Robot Visualization System for Windows (RVS4W)

User's Manual

W. Khan, H. Zhuang and J. Angeles

Centre for Intelligent Machines (CIM)
Department of Mechanical Engineering
McGill University, Montréal, Québec, Canada

June 11, 2007

Abstract

Visualization plays an important role in the design of new robots or work environments. Robot Visualization System for Windows (RVS4W) is an updated cross-platform version of its predecessor, the Robot Visualization System (RVS).

RVS was developed at the Centre for Intelligent Machines, McGill University, in the 1990s. It was developed on IRIX, the UNIX dialect of Silicon Graphics Inc. (SGI). Since IRIX is native to SGI workstations, RVS does not run on Intel-based PCs. Therefore, the source code of RVS had to be modified in order to use this tool on an Intel-based PC. Based on the two important attributes of RVS4W, i.e., open source and platform-independent, the proper development tools were chosen.

RVS4W is an updated version of RVS. The user interface was rewritten from scratch. We have made an effort to keep the user interface consistent and user-friendly. The existing kinematics engine was also debugged and improved. RVS4W incorporates many new features including routines to evaluate the characteristic length, maximum reach, optimum posture—for minimum condition number—and robot conditioning.

This manual introduces the user to the window-driven environment of RVS4W.

Acknowledgements

John Darcovich, the author of RVS, is the brain behind this project. RVS4W, like RVS, was made possible thanks to the sustained support of NSERC to the

third author's research.

Disclaimers

RVS4W and RVS are intended for research and education use only. The authors cannot be held responsible for any errors or damage caused by the use of this software packages.

Contents

1 Introduction

RVS4W is a 3D visualization tool to be used to determine the feasibility of a new robot or of a new path plan. This tool is an updated version of its IRIX predecessor, RVS (Darcovich, Angeles, Montagnier and Wu, 1999). In the *skeleton mode*, the Denavit-Hartenberg (DH) parameters of the robot at hand are read from a data file and rendered as a kinematic chain using a standard set of *skeleton primitives*. This set includes an L-shaped link with dimensions in proportion dictated by the corresponding DH parameters, a revolute joint, a prismatic joint and an end effector (EE). No geometric details on the actual shape of the links is provided in the skeleton mode.

Full rendering of the actual robot can also be made. Of course, this requires, that a database with the details of the robot-link geometry be supplied by the user.

1.1 RVS4W Features

Currently, RVS4W supports the following main features:

- Forward kinematics
- Inverse kinematics
- Individual joint limits
- Posture pre-storing
- Trajectory-tracking in joint space
- Trajectory-tracking in Cartesian space
- Maximum-reach, characteristic length, optimum-posture evaluation for a given robot
- Robot-Jacobian evaluation at a given posture
- Robot-conditioning evaluation at a given posture
- A snapshot of the scene to be saved in Encapsulated PostScript (EPS), Portable Document Format (PDF) or Scalable Vector Graphics (SVG) formats
- Display of reference frame, the moving frames and the EE frame
- Creation and visualization of work environments

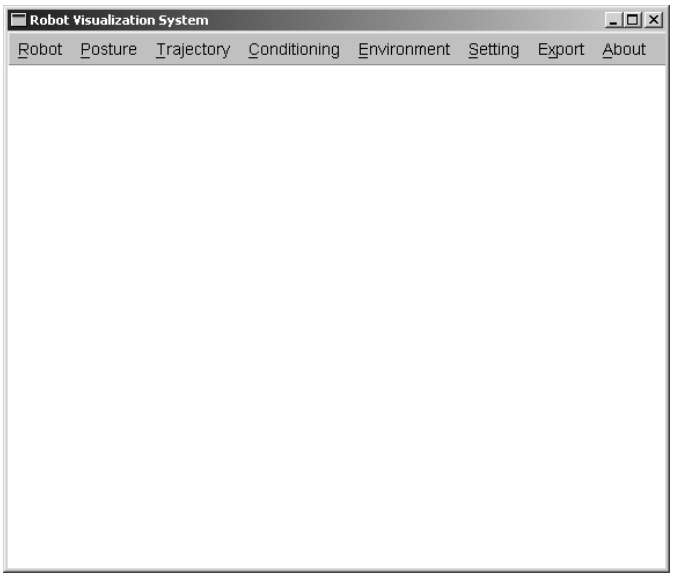

Figure 1: RVS4W Main Window

1.2 Document Conventions

Conventions used throughout the manual:

- `Typewriter` fonts are used to represent RVS4W commands, shell commands, directory paths, and code (C++ scripts).
- Uppercase *ITALICS* fonts are used to represent file names.

2 RVS4W Basics

This section introduces the user into the basic features of RVS4W. After reading this section, the user should be able to load RVS4W and use its main functions.

2.1 Starting RVS4W

To start RVS4W, go to ⟨`install directory`⟩`\rs` and double-click *RVS4W.EXE*. The RVS4W main window will appear as shown in Fig. 1. RVS4W commands are grouped in the main menu displayed at the top of the RVS4W window. To activate a command, navigate through the menu system and select the required menu item. The corresponding dialog is invoked and displayed on the screen. *Notice that, to activate some commands, the user must load a robot in RVS4W.*

2.2 Mouse Controls

Once a robot is loaded, the user can rotate, zoom and translate the rendering.

- Press and hold the left-mouse-button and drag the mouse to rotate the workspace with respect to the viewer.
- Press and hold the middle-mouse-button and drag the mouse upwards (downwards) to zoom-in (zoom-out).
- Press and hold the right-mouse-button and drag the mouse to translate the workspace in the corresponding direction.

2.3 Units

With regards to the geometric dimensions of the robot and the work environment, *RVS4W does not incorporate a unit manager*. The user is free to decide on the units of length while interacting with RVS4W. Whenever units of length are needed in this document, the legend usu, for *user-specified-units*, is included. The user is encouraged to indicate the selected units in the corresponding `Description` text box (Sec. 4) as an *aide-mémoire*. All angles must be specified in *degrees*.

2.4 The `Robot` Menu

2.4.1 `New Robot`: Creating a robot

Select `Robot > New Robot` (Fig. 2) to invoke the `Create Robot` dialog (Fig. 3). Next, input the desired number of links for the new robot and press `OK`. The `Create Robot - Parameters` dialog (Fig. 4) is displayed. This dialog has four main groups:

1. `Filename`: Input the name of the new robot.
2. `DH Parameters`: Input the corresponding robot dimentions using the DH convention described in (Angeles, 2007), i.e., **a**, **b**, **al** (α) and **th** (θ). By default, a joint is taken as revolute. To specify a prismatic joint check the corresponding `Prismatic` check box. Notice that RVS4W does not support the conditioning evaluation for robots with prismatic joints.
3. Joint limits: To specify the physical limits of a joint, check the corresponding `Limits` check box and input the minimum and maximum values in the `min` and `max` text boxes. The limits must be specified in degrees for the revolute joint, in agreement with the DH convention adopted at the outset. For a prismatic joint, the limit must be specified in usu.
4. `Description`: Input the description of the robot in this text box.

Click `Save` to save the robot and render its skeleton on the screen. Click `close` to exit the dialog.

