urcadpts_documentation

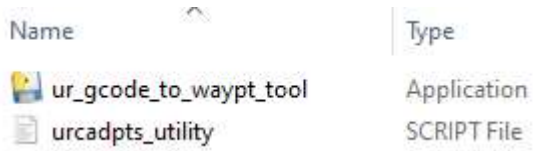
Contents

Description

UR's Toolpath Generator for SolidWorks is effective at creating a path for the robot to continuously follow. However, if the robot should instead perform an action at each point, extra work is needed to post-process the gcode file into waypoints. This project automates the extra work.

Setup

Download the files. They can be saved anywhere, as long as they are unzipped. The Application and Script file must be kept in the same folder.

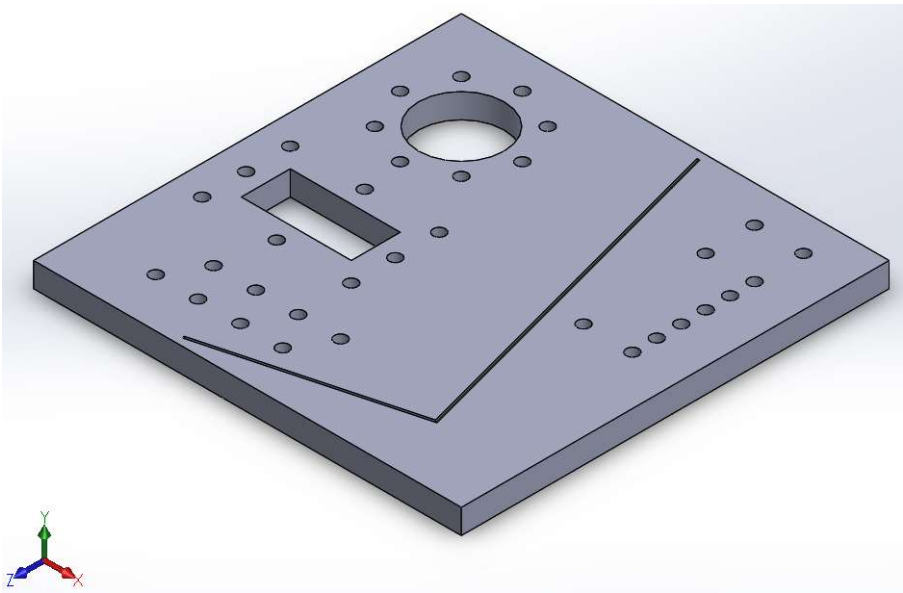


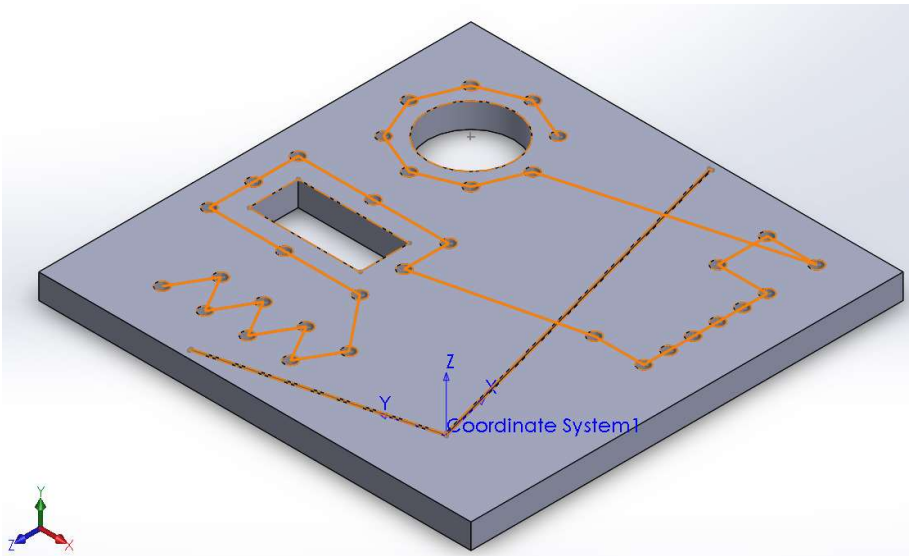
Name	Type
ur_gcode_to_waypt_tool	Application
urcadpts_utility	SCRIPT File

Get a .nc File from SolidWorks using UR's Toolpath Generator

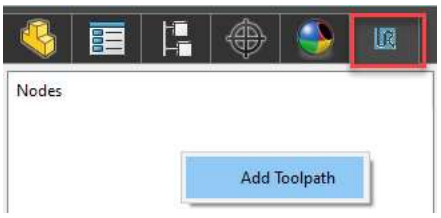
https://www.universal-robots.com/articles/ur/application-installation/universal-robots-toolpath-generator-for-solidworks/

Start with a SolidWorks part or assembly file:

