Modern Robotics: Mechanics, Planning, and Control Code Library

Introduction

This is the documentation for the code library accompanying *Modern Robotics: Mechanics, Planning, and Control*, by Kevin M. Lynch and Frank C. Park, Cambridge University Press, 2017. The code is written for MATLAB, Mathematica, and Python, and originates from students' solutions to programming assignments in courses using material from the book. The current version of the code is largely the work of Huan Weng, based on previous contributions from Mikhail Todes, Matthew Collins, Mojtaba Mozaffar, Chang Liu, and Wentao Chen.

The code is commented and mostly self-explanatory in conjunction with the book. An example use is provided with each function. The primary purpose of the software is to be easy to read and educational, reinforcing the concepts in the book. The code is optimized neither for efficiency nor robustness (it does not perform error-checking on its inputs). This is to keep the code as simple and unintimidating as possible. Users are encouraged to use and modify the code however they wish; the process of using and modifying the code certainly aids in understanding the concepts in the book.

This document provides an overview of the available functions using MATLAB syntax. Functions are organized according to the chapter in which they are introduced in the book. Basic functions, such as functions to calculate the magnitude of a vector, normalize a vector, test if a value is near zero, and perform matrix operations such as multiplication and inverses, are not documented here.

Notation that is used throughout this document is summarized below.

Math symbol	**Computer variable**	**Description**
R	`R`	3×3 rotation matrix in $SO(3)$.
ω	`omg`	3-vector angular velocity.
$\hat{\omega}$	`omghat`	3-vector unit rotation axis or unit angular velocity.
θ	`theta`	Angle of rotation about an axis or distance traveled along a screw axis.
$\hat{\omega}\theta$	`expc3`	3-vector of exponential coordinates for rotation.
$[\omega], [\hat{\omega}\theta]$	`so3mat`	3×3 skew-symmetric $so(3)$ representation of ω or $\hat{\omega}\theta$.
p	`p`	3-vector for a position in space.
T	`T`	4×4 transformation matrix in $SE(3)$ corresponding to (R, p).
$[\mathrm{Ad}_T]$	`AdT`	6×6 matrix adjoint representation of $T \in SE(3)$.
v	`v`	3-vector linear velocity.
$\mathcal{V}$	`V`	6-vector twist (ω, v).
$\mathcal{S}$	`S`	A normalized 6-vector screw axis (ω, v), where (a) $\|\omega\| = 1$ or (b) $\|\omega\| = 0$ and $\|v\| = 1$.
$\mathcal{S}\theta$	`expc6`	6-vector of exponential coordinates for rigid-body motion.
$[\mathcal{V}], [\mathcal{S}\theta]$	`se3mat`	4×4 $se(3)$ representation of $\mathcal{V}$ or $\mathcal{S}\theta$.