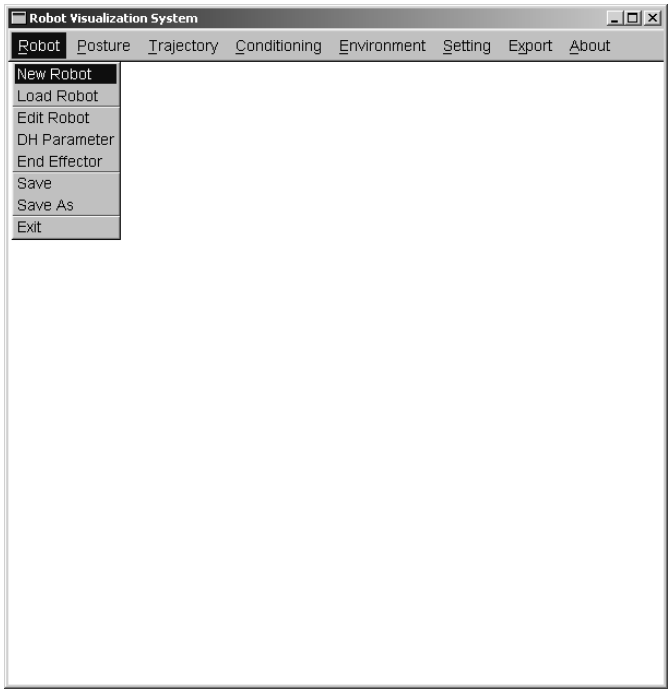

Figure 2: Menu: `Robot`

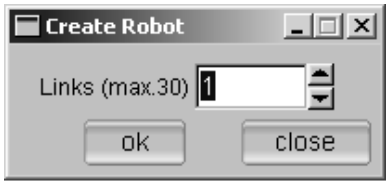

Figure 3: The `Create Robot` dialog

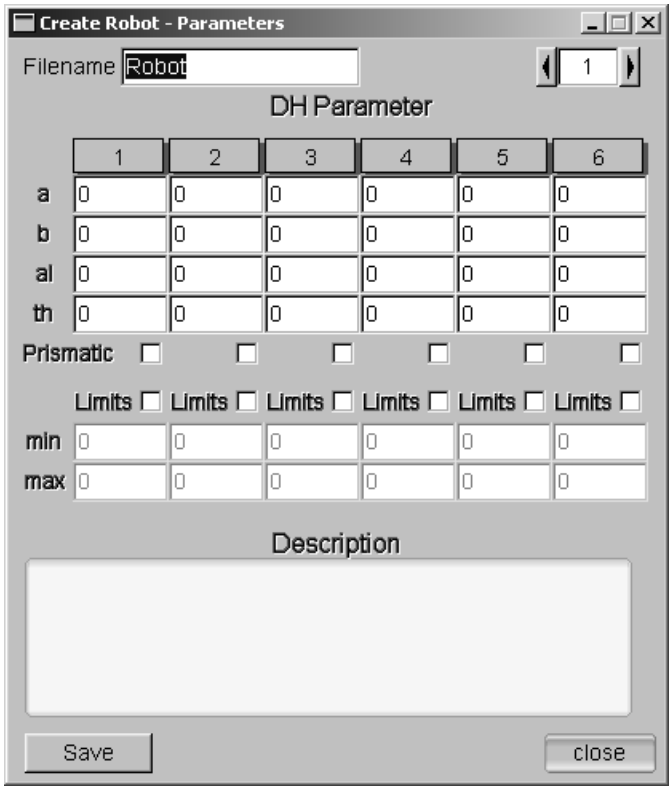

Figure 4: The `Robot Parameter` dialog

2.4.2 `Load Robot`: Loading an existing robot

Select `Robot > Load Robot` from the main menu (Fig. 2) to display the `Load Robot` dialog (Fig. 5). In this dialog a list box displays the list of existing robots. The user has several options:

- Select a robot to view its description
- To load a robot, choose the desired robot and click `Load`
- To delete a robot, select the robot to be deleted and click `Delete`. The confirmation dialog appears. Click `OK` to delete the robot or `Cancel` to keep it. Notice that when a robot is deleted, all the relevant data are erased from the system. This includes the architecture, configuration and trajectory files of the selected robot. This means that, if the user is to try a given trajectory on various robots, then the user must save the trajectory in a separate data file, then copy-and-paste it on the data file of every robot to be assigned this trajectory for tracking.

Figure 5: The `Load Robot` dialog

2.4.3 `Edit Robot`: Editing the robot

Selecting `Robot > Edit Robot` (Fig. 2) brings up the `Edit Robot` dialog on the screen (Fig. 6).

- To change the display geometry, click the appropriate radio button: `Up`, `Down` or `Full`. *The full-geometry option is available only for those robots whose detailed link-geometry has been previously input.*
- To modify the robot parameters, click the appropriate link button. This brings up the corresponding `Link Settings` dialog. The reader is referred to Subsection 2.4.4 for details on the `Link Settings` dialog.

Click `close` to exit the `Edit Robot` dialog.

2.4.4 The `Link Settings` dialog

The `Link Settings` dialog (Fig. 7) can be accessed through the `Edit Robot` dialog. There is a separate `Link Settings` dialog associated with each link of the robot. The link number is displayed at the top of the dialog. The dialog displays a set of options organized in three groups.

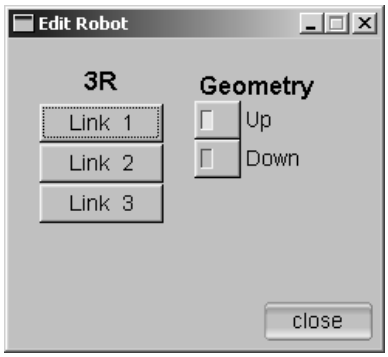

Figure 6: The `Edit Robot` dialog

- `Joint`: To select the required joint-type, revolute or prismatic, check the corresponding check box.
- DH Parameters: The DH parameters of a link that can be edited depend on the selected joint-type. To change the DH parameters of the link, input the desired value or use the arrow buttons to increase or decrease the value of the corresponding parameter. The robot is updated dynamically, i.e., as soon as the new value is input in the spinner.
- Joint Limits: To specify joint limits, check the `Limits` check box and enter the maximum and the minimum joint values.

Click `close` to exit the `Link Settings` dialog.

2.4.5 Displaying the DH parameters of the current robot

To display the DH parameters, select `Robot > DH Parameter` (Fig. 2). The `DH Parameters` dialog will appear on the screen (Fig. 8) showing the relevant parameters of the loaded robot. Click `close` to exit the dialog.

2.4.6 `End-Effector`: Selecting an end-effector

To select the required end-effector choose `Robot > End-Effector` (Fig. 2). This activates the `End Effector` dialog (Fig. 9). The available end-effectors for the loaded robot are displayed in a list box. Select the desired end-effector and click `close` to hide the dialog.

2.4.7 `Save`: Saving a robot

Select `Robot > Save` to save the current robot. Notice that the current posture of the robot is saved as the default posture. The default posture of a robot is one that is used to render the robot when the robot is loaded.

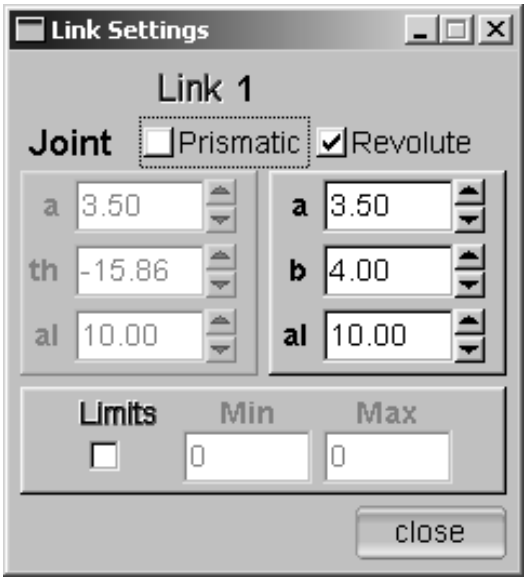

Figure 7: The `Link Settings` dialog

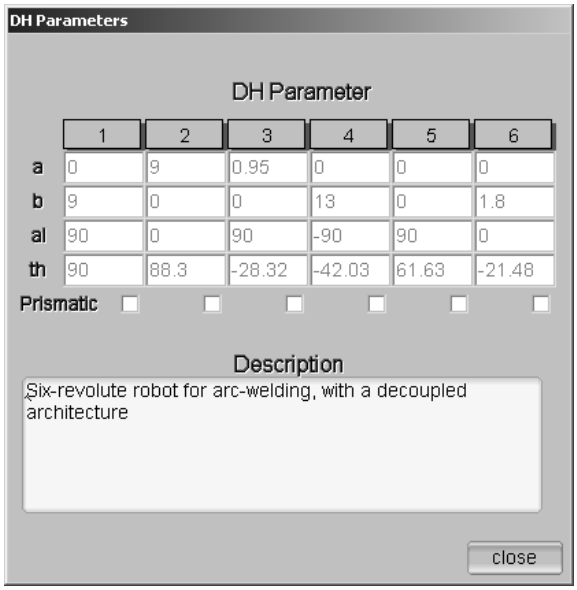

Figure 8: The `DH Parameters` dialog

Figure 9: The `End-Effector` dialog

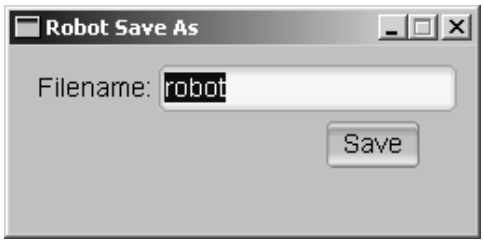

Figure 10: The `Robot Save As` dialog

2.4.8 `Save As`: Saving a robot with another name

To save the robot with a different name, select `Robot > Save As` (Fig. 2). The `Robot Save As` dialog will appear on the screen (Fig. 10). Input the new filename and click `Save`. The robot will be saved under the new filename.

