UTS Robotics and Intelligent System

- Nama : Brilliant Friezka Aina
- NIM : 1103194186
- Kelas : TK-43-GAB

```
pip install symforce
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting symforce
  Downloading symforce-0.7.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.4 MB)
     |████████████████████████████████| 4.4 MB 5.3 MB/s
Requirement already satisfied: graphviz in /usr/local/lib/python3.8/dist-packages (from symforce) (0.10.1)
Collecting symforce-sym==0.7.0
  Downloading symforce_sym-0.7.0-py3-none-any.whl (70 kB)
     |████████████████████████████████| 70 kB 5.4 MB/s
Collecting skymarshal==0.7.0
  Downloading skymarshal-0.7.0-py3-none-any.whl (82 kB)
     |████████████████████████████████| 82 kB 426 kB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from symforce) (1.7.3)
Collecting black
  Downloading black-22.10.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 27.5 MB/s
Collecting sympy~=1.11.1
  Downloading sympy-1.11.1-py3-none-any.whl (6.5 MB)
     |████████████████████████████████| 6.5 MB 46.3 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from symforce) (1.21.6)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-packages (from symforce) (2.11.3)
Collecting clang-format
  Downloading clang_format-15.0.4-py2.py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 42.4 MB/s
Collecting ply
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
     |████████████████████████████████| 49 kB 4.0 MB/s
Collecting argh
  Downloading argh-0.26.2-py2.py3-none-any.whl (30 kB)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from skymarshal==0.7.0->symforce) (1.15.0)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.8/dist-packages (from sympy~=1.11.1->symforce) (1.2.1)
Collecting pathspec>=0.9.0
  Downloading pathspec-0.10.2-py3-none-any.whl (28 kB)
Collecting click>=8.0.0
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
     |████████████████████████████████| 96 kB 4.6 MB/s
Collecting mypy-extensions>=0.4.3
  Downloading mypy_extensions-0.4.3-py2.py3-none-any.whl (4.5 kB)
Collecting platformdirs>=2
  Downloading platformdirs-2.5.4-py3-none-any.whl (14 kB)
Requirement already satisfied: tomli>=1.1.0 in /usr/local/lib/python3.8/dist-packages (from black->symforce) (2.0.1)
Requirement already satisfied: typing-extensions>=3.10.0.0 in /usr/local/lib/python3.8/dist-packages (from black->symforce) (4.1.1)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.8/dist-packages (from jinja2->symforce) (2.0.1)
Installing collected packages: ply, platformdirs, pathspec, mypy-extensions, click, argh, sympy, symforce-sym, skymarshal, clang-format,
  Attempting uninstall: click
    Found existing installation: click 7.1.2
    Uninstalling click-7.1.2:
      Successfully uninstalled click-7.1.2
  Attempting uninstall: sympy
    Found existing installation: sympy 1.7.1
    Uninstalling sympy-1.7.1:
      Successfully uninstalled sympy-1.7.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
flask 1.1.4 requires click<8.0,>=5.1, but you have click 8.1.3 which is incompatible.
Successfully installed argh-0.26.2 black-22.10.0 clang-format-15.0.4 click-8.1.3 mypy-extensions-0.4.3 pathspec-0.10.2 platformdirs-2.5.
```

```
import symforce

symforce.set_symbolic_api("sympy")
symforce.set_log_level("warning")

from symforce.notebook_util import display
import symforce.symbolic as sf
from symforce import ops


display(sf.Rot3())
```

```
<Rot3 <Q xyzw=[0, 0, 0, 1]>>
```

```
display(sf.Rot3.symbolic("R"))
```

```
<Rot3 <Q xyzw=[R_x, R_y, R_z, R_w]>>
```

```
theta = sf.Symbol("theta")
R_mat = sf.Matrix(
    [
        [1, 0, 0],
        [0, sf.cos(theta), -sf.sin(theta)],
        [0, sf.sin(theta), sf.cos(theta)],
    ]
)
R = sf.Rot3.from_rotation_matrix(R_mat)

display(R_mat)
display(R)
display(R.to_rotation_matrix())
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

