A Heuristic Approach for Constrained Real Time Motion Planning of a Redundant 7-DOF Mechanism for 3D Laser Scanning

Theodor Borangiu*, Anamaria Dogar,* Alexandru Dumitrache*

**Centre for Research and Training in Industrial Control,*
Robotics and Materials Engineering, University Politehnica of Bucharest,
Romania (Tel: +40/21 402 93 14; e-mail: {borangiu, dogar, alex} @cimr.pub.ro)

Abstract: This article presents a heuristic algorithm for motion planning of a short range 7-DOF Laser Scanning System consisting of a 6-DOF vertical robot arm and a rotary table holding the workpiece. The redundancy of the mechanism is exploited for specifying certain constraints such as imposing a smooth motion of the rotary table while respecting the acceleration and speed limits and avoiding undesirable configurations of the robot arm such as the kinematic singularities, out-of-range conditions and collisions between the mechanical elements of the system (robot, table, workpiece and laser probe). The paper also presents the control system of the rotary table, and a communication protocol which allows the integration of the above-mentioned planning algorithm into the 3D scanning system.

Keywords: 3D Laser Scanning, Path planning, Inverse kinematic problem, Heuristic and Metaheuristics, Optimization and Control, Computational Science

1. INTRODUCTION

The aim of this paper is to attempt to solve a motion planning problem for an automatic 7-DOF robot-based 3D scanning system used for surface reconstruction of existing objects. A comprehensive overview of the technologies used in industry for creating geometric models of existing objects can be found in Varady et al. (1996).

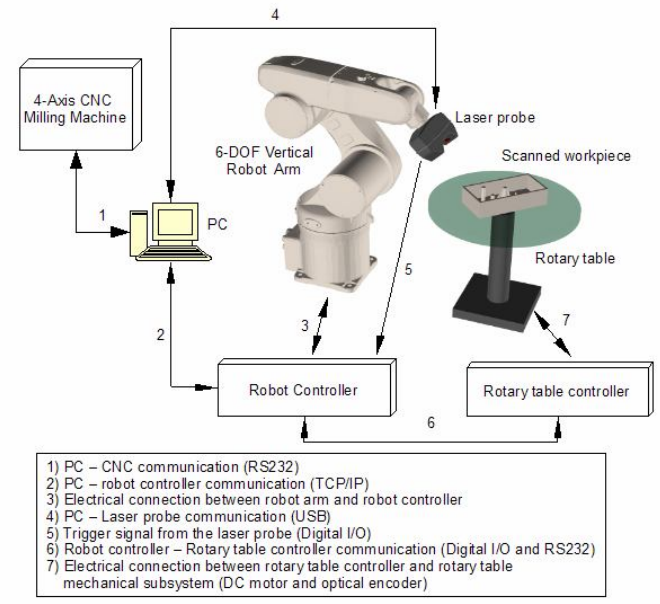

Fig. 1. Overview of the 3D laser scanning system.

The scanner presented in this paper uses short-range triangulation-based laser probe mounted on a 6-DOF vertical robot arm which moves a 3D laser probe around the analyzed workpiece, as it can be seen in Fig. 1. The manipulator has a spherical working envelope with a radius of 650 mm and a repeatability of 20 μm, and the laser probe is able to measure distances from 100 to 200 millimetres with 25 μm accuracy at constant ambient temperature, using one laser beam and two CCD cameras. In order to improve the coverage of a scan pass at the surface of the object of interest, it will be placed on a rotary table, which is the 7th degree of freedom to the scanning system. A similar setup was proposed by Rahayem et al. (2008), with a 810 mm reach of the robot arm and 30 μm repeatability, and a single-camera laser probe.

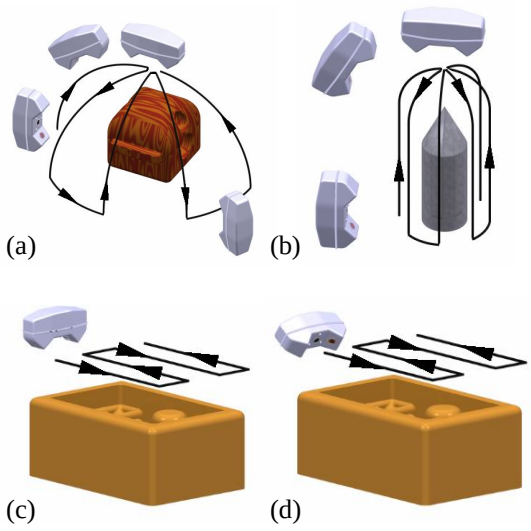

Fig. 2. Predefined scanning patterns for various object classes

The scanner will use either predefined scanning patterns (Fig. 2) based on the workpiece class and approximate dimensions, or adaptive scanning paths which automatically discover concave regions on the workpiece that were not covered by the predefined paths, and generate new paths in order to scan the hidden areas. The predefined scanning patterns from Fig. 2 may be used for spherical and cylindrical objects and also for interiors of the molds.

