ENMT482 Robotics

Manipulator Lab Assignment

Tim Hadler - 44663394
Aidan Ogilvie - 53581629

13/10/2020

1.0 Introduction

As technology advances we are seeing more and more jobs being automated by robotic manipulators. Automation of repetitive tasks can reduce production costs and production time, making it an interesting area of research for companies. However, manipulators are not always suitable for automating a task. The feasibility of automating a task with a manipulator depends on the specific task. Key factors to think about when considering replacing a human worker with a manipulator is the potential difference in production time, cost, and quality. This report details an investigation into the feasibility of automating a coffee making process, with a 6-DOF manipulator. The task of making an espresso coffee was automated using the UR5 manipulator, and an analysis done on the benefits/costs of automating this task.

2.0 Method

The project workspace is shown in Figure 1. The workspace included the UR5 manipulator, a set of UR5 barrister tools, a coffee machine, a grinder, cups, and a tamper/scrapper stand. A local frame was provided for each piece of equipment, as well as two points on each local frame in global (UR5 Base Frame) coordinates. Points of interest on each piece of equipment, buttons for example, were given in the equipment's local coordinates.

The RoboDK software was used to control the manipulator. Matlab was used to calculate the homogeneous transformation matrices (HTs) for each piece of equipment, each tool, and each point of interest on the equipment and tools. Once the HTs were found, they were entered into RoboDK to get the correct orientation and rough position of the end-effector. The manipulator was then adjusted in RoboDK as needed to avoid obstacles, have a more optimal configuration, or add offsets. The desired position/configuration was then recorded by copying the UR5 joint angles, which were used to tell the manipulator where to go during testing.