M	`M`	End-effector configuration in $SE(3)$ when manipulator is at its zero position.
$\mathcal{B}_i$	`Blist`	$\mathcal{B}_i$ is the screw axis of the ith joint expressed in the end-effector frame when the manipulator is at the zero position. `Blist` is a list of all the joint screw axes for the manipulator, $i = 1, \ldots, n$.
$\mathcal{S}_i$	`Slist`	$\mathcal{S}_i$ is the screw axis of the ith joint expressed in the space frame when the manipulator is at the zero position. `Slist` is a list of all the joint screw axes for the manipulator, $i = 1, \ldots, n$.
J_b, J_s	`Jb`, `Js`	The $6 \times n$ manipulator Jacobian for a robot with n joints, expressed in the end-effector frame (J_b) or the space frame (J_s).
ϵ_ω	`eomg`	A small positive tolerance on the end-effector orientation error when calculating numerical inverse kinematics.
ϵ_ν	`ev`	A small positive tolerance on the end-effector linear position error when calculating numerical inverse kinematics.
θ_0	`thetalist0`	A list of joint variables that serve as an initial guess for the inverse kinematics solution.
θ_i	`thetalist`	θ_i is the joint variable for joint i, and `thetalist` is $\theta = (\theta_1, \ldots, \theta_n)$.
	`thetamat`	An $N \times n$ matrix where each row represents θ one timestep after the row preceding it in the matrix.
$\dot{\theta}_i$	`dthetalist`	$\dot{\theta}_i$ is the rate of change of joint variable i, and `dthetalist` is $\dot{\theta} = (\dot{\theta}_1, \ldots, \dot{\theta}_n)$.
	`dthetamat`	An $N \times n$ matrix where each row represents $\dot{\theta}$ one timestep after the row preceding it in the matrix.
$\ddot{\theta}_i$	`ddthetalist`	$\ddot{\theta}_i$ is the acceleration of joint i, and `ddthetalist` is $\ddot{\theta} = (\ddot{\theta}_1, \ldots, \ddot{\theta}_n)$.
	`dthetamat`	An $N \times n$ matrix where each row represents $\ddot{\theta}$ one timestep after the row preceding it in the matrix.
$\mathfrak{g}$	`g`	3-vector for gravitational acceleration.
$\tilde{\mathfrak{g}}$	`gtilde`	A possibly incorrect model for $\mathfrak{g}$ used by a controller.
$M_{i-1,i}$	`Mlist`	$M_{i-1,i} \in SE(3)$ is the configuration of manipulator link i relative to link $i-1$ when the manipulator is at its zero position. The link frames are defined at the link centers of mass. `Mlist` is a list of all $M_{i-1,i}$ for $i = 1, \ldots, n+1$. The frame $\{n+1\}$ is the end-effector frame, and it is fixed relative to the frame $\{n\}$ of the last link. It simply offers the opportunity to define an end-effector frame other than at the center of mass of the last link.
	`Mtildelist`	A possibly incorrect model for `Mlist` used by a controller.
$\mathcal{F}_{\text{tip}}$	`Ftip`	6-vector wrench applied by the manipulator end-effector, expressed in the end-effector frame $\{n+1\}$.
	`Ftipmat`	An $N \times 6$ matrix where each row represents $\mathcal{F}_{\text{tip}}$ one timestep after the row preceding it in the matrix.
$\mathcal{G}_i$	`Glist`	$\mathcal{G}_i$ is the 6×6 spatial inertia matrix for link i of the manipulator, and `Glist` is a list of all $\mathcal{G}_i$ for $i = 1, \ldots, n$.
	`Gtildelist`	A possibly incorrect model for `Glist` used by a controller.
τ_i	`taulist`	τ_i is the generalized force applied at joint i, and `taulist` is the list of all joint forces/torques $\tau = (\tau_1, \ldots, \tau_n)$.
	`taumat`	An $N \times n$ matrix where each row represents τ one timesetep after the row preceding it in the matrix.

$[\text{ad}_{\mathcal{V}}]$	`adV`	6×6 matrix adjoint representation of $\mathcal{V} \in se(3)$, used to calculate the Lie bracket of two twists, $[\text{ad}_{\mathcal{V}_1}]\mathcal{V}_2$.
T_f	`Tf`	The total time of a motion in seconds from rest to rest when calculating trajectories.
Δt	`dt`	A timestep (e.g., between consecutive rows in a matrix representing a trajectory or force history).
t	`t`	The current time.
θ_{start}	`thetastart`	An n-vector of initial joint variables with which to start a trajectory.
θ_{end}	`thetaend`	An n-vector of final joint variables with which to end a trajectory.
X_{start}	`Xstart`	An initial end-effector configuration $X_{\text{start}} \in SE(3)$ with which to start a trajectory.
X_{end}	`Xend`	A final end-effector configuration $X_{\text{end}} \in SE(3)$ with which to end a trajectory.
e_{int}	`eint`	An n-vector of the time-integral of joint errors.
θ_d	`thetalistd`	An n-vector of reference joint variables θ_d.
	`thetamatd`	An $N \times n$ matrix where each row represents θ_d one timestep after the row preceding it in the matrix.
$\dot{\theta}_d$	`dthetalistd`	An n-vector of reference joint velocities $\dot{\theta}_d$.
	`dthetamatd`	An $N \times n$ matrix where each row represents $\dot{\theta}_d$ one timestep after the row preceding it in the matrix.
$\ddot{\theta}_d$	`ddthetalistd`	An n-vector of reference joint accelerations $\ddot{\theta}_d$.
	`dthetamatd`	An $N \times n$ matrix where each row represents $\ddot{\theta}_d$ one timestep after the row preceding it in the matrix.
K_p	`Kp`	A scalar feedback proportional gain.
K_i	`Ki`	A scalar feedback integral gain.
K_d	`Kd`	A scalar feedback derivative gain.
	`intRes`	The number of integration steps during each timestep Δt. The value must be a positive integer. Larger values result in slower simulations but less accumulation of integration error.

