Proceedings of the RAAD 2008
17th International Workshop on Robotics in Alpe–Adria–Danube Region
September 15-17, 2008, Ancona, Italy

Task Redundancy Resolution using Virtual Mechanism Approach

Bojan Nemec and Leon Žlajpah

Robotics Laboratory, Jožef Stefan Institute, Jamova 39, 1001 Ljubljana, Slovenia
E-mail: bojan.nemec@ijs.si

Abstract. The paper deals with the resolution of the kinematic redundancy due to the specific task. In many robot applications, such as brushing, polishing, grinding, sawing etc., there is a task redundancy due to the circular shape of the tool. However, it is difficult to express this type of redundancy in the cartesian coordinate system and consequently difficult to use it on existing robots. The paper proposes a new approach for task redundancy resolution based on the virtual mechanism. The same approach can be used in the control loop as well as for the off-line trajectory planning. The proposed method was verified and implemented in a shoe bottom roughing task.

Keywords. redundant manipulator, control, obstacle aviodance

1. INTRODUCTION

The basic definition of the kinematic redundancy is that the robot has more degrees of freedom that are needed to accomplish the specific task. In the past, redundancy resolution was intensively investigated and is now considered as a mature research topic. Many control schemes were presented which use kinematic redundancy for the optimization of secondary tasks, such as obstacle avoidance, torque optimization, singularity avoidance, etc. All schemes rely on a non-square Jacobian, which maps the joint velocities to the task space, which is in most cases described with the Cartesian coordinate system. If the task redundancy is such that it can be easily described in the Cartesian coordinates, i.e. the task is redundant in one of the Cartesian axes, then the solution is trivial an we can directly apply one of the existing control schemes. But there are tasks, such as brushing, polishing, grinding, sawing, etc. where the kinematic redundancy is hidden. It appears due to the circular shape of the tool, but all 6 Cartesian coordinates are needed to describe and accomplish the given task. Many authors have noticed this type of redundancy (9; 4; 14), but an efficient way how to solve it was not yet proposed. Redundancy of such a task can be described in the polar or toroidal coordinate system. There is always a possibility to express the robot Jacobian in such task coordinates, but this is tedious work and it should be avoided. Another possibility is to define the mapping from the Cartesian coordinates to the task coordinate system. In the paper we will show that this mapping can be very complex. As a alternative we propose the virtual mechanism approach, where the tool is described as a virtual kinematic chain.

2. TASK REDUNDANCY RESOLUTION

Robotic systems under study are n degrees of freedom (DOF) serial manipulators. We consider redundant systems, which have more DOF than needed to accomplish the task, i.e. the dimension of the joint space n exceeds the dimension of the task space m, $n > m$ and $r = n - m$ denote the degree of the redundancy. Let the configuration of the manipulator be represented by the a vector $\boldsymbol{q}_r$ of n joint positions, and the end-effector position (and orientation) by m-dimensional vector $\boldsymbol{x}_r$ of the robot tool center point positions (and orientations). The relation between the joints and the task velocities is given by the following well known expression

$$\dot{\boldsymbol{x}}_r = \mathbf{J}_r \dot{\boldsymbol{q}}_r \tag{1}$$

where $\mathbf{J}_r$ is the $m \times n$ manipulator Jacobian matrix. The solution of the above equation for $\dot{\boldsymbol{q}}_r$ can be given as a sum of particular and homogeneous solution

$$\dot{\boldsymbol{q}}_r = \bar{\mathbf{J}}_r \dot{\boldsymbol{x}}_r + \mathbf{N}_r \xi \tag{2}$$

where

$$\bar{\mathbf{J}}_r = \mathbf{W}^{-1}\mathbf{J}_r^T(\mathbf{J}_r\mathbf{W}^{-1}\mathbf{J}_r^T)^{-1}. \quad (3)$$

Here, $\bar{\mathbf{J}}_r$ is the weighted generalized-inverse of $\mathbf{J}_r$, $\mathbf{W}$ is the weighting matrix, $\mathbf{N}_r = (\mathbf{I} - \bar{\mathbf{J}}_r\mathbf{J}_r)$ is a $n \times n$ matrix representing the projection into the null space of $\mathbf{J}_r$, and ξ is an arbitrary n dimensional vector. We will denote this solution as the generalized inverse based redundancy resolution at the velocity level (10). The homogenous part of the solution belongs to the Jacobian null-space. Therefore, we will denote it as $\dot{\boldsymbol{q}}_n$, $\dot{\boldsymbol{q}}_n = \mathbf{N}_r\xi$.