```
<Rot3 <Q xyzw=[(1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1)))*(sqrt(Max(0, 2 - 2*cos(theta)))/2 + Max(0,
sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))/2) + 2*sin(theta)*Min(1 - Max(0, sign(-2*cos(theta) + Max(1, 2*cos(theta)
+ 1, cos(theta)) - 1)), 1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1)), Min(1 - Max(0, sign(-
cos(theta) + Max(1, 2*cos(theta) + 1, cos(theta)))), Max(1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) -
1))), 1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1)), Min(1 - Max(0, 1 - Max(0, sign(Max(1,
2*cos(theta) + 1, cos(theta)) - 1))), 1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1,
cos(theta)))))))))/(2*sqrt(Max(0, 2*cos(theta) + 2)) - 2*Min(1 - Max(0, sign(-2*cos(theta) + Max(1, 2*cos(theta) + 1,
cos(theta)) - 1)), 1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1)), Min(1 - Max(0, sign(-cos(theta)
+ Max(1, 2*cos(theta) + 1, cos(theta)))), Max(1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))), 1 -
Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1)), Min(1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1,
cos(theta)) - 1))), 1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1, cos(theta))))))) + 2), (1/2 - Min(1 - Max(0,
1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))), 1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1,
cos(theta)))))/2)*Min(1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))), 1 - Max(0, sign(-cos(theta)
+ Max(1, 2*cos(theta) + 1, cos(theta))))), (1/2 - Min(1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1,
cos(theta)))), 1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1)), Min(1 - Max(0, 1 - Max(0,
sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))), 1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1,
cos(theta)))))))/2)*Min(1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1, cos(theta)))), 1 - Max(0, 1 - Max(0,
sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1)), Min(1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1,
cos(theta)))))))/2)*Min(1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1, cos(theta)))), 1 - Max(0, 1 - Max(0,
sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))), Min(1 - Max(0, sign(-cos(theta) + Max(1,
2*cos(theta) + 1, cos(theta)))), Max(1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))), 1 - Max(0, 1
- Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1)), Min(1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1,
cos(theta)) - 1))), 1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1, cos(theta))))))))))/2 + 1/2)*Min(1 - Max(0,
sign(-2*cos(theta) + Max(1, 2*cos(theta) + 1, cos(theta)) - 1)), 1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1,
cos(theta)) - 1)), Min(1 - Max(0, sign(-cos(theta) + Max(1, 2*cos(theta) + 1, cos(theta)))), Max(1 - Max(0, 1 - Max(0,
sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))), 1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))),
Min(1 - Max(0, 1 - Max(0, sign(Max(1, 2*cos(theta) + 1, cos(theta)) - 1))), 1 - Max(0, sign(-cos(theta) + Max(1,
2*cos(theta) + 1, cos(theta)))))))))]>>
```

```
R = sf.Rot3.from_yaw_pitch_roll(0, 0, theta)
ypr = R.to_yaw_pitch_roll()

display(R)
display(ops.StorageOps.simplify(list(ypr)))
```

```
<Rot3 <Q xyzw=[sin(theta/2), 0, 0, cos(theta/2)]>>
```

$$[0,\ 0,\ \operatorname{atan}_2(\sin(\theta), \cos(\theta))]$$

