HW3 ASSIGNMENT

MSE 4401

Nov 13, 2020

Mohammed Iqbal
MIQBAL57@UWO.CA
250861168

Intro:

Figure 1 below shows the manipulator drawing as per the DH convention.

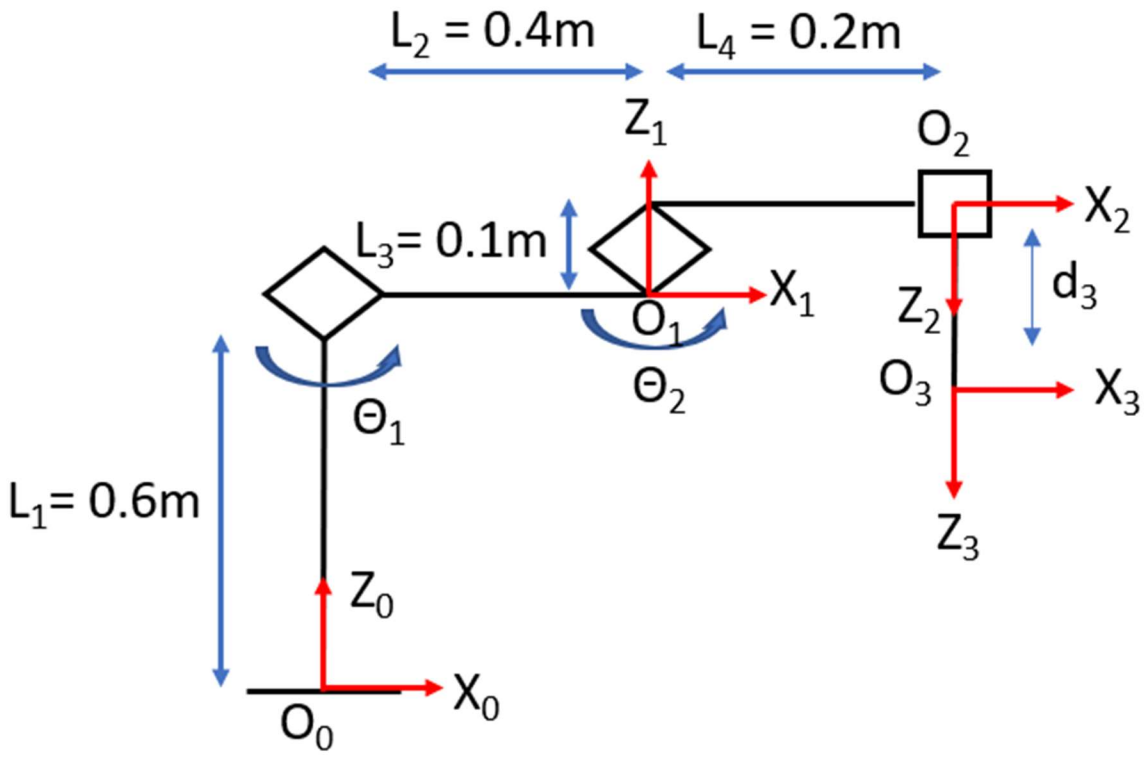

Figure 1: Manipulator reference drawing

The obstacles have changed from HW1, viewing Figure 2 below, there are railings on either side of the two conveyor systems and a structural pole in front of the manipulator. Each railing extends 15 cm above the conveyor system and is 4 cm thick. For this assignment the inner rails for both conveyor systems will be considered as obstacles. The structural pole is 20 cm in diameter and sits 65 cm along the positive y axis (base frame) of the manipulator, the height is 180 cm. The structural pole will be the third obstacle in the workspace of the manipulator. The coordinate system and link lengths used in the Solidworks model are to scale of Figure 1 above.

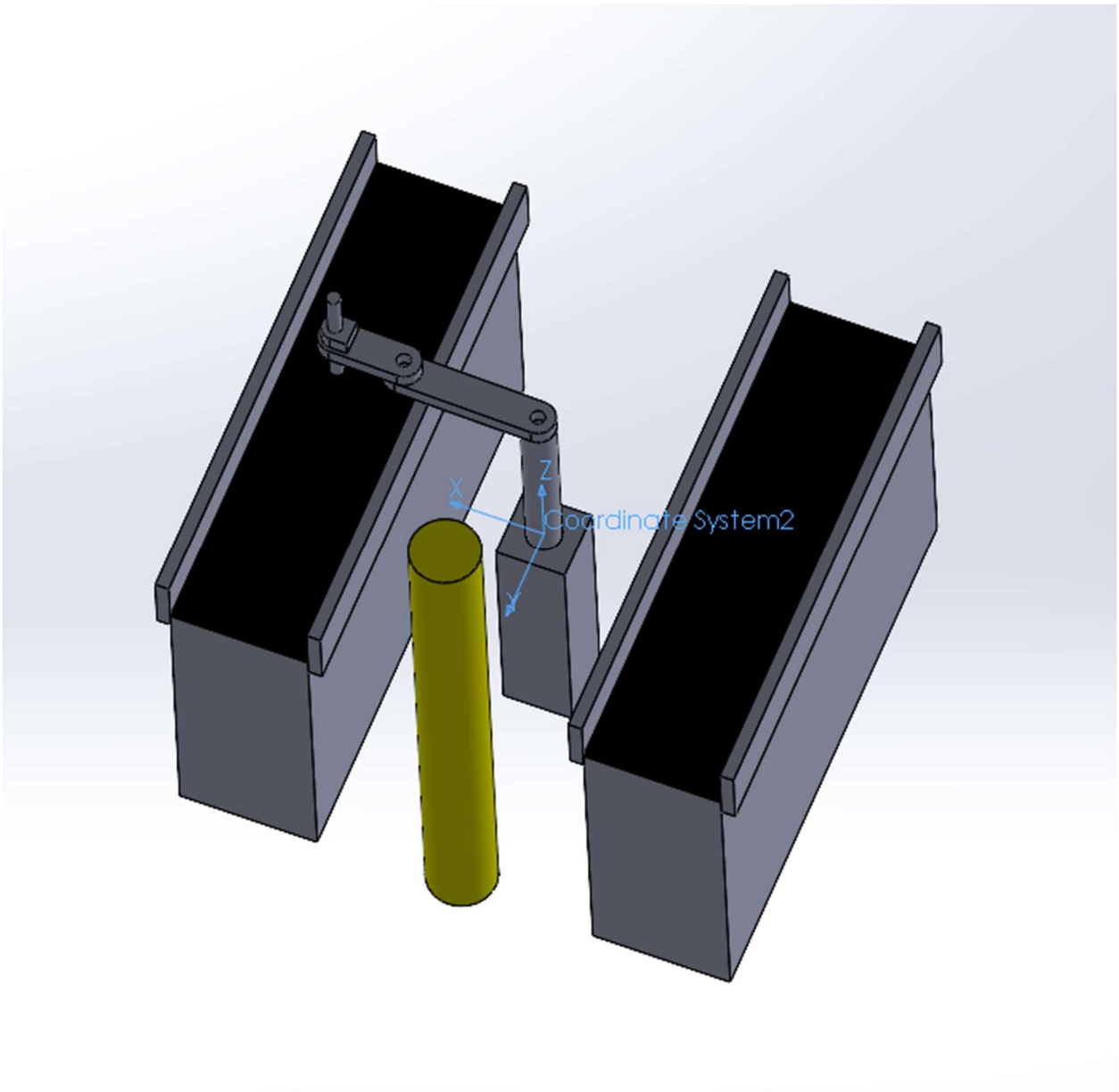

Figure 2: Manipulator in Q(0,0,0.2)