Now consider the case where the robot Jacobian matrix $\mathbf{J}_w$ is defined in Cartesian (world) coordinate system and the dimension of the Jacobian is $6 \times n$, but the task is described in the another coordinate system, denoted with $\mathbf{p}$. The relation between the task velocities and cartesian velocities can be described as

$$\dot{\mathbf{p}} = \mathbf{J}_t\dot{\boldsymbol{x}}, \quad (4)$$

where $\mathbf{J}_t$ is denoted as a task Jacobian. Let consider the case where the dimension of the task space m is less than the dimension of the Cartesian space, which is 6. It follows

$$\dot{\boldsymbol{x}} = \bar{\mathbf{J}}_t\dot{\mathbf{p}} + \mathbf{N}_t\mu. \quad (5)$$

Here, N_t is the $6 \times m$ task null space matrix and μ an arbitrary m dimensional vector. The redundancy resolution for such case can be expressed as

$$\dot{\boldsymbol{q}}_r = \bar{\mathbf{J}}_w(\bar{\mathbf{J}}_t\dot{\mathbf{p}} + \mathbf{N}_t\mu) + \mathbf{N}_r\xi, \quad (6)$$

where vectors ξ and μ can be used for the secondary task accomplishment. The problem with the above approach is that the task Jacobian $\mathbf{J}_t$ becomes very complex even for a simple relation between then task and the cartesian coordinates. In many cases the analytical solution might even not exist.

We will demonstrate this with the shoe bottom roughing task. In the shoe bottom roughing process, we have to press the shoe bottom against a rotary grindstone and control the contact force and the contact position between the shoe bottom and the grindstone. Robot holds the shoe, while the position/orientation of the tool, grindstone in this case, is fixed. In general, the contact position on the grindstone can be freely chosen. Let define a polar task coordinates system, which describes rotary brush, as shown in the figure 1 The cartesian coordinates are described with the $\boldsymbol{x}_r = (x, y, z, \phi, \theta, \psi)^T$ and the polar coordinates with the $\mathbf{p} = (R, y, \varphi, \phi, \theta + \varphi, \psi)^T$, where ϕ, θ, ψ are the roll, pitch and yaw angles respectively, R is the radius of the polar coordinates and φ is the angle of the polar coordinate. Coordinates x_0, y_0 and z_0 denote displacement of the center of the grindstone from the robot base. Obviously, coordinate φ can be freely chosen, because it does not matter which part of the rotary brush is used for the grinding. In general, also the

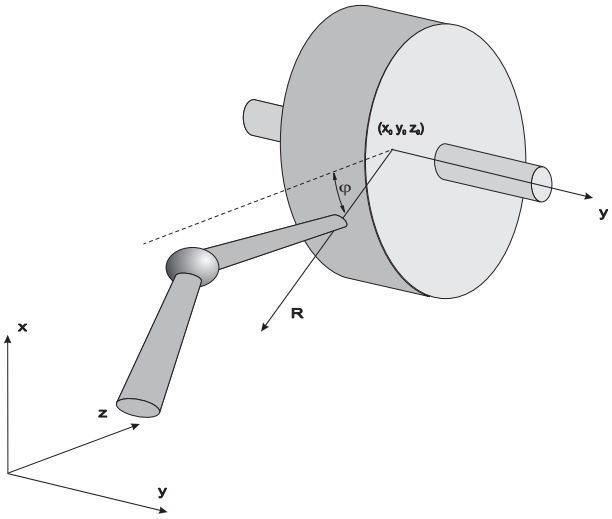

Fig. 1. polar coordinate system of the rotary tool

coordinate y could be freely chosen, but since the grindstone is narrow in most cases, we will treat it as a restricted coordinate. The resulting task Jacobian, where the third line is canceled due to the task redundancy, has the form

