This lecture will be recorded

ETH zürich

COMPAS

064-0026-00L: COMPAS II

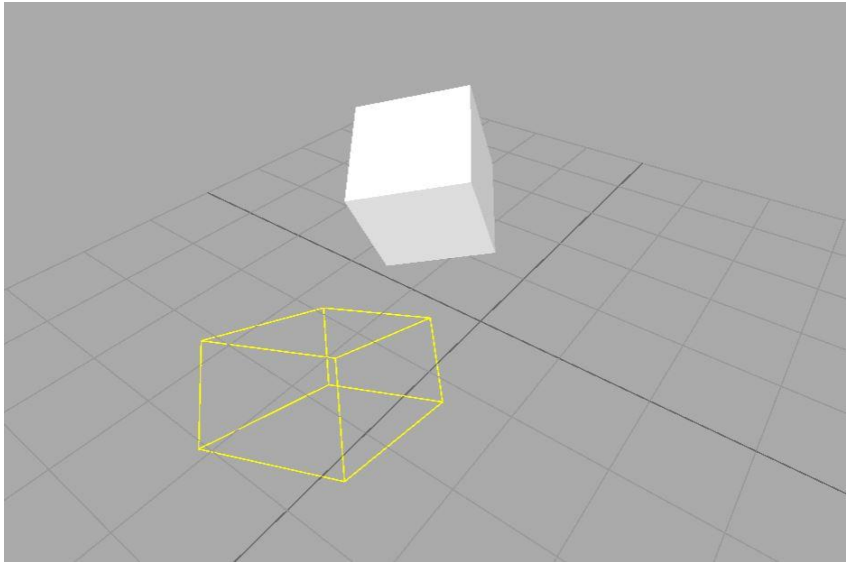
Introduction to Computational Methods for Digital Fabrication in Architecture slides and code

`https://tiny.cc/compas-ii`

Review of last week's assignment

Project box to xy-plane

1. Create a box at a certain location with a certain orientation.
2. Create a `Projection` (can be orthogonal, parallel or perspective)
3. Convert the box to a mesh and project the it onto the xy-plane.
4. Use artists to draw the result



