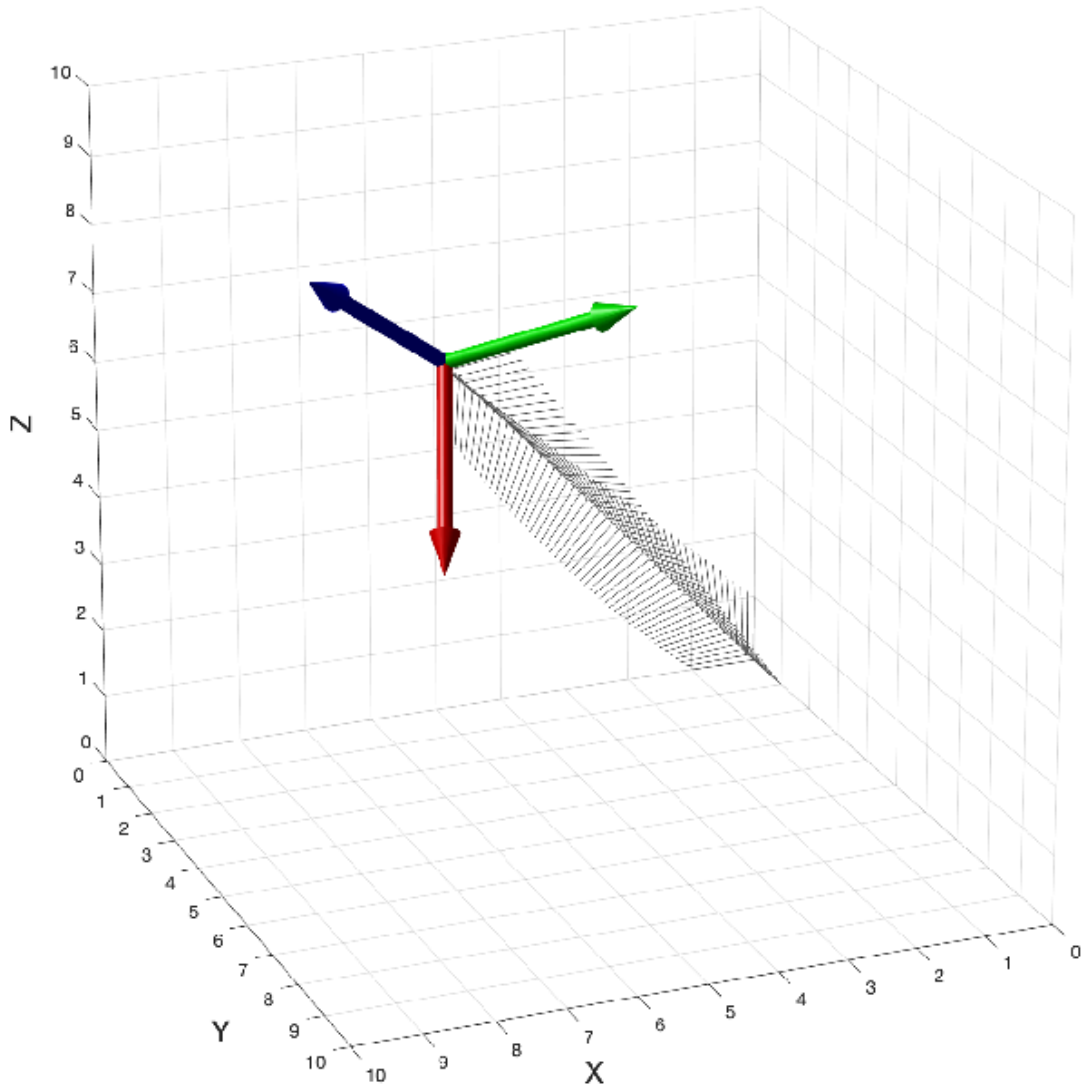
Spatial Math

Toolbox for MATLAB®

Release 1.0

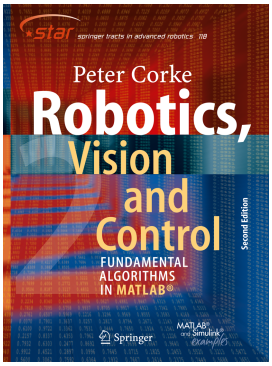


Peter Corke

Release	1.0
Release date	12 July, 2021
Licence	MIT
Toolbox home page	https://github.com/petercorke/spatial-math
Discussion group	https://tiny.cc/rvcforum

