RVS4W: A Visualization Tool for Robot Design

Waseem A. Khan, Haijun Zhuang and Jorge Angeles
Department of Mechanical Engineering & Centre for Intelligent Machines
McGill University
817 Sherbrooke Street West, Montreal, Quebec, Canada H3A 2K6
angeles@cim.mcgill.ca

Abstract

The Robot Visualization System for Windows (RVS4W) is a cross-platform version of its predecessor, the Robot Visualization System (RVS). Both RVS and RVS4W were developed at McGill University's Centre for Intelligent Machines. Although RVS4W is equally useful for design as well as for manufacturing purposes, it is also intended as an educational tool. It combines superb 3-D graphics with a user-friendly window-driven environment. RVS4W can provide quantitative information such as the characteristic length, the maximum reach, the optimum posture and the robot conditioning of a given robot. RVS4W is freeware, open-source and platform-independent. The package is a small-sized stand-alone application which makes it very easy to distribute and install, with no stringent hardware requirements. An elaborate User's Manual of RVS4W is also available for free downloading. The above-mentioned features make RVS4W a valuable pedagogical tool in robotics.

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 [1]. 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.

A *full rendering* of the actual robot can also be produced. Of course, this requires that a database

with the details of the robot-link geometry be supplied by the user.

Currently, RVS4W supports a few main features, namely,

- Forward and 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

2. Software Description

This section outlines the basics of RVS4W based on an implementation on Intel-based PCs. RVS4W can run on other types of platforms, the only required modifications being to recompile the whole project on the corresponding platform.

Figure 1: RVS4W main window

2.1. Starting RVS4W

When RVS4W is executed, the RVS4W main window appears as shown in Fig. 1. RVS4W commands are grouped in the main menu displayed at the top of the RVS4W window. Once a robot is loaded, the user can rotate, zoom and translate the rendering using the mouse.

2.2. Creating, Loading and Deleting a Robot

A new robot can be created by providing necessary information about the robot to RVS4W via the **Create Robot - Parameters** dialogue (Fig. 2). In this dialogue the corresponding robot dimensions using the DH convention described in [2], i.e., **a**, **b**, **al** (α) and **th** (θ) can be input. It is also possible to set individual joint limits as well as to provide a description of the robot being created.

Existing robots can be loaded or deleted using the **Load Robot** dialogue (Fig. 3). In this dialogue a list box displays the list of existing robots.

The user can edit all the relevant parameters of the robot at hand by activating the **Edit Robot** dialogue (Fig. 4). The geometry type can be selected in this dialogue. There is a separate **Link Settings** dialogue associated with each link of the robot (Fig. 5) that provides access to all relevant parameters of the selected robot link.

The user can also view the complete set of Denavit-Hartenberg (DH) parameters [2] via the **DH Parameters** dialogue (Fig. 6).

Figure 2: The Robot Parameter dialogue

2.3. Moving the robot to a saved posture

The robot can be moved to previously saved postures (Fig. 7). RVS4W provides two postures automatically:

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

Other postures can also be saved.

2.4. Forward and Inverse 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, the **Forward Kinematics** dialogue (Fig. 8) is used.

The inverse kinematics can be performed using the **Inverse Kinematics** dialogue (Fig. 9). The orientation matrix and the robot Jacobian can also be accessed via this dialogue. 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 [3], 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 θ as

$$\min_{\theta} \frac{1}{2} \phi^T \phi \quad (1)$$

Figure 3: The Load Robot dialogue

where ϕ is a 7-dimensional vector, namely,

$$\phi = \begin{bmatrix} \mathbf{m} \\ m \\ \Delta \mathbf{r} \end{bmatrix} = \begin{bmatrix} \text{vect}(\Delta \mathbf{Q}) \\ \text{tr}(\Delta \mathbf{Q}) \\ \Delta \mathbf{r} \end{bmatrix} \quad (2)$$

while $\text{vect}(\cdot)$ and $\text{tr}(\cdot)$ denote the linear invariants of the 3×3 matrix $(\cdot)$ [2]. Moreover,

$$\Delta \mathbf{Q} = \mathbf{Q} \mathbf{Q}_C^T - \mathbf{1}; \Delta \mathbf{r} = \mathbf{r} - \mathbf{r}_c \quad (3)$$

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 user-specified units (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

Figure 4: The Edit Robot dialogue

Figure 5: The Link Settings dialogue

Figure 6: The DH Parameters dialogue

Figure 7: The Select Configuration dialogue

Figure 8: Dialogue: Forward Kinematic

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.