TODAY

robot models

forward kinematics

inverse kinematics

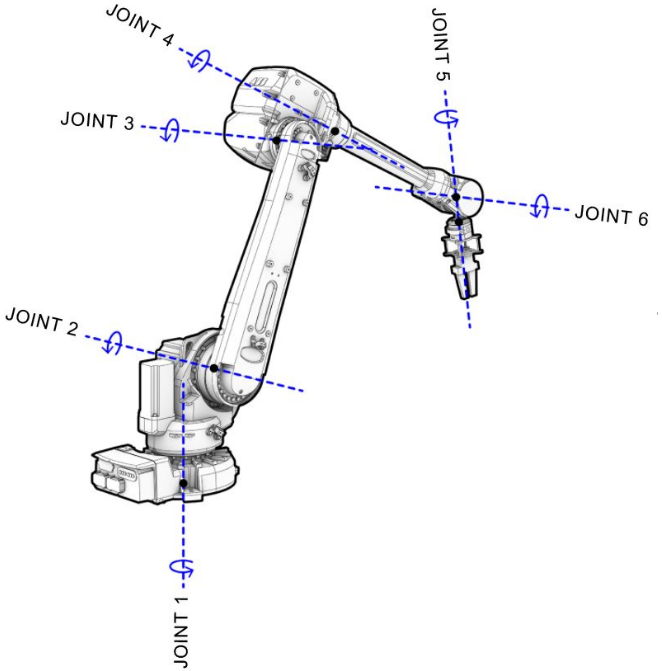
Today's goal

Understand **how a robot is represented** in COMPAS

robot models

forward kinematics

inverse kinematics



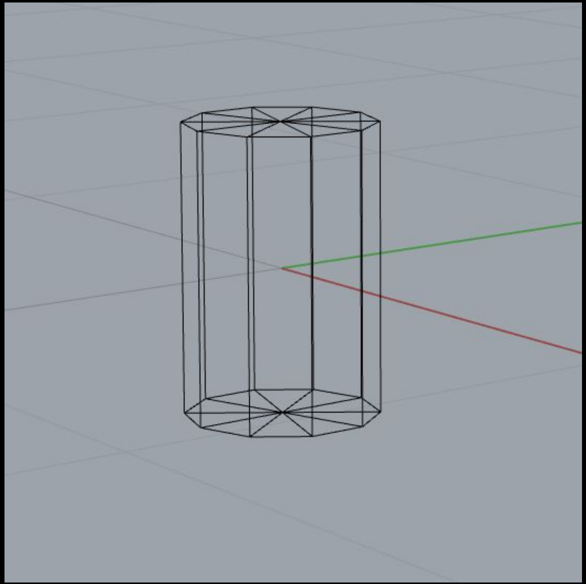
URDF format

Tree structure

Open source

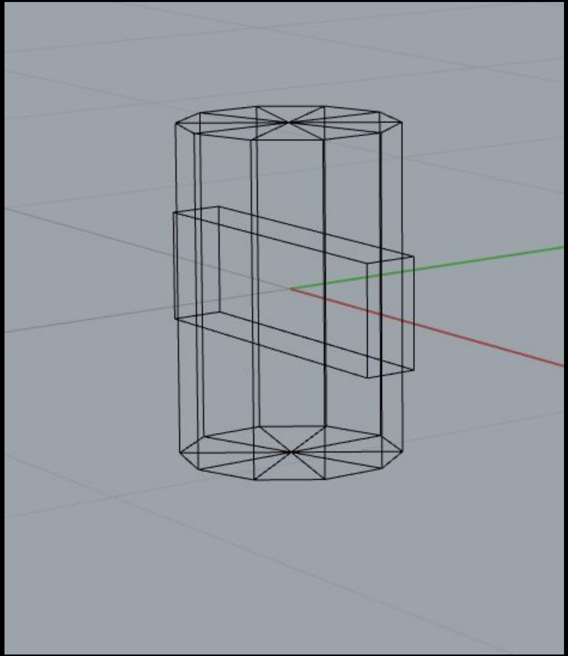
```xml
<?xml version="1.0"?>
<robot name="myfirst">
 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>
</robot>
```

```xml
<?xml version="1.0"?>
<robot name="myfirst">
 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>
</robot>
```



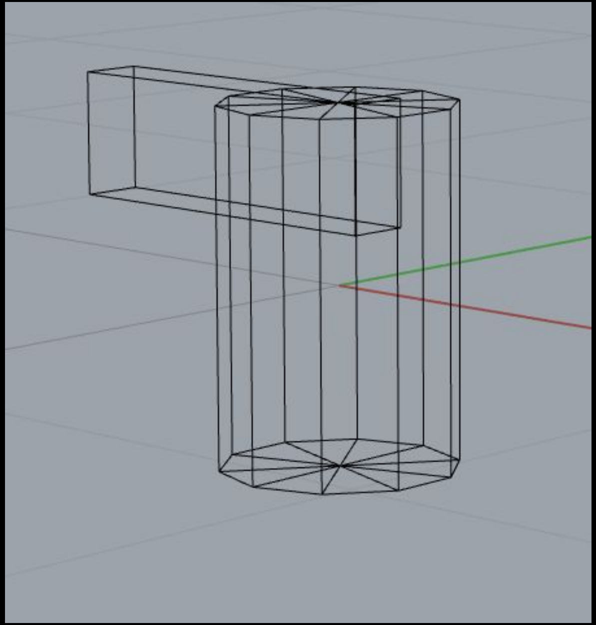
```xml
<?xml version="1.0"?>
<robot name="multipleshapes">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
 </joint>

</robot>
```

```xml
<?xml version="1.0"?>
<robot name="multipleshapes">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
 </joint>

</robot>
```



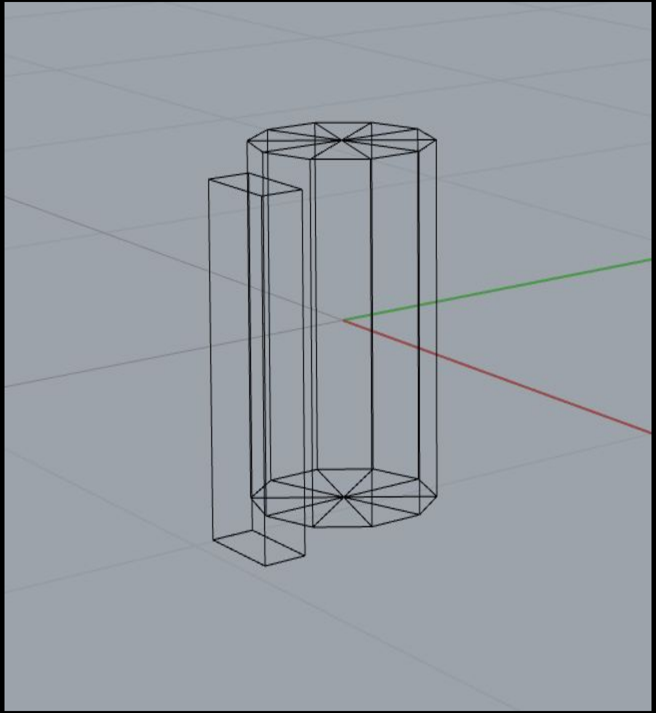
```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```

