right) = \mathcal{T}\left(-a_0, 0, -d_0\right) \cdot \mathcal{R}_Z\left(\theta_R^{(k)}\right) \tag{2}$$

The inverse kinematics (S. R. Buss , 2009) for the 6-DOF manipulator is solved by its controller unit, and will be named *IK*$_6$:

$$\theta_{1\ldots6}^{(k)} = \textit{IK}_6\left(T_{DK}^{(k)}\right) \tag{3}$$

3.2. Problem decomposition

The inverse kinematics of this 7-DOF chain can be decomposed in two steps:

1. Choose a suitable angle for the rotary table;
2. Solve the inverse kinematics for the 6-DOF arm and the angle chosen at step 1.

Supposing that at time step k, the laser probe has to be placed in the workpiece reference frame at the location $T_L^{(k)}$, and the angle $\theta_R^{(k)}$ was chosen, the position of the laser probe with respect to robot base should be:

$$T_M^{(k)}\left(\theta_R^{(k)}\right) = T_R^{(k)}\left(\theta_R^{(k)}\right) \cdot T_L^{(k)} \tag{4}$$

Since *IK*$_6$ usually has a finite number of solutions, and when a robot configuration (LEFTY / RIGHTY, ABOVE / BELOW or FLIP / NOFLIP) is selected, a unique solution can be chosen, and the planning problem would be solved:

$$\theta_{1\ldots6}^{(k)} = \textit{IK}_6\left(T_M^{(k)}\left(\theta_R^{(k)}\right) \cdot \left(T_{TL}\right)^{-1}\right) \tag{5}$$

Therefore, the 7-DOF planning problem is reduced to the planning of a single joint value, the rotary table angle $\theta_R^{(k)}$, for all time steps $k = \overline{1, n}$.

3.3. The configuration map

The configuration space for this problem can be represented in its discrete form as a $m \times n$ image, or map, where:

- the line index i corresponds to the rotary angle θ_R:

$$i = \begin{cases} 1 & \Longrightarrow \quad \theta_R = -180\,^\circ \\ m+1 & \Longrightarrow \quad \theta_R = \quad 180\,^\circ \end{cases}$$

- the column index j corresponds to the discrete time $k = \overline{1, n}$;

Since the rotary table can rotate continuously, the configuration map is periodic on the vertical axis.

The configuration map models *static constraints* such as:

- joint values obtaned by *IK*$_6$ should be inside the allowed limits;
- current robot configuration should not be singular;
- the robot arm should not collide with nearby equipment.

The pixel value at location (i, j) in the map represents whether the static constraints are fulfilled or not, for the rotary table angle $\theta_R(i)$ at time step j. For binary constraints:

$$M_B(i,j) = \begin{cases} 1 \text{ (white):} & \text{all constraints are fulfilled} \\ 0 \text{ (black):} & \text{at least one constraint is not fulfilled} \end{cases} \quad (6)$$

This image will be called the *binary configuration map* (Fig. 4). The white (allowed) regions on the map will be named C_{free} , while the black (forbidden) regions will be denoted as C_{obs}.

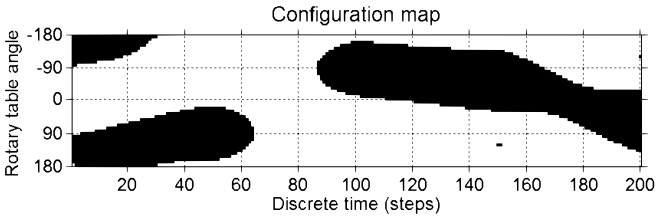


Figure 4: An example of binary configuration map

The configuration map depends only on the scanning patterns $T_L^{(j)}$ specified as inputs for the algorithm, after being discretized into j time steps.

Having defined the configuration map, the planning algorithm has to find a trajectory traversing C_{free}, from the starting rotary table angle, in the first column, to any position in the last column. This trajectory will be the one for the rotary table, and for each point, the corresponding robot position is found with *IK*$_6$.

3.4. The graylevel configuration map

As illustrated in S. M. LaValle et al. (2006), a shortest path solution through the binary configuration space is likely to touch the boundaries of C_{obs}. When this happens, the robot is either near limits of its joints or its working envelope, or close to a singular point, or very close a physical obstacle, which is not desirable.

In order to obtain a planning algorithm that does not touch the boundaries of C_{obs}, but maintains a sufficient distance, one has either to artificially enlarge C_{obs}, or modify the interpretation of the map values in C_{free} to indicate the proximity of a forbidden region.

The values of the map in C_{obs} remain zero, which means that these are forbidden states. The values in C_{free} will be in the $(0, 1]$ interval, showing that any of these states are allowed, but also indicating how desirable is the state. Therefore, states having higher values are more desirable than states having values close to 0.