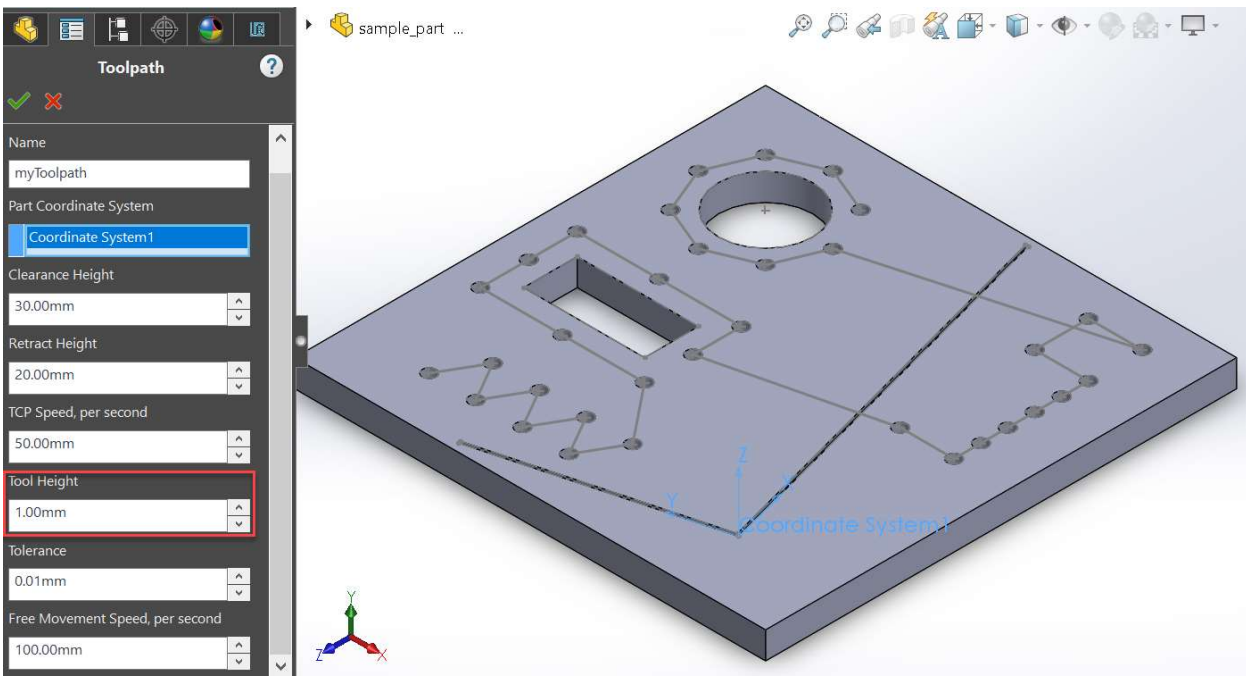




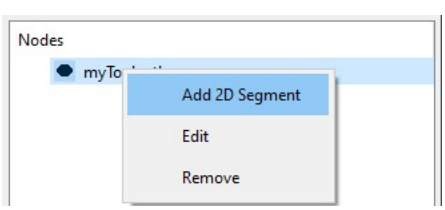
Create a sketch with a line segment connecting all the desired points (note: construction geometry is not converted into gcode), and a Coordinate System for the points to be referenced to.



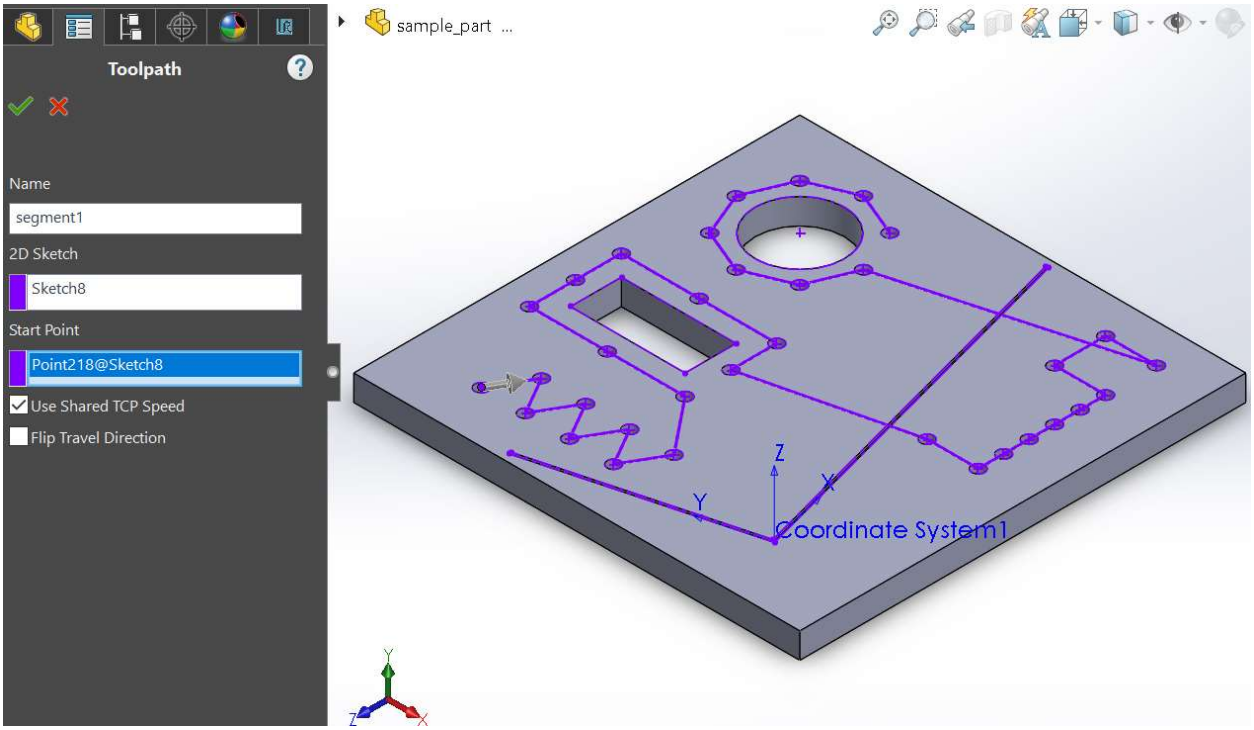
Navigate to the highlighted header of the part tree, right-click on the white space, and click "Add Toolpath"



Name the toolpath and select the reference coordinate system. Set the Tool Height to 0mm if no offset from the target points is desired. Clearance and Retract Height will be removed by default (named lead in / lead out in the converter application).

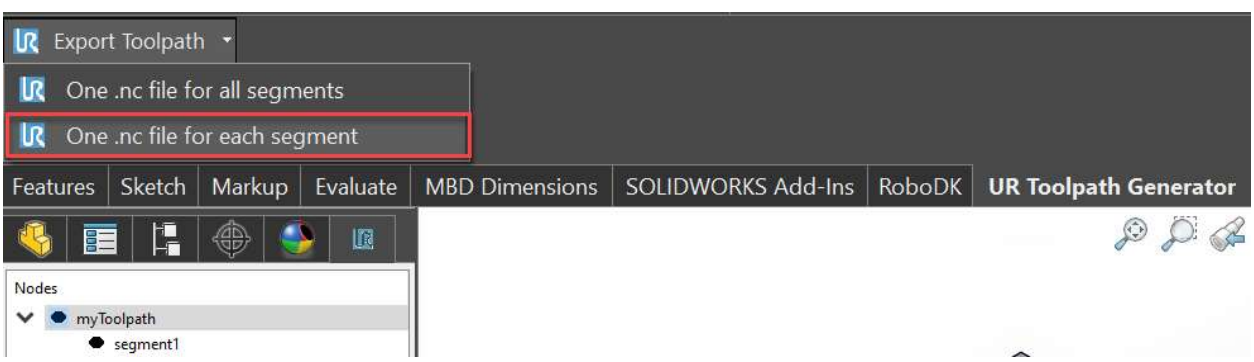


Right click on the toolpath, and click Add 2D Segment



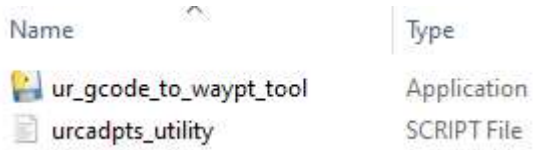
Name the segment, and select the sketch and start point

Selecting "One .nc file for each segment" produces files named *[toolpathname]_[segmentname].nc*. Selecting "One .nc file for all segments" produces one file named *[toolpathname].nc*.