Question 1:

Algorithm execution summary:

The gradient decent algorithm for this manipulator is run under HW3Script (see Appendix 1). With the initial configuration and goal point of the end effector specified it calls the invkinscript (see Appendix 2) which returns the configuration for this goal point. During the execution of the algorithm the calculation of the torque is performed by the torque function (see Appendix 3). The torque function calls attractive force function (see Appendix 4) to determine the attractive forces on each of the frames (O1 to O3). The attractive force function can do this computation with the assistance of a modified version of the transformation matrix script (returns transformation matrix of each frame) which can be found in

Appendix 5. The torque function then calls the repulsive function (see Appendix 6) which computes the repulsive forces of each of the obstacles. The railings were approximated as two separate line segments in 3D space that sit along the center of each rail and have a region of influence of 15 cm. The structural pole was also treated as a line segment (through center long axis of pole) with a region of influence of 15 cm. With the attractive/repulsive forces, the torque function uses a modified jacobian script (see Appendix 7), which produces intermediary matrices, to calculate the total torque.

The algorithm loops for up to 1000 iterations unless the current configuration is close enough to epsilon then the algorithm exits. Through trial and error alpha1 and alpha2 were set to 3 (for theta1 and two respectively) and alpha3 was set to 0.1 as the prismatic joint has a small range of values to cover. Epsilon was chosen to be 3. After the step for each joint is calculated, a series of if-else-if statements ensure that the step does not go beyond the limits of the joint. In the case the algorithm gets stuck as a local minimum, the algorithm recalculates the torque for the current configuration, however this is multiplied by a random number between -0.5 to 0.5 and amplified by two. The variation in sign and magnitude aim to push the manipulator out of a local minimum. A series of checks are then done to ensure this random step is not out of bounds in terms of joint limits.

Question 2:

Path 1:

Figure 3 below shows the manipulator in the goal and final configuration (q(45,0,0.2) and q(168,18,0.2) respectively), where the goal point is (-0.59, 0.06, 0.5). Here if only Θ_1 were to rotate clockwise then the end effector would collide with the structural pole.

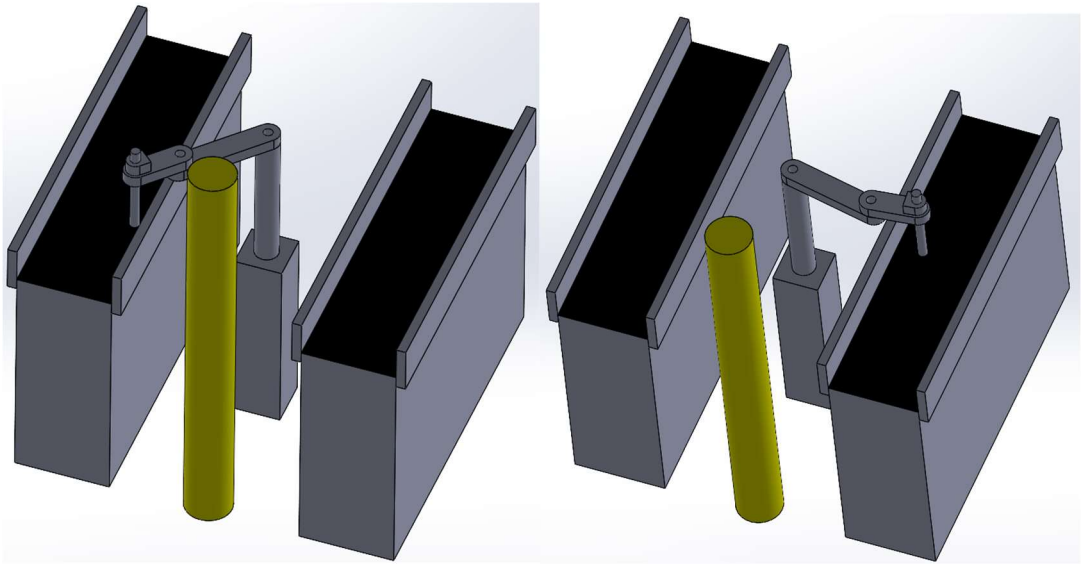

Figure 3: Manipulator in Start (Left) Config and End (Right) Config

While the full configuration list took 339 iterations (see sheet Path 1 of Path.xlsx for full list) Appendix 8 shows some key snippets of the configuration. Primarily how the manipulator responds as it approaches and leaves the structural pole. From the starting position the Θ_1 and Θ_2 receive a positive input as they currently are not near any obstacles. The prismatic joint receives no input as it is already in its final position and no obstacles are near that can cause it to move in its vertical direction. When the Θ_1 is rotated to about 70 degrees (around iteration 25), O2 and O3 begin to fall into the region of influence of the structural pole so the input for Θ_2 starts to go negative. As Θ_1 remains outside this region of influence it increases positively towards the goal position.

Once Θ_1 is at 90 degrees (around iteration 186) this is the position where if Θ_2 had an input of 0 degrees the end effector would collide with the pole. Observing Appendix 8, the algorithm assigns Θ_2 an input of about -51 degrees. Figure 4 shows the manipulator in the configuration q(90,-51,0.2).

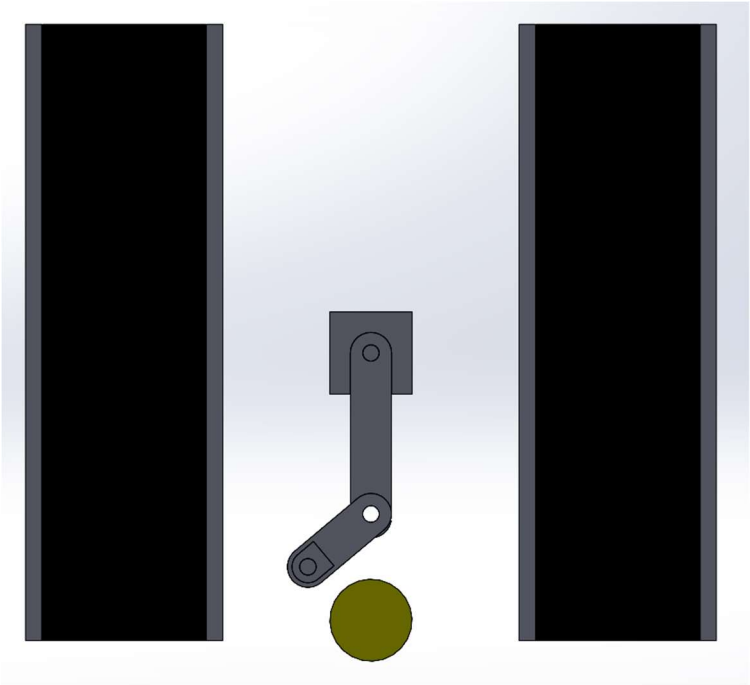

Figure 4: Manipulator in config q(90,-51,0.2)

Next at iteration 243, Θ_1 is at 118 degrees and the algorithm gives Θ_2 a value of about -91 degrees. Figure 5 below shows the manipulator in the configuration q(118,-91,0.2) and clearly shows how the manipulator is able to avoid hitting the structural pole.