Therefore, one can construct functions which evaluate a static robot pose with respect to a given criteria, with the following values:

$$\begin{cases} f = 0: & \text{constraint is not fulfilled} \\ f = 1: & \text{constraint is completely fulfilled} \\ 0 < f < 1: & \text{constraint is partially fulfilled} \end{cases}$$

Considering the joint limits of the robot for a given link k, it is possible to use a function which is equal to zero at the joint limits and reaches its maximum value at the middle of the interval:

$$f_{lim}(\theta_k) = \left(\sin\left(\frac{\theta_k - \theta_k^{\min}}{\theta_k^{\max} - \theta_k^{\min}} \cdot \pi \right) \right)^{\gamma_j}, \quad \theta_k^{\min} \leq \theta_k \leq \theta_k^{\max} \tag{7}$$

A function which evaluates the entire robot configuration, with respect to joint limits criteria, can be constructed by multiplying the individual joint functions:

$$f_{lim}\left(\theta_{1\ldots6}\right) = \prod_{k=1}^{6} f_{lim}(\theta_k) \tag{8}$$

By adjusting the values of the exponents γ_j, it is possible to control the shape of the functions (Fig. 5 a).

For collision avoidance, the geometric model is required for every component (robot links, sensor, rotary table). This model can be imported from CAD files or

approximated by primitive shapes. For each pair (i, j) of possibly colliding bodies, it is required to ensure a minimal distance A_{ij}, while the preferred value for this distance is B_{ij}. The distance between two bodies is considered the minimum distance between their surfaces, $d_{min}(\theta_{1...6}, i, j)$, and the function for evaluating the collision criteria for one pair of bodies is (Fig. 5 b):

$$f_c\left(\theta_{1...6}, i, j\right) = \begin{cases} 0, & d_{min} \leq A_{ij} \\ \left(\sin\left(\dfrac{d_{min}(\theta_{1...6}, i, j) - A_{ij}}{B_{ij} - A_{ij}} \cdot \dfrac{\pi}{2}\right)\right)^{\gamma_c}, & A_{ij} < d_{min} < B_{ij} \\ 1, & B_{ij} \leq d_{min} \end{cases} \tag{9}$$

For considering all possible pairs (i, j), the evaluation function is:

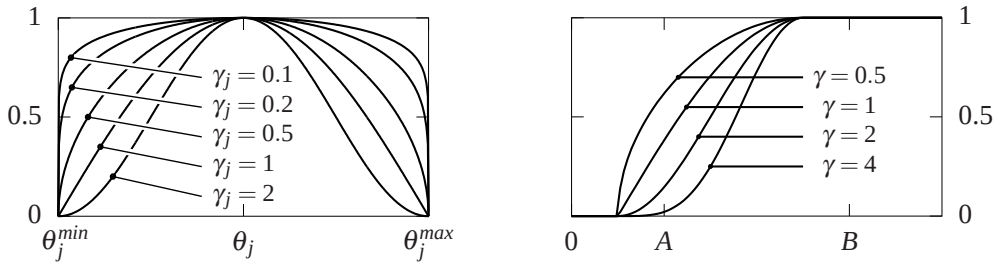
$$f_C(\theta_{1...6}) = \prod_{i,j} f_c\left(\theta_{1...6}, i, j\right) \tag{10}$$

Near kinematic singularities, maintaining a low speed at the end-effector may require very high velocities in robot joints (Adept , 2007), singular configurations, and also their proximity, should be avoided. A possibility is to use the ***manipulability index*** proposed by (T. Yoshikawa , 1985) as the distance between the current robot position and the closest singular configuration:

$$d_s(\theta_{1...6}) = \omega(\theta_{1...6}) \tag{11}$$

The criteria for avoiding proximity of singular configuration is similar to (9):

$$f_s(\theta_{1...6}) = \begin{cases} 0, & d_s \leq d_s^{min} \\ \left(\sin\left(\dfrac{d_s - d_s^{min}}{d_s^{max} - d_s^{min}} \cdot \dfrac{\pi}{2}\right)\right)^{\gamma_s}, & d_s^{min} \leq d_s \leq d_s^{max} \\ 1, & d_s \leq d_s^{max} \end{cases} \tag{12}$$



(a) For avoiding proximity of joint limits (b) For avoiding collisions and singularities

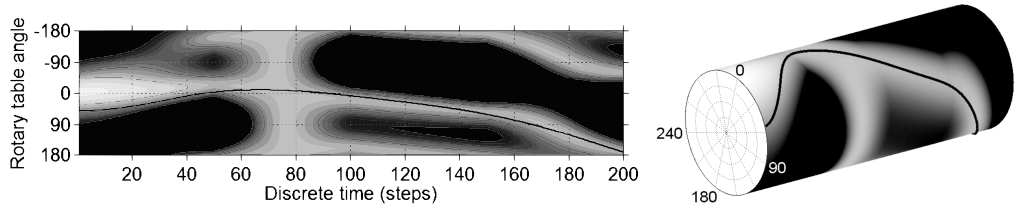
Figure 5: Functions for evaluating static robot positions

For evaluating a given robot configuration with respect to all the above mentioned static criteria, one has to multiply the individual functions, obtaining:

$$f_R(\theta_{1...6}) = f_{lim}(\theta_{1...6}) \cdot f_C(\theta_{1...6}) \cdot f_s(\theta_{1...6}) \tag{13}$$

which also takes real values between 0 and 1.

