COMPAS FAB Cheat Sheet

by gonzalocasas via cheatography.com/101168/cs/21156/

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.

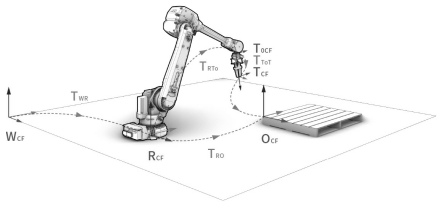
Frames

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

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

Transformations

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

By **gonzalocasas**

cheatography.com/gonzalocasas/

Published 17th November, 2019.
Last updated 17th November, 2019.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Transformations (cont)

`t.inve rse()`	Calculate inverse of `t` transformation.
`pt.tra nsf orm(t)`	In-place transform point `pt` with `t` transformation.
`pt.tra nsf orm(t * t2 * t 3)`	In-place transform point `pt` with `t`, `t2` and `t3` transformations.
`pt2 = pt.tra nsf orm ed(t )`	Return a transformed copy of point `pt` with `t` transformation.

`Transf orm ation` represents a 4x4 transformation matrix. `Tran sl ation`, `Scale`, `Reflection`, `Shear` and `Projection` are specialized sub-classes of it.

```
from compas.ge  ometry import Transf orm ation, Tra nsl ation, Rotation # , ...
```

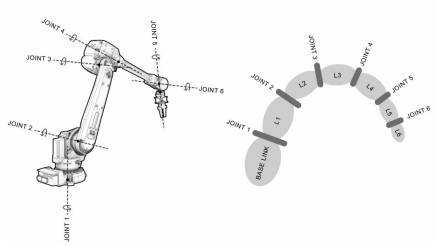
Geometric primitives

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

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

```
from compas.ge ometry 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 RobotM odel, Link, Joint
```

Joint types and units

`Joint.P RI SMATIC`	Meters
`Joint.R EV OLUTE`	Radians
`Joint.C ON TINUOUS`	Radians
`Joint.F IXED`	-

Configuration examples

```
Configuration([.5, pi], [Joint.PRISMATIC, Joint.REVOLUTE])
Config ura tio n.f rom _re vol ute _va lue s([0, 0, pi, 0, 0, 0])
Config ura tio n.f rom _pr ism ati c_a nd_ rev - olu te_ val ues ([8.3], [0.0] * 6)
```

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

```
from compas _fa b.r obots import Config uration
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

ROS backend

ROS
Open Source Robotics Foundation

Start ROS backend

Download or create a `docker -co mpo se.yml` file (examples) and start up with:

```
docker -co mpose up -d
```

Load robot from ROS

```
from compas_fab.backends import RosClient
with RosCli ent ('l oca lhost') as client:
        robot = client.lo ad_ robot()
```

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

Robotic planning with MoveIt!

MoveIt!

Forward Kinematics (FK)

Input	`Config uration`.
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.f or war d_k ine mat ics (co -
nfig)
```

Inverse Kinematics (IK)

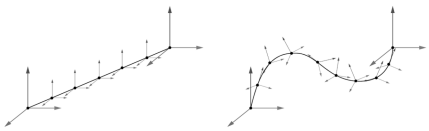
Input	`Frame` in `wcf` and start `Config uration`.
Output	`Config uration`

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.i ni t_c onf igu rat ion()
config = robot.i nv ers e_k ine mat ics (fr ame -
_wcf, start_ config)
```

Path planning



Cartesian vs free-space planning

Plan cartesian path

Input	List of `Frame` in `wcf` and start `Config uration`
Output	`JointT raj ectory`

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. `trajec tor y.f raction = 0.5` means 50% of trajectory completed.

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.i ni t_c onf igu rat ion()
trajectory = robot.p la n_c art esi an_ mot ion -
([f1, f2], start_ config)
```

Plan free-space motion

Input	List of goal `Constraint` and start `Config uration`.
Output	`JointT raj ectory`.

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

```
from compas _fa b.r obots import JointT raj ectory,
 Constr aint, JointC ons traint, Positi onC ons tra
int, Orient ati onC ons traint
```

Code example

```
pc = robot.constraints_from_frame(f1)
start_ config = robot.i ni t_c onf igu rat ion()
trajectory = robot.p la n_m oti on(pc, start_ -
config, planne r_i d=' RRT')
```

Planning scene operations

```
scene = PlanningScene(robot)
scene.a dd _co lli sio n_m esh (Co lli sio nMe -
sh( mesh, 'floor'))
scene.r em ove _co lli sio n_m esh ('f loor')
scene.a pp end _co lli sio n_m esh (Co lli sio -
nMe sh( 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 _fa b.r obots import Planni ngS cene, C
ollis ionMesh
```

Attach meshes to robot

```
scene = PlanningScene(robot)
gripper = Collis ion Mes h(mesh, 'gripper')
scene.a tt ach _co lli sio n_m esh _to _ro bot -
_en d_e ffe cto r(g ripper)
scene.r em ove _at tac hed _co lli sio n_m esh -
('g rip per')
beam = Collis ion Mes h(mesh, 'beam')
acm = Attach edC oll isi onM esh (beam, 'ee_link',
['ee_l ink'])
scene.a dd _at tac hed _co lli sio n_m esh (acm)
```

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

```
from compas _fa b.r obots import Planni ngS cene,
Collis ion Mesh, Attach edC oll isi onMesh
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com