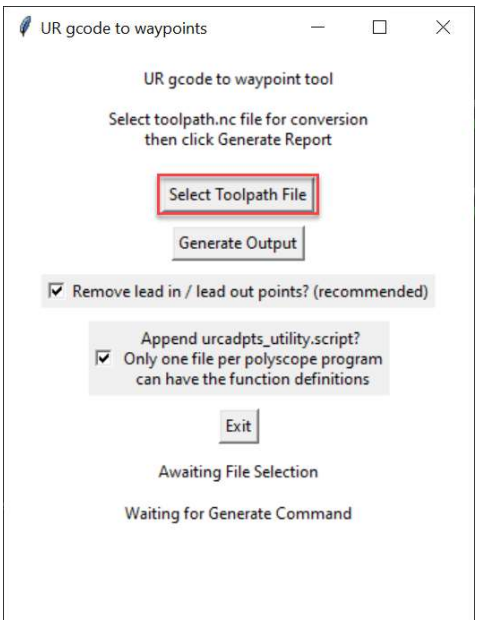


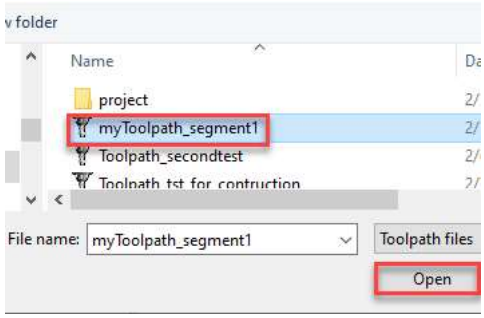
Name	Type
myToolpath_segment1	CNC/Tool path

Convert Toolpath File

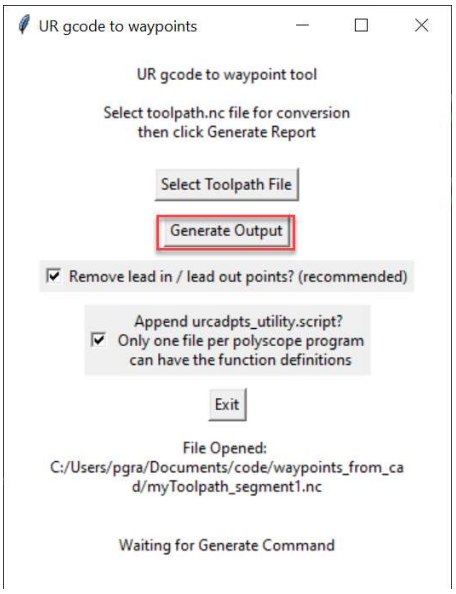


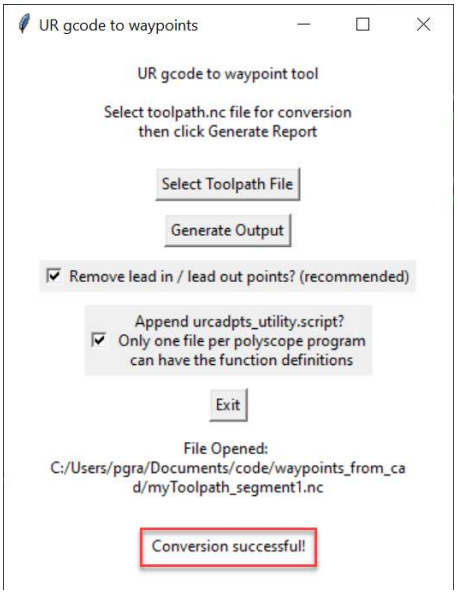
The application and the urcadpts_utility.script file must be in the same directory. Launch the application.





Select Toolpath File and click Open.





Click Generate Output, and the *[toolpathfilename]_points_functions.script* file will be generated.

Name	Type
myToolpath_segment1	CNC/Tool path
myToolpath_segment1_points_functions	SCRIPT File

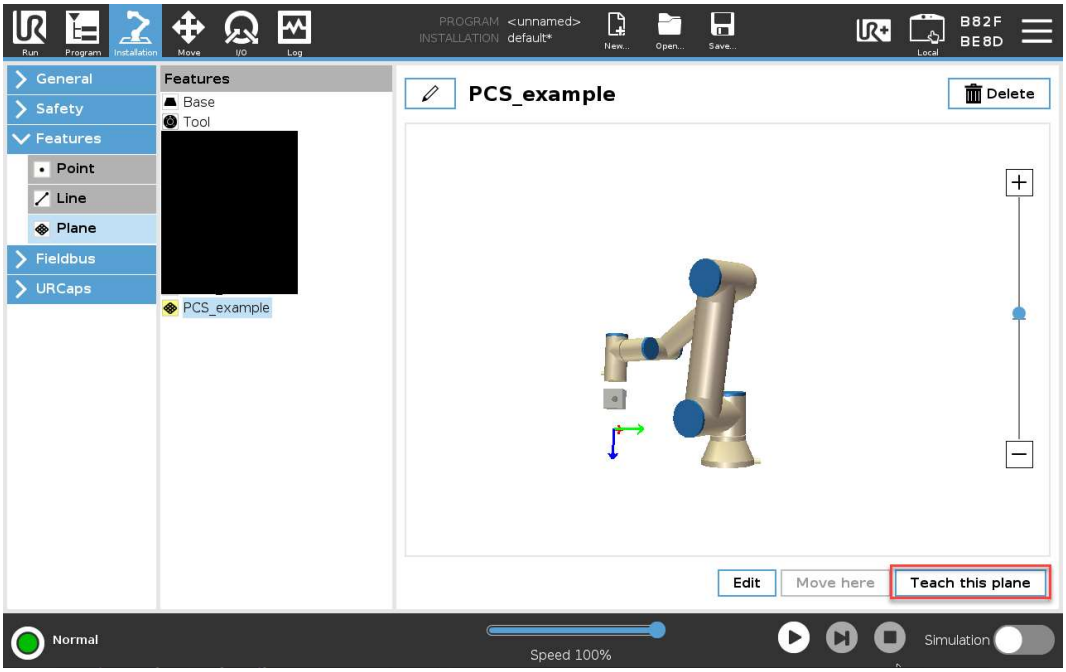
Here is the example original gcode file (right) and the reformatted data (left)