The *grayscale configuration map* will be represented as a 2D grayscale image, with pixel values representing f_R from (13), like the example in Fig. 6 (a).



(a) Two-dimensional view (b) Cylindrical view

Figure 6: An example of grayscale configuration map

Since the angle is represented on the vertical axis, and the table can rotate without restrictions, the map can also have a cylindrical representation, like in Fig. 6 (b).

The criteria which avoids the proximity of joint limits has another desirable property, illustrated in Fig. 7, which is ensuring a natural (or relaxed) joint configuration. In Fig. 7 (a)...(c), the laser sensor has the same position relative to the scanned part. The (c) configuration is the most desirable, and it maximizes (8).

(a) (b) (c)

Figure 7: Robot arm and laser looking at a workpiece from different positions:
(a) Robot is near a "too close" condition (table rotated at $-25\,^\circ$)
(b) Robot is near a "too far" condition (table rotated at $-95\,^\circ$)
(c) Robot is not close to its limits (table rotated at $-60\,^\circ$).

3.5. Performance criteria

This section defines a cost function for evaluating a path computed by a planning algorithm. This function models the criteria presented in Section 2, which can be divided into:

- Static criteria, like joint limits, singular configurations or collision avoidance, specified by the greylevel values in the configuration map;
- Dynamic criteria: the speed and acceleration of the rotary table.

In this application, there were no explicit speed and acceleration constraints for the robot arm itself; excessive speeds of the robot joints can appear only in the proximity of kinematic singularities, which are avoided using a static constraints; therefore, speeds and accelerations in the robot arm are avoided indirectly with (12).

Let n be the number of discrete time steps for a given planning problem. The initial conditions are the rotary table angle $\theta_R^{(0)}$ and the rotary table angular velocity $\omega_R^{(0)}$. The planner will compute the angles $\theta_R^{(1)}$ through $\theta_R^{(n)}$, which are obtained by integrating the angular velocities $\omega_R^{(0)}$ to $\omega_R^{(n-1)}$ or by performing a double integration of the angular accelerations $a_R^{(0)}$ to $a_R^{(n-1)}$.

During the discrete moments k and $k+1$, which correspond to continuous time moments t_k and $t_{k+1} = t_k + \Delta t$, the table rotates using the constant acceleration $a_R^{(k)}$. Therefore,

$$\omega_R^{(k+1)} = \omega_R^{(k)} + a_R^{(k)}\ \Delta t \tag{14}$$

$$\theta_R^{(k+1)} = \theta_R^{(k)} + \omega_R^{(k)}\ \Delta t + a_R^{(k)} \frac{(\Delta t)^2}{2} \tag{15}$$

The penalty due to rotary table motion (high speeds or high accelerations) can be expressed as:

$$C_{a\omega} = \sum_{i=0}^{n-1} \left(k_a \left(a_R^{(i)} \right)^2 + k_\omega \left(\omega_R^{(i)} + a_R^{(i)}\ \Delta t \right)^2 \right) \tag{16}$$

where k_a and k_ω are scalar weights for the angular acceleration and speed of the rotary table.

The cost due to partial fulfillment of static constraints modelled by the configuration map can be defined as:

$$C_{adv} = \sum_{i=1}^{n} \left(1 - f_R \left(\theta_i,\ T_L^{(i)} \right) \right) \tag{17}$$

If the robot at step i can position the laser probe at the position $T_L^{(i-1)}$, but it cannot reach $T_L^{(i)}$ by rotating the table in the time Δt while obeying the maximum acceleration and speed, the scanning process has to wait until the rotary table comes into an acceptable position, so that the robot is able to reach $T_L^{(i)}$.

Therefore, a delay in the scanning process is introduced, and it will be named $d^{(i)}$. The delay depends on the amount of the rotation needed, and the maximum speed and acceleration limits. If the table rotation is small enough, no delay is introduced from time steps $i-1$ to i, and therefore, $d^{(i)} = 0$. By summing the delays for every time step and weighting them with the scalar k_d, the delay cost for a given path is:

$$C_{delay} = k_d \sum_{i=1}^{n} d(i) \tag{18}$$

The total cost function of a given path will be:

$$C_{path}\left(\theta_R^{(1\ldots n)},\ \omega_R^{(0\ldots n-1)},\ a_R^{(0\ldots n-1)},\ d^{(1\ldots n)}\right) = C_{a\omega} + C_{delay} + C_{adv} \tag{19}$$

4. Algorithms

Any planning algorithm presented here will start from an initial condition vector of the rotating system, which contains the rotary table angle $\theta_R^{(0)}$ and the rotary table angular velocity $\omega_R^{(0)}$. The destination is not fixed, the planned path has only to reach the last column in the configuration map, at any rotary angle and angular velocity, $\theta_R^{(n)}$ and $\omega_R^{(n)}$. After the map is traversed, the scanning trajectory is completed and the rotary table may either decelerate and stop, or prepare for another scanning trajectory.