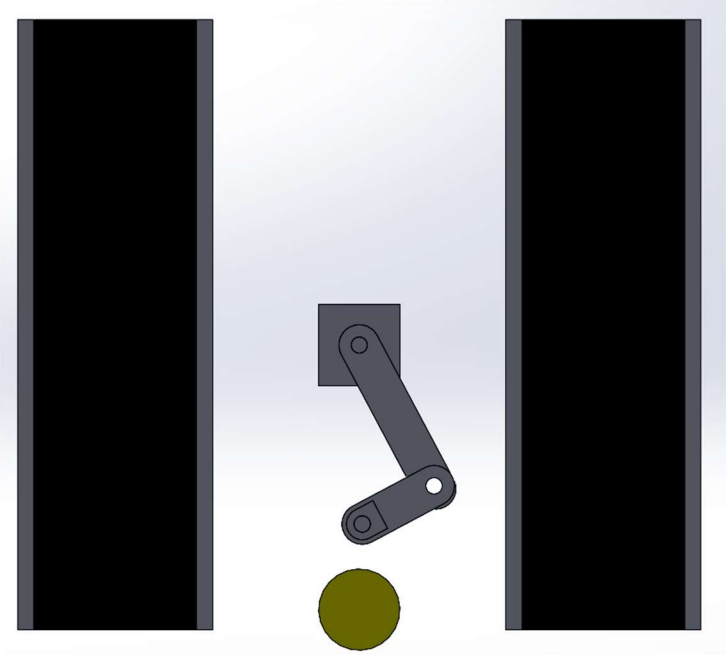

Figure 5: Manipulator in configuration q(118,-91,0.2)

Once the manipulator has cleared the region of influence of the structural pole, Θ_2 is then only affected by the goal position so beings to increase positively in input (see Appendix 8 at around iteration number 261). By iteration number 261 Θ_1 has reached its goal configuration, so the algorithm continues to run getting Θ_2 to its goal configuration. At iteration 339 the algorithm exits as manipulator close enough to the goal configuration to be acceptable. Observing Appendix 8 and comparing the configuration of q(170, 16, 0.2) to the goal configuration q(168,18,0.2), this can be seen to be true. Figure 6 below shows the manipulator in this final configuration.

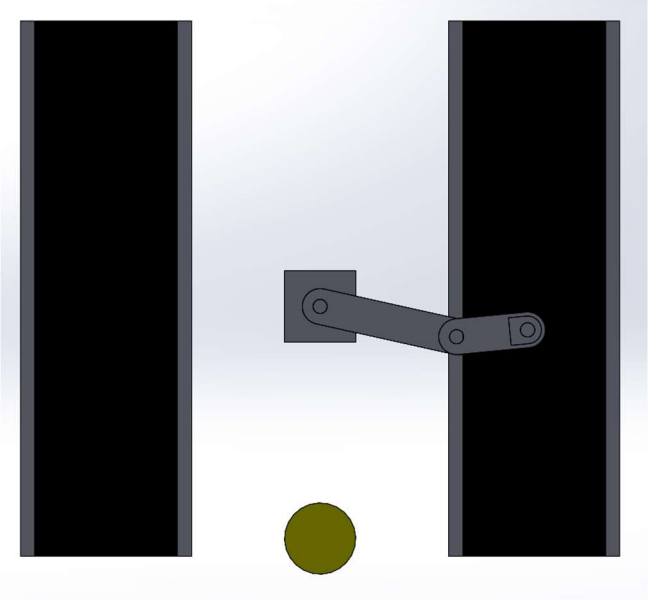

Figure 6: Manipulator in configuration q(170,16,0.2)

Path 2:

Path 2 will follow the same path as Path 1, the difference being Θ_1 starts at 10 degrees and the prismatic joint will be fully extended, such that if the end effector does not move in the z direction (of base frame) then it will collide with the rails of the two conveyor systems. Figure 7 shows the manipulator in the start and goal configurations (q(10,0,0.4) and q(168,18,0.4). The goal point is (-0.59,0.06,0.3).

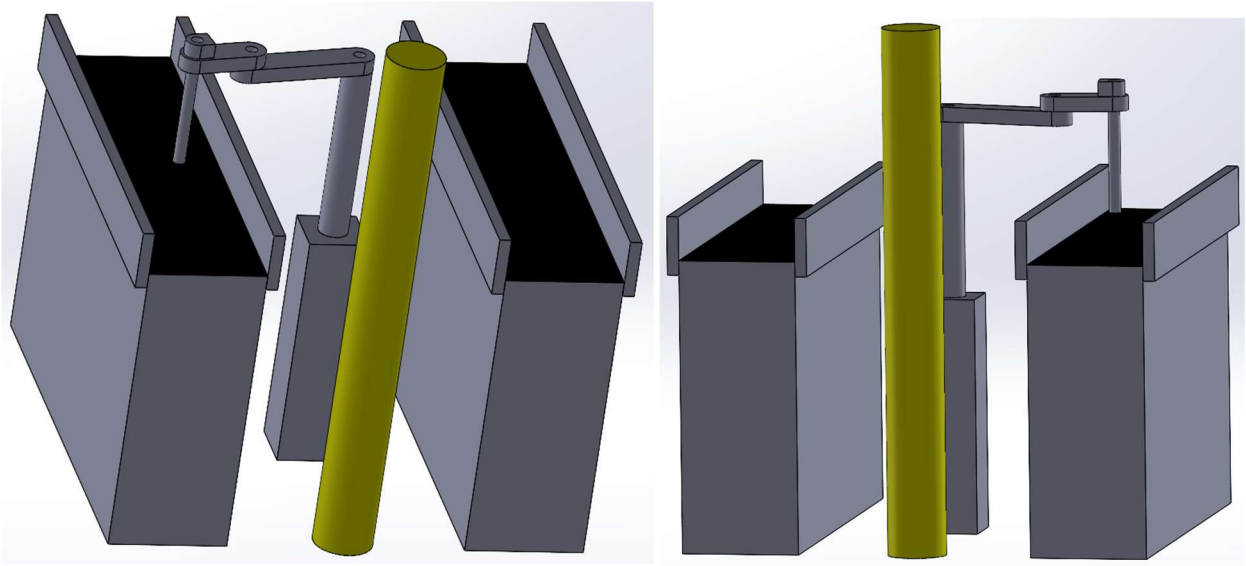

Figure 7: Manipulator in Start (Left) Config and End (Right) Config

As Θ_1 starts at 10 degrees instead of 45 and d3 (prismatic joint) also must move, the increased movement of the manipulator results in 434 iterations (reasonably longer compared to Path 1). This longer iteration time is also due in part to the smaller step size of the prismatic joints (alpha3 = 0.1) compared to the rotary joints (alpha1 = alpha2 = 3). The full configuration list can be found in Path2 of Path.xlsx, while a summary version can be found in Appendix 9. Around iteration 8 the manipulator is close to the left rail (see Figure 8 below) there for d3, the prismatic joint, is retracted to avoid the rail.