2.4.9 `Exit`: Exiting RVS4W

To exit RVS4W, select `Robot > Exit`.

2.5 The `Posture` Menu

2.5.1 `Joint Limits`: Setting joint limits

RVS4W allows the user to set joint limits. When active, the joint limits are considered in the forward and inverse kinematics of the robot. Select `Posture > Joint Limits` (Fig. 11) to activate the `Joint Limits` dialog (Fig. 12).

- To activate joint limits, check the `Joint Limits On` check box.

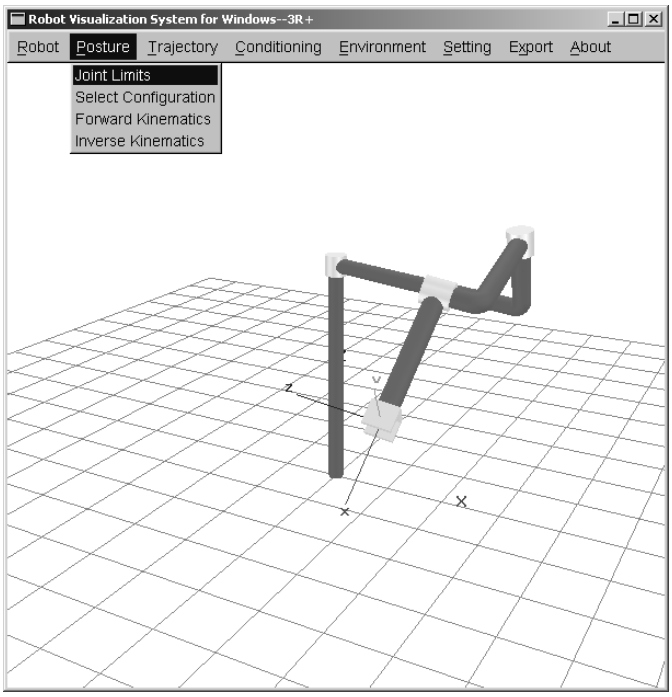

Figure 11: Menu: Posture

- To deactivate joint limits, check the `Joint Limits Off` check box.
- To specify joint limits, input the maximum and minimum values in the corresponding joint-limit spinner.
- To exit the dialog, click `close`.

2.5.2 `Select Configuration`: Moving the robot to a saved posture

The robot can be moved to previously saved postures. RVS4W provides two postures automatically:

1. `<robotname>.Characteristic`: The *characteristic* posture is that with the best conditioning. See Subsection 2.7 for further details.
2. `<robotname>.Maxreach`: A posture at the maximum reach of the robot.

It is also possible to save a robot posture. Please refer to Subsection 2.5.3 below to find how this can be done.

To move to a previously saved posture, select `Posture > Select Configuration` from the menu (Fig. 11). The `Select Configuration` dialog will popup on the screen

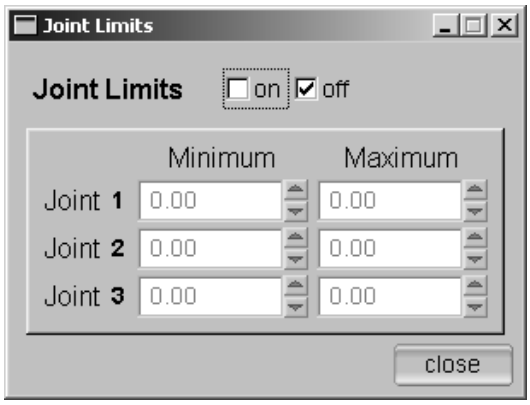

Figure 12: The `Joint Limits` dialog

(Fig. 13). All the saved postures of the loaded robot are displayed in a list box. The `Select Configuration` dialog offers three functions:

- To move the robot to a saved configuration, select the desired configuration from the list box.
- `Spline`: When this option is selected, the robot translates from the current posture to the selected posture.
- `Jump`: Select this option if you want the robot to jump to the selected configuration.

Click `close` to hide the `Select Configuration` window.

2.5.3 `Forward Kinematics`

To change the posture of the robot by changing the value of the joint coordinates (forward kinematics) or to save the current joint configuration, click `Posture > Forward Kinematics` (Fig. 11). This brings forth the `Forward Kinematics` dialog (Fig. 14). The symbol th_i represents the joint number. Next to the joint number is the respective joint-coordinate value.

- To change the value of the joint coordinate, do one of three actions:
  1. Input the new value of the joint coordinate in the appropriate text box.
  2. Use the arrow buttons to increase or decrease the joint-coordinate value.
  3. Drag the horizonal scroll bar of the corresponding joint.
- To save the joint-coordinate values of the current posture, input the desired filename in the `File` text box and click `Save`.

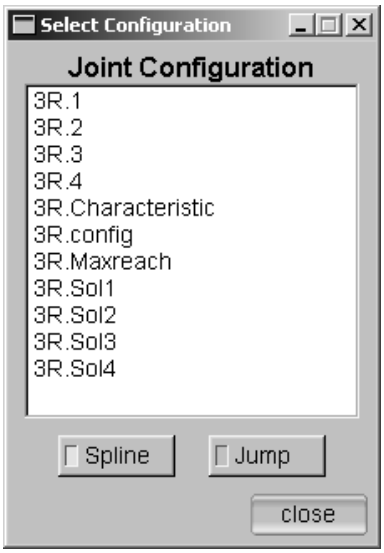

Figure 13: The `Select Configuration` dialog

- To exit the dialog, click `close`.

2.5.4 `Inverse Kinematics`

To change the pose of the end effector and hence change the posture of the robot (inverse kinematics), select `Posture > Inverse Kinematics` from the menu (Fig. 11). The `Inverse Kinematics` dialog will popup on the screen (Fig. 15).

- `Position`: To change the position of the end-effector, input the desired value in the `x`, `y`, and `z` text boxes or use the arrow keys to increase or decrease the value.
- `Orientation`: To change the orientation of the end-effector, input the desired value in the Rx, Ry, and Rz text boxes or use the arrow keys to increase or decrease the value.
- To view the orientation matrix, click on the `Q` button.
- To view the robot Jacobian, click on the `J` button.
- Click `close` to exit dialog.

An outline of the inverse kinematics algorithm is available in the Appendix.

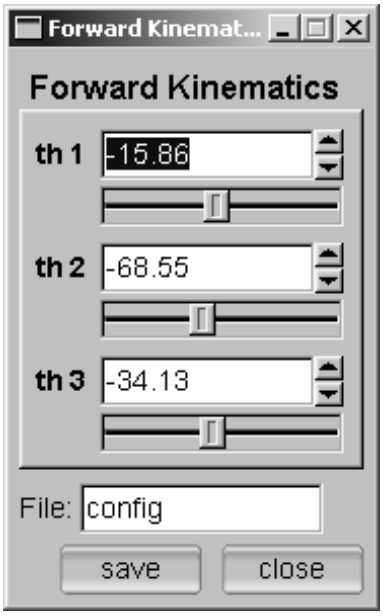

Figure 14: Dialog: Forward Kinematic

2.6 The `Trajectory` Menu

2.6.1 `Joint Trajectory`: Making the robot follow a joint trajectory

The `Joint Trajectory` command allows the robot to follow a pre-stored joint trajectory. To activate the `Joint Trajectory` command, select `Joint Trajectory` from the `Trajectory` menu (Fig. 16). The `Joint Trajectory` dialog will popup on the screen (Fig. 17). In this dialog a list box displays the stored joint trajectories for the loaded robot.