Related work in planning algorithms focus on two or more dimensions in the configuration space (Latombe et al. , 1999), (M. Sharir , 2004). Successful solutions include roadmap-based planners, which use a graph that represents the connectivity of the free configuration space, and potential field methods (Y. K. Hwang, N. Ahuja , 1992), which represent forbidden areas as repulsive forces and goals as attractive forces, allowing the planning problem to be solved using gradient-based optimization method. Recent approaces use genetic algorithms (Kazem et al. , 2008), probabilistic methods and random sampling schemes (J. Barraquand et al. , 1997).

The planning problem presented in this article is similar to a smooth path finding problem for a point moving in a 2D space; however, the second dimension represents the time, since the duration of the motion and the speed profile for the

laser sensor with respect to the workpiece are known a priori; therefore, state-of-art algorithms for 2D problems cannot be applied directly.

The planning problem has speed and acceleration constraints, and therefore it can be considered nonholonomic (Tanner et al. , 2000). A possible solution is to expand the configuration space with a third dimension representing the rotational speed of the table; in this way, the problem can be solved optimally using a Dijkstra-like algorithm (Cormen et al. , 2000).

4.1. Rotate when out of range

This algorithm, given in pseudocode as Algorithm 1, is the simplest strategy for rotating the table. Whenever the trajectory hits an obstacle on the binary configuration map, the algorithm finds the minimum amount of rotation that gets out of the forbidden map area. However, this will introduce high delays in the scanning process, but this strategy can be used as a fall-back mechanism. This is possible due to the nature of the current application, where scanning (and robot movement) can be paused, while only rotating the table until the system reaches an allowed configuration.

Algorithm 1: Rotate when out of range

$\theta \leftarrow \theta_R^{(0)}$
for $j = 1$ **to** n **do**
  if $f_R\left(\theta,\, T_L^{(j)}\right)$ **then**
    for $\varphi = 0$ **to** 180 **step** $\Delta\theta$ **do**
      if $f_R\left(\theta+\varphi,\, T_L^{(j)}\right)$ **then**
        $\theta \leftarrow \theta + \varphi$
        break
      if $f_R\left(\theta-\varphi,\, T_L^{(j)}\right)$ **then**
        $\theta \leftarrow \theta - \varphi$
        break
  $\theta_R^{(j)} \leftarrow \theta$

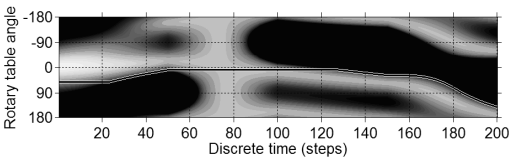


Figure 8: Sample solution for Algorithm 1. The planned path touches the edges of the obstacles.

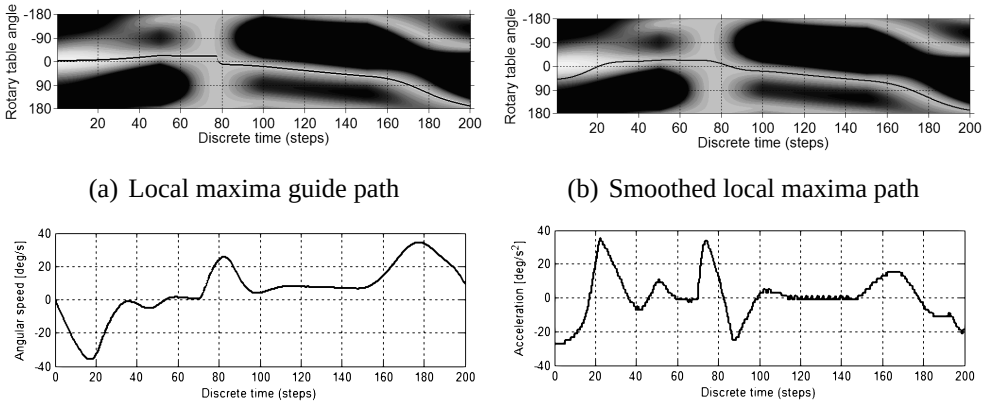
In this and all subsequent examples, the time step used is $\Delta t = 0.1\,\mathrm{s}$, therefore the test motion, which has 200 discrete steps, takes 20 seconds to complete, unless otherwise stated.