$$\mathbf{J}_t = \begin{bmatrix} -\frac{(x-x_0)}{(-z_0+z)\sqrt{\frac{\eta}{(-z_0+z)^2}}} & 0 & -\frac{1}{\sqrt{\frac{\eta}{(-z_0+z)^2}}} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{-z_0+z}{\eta} & 0 & -\frac{x-x_0}{\eta} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where we used the substitute

$$\eta = (z_0 - z)^2 + (x - x_0)^2$$

We can notice that even for the simplest case, the task Jacobian J_t becomes rather complex. For the case when the toroidal shaped brush is used, we were not able to find the analytical solution for the task Jacobian using Matlab symbolical computation toolbox.

An alternative approach we propose to model the tool as a serial kinematic link. Let consider more general case where the robot holds the object to be machined and the work tool is fixed, as illustrated in Fig. 2. In such a case, we can define direct kinematic transformation as

$$\boldsymbol{x}_r + \left[\, \mathbf{R} \,\vdots\, \mathbf{I} \,\right] \boldsymbol{x}_o = \boldsymbol{x}_d + \boldsymbol{x}_v \quad (7)$$

where $\boldsymbol{x}_r$ is the robot Cartesian position and orientations, $\mathbf{R}$ is the robot tool rotation 3×3 dimensional matrix, $\boldsymbol{x}_o$ is the 6×1 vector of the object position and

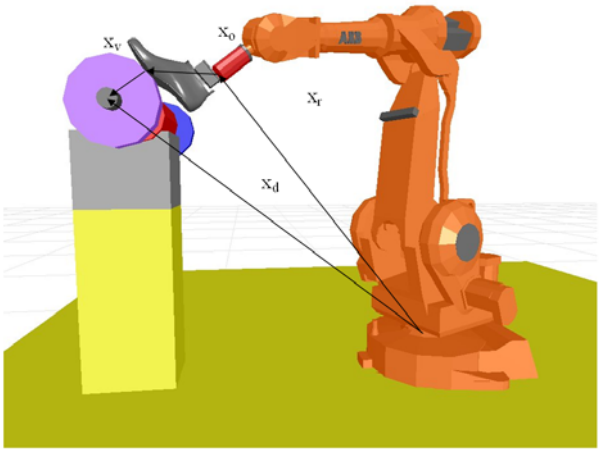

Fig. 2. The case when the robot holds an object and the work toll is fixed

orientation, $\mathbf{R}_v$ is the work tool rotation 3×3 matrix and defines how the work tool is rotated with the respect to the base (robot) coordinates system, $\boldsymbol{x}_v$ is the 6×1 vector of position and orientation of the top of the virtual mechanism and 6×1 vector $\boldsymbol{x}_d$ describes the distance and orientation between the base coordinates system and the work tool coordinate system.

Let consider robot and virtual mechanism as one mechanism with $n+n_v$ degrees of freedom, where n_v is the degree of freedom of the virtual mechanism. We rewrite the Eq. 7 in the form

$$\boldsymbol{x}_r - \left[\,\mathbf{R}\,\vdots\,\mathbf{I}\,\right]\boldsymbol{x}_v = \boldsymbol{x}_d - x_o \tag{8}$$

Jacobian of this new mechanism can be expressed as

$$\mathbf{J} = [\mathbf{J}_r|\mathbf{J}_v] \tag{9}$$

where $\mathbf{J}_r$ is the Jacobian of the robot and $\mathbf{J}_v$ Jacobian of the virtual mechanism is defined as

$$\mathbf{J}_v = \frac{\partial \boldsymbol{x}_v}{\partial \boldsymbol{q}_v}. \tag{10}$$

Note that we assume that the work tool rotation $\mathbf{R}_v$ remains fixed during the execution of the task. Vector $\boldsymbol{q}_v$ of dimension n_vx1 corresponds to the joints of the virtual mechanism.

As we can see, the task space preserves it's dimension, while the joint space is increased with the dimension of the virtual mechanism. This approach has two major advantages. First, we can use existing robot Jacobian, which is assumed to be known. Second, the augmented part of the Jacobian has very simple structure in most cases. As an example we present the direct kinematic transformation for the work cell shown in Fig. 5. The surface of the grinding disc is naturally described with outer surface of the torus, where R and r are the corresponding radius of the brush, as shown in the Fig. 3 and x is the task (Cartesian) coordinate of the whole system.