The trajectories from Fig. 2 (a) and (b) require the object to be rotated on the table, and the solution for finding the amount of rotation seems to be immediate. The trajectories from Fig. 2 (c) and (d) may not need rotating the table if the scanned object is small enough, but if this is not the case, the robot may reach an out-of-range configuration, which can be avoided by turning the table to a convenient angle. In the case of adaptive scanning paths, they are not predictable and the amount of table rotation needed is not easy to compute.

An adaptive scanning strategy is presented by Impoco et al., (2005), and is described using 6 degrees of freedom. A reasonable solution for implementation of the 7-DOF mechanism in the scanning process will be using a *rotary table planner* whose function is to automatically rotate the table so that any given scanning trajectory will be followed without reaching an out-of-range condition for the robot arm. The planner will perform the inverse kinematics of the 7-DOF mechanism, and because it has an extra degree of freedom, it will be able to satisfy other constraints, such as kinematic singularity avoidance, smooth table motion while respecting speed and acceleration limits, and also avoiding collisions between the robot arm, laser probe, rotary table and the scanned workpiece.

2. ROTARY TABLE CONTROLLER

2.1 Hardware solution

The existing rotary table mechanism is actuated by a 12V DC motor which has an optical quadrature encoder attached. The transmission is performed by a spur gears chain with 1:480 ratio. The controller is based on an 8-bit RISC microcontroller which reads the encoder and drives the motor using PWM, rotating the table at the desired angular position using the programmed speed and acceleration values. The loop is closed by a PID regulator and a trapezoidal velocity profile (Fig. 3). The robot controller regards the table as a slave device, and the motions between the robot and table are synchronized with the data from the laser probe.

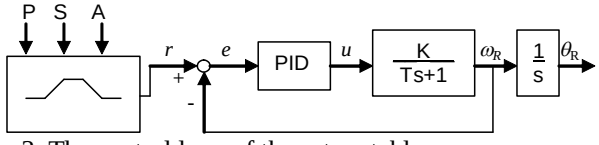

Fig. 3. The control loop of the rotary table.

The motions executed by the rotary table use either constant acceleration or constant speed. Therefore, the control loop has two sets of tuning parameters: in the first case it has to follow a ramp, while in the second case it has to follow a step reference, having also to reject a step perturbation.

The velocity profile, which is the reference for the PID loop, is computed using the destination angular position *P*, top speed *S* and acceleration *A*, and the PID algorithm outputs the motor command *u* based on the error signal *e*, which is the difference between the reference speed *r* and the actual motor speed read from the encoders, ω_R. When the table position θ_R reaches the planned position *P*, the motion stops. For this application it is not required for the table to stop exactly in the planned position *P*, since the matrices for aligning the 3D data are computed using the values read for the encoders.

2.2 Communication protocol

The rotary table controller is interfaced to the robot controller using a serial interface and an ASCII-based protocol summarized in Table 1.

Table 1. Communication protocol summary

Cmd.	Reply	Meaning
`H <cr>`	`H` *nn.n* `<cr>`	Read instantaneous position of the rotary table from encoders, in degrees.
`L <cr>`	`L` *nn.n* `<cr>`	Read the position of the rotary table, which was latched by the synchronization signal from the 3D probe.
`P` *nn.n* `<cr>`	`OK <cr>`	Absolute positioning of the table.
`R` *nn.n* `<cr>`	`OK <cr>`	Incremental positioning of the table
`S` *nn.n* `<cr>`	`OK <cr>`	Sets maximum rotation speed for the next motion command (P or R), in degrees/sec.
`A` *nn.n* `<cr>`	`OK <cr>`	Sets the acceleration for the next motion command, in degrees/second^2.