4.2. Local maxima search

This algorithm attempts to use the graylevel configuration map as a *potential field*, and at every time step, i. e. on every column on the map, steer the table towards a local maxima of the current column. The speed of steering, and also

the change in speed, are weighted according to the performance criteria specified previously, in order to ensure a smooth motion.

This approach works better than the first method; however, it is sensitive to local maximas in the graylevel map, which in many cases may lead to deadlock situations.



(a) Local maxima guide path

(b) Smoothed local maxima path

(c) Angular speed (smooth path) $(\deg/\mathrm{s})$

(d) Angular acceleration $(\deg/\mathrm{s}^2)$

Figure 9: Results of local maxima search algorithm

The guide path cannot be followed exactly due to possible sharp corners (Fig. 9 a), which require high instantaneous acceleration rates. A smoothed path (Fig. 9 b,c,d) can be followed easier, but it still requires accelerations much higher than needed. Accelerations could be lowered by a stronger smoothing filter, but with the risk of the smoothed path touching the forbidden areas on the map.

4.3. A Dijkstra-like algorithm

The Dijkstra algorithm finds the minimal cost path through a graph with positive weights (or costs) on its edges, from a given node to all the other nodes that can be reached from it Cormen et al. (2000). Given the discrete modelling of the planning problem, it can be formulated in terms of graph theory.

In order to put constraints on the speeds and accelerations, a third dimension has to be added to the configuration space: the angular velocity. The graph nodes form a finite 3D matrix, as they map to a continuous 3D space, having on the first dimension the rotary table angle θ , on the second dimension the continuous time t , and on the third dimension, the angular velocity ω. A discrete representation (i, j, k) maps to an unique continuous configuration $(\theta_i,\ t_j,\ \omega_k)$. The reverse

transform, from continuous to discrete, assigns a single discrete node (i, j, k) to a continuous interval:

$$\begin{pmatrix} \theta_i - \frac{\Delta\theta}{2} & \leq \theta < & \theta_i + \frac{\Delta\theta}{2} \\ t_j - \frac{\Delta t}{2} & \leq t < & t_j + \frac{\Delta t}{2} \\ \omega_k - \frac{\Delta\omega}{2} & \leq \omega < & \omega_k + \frac{\Delta\omega}{2} \end{pmatrix} \Rightarrow \begin{pmatrix} i \\ j \\ k \end{pmatrix} \tag{20}$$

The method for obtaining a Dijkstra-like algorithm for a continuous problem is presented in S. M. LaValle et al. (2006).

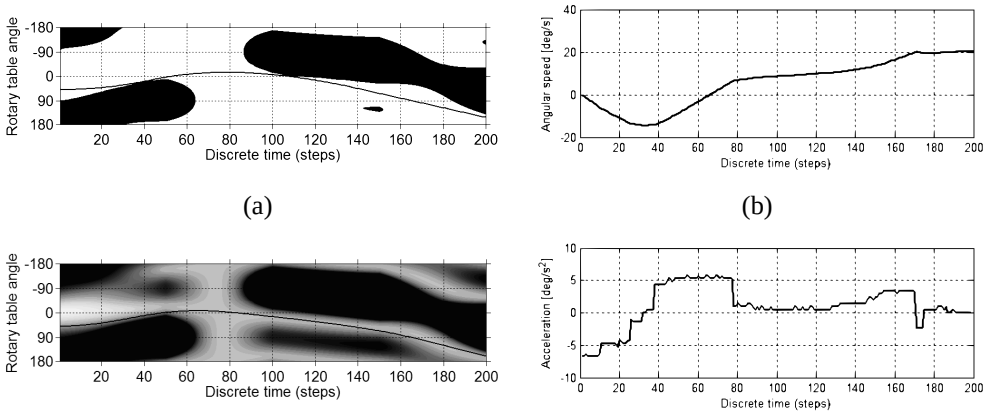


Figure 10: Examples for Dijkstra algorithm:
(a) Path computed for a binary configuration map
(b) Path computed for a grayscale configuration map
(c) Angular speed for the grayscale path $[\,\mathrm{deg/s}]$
(d) Angular acceleration for the grayscale path $[\,\mathrm{deg/s^2}]$

From the node $(\theta_i,\ t_j,\ \omega_k)$, is it possible to advance using the acceleration a and reach the node corresponding to $(\theta_i + \omega_k\ \Delta t + a\frac{(\Delta t)^2}{2},\ t_j + \Delta t,\ \omega_k + a\Delta t)$. This is an edge in the graph having the cost $C^*_{edge} = k_a\ a^2$.

The costs put on velocities are modelled as node costs. Every node corresponding to $(\theta_i,\ t_j,\ \omega_k)$ has a velocity cost equal to $k_\omega\ (\omega_k)^2$.