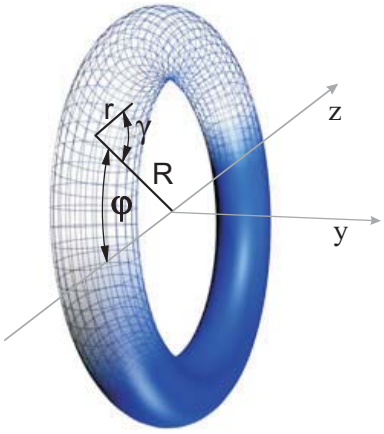

Fig. 3. rotary brush presented as torus

Assuming that the robot tool position and robot Jacobian is known, the forward kinematics can be easily expressed as

$$\boldsymbol{x} = \boldsymbol{x}_r + \begin{bmatrix} s_\varphi\,(R + r\,c_\gamma) \\ r\,s_\psi \\ c_\varphi\,(R + r\,c_\gamma) \\ 0 \\ -\varphi \\ \psi \end{bmatrix}, \tag{11}$$

and the corresponding Jacobian is

$$\mathbf{J} = \begin{bmatrix} & c_\varphi\,(R + r\,c_\psi) & -s_\varphi r\,s_\gamma \\ & 0 & r\,c_\psi \\ \mathbf{J}_r & -s_\varphi\,(R + r\,c_\psi) & -c_\varphi r\,s_\gamma \\ & 0 & 0 \\ & -1 & \\ & 0 & 1 \end{bmatrix}. \tag{12}$$

Here, we used the abbreviation $c_\varphi = \cos(\varphi)$, $c_\gamma = \cos(\gamma)$, $s_\varphi = \sin(\varphi)$ and $s_\gamma = \sin(\gamma)$.

Note that Eq. 7 does not handle orientations correctly, since orientation vectors can not be simply added in general case. Therefore, using Eq. 7 and 8 we obtain an approximate solution regarding the orientation vector. In most cases, the difference between the desired and the real orientation vector is small and is acceptable for operations like brushing and polishing. If orientations are important, we can use equation 7 and 8 for the calculation of positions only, while the orientations have to be

calculated using rotation matrices as follows.

$$\mathbf{R}_o = \mathbf{R}_v{}^T \mathbf{R} \tag{13}$$

Here, $\mathbf{R}_o$ and $\mathbf{R}_v$ are 3×3 rotation matrices describing object rotation against virtual mechanism and virtual mechanism rotation expressed in the robot base coordinate system. The corresponding orientation vector can be than obtained using the transformation of the rotation matrix to the orientation vector described with euler or roll pitch yaw notation. Note that using this 'correct' transformation also rotation part of the Jacobian described by Eq. **??** becomes more complex. On the other hand, when the Jacobian is used in the control loop, only approximate values of Jacobian are needed. Therefore, control algorithm based on Jacobian defined by Eq. **??** gives equal results as control algorithm using the Jacobian calculated from rotation matrices, as described with Eq. 13.

Using this formalism, we can directly apply any control and optimization algorithms developed for the kinematically redundant manipulators.

3. CONTROL

As we mentioned in the previous section, we can directly apply any control algorithm for the kinematically redundant robot. A suitable choice is the control law using generalized inverse-based redundancy resolution at velocity level in the extended operational space. Redundancy resolution at the velocity level is favorable because it enables direct implementation of the gradient optimization scheme for the secondary task. The gradient projection technique has been widely used for the calculation of the null space velocity that optimizes the given criteria. The reason for this is that a variety of performance criteria can be easily expressed as gradient function of joint coordinates. Although the control law using generalized inverse-based redundancy resolution at velocity level can not completely decouple the task and the null space (12; 13; 7; 8), it enables good performance in real implementation. The joint space control law is

