Matrices Manipulation Documentation

Release 1.0

Thiago Souto

Apr 27, 2020

CONTENTS:

1 MatrixManipulation module 1

2 MatrixManipulationSymbolic module 3

3 Example7 module 5

4 Indices and tables 7

4.1 Running the documentation with Sphinx 7

4.2 GitHub Repository 7

Python Module Index 8

CHAPTER

ONE

MATRIXMANIPULATION MODULE

class MatrixManipulation.**Matrix**(***kwargs*)

Bases: `object`

Definition: This class generates Homogeneous transform matrices, that can be used to multiply any matrix and obtain the translation or rotation.

It uses *numpy* to generate the matrices:

np.float32: creates the array with 16 float32 elements

np.reshape: np.reshape rearrange the array into a 4X4 matrix

Returns: It returns Rotation and translation matrices.

Obs: **kwargs (keyword arguments) are used to facilitate the identification of the parameters, so initiate the object like: Matrix(x_angle='45', x_dist='100', z_angle='60', z_dist='100'), if an argument is not provided, the default 0 will be put to the argument.

rot_x(*gamma=0*, *degrees=True*)

Definition: Receives an alpha angle and returns the rotation matrix for the given angle at the *X* axis. If the angle is given in radian degrees should be False.

Parameters

- **gamma** (`float`) – Rotation Angle around the X axis
- **degrees** (`bool`) – Indicates if the provided angle is in degrees, if yes It will be converted to radians

Returns: The Rotational Matrix at the X axis by an *gamma* angle

rot_y(*beta=0*, *degrees=True*)

Definition: Receives an theta angle and returns the rotation matrix for the given angle at the *Z* axis. If the angle is given in radian degrees should be False.

Parameters

- **beta** (`float`) – Rotation Angle around the Z axis
- **degrees** (`bool`) – Indicates if the provided angle is in degrees, if yes It will be converted to radians

Returns: The Rotational Matrix at the Z axis by an *beta* angle

rot_z(*alpha=0*, *degrees=True*)

Definition: Receives an theta angle and returns the rotation matrix for the given angle at the *Z* axis. If the angle is given in radian degrees should be False.

Parameters

- **alpha** (*float*) – Rotation Angle around the Z axis
- **degrees** (*bool*) – Indicates if the provided angle is in degrees, if yes It will be converted to radians

Returns: The Rotational Matrix at the Z axis by an *alpha* angle

trans_x(*a=0*)

Definition: Translates the matrix a given amount *a* on the *X* axis by Defining a 4x4 identity matrix with *a* as the (1,4) element.

Parameters **a** (*float*) – Distance translated on the X-axis

Returns: The Translation Matrix on the *X* axis by a distance *a*

trans_y(*b=0*)

Definition: Translate the matrix a given amount *d* on the *Z* axis. by Defining a matrix T 4x4 identity matrix with *b* (3,4) element position.

Parameters **b** (*float*) – Distance translated on the Z-axis

Returns: The Translation Matrix on the *Z* axis by a distance *b*

trans_z(*d=0*)

Definition: Translate the matrix a given amount *d* on the *Z* axis. by Defining a matrix T 4x4 identity matrix with *c* (3,4) element position.

Parameters **d** (*float*) – Distance translated on the Z-axis

Returns: The Translation Matrix on the *Z* axis by a distance *c*

MatrixManipulation.**main**()

Example 3

CHAPTER

TWO

MATRIXMANIPULATIONSYMBOLIC MODULE

class MatrixManipulationSymbolic.**MatrixSymbolic**(***kwargs*)

Bases: object

Definition: This class generates Homogeneous transform matrices, although it uses a symbolic approach that can be used to multiply any matrix and obtain the translation or rotation.

It uses *sympy* to generate the matrices:

sympy.Matrix: creates a sympy matrix object.

sympy.Symbol: creates a symbol, Symbols are identified by name and assumptions. First, you need to create symbols using Symbol("x") We are assuming here that the symbols are "Real" number. All newly created symbols have assumptions set according to *args*, for example:

```
>>> a = symbols('a', integer=True)
>>> a.is_integer
True
>>> x, y, z = symbols('x,y,z', real=True)
>>> x.is_real and y.is_real and z.is_real
True
```

sympy.cos and sympy.sin: cos and sin for sympy

sympy.simplify: SymPy has dozens of functions to perform various kinds of simplification. simplify() attempts to apply all of these functions in an intelligent way to arrive at the simplest form of an expression.

Returns: It returns Rotation and translation matrices.