The robot controller is able to set the motion parameters (speed and acceleration) and to move the table at the desired position. The rotary table controller accepts commands while motions are in progress, and it is able to interrupt the existing motion and start a new one, obtaining more complex speed profiles like in Fig. 4. This property also allows synchronization of the motion between the robot and the rotary table and integration with the rotary table planner.

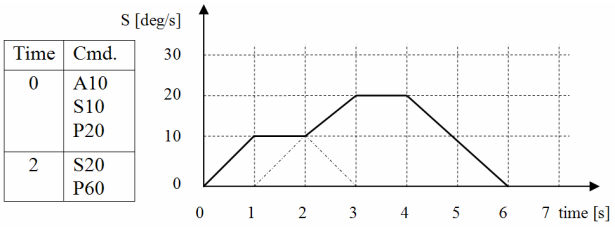

Fig. 4. A complex speed profile obtained by sending a new motion command before the last one has completed.

The synchronization between the laser probe measurements and the motions of the robot and table is performed using a trigger signal, sent by the laser probe to the robot/table controlers every time a new data set is acquired. The instantaneous position of the robot and table is latched on the trigger signal, and used in order to compute the transformation matrix which aligns the data from the laser probe into a unique reference frame.

3. ROTARY TABLE PLANNER

As mentioned in the beginning of the article, the rotary table planner will perform the inverse kinematics (IK) of the 7-DOF mechanism. The scanning trajectory, which may be a predefined pattern from Fig. 2 or an adaptive path based on the features of the workpiece analyzed, is a sequence of scanning locations, defined in the reference frame of the workpiece. One location contains information about cartesian position (X/Y/Z) and orientation (Yaw/Pitch/Roll) of the laser probe with respect to the workpiece, and it is represented as a homogeneous transformation matrix. Since there are six variables completely describing the location of the laser probe, and the mechanism has 7 degrees of freedom, it is usually possible to choose from an infinite number of solutions for the IK problem.

If the angular position of the rotary table, θ_R, is fixed to an arbitrary value, the IK problem is solved by computing the closed-form inverse kinematics solution (CIKS) for the 6-DOF manipulator with spherical wrist (Borangiu et al. 2002), and choosing a single solution based on the desired arm configuration (LEFTY or RIGHTY, ABOVE or BELOW, FLIP or NOFLIP). This allows the 7-DOF IK problem to be reduced to planning the path for a single DOF, while the remaining six DOF are determined analytically, using the Denavit-Hartenberg convention (Spong et al., 2005). The 7-DOF kinematic model of the system was presented in detail in Borangiu et. al (2008a, b) with simulation results and issues regarding calibration.

For certain values of θ_R it is possible that no solution exists for the remaining 6 joints, or if there is a mathematical solution, the joint values are out of their allowed range. For other values of θ_R, the unique solution chosen by the CIKS procedure may or may not violate other constraints, for example the robot may collide with the rotary table or the mechanism may be near a singular configuration, which has to be avoided. Therefore, for a given location of the laser probe with respect to the workpiece, a value function which depends on θ_R can be defined: $f_R(\theta_R, T_L)$.

This function is used for evaluating the robot pose computed by the CIKS procedure for any fixed θ_R, and it takes real values between 0 and 1, where 0 means that the solution for the given θ_R is not acceptable at all, or it does not exist, and 1 means that the solution does not violate any constraint. Lower values of the function should be avoided if possible, but they have to be used if no other solution is possible.

The value function for a given configuration will be defined as a product of individual functions, which also has values between 0 and 1. For example, the component which avoids an out-of-range situation for a given robot joint *j*, and also keeps the joint away from its extreme values, can be:

$$f_j(\theta_j) = \left(\sin\left(\pi \cdot \frac{\theta_j - \theta_j^{\min}}{\theta_j^{\max} - \theta_j^{\min}} \right) \right)^{\gamma_j} \qquad (1)$$

where $\theta_j^{\min} \le \theta_j \le \theta_j^{\max}$ represent the joint variable and its minimum and maximum limits, and the exponent γ_j adjusts the shape of the function (Fig. 5).

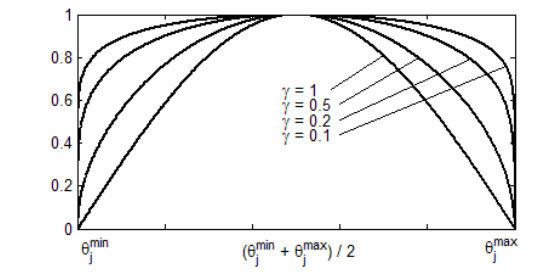

Fig. 5. Shape of Eq.(1) for out-of-range condition of joints.