$$\begin{aligned} \tau_c = \quad & \mathbf{H}\bar{\mathbf{J}}(\ddot{\boldsymbol{x}}_d + \mathbf{K}_v\dot{\boldsymbol{e}}_x + \mathbf{K}_p\boldsymbol{e}_x - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) + \\ & \mathbf{H}\mathbf{N}(\ddot{\boldsymbol{q}}_{dn} + \mathbf{K}_n\dot{\boldsymbol{e}}_n - \dot{\mathbf{N}}\dot{\boldsymbol{q}}) + \boldsymbol{h} + \mathbf{J}^T\boldsymbol{F} \end{aligned} \tag{14}$$

where Jp is the inertia weighted pseudo-inverse of the Jacobian matrix $\mathbf{J}$, $\mathbf{H}$ is $n \times n$ the inertia matrix, $\boldsymbol{h}$ is n-dimensional vector of the centrifugal, coriolis and gravity forces, $\boldsymbol{F}$ is n-dimensional vector of the external forces acting on the manipulator's end effector and $\mathbf{K}_p$, $\mathbf{K}_v$ and $\mathbf{K}_n$ are the corresponding nxn diagonal matrices with the positional, velocity and the null-space feedback gains. The first term of the control law corresponds to the task-space control τ_x, the second to the null-space control τ_n and the third and the fourth correspond to the compensation of the non-linear system dynamics and the external force, respectively. Here, $\boldsymbol{e}_x = \boldsymbol{x}_d - \boldsymbol{x}$ is the task-space tracking error and $\dot{\boldsymbol{e}}_n = \dot{\boldsymbol{q}}_{nd} - \dot{\boldsymbol{q}}_n$ is the null-space tracking error. $\boldsymbol{x}_d$ and $\dot{\boldsymbol{q}}_{nd}$ are the desired task coordinates and the null space velocity, respectively. The details of the control law derivation can be found in (7).

An attention should be paid on the selection of the inertia of the virtual link. The inertia matrix $\mathbf{H}$ has the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_m & 0 \\ 0 & \mathbf{H}_v \end{bmatrix} \tag{15}$$

where $\mathbf{H}_m$ is the manipulator inertia matrix and $\mathbf{H}_v$ is the diagonal matrix describing the virtual mechanism inertia. Clearly, $\mathbf{H}_v$ can not be zero, but arbitrary small values can be chosen describing the lightweight virtual mechanism. Selection of the inertia matrix of the virtual mechanism affects only the null space behavior of the whole system. Heavy virtual links with high inertia will slow down the movements of the virtual links. Therefore, low inertia the virtual link makes suitable choice. On contrary, we can assume that the virtual links have no gravity and no coriolis and centrifugal forces and the corresponding terms in the vector $\boldsymbol{h}$ can be set to zero. Control law 14 assumes the feedback from all joints, including non-existing virtual joints. There are multiple choices how to provide the joint coordinates and the joint velocities of the virtual link. A suitable method is to build a simple model composed of a double integrator

$$\begin{aligned} \dot{\boldsymbol{q}}_v &= \int \mathbf{H}_v^{-1}\tau_{cv} \\ \boldsymbol{q}_v &= \int \dot{\boldsymbol{q}}_v \end{aligned} \tag{16}$$

where τ_{cv} is the part of the control signals corresponding to the virtual link.

4. NULL SPACE MOTION OPTIMIZATION

The desired extended null space velocities which minimize the given criteria ν, can be obtained using the weighted gradient optimization procedure (1)

$$\dot{\boldsymbol{q}}_n = \mathbf{N}\mathbf{H}^{-1}\frac{\partial \nu}{\partial \boldsymbol{q}}k_o, \tag{17}$$

which assures the best optimization step in the case of the inertia weighted generalized inverse. k_o is a negative constant and defines the optimization step.

The force and the position tracking are usually of the highest priority for a force controlled robot. The selection of the sub-tasks with lower priority depends on the specific application. Nowadays in many industrial applications robot trajectories are computer generated using the CAD model. Verification of such trajectories is problematic especially in small batches customized production (9). Therefore, we should be able to generate the fault-tolerant trajectories. The same problem arises in the use of the service robots, where the motion trajectories are generated using various sensors on-line. The major problems in the automatic trajectory generation are the joint limits of the manipulator, mechanism singularity and the possible collision with the environment. This problem is even more evident if the kinematically redundant mechanism is used. In most cases the motion is not guaranteed to be conservative. Therefore, one successfully executed cycle does not imply the next fault-tolerant free cycle. Fortunately, redundancy can be efficiently used to avoid the addressed problems. Details how to implement the joint limits avoidance and the singularity avoidance can be found in (2; 4; 9). Here, we will address only the collision avoidance problem, because this will be the only criteria used in the experiment described in the next chapter.

