rkd Documentation

Release 0.1.0

IRO

Aug 14, 2019

CONTENTS

rkd is a Python library for kinematic analysis of robots

Source code: https://github.com/iro-upgto/rkd

CHAPTER

ONE

CONTENTS

1.1 Didactic

Universidad Politécnica de Guanajuato Departamento de Ingeniería Robótica, (c) 2019

This module has been designed for academic purposes, using SymPy as base library. It's easy to check that SymPy is slower than NumPy specially in matrix algebra, however SymPy is more convenient to use as didactic tool due to the given facilities as the symbolic manipulation, calculation of partial and ordinary derivatives, matricial multiplication using asterisk symbol, "init_printing" function and so on.

1.1.1 core

class `rkd.didactic.core.`**`Robot`**(**args*)

Bases: `object`

Define a robot-serial-arm given the Denavit-Hartenberg parameters and joint type, as tuples:

`J`

Geometric Jacobian matrix

`J_i`(*i*)

Geometric Jacobian matrix

`T`

T_n^0 Homogeneous transformation matrix of N-Frame respect to Base-Frame

`p`(*i*)

Position for every i-Frame wrt 0-Frame

`plot_workspace`()

TODO

`z`(*i*)

z-dir of every i-Frame wrt 0-Frame

class `rkd.didactic.core.`**`RigidBody2D`**(*points*)

Bases: `object`

Defines a rigid body through a series of points that make it up.

`draw`(*color='r'*, *kaxis=None*)

Dibuja el cuerpo rígido en sus estatus actual

`move`(*q*)

Traslada el cuerpo rígido un vector q

rotate (*angle*)
Rota el cuerpo rígido un ángulo determinado alrededor del eje coordenado z.

scale (*sf*)
Escala el cuerpo rígido

1.1.2 transformations

rkd.didactic.transformations.**axa2rot** (*k*, *theta*)
Given a R^3 vector (k) and an angle (theta), return the SO(3) matrix associated.

rkd.didactic.transformations.**compose_rotations** (**rotations*)
Composes rotation matrices w.r.t. fixed or movable frames

rotations [tuple] A tuple that contains (angle, axis, frame, deg)

R [sympy.matrices.dense.MutableDenseMatrix] Rotation matrix

```
>>> compose_rotations((45, "z", "fixed", True), (30, "x", "local", True))
0.707106781186548  -0.612372435695794  0.353553390593274

0.707106781186547  0.612372435695795   -0.353553390593274

        0                 0.5          0.866025403784439
```

rkd.didactic.transformations.**dh** (*a*, *alpha*, *d*, *theta*)
Calculates Denavit-Hartenberg matrix given the four parameters.

a [int, float or symbol] DH parameter

alpha [int, float or symbol] DH parrameter

d [int, float or symbol] DH parameter

theta [int, float or symbol] DH parameter

H [sympy.matrices.dense.MutableDenseMatrix] Denavit-Hartenberg matrix (4x4)

With numerical values:

```
>>> dh(100,pi/2,50,pi/2)
0  0  1   0

1  0  0  100

0  1  0  50

0  0  0   1
```

Using symbolic values:

```
>>> a = symbols("a")
>>> t = symbols("t")
>>> dh(a,0,0,t)
cos(t)  -sin(t)   0  acos(t)

sin(t)  cos(t)    0  asin(t)
```

(continues on next page)

(continued from previous page)

```
 0          0      1      0

 0          0      0      1
```

`rkd.didactic.transformations.`**`eul2htm`**(*phi*, *theta*, *psi*, *seq='zxz'*, *deg=False*)

Given a set of Euler Angles (phi,theta,psi) for specific sequence this function returns the homogeneous transformation matrix associated. Default sequence is ZXZ.

phi [int,float,symbol] phi angle

theta [int,float,symbol] theta angle

psi [int,float,symbol] psi angle

seq [str] Rotation sequence

deg [bool] True if (phi,theta,psi) are given in degrees

H [`sympy.matrices.dense.MutableDenseMatrix`] Homogeneous transformation matrix

```
>>> eul2htm(90,90,90,"zxz",True)
0  0   1  0

0  -1  0  0

1  0   0  0

0  0   0  1
```

```
>>> eul2htm(pi/2,pi/2,pi/2)
0  0   1  0

0  -1  0  0

1  0   0  0

0  0   0  1
```

```
>>> eul2htm(0,pi/2,0,"zyz")
0   0  1  0

0   1  0  0

-1  0  0  0

0   0  0  1
```

`rkd.didactic.transformations.`**`htm2eul`**(*H*, *seq='zxz'*, *deg=False*)

Given a homogeneous transformation matrix this function return the equivalent set of Euler Angles.

If "deg" is True then Euler Angles are converted to degrees.

```
>>> H = htmrot(pi/3,"y")*htmrot(pi/4,"x")
>>> H
     6    6
1/2  ──   ──    0
     4    4
```

(continues on next page)

(continued from previous page)

```
      2   -2
 0    ──  ───   0
      2    2

-3    2    2
───   ──   ──   0
 2    4    4

 0    0    0    1
>>> htm2eul(H)
    3
atan──, atan(7), -atan(6)
    2
>>> htm2eul(H, deg=True)
(40.8933946491309, 69.2951889453646, -67.7923457014035)
```

rkd.didactic.transformations.**htmrot**(*theta*, *axis='z'*, *deg=False*)
Return a homogeneous transformation matrix that represents a rotation "theta" about "axis".

theta [float, int or *symbolic*] Rotation angle (given in radians by default)

axis [str] Rotation axis

deg [bool] ¿Is theta given in degrees?