```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```



```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
     <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```

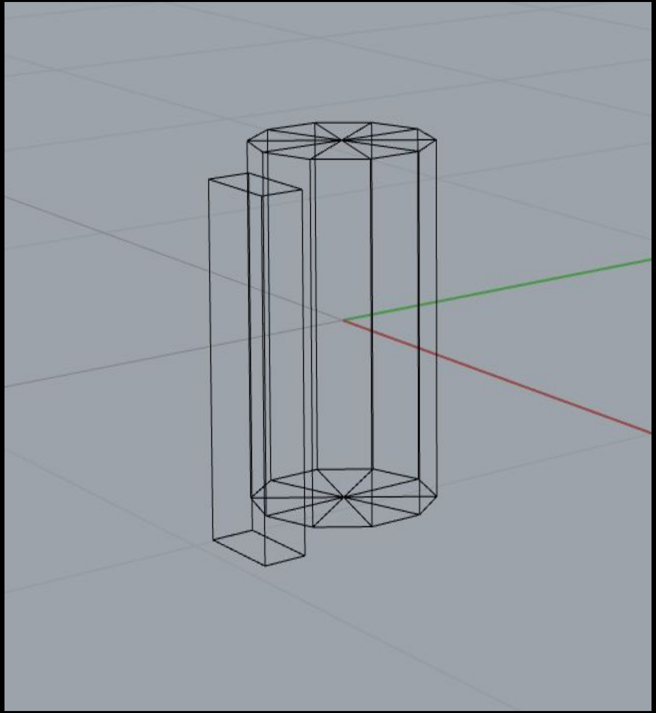
```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
     <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```



Visualize model

```
from compas_rhino.artists import RobotModelArtist
from compas.robots import RobotModel

model = RobotModel.from_urdf_file('models/01_myfirst.urdf')

artist = RobotModelArtist(model, layer='COMPAS::Robot Viz')
artist.clear_layer()
artist.draw_visual()
```

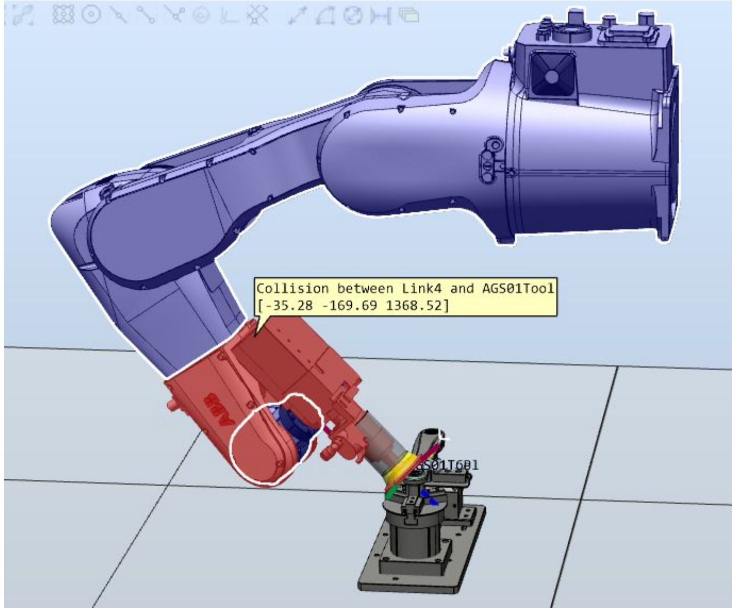
```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <mesh filename="package://basic/cylinder.obj"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <mesh filename="package://basic/box.obj"/>
     </geometry>
     <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```



Visualize model with meshes

```
from compas_rhino.artists import RobotModelArtist
from compas.robots import RobotModel
from compas.robots import LocalPackageMeshLoader

model = RobotModel.from_urdf_file('models/05_origins_meshes.urdf')

loader = LocalPackageMeshLoader('models', 'basic')
model.load_geometry(loader)