- Click on the desired trajectory to load it. The *corresponding* Cartesian trajectory is evaluated and displayed in the main RVS4W window in *blue*.
- `Points`: Click once to display the corresponding Cartesian-position points of the trajectory. Click again to switch the option off.
- `Frames`: Click once to display the Cartesian frames of the trajectory. Click again to switch the option off.
- `Cont`: To make the robot follow the trajectory continuously, check the `Cont` text box.
- `Step`: To make the robot follow the trajectory step-by-step, check the `Step` text box.

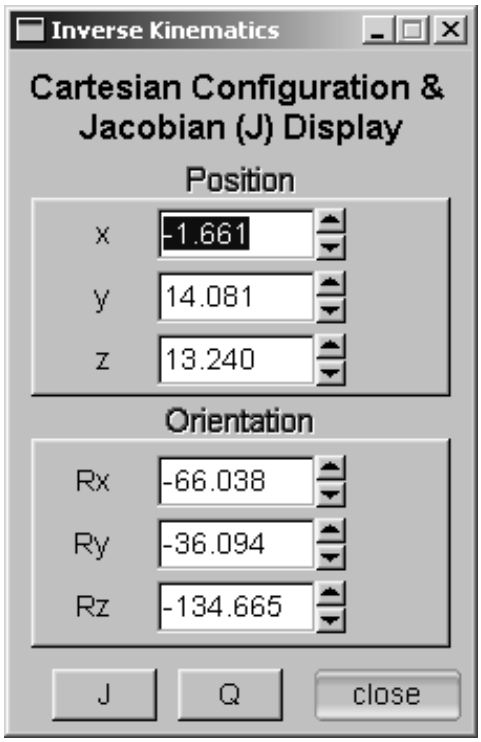

Figure 15: The `Inverse Kinematics` dialog

- `Reset`: Click to move the end-effector to the initial position.
- `Clear`: Click to unload the selected trajectory, clear the trajectory display from the workspace and reset the dialog.
- Forward: Click the forward button ▶ to start the animation in the forward direction.
  - If the `Cont` option is selected, the end-effector will move to the initial point of the trajectory and follow the trajectory to the end. Click the forward button ▶ again to stop the animation in-between.
  - If the `Step` option is selected, the end-effector will move to the next step in the trajectory.
- Backward: Click the backward button ◀ to start the animation in the backward direction.
- Use the scroll bar to drag the robot along the trajectory manually.
- Click `close` to exit the dialog.

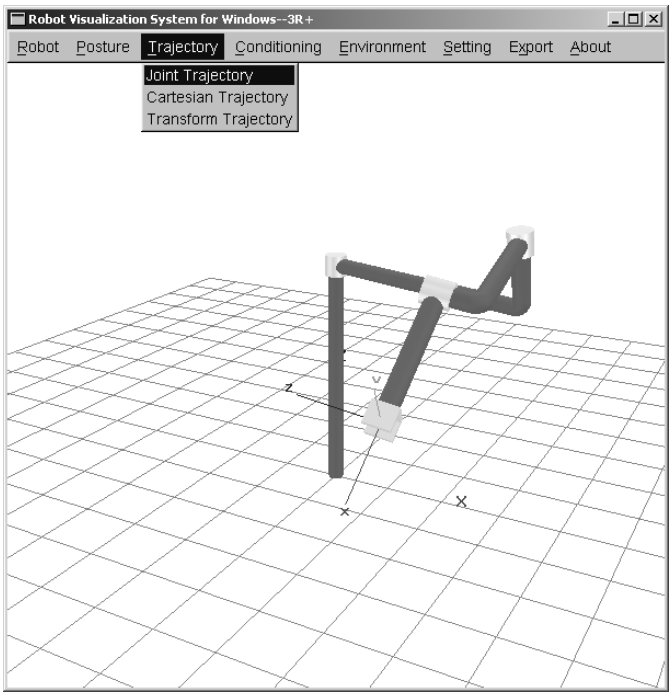

Figure 16: Menu: Trajectory

2.6.2 `Cartesian Trajectory`: Making the robot follow a Cartesian trajectory

The `Cartesian Trajectory` command allows the robot to follow a pre-stored end-effector trajectory that is described in the Cartesian coordinates. In order to activate the `Cartesian Trajectory` command, select `Trajectory > Joint Trajectory` from the menu (Fig. 16). This brings up the `Cartesian Trajectory` dialog (Fig. 18). The list box in this dialog displays the Cartesian trajectories available for the loaded robot.

- Click on the desired trajectory to load it. The Cartesian trajectory is displayed in the main RVS4W window in *pink*.
- `Points`: Click once to display the corresponding Cartesian position points of the trajectory. Click again to switch the option off.
- `Frames`: Click once to display the Cartesian frames of the trajectory. Click again to switch the option off.
- `Cont`: To make the robot follow the trajectory continuously, check the `Cont` text box.
- `Step`: To make the robot follow the trajectory step-by-step, check the `Step` text box.

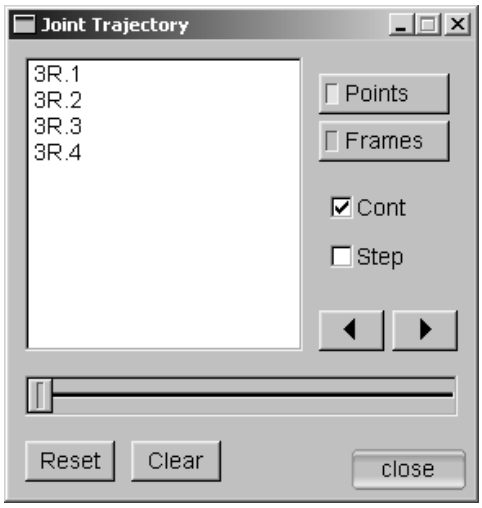

Figure 17: The `Joint Trajectory` dialog

- `Reset`: Click to move the end-effector to the initial position.
- `Clear`: Click to unload the selected trajectory, clear the trajectory display from the workspace and reset the dialog.
- Forward: Click the forward button ▶ to start the animation in the forward direction.
  - If the `Cont` option is selected, the end-effector will move to the initial point of the trajectory and follow the trajectory to the end. Click the forward button ▶ again to stop the animation in-between.
  - If the `Step` option is selected, the end-effector will move to the next step in the trajectory.
- Backward: Click the backward button ◀ to start the animation in the backward direction.
- Use the scroll bar to drag the robot along the trajectory manually.
- RVS4W can save the corresponding joint trajectory to a file. This is done by following the steps below:
  1. Load the desired Cartesian trajectory and check the `Cont` check box.
  2. Make the robot follow the complete trajectory by clicking the forward button ▶ and letting it run to the end. This step evaluates the corresponding joint angle trajectory of the Cartesian trajectory at hand.

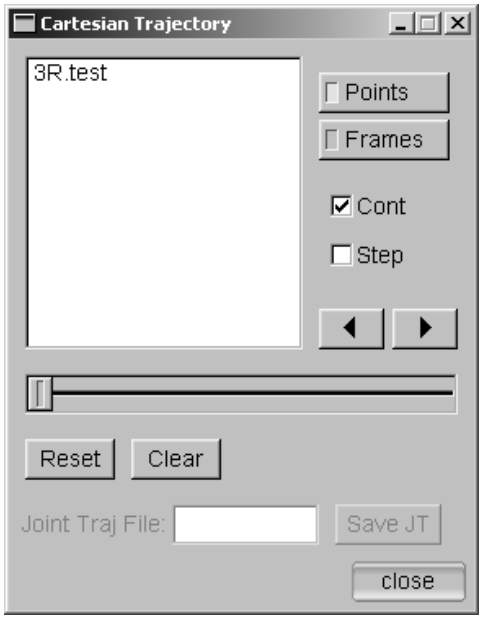

Figure 18: The `Cartesian Trajectory` dialog

3. Specify a filename for the corresponding joint trajectory in the `Joint Traj File` text box and click `Save`.

- Click `close` to exit the dialog.

2.6.3 `Transform Trajectory`: Transform a displayed trajectory

The `Transform Trajectory` command allows the user to transform a pre-stored trajectory in the workspace. Notice that the user must load a trajectory, joint or Cartesian, into the system before activating the `Transform Trajectory` command. To activate the `Transform Trajectory` command, select `Transform Trajectory` from the `Trajectory` menu (Fig. 16), the `Transform Trajectory` dialog then popping up on the screen (Fig. 19).

- `Position`: To translate the trajectory, input the x, y and z translation components in the corresponding text boxes.

- `Orientation`: To rotate the trajectory about the origin of the base frame, input the relevant `Rx`, `Ry` and `Rz` components.

- `Scale`: To scale the trajectory with respect to the origin of the base frame, input the required scaling in the `Scale` text box.

- `Traj File`: To save the trajectory, type-in the filename in the `Traj File` text box press `Save`.
- Click `close` to hide the dialog window.

2.6.4 Converting joint trajectories to Cartesian trajectories

The `Transform Trajectory` command can also be used to convert a joint trajectory to its corresponding end-effector Cartesian trajectory. This is done by following the steps below:

1. Load the desired joint trajectory (Sec. 2.6.1).
2. Activate the `Transform Trajectory` dialog.
3. Input the filename for the corresponding Cartesian trajectory in `Traj File` and click `save`.

2.7 Conditioning

For a detailed account of the theoretical issues pertaining to this subsection, the reader is referred to (Khan and Angeles, 2006). The `Conditioning` dialog provides useful information to the user, namely, the conditioning, the characteristic length and the maximum reach of the robot at hand. Select `Conditioning` from the main menu to activate the `Conditioning` command. The `Conditioning` dialog will popup on the screen (Fig. 20).

In RVS4W the robot conditioning is computed as the reciprocal of the minimum condition number κ_F of the *dimensionless* (also termed *homogeneous*) Jacobian $\mathbf{H}$, defined as

$$\kappa_F \equiv \frac{1}{m}\sqrt{||\mathbf{H}||_F||\mathbf{H}^{-1}||_F} \tag{1}$$

where $||\cdot||_F$ is the Frobenius norm of the matrix argument, m is 3 for planar and spherical robots, 6 for spatial; moreover, we assume n joints, with $n \geq m$. The factor $1/m$ in the right-hand side of eq.(1) stems from the use of the *weighted* Frobenius norm of $\mathbf{H}$, namely,

$$||\mathbf{H}||_F \equiv \sqrt{\mathrm{tr}(\mathbf{H}\mathbf{W}\mathbf{H}^T)}, \quad \mathbf{W} = \frac{1}{m}\mathbf{1} \tag{2}$$

Hence, the weighted Frobenius norm of a $m \times n$ matrix, with $n \geq m$, yields the rms value of the set of m singular values of $\mathbf{H}$ (Strang, 1986). The conditioning lies in the interval $[0, 1]$, with 1 being best and 0 worst.

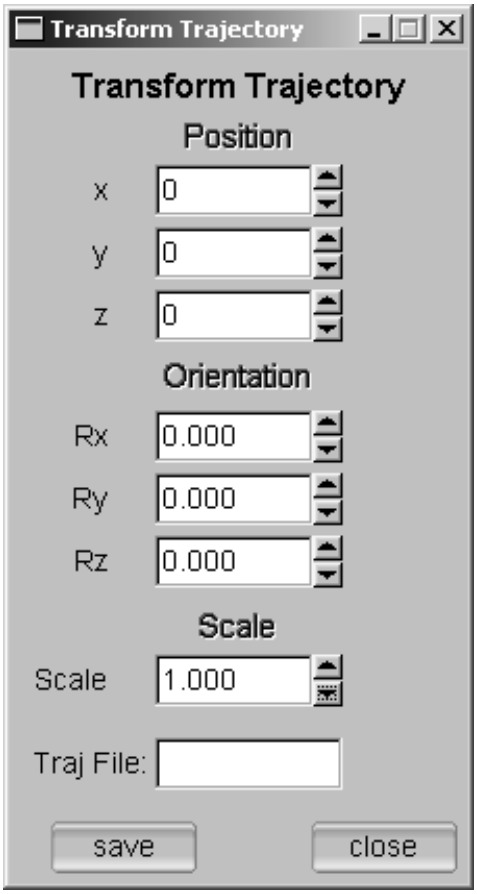

Figure 19: The `Transform Trajectory` dialog

If the robot at hand is redundant, with $n > m$ joints, then $\mathbf{H}$ is a $m \times n$ matrix, formula (1) still applying, but with $\mathbf{H}^{-1}$ replaced by the *right Moore-Penrose generalized inverse* of $\mathbf{H}$, namely $\mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}$ (Strang, 1986). It is not difficult to prove that the Frobenius norm of the foregoing generalized inverse reduces to

$$||\mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}||_F = \sqrt{\frac{1}{m}\mathrm{tr}[(\mathbf{H}\mathbf{H}^T)^{-1}]} \tag{3}$$

In order to avoid dimensional inhomogeneity[1], RVS4W evaluates the *characteristic length* of the robot. The numerical value of the characteristic length is shown in the `Characteristic length` display box in the `Conditioning` window, in usu. Further details on conditioning, characteristic length and the optimum posture can be found in (Angeles, 2007; Khan and Angeles, 2006).

The evaluation of the characteristic length, given the DH parameters of a robot,

[1]That is, a Jacobian with entries bearing disparate physical units

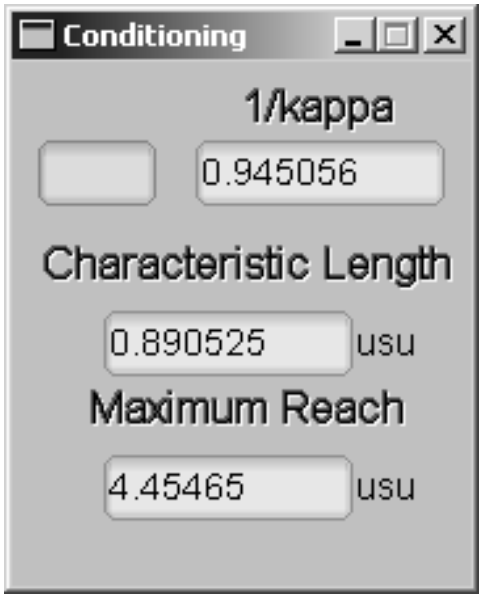

Figure 20: The `Conditioning` dialog

is known as the *forward problem* (Khan and Angeles, 2006), the inverse problem consisting in the computation of all DH parameters of an optimum robot with maximum conditioning, under user-specified constraints. RVS4W attempts to solve the above-mentioned problem using the Nelder-Mead simplex method. This is done when the robot is loaded. RVS4W displays a '+' besides the robot name upon convergence. The characteristic length is displayed in the `Characteristic length` text box. Should the Nelder-Mead method fail to converge, a '−' sign is displayed besides the filename and a value of −1 is put in the `Characteristic length` text box. For cases where evaluation of the characteristic length is either irrelevant or undefined, e.g., in the case of spherical robots, RVS4W displays 0 in the `Characteristic length` text box.

2.8 The `Environment` Menu

2.8.1 `Set Work Environment`: Creating and editing the work environment

The `Set Work Environment` command allows the user to set the work environment. Click `Environment > Set Work Environment` from the menu (Fig. 21). This brings up the `Set Work Environment` dialog (Fig. 22) to select and place objects in the work environment, follow the steps below:

1. Input the required number of objects in the `Total` text box.
2. Select the first object by choosing '1' from the `ID` list.