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

$$z^k = \frac{1}{2} \boldsymbol{\phi}(\boldsymbol{\theta}^k)^T \boldsymbol{\phi}(\boldsymbol{\theta}^k) \quad (4)$$

The sequence is generated in the form:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \Delta \boldsymbol{\theta}^k \quad (5)$$

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

$$\boldsymbol{\Phi} \Delta \boldsymbol{\theta}^k = -\boldsymbol{\phi}(\boldsymbol{\theta}^k) \quad (6)$$

where

$$\boldsymbol{\Phi}(\boldsymbol{\theta}^k) = \left. \frac{\partial \boldsymbol{\phi}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^k} \quad (7)$$

is the 7×6 Jacobian matrix of the nonlinear system (2). Notice that eq. (6) represents a linear system of seven equations in six unknowns. At each iteration, the correction $\Delta \boldsymbol{\theta}^k$ is computed using *Householder reflections* [4].

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 [2].

2.5. Computation of the Jacobian Matrix

Here, we compute the Jacobian matrix $\boldsymbol{\Phi}(\boldsymbol{\theta})$ of $\boldsymbol{\phi}$ with respect to $\boldsymbol{\theta}$. We first recall some important results and theorems that we will need in the sequel.

Theorem 1 *Let $\mathbf{R}$ be a 3×3 rotation matrix and $\mathbf{B}$ be any arbitrary 3×3 matrix. Then*

$$\text{vect}(\mathbf{RBR}^T) = \mathbf{R} \text{vect}(\mathbf{B}) \quad (8)$$

Theorem 2 Let $\mathbf{S}$ be a skew-symmetric 3×3 matrix and $\mathbf{P}$ be any arbitrary 3×3 matrix. Then

$$\text{vect}(\mathbf{SP}) = \frac{1}{2}(\mathbf{1} \text{tr}(\mathbf{P}) - \mathbf{P}) \text{vect}(\mathbf{S}) \quad (9)$$

Theorem 3 Let $\mathbf{S}$ be a skew symmetric 3×3 matrix and $\mathbf{P}$ an arbitrary 3×3 matrix. Then

$$\begin{aligned} \text{tr}(\mathbf{SP}) &= 2[\text{vect}(\mathbf{S})]^T \text{vect}(\mathbf{P}^T) \\ &= -2[\text{vect}(\mathbf{S})]^T \text{vect}(\mathbf{P}) \end{aligned} \quad (10)$$

The reader is referred to [2, 3] for details on the above theorems. The Jacobian Matrix $\Phi(\theta)$ can be written in block matrix form as

$$\Phi(\theta) = \begin{bmatrix} \partial \mathbf{m} / \partial \theta_1 & \cdots & \partial \mathbf{m} / \partial \theta_6 \\ \partial m / \partial \theta_1 & \cdots & \partial m / \partial \theta_6 \\ \partial \Delta \mathbf{r} / \partial \theta_1 & \cdots & \partial \Delta \mathbf{r} / \partial \theta_6 \end{bmatrix} \quad (11)$$

The partial derivative $\partial \mathbf{m} / \partial \theta_i$ is computed first. In this vein we obtain

$$\frac{\partial \mathbf{m}}{\partial \theta_i} = \frac{\partial}{\partial \theta_i} \text{vect}(\mathbf{Q} \mathbf{Q}_C^T - \mathbf{1}) = \text{vect}\left(\frac{\partial \mathbf{Q}}{\partial \theta_i} \mathbf{Q}_C^T\right) \quad (12)$$

The partial derivative appearing in the right-hand side can be expanded as

$$\frac{\partial \mathbf{Q}}{\partial \theta_i} = \mathbf{Q}_1 \cdots \frac{\partial \mathbf{Q}_i}{\partial \theta_i} \cdots \mathbf{Q}_6 \quad (13)$$

where $\mathbf{Q}_i$ is the rotation matrix that represents the orientation of the i th joint-frame with respect to the base-frame and expressed in the base-frame. The above equation can be cast in the form

$$\frac{\partial \mathbf{Q}}{\partial \theta_i} = \mathbf{Q}_1 \cdots \mathbf{Q}_{i-1} \mathbf{E} (\mathbf{Q}_1 \cdots \mathbf{Q}_{i-1})^T \mathbf{Q} = \mathbf{S} \mathbf{Q} \quad (14)$$

where we have used the relation

$$\frac{\partial \mathbf{Q}_i}{\partial \theta_i} = \mathbf{E} \mathbf{Q}_i \quad \text{with } \mathbf{E} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Since $\mathbf{E}$ is skew-symmetric, the matrix $\mathbf{S} = \mathbf{Q}_1 \cdots \mathbf{Q}_{i-1} \mathbf{E} (\mathbf{Q}_1 \cdots \mathbf{Q}_{i-1})^T$ is skew-symmetric as well. Substituting eq. (14) in eq. (12) and using Theorem 2 we obtain the partial derivative at hand as