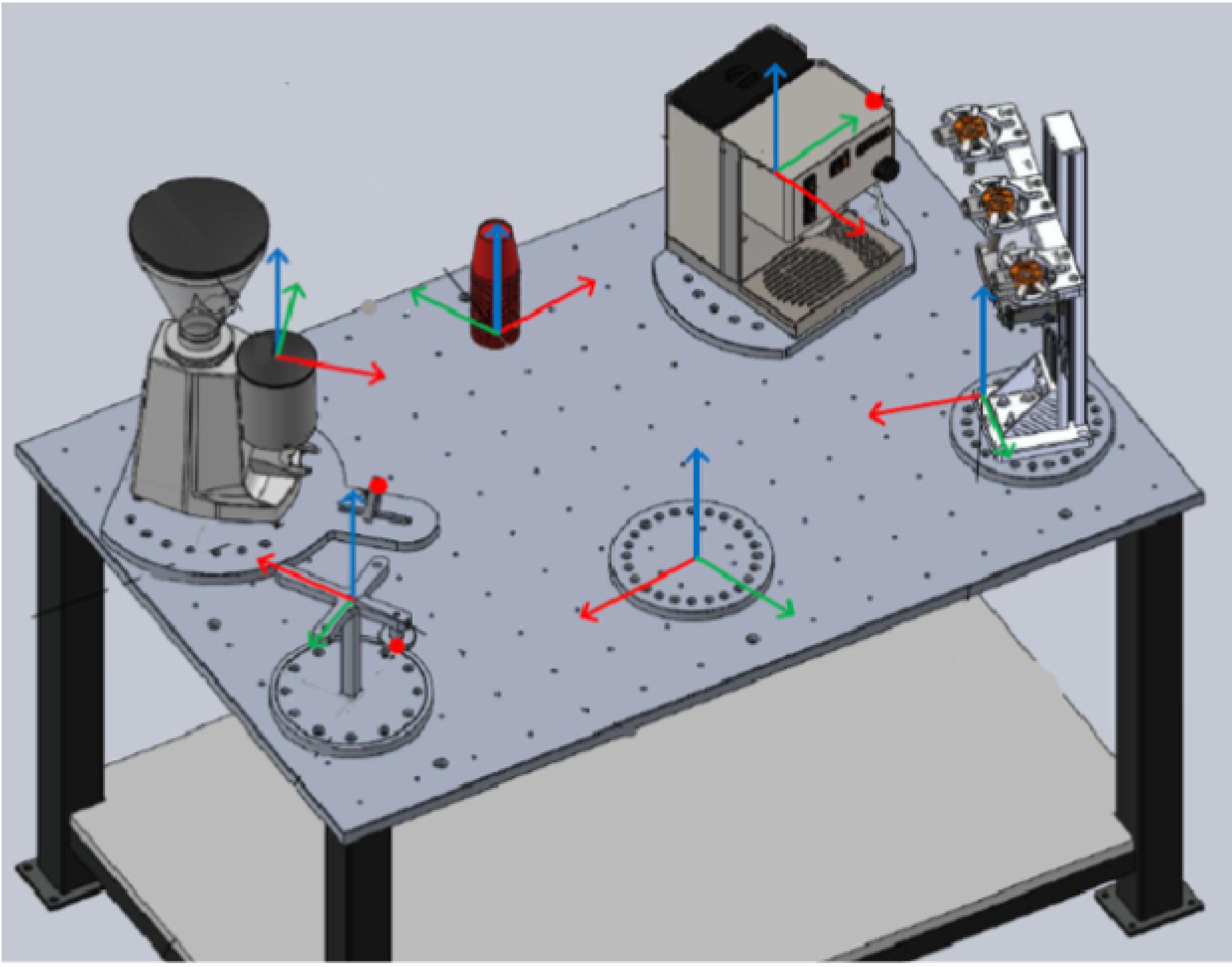


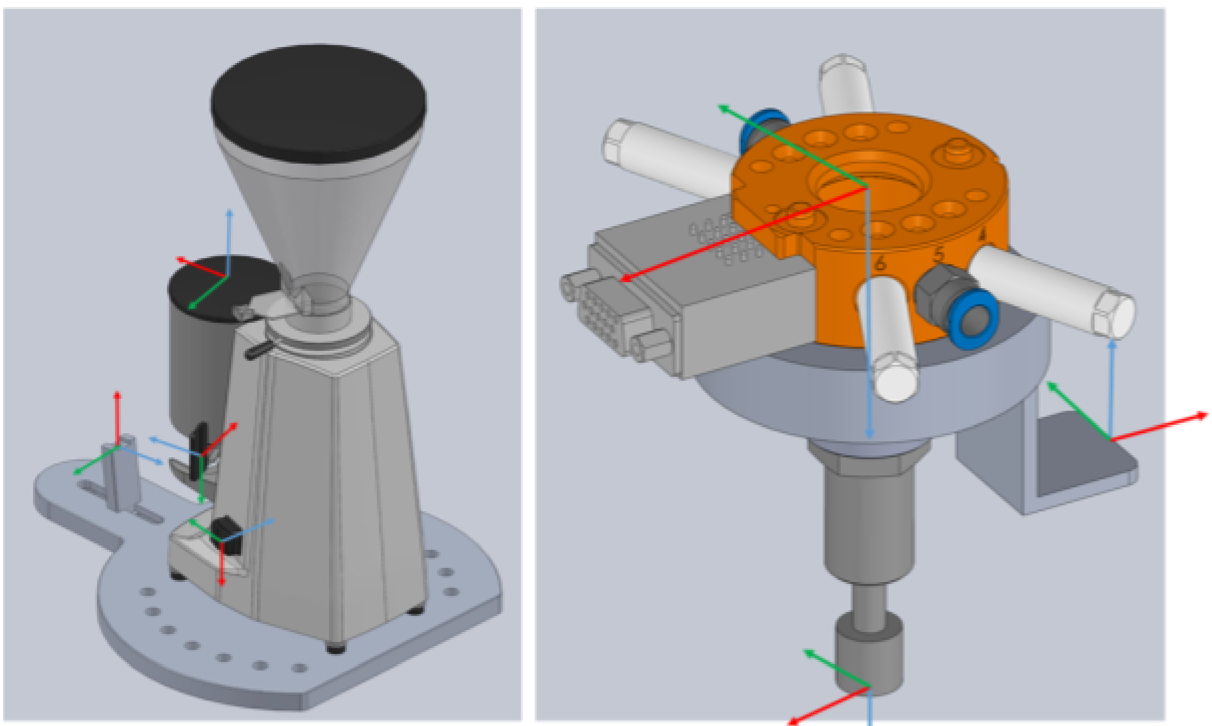
Figure 1: The assignment workspace with equipment local frames - UR5 not shown

2.1 Frame assignment

Local frames for each piece of equipment and tool were given, but to get the UR5 end-effector to line up with points of interest with a certain orientation, frames must also be assigned to the points of interest. The points frame can then be aligned with a tools frame to get the desired orientation. Frames for key pieces of equipment and tools are shown in Figures 2-5 . For the frames for all pieces of equipment, see Appendix A.

As a general rule, the z direction was assigned to the direction of motion of the end-effector, or the direction the tool was pointing. For example, frames assigned to buttons had the z-axis aligned with the direction of pressing the button. Having the grinder tool z-axis as shown in Figure 2, meant that aligning these frames would result in the grinder tool pushing the button with the axial pusher. The x-axis of the grinder lever and the grinder tool lever were aligned such that the tool was on the correct side of the grinder when pulling the lever.

The frames for the rest for the portafilter tool, shown in Figure 3, were assigned so that when the portafilter is placed on the grinder rest, the filter will be level with the world frame. The filter center frame was orientated the same way so that when tamping and scraping the coffee with the tamper stand, the filter will be horizontally level with the world frame.



a) b)

Figure 2: Assigned local frames for the coffee grinder (a) and grinder tool (b)

