SIMULATION OF THE KINEMATICALLY REDUNDANT MOBILE MANIPULATOR

Bohumil Honzík

Department of Control, Measurement and Instrumentation
Faculty of Electrical Engineering and Computer Science
Brno University of Technology

Abstract. The mobile manipulator consists of the manipulator arm mounted on the mobile platform. This system is kinematically redundant, formed by the addition of the degrees of freedom of the platform to those of the arm. The task considered in this study is that the gripper of the manipulator is guided by a human operator in real-time. Paper describes the simulation and visualization of such robotic system with the stress on the way how it was implemented rather than on the theoretical background of the problem itself. Presented approach can be applied in the similar fashion to other robotic or mechatronic systems.

1 Introduction

Mobile manipulators, i.e. robotic systems consisting of the manipulator arm mounted on the mobile platform, are attracting significant interest in many areas, such as nuclear plant maintenance, military, bomb disposal, handling chemicals or space exploration. The growing field of service robots demands new systems with increased flexibility. This can be achieved in many different ways. Mobile manipulation – the coordinated use of manipulation capabilities and mobility – is an approach to increase robots flexibility with regard to their motion capabilities.

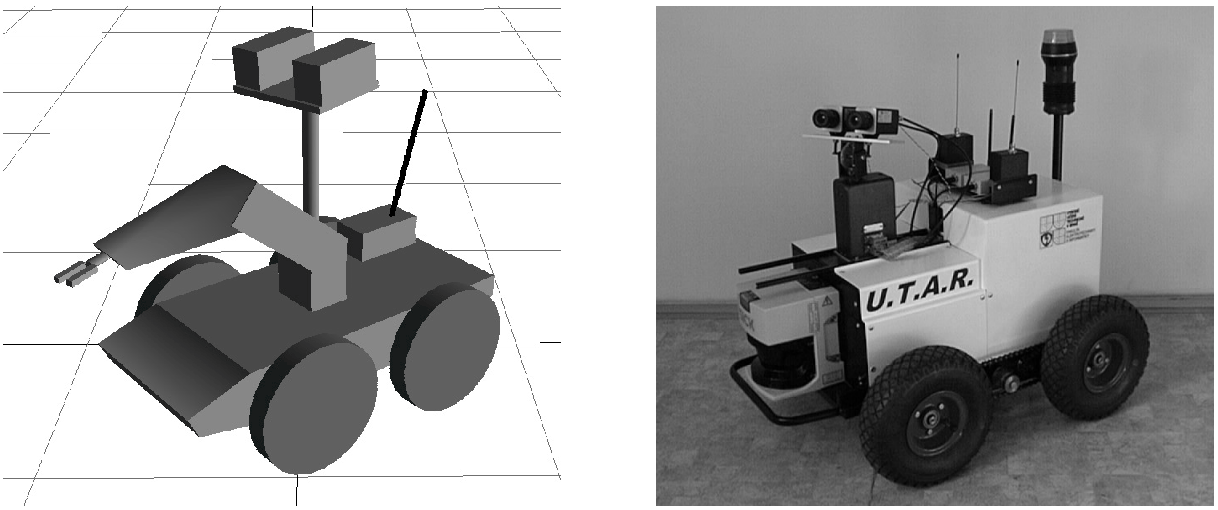
The ultimate goal of robotics is to develop autonomous robots – machines that will be able to accept high-level descriptions of tasks and execute them without further human intervention. Such input descriptions will specify what the user wants to do rather than how to do it. Unfortunately, at present it seems to be impossible with the state-of-the-art artificial intelligence. Nevertheless, in many cases, such as the work in environments inaccessible to or very hazardous for humans, it is not necessary to use an "absolutely autonomous" robot and the solution can be found using the telepresence approach. Thus, perception, decision making and control may be divided into two levels: the high-level tasks can be made by the human (global planning, vision and scene analysis, gripper control), while the low-level processes may be done by the computer (redundancy resolution, collision avoidance).

The basic concept of such mobile manipulator is shown in Fig. 1a. It consists of the wheeled mobile platform, which carries the manipulator arm and a vision system (cameras). DCMI is currently developing a mobile robot UTAR (Universal Telepresence and Autonomous Robot). It consists of the four-wheels skid-steered mobile platform, which carries the pan/tilt robotic head with two-camera stereovision system. Robot is controlled using joystick by the human operator, who is wearing the head mounted display. Stereovision system guarantees the 3D immersion to the operator, which simplifies the estimation of distances between observed objects. Communication is realized via radio modules. UTAR is suitable not only for indoor, but also for outdoor missions in relatively hard terrain.

UTAR is primarily intended for telepresence applications. At the moment it can be used for inspection tasks. The robot is also equipped with the laser scanner, measuring the distance to obstacles in the halfplane in the front of the robot. This information can be used in computer supported telepresence or in the autonomous mode.