artist = RobotModelArtist(model, layer='COMPAS::Robot Viz')
artist.clear_layer()
artist.draw_visual()
```

Collision checking



Source: https://forums.robotstudio.com/discussion/10611/how-to-generate-collision-free-path-with-powerpacs

Use different visual/collision geometry

Use bounding volumes

Use primitives

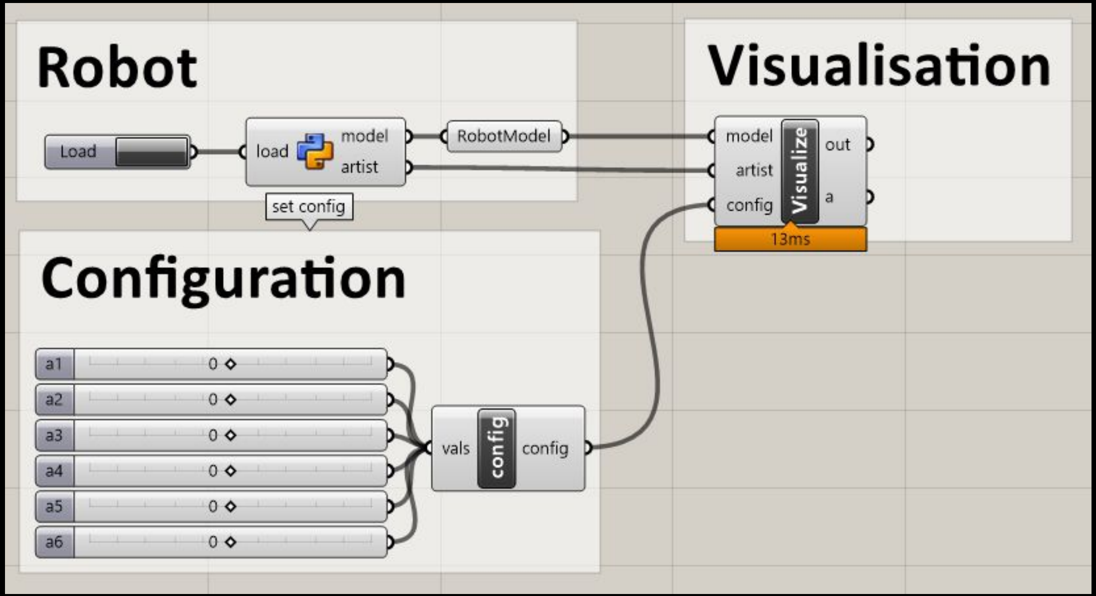
Load local model

```
import compas
from compas.robots import LocalPackageMeshLoader
from compas.robots import RobotModel

compas.PRECISION = '12f'

loader = LocalPackageMeshLoader('models', 'ur_description')
model = RobotModel.from_urdf_file(loader.load_urdf('ur5.urdf'))
model.load_geometry(loader)
```

Load Github model

```
import compas
from compas.robots import GithubPackageMeshLoader
from compas.robots import RobotModel

compas.PRECISION = '12f'

r = 'ros-industrial/abb'
p = 'abb_irb6600_support'
b = 'kinetic-devel'

github = GithubPackageMeshLoader(r, p, b)
urdf = github.load_urdf('irb6640.urdf')

model = RobotModel.from_urdf_file(urdf)
model.load_geometry(github)
```

Grasshopper



Loading external models

```
# Set high precision to import meshes defined in meters
compas.PRECISION = '12f'

# Load robot and its geometry
with RosClient('localhost') as ros:
    robot = ros.load_robot(load_geometry=True)
    print(robot.model)
```

Exercise

Building your own robot

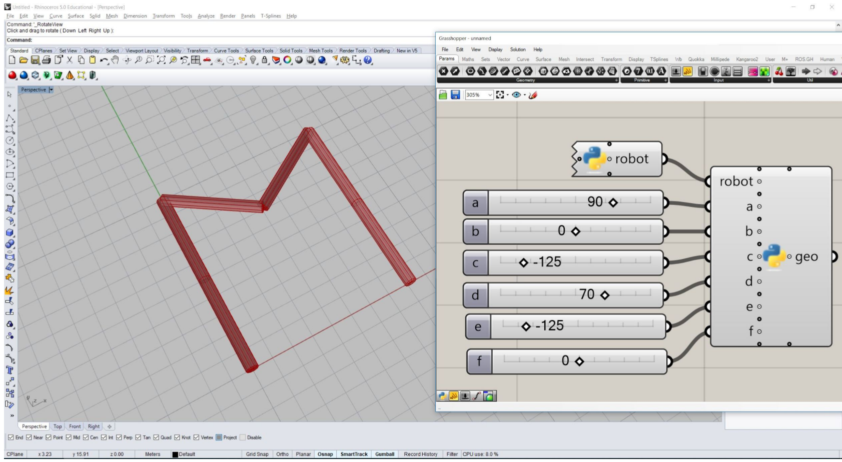
```
model = RobotModel('ur10e',
              joints=[
                  Joint('shoulder_pan_joint', 'revolute', parent='base_link', child='shoulder_link'),
                  Joint('shoulder_lift_joint', 'revolute', parent='shoulder_link', child='upper_arm_link'),
                  Joint('elbow_joint', 'revolute', parent='upper_arm_link', child='forearm_link'),
                  Joint('wrist_1_joint', 'revolute', parent='forearm_link', child='wrist_1_link'),
                  Joint('wrist_2_joint', 'revolute', parent='wrist_1_link', child='wrist_2_link'),
                  Joint('wrist_3_joint', 'revolute', parent='wrist_2_link', child='wrist_3_link'),
              ], links=[
                  Link('base_link'),
                  Link('shoulder_link'),
                  Link('upper_arm_link'),
                  Link('forearm_link'),
                  Link('wrist_1_link'),
                  Link('wrist_2_link'),
                  Link('wrist_3_link'),
              ])
print(model)
```

Building your own robot

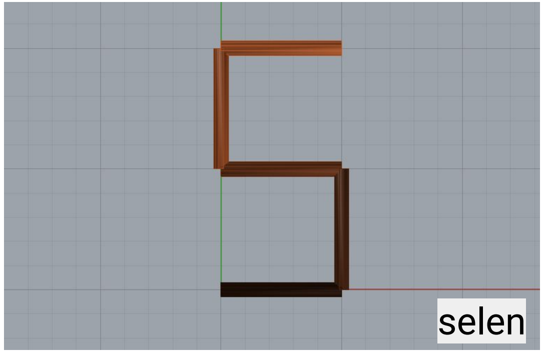
```
# create robot model
model = RobotModel("robot", links=[], joints=[])