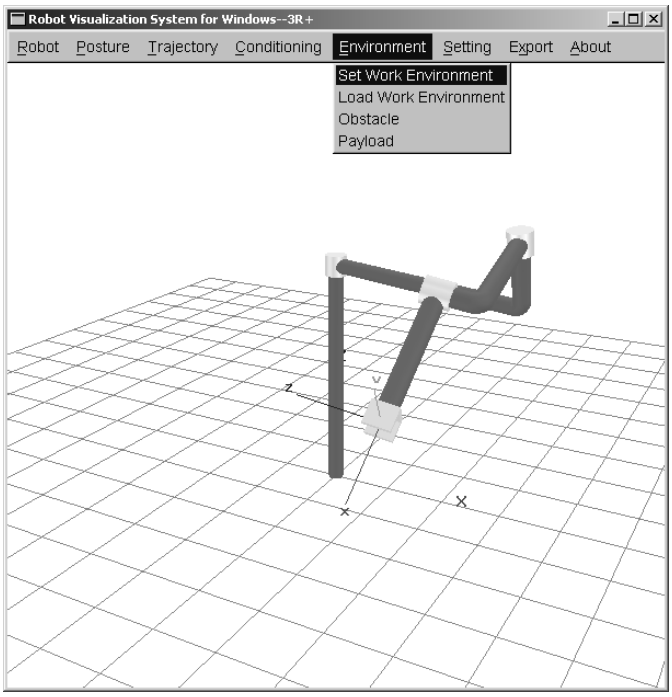

Figure 21: Menu: Environment

3. Turn on the `Visible` button.

4. Select the type of object from the `Object` drop-down list.

5. Enter the dimensions of the selected object in the `Length`, `Height`, and `Width` text boxes.

6. Enter the position of the selected object in the `Position x`, `Position y`, and `Position z` text boxes.

7. Orient the object by selecting an axis of rotation and inputting the desired angle of rotation by using the `Rotation` text box.

8. From the `Color` list, select the color of the object.

9. Select the next ID from the `ID` list and repeat steps 3 to 8.

To save the environment, input the filename in the `File` text box and click `Save`. Click `close` to exit the dialog.

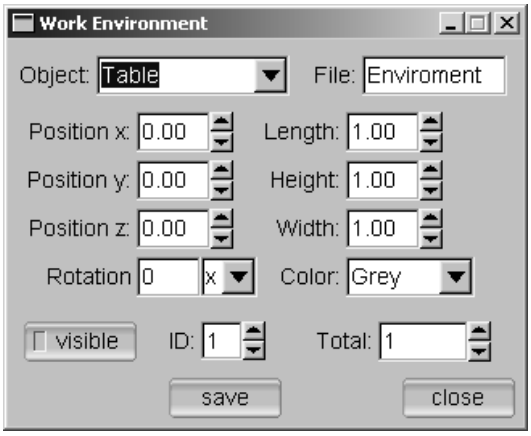

Figure 22: The `Set Work Environment` dialog

2.8.2 `Load Work Environment`: Loading a work environment

To load the work environment of a robot, select `Load Work Environment` from the `Environment` menu (Fig. 21). The `Load Work Environment` dialog will popup on the screen (Fig. 23). The types of Work environment available are displayed in a list box. Select the desired work environment to load and display it in the main window.

2.8.3 `Obstacle`: Creating and editing obstacles

RVS4W allows the user to place obstacles in the workspace to detect collisions visually. Select `Obstacle` from the `Environment` menu (Fig. 21). The `Obstacle` dialog will pop up on the screen (Fig. 24). The position and velocity of the obstacle can be specified from the `Obstacle Data` group. The position of the obstacle is defined with respect to the base frame. Currently the obstacle velocity option is not operational. The shape of the obstacle is spherical. Its radius can be modified by dragging the `Radius` scroll bar or by entering the new values, in usu, in the corresponding text fields. To display the obstacle, click the `Visible` push button. When the obstacle is visible, the user will see a grey sphere inside the workspace. Click `close` to exit the dialog.

2.8.4 `Payload`: Creating and editing payloads

RVS4W allows the user to mount a payload at the tip of the end-effector. Select `Payload` from from the `Environment` menu (Fig. 21). The `Payload` dialog will appear on the screen (Fig. 25). The user can position the payload with respect to the end-effector frame. The default position is the origin of the end-effector frame. To modify, type-in the new value in the appropriate text box. The shape of the payload is

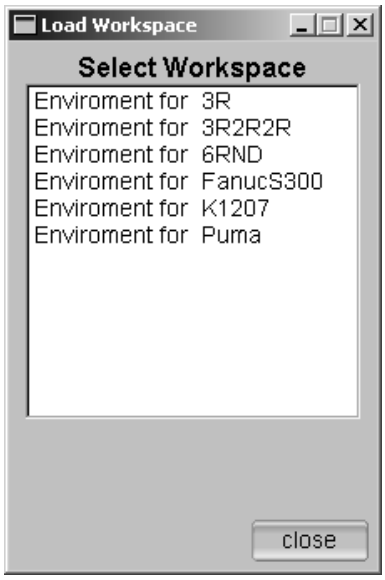

Figure 23: The `Load Work Environment` dialog

spherical by default[2]. Its radius is modified by using the scroll bar or typing in the new value in the `Radius` text field. To visualize the payload, click on the `Visible` button. When the payload is visible, the user will see a green sphere inside the workspace. Click on the `close` button to exit the dialog.

2.9 Setting

Use the `Set System` dialog to change various global settings in RVS4W. These include the background color, projection, axis-display and settings related to the inverse kinematics engine. Select `Setting > Set System` from the menu (Fig. 26). The `Set System` dialog will appear on the screen (Fig. 27). The dialog is divided into four groups.

- `Background Color`: the `R`, `G` or `B` slider to change the color of the background. A palette of three colors is also provided.
- `View`: Use the radio buttons to select the type of projection: perspective or orthogonal.
- `Axes`: There three options in this group, i.e., `Reference`, `Joint` and `End-Effector` which allow the user to display or hide the coordinate axes.

[2]The user who wants to develop a feature allowing for user-defined payload shapes is invited to request the source code by writing to the address provided in the RVS4W Web site.

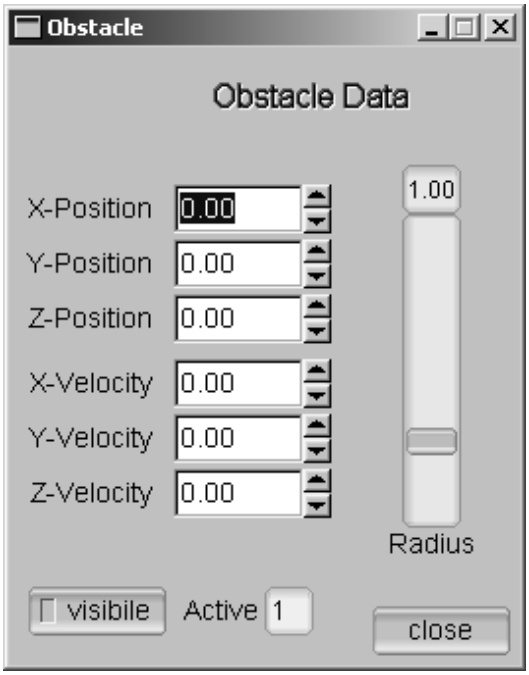

Figure 24: The `Obstacle` dialog

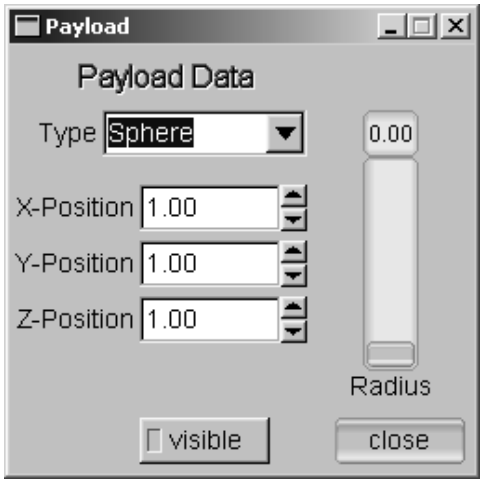

Figure 25: The `Payload` dialog

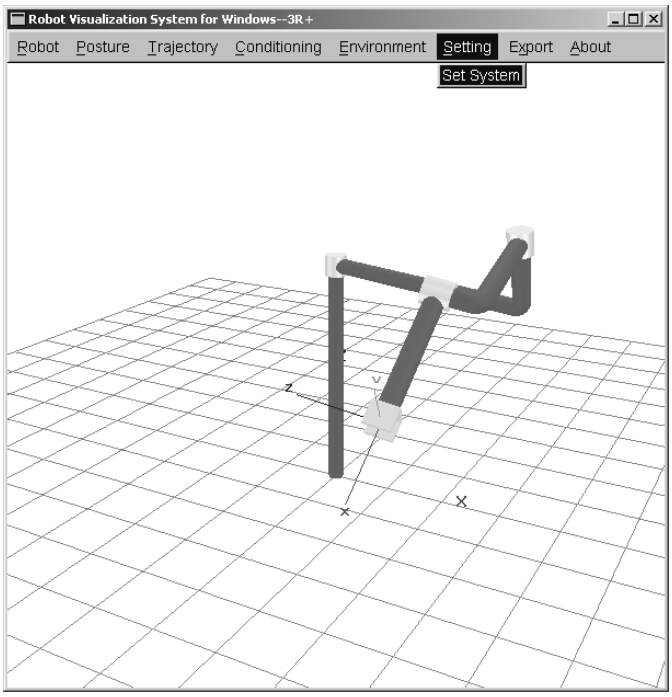

Figure 26: Menu: Setting

- `Inverse Kinematics`: This group allows the user to fine-tune the inverse kinematics engine. Please refer to the Appendix for further details.