peter.i.corke@gmail.com
http://www.petercorke.com

Preface



This is the first release of the Spatial Math Toolbox which has been refactored from the Robotics Toolbox for MATLAB. The latter represents over twenty five years of continuous development and a substantial level of maturity – a significant part of that code base was concerned with representing position, orientation and pose in 2D and 3D as well as lines in 3D using Plücker coordinates.

This MATLAB® Toolbox has a rich collection of functions for manipulating and converting between datatypes such as vectors, rotation matrices, unit-quaternions, quaternions, homogeneous transformations and twists which are necessary to represent position and orientation in 2- and 3-dimensions. These are useful in the study of robotics and computer vision, but also for other fields of engineering and physics.

The Toolbox makes strong use of classes to represent many of the mathematical objects and also includes Simulink® blocks for some conversions. The code is written in a straightforward manner which allows for easy understanding, perhaps at the expense of computational efficiency. If you feel strongly about computational efficiency then you can always rewrite the function to be more efficient, compile the M-file using the MATLAB compiler, or create a MEX version.

The bulk of this manual is auto-generated from the comments in the MATLAB code itself. For elaboration on the underlying principles, extensive illustrations and worked examples please consult "*Robotics, Vision & Control*" which provides a detailed discussion (720 pages, nearly 500 figures and over 1000 code examples) of how to use the Toolbox functions to solve many types of problems in robotics. This version corresponds to the **second edition** of the book "*Robotics, Vision & Control*" published in June 2017 – aka RVC2.

Contents

Chapter 1

Introduction

As already mentioned this code has been refactored from the Robotics Toolbox for MATLAB. As that Toolbox evolved there has been increasing adoption of classes, even for objects like rotation matrices and homogeneous transformation matrices which can be represented easily using native MATLAB matrices. The motivations for this are:

1. Classes ensure type safety. For example a 3x3 matrix could be an SO(3) rotation matrix or an SE(2) homogeneous transformation, or the transpose of an SE(3) homogeneous transformation is invalid. Overloaded class operators ensure that only valid operations can be performed.

2. The classes support more descriptive constructors with names like `SO3.eul` which constructs an SO(3) object from Euler angles.

3. A sequence, or trajectory, using native matrices, has to represented by a 3-dimensional matrix, eg. $4 \times 4 \times N$. Using objects we can represent this instead using a 1-dimensional vector of objects.

In RTB10 a set of classes have been introduced to represent orientation and pose in 2D and 3D: `SO2`, `SE2`, `SO3`, `SE3`, `Twist` and `UnitQuaternion`. These classes are fairly polymorphic, that is, they share many methods and operators[1]. All have a number of static methods that serve as constructors from particular representations. A trajectory is represented by a vector of these objects which makes code easier to read and understand. Overloaded operators are used so the classes behave in a similar way to native matrices[2]. The relationship between the classical Toolbox functions and the new classes are shown in Fig **??**.

You can continue to use the classical functions. The new classes have methods with the names of classical functions to provide similar functionality. For instance

```
>> T = transl(1,2,3);  % create a 4x4 matrix
>> trprint(T)  % invoke the function trprint
>> T = SE3(1,2,3);  % create an SE3 object
>> trprint(T)  % invoke the method trprint
```

[1] For example, you could substitute objects of class `SO3` and `UnitQuaternion` with minimal code change.