Obs: **kwargs (keyword arguments) are used to facilitate the identification of the parameters, so initiate the object

rot_x(*gamma='gamma_i-1'*)

Definition: Receives an alpha angle and returns the rotation matrix for the given angle at the *X* axis. If the angle is given in radian degrees should be False.

Parameters **gamma** (*string*) – Rotation Angle around the X axis

Returns: The Rotational Matrix at the X axis by an *given* angle

rot_y(*beta='beta_i-1'*)

Definition: Receives an theta angle and returns the rotation matrix for the given angle at the *Z* axis. If the angle is given in radian degrees should be False.

Parameters **beta** (*string*) – Rotation Angle around the Y axis

Returns: The Rotational Matrix at the Y axis by an *given* angle

rot_z (*alpha='alpha_i-1'*)

Definition: Receives an theta angle and returns the rotation matrix for the given angle at the *Z* axis. If the angle is given in radian degrees should be False.

Parameters **alpha** (*string*) – Rotation Angle around the Z axis

Returns: The Rotational Matrix at the Z axis by an *given* angle

trans_x (*a='a_i-1'*)

Definition: Translates the matrix a given amount *a* on the *X* axis by Defining a 4x4 identity matrix with *a* as the (1,4) element.

Parameters **a** (*string*) – Distance translated on the X-axis

Returns: The Translation Matrix on the *X* axis by a given distance

trans_y (*b='b_i-1'*)

Definition: Translate the matrix a given amount *d* on the *Z* axis. by Defining a matrix T 4x4 identity matrix with *b* (3,4) element position.

Parameters **b** (*string*) – Distance translated on the Z-axis

Returns: The Translation Matrix on the *Z* axis by a given distance

trans_z (*d='d_i-1'*)

Definition: Translate the matrix a given amount *d* on the *Z* axis. by Defining a matrix T 4x4 identity matrix with *c* (3,4) element position.

Parameters **d** (*string*) – Distance translated on the Z-axis

Returns: The Translation Matrix on the *Z* axis by a given distance

MatrixManipulationSymbolic.**main**()

Example 6:

Calculates the Three-link manipulator kinematics. At the end we can express a Transform from link 0 to link 3.

CHAPTER

THREE

EXAMPLE7 MODULE

```
import sympy as sympy
from src.MatrixManipulationSymbolic import MatrixSymbolic


def main():
    """
    Example 7, First part homogeneous transform:

    Calculates the Three-link manipulator kinematics.
    At the end we can express a Transform from link 0 to link 4.
    """
    print('Example 7:')

    a1 = MatrixSymbolic()        # Rx(a_i-1)
    a2 = MatrixSymbolic()        # Dx(a_i-1)
    a3 = MatrixSymbolic()        # Dz(d_i)
    a4 = MatrixSymbolic()        # Rz(theta_i)

    print()
    print('t_0_1:')
    t_0_1 = (a1.rot_x('0')) * (a2.trans_x('0')) * (a3.trans_z('0')) * (a4.rot_z('theta_1'))
    print(sympy.pretty(t_0_1))

    print('t_1_2:')
    t_1_2 = (a1.rot_x('90.0')) * (a2.trans_x('0')) * (a3.trans_z('0')) * (a4.rot_z('theta_2'))
    print(sympy.pretty(t_1_2))

    print()
    print('t_2_3:')
    t_2_3 = (a1.rot_x('0')) * (a2.trans_x('l2')) * (a3.trans_z('0')) * (a4.rot_z('theta_3'))
    print(sympy.pretty(t_2_3))

    print()
    print('t_3_4:')
    t_3_4 = (a1.rot_x('0')) * (a2.trans_x('l3')) * (a3.trans_z('0')) * (a4.rot_z('0'))
    print(sympy.pretty(t_3_4))

    t_0_4 = t_0_1 * t_1_2 * t_2_3 * t_3_4

    print()
    print('t_0_4:')
```

(continues on next page)

(continued from previous page)

```
    print(sympy.pretty(sympy.simplify(t_0_4)))

    t_0_4f = t_0_4.evalf()

    print()
    print('t_0_4f:')
    print(sympy.pretty(sympy.simplify(t_0_4f)))

    return


if __name__ == '__main__':
    main()
```

CHAPTER

FOUR

INDICES AND TABLES

At the website you can navigate through the menus below:

- genindex
- modindex
- search

4.1 Running the documentation with Sphinx

To run the documentation for this project run the following commands, at the project folder:

Install Spinxs:

python -m pip install sphinx

Install the "Read the Docs" theme:

pip install sphinx-rtd-theme

make clean

make html

4.2 GitHub Repository

Find all the files at the GitHub repository here.

PYTHON MODULE INDEX

m

`MatrixManipulation`, 1
`MatrixManipulationSymbolic`, 3