UTAR will be equipped with the robotic arm for performing of various manipulation tasks. In the future, other sensors will be added and artificial intelligence techniques will be implemented

to allow the system to perform some tasks autonomously and will also help the operator to solve some difficult situations, e.g., when the whole scene cannot be viewed using the vision system and collisions with obstacles should be avoided. Detailed description of the UTAR system can be found in [2], its current state is shown in Fig. 1b.



(a) (b)

Figure 1: Idea scheme of the UTAR system (a), and its current state (b)

2 Mathematical Background

Mobile manipulator considered in this study consists of the 6-DOF (degrees-of-freedom) arm mounted on the 3-DOF mobile platform. This is kinematically redundant mechanism, since it has more DOF (nine in this case) than required to perform the given task (unique definition of the task in 3D can be realized using six coordinates). Process of calculating the inverse kinematics of the redundant manipulator is referred to *redundancy resolution*.

Kinematic equations that relate locations of the gripper to the corresponding joint angles are mostly highly non-linear and there is no analytical closed-form solution of the inverse kinematics for a general robot structure. This is why the inverse kinematics is solved at the velocity level:

$$\dot{\mathbf{q}} = \mathbf{J}_{\mathrm{H}}^{+}\,\dot{\mathbf{x}} + \left(\mathbf{I} - \mathbf{J}_{\mathrm{H}}^{+}\mathbf{J}\right)\left(-\frac{\partial h(\mathbf{q})}{\partial \mathbf{q}}\right)^{\mathrm{T}} \tag{1}$$

where $\dot{\mathbf{q}}$ is the $[n \times 1]$ joint velocity vector, $\mathbf{J}$ is the $[m \times n]$ Jacobian matrix, $\mathbf{J}_{\mathrm{H}}^{+}$ is the inertia weighted Jacobian pseudoinverse defined as $\mathbf{J}_{\mathrm{H}}^{+} = \mathbf{H}^{-1}\,\mathbf{J}^{\mathrm{T}}\left(\mathbf{J}\,\mathbf{H}^{-1}\,\mathbf{J}^{\mathrm{T}}\right)^{-1}$, $\mathbf{H}$ is the manipulator inertia matrix, $\dot{\mathbf{x}}$ is the $[m \times 1]$ gripper velocity vector, $\mathbf{I}$ is the identity matrix and $h(\mathbf{q})$ is the scalar function, which becomes large in configurations close to 6-DOF manipulator workspace boundary, singular configurations and is also used for collision avoidance. Detailed explanation of the inverse kinematics is out of the scope of this paper, but can be found in [1].

3 Simulation

Simulation of the redundancy resolution algorithm is performed in MATLAB. To speed up the computation no Simulink model is used and everything is programmed in the form of pure MATLAB m-files. The key part of the program code is the m-file corresponding to (1). This function is then integrated using an explicit Runge-Kutta integration method (`ode23`). The input of the system is vector of gripper velocities $\dot{\mathbf{x}}$, the output is the vector of joint coordinates $\mathbf{q}$.

To better understand the behaviour of the control algorithm it is advantageous to run the simulation on-line, rather than off-line. Since the integration algorithm used is a variable-step

one, simulation is running in pseudo-real-time only. Gripper velocities may be controlled by standard computer input devices – mouse or joystick.

Usually, the result of the integration process is not available until the ODE solver finishes the computation. Since for the on-line visualization the actual data are required, the `OutputFcn` property of the ODE solver is used. Then, the visualization function is called after computing each output point.

4 Visualization

For simulations in the field of robotics and mechatronics the visualization of the computed results is crucial. MATLAB itself offers powerful tools for visualization both in 2D and 3D. For drawing of robotic systems MATLAB patch graphics objects can be used. Thus, such manipulator consists of several polyhedra, cylinders, spheres etc. which are drawn during the initialization of the whole scene. Then, the particular vertices of all patches are translated and rotated accordingly every time the new output values are computed. For this purpose the `set` command is used, which is changing the properties `XData`, `YData` and `ZData` of the particular patch objects.

Although the patch objects make possible to visualize complicated 3D shapes, there are some facts, that complicate the on-line drawing and animation. First of all, the coordinates of all the particular vertices of every graphics object that is moving have to be recomputed at every integration step. Although this can be relatively easy made using the method of homogeneous transformations, all the transformation matrices and other necessary computations must be programmed by the user, which could be difficult for more complicated objects consisting of many patches. Moreover, every change of the model structure needs thorough studying of the corresponding source code and is very inconvenient. Another disadvantage of this approach is the fact that the rendering is slow, especially for large objects. Even if the OpenGL renderer is chosen, animation is too slow and, in addition, the window size is limited.