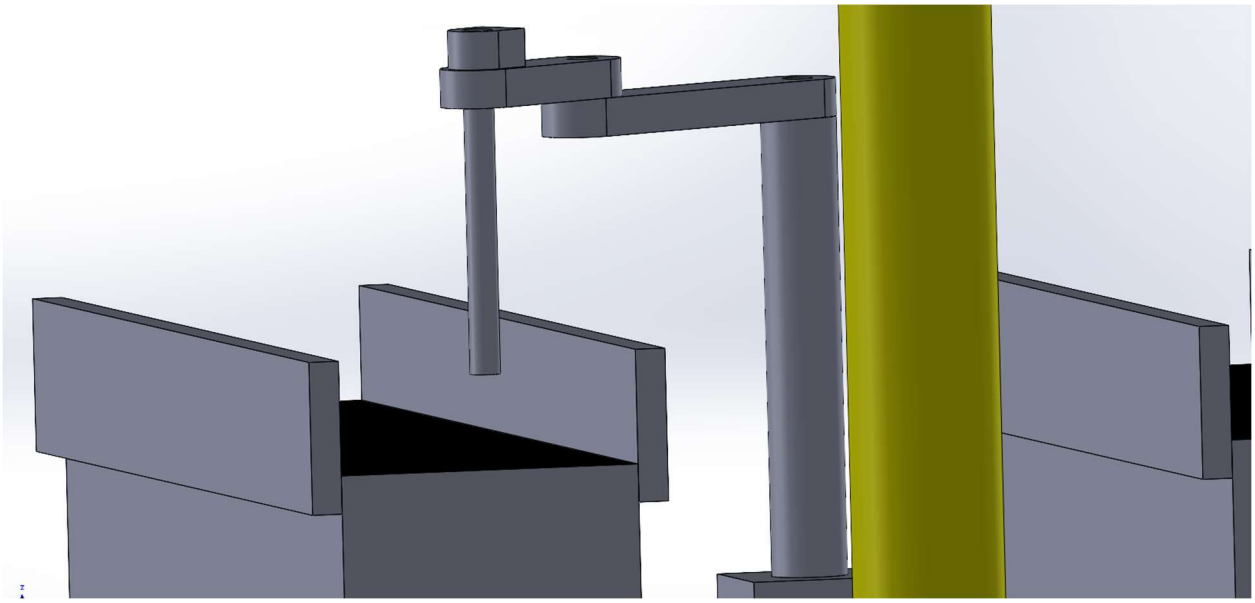

Figure 8: Manipulator in Configuration q(25,2,0.35)

By iteration 19 the end effector is on top of the rail and the algorithm assigns the configuration q(49,6,0.27). Figure 9 below shows the manipulator in this configuration and how the end effector clears the left rail.

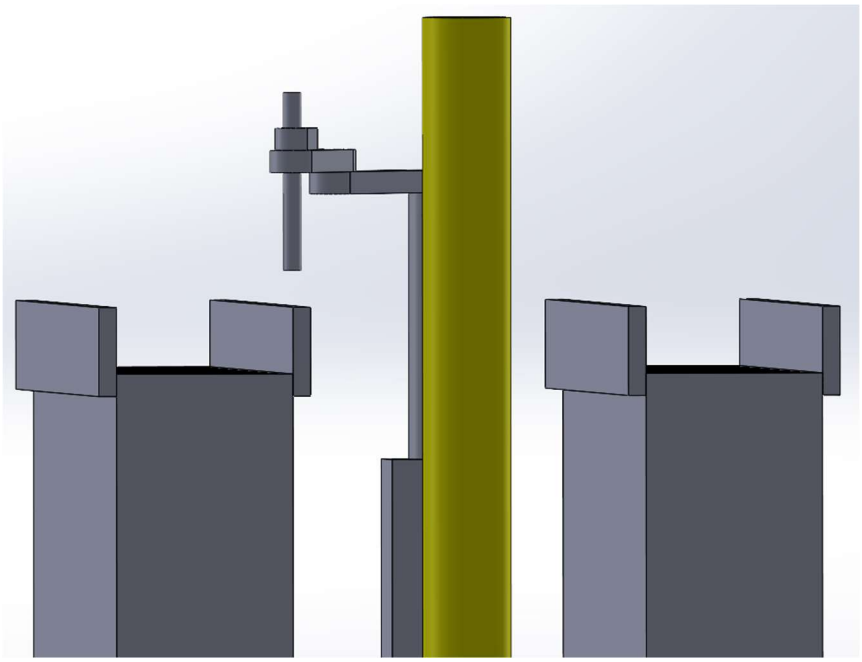

Figure 9: Manipulator in Configuration q(49,6,0.27)

By iteration 97 the end effector leaves the region of influence of the left rail and is around its goal position (for prismatic joint). Figure 10 below shows the manipulator avoiding the structural pole (around iteration 287) and how the prismatic joint is still extended as changing its height does not reduce the influence of the pole.

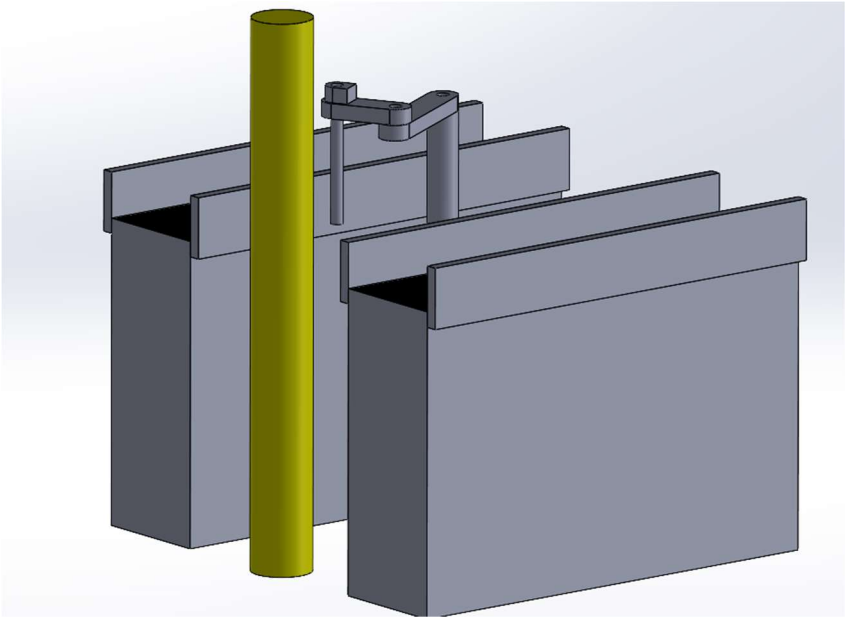

Figure 10: Manipulator in Configuration q(119,-90,0.4)

Around iteration 300 the end effector approaches the right rail and begins to retract. Figure 11 shows the manipulator at iteration 324 and is positioned right over the right rail. Here it can again be seen the end effector clears this rail as well.