```
R = sf.Rot3.from_angle_axis(angle=theta, axis=sf.Vector3(1, 0, 0))

display(R)
```

```
<Rot3 <Q xyzw=[sin(theta/2), 0, 0, cos(theta/2)]>>
```

```
world_R_body = sf.Rot3.symbolic("R")

body_t_point = sf.Vector3.symbolic("p")

world_t_point = world_R_body * body_t_point

display(world_t_point)
```

$$\begin{bmatrix} p_0 \left(-2R_y^2 - 2R_z^2 + 1\right) + p_1 \left(-2R_w R_z + 2R_x R_y\right) + p_2 \cdot \left(2R_w R_y + 2R_x R_z\right) \\ p_0 \cdot \left(2R_w R_z + 2R_x R_y\right) + p_1 \left(-2R_x^2 - 2R_z^2 + 1\right) + p_2 \left(-2R_w R_x + 2R_y R_z\right) \\ p_0 \left(-2R_w R_y + 2R_x R_z\right) + p_1 \cdot \left(2R_w R_x + 2R_y R_z\right) + p_2 \left(-2R_x^2 - 2R_y^2 + 1\right) \end{bmatrix}$$

```
body_R_cam = sf.Rot3.symbolic("R_cam")
world_R_cam = world_R_body * body_R_cam

cam_R_body = body_R_cam.inverse()
display(body_R_cam)
display(cam_R_body)
```

```
<Rot3 <Q xyzw=[R_cam_x, R_cam_y, R_cam_z, R_cam_w]>>
<Rot3 <Q xyzw=[-R_cam_x, -R_cam_y, -R_cam_z, R_cam_w]>>
```

```
world_R_body_numeric = sf.Rot3.from_yaw_pitch_roll(0.1, -2.3, 0.7)
display(world_t_point.subs(world_R_body, world_R_body_numeric))
```

$$\begin{bmatrix} -0.662947416398295 p_0 - 0.554353314451006 p_1 - 0.503182994394693 p_2 \\ -0.0665166116342196 p_0 + 0.713061539471145 p_1 - 0.697938952419008 p_2 \\ 0.74570521217672 p_0 - 0.429226797490819 p_1 - 0.509596009450867 p_2 \end{bmatrix}$$

```
world_T_body = sf.Pose3.symbolic("T")
display(world_T_body)
```

```
<Pose3 R=<Rot3 <Q xyzw=[T.R_x, T.R_y, T.R_z, T.R_w]>>, t=(T.t0, T.t1, T.t2)>
```

```
world_R_body = sf.Rot3.symbolic("R")

world_t_body = sf.Vector3.symbolic("t")

world_T_body = sf.Pose3(R=world_R_body, t=world_t_body)
display(world_T_body)
```

```
<Pose3 R=<Rot3 <Q xyzw=[R_x, R_y, R_z, R_w]>>, t=(t0, t1, t2)>
```

```
body_T_cam = sf.Pose3.symbolic("T_cam")
world_T_cam = world_T_body * body_T_cam

body_t_point = sf.Vector3.symbolic("p")
world_t_point = world_T_body * body_t_point
display(world_t_point)
```

$$\begin{bmatrix} p_0 \left(-2R_y^2 - 2R_z^2 + 1\right) + p_1 \left(-2R_w R_z + 2R_x R_y\right) + p_2 \cdot \left(2R_w R_y + 2R_x R_z\right) + t_0 \\ p_0 \cdot \left(2R_w R_z + 2R_x R_y\right) + p_1 \left(-2R_x^2 - 2R_z^2 + 1\right) + p_2 \left(-2R_w R_x + 2R_y R_z\right) + t_1 \\ p_0 \left(-2R_w R_y + 2R_x R_z\right) + p_1 \cdot \left(2R_w R_x + 2R_y R_z\right) + p_2 \left(-2R_x^2 - 2R_y^2 + 1\right) + t_2 \end{bmatrix}$$

```
body_T_world = world_T_body.inverse()
display(world_T_body)
display(body_T_world)
```