By multiplying the individual joint functions, it is possible to obtain the value function for a given robot pose, that depends on the six joint angles $\theta_{1\cdots6}$:

$$f_R(\theta_{1\cdots6}) = \prod_{j=1}^{6} f_j(\theta_j) \qquad (2)$$

This method of evaluating robot configuration is flexible and allows specifying additional constraints. For example, a collision detection module may evaluate a given configuration to 0 if there is a colliding situation, to 1 if there is no collision and to intermediar values if two mechanical elements are one in the proximity of the other, and it is preferred, but not mandatory, to avoid this situation. Another example will be a constraint that avoids wrist singularity (see the technical note from Adept, 2007) , where the *Z* axes of the joints 4 and 6 are aligned. The value function would compute the angle between the two vectors and return zero (forbidden) if the angle is smaller than a given threshold, forcing the planning algorithm to avoid the singularity.

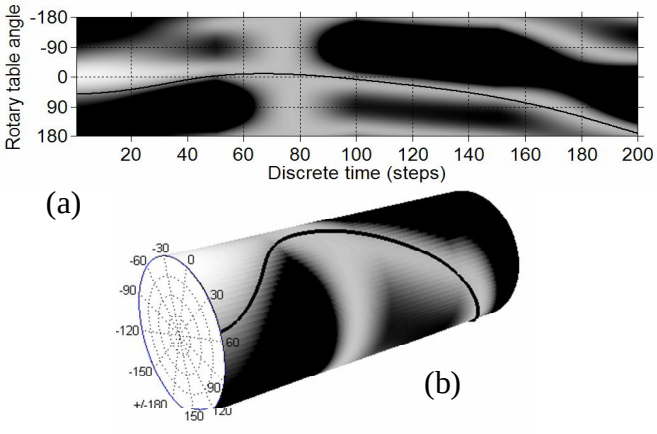

Fig. 6. An example of graylevel configuration map, with a possible solution. (a) 2D map view; (c) Cylindrical view

The value function can be evaluated for every location from a given scanning path, and it may be represented as a *grayscale configuration map*, which is a 2D image having black regions which are forbidden and gray/white regions which are allowed. The planning algorithm will have to choose a path that crosses the map without intersecting the forbidden

regions, and finding a solution that passes on the whiter areas on the map, while maintaining a smooth table motion.

The configuration map can be seen in Fig. 6. Since the angle of the table is represented on the *Y* axis, and the table is able to perform an unlimited number of rotations, the path can wrap around on this axis, and the map can also have a cylindrical representation, like in Fig. 6b.

The proposed solution uses a heuristic planning algorithm, named *ray shooting*, which is able to find a smooth trajectory for the rotary table, that crosses the map in real-time, although the solution found is suboptimal.

The planning algorithm will start from an initial condition vector of the rotary table, which contains the angular position $\theta_R^{(0)}$ and the angular velocity $\omega_R^{(0)}$. The destination angular position 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 $\omega_R^{(n)}$. The planner will compute the angles $\theta_R^{(1)}$ through $\theta_R^{(n)}$. The transition from the discree 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,

$$\omega_R^{(k+1)} = \omega_R^{(k)} + a_R^{(k)} \Delta t \tag{3}$$

$$\theta_R^{(k+1)} = \theta_R^{(k)} + \omega_R^{(k)} \Delta t + a_R^{(k)} \frac{(\Delta t)^2}{2} \tag{4}$$

Since the planner runs on the PC, the planned rotary table motion and also the robot motion are transmitted to the robot controller, and the rotary table part is forwarded to the controller using the communication protocol described in Section 2.2.

In order to maintain a smooth path on the rotary table, a cost function is also assigned to any given path on the configuration map, based on the speed and acceleration used for the table. The penalty due to rotary table motion (high speeds or high accelerations) can be expressed as:

$$C_{a\omega} = \sum_{i=0}^{n-1} \left(k_a \left(a_R^{(i)} \right)^2 + k_\omega \left(\omega_R^{(i)} + a_R^{(i)} \Delta t \right)^2 \right) \tag{5}$$

The coefficients k_a and k_ω are scalar weights for the angular acceleration and speed of the rotary table, and they may be used for tuning the algorithms.