Also, every node has a cost corresponding to equation (17), which is equal to $1 - f_R(\theta_i,\ T_L^{(j)})$. Therefore, the cost of a node is:

$$C_{node} = k_\omega(\omega_k)^2 + 1 - f_R\left(\theta_i,\ T_L^{(j)}\right) \tag{21}$$

The costs on nodes on a graph can be transferred on the incoming edges, therefore the edge cost becoming:

$$C_{edge} = k_a \; a^2 + k_\omega(\omega_k)^2 + 1 - f_R\left(\theta_i,\; T_L^{(j)}\right) \tag{22}$$

If a delay has to be introduced, the edge cost increases with a component corresponding to C_{delay} from equation (18). This allows edges to traverse C_{obs}, in order to ensure that a solution is found even if C_{free} is not conex; however, these edges should have a much higher cost, so that this solution is selected as a last resort.

By summing the edge costs for an entire path, the cost function is equal to C_{path} from equation (19).

Because of this mode of cost assignment, the Dijkstra algorithm will minimize C_{path}, finding the optimal solution for the discretized problem.

An example of running the Dijkstra algorithm over a binary configuration map is given in Fig. 10. This is the smoothest possible solution for the rotary table (with respect to $C_{a\omega}$), but the path touches the obstacles, and therefore the robot arm will reach its joint limits, which is not desirable.

Running the same algorithm on the graylevel configuration map gives the result from Fig. 10 (b), where the planned path does not touch the edges of the obstacles, the motion is smooth and the acceleration rates are much lower than those in Fig. 9, obtained with the Local Maxima heuristic algorithm. As the discretization steps become smaller, the solution found by the Dijkstra algorithm converges to the optimal solution for the continuous problem.

The graph used in this algorithm can grow very large, as its size grows with the third power of the number of discretization steps of the 3D configuration space. If the space uses 100 steps on every dimension, the graph has 1,000,000 nodes, which is too much for a real time solution. However, the Dijkstra algorithm offers a reference trajectory, to which the solutions found by the other algorithms can be compared.

4.4. "Ray Shooting" heursitic algorithm

This algoritm attempts to provide a solution close to the one obtained with Dijkstra's algorithm, while being suitable for a real-time implementation, which is required because the adaptive path generator may change the scanning trajectories or add new ones while the geometry of the scanned part is being acquired.

At every time step k, the algorithm looks ahead p future time steps, that is, from $k + 1$ through $k + p$. Over this range, it tries to perform a motion with

constant angular acceleration, to ensure the smoothness of the planned path. The algorithm uses a finite set of acceleration values $a_j,\ j = \overline{1, n_a}$, and for every acceleration a_j, a possible path is evaluated, starting from the current state and spanning on the following p time steps. From the set of paths, the path having minimal cost value is chosen. This path corresponds to an acceleration equal to $a_{j,best}$, and the motion from time step k through time step $k+1$ will be performed using this acceleration. The planning algorithm advances to the next step and recomputes the paths, therefore at every time step it makes a decision. This means that at every time step there is a finite number of computations that have to be done, which make possible running the algorithm in real time.

Computing the path using a given acceleration is done by applying equations (14) and (15) over the desired range, and evaluating the cost is done using (19). The path computation stops if an obstacle is reached, and a cost penalty C_{pen} is added to the path, in order to favor the rays that did not hit any obstacle.

Algorithm 2: "Ray Shooting" heuristic

$\theta \leftarrow \theta_R^{(0)}$
$\omega \leftarrow \omega_R^{(0)}$
for $j = 1$ **to** $n-1$ **do**
  for $i = 1$ **to** n_a **do**
    $C_i \leftarrow$ **evalpath**$(\theta,\ \omega,\ A_i,\ j+1,\ \min(j+p,\ n))$
  $i \leftarrow \text{argmin}(C_i)$
  $a \leftarrow A_i$
  $\theta_{new} \leftarrow \theta + \omega\,\Delta t + a\,\frac{(\Delta t)^2}{2}$
  if $f_R\left(\theta_{new},\ T_L^{(j)}\right)$ **then**
    $\omega \leftarrow \omega + a\,\Delta t$
    $\theta_R^{(j)} \leftarrow \theta_{new}$
    $\omega_R^{(j)} \leftarrow \omega$
  else
    fall back using *Algorithm 1*

Function $C =$ **evalpath**$(\theta_0,\ \omega_0,\ a,\ j_{\text{ini}},\ j_{\text{fin}})$

$\theta \leftarrow \theta_0$
$\omega \leftarrow \omega_0$
$C \leftarrow 0$
for $j = j_{\text{ini}}$ **to** j_{fin} **do**
  $\omega \leftarrow \omega + a\,\Delta t$
  $\theta \leftarrow \theta + \omega\,\Delta t + a\,\frac{(\Delta t)^2}{2}$
  if $f_R\left(\theta,\ T_L^{(j)}\right)$ **then**
    $C \leftarrow C + k_a\,a^2 + k_\omega\,\omega^2 - f_R\left(\theta,\ T_L^{(j)}\right)$
  else
    $C \leftarrow C + C_{pen}$
    return

An example of running the Ray Shooting heuristic over the tested configuration map is given in Fig. 11. The figures show the path chosen until the current step, and the rays which determine the choice for the next step. In this example, the rays analyze the configuration map within 70 discrete time steps ahead.