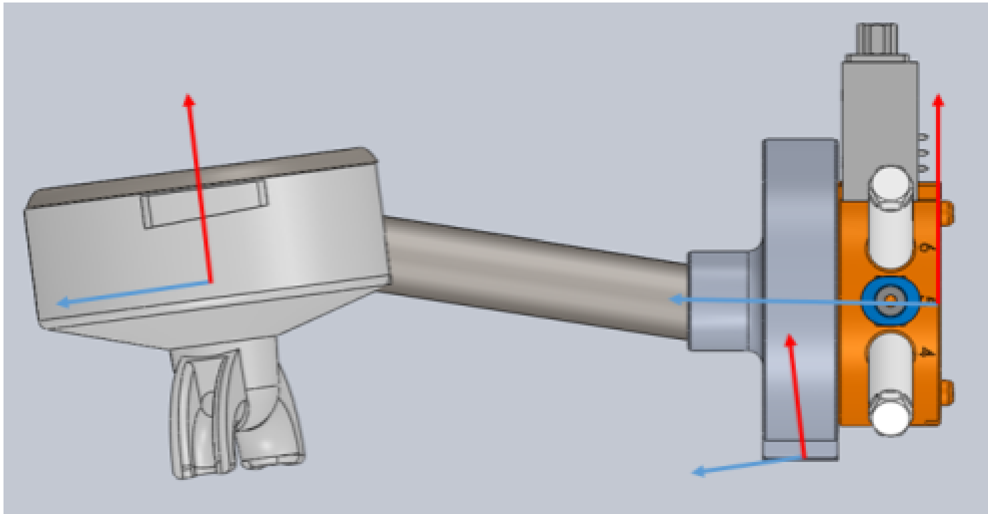
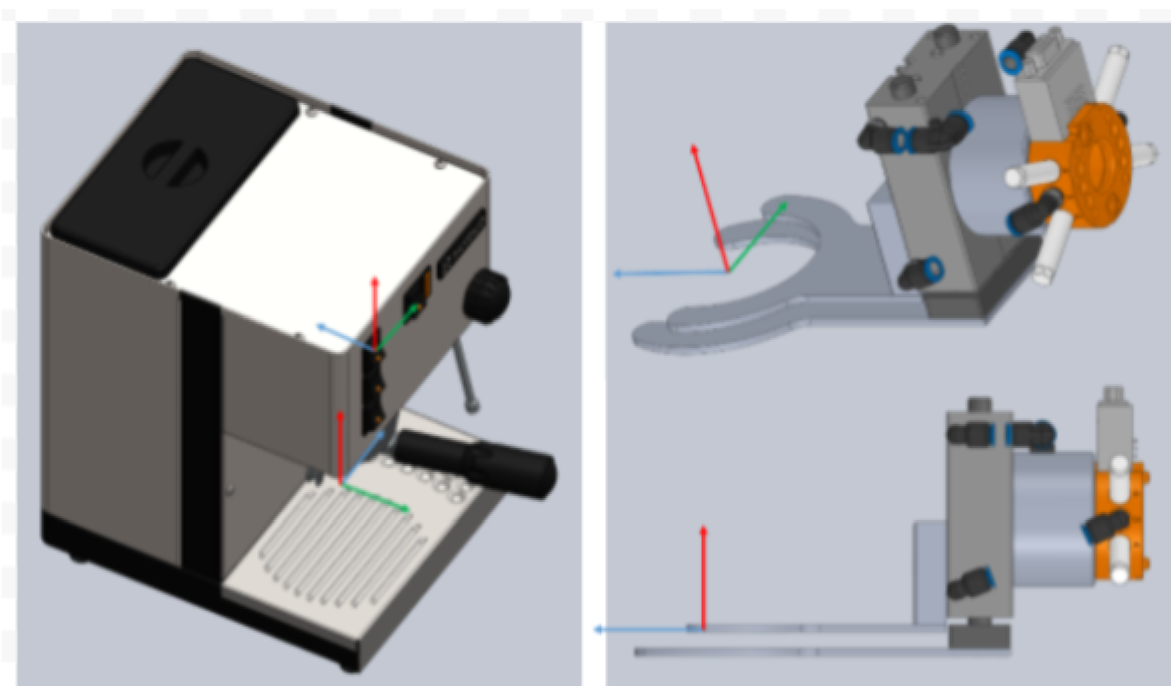


Figure 3: Assigned local frames for the portafilter tool



(a) (b)

Figure 4: Assigned local frames for the coffee machine (a) and cup tool (b)

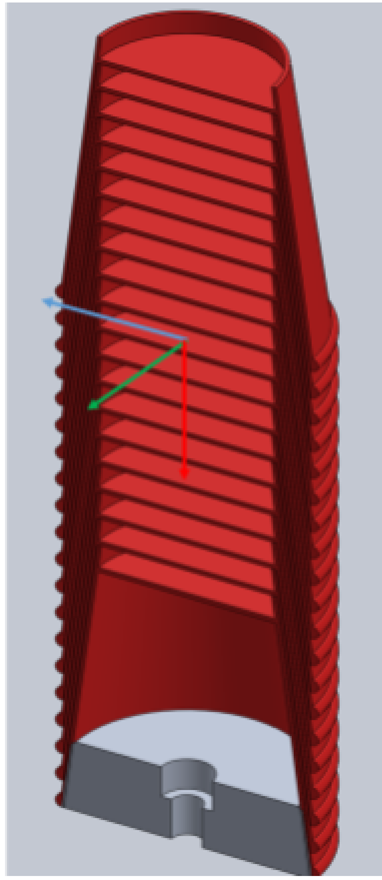


Figure 5: Assigned local frame for the cup stack

2.2 Transformation matrices

To convert each of these frames into the global coordinate system, we used the MATLAB scripts shown in Appendix B. Each transformation required a number of transformation matrices to convert any given frame from the local machine frame to global coordinates, and then to convert that point to the required location of the robot arm to place the tool point at the desired location.

2.3 Path

To make pathing easier, a single intermediate point was placed between all of the machines and tools. This point was accessible via a single move command from all other places of importance. Each time the robot needed to cross from one side of the workstation to the other without being precise it would pass by that point.

Between each step tools had to be swapped. This was done simply by moving the robot to the intermediate position, calling the appropriate function to remove or acquire the correct tool, and then returning to the intermediate position for the next task.

The path for moving the cup from the pile to the coffee machine required six waypoints at minimum. These points were defined with respect to the top cup frame and the machine cup location frame, generally 100mm in the appropriate direction. The four waypoints for lifting the cup form a 100mm square in the x/z plane of the top cup frame, then the cup is rotated before the tool is moved to a point aligned with the z axis of the machine cup location and then carefully moved to its final position. We chose to approach the coffee machine from its side to avoid collisions with the tool stand and the portafilter tool in the machine.

To approach the buttons the robot moves from the intermediate position to a point offset from the button frame a small distance in negative z and about 100mm in negative y with respect to the button frame. It then moves along the y axis until aligned with the button's centre, and to push the button moves simultaneously along positive z and either positive or negative x to turn the switch on or off.