The cost due to advancing on the gray region of configuration map can be defined as:

$$C_{adv} = \sum_{i=1}^{n} \left(1 - f_R \left(\theta_i, T_L^{(i)} \right) \right) \tag{6}$$

Therefore, the cost of a given path is given by:

$$C_{path} \left(\theta_{1..n}, \omega_{0..n-1}, a_{0...n-1} \right) = C_{a\omega} + C_{adv} \tag{7}$$

The pixels on the map, coupled with a third dimension representing the rotation speed, may form a 3D graph on which edges have costs given by $C_{a\omega}$ and C_{adv}.

Since in this application, the speed of the rotary table is much lower than the one of the robot joints, and the robot is not in a singular configuration since they are forbidden on the configuration map, minimizing Eq. (7) will provide an adequate solution for the motion planning problem.

A continuous interval in the position-time-speed space can be mapped to a discrete node (i, j, k) in the graph, using (8). The reverse assigns a point $(\theta_i, t_j, \omega_k)$ in continuous space (the center of the interval) to any node (i, j, k).

$$\begin{pmatrix} \theta_i - \frac{\Delta\theta}{2} \le \theta < \theta_i + \frac{\Delta\theta}{2} \\ t_j - \frac{\Delta t}{2} \le t < t_j + \frac{\Delta t}{2} \\ \omega_k - \frac{\Delta\omega}{2} \le \omega < \omega_k + \frac{\Delta\omega}{2} \end{pmatrix} \Rightarrow \begin{pmatrix} i \\ j \\ k \end{pmatrix} \tag{8}$$

Using this mapping, the problem can be formulated in terms of minimum path and can be solved using the Dijkstra algorithm, presented in Cormen et al. (2001), adapted to the continuous problem presented here after the method described in LaValle et al. (2006). This algorithm is able to find the optimal path for the discrete problem (Fig. 7), which minimizes (7), but it will not necessary be also optimal for the continuous problem. The major disadvantage of this solution is that even for a coarse discretization, the algorithm requires a high amount of memory and computation time. For example, using 100 steps in every dimension of the graph, the graph has 1,000,000 nodes, and the algorithm is not able to find a real-time solution on a standard PC. The Dijkstra algorithm is resolution-complete, and offers a trajectory that can be used as a reference model for the heuristic solution.

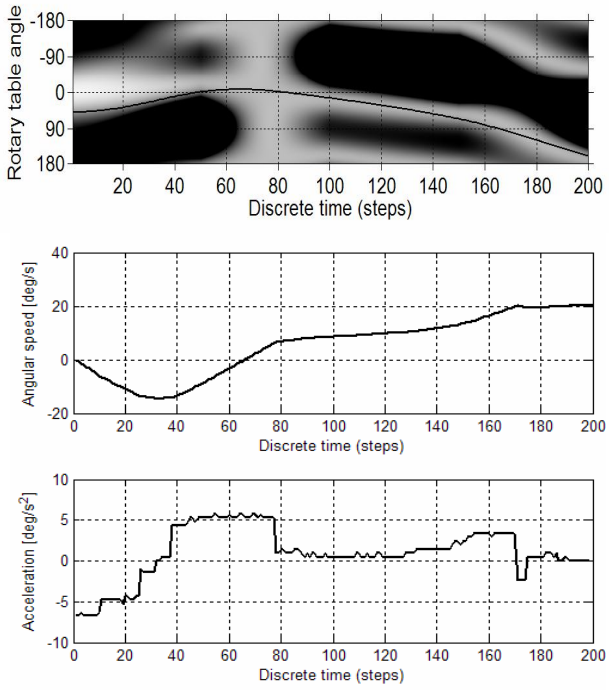

Fig. 7. Path computed by Dijkstra algorithm for a grayscale configuration map. The path does not touch the forbidden areas on the map.
(a) Position; (b) Angular speed; (c) Angular acceleration

3.1 The "Ray Shooting" heuristic algorithm

This algorithm attempts to provide a good, but suboptimal, solution in real-time. At every time step *k*, the planner analyzes *p* time steps in the future, that is, from *k*+1 to *k*+*p*. Over this range, it tries to perform a finite number of trajectories, each one using a constant acceleration value, to ensure smoothness of the planned path. Each trajectory from this set is called a ray. The set of n_a accelerations used is non-uniformly spaced, the choices being closer-spaced at small values. The n_a trajectories are evaluated using (7), and the one having smallest cost is selected. The motion from time step k to k+1 is performed using the acceleration of the chosen trajectory, and the algorithm advances. Since the decision performed at every time step is definitive, the algorithm has a finite amount of computations and it is able to perform the planning in real time.