To exit the dialog, click `close`.

2.10 `Export`: Exporting a scene

RVS4W can export the current scene to various file formats. To export a scene:

1. Select `Export > Export` (Fig. 28). The `Export` dialog will popup on the screen (Fig. 29).
2. Input the filename in the `Filename` text box.
3. Select the desired file format from the drop-down list and click `Save`

2.11 About

The `About` command provides some information about RVS4W. Select `About > About` (Fig. 30) to display the `About` dialog on the screen (Fig. 31).

Configurations

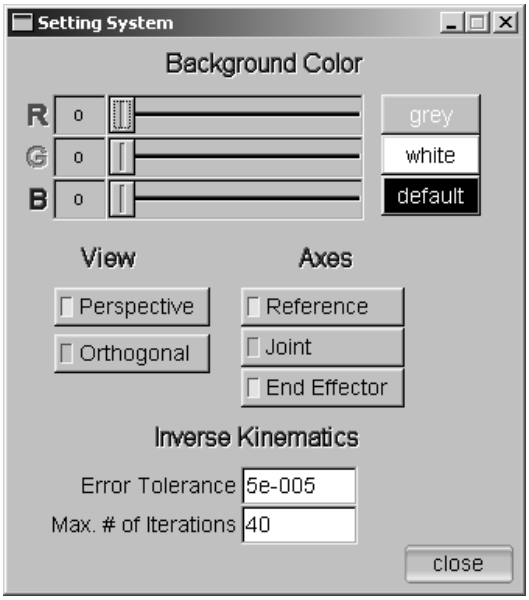

Figure 27: The `Set System` dialog

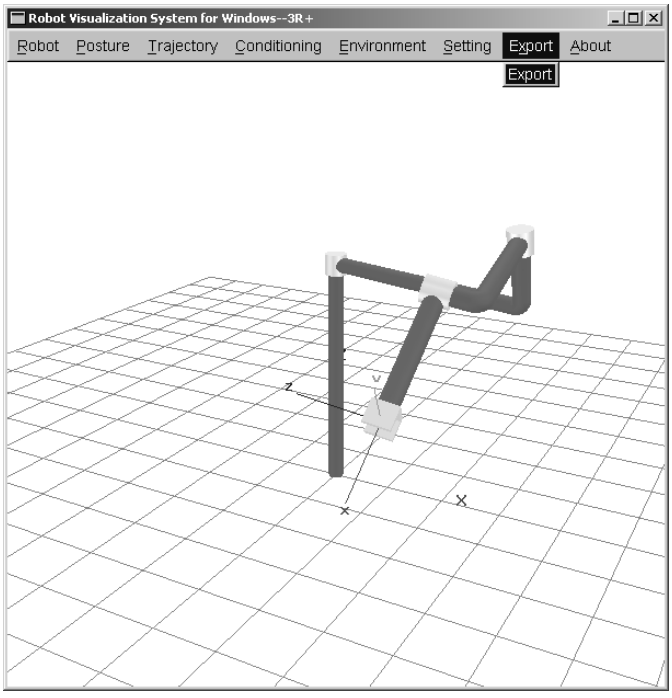

Figure 28: Menu: Export

Figure 29: The `Export` dialog

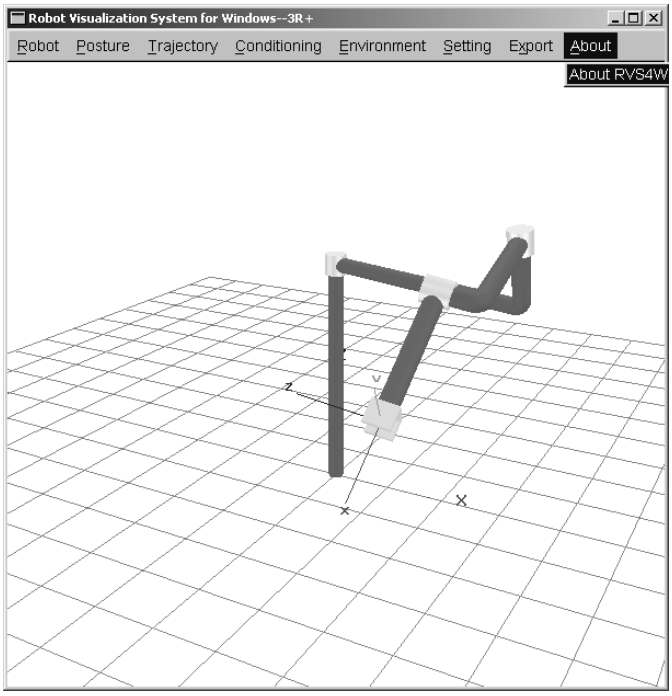

Figure 30: Menu: about

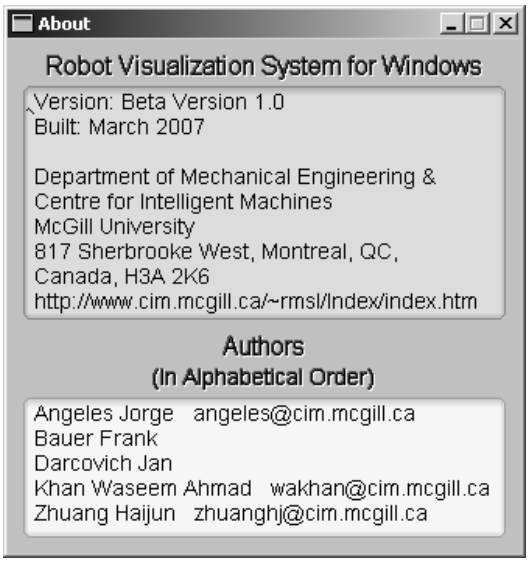

Figure 31: The `About` dialog

3 Joint Configuration

RVS allows the user to move the robot to pre-stored postures (configurations). The pre-stored configurations must be described in terms of the joint coordinates. The filename must be in the form:

`<Robot name>.Configuration name`

The joint configuration file must be provided in text format with the structure,

$$\text{th} = [\ \theta_1\ ,\ \theta_2\ ,\ \theta_3\ ,\ \ldots\ ]$$

where θ_i is the variable of the ith joint, given in degree (or in usu for prismatic joints). The file must be stored in the `RVS/JConfig/` directory.

4 Trajectories

4.1 Cartesian Trajectories

RVS4W allows the end-effector to follow pre-stored Cartesian trajectories. In this vein, the trajectory is discretized to the desired resolution, thereby obtaining a set of frames $\mathcal{F}$. Each frame $\mathcal{F}(i)$ then represents the desired end-effector pose with respect to the base frame, at sample time i.