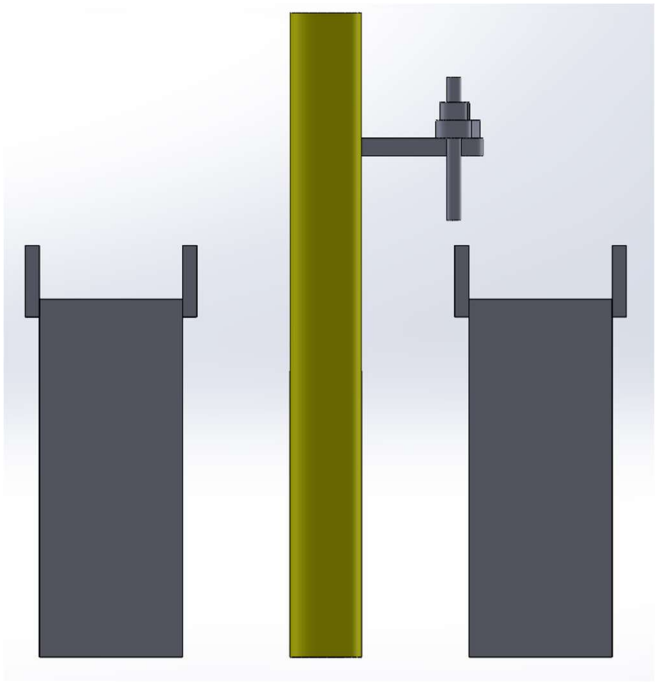

Figure 11: Manipulator in Configuration q(168,-80,0.3)

Once the end effector has cleared this final rail (around iteration 362), it is only affected by the attractive forces of the goal point. Thus, Figure 12 shows the manipulator at iteration 434 where the goal position is met q(169,16,0.4). Compared to the goal configuration of q(168,18,0.4), the algorithm is sufficiently close to its goal.

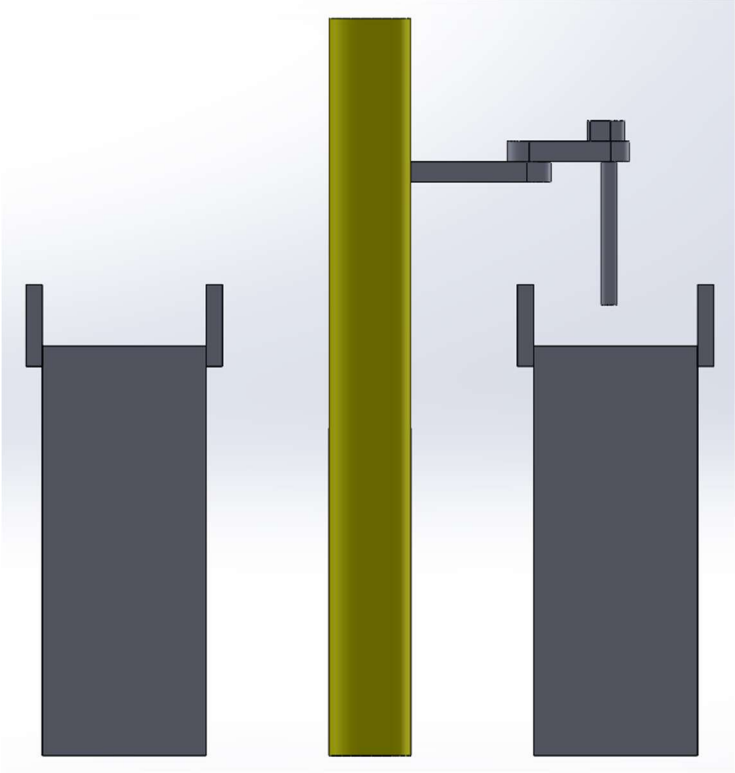

Figure 12: Manipulator in Configuration q(169,16,0.4)

Question 3:

Path 3:

For Path 3 the start and end position of the manipulator is the same as Path 2 (qs(10,0,0.4) and qf(168,18,0.4)) however the region of influence of the left rail (quadrant 1 and 4) has been increased from 15 cm radius to 20 cm radius (in the repulsive function). Likewise, the region of influence of the structural pole has been increased to a 20 cm radius (from 15 cm). The entire configuration list can be viewed under sheet Path3 in Path.xlsx. A summary list can be found in Appendix 10. Observing iterations 1 to 10 the increased region of influence of the rail starts affecting the manipulator quite earlier compared to Path 2 (manipulator begins responding at around iteration 10). Figure 13 shows the manipulator at iteration 3 in the algorithm assigned configuration of q(7.5,-0.8,0.23). Here it can be seen how much more clearance the manipulator is giving to the left rail.

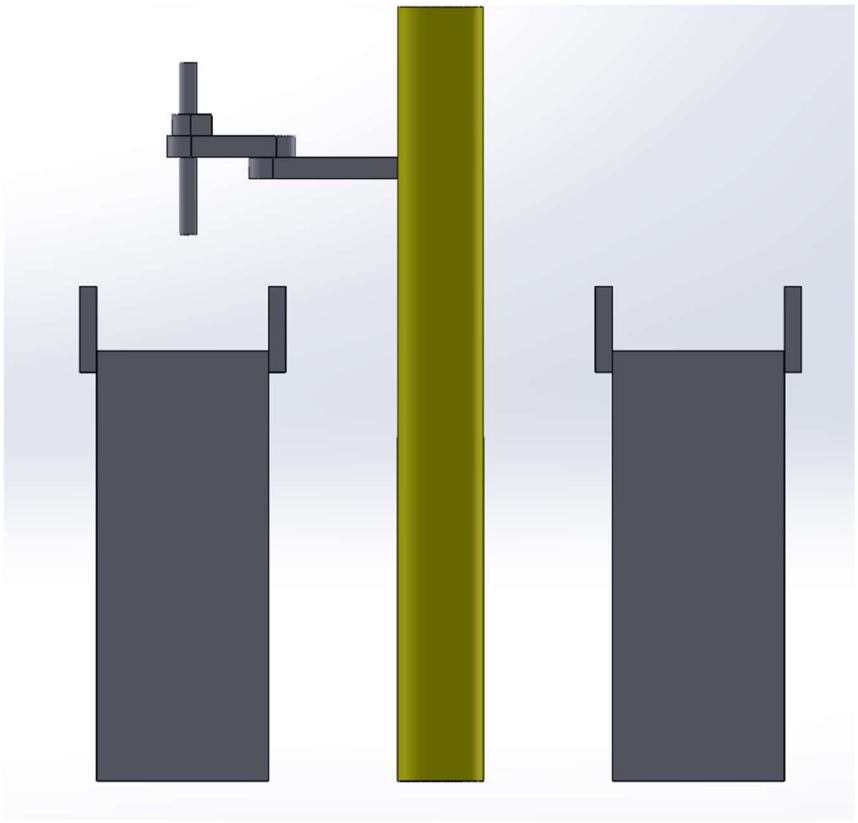

Figure 13: Manipulator in Configuration q(7.5,-0.8,0.23)