Algorithm 1: Rotate when out of range

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

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 inrange$\left(\theta_{new}, T_L^{(j)}\right)$ **then**
    $\omega \leftarrow \omega + a\,\Delta t$
    $\theta_R^{(j)} \leftarrow \theta_{new}$
    $\omega_R^{(j)} \leftarrow \omega$
  else
    fall back using *Algorithm 1*

Since there is no guarantee that the heuristic approach will find a solution, there may be a situation when the algorithm may fail, i.e. reach a forbidden area on the map. In this case, a fall back mechanism is provided by Algorithm 1, which pauses the scanning process and rotates the table until a non-zero value is found on the configuration map at current step.

The Ray Shooting algorithm is presented in pseudocode (Algorithm 2), and the `evalpath` subroutine for evaluating a constant-acceleration path is also given.

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 inrange$\left(\theta, T_L^{(j)}\right)$ **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 Ray Shooting algorithm is given in Fig. 8. In the first graph, Fig. 8 (a), the rays can "see" a solution for avoiding the obstacles by turning the table so that decreases, 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 the rotation is smaller than the acceleration needed to avoid the second obstacle on the left, the planner chooses to reverse the rotation direction, choosing the solution from Fig. 8 (c).

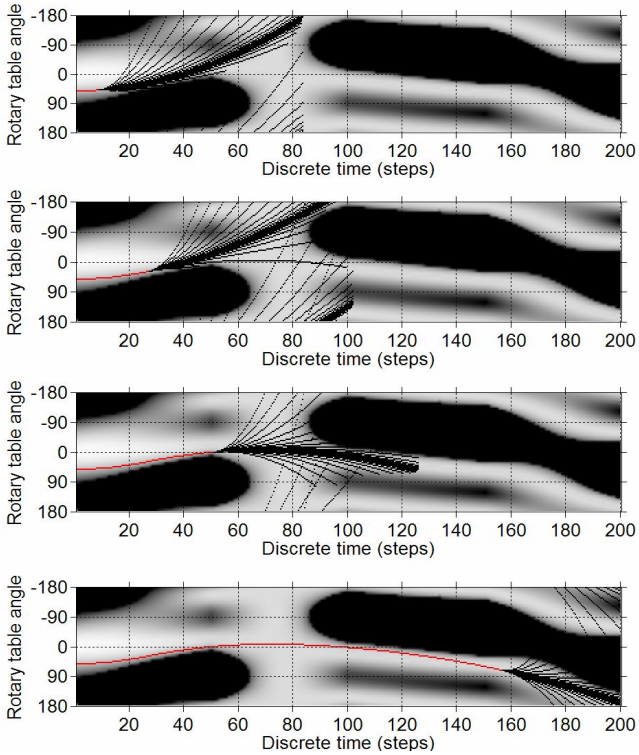

Fig. 8. Ray Shooting example: (a) The rays found a solution by avoiding both obstacles on the left side; (b) One of the rays finds alternative path: avoiding the second obstacle on the right side; (c) The second alternative has lower cost than the first one, and the algorithm chooses it; (d) The algorithm has almost finished.

In the last graph, one can view the planned solution, which is a smooth trajectory for the rotary table, and does not touch the obstacles, therefore the robot arm is not driven close to its joint limits. This solution satisfies the design requirements, is has the same shape as 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 similar, as it can be observed by comparing the magnitudes of acceleration rates from Fig. 9 with Fig. 7. The planned path exhibits some small peaks in the acceleration, whose effects are small and can be removed with a smoothing post-processing routine. There is also a tendency of the planned path coming closer to the obstacles, compared the Dijkstra alternative.