```
<Pose3 R=<Rot3 <Q xyzw=[R_x, R_y, R_z, R_w]>>, t=(t0, t1, t2)>
<Pose3 R=<Rot3 <Q xyzw=[-R_x, -R_y, -R_z, R_w]>>, t=(-t0*(-2*R_y**2 - 2*R_z**2 + 1) - t1*(2*R_w*R_z + 2*R_x*R_y) - t2*
(-2*R_w*R_y + 2*R_x*R_z), -t0*(-2*R_w*R_z + 2*R_x*R_y) - t1*(-2*R_x**2 - 2*R_z**2 + 1) - t2*(2*R_w*R_x + 2*R_y*R_z), -t0*
(2*R_w*R_y + 2*R_x*R_z) - t1*(-2*R_w*R_x + 2*R_y*R_z) - t2*(-2*R_x**2 - 2*R_y**2 + 1))>
```

```
m1 = sf.Matrix([[1, 2, 3], [4, 5, 6]])

m2 = sf.Matrix(2, 3, [1, 2, 3, 4, 5, 6])

m3 = sf.Matrix23(1, 2, 3, 4, 5, 6)
m4 = sf.Matrix23([1, 2, 3, 4, 5, 6])

m5 = sf.M([[1, 2, 3], [4, 5, 6]])
m6 = sf.M(2, 3, [1, 2, 3, 4, 5, 6])
m7 = sf.M23(1, 2, 3, 4, 5, 6)
m8 = sf.M23([1, 2, 3, 4, 5, 6])

m9 = sf.Matrix23.block_matrix([[sf.M13([1, 2, 3])], [sf.M13([3, 4, 5])]])

v1 = sf.Matrix([[1], [2], [3]])

v2 = sf.Matrix([1, 2, 3])
```

```
v3 = sf.Matrix31(1, 2, 3)
v4 = sf.M31(1, 2, 3)
v5 = sf.Vector3(1, 2, 3)
v6 = sf.V3(1, 2, 3)


z1 = sf.Matrix23.zero()
z2 = sf.Matrix.zeros(2, 3)

o1 = sf.Matrix23.one()
o2 = sf.Matrix.ones(2, 3)


zero_matrix = sf.Matrix33.zero()
identity_matrix = sf.Matrix33.eye()

zero_matrix = ops.GroupOps.identity(sf.Matrix33)

display(zero_matrix)
display(identity_matrix)
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
m23 = sf.M23.symbolic("lhs")
m31 = sf.V3.symbolic("rhs")
display(m23 * m31)
```

$$\begin{bmatrix} lhs_{00}rhs_0 + lhs_{01}rhs_1 + lhs_{02}rhs_2 \\ lhs_{10}rhs_0 + lhs_{11}rhs_1 + lhs_{12}rhs_2 \end{bmatrix}$$

```
norm = m31.norm()
squared_norm = m31.squared_norm()
unit_vec = m31.normalized()
display(unit_vec)
```

$$\begin{bmatrix} \frac{rhs_0}{\sqrt{rhs_0^2+rhs_1^2+rhs_2^2}} \\ \frac{rhs_1}{\sqrt{rhs_0^2+rhs_1^2+rhs_2^2}} \\ \frac{rhs_2}{\sqrt{rhs_0^2+rhs_1^2+rhs_2^2}} \end{bmatrix}$$

```
m33 = 5 * sf.Matrix33.eye()
display(m33.inv())
```

$$\begin{bmatrix} \frac{1}{5} & 0 & 0 \\ 0 & \frac{1}{5} & 0 \\ 0 & 0 & \frac{1}{5} \end{bmatrix}$$

```
R0 = sf.Rot3.symbolic("R0")
R1 = sf.Rot3.symbolic("R1")
residual = sf.M(R0.local_coordinates(R1))
display(residual)
```

$$\begin{bmatrix} \frac{2\cdot(2\min(0,\operatorname{sign}(R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}))+1)(R_{0w}R_{1x}-R_{0x}R_{1w}-R_{0y}R_{1z}+R_{0z}R_{1y})\operatorname{acos}(\min(1.0,|R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}|))}{\sqrt{1-\min(1.0,|R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}|)^2}} \\ \frac{2\cdot(2\min(0,\operatorname{sign}(R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}))+1)(R_{0w}R_{1y}+R_{0x}R_{1z}-R_{0y}R_{1w}-R_{0z}R_{1x})\operatorname{acos}(\min(1.0,|R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}|))}{\sqrt{1-\min(1.0,|R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}|)^2}} \\ \frac{2\cdot(2\min(0,\operatorname{sign}(R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}))+1)(R_{0w}R_{1z}-R_{0x}R_{1y}+R_{0y}R_{1x}-R_{0z}R_{1w})\operatorname{acos}(\min(1.0,|R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}|))}{\sqrt{1-\min(1.0,|R_{0w}R_{1w}+R_{0x}R_{1x}+R_{0y}R_{1y}+R_{0z}R_{1z}|)^2}} \end{bmatrix}$$