To move the cup on the table the same two points used to place it in the machine are used to remove it. The cup is then carefully moved through two intermediate waypoints to be placed on the table between the cup stack and the coffee machine.

Results

3.1 Time and motion study

A Time and Motion Study is important when considering using a robot for a task. If it cannot work as quickly as a human, it will not produce as much profit as a human and may never be worth the considerable upfront expense.

In our test, the robot successfully produced one simple cup of coffee in 10 minutes and 35 seconds. An experienced human barista is capable of producing one every 58 seconds for $27 per hour. The UR5 robot and its custom tools together cost $4,500 to purchase, followed by $400 every year for maintenance, which takes away 3 hours of work time. The following assumptions are made for the results of this study:

- $4.50 per coffee
- 8 hours of work per day, with total 1 hour break time for human workers
- Constant coffee orders over the course of the day
- Tax and the time-value of money can be ignored

Given all this, Table 1 shows the profit results for one human barista and one UR5. A human worker produces almost nine times the profit of one robot.

Table 1: Profit and Output results for one single human worker compared to a UR5 robot

Worker	Time per coffee
Human	58 seconds
UR5	10:35 minutes

Worker	Initial Costs	Annual costs	Coffees per day	Daily Revenue	Annual Revenue	Annual Profit
Human	-	$53,055	434	$1953.00	$511,686.00	$458,631.00
UR5	$45,000	$400	45	$202.50	$52,978.50	$52,578.50

4.0 Discussion/Conclusion

For robot baristas to be even barely worthwhile would require replacing each human worker with ten robots. This would allow them to produce slightly more coffees per day than a human, and produce 15% more annual profit after wages and maintenance. The purchase cost of the robots would be paid for in 6.7 years.

However, this is only true using unrealistic assumptions. Given the time-value of money it will take longer for the robots to be profitable. Given that coffee orders would not be consistent with significantly more around the middle of the day than closing time, and that human workers could be hired part time if necessary, robots are likely to be even less cost effective. Also, 10 robots take up a lot more space than one human, further increasing costs.

With all that said, it is likely that the UR5 robot could produce coffees faster than our program did. If someone did plan to buy ten robots they would put money toward optimising the efficiency of the UR5's path, increasing the cost efficiency per robot.

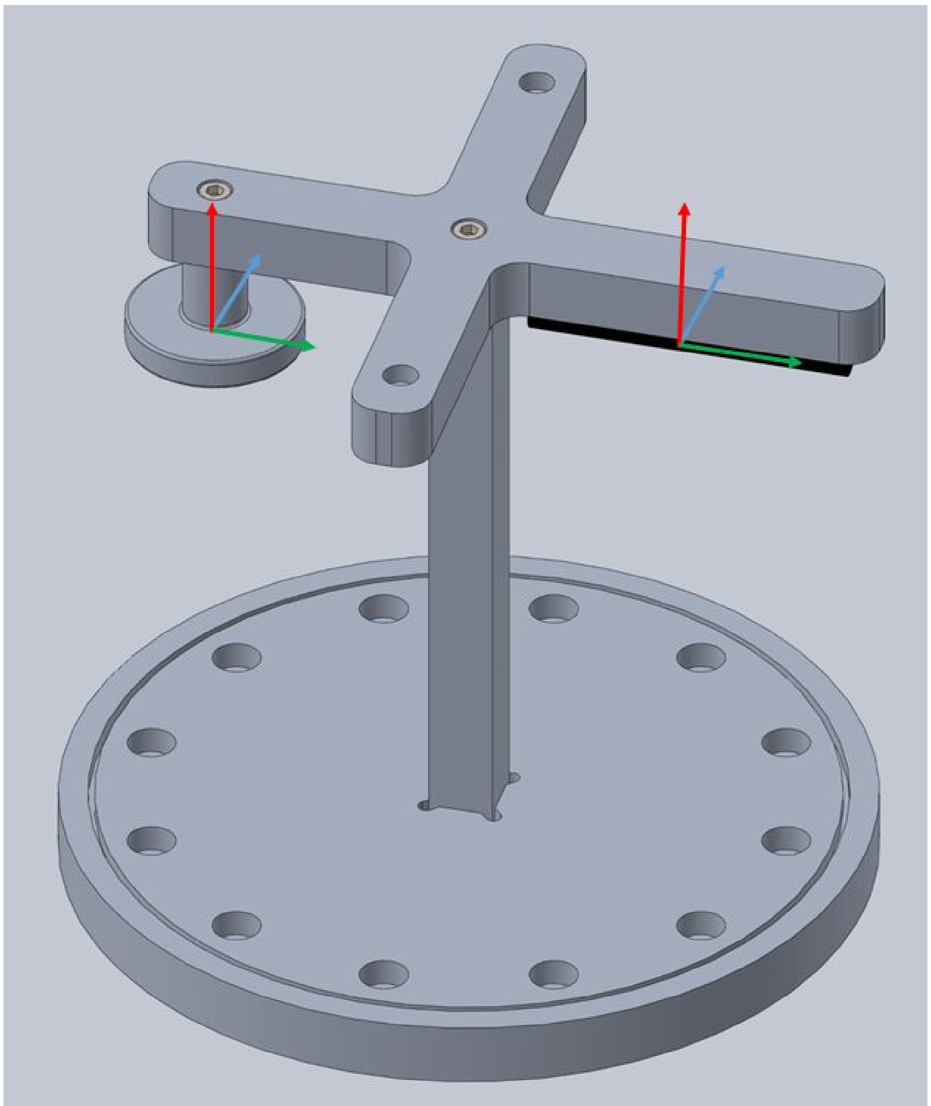
Overall, I do not believe automating the production of coffee using the UR5 robot would be worth it. There is not likely to be any return on investment for 7 years, and the initial cost is too high.