# add links
link0 = model.add_link("world")
link1 = model.add_link("link1", visual_mesh=mesh1,)
link2 = model.add_link("link2", visual_mesh=mesh2,)

# add the joints between the links
model.add_joint("joint1", Joint.REVOLUTE, link0, link1, origin1, axis1)
model.add_joint("joint2", Joint.REVOLUTE, link1, link2, origin2, axis2)
```

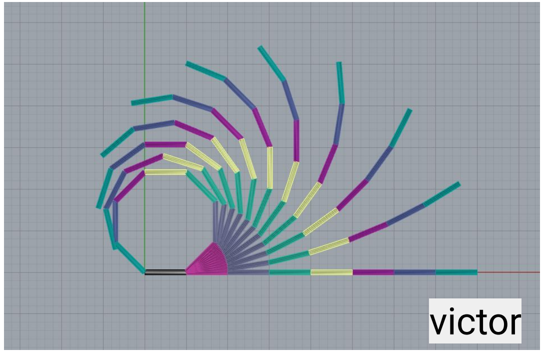
Assignment

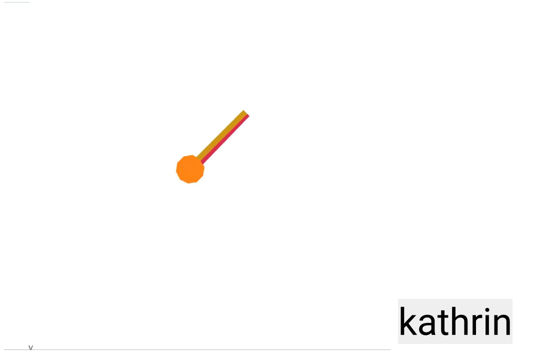
1. Build your own robot with a certain number n of links and n - 1 configurable joints.
2. Create a `Configuration` with certain values and the correct joint types.
3. Create a `RobotModelArtist` of your preference (e.g. `compas_ghpython` or `compas_rhino`)
4. Use the artist to `update` the robot with the created configuration (using its `joint_dict`), such that it configures into the letter of your choice (or any other identifiable figure).

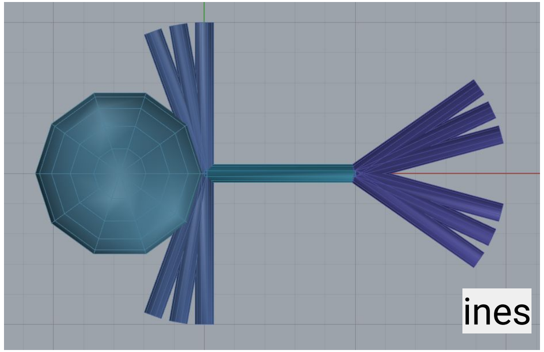


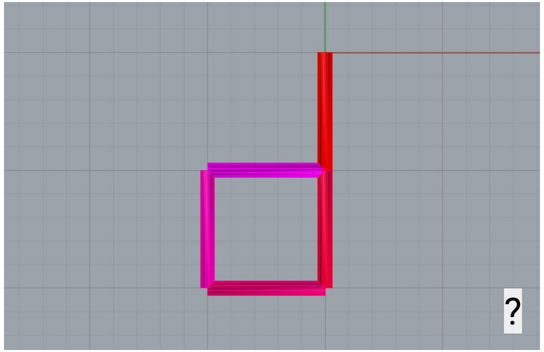
Robot gallery

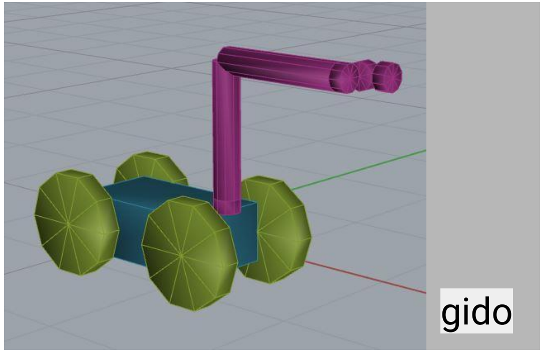










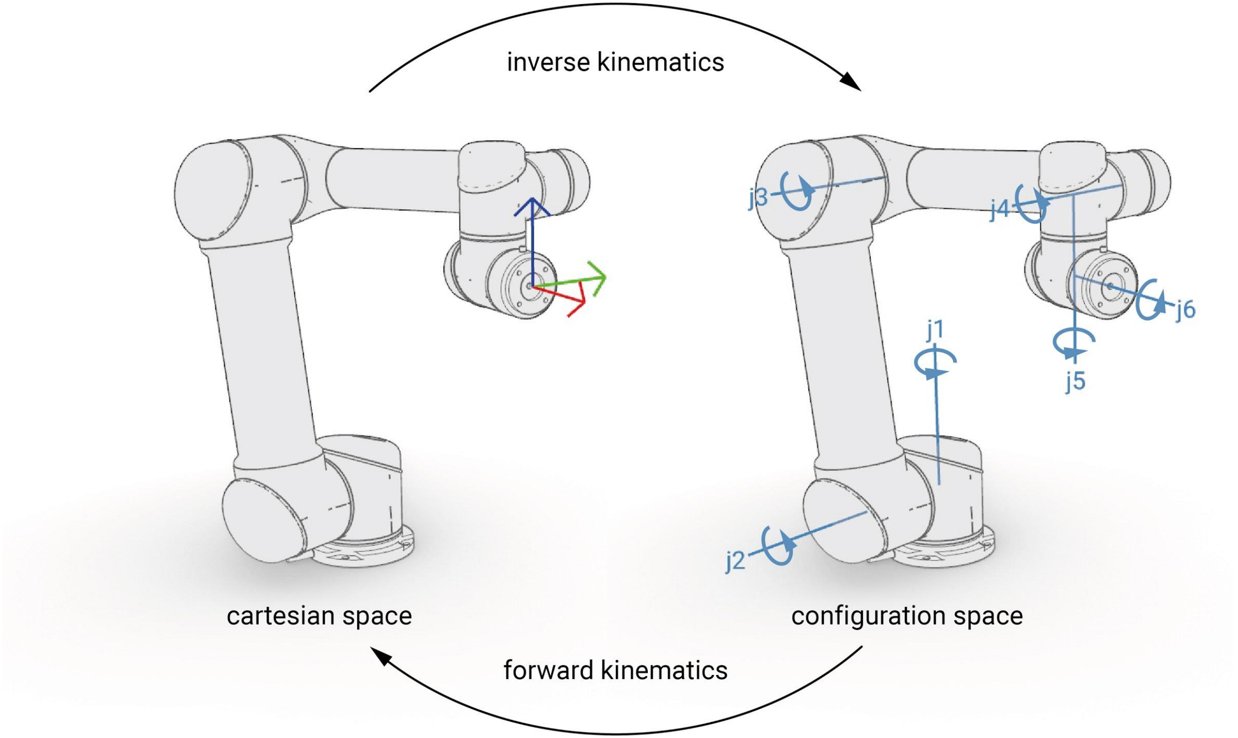


