About G-code, position and orientation in robotics. Application on an Universal Robot (UR3e)

Jean-Louis Boimond
University of Angers

Recall:

- the *Tool Center Point* corresponds to the frame attached to the 'central' point of the tool (linked with the robot arm). For example, if the tool is a pencil, its tip is often considered to be the *Tool Center Point* (TCP) in the sense that it is the interested part of the pencil to control;
- a *point* (such as the TCP) in robotics can be described by a frame (a frame being defined by the *position* (in R^3) of its origin and the *orientation* of its three axes).

A (3×3) dimensional rotation matrix is often used in robotics to define the TCP orientation. This matrix, noted R, is the product of 3 rotation matrices about the axes X, Y or Z of the base frame of the robot. Let us recall the expressions of theses rotation matrices by an angle θ:

$$R(X,\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}, R(Y,\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix},$$

$$R(Z,\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The choice of axes considered for these 3 matrices depends on the convention chosen by the robot manufacturer. Thereafter, we consider Universal Robots (UR) e-Series robots, specifically the UR3e robot, where the convention is ZYX (in this order -and not another- because the product of matrices is not commutative!!). Also concerning these robots, we will have: $R = R(Z,\alpha) \times R(Y,\beta) \times R(X,\gamma)$ where α, β, γ are 3 angular values corresponding to rotations about axes Z, Y and X respectively.

The G-code language was originally developed to program numerically-controlled machine tools by producing (in particular) motion instructions to the controller to enable the tool (attached to the machine tool) to realize lines, arcs, splines, etc. Today, this language is used for other types of applications such as 3D printing and robotics, see https://en.wikipedia.org/wiki/G-code and https://docs.duet3d.com/User_manual/Reference/Gcodes for details.

For example, let us consider the following G-code script:

```
G90 G21
G0 X10 Y20 Z30 A15 B25 C35
```

The first line (`G90 G21`) indicates that:

- the coordinates given in the following are absolute (not incremental) (due to instruction `G90`),
- the units are defined in mm (not in inches) (due to instruction `G21`).

The second line (`G0 X10 Y20 Z30 A15 B25 C35`) is a movement instruction. The `G0` instruction performs a movement generated in the joint space which allows a fast movement (the use of a `G1` instruction, rather than `G0`, would perform a straight-line movement (generated in the task space)). After the execution of this movement instruction:

- the TCP will be *positioned* to the values $10, 20, 30$ (mm) relative to the frame X, Y, Z associated with the *Part Coordinate System* (the *Part Coordinate System* corresponds to a frame linked to a

part (or assembly), it is defined in line L8 of the script (to put in a Polyscope program) described in the following),

- the TCP will be *oriented* through the values $15, 25, 35$ (degrees) attached to parameters A, B, C which corresponds to the rotation matrix $R = R(Z, 35) \times R(Y, 25) \times R(X, 15)$ (due to the ZYX convention adopted for UR e-Series robots).

In the case of an UR robot, the use of a G-code file in a Polyscope program is done *via* a 'script' written in a language close to Python (see https://www.universal-robots.com/blog/simplify-robot-programming-with-g-code/ and Example 2 (adapted to a robot with 5 degree of liberty) given in https://www.universal-robots.com/articles/ur/programming/remote-tcp-toolpath-urcap-urscript-functions-and-examples/).

An example of such a script (for UR3e robot) is given below:

```
(L1)          global Waypoint_1_p=p[0.2,0.1,0.2,0.0,0.0,0.0]
(L2)          global Waypoint_1_q=[1.09,-1.33,-2.57,-0.81,1.57,-0.48]
(L3)          set_tcp(p[0.0,0.0,0.04,3.14159,0.0,0.0])
(L4)          while (True):
(L5)            movej(get_inverse_kin(Waypoint_1_p,qnear=Waypoint_1_q),a=1.5,v=1.0)
(L6)            sleep(3.0)
(L7)            mc_initialize(0,p[0.0,0.0,0.04,3.14159,0.0,0.0],6)
(L8)            mc_set_pcs(p[0.3,0.1,0.0,0.0,0.0,0.0])
(L9)            id_pa=mc_load_path("/programs/File_Gcode.nc",use_feedrate=False)
(L10)           id_1_1=mc_add_path(id_pa,1.0,0.05,0.0)
(L11)           mc_run_motion(id_1_1)
(L12)         end
```

where:

- Line `L7` initializes a motion sequence by using the `mc_initialize` function with the following input parameters:
  - the value 0 in the first parameter to indicate that the TCP is attached to the (free) end of the robot arm,
  - in the second parameter, the point defined by the coordinates X, Y, Z, RX, RY, RZ (in the operational space) corresponding to the *situation* of the TCP with respect to the frame associated with the robot flange,
  - the value 6 in the third parameter to indicate that the points described in the following are defined in R^6, enabling the TCP to be *situated* (that is, *positioned* and *oriented*);
- Line `L8` *situates* the *Part Coordinate System* (PCS), the latter represents a *reference frame* defined as follows:

  $X = 0.3, Y = 0.1, Z = 0$ (in m), $R_x = 0, R_y = 0, R_z = 0$ (in rd),

  with respect to the robot base frame (R_0), for the points defined in the G-code file.

The other lines of this script are described at pages 11, 12 of the following additional pdf document.