Another possibility how the visualization of various 3D objects can be realized is the use of the Virtual Reality Toolbox, which is extending graphics capabilities of MATLAB, [3]. This toolbox is utilizing standard VRML technology, which is an open, text-based, WWW-oriented format, allowing description of 3D scene, sound, internal actions, WWW anchors etc. It can be viewed in any WWW browser with the corresponding VRML plug-in installed.

The principle of operation is as follows: First, the VRML model of the object and its environment must be created. This can be accomplished by using of any plain text editor. Nonetheless, writing the VRML code by hand requires deep knowledges of the VRML syntax and is a lengthy process. It is easier to build the scene in any 3D modeler and export it into VRML format or create the model directly using native VRML authoring tool. Once the model is created, the object of class `vrworld` within MATLAB environment is created. Then, it is opened and viewed in the WWW browser. The connection between MATLAB and the VRML enabled browser via TCP/IP protocol is established. A flexible MATLAB interface to the VRML virtual reality world provided by Virtual Reality Toolbox allows now the easy control of selected model parameters.

The above mentioned approach will be now presented on an easy example. First, the model of the mobile manipulator, consisting of the four-wheeled mobile platform with the 6-DOF manipulator arm attached, is created. As the authoring tool the native VRML editor VRealm Builder is used. The screenshot of this editor is shown in Fig. 2. On the left, the hierarchical tree structure of objects – *nodes* – is shown. Each node contains a list of *fields*, which hold *values* that define parameters for its function. Nodes can be placed in the top level of the tree or as children of other nodes in the tree hierarchy. When a value in a field of certain node is changed, all nodes of its sub-tree will be affected. This allows to define relative positions inside complicated compound objects. The design and creation process of the VRML model is simplified by the friendly graphical user interface. Placing, moving and changing orientation or size of objects can be done using the mouse. Only the basic knowledge of the VRML language principles is necessary.

For example, the mobile manipulator in Fig. 2 consists of the parent node `PLATFORM`, which contains children node `WHEELS` with subordinated nodes `WHEEL_1`, `WHEEL_2`, `WHEEL_3`, `WHEEL_4`, and children node `LINK_1` with subordinated nodes `JOINT_1` and `LINK_2`. Node `LINK_2` contains children nodes `JOINT_2` and `LINK_3` and so on. Now, when changing the position and orientation of the node `PLATFORM`, all subordinated nodes – wheels and links are moving accordingly. Similarly, moving the `LINK_2` moves also the `LINK_3`, `LINK_4` etc. As the VRML world is opened in the WWW browser, particular fields of the VRML nodes can be accessed and modified as common MATLAB objects and their properties.

5 Presentation on the WWW

Since the connection between MATLAB and the WWW browser is achieved through the TCP/IP protocol, it is possible to watch simulated worlds not only on the same computer where MATLAB is running, but also on other computers, connected locally or over the Internet. This allows the presentation of the simulations on the World Wide Web. Disadvantages of this approach are that MATLAB must be running all the time on the host computer, and that the fluency of the animation depends on the network transfer capacity and load.

Another solution of the presentation on the WWW is the utilization of the VRML language capabilities. By means of *time sensors* and *position interpolators* it is possible to incorporate the data obtained during the simulation directly into the VRML model. Moreover, certain level of interactivity can be added to the model, allowing the user to start and stop the animation, manipulate the nodes etc. Such VRML model is first downloaded in the WWW browser and after that viewed locally. This is in fact *off-line viewing* of the simulation results, compared to the observation of the *on-line simulation* run, which was described above.

6 Conclusions

MATLAB is a powerful tool for simulations in the field of robotics. It allows easy implementation of various control algorithms with a little effort. Although the visualization can be made using the standard MATLAB objects and functions, Virtual Reality Toolbox significantly simplifies this process. The design and any modification of the 3D model is much faster and easier, the rendering speed and quality is considerably better. The sideeffect of the VRML technology used by the toolbox is the fact, that the simulation can be viewed also on other computers, connected to the network, which allows the on-line presentation of the simulation on the WWW. Nonetheless, in some situations it is better for the WWW presentation to incorporate the output data directly into the VRML model, which makes the animation fluent and independent on the network traffic.

References

[1] B. Honzík and F. Zezulka. Redundancy resolution techniques for mobile manipulators. In *Proc. of the 2nd Int. Symp. AMPST*, pages 79–89, Bradford, UK, March 1999.

[2] L. Žalud, B. Honzík, and F. Šolc. Human–machine interface of the UTAR system. In *Proc. 11th DAAAM International Symposium*, Opatija, Croatia, October 2000.

[3] *Virtual Reality Toolbox User's Manual*. Humusoft s.r.o., Prague, 2000.