Chapter 3: Rigid-Body Motions

```
invR = RotInv(R)
```

Input:
`R`: Rotation matrix.
Output:
`invR`: The inverse of `R`.

For efficiency, the inverse is calculated as the transpose rather than a matrix inverse.

```
so3mat = VecToso3(omg)
```

Input:
`omg`: A 3-vector.
Output:

`so3mat`: The corresponding 3×3 skew-symmetric matrix in $so(3)$.

```
omg = so3ToVec(so3mat)
```

Input:
`so3mat`: A 3×3 skew-symmetric matrix (an element of $so(3)$).
Output:
`omg`: The corresponding 3-vector.

```
[omghat,theta] = AxisAng3(expc3)
```

Input:
`expc3`: A 3-vector of exponential coordinates for rotation $\hat{\omega}\theta$.
Output:
`omghat`: The corresponding unit rotation axis $\hat{\omega}$.
`theta`: The corresponding rotation angle θ.

```
R = MatrixExp3(so3mat)
```

Input:
`so3mat`: An $so(3)$ representation of exponential coordinates for rotation, $[\hat{\omega}\theta]$.
Output:
`R`: The $R' \in SO(3)$ that is achieved by rotating about $\hat{\omega}$ by θ from an initial orientation $R = I$.

```
so3mat = MatrixLog3(R)
```

Input:
`R`: Rotation matrix.
Output:
`so3mat`: The corresponding $so(3)$ representation of exponential coordinates.

```
T = RpToTrans(R,p)
```

Input:
`R`: Rotation matrix.
`p`: A position $p \in \mathbb{R}^3$.
Output:
`T`: The corresponding homogeneous transformation matrix $T \in SE(3)$.

```
[R,p] = TransToRp(T)
```

Input:
    `T`: Transformation matrix.
Output:
    `R`: The corresponding rotation matrix.
    `p`: The corresponding position.

```
invT = TransInv(T)
```

Input:
    `T`: Transformation matrix.
Output:
    `invT`: Inverse of `T`.

Uses the structure of transformation matrices to avoid taking a matrix inverse, for efficiency.

```
se3mat = VecTose3(V)
```

Input:
    `V`: A 6-vector (representing a twist, for example).
Output:
    `se3mat`: The corresponding 4×4 $se(3)$ matrix.

```
V = se3ToVec(se3mat)
```

Input:
    `se3mat`: A 4×4 $se(3)$ matrix.
Output:
    `V`: The corresponding 6-vector.

```
AdT = Adjoint(T)
```

Input:
    `T`: Transformation matrix.
Output:
    `AdT`: The corresponding 6×6 adjoint representation $[\mathrm{Ad}_T]$.

```
S = ScrewToAxis(q,s,h)
```

Input:
    `q`: A point $q \in \mathbb{R}^3$ lying on the screw axis.

`s`: A unit vector $\hat{s} \in \mathbb{R}^3$ in the direction of the screw axis.
`h`: The pitch $h \in \mathbb{R}$ (linear velocity divided by angular velocity) of the screw axis.
Output:
`S`: The corresponding normalized screw axis $\mathcal{S} = (\omega, v)$.

```
[S,theta] = AxisAng6(expc6)
```

Input:
`expc6`: A 6-vector of exponential coordinates for rigid-body motion, $\mathcal{S}\theta$.
Output:
`S`: The corresponding normalized screw axis $\mathcal{S}$.
`theta`: The distance traveled along/about $\mathcal{S}$.

```
T = MatrixExp6(se3mat)
```

Input:
`se3mat`: An $se(3)$ representation of exponential coordinates for rigid-body motion, $[\mathcal{S}\theta]$.
Output:
`T`: The $T' \in SE(3)$ that is achieved by traveling along/about the screw axis $\mathcal{S}$ a distance θ from an initial configuration $T = I$.

```
se3mat = MatrixLog6(T)
```

Input:
`T`: Transformation matrix.
Output:
`se3mat`: The corresponding $se(3)$ representation of exponential coordinates.