H [`sympy.matrices.dense.MutableDenseMatrix`] Homogeneous transformation matrix

```
>>> htmrot(pi/2)
0  -1   0   0

1  0    0   0

0  0    1   0

0  0    0   1
>>> htmrot(pi/2, "x")
1  0  0    0

0  0  -1   0

0  1  0    0

0  0  0    1
>>> htmrot(30, "y", True)
0.866025403784439  0         0.5          0

        0          1          0           0

      -0.5         0  0.866025403784439   0

        0          0          0           1
>>> t = symbols("t")
>>> htmrot(t, "x")
1    0        0      0

0  cos(t)  -sin(t)   0
```

(continues on next page)

(continued from previous page)

```
0  sin(t)  cos(t)   0

0    0        0     1
```

`rkd.didactic.transformations.`**`htmtra`**`(`*d*`)`

Calculate the homogeneous transformation matrix of a translation

d [list, tuple] Translation vector

H [`sympy.matrices.dense.MutableDenseMatrix`] Homogeneous transformation matrix

```
>>> htmtra([50,-100,30])
1  0  0   50

0  1  0  -100

0  0  1   30

0  0  0   1
```

```
>>> a,b,c = symbols("a,b,c")
>>> htmtra([a,b,c])
1  0  0  a

0  1  0  b

0  0  1  c

0  0  0  1
```

`rkd.didactic.transformations.`**`rot2axa`**`(`*R*, *deg=False*`)`

Given a SO(3) matrix return the axis-angle representation

`rkd.didactic.transformations.`**`rotx`**`(`*theta*, *deg=False*`)`

Calculates the rotation matrix about the x-axis

theta [float, int or *symbolic*] Rotation angle (given in radians by default)

deg [bool] ¿Is theta given in degrees?

R [*sympy.matrices.dense.MutableDenseMatrix*] Rotation matrix (SO3)

```
>>> rotx(pi)
1  0   0

0  -1  0

0  0   -1
>>> rotx(60, deg=True)
1          0                  0

0         0.5         -0.866025403784439

0  0.866025403784439         0.5
```

`rkd.didactic.transformations.`**`roty`**`(`*theta*, *deg=False*`)`

Calculates the rotation matrix about the y-axis

theta [float, int or *symbolic*] Rotation angle (given in radians by default)

deg [bool] ¿Is theta given in degrees?

R [*sympy.matrices.dense.MutableDenseMatrix*] Rotation matrix (SO3)

```
>>> roty(pi/3)
          3
1/2    0  ──
          2

 0     1   0

-3
────   0  1/2
 2
```

```
>>> roty(30, deg=True)
0.866025403784439  0         0.5

        0          1           0

      -0.5         0  0.866025403784439
```

`rkd.didactic.transformations.`**`rotz`**`(`*theta*, *deg=False*`)`

Calculates the rotation matrix about the z-axis

theta [float, int or *symbolic*] Rotation angle (given in radians by default)

deg [bool] ¿Is theta given in degrees?, False is default value

R [*sympy.matrices.dense.MutableDenseMatrix*] Rotation matrix (SO3)

Examples

Using angle in radians,

```
>>> rotz(pi/2)
0  -1   0

1   0   0

0   0   1
```

Or symbolic variables,

```
>>> x = symbols("x")
>>> rotz(x)
cos(x)  -sin(x)  0

sin(x)  cos(x)   0

  0        0     1
```

Using angles in degrees,

```
>>> rotz(45, deg=True)
0.707106781186548  -0.707106781186547  0

0.707106781186547  0.707106781186548   0

        0                  0           1
```

`rkd.didactic.transformations.skew(u)`
Return skew-symmetric matrix associated to u vector

1.1.3 plotting

1.1.4 util

`rkd.didactic.util.deg2rad(theta, evalf=True)`
Convert degrees to radians

theta : float, int, symbolic

theta_rad : symbolic

`rkd.didactic.util.ishtm(H)`
Is H a homogeneous transformation matrix ?

`rkd.didactic.util.isorthonormal(R)`
Check if R is orthonormal

R : *sympy.matrices.dense.MutableDenseMatrix*

False or True

`rkd.didactic.util.isrot(R)`
Is R a rotation matrix ?

R : *sympy.matrices.dense.MutableDenseMatrix*

False or True

`rkd.didactic.util.rad2deg(theta, evalf=True)`
Convert radians to degrees

theta : float, int, symbolic

theta_deg : symbolic

`rkd.didactic.util.sympy2float(sympy_object)`
Convert a SymPy object to float object

`rkd.didactic.util.sympy_matrix_to_numpy_float(H)`
Convert SymPy Matrix (numerical) to NumPy array

H : *sympy.matrices.dense.MutableDenseMatrix*

Hf : array

`rkd.didactic.util.issympyobject(obj)`
Determine if input (obj) is a sympy object.

```
>>> from sympy import symbols
>>> x = symbols("x")
>>> issympyobject(x)
True
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r