Appendix B

A Mathematica Package for Screw Calculus

This appendix contains a brief description of a Mathematica package, Screws.m, which facilitates the use of screws, twists, and wrenches for analyzing robot kinematics. The Screws package implements all of the functions described in Chapter 2 and, when combined with the supplementary package RobotLinks.m, allows symbolic and numerical computation of the kinematics of open-chain robot manipulators as well as many other functions. The Mathematica program itself is described in [?].

The Screws package is available via anonymous ftp from the host avalon.caltech.edu and may be used free of charge. Documentation and installation instructions are included with the source code for the package. The Screws package was written by R. Murray and S. Sur at the California Institute of Technology. All correspondence concerning the software should be sent to via e-mail to murray@avalon.caltech.edu. The authors assume no responsibility for the correctness or maintenance of the Screws package. The source code is currently available only via anonymous ftp.

The remainder of this appendix contains a brief description of the Screws package, describing the functions which are available and their syntax. Although not strictly necessary, some familiarity with Mathematica is assumed. This appendix can also be used as a guide for implementing a screw calculus package in other symbolic and numerical programming languages.

Using the Screws package

The Screws package implements screw theory in 3-dimensional Euclidean space, $\mathbb{R}^3$. It uses homogeneous coordinates to represent points, vectors,
and rigid motions, making it easy to integrate into other Mathematica packages.

The \texttt{Screws} package consists of two groups of functions. The first group operates on rotation matrices and implements all of the mathematical operations described in Section 2 of Chapter 2. The following functions are defined for computing in $SO(3)$:

- \texttt{AxisToSkew[w]} \\
  Generate a skew-symmetric matrix given a vector $w \in \mathbb{R}^3$.

- \texttt{RotationAxis[R]} \\
  Calculate the axis of rotation for a matrix $R \in SO(3)$.

- \texttt{SkewExp[S, theta]} \\
  Calculate the exponential of a skew-symmetric matrix. If \texttt{theta} is not specified, it defaults to 1. If the first argument to \texttt{SkewExp} is a vector, \texttt{SkewExp} first converts it to a skew-symmetric matrix and then takes its exponential.

- \texttt{SkewToAxis[S]} \\
  Generates a vector given a skew-symmetric matrix.

Limited error checking is used to ensure that the arguments to the functions are in the proper form.

The second group of functions implements calculations on $SE(3)$. Rigid body transformations are represented using $4 \times 4$ matrices. Functions are provided for transforming points and vectors to and from homogeneous coordinates, as well as converting a translation and rotation pair into a $4 \times 4$ matrix. The following functions are defined for use in $SE(3)$:

- \texttt{HomogeneousToTwist[xi]} \\
  Convert $xi$ from a $4 \times 4$ matrix to a 6-vector.

- \texttt{PointToHomogeneous[q]} \\
  Generate the homogeneous representation of a point $q \in \mathbb{R}^3$.

- \texttt{RigidAdjoint[g]} \\
  Generate the adjoint matrix corresponding to $g$.

- \texttt{RigidOrientation[g]} \\
  Extract the rotation matrix $R$ from a homogeneous matrix $g$.

- \texttt{RigidPosition[g]} \\
  Extract the position vector $p$ from a homogeneous matrix $g$.

- \texttt{RigidTwist[g]} \\
  Compute the twist $xi \in \mathbb{R}^6$ which generates the homogeneous matrix $g$. 

438
• **RPToHomogeneous[R,p]**
  Construct a $4 \times 4$ homogeneous matrix from a rotation matrix $R$ and a translation $p$.

• **ScrewToTwist[h, q, w]**
  Return the twist coordinates of a screw with pitch $h$ through the point $q$ and in the direction $w$. If $h = \text{Infinity}$, then a pure translational twist is generated. In this case, $q$ is ignored and $w$ gives the direction of translation.

• **TwistAxis[xi]**
  Compute the axis of the screw corresponding to a twist. The axis is represented as a pair $\{q, w\}$, where $q$ is a point on the axis and $w$ is a unit vector describing the direction of the axis. The twist $\xi$ can be specified either as a 6-vector or a $4 \times 4$ matrix.

• **TwistExp[xi, theta]**
  Compute the matrix exponential of a twist $\xi$. The default value of $\theta$ is 1. If the first argument to TwistExp is a 6-vector, it is automatically converted to a $4 \times 4$ matrix.

• **TwistPitch[xi]**
  Compute the pitch of a twist.

• **TwistMagnitude[xi]**
  Compute the magnitude of a twist.

• **TwistToHomogeneous[xi]**
  Convert $\xi$ from a 6-vector to a $4 \times 4$ matrix.

• **VectorToHomogeneous[q]**
  Generate the homogeneous representation of a vector.

Limited error checking is used to ensure that the arguments to the functions are in the proper form.

### Manipulator kinematics

The functions defined in the **Screws** package can be used to analyze the kinematics of a robot manipulator. This section describes this process and defines some new functions which streamline the analysis of manipulator kinematics. These functions are contained in the package **RobotLinks.m**, which is included with in **Screws** package distribution.

The forward kinematics for a robot manipulator can be written as a product of exponentials (of twists). The following functions are defined for creating twists specifically for robot manipulators:
- **RevoluteTwist[q, w]**
  Construct the unit twist corresponding to a revolute joint in the direction \( w \) going through the point \( q \).

- **PrismaticTwist[q, w]**
  Construct the unit twist corresponding to a prismatic joint in the direction \( w \) going through the point \( q \).

These functions use the **ScrewToTwist** function defined in **Screws.m**.

Once the twists are defined, the forward kinematic map and the manipulator Jacobian can be calculated using matrix multiplication combined with the **TwistExp** and **RigidAdjoint** functions. These computations are automated by the following functions:

- **ForwardKinematics[{xi1, th1}, {xi2, th2}, ..., gst0]**
  Compute the forward kinematics map using the product of exponentials formula. The pairs \( \{xi, th\} \) define the joint twist and joint angle (or displacement) for each joint of the manipulator.

- **SpatialJacobian[{xi1, th1}, {xi2, th2}, ..., gst0]**
  Compute the spatial manipulator Jacobian for the manipulator. The pairs \( \{xi, th\} \) are given as in the **ForwardKinematics** function.

An example of the usage of **Screws** and **RobotLinks** packages is shown below for computing the kinematics of a SCARA manipulator. The notation corresponds to the notation used to describe the SCARA manipulator in Chapter 2.

```
<<Screws.m (* screws package *)
<<RobotLinks.m (* additional functions *)

(* Twist axes for SCARA robot, starting from the base *)
xi1 = RevoluteTwist[{0,0,0}, {0,0,1}]; (* base *)
xi2 = RevoluteTwist[{0,l1,0}, {0,0,1}]; (* elbow *)
xi3 = RevoluteTwist[{0,l1+l2,0}, {0,0,1}]; (* wrist *)
xi4 = PrismaticTwist[{0,0,0}, {0,0,1}];

(* Location of the tool frame at reference configuration *)
gst0 = RPToHomogeneous[IdentityMatrix[3], {0,l1+l2,0}];

(* Forward kinematics map *)
gst = Simplify[
    ForwardKinematics[
        {xi1,th1}, {xi2,th2}, {xi3,th3}, {xi4,th4}, gst0
    ]
];
```
(* Spatial manipulator Jacobian *)
Js = Simplify[
    SpatialJacobian[{xi1, th1}, {xi2, th2}, {xi3, th3}, {xi4, th4}, gst0]
];