While automation can often save money, the job of a barista is too complex and delicate to be performed effectively by the UR5 robot. Even if the path was further optimised to take the shortest time possible, the robot moves too slowly to be of any use in this case. If a smaller, safer robot capable of faster movements was used alongside automated buttons and levers that did not require the robot arm to press them, then automation might be more viable.

Appendix

Appendix A - Reference Frames

Assigned local frames for the tamper/scraper stand



Appendix B - Transformation Matrices

zRotation function

```
function [T, Rz, deg] = zRotation(global_pt1, global_pt2, local_pt2)
%ZROTATION Returns the rotation matrix relating Frame A to Frame B
%   Requries two points in the global frame that are along the same axis of
%   the local frame
%   global_pt2 must be the same point as local_pt2
%   local_pt2 is used to determine which axis is defined by the two global
%   poiints, and in what direction
%   Assumes there is a roation about the z axis
%   Assumes global_pt1 is the origin of Frame B in global coords


for i = 1:3
    if (local_pt2(i) == 0)
        break
    end
end

% Determines which axis is defined by the two global points
if (i == 1)
    k = 2;
elseif (i == 2)
    k = 1;
end

% Ignore z
if (local_pt2(k) > 0)
    local_ax = global_pt2(1:2) - global_pt1(1:2);
else
    local_ax = global_pt1(1:2) - global_pt2(1:2);
end

% Normalize
local_ax = local_ax ./ norm(local_ax);

if (i == 1)
    dot_p = dot([0;1], local_ax);
elseif (i == 2)
    dot_p = dot([1;0], local_ax);
end

rad = acos(dot_p);
deg = acosd(dot_p);

% If the two global points define an axis in the negative direction,
% flip it to get the positive direction
if (local_pt2(k) < 0)
    rad = 2*pi - rad;
    deg = 360 - deg;
end


Rz = [cos(rad) -sin(rad) 0;sin(rad) cos(rad) 0; 0 0 1];

T = [Rz global_pt1; 0 0 0 1];
```

Machine transformation script

```
% Script to find the equipment transformations and to assign frames to
% the points of interest on the equipment
clc, clear, close all

% Find local frame transformations
% Grinder
grinder_frame_offset = [482.29; -433.74; 314.13];
grinder_rest_global = [370.66; -321.55; 66.86];
grinder_rest_local = [157.61; 0; -250.45];

[Tw_grinder, Rw_grinder, deg_grinder] = zRotation(grinder_frame_offset,↙
grinder_rest_global, grinder_rest_local);

% Tamper stand
tamper_frame_offset = [599.13; 0; 211.07];
tamper_global = [559; 68.77; 156.07];
tamper_local = [-80; 0; -55];

[Tw_tamper, Rw_tamper, deg_tamper] = zRotation(tamper_frame_offset, tamper_global,↙
tamper_local);

% Cups
cups_frame_offset = [1.49; -600.54; -20];
deg_cups = 180;
Rw_cups = rotz(deg_cups);
Tw_cups = [Rw_cups cups_frame_offset; 0 0 0 1];


% Silvia
silvia_frame_offset = [-366.2; -389.8; 341.38];
silvia_pt2_global = [-577.06; -446.46; 341.38];
silvia_pt2_local = [0; 218.0; 0];

[Tw_silvia, Rw_silvia, deg_silvia] = zRotation(silvia_frame_offset, silvia_pt2_global,↙
silvia_pt2_local);


% Find local point transfomrations
% Local points
grinder_rest_local = [157.61; 0; -250.45];
grinder_start_local = [-80.71; 94.26; -227.68];
grinder_stop_local = [-64.42; 89.82; -227.68];
grinder_lever_local = [-35.82; 83.8; -153];

tamper_level_local = [70; 0; -32];
tamper_press_local = [-80; 0; -55];

cup_approach_local = [0; -100; 204];
cup_get_local = [0; 0; 204];

silvia_cup_local = [-10; 55.00; -201.38];
silvia_approach_local = [150; 55.00; -201.38];

silvia_but1_local = [50.67; 98.75; -27.89];
silvia_but2_local = [50.67; 35.75; -27.89];
```

Tool Transformations script

```
% Assigning frames to points of interest on the coffee making tools
% with reference to the UR5 master tool frame
% Calculates and saves the HT for each point on each tool

clc, clear, close all

% All tools have a -50 deg rotation from the master tool frame, 0 offset
Ttcp_tool = [rotz(-50) [0; 0; 0]; 0 0 0 1];

% Grinder tool
Ttcp_grinderTool = Ttcp_tool
TgrinderTool_push = [zeros(3) [0; 0; 102.82]; 0 0 0 1];
TgrinderTool_pull = [roty(180) [-50; 0; 67.06]; 0 0 0 1];

Ttcp_grinderPush = Ttcp_grinderTool*TgrinderTool_push;
Ttcp_grinderPull = Ttcp_grinderTool*TgrinderTool_pull;

% Portafilter tool
Ttcp_portaTool = Ttcp_tool;
TportaTool_rest = [roty(-7.5) [-32; 0; 27.56]; 0 0 0 1];
TportaTool_center = [roty(-7.5) [4.71; 0; 144.76]; 0 0 0 1];

Ttcp_portaRest = Ttcp_portaTool*TportaTool_rest;
Ttcp_portaCenter = Ttcp_portaTool*TportaTool_center;

% Cup tool
Ttcp_cupTool = Ttcp_tool;
TcupTool_center = [noRot [-47; 0; 186.11]; 0 0 0 1];
Ttcp_cupCenter = Ttcp_cupTool*TcupTool_center;


save("Tool Transformations")
```