[2] The capability is extended so that we can element-wise multiple two vectors of transforms, multiply one transform over a vector of transforms or a set of points.


```
>> T.T   % the equivalent 4x4 matrix
>> double(T) % the equivalent 4x4 matrix

>> T = SE3(1,2,3);  % create a pure translation SE3 object
>> T2 = T*T;  % the result is an SE3 object
>> T3 = trinterp(T, T2,, 5); % create a vector of five SE3 objects between T and T2
>> T3(1)  % the first element of the vector
>> T3*T  % each element of T3 multiplies T, giving a vector of five SE3 objects
```

Options to RTB functions can now be strings[3] or character arrays, ie. `rotx(45, 'deg')` or `rotx(45, "deg")`.

1.1 Installing the Toolbox

1.1.1 Automatically from GitHub

From MATLAB Desktop or Online use the AddOn Manager on the Home tab, and search for "spatial math" and click on the Spatial Math Toolbox. It will be installed into the folder `MATLAB/Add-Ons/Collections/Spatial Math Toolbox/petercorke-spatial-math-xxxx` in your default MATLAB documents folder[4].

This also works from MATLAB Online in which case it will be stored in `/MATLAB Add-Ons/Collections/Spatial Math Toolbox/petercorke-spatial-math-xxxx`.

The Toolbox will be automatically added to the end of your path. If you have the Phase Array Toolbox also installed then note that some of the Spatial Math functions will be shadowed. To check for this run

```
>> which rotx
```

If this indicates a path not as shown above then either:

1. use `pathtool` to move Phase Array Toolbox to the end of the path
2. remove the Phase Array Toolbox, if you don't need it, using the AddOn Manager.

1.1.2 Manually from GitHub

Clone the repository to your own computer

```
>> git clone https://github.com/petercorke/spatial-math
```

and ensure that the folder `spatial-math` is added to your MATLAB path.

1.1.3 Notes on implementation and versions

The Simulink blocks are implemented in Simulink itself with calls to MATLAB code, or as Level-1 S-functions (a proscribed coding format which MATLAB functions to interface with the Simulink simulation engine).

Simulink allows signals to have matrix values but not (yet) object values. Transformations must be represented as matrices, as per the classic functions, not classes. Very old versions of Simulink (prior to version 4) could only handle scalar signals which limited its usefulness for robotics.

1.1.4 Documentation

This document `spatialmath.pdf` is a comprehensive manual that describes all functions in the Toolbox. It is auto-generated from the comments in the MATLAB code and is fully hyperlinked: to external web sites, the table of content to functions, and the "See also" functions to each other.

[3] Introduced from MATLAB 2016b.

[4] `xxxx` is part of git's hash and represents the version number.

Orientation		Pose	
Classic	New	Classic	New
rot2	SO2	trot2	SE2
		transl2	SE2
trplot2	.plot	trplot2	.plot
rotx, roty, rotz	SO3.Rx, SO3.Ry, SO3.Rz	trotx, troty, trotz	SE3.Rx, SE3.Ry, SE3.Rz
		T = transl(v)	SE3(v)
eul2r, rpy2r	SO3.eul, SO3.rpy	eul2tr, rpy2tr	SE3.eul, SE3.rpy
angvec2r	SO3.angvec	angvec2tr	SE3.angvec
oa2r	SO3.oa	oa2tr	SE3.oa
		v = transl(T)	.t, .transl
tr2eul, tr2rpy	.toeul, .torpy	tr2eul, tr2rpy	.toeul, .torpy
tr2angvec	.toangvec	tr2angvec	.toangvec
trexp	SO3.exp	trexp	SE3.exp
trlog	.log	trlog	.log
trplot	.plot	trplot	.plot

Functions starting with dot are methods on the new objects. You can use them in functional form `toeul(R)` or in dot form `R.toeul()` or `R.toeul`. It's a personal preference. The trailing parentheses are not required if no arguments are passed, but it is a useful convention and reminder that you that you are invoking a method not reading a property. The old function `transl` appears twice since it maps a vector to a matrix as well as the inverse.