In the first graph, Fig. 11a, the rays can "see" a solution for avoiding the obstacles by turning the table clockwise, i. e. θ_R decreasing, thus avoiding both obstacles on their "left" side. In the second graph, the rays begin to see an alternate solution, which would avoid the second obstacle on its "right" side, by changing the direction of the rotation of the table. Since the acceleration needed to change

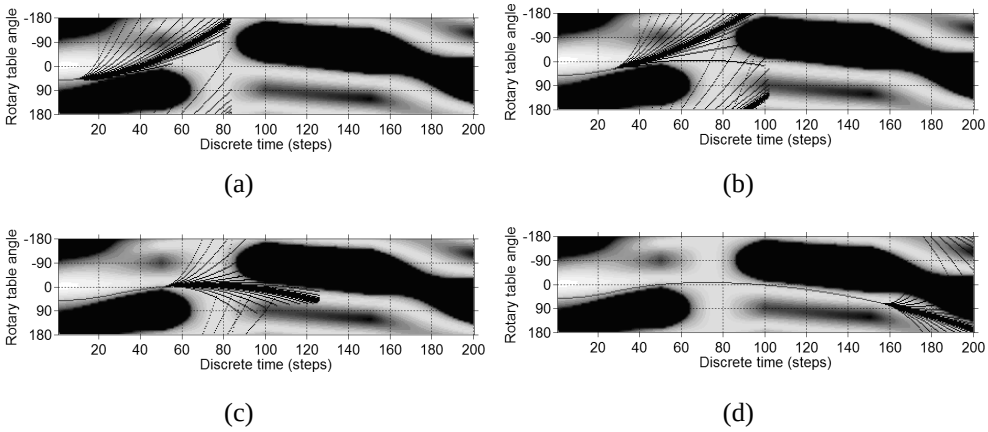


Figure 11: Ray Shooting example:
(a) The rays found a solution by avoiding both obstacles on the left side
(b) One of the rays avoids the second obstacle on the right side
(c) The algorithm decides to avoid the second obstacle on the right
(d) The algorithm has almost finished.

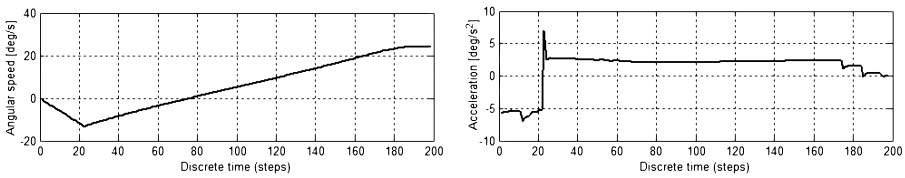


Figure 12: Speed and acceleration obtained by Ray Shooting algorithm

the rotation is smaller than the acceleration needed to avoid the second obstacle on the left, the planner chooses to reverse the rotation direction, resulting the solution from Fig. 11c.

The planned solution is shown in the last graph. The rotary table trajectory is smooth and far from the forbidden regions, therefore the robot arm is not driven close to its joint limits. This solution satisfies the design requirements, has a shape similar to the one computed by Dijkstra algorithm, and it can be computed in real time.

The solution obtained by this algorithm is much smoother than the local maxima path, and is closer to the path computed by the Dijkstra algorithm. The acceleration rate for the two paths is comparable, as it can be observed by comparing Fig. 12 with Fig. 10. The planned path exhibits some small peaks in the

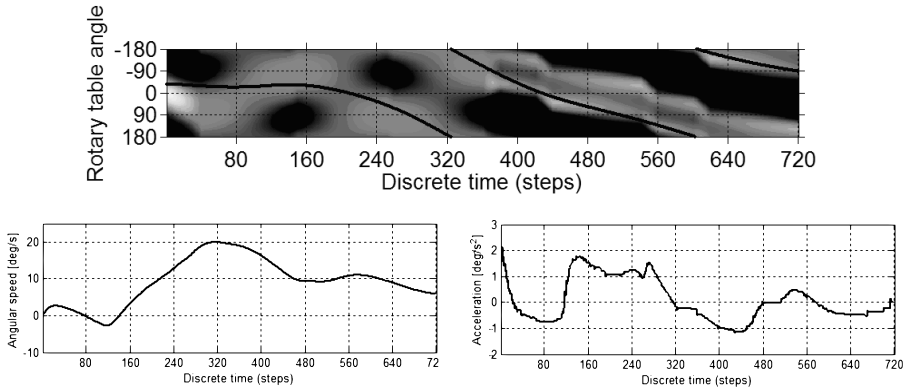


Figure 13: Another example of Ray Shooting algorithm. The planned path wraps around (i.e. the table performs more than one complete rotation)

acceleration, whose effects are minor and can be removed with a smoothing post-processing routine.

