rkd Documentation

Release 0.1.0

IRO

Aug 12, 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 Transformations

`rkd.transformations.`**`htmDH`**(*a*, *al*, *d*, *t*, *deg=False*)

Calculates the homogeneous matrix with the Denavir - Hartenberg (DH) parameters

** The angles must be given in radians by default **

`rkd.transformations.`**`rot2RPY`**(*R*, *deg=False*, *sol=False*)

Calculates the Roll, Pitch, Yaw angles from a rotation matrix on the XYZ axis

** The angles must be given in radians by default **

Important: The rotation matrix must be 3x3

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

Calculates the axis / angle ratio from a rotation matrix

** The angles must be given in radians by default **

Important: The rotation matrix must be 3x3

`rkd.transformations.`**`rot2eul`**(*R*, *axis*, *deg=False*, *sol=False*)

Calculates the Euler angles from a rotation matrix with differents combinations of axis ** The angles must be given in radians by default **

Important: The rotation matrix must be 3x3

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

Calculates the rotation matrix about the x-axis

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

deg [bool] ¿Is theta given in degrees?

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

Calculates the rotation matrix about the y-axis

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

deg [bool] ¿Is theta given in degrees?

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

Calculates the rotation matrix about the z-axis

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

deg [bool] ¿Is theta given in degrees?

1.2 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.

1.2.1 core

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

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

1.2.2 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)

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

1  0  0  100

0  1  0  50

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

`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.

`rkd.didactic.transformations.`**`htmrot`**`(`*theta*, *axis='z'*, *deg=False*`)`

Return a homogeneous transformation matrix that represents a rotation "theta" about "axis".

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

Calculate the homogeneous transformation matrix of a translation

`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)

`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)

`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?

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

`rkd.didactic.transformations.`**`skew`**`(`*u*`)`

Return skew-symmetric matrix associated to u vector

1.2.3 plotting

1.2.4 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

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX