

Installation

```
conda install compas_fab
```

Anaconda/Miniconda must be installed and the conda-forge channel must be added. Use virtual environments to avoid dependency issues.

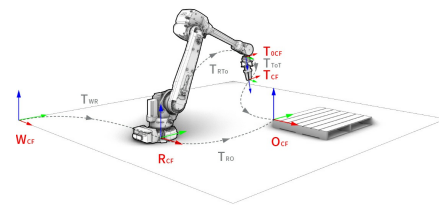
Frames

<code>Frame.w or ldXY()</code>	World XY frame.
<code>Frame(pt, xaxis, yaxis)</code>	Create new frame with origin and vectors x and y.
<code>Frame(pt1, pt2, pt3)</code>	Create new frame from 3 points.
<code>Frame.from_euler_angles(y, p, r)</code>	Create new frame from euler angles (yaw, pitch, roll).
<code>frame.p oint</code>	Origin of the frame.
<code>frame.n ormal, frame.z axis</code>	Normal of the frame.
<code>frame.q ua ternion</code>	Rotation as quaternion.
<code>frame.axis_angle_vector</code>	Rotation as axis-angle vector.
<code>frame.e ul er_ ang les()</code>	Rotation as euler angles.
<code>frame.euler_angles(False)</code>	Rotation as non-static euler angles.

Frames belong to the robotics fundamentals. Each joint contains a frame and there are several frames as coordinate systems involved (eg. `wcf`, `rcf`).

```
from compas.ge ometry import Frame
```

Frames as cartesian coordinate systems



Coordinate systems

<code>wcf</code>	Root coordinate frame of the world.
<code>rcf</code>	Robot coordinate frame, usually on the base of the robot.
<code>t0cf</code>	Tool0 coordinate frame on the last link of the robot.
<code>tcf</code>	Tool coordinate frame on the tool of the robot.
<code>ocf</code>	Object coordinate frame, origin of the work area.
<code>rcf.to_world_coords(pt)</code>	Transform point in local coordinates of <code>rcf</code> to world coordinates.
<code>rcf.to_local_coords(pt)</code>	Transform point in world coordinates to local coordinates of <code>rcf</code> .

Transformations

<code>Transformation.from_frame(f)</code>	Create transformation from world XY to <code>f</code> frame.
<code>Translation([10, 5, 0])</code>	Create translation along <code>x=10.0, y=5.0, z=0.0</code> vector.



By [gonzalocasas](#)

Published 17th November, 2019.

Last updated 17th November, 2019.

Page 1 of 4.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Transformations (cont)

<code>t.inverse()</code>	Calculate inverse of <code>t</code> transformation.
<code>pt.transform(t)</code>	In-place transform point <code>pt</code> with <code>t</code> transformation.
<code>pt.transform(t * t2 * t3)</code>	In-place transform point <code>pt</code> with <code>t</code> , <code>t2</code> and <code>t3</code> transformations.
<code>pt2 = pt.transformed(t)</code>	Return a transformed copy of point <code>pt</code> with <code>t</code> transformation.

Transformation represents a 4x4 transformation matrix. Translation, Scale, Reflection, Shear and Projection are specialized sub-classes of it.

```
from compas.geometry import Transformation, Translation, Rotation # , ...
```

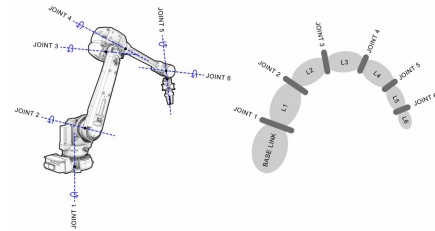
Geometric primitives

<code>Point(3, 2, 5)</code> or <code>[3, 2, 5]</code>	3D point with <code>x=3.0</code> , <code>y=2.0</code> , <code>z=5.0</code> coordinates.
<code>Vector(1, 0, 0)</code> or <code>[1, 0, 0]</code>	Vector <code>x=1.0</code> , <code>y=0.0</code> , <code>z=0.0</code>
<code>Frame(point, xaxis, yaxis)</code> or <code>Frame(point, xaxis, yaxis)</code>	Frame at point origin and <code>x</code> axis and <code>yaxis</code> vectors.

In COMPAS, geometric primitives can be created either as objects or as iterables (lists, tuples, etc).

```
from compas.geometry import Point, Vector, Frame
```

Robot models



Parts of robot model

Link	Contains visual & collision meshes, inertial info, etc.
Joint	Contains parent, child links, origin frame, etc.
RobotModel	Root of robot model tree.