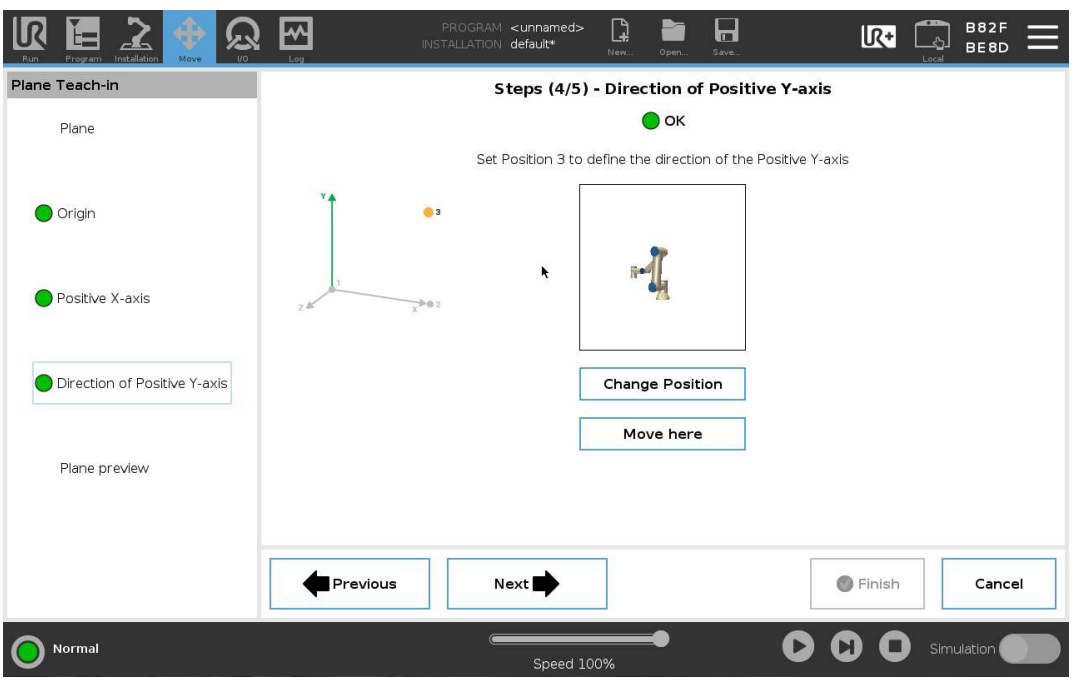
```
urcadpts_data = [

[23.64, 107.546, 1.0],
[35.888, 95.298, 1.0],
[19.158, 90.816, 1.0],
[31.405, 78.568, 1.0],
[14.675, 74.085, 1.0],
[26.922, 61.838, 1.0],
[10.192, 57.355, 1.0],
[22.439, 45.108, 1.0],
[48.251, 56.435, 1.0],
[56.144, 85.896, 1.0],
[64.038, 115.356, 1.0],
[81.425, 110.698, 1.0],
[98.812, 106.039, 1.0],
[90.918, 76.578, 1.0],
[83.024, 47.117, 1.0],
[65.637, 51.776, 1.0],
[64.079, -11.994, 1.0],
[58.903, -31.312, 1.0],
[68.562, -33.9, 1.0],
[78.221, -36.489, 1.0],
[87.881, -39.077, 1.0],
[97.54, -41.665, 1.0],
[107.199, -44.253, 1.0],
[112.376, -24.935, 1.0],
[131.694, -30.111, 1.0],
[126.518, -49.429, 1.0],
[125.663, 43.974, 1.0],
[110.483, 55.622, 1.0],
[107.986, 74.593, 1.0],
[119.634, 89.773, 1.0],
[138.604, 92.271, 1.0],
[153.784, 80.622, 1.0],
[156.282, 61.652, 1.0],
[144.634, 46.472, 1.0]

]

##########################################
# below is the standard functions to inte
##########################################

# function to get the next point in the l
def urcadpts_get_next_point():
    local xyz_i=urcadpts_data.get_row(urc
    local my_waypt=p[xyz_i[0]/1000,xyz_i[
    my_waypt=pose_trans(urcadpts_pcs,my_w
    my_waypt[3]=urcadpts_orientation[3]
    my_waypt[4]=urcadpts_orientation[4]
```

```
(G - code output for Universal Robots' Remote TCP & Toolpa
(UR Toolpath Generator Version : 1.0)
%PM0
N10 G21 G90
N30 G01 X23.640466 Y107.545936 Z30.000000 F6000.000000
N40 G01 X23.640466 Y107.545936 Z1.000000 F6000.000000
N50 G01 X23.640466 Y107.545936 Z1.000000 F3000.000000
N60 G01 X35.887915 Y95.298487 Z1.000000 F3000.000000
N70 G01 X35.887915 Y95.298487 Z1.000000 F3000.000000
N80 G01 X19.157588 Y90.815610 Z1.000000 F3000.000000
N90 G01 X19.157588 Y90.815610 Z1.000000 F3000.000000
N100 G01 X31.405037 Y78.568161 Z1.000000 F3000.000000
N110 G01 X31.405037 Y78.568161 Z1.000000 F3000.000000
N120 G01 X14.674711 Y74.085284 Z1.000000 F3000.000000
N130 G01 X14.674711 Y74.085284 Z1.000000 F3000.000000
N140 G01 X26.922160 Y61.837835 Z1.000000 F3000.000000
N150 G01 X26.922160 Y61.837835 Z1.000000 F3000.000000
N160 G01 X10.191834 Y57.354958 Z1.000000 F3000.000000
N170 G01 X10.191834 Y57.354958 Z1.000000 F3000.000000
N180 G01 X22.439282 Y45.107509 Z1.000000 F3000.000000
N190 G01 X22.439282 Y45.107509 Z1.000000 F3000.000000
N200 G01 X48.250502 Y56.434821 Z1.000000 F3000.000000
N210 G01 X48.250502 Y56.434821 Z1.000000 F3000.000000
N220 G01 X56.144483 Y85.895559 Z1.000000 F3000.000000
N230 G01 X56.144483 Y85.895559 Z1.000000 F3000.000000
N240 G01 X64.038464 Y115.356296 Z1.000000 F3000.000000
N250 G01 X64.038464 Y115.356296 Z1.000000 F3000.000000
N260 G01 X81.425129 Y110.697554 Z1.000000 F3000.000000
N270 G01 X81.425129 Y110.697554 Z1.000000 F3000.000000
N280 G01 X98.811794 Y106.038811 Z1.000000 F3000.000000
N290 G01 X98.811794 Y106.038811 Z1.000000 F3000.000000
N300 G01 X90.917813 Y76.578073 Z1.000000 F3000.000000
N310 G01 X90.917813 Y76.578073 Z1.000000 F3000.000000
N320 G01 X83.023832 Y47.117335 Z1.000000 F3000.000000
N330 G01 X83.023832 Y47.117335 Z1.000000 F3000.000000
N340 G01 X65.637167 Y51.776078 Z1.000000 F3000.000000
N350 G01 X65.637167 Y51.776078 Z1.000000 F3000.000000
N360 G01 X64.079274 Y-11.993609 Z1.000000 F3000.000000
N370 G01 X64.079274 Y-11.993609 Z1.000000 F3000.000000
N380 G01 X58.902893 Y-31.312125 Z1.000000 F3000.000000
N390 G01 X58.902893 Y-31.312125 Z1.000000 F3000.000000
N400 G01 X68.562151 Y-33.900316 Z1.000000 F3000.000000
N410 G01 X68.562151 Y-33.900316 Z1.000000 F3000.000000
N420 G01 X78.221410 Y-36.488506 Z1.000000 F3000.000000
N430 G01 X78.221410 Y-36.488506 Z1.000000 F3000.000000
N440 G01 X87.880668 Y-39.076697 Z1.000000 F3000.000000
N450 G01 X87.880668 Y-39.076697 Z1.000000 F3000.000000
N460 G01 X97.539926 Y-41.664887 Z1.000000 F3000.000000
N470 G01 X97.539926 Y-41.664887 Z1.000000 F3000.000000
N480 G01 X107.199184 Y-44.253078 Z1.000000 F3000.000000
```