Tool - Equipment transformations script

```
% Finds the HTs to get points of interest of tools to points of
% interest on equipment in the world frame.

clc, clear, close all

load("Machine Transforms")
load("Tool Transformations")

% Forward slash operator to inverse Ttcp_tool transform

% Grinder approach with portafilter tool
T_grinder_rest / Ttcp_portaRest;

% Grinder Start Button
T_grinder_start / Ttcp_grinderPush;

% Grinder stop button
T_grinder_stop / Ttcp_grinderPush;

% Grinder lever
T_grinder_lever / Ttcp_grinderPull;

% Tamper level
T_tamper_level / Ttcp_portaCenter;

% Tamper press
T_tamper_press / Ttcp_portaCenter;

% Silvia deliver
T_silvia_deliver / Ttcp_portaCenter;

% Silvia Cup
T_silvia_cup / Ttcp_cupCenter;
T_silvia_approach / Ttcp_cupCenter;

%Silvia buttons
T_silvia_but1 / Ttcp_grinderPush;

% Cup
T_cup_approach / Ttcp_cupCenter;
T_cup_get / Ttcp_cupCenter;
```

Appendix C - Code

RoboDK Python script

```
# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Documentation: https://robodk.com/doc/en/RoboDK-API.html
# Reference:     https://robodk.com/doc/en/PythonAPI/index.html
# Note: It is not required to keep a copy of this file, your python script is
saved with the station
from robolink import *    # RoboDK API
from robodk import *      # Robot toolbox
import numpy as np
RDK = Robolink()


# Initialize
robot = RDK.Item('UR5')
world_frame = RDK.Item('UR5 Base')
target_home = RDK.Item('Home')

robot.setPoseFrame(world_frame)
robot.setPoseTool(robot.PoseTool())


# Targets
grinder_approach = RDK.Item('Lux Rest Approach')
grinder_mate = RDK.Item('Lux Rest Mated')

# Custom joint angle targets
intermediate = [-80.640000, -85.030000, -72.070000, -113.070000, 89.500000,
-185.690000]
#intermediate_pt2 = [38.290695, -103.137039, 114.532374, -61.135172, 67.923789,
-79.650850]
intermediate_pt2 = [70.150000, -94.990000, 72.160000, -67.260000, -90.5200000,
145.100000]
intermediate_pt3 = [86.300000, -93.210000, 61.070000, -36.250000, 162.340000,
139.000000]
grinder_intermediate = [-6.476325, -81.638717, -149.649052, -111.057543,
-37.627386, 133.643]

grinder_rest_approach = [-13.815385, -92.211762, -144.12, -114.908578,
-58.969983, 135.458]
grinder_stop_approach = [-61.107196, -148.511559, -51.002562, -160.515472,
176.405029, -40.031784]
grinder_start_approach = [-61.208056, -145.437313, -57.067153, -157.524321,
176.304169, -40.030976]
grinder_stop_press = [-58.752518, -147.601066, -52.803127, -159.681532,
178.759706, -40.087954]
grinder_start_press = [-58.787624, -144.584541, -58.741238, -156.757587,
178.724600, -40.085595]

grinder_lever_approach = [-50.601499, -119.661808, -91.614578, -148.721936,
-110.601373, -129.997491]
grinder_lever_pull_0 = [-48.257997, -115.858025, -97.690815, -146.449506,
-108.257871, -129.997563]
grinder_lever_pull_1 = [-46.489258, -111.275445, -104.688084, -144.035216,
-97.295584, -129.997695]
```

```
grinder_lever_pull_2 = [-33.143411, -98.457705, -121.299577, -140.242213,
-64.418488, -129.997781]
#grinder_lever_pull_2 = [-35.891320, -99.516541, -120.008226, -140.474584,
-70.916396, -129.997810]

tamper_level_approach = [-9.244861, -113.499031, -113.575156, -123.244162,
-129.107685, 146.141503]
tamper_press_approach = [130.247688, -87.550834, -208.082242, -101.577011,
12.462490, 176.553340]
tamper_level_1 = [-9.244831, -110.400855, -112.261502, -127.655996, -129.107656,
146.141497]
tamper_level_2 = [-18.107311, -104.576578, -120.812969, -123.402882,
-137.824630, 148.353090]
tamper_press_1 = [148.796873, -79.567094, -222.797280, -73.042598, 29.420526,
153.496873]
tamper_press_2 = [7.321906, -98.160660, -134.443571, -119.261240, -112.735810,
143.161025]

silvia_deliver_1 = [-22.012830, -96.710934, -146.480964, -65.766794, -9.661230,
89.359257]
silvia_deliver_2 = [-72.300000, -119.290000, -109.030000, 39.900000, -45.000000,
-36.140000]

cup_get_1 = [ 53.324239, -97.393798, 156.574035, -59.180493, 53.324034,
-39.999341]
#cup_get_1 = [-53.324707, -82.606202, -156.574035, -120.819507, -53.324912,
-39.999647]
cup_get_2 = [ 67.303327, -82.065674, 141.093869, -59.028418, 67.303122,
-39.999409]
cup_got_1 = [ 67.303327, -97.012459, 134.231551, -37.219315, 67.303122,
-39.999409]