The next change can be seen at around iteration 34, O2 begins to fall into the region of influence of the structural pole, when Θ_1 at 70 degrees. However in the case of Path 2, this occurred at iteration 27 as Θ_1 was not influenced as much by the railing (due to end effector begin at farther edges of region on influence) so Θ_1 reached 70 degrees sooner. Additionally, d3 remains retracted for a longer period and does not begin to extend until iteration 300 (d3 extended by iteration 80 in Path 2). Figure 14 below shows the manipulator in configuration q(90, -59, 0.15) at iteration 281. While the manipulator avoids the structural pole, as in the case of the railing, it gives more clearance to the pole (Θ_2 retracted 10 degrees more than during Path 2)

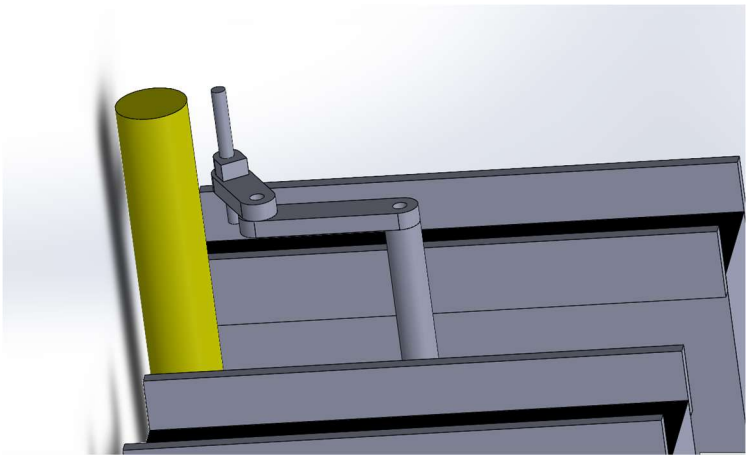

Figure 14: Manipulator in Configuration q(90,-59,0.15)

The total number of iterations is also longer than Path 2 (472 to 434 respectively). This is due in part to how much longer Θ_2 stays in the negatively retracted position to avoid the structural pole. Θ_2 does not begin rotating positively until iteration 346 (compared to iteration 300 of Path2). As the region of influence for the right rail was not changed, it does not overlap and contain the goal point. Thus, by iteration 472 the algorithm exits as the manipulator has finally reached goal position (see Appendix 10). With the final configuration specified by the algorithm to be q(169.16,0.4) this is the same as Path 2.

Question 4:

Path 4:

Again Path 4 follows the same path as Path 2 in terms of start and end configuration/point. While the region of influence stays the same as Path 2, the attractive and repulsive constants (for goal and obstacles respectively) changed to zeta1 = 1, zeta2 = 0.125, zeta3 = 0.125 and aeta1 = 0.25, aeta2 =1, aeta3 = 1. Where zeta1 corresponds to attractive constant for O1 (and so on) and aeta1 is repulsive constant for rail in quadrant 2,3, aeta2 is repulsive constant for rail in quadrant 1,4, and aeta3 is the repulsive constant for the structural pole. The full configuration list can be found in sheet Path4 in Path.xlsx, the summary version can be found in Appendix 11. While both Path 4 and 2 result in d3 retracting at iteration 8 (since same region of influence), the difference in repulsive and attractive constants become clear at around iteration 115 where the manipulator has left the region of influence of the left rail but due to the small attractive constants the prismatic joint still is not fully extended. Figure 15 shows the manipulator in this configuration (q(80,-29,0.375)).

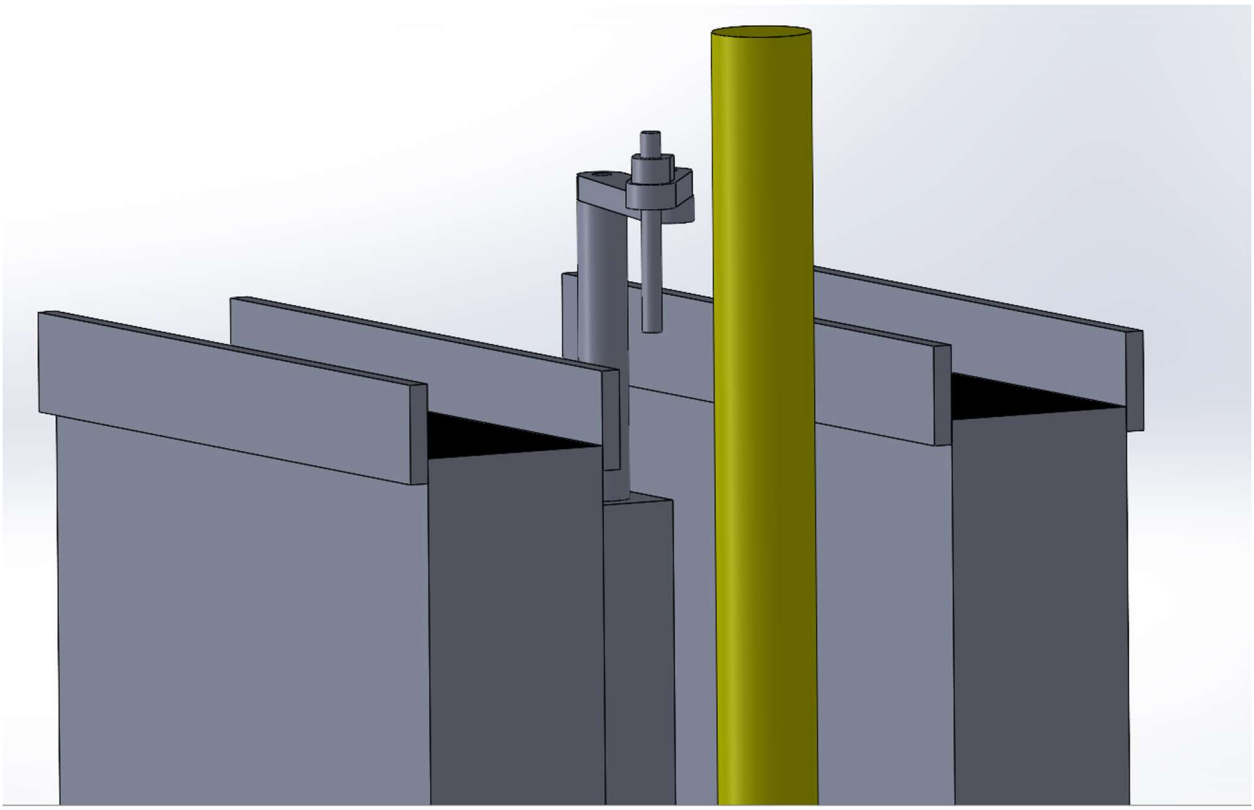

Figure 15: Manipulator in configuration q(80,-29,0.375)

The push to reach its goal position is a lot weaker which results in smaller steps taken (unit torque vector does not change direction as fast) and a longer iterations (528 in total compared to the 434 of Path 2). The larger repulsive constants result in obstacles have a stronger initial rejection once a frame enters its region of influence. Figure 16 below shows the configuration iterations for Path 2 and 4 respectively. Between iteration 15 and 16 for both Paths, Path 4 exhibits a stronger rejection from the rail than Path 2. Then from iterations 16 to 19 the prismatic joint in Path 4 takes longer to recover and reextend. However, since the prismatic joint in Path 2 falls further into the region of influence of the rail it too sees a large rejection force.