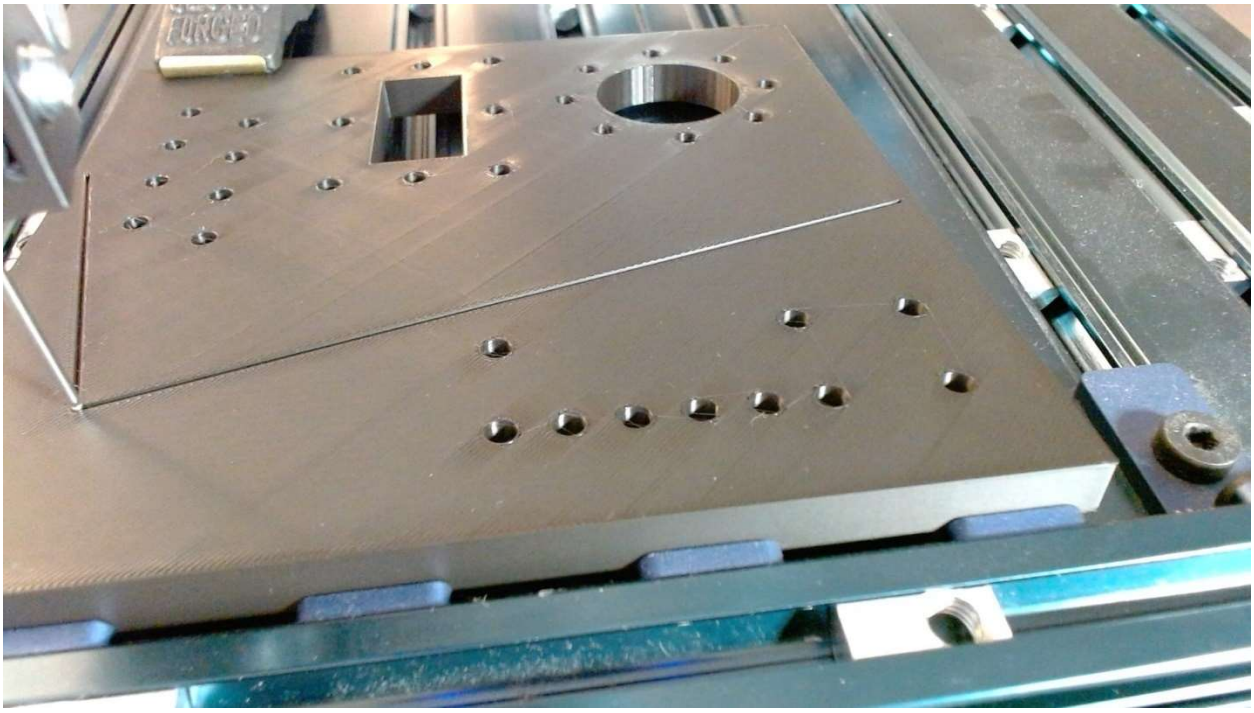
On the Robot

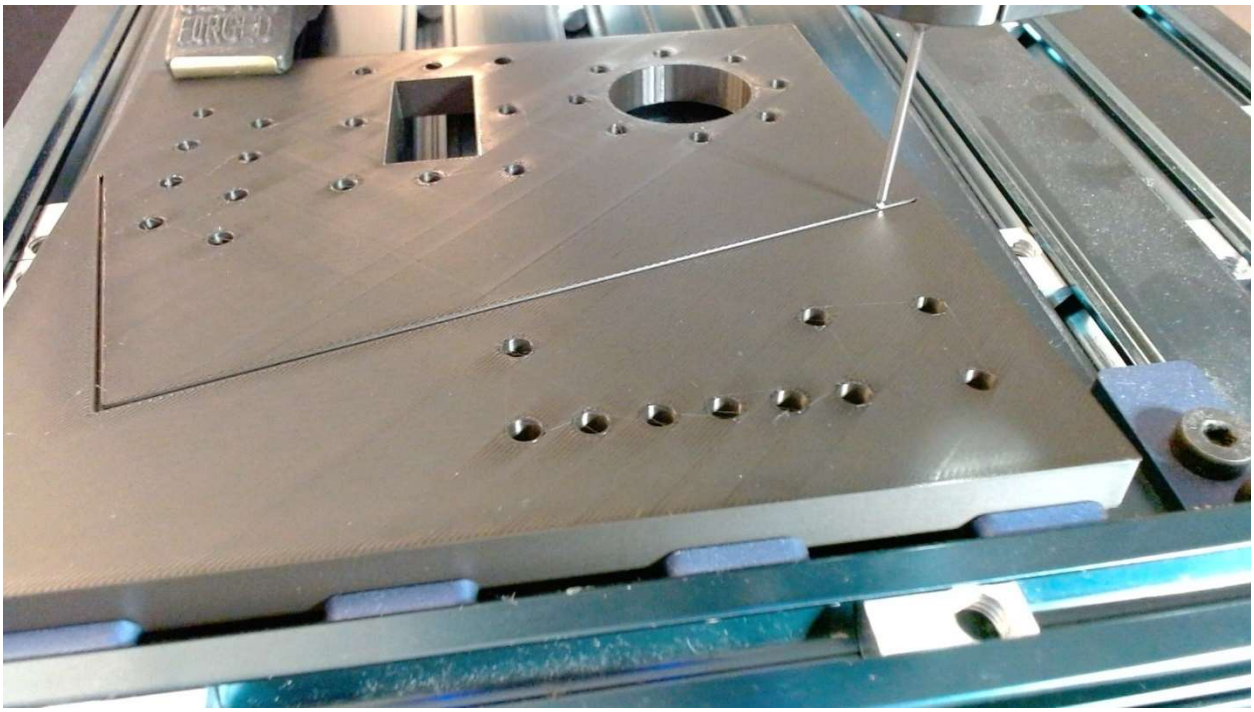
Create an Installation Feature to match the SolidWorks Coordinate System.