Following the idea of the obstacle avoidance using the potential field (5) we define the cost function $\nu = \frac{1}{2}\mathbf{E}d_0^2$, where $\mathbf{E}$ is an $l \times l$ rotation matrix describing the direction of an artificial potential field pointing from the obstacle, l is the dimension of the position sub-space and d_0 is the shortest distance between the obstacle and the robot body. In our case the desired objective is fulfilled if the imaginary force is applied only on the robot joints. In this case we can obtain the cost function gradient in a simple form as

$$\frac{\partial \nu}{\partial \boldsymbol{q}} = (\boldsymbol{d}_1\mathbf{J}^{0,1} + \boldsymbol{d}_2\mathbf{J}^{0,2} + ... + \boldsymbol{d}_{n-1}\mathbf{J}^{0,n-1})\mathbf{E}, \quad (18)$$

where $\boldsymbol{d}_i$ is the vector of the shortest distances between the i-th joint, as illustrated in fig 4 and the obstacle, and $\mathbf{J}^{0,i}$ denotes Jacobian matrices between base (the first index in the superscript) and ith joint (the second index in the superscript) regarding the robot positions only.

5. EXPERIMENT - SHOE GRINDING EXAMPLE

Our experimental setup consists of a 7-DOF robot arm Mitsubishi PA10, mounted on the homonomous mobile platform Nomad XR4000 with 3-DOF. The task of the robot was the shoe bottom roughing (3). In shoe assembly process in order to attach the upper with the corresponding sole, it is necessary to remove a thin layer of material off the upper surface so that the glue can penetrate the leather. To do this, the robot has to press the shoe against the grinding disc with the desired force while executing the desired trajectory. Whole setup is presented in Fig. 5. The mobile platform was not moving during the task and was therefore not considered in the control law. The grinding disc can be presented as a torus and it was described as the virtual mechanism with 2 D.O.F. The corresponding radii were $R = 0.08m$ and $r = 0.01m$. Therefore, the degree of redundancy for this case was 3. There was an moving obstacle under the grinding machine as shown in fig. 5.

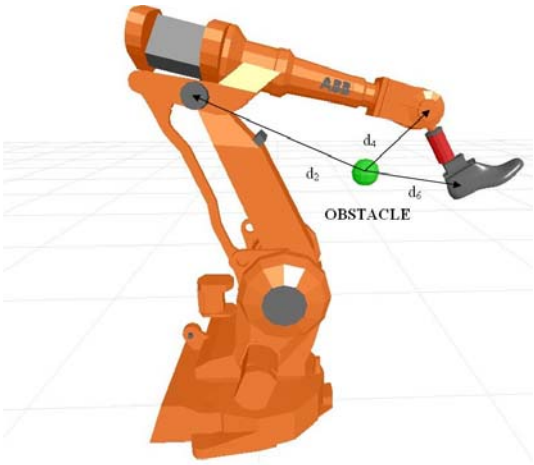

Fig. 4. Obstacle avoidance using potential field approach

Fig. 5. experimental cell for shoe bottom roughing

The obstacle was mounted on the linear axis and was moving randomly during the execution of the task in such a way, that the shoe would collide with this obstacle. The distance between the shoe and the moving