cup_got_2 = [ 50.808510, -120.143402, 147.951690, -27.808553, 50.808305,
-39.999327]
#cup_got_3 = [ 54.453591, -123.107305, 134.504375, -11.397323, 54.453386,
140.000]
cup_got_3 = [54.453591, -123.107305, 134.504375, -11.397323, 54.453386,
140.000000]

silvia_cup_1 = [86.524128, -95.697888, 131.839344, -36.141440, 162.774137,
140.000002]
#silvia_cup_2 = [52.710196, -85.775029, 123.476767, 322.536165, -231.039405,
140.000188]  #might need to be better
silvia_cup_2 = [-91.877414, -92.664577, -121.998134, -146.024001, -15.627840,
140.511925]

#silvia_but_all_a=[-11.223930, -81.315641, 128.851895, -227.53688,  26.231573,
-130.001561]
silvia_but_all_a=[-11.223930, -96.652221, 112.051698, -15.400103, -26.231573,
49.998439]

silvia_but_2_a = [ -1.517778, -65.017866, 104.515054, 140.501838, 16.525420,
-130.001191]
silvia_but_2_c = [  2.512836, -65.136388, 104.703740, 140.431369, 12.494806,
-130.000875]
```

```
silvia_but_2_off=[  2.512839, -64.689016, 104.940262, 139.747474, 12.494804,
-130.000875]
silvia_but_2_on= [  2.512835, -65.576850, 104.458604, 141.116966, 12.494808,
-130.000875]

#cup_drop_1 = [47.441334, -130.826791, 150.195265, 340.630484, -265.058656,
139.999631]
cup_drop_1 = [4.074385, -112.974729, 148.826667, 324.105159, -357.175603,
140.041143]
cup_drop_2 = [4.071791, -124.418215, 125.024500, 359.350773, -357.178197,
140.041182]
cup_dest = [42.231462, -82.043086, 134.097959, -52.055087, 42.231603,
138.748587]


# Some HTs calculated using matlab
bstart_approach_np = np.array([

   [  -0.4116,   -0.3454,  0.8434,  385.6817],
   [   0.6461,    0.5421,  0.5373, -612.6833],
   [  -0.6428,    0.7660,       0,   86.4500],
   [       0 ,         0,       0,   1.0000]],)

bstop_approach_np = np.array([
   [  -0.4116, -0.3454,  0.8434,  377.3392],
   [   0.6461,  0.5421,  0.5373, -598.0040],
   [  -0.6428,  0.7660,       0,   86.4500],
   [        0,       0,       0,   1.0000]])

rest_approach_np = np.array([
   [ -0.6022,  -0.3851,   0.6993,  348.9033],
   [ -0.4808,  -0.5243,  -0.7028, -299.6841],
   [  0.6373,  -0.7595,   0.1305,   91.8089],
   [       0,        0,        0,    1.0000]])

lever_approach_np = np.array([

   [  0.4545,  -0.5417,   0.7071,   436.0883],
   [  0.4545,  -0.5417,  -0.7071,  -435.4650],
   [  0.7660,   0.6428,       0 ,   161.1300],
   [       0,        0,       0 ,    1.0000]])

level_approach_np = np.array([
   [  0.3136,   0.4103,   0.8563,   510.9815],
   [ -0.7039,  -0.5048,   0.4997,  -132.4847],
   [  0.6373,  -0.7595,   0.1305,   155.5053],
   [       0,        0,        0,    1.0000]])

tamp_approach_np = np.array([
   [  0.3136,   0.4103,  0.8563,  435.3809],
   [ -0.7039,  -0.5048,  0.4997,   -2.9294],
   [  0.6373,  -0.7595,  0.1305,  132.5053],
   [       0,        0,       0,   1.0000]])
```

```
silvia_approach_np = np.array([
    [ 0.6010,   0.3838,  -0.7011,   -48.9496],
    [ 0.4823,   0.5252,   0.7011,  -401.0504],
    [ 0.6373,  -0.7595,   0.1305,   276.4353],
    [      0,        0,        0,     1.0000]])


bstart_approach = Mat(bstart_approach_np.tolist())
bstop_approach = Mat(bstop_approach_np.tolist())
rest_approach = Mat(rest_approach_np.tolist())
lever_approach = Mat(lever_approach_np.tolist())
level_approach = Mat(level_approach_np.tolist())
tamp_approach = Mat(tamp_approach_np.tolist())
silvia_approach = Mat(silvia_approach_np.tolist())


#-----------------------------------------------------
# Program Start
#-----------------------------------------------------

# Move robot to home
robot.MoveJ(target_home, blocking=True)

# Pick up portafilter tool
robot.MoveJ(intermediate)
RDK.RunProgram('Portafilter Tool Attach (Stand)', True)
robot.MoveJ(intermediate, blocking=True)

# Place portafilter in Grinder
robot.MoveJ(grinder_intermediate, True)
robot.MoveJ(grinder_rest_approach, True)
robot.MoveL(grinder_approach, True)
robot.MoveL(grinder_mate, True)
RDK.RunProgram('Portafilter Tool Detach (Grinder)', True)
robot.MoveL(grinder_rest_approach, True)

# Pick up grinder tool
robot.MoveJ(intermediate, True)
RDK.RunProgram('Grinder Tool Attach (Stand)', True)
robot.MoveJ(intermediate, True)

# Press grinder start button
robot.MoveJ(grinder_start_approach, True)
robot.MoveL(grinder_start_press, True)