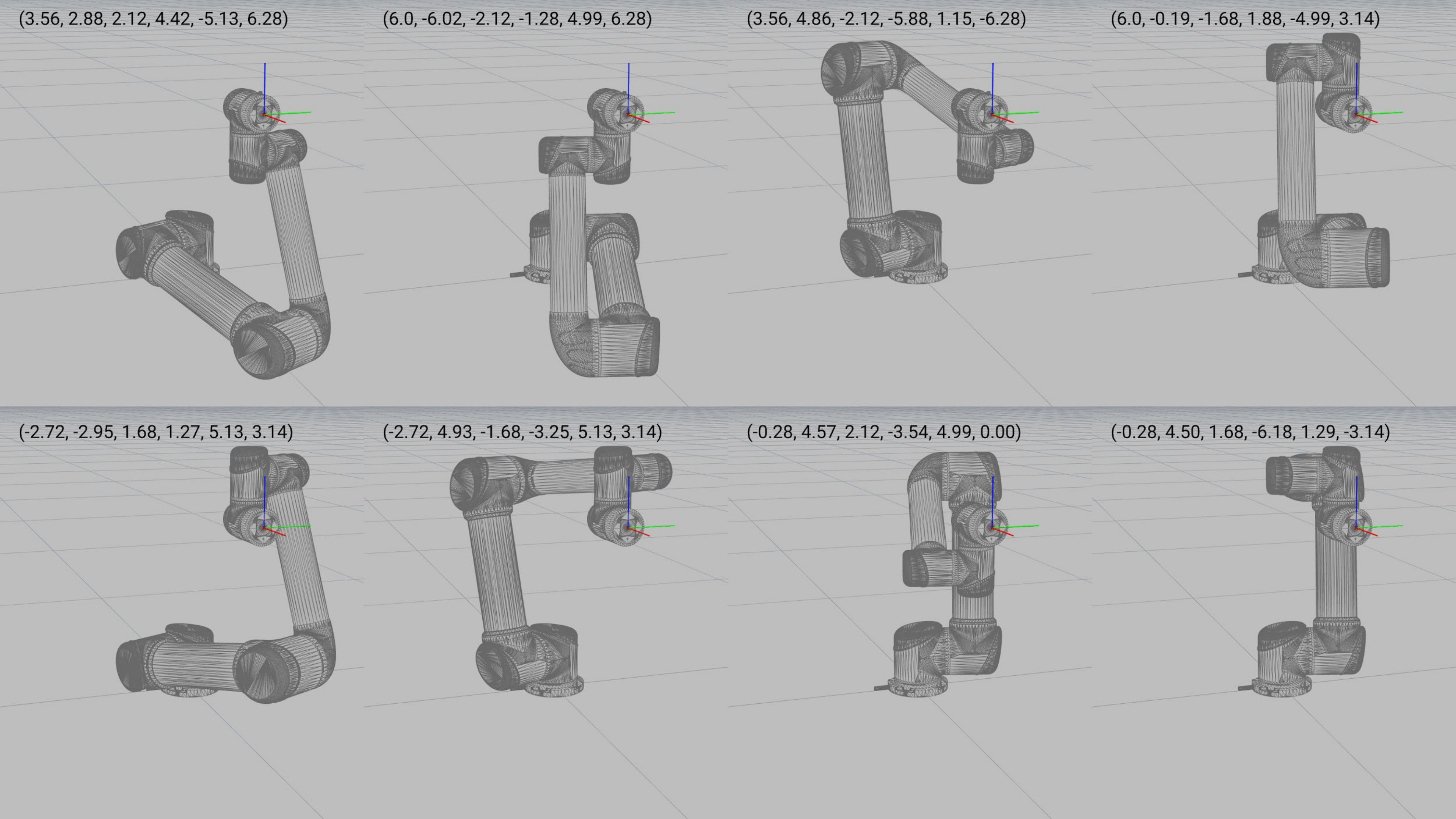
robot models

forward kinematics

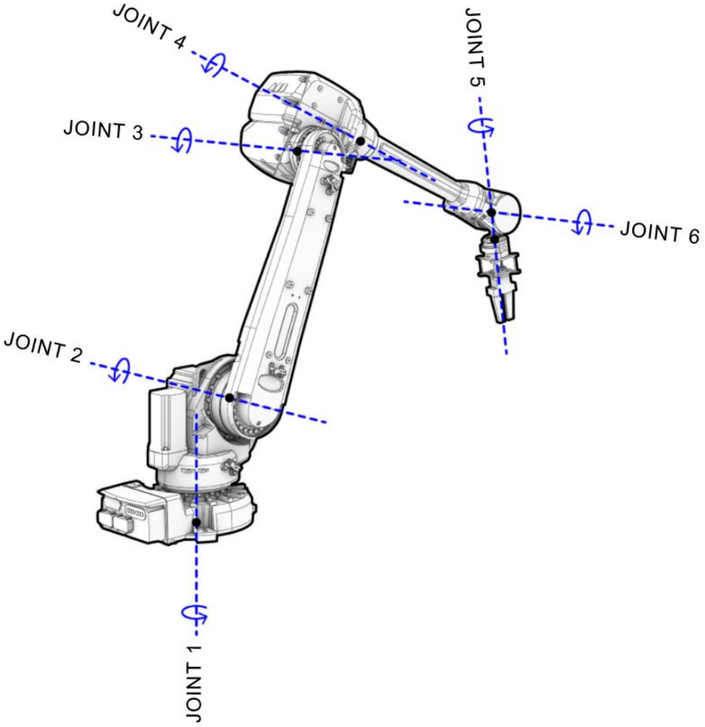
inverse kinematics

Joint vs Cartesian space



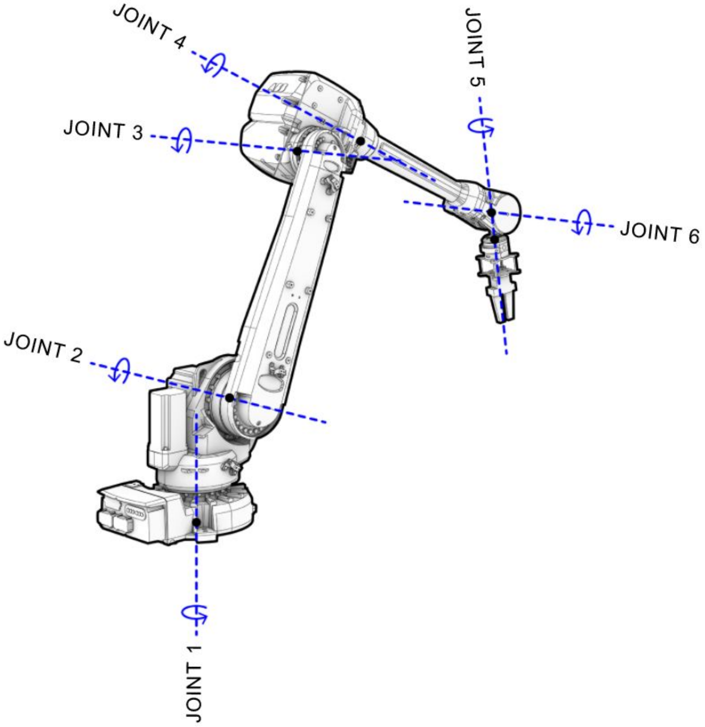

(3.56, 2.88, 2.12, 4.42, -5.13, 6.28)
(6.0, -6.02, -2.12, -1.28, 4.99, 6.28)
(3.56, 4.86, -2.12, -5.88, 1.15, -6.28)
(6.0, -0.19, -1.68, 1.88, -4.99, 3.14)
(-2.72, -2.95, 1.68, 1.27, 5.13, 3.14)
(-2.72, 4.93, -1.68, -3.25, 5.13, 3.14)
(-0.28, 4.57, 2.12, -3.54, 4.99, 0.00)
(-0.28, 4.50, 1.68, -6.18, 1.29, -3.14)

Configuration