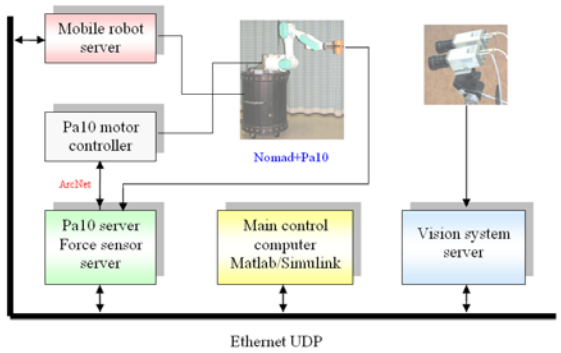

Fig. 6. system block diagram

obstacle was detected with a vision system. The control algorithm was realized in the Simulink using the Real Time Workshop tool. The robot arm was torque controlled using the ArcNet connection between the Michubishi robot joint controller and the PC based control computer. The block diagram of the whole system is outlined in Fig. 6. Each system block represents a PC based computer. Computers are connected trough the ethernet using the UDP protocol. For the demonstration purpose in this example we track a straight line on the shoe bottom. Figure 7 shows time instances of the execution of the task. Time plot of the virtual joints $\boldsymbol{q}(8)$ and $\boldsymbol{q}(9)$ is shown if fig. 8. From figures it can be observed how the robot avoids the obstacle using the task redundancy. Initially, $\boldsymbol{q}(8)$ and $\boldsymbol{q}(9)$ changes slowly only due to the kinetic energy optimization caused by the inertia weighed pseudo-inverse. Starting from the 12th second the robot avoids the obstacle by rotating the virtual joint $\boldsymbol{q}(8)$, while the virtual joint $\boldsymbol{q}(9)$ changes only due to the kinetic energy optimization. Note that very complex trajectories on the shoe bottom can be executed without colliding the environment using the proposed approach.

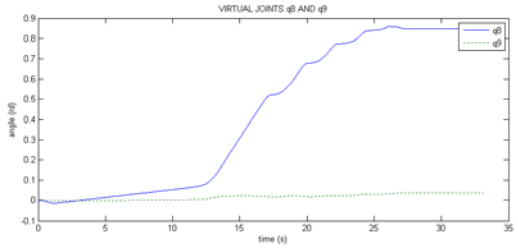

Fig. 8. time plot of virtual angles $\boldsymbol{q}(8)$ and $\boldsymbol{q}(9)$

6. CONCLUSION

In the paper we proposed a new method of solving the kinematic redundancy which arises from the shape of the work tool. The main benefit of the proposed method is the simplicity and efficiency. It can be used on the existing robot controllers with very moderate changes of the control algorithm. We implemented the proposed approach in the shoe bottom roughing task. To show the flexibility of the approach, we placed an obstacle in the work cell. During the task, the obstacle was randomly moving. We have detected the position of the obstacle with the robot vision system. The robot was able to avoid the obstacle and accomplish the given task at the same time using the task redundancy. Another, perhaps for the practical implementation even more attractive possibility is to use the proposed approach in the trajectory generation module instead of in the control algorithm. In this case we can use existing standard industrial robots and we get the advantages of the kinematic redundancy due to circular shape of the tool. This latter approach was successfully implemented in the cell for custom finishing operations in shoe assembly (9). Unfortunately, in this case we have to deny to the real-time trajectory modifications, which can be only implemented in the control loop.

[1] H. Asada and J.-J.E. Slotine. *Robot Analysis and Control*. John Wiley & Sons, (1986).

[2] F. Chaumette and . Marchand. A Redundancy-Based Iterative Approach for Avoiding Joint Limits: Application to Visual Servoing IEEE Transactions on Robotics and Automation, Vol. 17, No. 5, October 2001 719

[3] F. Jatta, L. Zanoni, I. Fassi, I and S. Negri. A Roughing/Cementing Robotic Cell for Custom Made Shoe Manufacture. Int. J. Computer Intergrated Manufacturing, Vol.17, No.7, 2004, p645-652

[4] C. Pholsiri, C. Kapoor, D. Tesar, Manipulator Task-Based Performance Optimization. Proc of DECT'04 ASME Conference, Salt Lake City, 2004

[5] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. of Robotic Research*, 5:90 – 98, (1986).

[6] O. Khatib. A unified approach for motion and force control of robot manipulators:the operational space formulation. *IEEE Trans. on Robotics and Automation*, 3(1):43 – 53, (1987).

[7] B. Nemec and L. Zlajpah. Null velocity control with dinamically consistent pseudo-inverse. *Robotica*, 18:513 – 518, (2000).