8	25.29393	2.079744	0.35081
9	28.2421	2.60177	0.357118
10	31.19051	3.134076	0.362228
11	29.58884	2.568892	0.279798
12	32.51345	3.095133	0.293531
13	35.44507	3.631138	0.304995
14	38.38096	4.175283	0.314684
15	37.57364	3.890389	0.218843
16	40.47774	4.429911	0.236326
17	43.39212	4.977315	0.251481
18	46.31367	5.53128	0.264717
19	49.24027	6.090751	0.27636
20	49.3076	6.111349	0.176387
21	52.2046	6.667153	0.194598
22	55.11066	7.228266	0.210926

8	25.48189	0.547715	0.350551
9	28.47043	0.791688	0.353725
10	31.45894	1.040477	0.356494
11	29.94053	0.524068	0.271984
12	32.92225	0.769867	0.279358
13	35.90534	1.019945	0.28591
14	38.88938	1.273749	0.291787
15	41.87409	1.53082	0.2971
16	41.28839	1.331673	0.199249
17	44.26413	1.587957	0.208636
18	47.24173	1.847097	0.217246
19	50.22078	2.108769	0.225189
20	53.20097	2.372694	0.232555
21	56.18206	2.638631	0.239419
22	59.16386	2.906366	0.245844

Figure 16: Snippet of Appendix 11, Path2 Configuration List (Left), Path 4 Configuration List (Right)

The manipulator follows a similar pattern in the case of the structural pole, observing iteration 163 in Appendix 11 the manipulator exhibits a similar configuration to Path 2 (q(90,-51,0.38)) vs q(89,-50,0.399) respectively). Since the rail in quadrant 2,3 is unmodified in terms constants, the algorithm achieves the goal position (q(167,15.7,0.4)) at iteration 528 (see Figure 17 below).

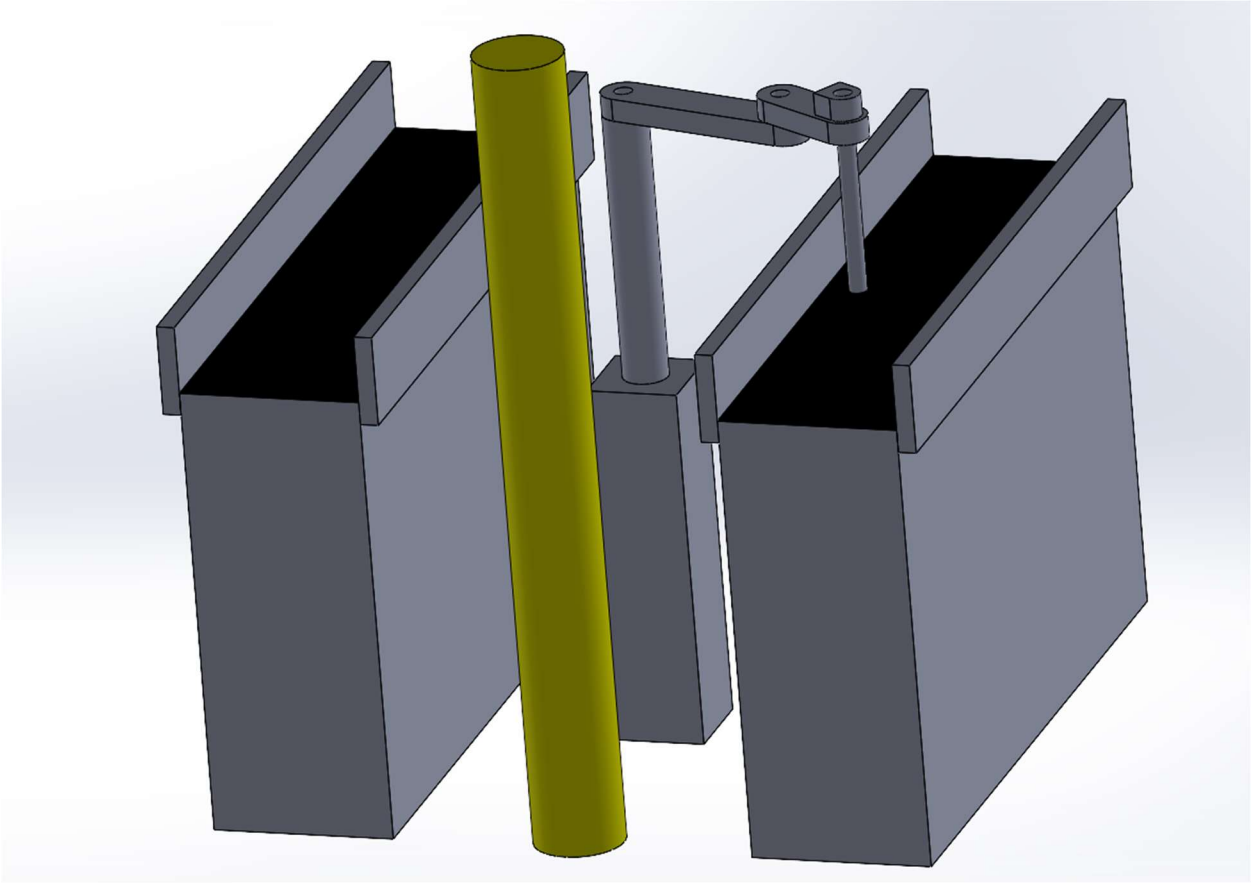

Figure 17: Manipulator in final configuration q(167,15.7,0.4)

In summary the noticeable changes were: the increased iteration times due to weaker goal attraction, and the stronger initial rejection of obstacles in Path. However, the strong initial rejection of Path 4 ends up being a wash compared to Path 2 due to the eventual strong rejection in Path 2 due to the manipulator straying deeper into the region of influence of the obstacle.