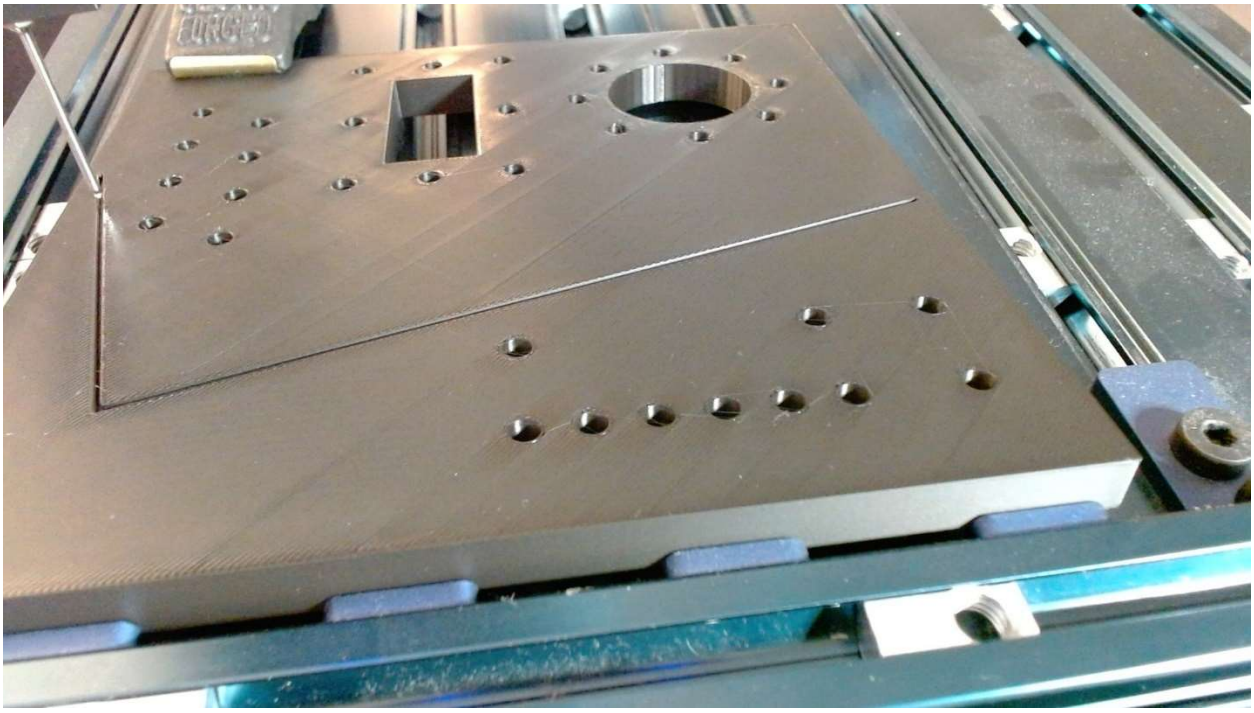




Origin, X, and Y locations on the real part for the example:

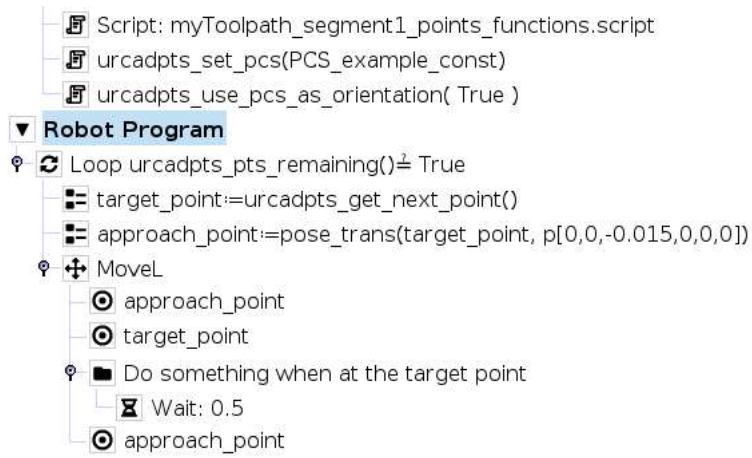


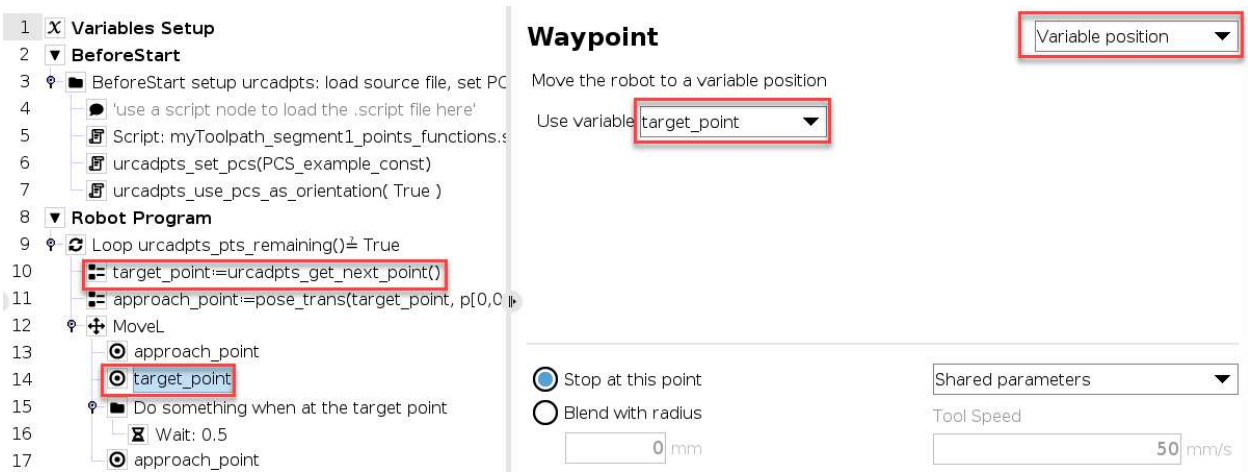




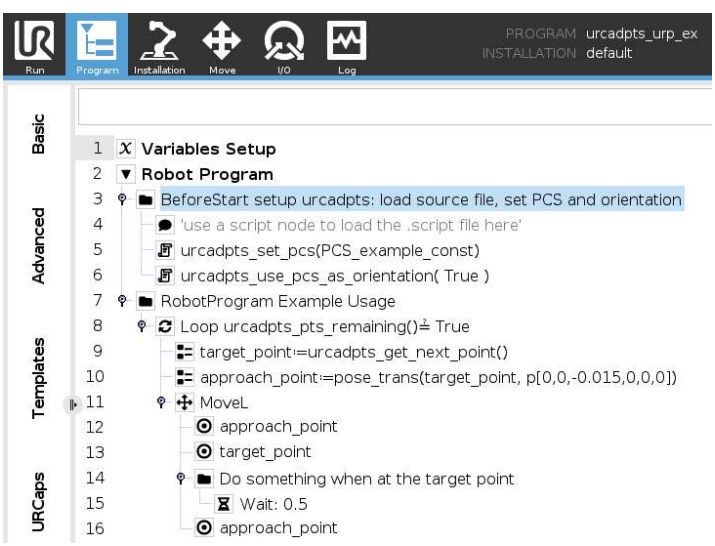
Use the .script file by:

1. In the Before Start Sequence:
    a. Load the *[filename]_points_functions.script* as a script file
    b. Set the Part Coordinate System (PCS) as the previous installation feature
    c. Set the tool orientation to be used at the points
2. In the robot program:
    a. Assign a variable value as *urcadpts_get_next_point()*
    b. (optional) make approach points based on the target point
    c. Move to the variable waypoint (optionally going to an approach point) and perform an action at the target waypoint.



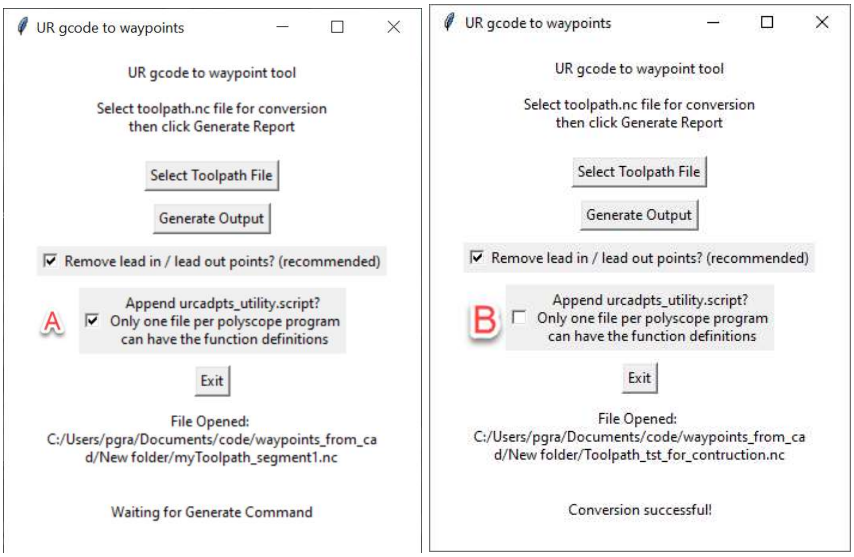


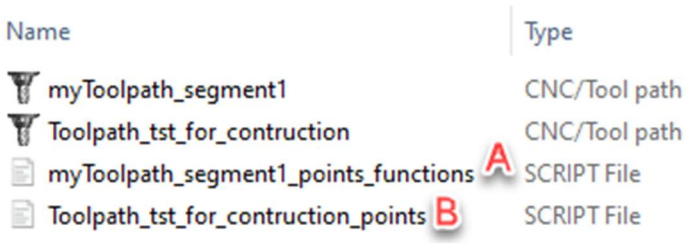
The provided sample .urp file can be loaded in as a sub-program. If desired, the folders can be cut/pasted into appropriate program locations, and the subprogram dissolved.



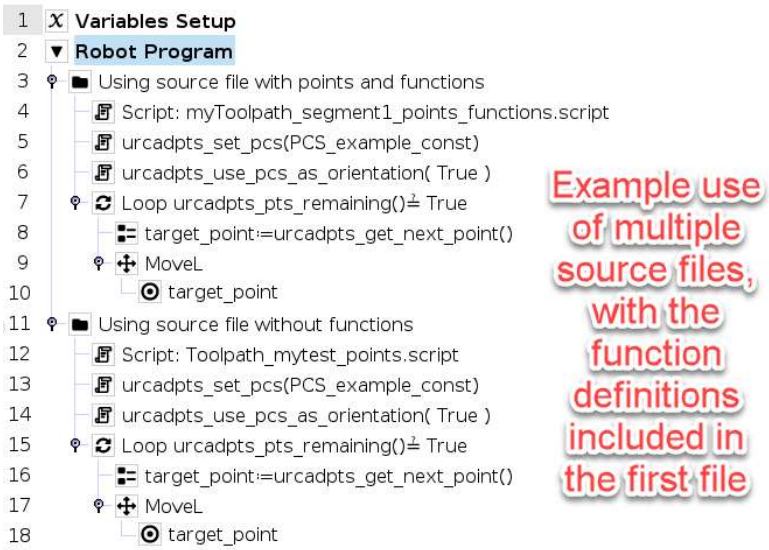
Using Multiple Toolpath Files in One PolyScope Program