robot.MoveL(grinder_start_approach, True)
robodk.pause(3)

# Press grinder stop button
robot.MoveJ(grinder_stop_approach, True)
robot.MoveL(grinder_stop_press, True)
robot.MoveL(grinder_stop_approach, True)

# Pull grinder lever 3 times
robot.MoveJ([-50.993480, -116.190224, -77.674069, -166.148459, 43.464147,
```

```
-83.291969], True)
robot.MoveJ(grinder_lever_approach, True)

robot.MoveJ(grinder_lever_pull_0, True)
robot.MoveL(grinder_lever_pull_1, True)
robot.MoveL(grinder_lever_pull_2, True)
robot.MoveL(grinder_lever_pull_1, True)
robot.MoveL(grinder_lever_pull_0, True)

robot.MoveL(grinder_lever_pull_1, True)
robot.MoveL(grinder_lever_pull_2, True)
robot.MoveL(grinder_lever_pull_1, True)
robot.MoveL(grinder_lever_pull_0, True)

robot.MoveL(grinder_lever_pull_1, True)
robot.MoveL(grinder_lever_pull_2, True)
robot.MoveL(grinder_lever_pull_1, True)
robot.MoveL(grinder_lever_pull_0, True)
robot.MoveJ(grinder_lever_approach, True)

# Return grinder tool to stand
robot.MoveJ(intermediate, True)
RDK.RunProgram('Grinder Tool Detach (Stand)', True)
robot.MoveJ(intermediate, True)

# Pick portafilter up from grinder
robot.MoveJ(grinder_intermediate, True)
robot.MoveJ(grinder_rest_approach, True)
RDK.RunProgram("Portafilter Tool Attach (Grinder)", True)
robot.MoveL(grinder_rest_approach, True)

# Scrape coffee grinds
robot.MoveJ(tamper_level_approach, True)
robot.MoveJ(tamper_level_1, True)
robot.MoveL(tamper_level_2, True)

# Tamp coffee
robot.MoveL(tamper_press_approach, True)
robot.MoveL(tamper_press_1, True)
robot.MoveL(tamper_press_2, True)
robot.MoveL(tamper_press_1, True)
robot.MoveL(tamper_press_approach, True)

# Move to Silvia
robot.MoveL(silvia_deliver_1, True)
robot.MoveL(silvia_deliver_2, True)
robodk.pause(30)

# Pick up cup tool
robot.MoveJ(intermediate, blocking=True)
RDK.RunProgram('Cup Tool Attach (Stand)', True)
robot.MoveJ(intermediate, blocking=True)
robot.MoveJ(intermediate_pt2, blocking=True)
```

```
# Get cup
robot.MoveJ(cup_get_1, blocking=True)
RDK.RunProgram('Cup Tool Open', True)
robot.MoveL(cup_get_2, blocking=True)
RDK.RunProgram('Cup Tool Close', True)
    #lift
robot.MoveL(cup_got_1, blocking=True)
    #back
robot.MoveL(cup_got_2, blocking=True)
robot.MoveJ(cup_got_3, blocking=True)
    #approach silvia
robot.MoveJ(silvia_cup_1, blocking=True)
    #place cup
robot.MoveL(silvia_cup_2, blocking=True)
RDK.RunProgram('Cup Tool Open', True)
robot.MoveL(silvia_cup_1, blocking=True)
RDK.RunProgram('Cup Tool Close', True)
    #lose tool
robot.MoveJ(intermediate_pt3, blocking=True)
RDK.RunProgram('Cup Tool Detach (Stand)', True)
    #get button presser
RDK.RunProgram('Grinder Tool Attach (Stand)', True)
robot.MoveJ(intermediate, blocking=True)
robot.MoveJ(intermediate_pt2, blocking=True)
    #press buttons
robot.MoveJ(silvia_but_all_a, blocking=True)
    #button
robot.MoveL(silvia_but_2_a, blocking=True)
robot.MoveL(silvia_but_2_on, blocking=True)
robot.MoveL(silvia_but_2_a, blocking=True)
    #wait
robodk.pause(3)
    #button
robot.MoveL(silvia_but_2_a, blocking=True)
robot.MoveL(silvia_but_2_off, blocking=True)
robot.MoveL(silvia_but_2_a, blocking=True)
    #escape
robot.MoveL(silvia_but_all_a, blocking=True)
    #drop tool
robot.MoveJ(intermediate_pt2, blocking=True)
RDK.RunProgram('Grinder Tool Detach (Stand)', True)
    #get cup tool
RDK.RunProgram('Cup Tool Attach (Stand)', True)
robot.MoveJ(intermediate, blocking=True)
robot.MoveJ(intermediate_pt2, blocking=True)
RDK.RunProgram('Cup Tool Open', True)
    #get cup of coffee
robot.MoveJ(cup_got_3, blocking=True)
robot.MoveJ(silvia_cup_1, blocking=True)
robot.MoveL(silvia_cup_2, blocking=True)
RDK.RunProgram('Cup Tool Close', True)
robot.MoveL(silvia_cup_1, blocking=True)
    #place cup on table
robot.MoveL(cup_drop_1, blocking=True)
```

```
robot.MoveL(cup_dest, blocking=True)
RDK.RunProgram('Cup Tool Open', True)
robot.MoveL(cup_drop_1, blocking=True)
RDK.RunProgram('Cup Tool Close', True)
robot.MoveL(cup_drop_2, blocking=True)
    #return tool
robot.MoveJ(intermediate_pt2, blocking=True)
RDK.RunProgram('Cup Tool Detach (Stand)', True)
    #be done
robot.MoveJ(intermediate, blocking=True)
robot.MoveJ(target_home, blocking=True)
```