In Fig. 13, another example of a path planned with the Ray Shooting algorithm is presented. The path starts from $-38\,^\circ$ and ends at $637\,^\circ$, or $-83\,^\circ$ when wrapped around, the total amount of rotation being $675\,^\circ$.

5. Conclusions

This article presented the problem of solving a redundant 7-DOF inverse kinematics problem, which usually has an infinite number of solutions. The motion planning problem has two constraints, the speed and the acceleration of the rotary table, which are used in order to achieve a smooth table motion, since there is no rigid mechanical fixture between the table and the scanned part.

The problem was modelled in order to find the minimal cost path through a configuration space. The Dijkstra algorithm gave the optimal solution to the problem, however it was too slow for a real time implementation. A heuristic algorithm was proposed, which is able to compute a smooth path in real time, while respecting the imposed restrictions.

The heuristic algorithm is implemented as a module for the laser scanning software, and it runs on the PC which serves the 3D reconstruction system by performing the real-time planning of the rotary table and sending the planned result to the robot and rotary table controllers.

The proposed algorithm can be used in any other applications which involve a robot moving on a 3D continuous path along a workpiece, while performing

a technological operation, i. e. welding, edge polishing, and there is a redundant degree of freedom whose motion has to be planned in order to achieve various constraints, such as collision and singularity avoidance, while maintaining a smooth trajectory.

Acknowledgements

This work is funded by the National Council for Scientific University Research, in the framework of the National Plan for Research, Development and Innovation, grant 69/2007, and by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/6/1.5/S/19.

References

M. Spong, M. Vidyasagar, Forward Kinematics: The Denavit-Hartenberg Convention. In: John Wiley and Sons, Inc., Robot Modeling and Control. 2005, p. 71-83.

L. Sciavicco, B. Siciliano, Modelling and Control of Robot Manipulators, McGraw-Hill, London, 1996.

S. M. LaValle, Planning Algorithms. Cambridge University Press, 2006.

P. I. Corke, A robotics toolbox for MATLAB, IEEE Robotics and Automation Magazine, Vol. 3, n. 1, pp. 24-32, March 1996.

Th. Borangiu, F. Ionescu, Robot Kinematics, In: AGIR and Editura Academiei Romne, Robot Modelling and Simulation, Bucharest 2002, p. 122-140

Adept Technology, Inc., Six-Axis Robot Configuration Singularities, March 2007

T. H. Cormen et al., Single Source Shortest Paths, In: The MIT Press, Introduction to Algorithms, 2000, p. 527-531.

G. Impoco, P. Cignoni, R. Scopigno, A Six-Degrees-of-Freedom Planning Algorithm for the Acquisition of Complex Surfaces, International Journal of Shape Modeling, Vol. 11, no. 1, June 2005, p. 1-23.

Herbert G. Tanner and Kostas J. Kyriakopoulos, Nonholonomic Motion Planning for Mobile Manipulators, Proceedings of ICRA 2000

B. I. Kazem, A. I. Mahdi, A. T. Oudah, Motion Planning for a Robot Arm by Using Genetic Algorithm, Jordan Journal of Mechanical and Industrial Engineering, Vol. 2, No. 3, Sep. 2008, p. 131-136

J. Barraquand et al. (1997) A Random Sampling Scheme for Path Planning, The International Journal of Robotics Research, Vol. 16, No. 6, 759-774 (1997)

Micha Sharir, Algorithmic Motion Planning, in Handbook of Discrete and Computational Geometry Second Edition, CRC Press, Chapter 47, 2004

D. Halperin, L.E. Kavraki, J.C. Latombe, Robot Algorithms. In Algorithms and Theory of Computation Handbook, CRC Press, Chapter 21, pp. 21-1–21-21, 1999.

Y.K. Hwang, N. Ahuja, A potential field approach to path planning, IEEE Transactions on Robotics and Automation, Vol. 8, Issue 1, p.23-32, 1992.

S. R. Buss, Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods, Department of Mathematics University of California, San Diego, 2009.

T. Yoshikawa, Dynamic manipulability of robot manipulators, Journal of Robotic Systems, 2 (1985), pp. 113–124.

Th. Borangiu, A. Dogar, A. Dumitrache, Calibration of Wrist-Mounted Profile Laser Scanning Probe using a Tool Transformation Approach, The 18^{th} Intl. Workshop on Robotics in Alpe-Adria-Danube Region - RAAD'09, Brasov, Romania.