[8] B. Nemec and L. Zlajpah. Experiments with force control of redundant robots in unstructured environment using minimal null-space formulation. *Journal of Advanced Computational Intelligence*, 5(5):263 –

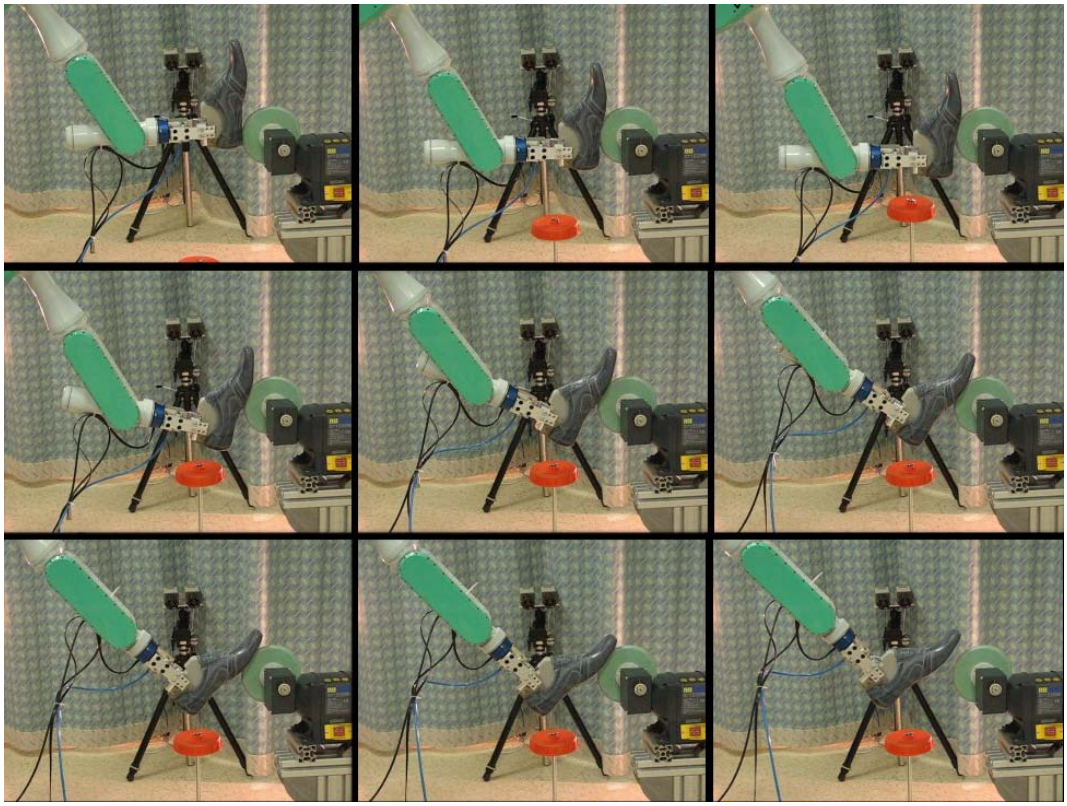

Fig. 7. time instances of the task

268, (2001).

[9] B. Nemec and L Zlajpah. Robotic cell for custom finishing operations. Int. J. Computer Intergrated Manufacturing, 2007, in press.

[10] D. N. Nenchev. Redundancy resolution through local optimization: A review. *J. of Robotic Systems*, 6(6):769 – 798, (1989).

[11] D. Omrcen, L. Zlajpah, and B. Nemec . Autonomous motion of a mobile manipular using a combined torque and velocity control. *Robotica*, 22:623 – 632, (2004).

[12] Y. Oh, W.K. Chung, Y. Youm, and I. Suh. Experiments on extended impedance control of redundant manipulator. In *Proc. IEEE/RJS Int. Conf. on Intelligent Robots and Systems*, pages 1320 – 1325, Victoria, (1998).

[13] J. Park, W. Chung, and Y. Youm. Characterization of Instability of Dynamic Control for Kinematically Redundant Manipulators. In *Proc. IEEE Conf. Robotics and Automation*, pages 2400 – 2405, Washington DC, (2002).

[14] J. Sevier, C. Kapoor, D. Tesar, Benefitting From Underutilized Task Specific Resources for Industrial Robotic Systems. International Symposium on Robotics and Applications (ISORA) Budapest, July 2006.