$$\frac{\partial \mathbf{m}}{\partial \theta_i} = \frac{1}{2}[\mathbf{1} \text{tr}(\mathbf{P}) - \mathbf{P}] \text{vect}(\mathbf{S}) \quad (15)$$

where

$$\mathbf{P} \equiv \mathbf{Q} \mathbf{Q}_C^T$$

Using the results of Theorem 1, the above equation can be written as

$$\frac{\partial \mathbf{m}}{\partial \theta_i} = \frac{1}{2}[\mathbf{1} \text{tr}(\mathbf{P}) - \mathbf{P}] (\mathbf{Q}_1 \cdots \mathbf{Q}_{i-1}) \text{vect}(\mathbf{E}) \quad (16)$$

However, $\text{vect}(\mathbf{E}) = [0 \ 0 \ 1]^T \equiv \mathbf{e}$. Hence, $(\mathbf{Q}_1 \cdots \mathbf{Q}_{i-1}) \text{vect}(\mathbf{E})$ is nothing but the direction cosines of the i th revolute axis. Hence we obtain the following compact form

$$\frac{\partial \mathbf{m}}{\partial \theta} = (\mathbf{1} \text{tr}(\mathbf{P}) - \mathbf{P}) \mathbf{A} \quad (17)$$

where $\mathbf{A}$ is defined as a 3×6 matrix, namely,

$$\mathbf{A} = [\mathbf{e}_1 \quad \cdots \quad \mathbf{e}_6] \quad (18)$$

Next, we compute $\partial m / \partial \theta_i$ of eq.(11):

$$\frac{\partial m}{\partial \theta_i} = \frac{\partial}{\partial \theta_i} \text{tr}(\mathbf{Q} \mathbf{Q}_C^T - \mathbf{1}) = \text{tr}\left(\frac{\partial \mathbf{Q}}{\partial \theta_i} \mathbf{Q}_C^T\right) \quad (19)$$

Substituting the value of $\partial \mathbf{Q} / \partial \theta_i$ from eq.(14) in the above expression and using Theorem 3, we obtain

$$\begin{aligned} \frac{\partial m}{\partial \theta_i} &= \text{tr}(\mathbf{S} \mathbf{Q} \mathbf{Q}_C^T) \\ &= -2 \text{vect}(\mathbf{S}) \text{vect}(\mathbf{P}) = -2 \mathbf{e}_i^T \text{vect}(\mathbf{P}) \\ &= -2[\text{vect}(\mathbf{P})]^T \mathbf{e}_i \end{aligned} \quad (20)$$

Hence,

$$\frac{\partial m}{\partial \theta} = -2 \text{vect}(\mathbf{P})^T \mathbf{A} \quad (21)$$

Finally, it can be shown [3] that

$$\frac{\partial \Delta \mathbf{r}}{\partial \theta_i} = \frac{\partial \mathbf{r}}{\partial \theta_i} = \mathbf{e}_i \times \mathbf{r}_i \quad (22)$$

where $\mathbf{r}_i$ is the position vector of the operation point of the end-effector with respect to the origin of the i th joint frame. Hence,

$$\frac{\partial \Delta \mathbf{r}}{\partial \theta} = [\mathbf{e}_1 \times \mathbf{r}_1 \cdots \mathbf{e}_i \times \mathbf{r}_i \cdots \mathbf{e}_6 \times \mathbf{r}_6] \equiv \mathbf{B} \quad (23)$$

Notice that the 6×6 Jacobian matrix $\mathbf{J}$ appearing in the velocity analysis of six-revolute robots is known to have the form [2]:

$$\mathbf{J} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \quad (24)$$

and hence, the Jacobian Matrix $\Phi(\theta)$ can be computed as

$$\Phi(\theta) = \begin{bmatrix} \frac{1}{2}(\mathbf{1} \text{tr}(\mathbf{P}) - \mathbf{P}) & \mathbf{O} \\ 2[\text{vect}(\mathbf{P})]^T & \mathbf{0}^T \\ \mathbf{O} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \quad (25)$$

where $\mathbf{O}$ is the 3×3 zero matrix, $\mathbf{0}$ the 3-dimensional zero vector and $\mathbf{1}$ is the 3×3 identity matrix.

Figure 9: The Inverse Kinematics dialogue

It is interesting to note that, upon convergence, $\mathbf{P} = \mathbf{1}$, which leads to

$$\frac{\partial \phi}{\partial \theta} = \begin{bmatrix} \mathbf{A} \\ \mathbf{0}^T \\ \mathbf{B} \end{bmatrix} \quad (26)$$

and hence, the rank of the Jacobian matrix Φ cannot be smaller than the rank of the robot Jacobian $\mathbf{J}$ upon convergence.