Robot models are a tree structure of links and joints based on URDF format.

```
from compas.robots import RobotModel, Link, Joint
```

Joint types and units

Joint.PRISMATIC	Meters
Joint.REVOLUTE	Radians
Joint.CONTINUOUS	Radians
Joint.FIXED	-

Configuration examples

```
Configuration([.5, pi], [Joint.PRISMATIC, Joint.REVOLUTE])
Configuration.from_revolute_values([0, 0, pi, 0, 0, 0])
Configuration.from_prismatic_and_revolute_values([8.3], [0.0] * 6)
```

Configuration describes the positions of each of the joints of a robot model in its corresponding unit.

```
from compas_fab.robots import Configuration
```



By gonzalocasas

cheatography.com/gonzalocasas/

Published 17th November, 2019.

Last updated 17th November, 2019.

Page 2 of 4.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

ROS backend



Start ROS backend

Download or create a `docker-compose.yml` file ([examples](#)) and start up with:

```
docker-compose up -d
```

Load robot from ROS

```
from compas_fab.backends import RosClient
with RosClient('localhost') as client:
    robot = client.load_robot()
```

If using Docker Toolbox, replace localhost with **192.168.99.100**.

Robotic planning with MoveIt!



Forward Kinematics (FK)

Input: Configuration.
Output: Frame in `rcf`.

Calculate end-effector frame in `rcf` for a given configuration.

Code example

```
config = Configuration.from_revolute_values([0, 0, 0, 3.14, 0, 0])
frame_rcf = robot.forward_kinematics(config)
```

Inverse Kinematics (IK)

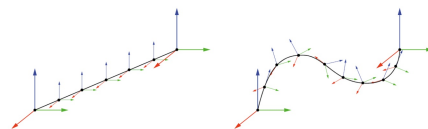
Input: Frame in `wcf` and start Configuration.
Output: Configuration

Calculate possible configuration(s) for a given frame in `wcf`.

Code example

```
frame_wcf = Frame([0.3, 0.1, 0.5], [1, 0, 0], [0, 1, 0])
start_config = robot.init_configuration()
config = robot.inverse_kinematics(frame_wcf, start_config)
```

Path planning



Cartesian vs free-space planning

Plan cartesian path

Input: List of Frame in `wcf` and start Configuration
Output: JointTrajectory

Calculate linear joint trajectory for a given list of waypoints defined by frames.

This might return partial solutions, `fraction` attribute indicates degree of completion, e.g. `trajectory.fraction = 0.5` means 50% of trajectory completed.



By [gonzalocasas](#)

Published 17th November, 2019.
Last updated 17th November, 2019.
Page 3 of 4.

Sponsored by [CrosswordCheats.com](#)
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Code example

```
f1 = Frame([0.3, 0.1, 0.5], [1, 0, 0], [0, 1, 0])
f2 = Frame([0.6, 0.1, 0.4], [1, 0, 0], [0, 1, 0])
start_config = robot.init_configuration()
trajectory = robot.plan_cartesian_motion([f1, f2], start_config)
```

Plan free-space motion

Input List of goal Constraint and start Configuration.

Output JointTrajectory.

Calculate joint trajectory for a given start configuration and a list of goal constraints.

```
from compas_fab.robots import JointTrajectory,
    Constraint, JointConstraint, PositionConstraint,
    OrientationConstraint
```

Code example

```
pc = robot.constraints_from_frame(f1)
start_config = robot.init_configuration()
trajectory = robot.plan_motion(pc, start_config,
    planner_id='RRT')
```

Planning scene operations

```
scene = PlanningScene(robot)
scene.add_collision_mesh(CollisionMesh(mesh, 'floor'))
scene.remove_collision_mesh('floor')
scene.append_collision_mesh(CollisionMesh(mesh, 'brick'))
```

Append will group multiple meshes under the same name, and they can be removed with a single call to remove for that name.

```
from compas_fab.robots import PlanningScene, CollisionMesh
```

Attach meshes to robot

```
scene = PlanningScene(robot)
gripper = CollisionMesh(mesh, 'gripper')
scene.attach_collision_mesh_to_robot(end_effector(gripper))
scene.remove_attached_collision_mesh('gripper')
beam = CollisionMesh(mesh, 'beam')
acm = AttachedCollisionMesh(beam, 'ee_link', ['ee_link'])
scene.add_attached_collision_mesh(acm)
```

These operations can be used to attach an end-effector geometry, or to attach an element to the end-effector itself.

```
from compas_fab.robots import PlanningScene,
    CollisionMesh, AttachedCollisionMesh
```



By gonzalocasas

Published 17th November, 2019.

Last updated 17th November, 2019.

Page 4 of 4.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>