The Cartesian trajectory file must be provided in text format with the structure

$$\begin{array}{ccc} x(1) & y(1) & z(1) \\ Q_{11}(1) & Q_{12}(1) & Q_{13}(1) \\ Q_{21}(1) & Q_{22}(1) & Q_{23}(1) \\ Q_{31}(1) & Q_{32}(1) & Q_{33}(1) \\ x(2) & y(2) & z(2) \\ Q_{11}(2) & Q_{12}(2) & Q_{13}(2) \\ Q_{21}(2) & Q_{22}(2) & Q_{23}(2) \\ Q_{31}(2) & Q_{32}(2) & Q_{33}(2) \\ \vdots & \vdots & \vdots \end{array}$$

where $x(i)$, $y(i)$ and $z(i)$ are the position coordinates of the origin of the frame $\mathcal{F}(i)$, and $Q_{mn}(i)$ are the nine entries of the 3×3 rotation matrix $\mathbf{Q}(i)$ expressed in the base frame.

For *positioning* robots, the structure of the text file must be

$$\begin{array}{ccc} x(1) & y(1) & z(1) \\ x(2) & y(2) & z(2) \\ \vdots & \vdots & \vdots \end{array}$$

For *orienting* robots, only nine entries of the orientation matrix $\mathbf{Q}$ are required, i.e.,

$$\begin{array}{ccc} Q_{11}(1) & Q_{12}(1) & Q_{13}(1) \\ Q_{21}(1) & Q_{22}(1) & Q_{23}(1) \\ Q_{31}(1) & Q_{32}(1) & Q_{33}(1) \\ Q_{11}(2) & Q_{12}(2) & Q_{13}(2) \\ Q_{21}(2) & Q_{22}(2) & Q_{23}(2) \\ Q_{31}(2) & Q_{32}(2) & Q_{33}(2) \\ \vdots & \vdots & \vdots \end{array}$$

The trajectory file must be stored in the `/RVS/CTraj` directory. The filename must be in the form

`<Robot name>.Cartesian trajectory name`

4.2 Joint Trajectory

RVS4W allows the robot to follow a pre-stored joint trajectory. The trajectory file must be stored in the `/RVS/JTraj` directory. The filename must be in the form

`<Robot name>.Joint trajectory name`

The joint trajectory file must be provided in text format with the structure

$$\begin{array}{cccc} \theta_1(1) & \theta_2(1) & \theta_3(1) & \dots \\ \theta_1(2) & \theta_2(2) & \theta_3(2) & \dots \\ \theta_1(3) & \theta_2(3) & \theta_3(3) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

where $\theta_i(j)$ is the variable of the i^{th} joint, given in degree (or usu) at sample time j. An example of the joint trajectory file is shown below, as applicable to a redundant, seven-revolute robot. Note that the text file contains correspondingly seven columns.

$$\begin{array}{ccccccc} -101.7014 & -17.6766 & 98.9590 & 88.8606 & -25.0257 & 119.4901 & -113.3444 \\ -101.6753 & -17.6985 & 98.8462 & 88.9656 & -25.2299 & 119.6729 & -113.5423 \\ -101.6438 & -17.7185 & 98.7321 & 89.0719 & -25.4328 & 119.8611 & -113.7414 \\ -101.6070 & -17.7365 & 98.6165 & 89.1794 & -25.6346 & 120.0545 & -113.9415 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}$$

References

Angeles, J., 1985, "On the numerical solution of the inverse kinematic problem," *The International Journal of Robotics Research*, Vol. 4, No. 2, pp. 21–37.

Angeles, J., 2007, *Fundamentals of Robotic Mechancal Systems: Theory, Methods, and Algorithms*, Third edition, Springer, New York.

Darcovich, J., Angeles, J., Montagnier, P. and Wu, C.-J., 1999, "RVS: A robot visualisation software package," in Batoz, J.-L., Chedmail, P., Cognet, G. and Fortin, C. (editors), *Integrated Design and Manufacturing in Mechanical Engineering*, Kluwer Academic Publisher, Dordrecht, pp. 265–272.

Strang, G., 1986, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley.

Khan, W. A. and Angeles, J., 2006, "The kinetostatic optimization of robotic manipulators: the inverse and the direct problems," *ASME Journal of Mechanical Design*, No. 128, pp. 168–178.

Appendix

Here, we provide a brief account of the numerical procedure that RVS4W uses to solve the inverse kinematics (displacement) problem. For details, the reader is referred to

Angeles (1985), as a basic reference. We have fine-tuned the numerics behind the algorithm therein.

We set the task as a nonlinear least-square problem in the six joint variables $\boldsymbol{\theta}$ as

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\boldsymbol{\phi}^T\boldsymbol{\phi} \tag{4}$$

where

$$\boldsymbol{\phi} = \begin{bmatrix} \text{vect}(\Delta\mathbf{Q}) \\ \text{tr}(\Delta\mathbf{Q}) \\ \Delta\mathbf{r} \end{bmatrix};\ \Delta\mathbf{Q} = \mathbf{Q}\mathbf{Q}_C^T - \mathbf{1};\ \Delta\mathbf{r} = \mathbf{r} - \mathbf{r}_c \tag{5}$$

Furthermore, $\mathbf{1}$ is the 3×3 identity matrix, $\mathbf{Q}$ is the rotation matrix that represents the current orientation of the end-effector with respect to that of the base-frame, $\mathbf{Q}_C$ being the target counterpart of $\mathbf{Q}$—i.e., the representation of the desired EE orientation—while $\mathbf{r}$ is the *homogenized* position vector of the end-effector. That is, $\mathbf{r}$ is a dimensionless position vector, obtained upon dividing the current position vector, in usu, by the robot characteristic length, in usu as well. Notice that the characteristic length is calculated by RVS4W upon loading a given robot. The iterative procedure involved can be either successful or unsuccessful in reaching convergence. If successful, the computed value is transferred to the inverse-kinematics routine; otherwise, this routine uses the mean value of the link lengths in lieu of the characteristic length. As well,, $\mathbf{r}_C$ is the target position vector of the operation point. Both $\mathbf{Q}$ and $\mathbf{r}$ are defined in the base frame. Finally, vect$(\cdot)$ and tr$(\cdot)$ are, correspondingly, the *vector* and the *trace* (Angeles, 2007) of the argument matrix—a 3×3 rotation matrix.

RVS4W uses the Newton-Gauss method to solve problem (4) iteratively: Given an initial guess $\boldsymbol{\theta}^0$, a sequence $\boldsymbol{\theta}^1$, $\boldsymbol{\theta}^2$, ..., $\boldsymbol{\theta}^k$, $\boldsymbol{\theta}^{k+1}$, ... is generated, which produces a monotonically decreasing sequence of values z^1, z^2, ..., z^k, z^{k+1}, ..., with z^k defined as

$$z^k = \frac{1}{2}[\boldsymbol{\phi}(\boldsymbol{\theta}^k)^T\boldsymbol{\phi}(\boldsymbol{\theta}^k)] \tag{6}$$

The sequence is generated in the form:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \Delta\boldsymbol{\theta}^k \tag{7}$$

with the correction $\Delta\boldsymbol{\theta}^k$ computed as the least-square approximation of an over-determined linear system of seven equations in six unknowns, namely,

$$\boldsymbol{\Phi}\Delta\boldsymbol{\theta}^k = -\boldsymbol{\phi}(\boldsymbol{\theta}^k) \tag{8}$$

where

$$\boldsymbol{\Phi}(\boldsymbol{\theta}^k) = \frac{\partial\boldsymbol{\phi}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} \tag{9}$$

is the 7×6 Jacobian matrix of the nonlinear system (5). Notice that problem (8) is a linear system of seven equations in six unknowns. At each iteration, the correction $\Delta\boldsymbol{\theta}^k$ is computed using *Householder reflections*.

The process terminates when the termination criterion $||\boldsymbol{\phi}(\boldsymbol{\theta}^k)|| \leq \epsilon$ is met, for a given tolerance ϵ. Notice that ϵ is *dimensionless*.

Actually, the convergence criterion for a nonlinear least-square problem is $||\Delta\boldsymbol{\theta}^k|| \leq \epsilon_\theta$, for a prescribed tolerance ϵ_θ. The criterion adopted in RVS4W is stricter, in that it imposes the vanishing of $\boldsymbol{\phi}(\boldsymbol{\theta}^k)$, which is possible in our case because the four scalar orientation equations are not independent; they obey a quadratic constraint (Angeles, 2007).

The `Set System` (Subsection 2.9) dialog provides access to the above-mentioned tolerance via the `Error Tolerance` text box. In this dialog the maximum number of iterations can also be set .

Index