Contact: Bohumil Honzík, ÚAMT FEI VUT, Božetěchova 2, 612 66 Brno,
e-mail: `honzikb@dame.fee.vutbr.cz`, url: `http://www.fee.vutbr.cz/UAMT/robotics`

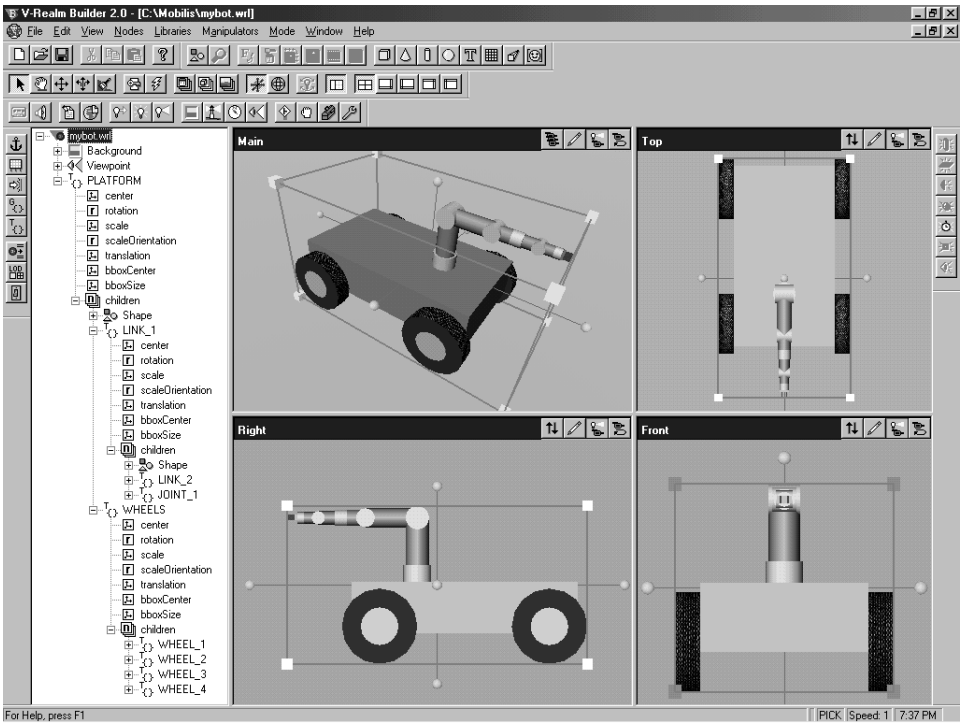


Figure 2: Creation of the VRML model in VRealm Builder

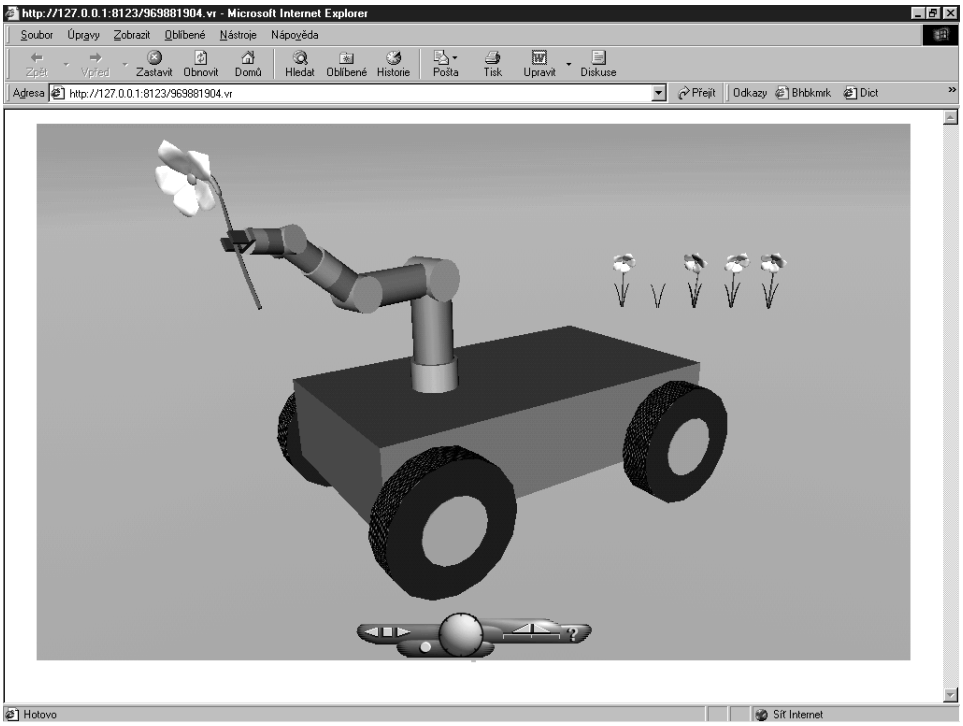


Figure 3: Mobile manipulator in virtual environment – viewed in the WWW browser