	Output type										
Input type	t	Euler	RPY	θ, v	R	T	Twist vector	Twist	Unit-Quaternion	SO3	SE3
t (3-vector)						transl		Twist('T')			SE3()
Euler (3-vector)					eul2r	eul2tr			UnitQuaternion.eul()	SO3.eul()	SE3.eul()
RPY (3-vector)					rpy2r	rpy2tr			UnitQuaternion.rpy()	SO3.rpy()	SE3.rpy()
θ, v (scalar + 3-vector)					angvec2r	angvec2tr			UnitQuaternion.angvec()	SO3.angvec()	SE3.angvec()
R (3×3 matrix)		tr2eul	tr2rpy	tr2angvec		r2t	trlog		UnitQuaternion()	SO3()	SE3()
T (4×4 matrix)	transl	tr2eul	tr2rpy	tr2angvec	t2r		trlog	Twist()	UnitQuaternion()	SO3()	SE3()
Twist vector (3- or 6-vector)					trexp	trexp		Twist()		SO3.exp()	SE3.exp()
Twist						.T	.S				.SE
Unit-Quaternion		.toeul	.torpy	.toangvec	.R	.T				.SO3	.SE3
SO3		.toeul	.torpy	.toangvec	.R	.T	.log		.UnitQuaternion		.SE3
SE3	.t	.toeul	.torpy	.toangvec	.R	.T	.log	.Twist	.UnitQuaternion	.SO3	

Dark grey boxes are not possible conversions. Light grey boxes are possible conversions but the Toolbox has no direct conversion, you need to convert via an intermediate type. Red text indicates classical Robotics Toolbox functions that work with native MATLAB® vectors and matrices. Class.type() indicates a static factory method that constructs a Class object from input of that type. Functions shown starting with a dot are a method on the class corresponding to that row.

Figure 1.1: (top) new and classic methods for representing orientation and pose, (bottom) functions and methods to convert between representations. Reproduced from "*Robotics, Vision & Control, second edition, 2017*"

1.2 Compatible MATLAB versions

The Toolbox has been tested under R2018b and R2019aPRE. Compatibility problems are increasingly likely the older your version of MATLAB is.

1.3 Use in research

If the Toolbox helps you in your endeavours then I'd appreciate you citing the Toolbox when you publish. The details are:

```
@book{Corke17a,
    Author = {Peter I. Corke},
    Note = {ISBN 978-3-319-54413-7},
    Edition = {Second},
    Publisher = {Springer},
    Title = {Robotics, Vision \& Control: Fundamental Algorithms in {MATLAB}},
    Year = {2017}}
```

or

> P.I. Corke, Robotics, Vision & Control: Fundamental Algorithms in MATLAB. Second edition. Springer, 2017. ISBN 978-3-319-54413-7.

which is also given in electronic form in the CITATION file.

1.3.1 Octave

GNU Octave (www.octave.org) is an impressive piece of free software that implements a language that is close to, but not the same as, MATLAB. The Toolboxes currently do not work well with Octave, though as time goes by compatibility improves. Many Toolbox functions work just fine under Octave, but most classes do not.

For uptodate information about running the Toolbox with Octave check out the page http://petercorke.com/wordpress/toolboxes/other-languages.

1.4 Support

There is no support! This software is made freely available in the hope that you find it useful in solving whatever problems you have to hand. I am happy to correspond with people who have found genuine bugs or deficiencies but my response time can be long and I can't guarantee that I respond to your email.

I can guarantee that I will not respond to any requests for help with assignments or homework, no matter how urgent or important they might be to you. That's what your teachers, tutors, lecturers and professors are paid to do.

You might instead like to communicate with other users via the Google Group called "Robotics and Machine Vision Toolbox"

> http://tiny.cc/rvcforum

which is a forum for discussion. You need to signup in order to post, and the signup process is moderated by me so allow a few days for this to happen. I need you to write a few words about why you want to join the list so I can distinguish you from a spammer or a web-bot.

1.5 Contributing to the Toolboxes

I am very happy to accept contributions for inclusion in future versions of the toolbox. You will, of course, be suitably acknowledged.

Chapter 2

Functions and classes