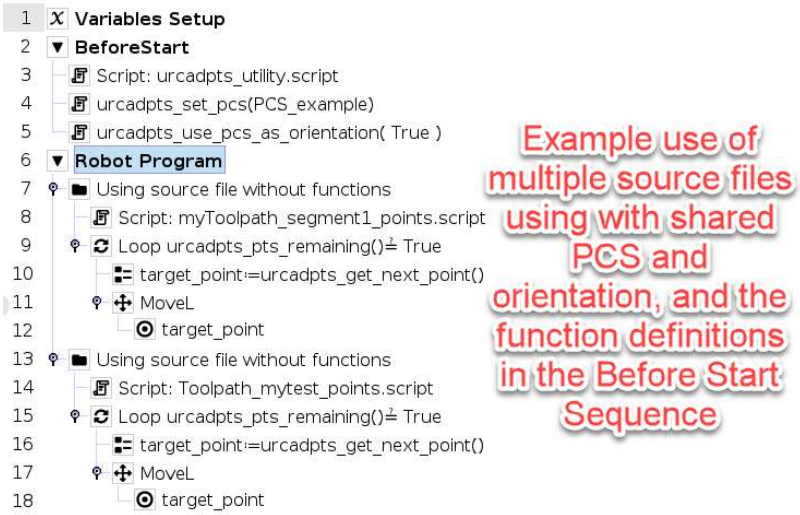
In order to use multiple toolpath source files in one PolyScope program, you must be diligent to redefine urscript functions. The urcadpts_utility.script file can only be added to one file per polyscope program. The checkbox in the below illustrations controls whether or not the script file with the function definitions will be added to the end of the converted file. The resultant file will indicate via its file name if the functions have been added. *[filename]_points_functions.script* has the functions, whereas *[filename]_points.script* does not.





If the checkbox is not selected for any of the files used in a PolyScope program, then the function definitions will need to be added in separately. This can be done by directly adding the *urcadpts_utility.script* function to the Before Start sequence of the PolyScope Program.





Note that if the function definitions are included in one of several source files used in a PolyScope program, the *[filename]_points_functions.script* must occur before any of the functions in *urcadpts_utility.script* are called.

URScript Function Documentation

The global variables created are:

- **urcadpts_i**
  - **Integer**. Loop variable that increments through the list of points
- **urcadpts_len**
  - **Integer**. Number of points in the data, indexed at 0
- **urcadpts_pcs**
  - **Waypoint**. Part Coordinate System that the points are relative to
- **urcadpts_orientation**
  - **Waypoint**. The rx/ry/rz values are applied to the generated waypoints
- **urcadpts_x_list**
  - **List**. Variable length list of floats with a capacity of 5000. X coordinates in mm for each provided point relative to the PCS.
- **urcadpts_y_list**
  - **List**. Variable length list of floats with a capacity of 5000. Y coordinates in mm for each provided point relative to the PCS.
- **urcadpts_z_list**
  - **List**. Variable length list of floats with a capacity of 5000. Z coordinates in mm for each provided point relative to the PCS.

The provided functions are:

- *urcadpts_get_next_point()*
  - Arguments: none
  - Returns: the next waypoint in the list
  - Use: **Var_1**:=*urcadpts_get_next_point()*
    - Assigns the next waypoint to **Var_1**, primary function for getting data out of this project
  - Also: increments **urcadpts_i**
- *urcadpts_pts_remaining()*
  - Arguments: none
  - Returns: True or False as to if there are more points in the list
  - Use: **Var_1**:=*urcad_pts_remaing()*
    - Assigns a boolean result to **Var_1**, used to check if data is available
- *urcadpts_get_status()*
  - Arguments: none

- Returns: **List** of **Integers**: **[num_of_points, current_index, are_there_more_points]**
  - **num_of_points**: number of points in the data, indexed at 1
  - **current_index**: loop variable, indexed at 1
  - **are_there_more_points**: 1 or 0, coresponding to True or False

- *urcadpts_copy_orientation(**waypt**)*
  - Arguments: **waypoint** variable in the form of p[x,y,z,rx,ry,rz]
  - Returns: none
  - Use: *urcadpts_copy_orientation(**waypt**)*
    - Assigns the values of **waypt** to **urcadpts_orientation**
- *urcadpts_set_pcs(**waypt**)*
  - Arguments: **waypoint** variable in the form of p[x,y,z,rx,ry,rz]
  - Returns: none
  - Use: *urcadpts_set_pcs(**waypt**)*
    - Assigns the values of **waypt** to **urcadpts_pcs**
- *urcadpts_config(**pcs**=p[0,0,0,0,0,0],**orientation**=p[0,0,0,0,0,0])*
  - Arguments: 2 **waypoint** variables in the form of p[x,y,z,rx,ry,rz]
  - Returns: none
  - Use: *urcadpts_config(**waypt1**, **waypt2**)*
    - Calls *urcadpts_set_pcs(**waypt1**)* and *urcadpts_copy_orientation(**waypt2**)*
- *urcadpts_use_pcs_as_orientation(**invert**=False)*
  - Arguments: **Boolean**
  - Return: none
  - Convenience function that sets the orientation of the resultant waypoints to be aligned with the z-axis of the Part Coordinate System. If the optional Boolean argument is True, then the waypoint orientation is set to aligned to and inverted with the z-axis of the PCS.

Misc.

- The *urcadpts_utility.script* file uses struct methods and variable length lists, so Polyscope must be at version 5.16 or higher.
- The file converter only retains 3 significant figures past the decimal point to minimize file size.
- Points that are less than 0.05mm from the previous converted point are ignored.
- When opening a file with greater than 1000 points on Polyscope 5.21, there can be some lag. Size limit is capped at 5000 points.
- UR's Toolpath Generator for SolidWorks exports units in mm, even if the SolidWorks document units are not in mm. Unit conversion should not be necessary.

Related

https://www.universal-robots.com/articles/ur/application-installation/universal-robots-toolpath-generator-for-solidworks/

https://www.universal-robots.com/articles/ur/programming/script-library-tutorial/

https://github.com/phillipgramboUR/urcadpts

Building the application from scratch

The completed application is provided. However, to make edits to the application, its possible to edit the .py source file and recompile the application.

To build the .exe from the .py file:

- Open command prompt
- Install pyinstaller with: *pip install pyinstaller*
- Navigate to the folder location with the .py file
- Build with: *pyinstaller ur_gcode_to_waypt_tool.py --onefile --noconsole*