```python
from compas.robots import Joint

print(Joint.REVOLUTE)
print(Joint.PRISMATIC)
print(Joint.FIXED)
```

Configuration




```python
from math import pi
from compas_fab.robots import Configuration

values = [1.5, pi]
types = [Joint.PRISMATIC, Joint.REVOLUTE]
config = Configuration(values, types)

config =
Configuration.from_revolute_values([pi/2, 0., 0.,
pi/2, pi, 0])

config =
Configuration.from_prismatic_and_revolute_values(
[8.312], [pi/2, 0., 0., 0., 2*pi, 0.8])
```

Forward Kinematics

```
from compas.robots import LocalPackageMeshLoader
from compas.robots import RobotModel

loader = LocalPackageMeshLoader('models', 'ur_description')
model = RobotModel.from_urdf_file(loader.load_urdf('ur5.urdf'))

joints = dict(shoulder_pan_joint=0, shoulder_lift_joint=0, elbow_joint=0,
              wrist_1_joint=0, wrist_2_joint=0, wrist_3_joint=0)

frame = model.forward_kinematics(joints)
```

robot models

forward kinematics

inverse kinematics

Inverse Kinematics without backend?

Inverse Kinematics

```
from ur_kinematics import inverse_kinematics_ur5

loader = LocalPackageMeshLoader('models', 'ur_description')
model = RobotModel.from_urdf_file(loader.load_urdf('ur5.urdf'))

sols = inverse_kinematics_ur5(frame)
```

Next week

- Assignment submission due: Wed 17th March, 9AM.
- Ask for help if needed: Slack, Forum, Office Hours (Fridays, request via Slack)
- Next week:
  - Robot backends: ROS

Thanks!