2.6. Making the Robot Follow a Pre-stored Trajectory

RVS4W is capable of following pre-stored joint as well as Cartesian trajectories. The **Joint Trajectory** and the **Cartesian Trajectory** dialogues (Figs. 10 & 11) provide access to these features. In these dialogues a list box displays the stored trajectories for the loaded robot. The user can load and make the robot follow the loaded trajectory. The **Cartesian Trajectory** also provides a means to transform a Cartesian trajectory to the joint trajectory of the robot at hand.

2.7. Conditioning

For a detailed account of the theoretical issues pertaining to this subsection, the reader is referred to [5]. The **Conditioning** dialogue (Fig. 12) provides quantitative information to the user, namely, the

Figure 10: The Joint Trajectory dialogue

Figure 11: The Cartesian Trajectory dialogue

conditioning, the characteristic length and the maximum reach of the robot at hand.

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

$$\kappa_F \equiv \frac{1}{m} \sqrt{\|\mathbf{H}\|_F \|\mathbf{H}^{-1}\|_F} \quad (27)$$

where $\|\cdot\|_F$ is the Frobenius norm [4] of the matrix argument, m is 3 for planar and spherical, 6 for spatial robots; moreover, we assume n joints, with $n \geq m$. The factor $1/m$ in the right-hand side of eq.(27) stems from the use of the *weighted* Frobenius norm of $\mathbf{H}$, namely,

$$\|\mathbf{H}\|_F \equiv \sqrt{\text{tr}(\mathbf{H}\mathbf{W}\mathbf{H}^T)}, \quad \mathbf{W} = \frac{1}{m} \mathbf{1} \quad (28)$$

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 $\mathbf{H}$ [6]. The conditioning lies in the interval $[0, 1]$, with 1 being best and 0 worst.

If the robot at hand is redundant, with $n > m$ joints, then $\mathbf{H}$ is a $m \times n$ matrix, formula (27) still applying, but with $\mathbf{H}^{-1}$ replaced by the *right Moore-Penrose generalized inverse* of $\mathbf{H}$, namely $\mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}$ [6]. It is not difficult to prove that the Frobenius norm of the foregoing generalized inverse reduces to

$$\|\mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}\|_F = \sqrt{\frac{1}{m} \text{tr}[(\mathbf{H}\mathbf{H}^T)^{-1}]} \quad (29)$$

In order to avoid dimensional inhomogeneity¹, 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 [2, 5].

The evaluation of the characteristic length, given the DH parameters of a robot, is known as the *forward problem* [5], 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 problems 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**

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

Figure 12: The **Conditioning** dialogue

Figure 13: The **Set Work Environment** dialogue

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. Work Environments, Obstacles and Payload

RVS4W supports work environments. Objects such as tables, pallets, etc., can be placed around the robot. The **Set Work Environment** dialogue (Fig. 13) is used for this purpose. This dialogue also allows the user to save the created work environments for later use. Saved work environments can be loaded using the **Load Work Environment** dialogue (Fig. 14).

RVS4W also allows the user to place obstacles in the workspace to detect collisions visually. The **Obstacle** dialogue provides access to this feature (Fig. 15). The position and velocity of the obsta-

Figure 14: The Load Work Environment dialogue

Figure 16: The Payload dialogue

Figure 15: The Obstacle dialogue

cle can be specified from the **Obstacle Data** group. The position of the obstacle is defined with respect to the base frame.

The user can also attach a payload to the end-effector of the loaded robot. This is done via the **Payload** dialogue (Fig. 16). The user can position the payload with respect to the end-effector frame. The shape of the payload is spherical by default.

3. Conclusions

RVS4W combines 3-D visualization with a user-friendly window-driven environment. RVS4W is freeware, open-source and platform-independent. The package is a small-sized stand-alone application, which makes it easy to distribute and install,

with no stringent hardware requirements. An elaborate User's Manual of RVS4W is also available for free download.

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.

References

- [1] Darcovich, J., Angeles, J., Montagnier, P. and C.-J. Wu, "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, 1999.
- [2] Angeles, J., *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*, Third edition, Springer, New York, 2007.
- [3] Angeles, J., "On the numerical solution of the inverse kinematic problem," *The International Journal of Robotics Research*, Vol. 4, No. 2, pp. 21–37, 1985.
- [4] Golub, G.H. and C.F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, 1989.
- [5] Khan, W. A. and J. Angeles, "The kinetostatic optimization of robotic manipulators: the

inverse and the direct problems,” *ASME Journal of Mechanical Design*, No. 128, pp. 168–178, 2006.

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