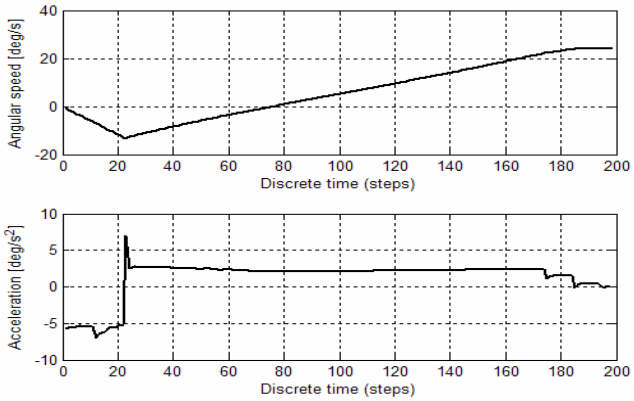

Fig. 9. Speed and acceleration obtained with Ray Shooting.

CONCLUSIONS

This paper presented a heuristic algorithm for a 7-DOF mechanism, which decomposes the 6-DOF scanning paths into independent motions for the rotary table and for the robot arm, performing a 7-DOF inverse kinematics of a redundant manipulator while respecting certain constraints such as speed/acceleration limits for the table, avoidance of kinematic singularities in the mechanism and possibility of integration with a collision detection module. The laser scanning system is currently under development (Fig. 10).

The article also summarized issues regarding the integration of an existing rotary table mechanism into the 3D laser scanning system. A controller for the rotary table was developed using PID loop to drive the table into desired scanning positions, achieving communication with the robot controller using a serial line, and using digital I/O signal to synchronize the laser probe measurements with the motions of the mechanical subsystem.

A possible further development is to use the rotary axis of the existing 4-axis CNC milling machine from Fig. 10 as the workpiece holder, in order to perform in-place inspection of the workpiece, make tool alignment operations and measure the raw workpiece dimensions and shape in order to optimize the milling process.

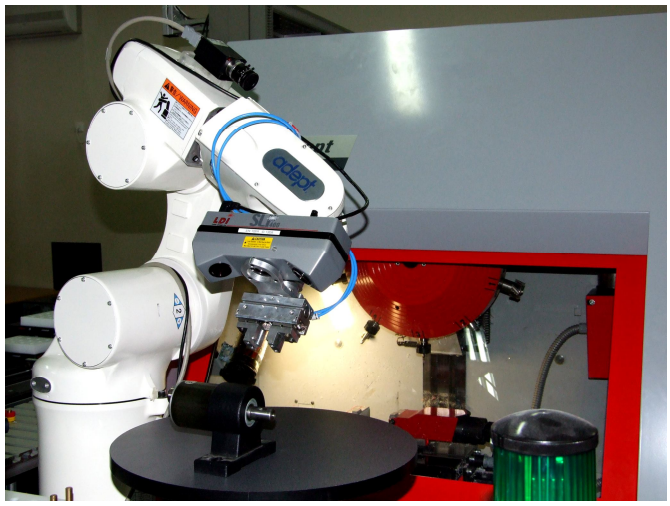

Fig. 10. The 3D scanning probe mounted on the 6-DOF robot arm and a workpiece of interest placed on the rotary table.

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.

REFERENCES

Adept Technology, Inc. (2007), Six-Axis Robot Configuration Singularities. Technical documentation.

Borangiu, Th., Ionescu, F. (2002) *Robot Modelling and Simulation*, Chapter 2: Robot Kinematics. Agir (Ed.), Bucharest.

Borangiu, Th. et al. (2008a), *Integrating a Short Range Laser Probe with a 6-DOF Vertical Robot Arm and a Rotary Table*, Proceedings of RAAD 2008, Ancona, Italia

Borangiu, Th. et al. (2008b), *Modeling and Simulation of Short Range 3D Triangulation-Based Laser Scanning System*, Proceedings of ICCCC 2008, Baile Felix-Spa Oradea

Cormen T. H. et al. (2001), Introduction to Algorithms. Chapter 3: Single Source Shortest Paths. The MIT Press, (Ed.).

Impoco, G. et al. (2005), *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.

LaValle S. M. (2006) et. al, *Planning Algorithms.* Cambridge University Press.

Spong M. W. et al. (2005), *Robot Modeling and Control*, Chapter 3: Forward Kinematics: The Denavit-Hartenberg Convention. John Wiley and Sons, Inc.

Varady T. et al. (1996), *Reverse Engineering of Geometric Models – An Introduction.* Computer-Aided Design, Vol. 29, No. 4, April 1997 , pp. 255-268(14). Elsevier.

Rahayem, M. et al. (2008), *Geometric Reverse Engineering Using A Laser Profile Scanner Mounted On An Industrial Robot*, DAAAM 2008, Tallinn, Estonia