```
jacobian = residual.jacobian(R1)

display(jacobian.shape)
```

$$(3,\ 3)$$

```
rot = sf.Rot3()
elements = rot.to_storage()
assert len(elements) == rot.storage_dim()
display(elements)
```

$$[0,\ 0,\ 0,\ 1]$$

```
rot2 = sf.Rot3.from_storage(elements)
assert rot == rot2
```

```
rot_sym = sf.Rot3.symbolic("rot_sym")
rot_num = rot_sym.subs(rot_sym, rot)

display(rot_sym)
display(rot_num)
display(rot_num.simplify())
display(rot_num.evalf())
```

```
<Rot3 <Q xyzw=[rot_sym_x, rot_sym_y, rot_sym_z, rot_sym_w]>>
<Rot3 <Q xyzw=[0, 0, 0, 1]>>
<Rot3 <Q xyzw=[0, 0, 0, 1]>>
<Rot3 <Q xyzw=[0, 0, 0, 1.00000000000000]>>
```

```
R1 = sf.Rot3.random()
R2 = sf.Rot3.random()

display(R1.compose(R2))
```

```
<Rot3 <Q xyzw=[0.246284344195380, -0.870464946784933, 0.200351758214716, 0.376156843887264]>>
```

```
R_identity = sf.Rot3.identity()
display(R1)
display(R_identity * R1)
```

```
<Rot3 <Q xyzw=[0.0750292497795149, 0.559705991078823, -0.228445619351984, 0.793039982741655]>>
<Rot3 <Q xyzw=[0.0750292497795149, 0.559705991078823, -0.228445619351984, 0.793039982741655]>>
```

```
R1_inv = R1.inverse()
display(R_identity)
display(R1_inv * R1)
```

```
<Rot3 <Q xyzw=[0, 0, 0, 1]>>
<Rot3 <Q xyzw=[0, 0, 0, 1.00000000000000]>>
```

```
R_delta = R1.between(R2)
display(R1 * R_delta)
display(R2)
```

```
<Rot3 <Q xyzw=[0.253806390770475, -0.829555923827274, 0.447975492923884, -0.216187980671325]>>
<Rot3 <Q xyzw=[0.253806390770475, -0.829555923827274, 0.447975492923884, -0.216187980671325]>>
```

```
R1 = sf.Rot3.random()
tangent_vec = R1.to_tangent()
R1_recovered = sf.Rot3.from_tangent(tangent_vec)

assert len(tangent_vec) == R1.tangent_dim()
display(R1)
display(R1_recovered)
```

```
<Rot3 <Q xyzw=[-0.868754989606119, -0.450984750989883, 0.0921587820603280, -0.182713659309375]>>
<Rot3 <Q xyzw=[0.868754989606119, 0.450984750989883, -0.0921587820603280, 0.182713659309375]>>
```

```
R2 = R1.retract([0.1, 2.3, -0.5])

recovered_tangent_vec = R1.local_coordinates(R2)

display(recovered_tangent_vec)
```

$$[0.1,\ 2.3,\ -0.5]$$

```
jacobian = R1.storage_D_tangent()
assert jacobian.shape == (R1.storage_dim(), R1.tangent_dim())
```

✓ 0s completed at 12:32 PM