Appendix:

Appendix 1 – HW3Script.m:

```
format short
close all
clear all
%Determine initial configuration
Q1 = [10,0,0.4];


%Goal Point in Workspace
xG = -0.59;
```

```
yG = 0.06;
zG = 0.3;

%Get configuration for this point using inverse kinematics
[sol1, sol2] = invkinscript(xG,yG,zG);
QG = sol1;

disp('Starting Configuration: ');
disp(Q1);
disp('Final Configuration: ');
disp(QG);

%specify various parameters for torque calculation
qCurr = Q1; %Passing starting configuration to algorithm
EndQ = QG;  %Passing ending configuration to algorithm
zeta1 = 2; %attractive force constant for O1, O1 set as leader
zeta2 = 0.75; %attractive force constant for O2
zeta3 = 0.75; %attractive force constant for O3
d = 0.01; %distance in cm that changes conic potential to parabolic
aeta1 = 0.25; %repulsive constant for Left Conveyor Rail (Quadrant 2,3)
aeta2 = 0.5; %repulsive constant for Right Conveyor Rail (Quadrant 1,4)
aeta3 = 0.5 ;%repulsive constant for Structural Pole (Quadrant 1,2)

%specify various parameters for gradient decent algorithm
alpha1 = 3; %step value for theta1
alpha2 = 3; %step value for theta2
alpha3 = 0.1; %step value for d3
epsilon = 3;
epsilonM = 0.1;
configMat = Q1; %matrix to hold configuration list
FattMat =[];

%Gradient Descent Algorithm
for i =1:1000
    %calculate the unit torque vector
    torqVec = torque(qCurr,EndQ,zeta1,zeta2,zeta3,d,aeta1,aeta2,aeta3);
    unitTorqVec = torqVec/(norm(torqVec));

    %Save previous configuration to make sure we aren't in local minimum
    qPrev = qCurr;

    %See if current configuration is close enough to goal configuration
    if norm(qCurr - QG) > epsilon

        %Determine step for each joint
        stepVec =
[alpha1*unitTorqVec(1),alpha2*unitTorqVec(2),alpha3*unitTorqVec(3)];

        %Ensuring Step is within bounds of rotary joint theta1
        if qCurr(1) + stepVec(1) > 360
            qCurr = [360, qCurr(2), qCurr(3)];
        elseif qCurr(1) + stepVec(1) < 0
            qCurr = [0, qCurr(2), qCurr(3)];
        else
            qCurr = [qCurr(1)+stepVec(1), qCurr(2), qCurr(3)];
```

```
        end

        %Ensuring Step is within bounds of rotary joint theta2
        if qCurr(2) + stepVec(2) > 180
            qCurr = [qCurr(1), 180, qCurr(3)];
        elseif qCurr(2) + stepVec(2) < -180
            qCurr = [qCurr(1), -180, qCurr(3)];
        else
            qCurr = [qCurr(1),qCurr(2)+stepVec(2),qCurr(3)];
        end

        %Ensuring Step is within bounds of prismatic joint d3
        if qCurr(3) + stepVec(3) > 0.4
            qCurr = [qCurr(1),qCurr(2),0.4];
        elseif qCurr(3) + stepVec(3) < 0
            qCurr = [qCurr(1),qCurr(2),0];
        else
            qCurr = [qCurr(1),qCurr(2),qCurr(3)+stepVec(3)];
        end

    %Current config is close enough to goal, so exit algorithm
    else
        configMat = [configMat;qCurr];
        break
    end

   %Add current configuration to list and keep looping towards goal
    configMat = [configMat;qCurr];

    %Determine if stuck in local minimum
    if i >1 && (norm(qCurr - qPrev) < epsilonM)
        disp('Manipulator in Local Minima');

        %Take a random walk
        walkConstant = 2*(rand()-0.5);
        randomStepVec = transpose(walkConstant*unitTorqVec);

        %Verify random step is within bounds of theta1
        if (qCurr(1) + randomStepVec(1)) > 360 ||(qCurr(1) +
randomStepVec(1)) < 0
            qCurr = [qCurr(1) - randomStepVec(1), qCurr(2),qCurr(3)];
        elseif (qCurr(1) - randomStepVec(1)) > 360 ||(qCurr(1) -
randomStepVec(1)) < 0
            qCurr = [qCurr(1) + randomStepVec(1), qCurr(2),qCurr(3)];
        else
            qCurr = [qCurr(1) - randomStepVec(1), qCurr(2),qCurr(3)];
        end

        %Verify random step is not going past joint limits of theta2
        if (qCurr(2) + randomStepVec(2)) > 180 ||(qCurr(2) +
randomStepVec(2)) < -180
            qCurr = [qCurr(1), qCurr(2)- randomStepVec(2),qCurr(3)];
        elseif (qCurr(2) - randomStepVec(2)) > 180 ||(qCurr(2) -
randomStepVec(2)) < -180
            qCurr = [qCurr(1), qCurr(2)+ randomStepVec(2),qCurr(3)];
```

```
        else
            qCurr = [qCurr(1), qCurr(2) - randomStepVec(2),qCurr(3)];
        end

        %Using modulo operator to ensure random step is within bounds of d3
        qCurr = [qCurr(1), qCurr(2),mod(qCurr(3)- randomStepVec(3),0.4)];

    end

end
```

Appendix 2 – invkinscript.m:

```
function [Sol1 Sol2] = invkinscript(givenx,giveny,givenz)
%function that takes the end effector position as input(x,y,z) and outputs
%two configuration vectors, the first being the elbow up configuration and
%the second the elbow down configuration

%rounding to 10 decimal places to prevent issues with truncation errors in
%matlab
x = round(givenx,10);
y = round(giveny,10);
z = round(givenz,10);
%-------------------------------------------------------------------------%
%Link Lengths for calculation
L1 = 0.6;
L2 = 0.4;
L3 = 0.1;
L4 = 0.2;

%-------------------------------------------------------------------------%
%calculating joint variable d3
d3 = L1 + L3 - z;

%-------------------------------------------------------------------------%
%calculating theta2
C2 = ((x^2) + (y^2) - ((L2)^2) - ((L4)^2))/(2*(L2)*(L4));

%calculating both positive and negative case for S2
S2_pos = sqrt(1-((C2)^2));
S2_neg = -S2_pos;
theta2_pos = atan2d(S2_pos,C2);
theta2_neg = atan2d(S2_neg,C2);

%-------------------------------------------------------------------------%
%calculating theta1 for each case of theta2

alpha = atan2d(y,x);
beta = acosd((((L2)^2)+ (x^2) + (y^2) - (L4)^2)/(2*(L2)*sqrt((x^2)+(y^2))));

%for case of theta2_pos calculate respective theta1_pos
```

```
    theta1_pos = alpha - beta;
%map theta1_pos to be between 0 and 360
    theta1_pos = theta1_pos + 360*(theta1_pos < 0);
%for case of theta2_pos calculate respective theta1_pos
    theta1_neg = alpha + beta;

%output the two configuration vectors
Sol1 = [theta1_pos,theta2_pos,d3];
Sol2 = [theta1_neg,theta2_neg,d3];

end
```

Appendix 3 – torque.m:

```
function torqueVec = torque(CurrQ,EndQ,zeta1,zeta2,zeta3,d,aeta1,aeta2,aeta3)
%torque function that calls attractive func and repulsive func to determine
%the attractive and repulsive forces for O1,O2,O3. Then calls the jacobian
%function and converts the forces into torques in the configuration space.

%Determine attractive forces for current configuration
[FattO1, FattO2, FattO3] = attractive(CurrQ,EndQ,zeta1,zeta2,zeta3,d);

%Determine repuslive forces for current configuration
[FrepO1, FrepO2, FrepO3] = repulsive(CurrQ,aeta1,aeta2,aeta3);

%Determine jacobian matricies for O1,O2,O3
[JVO1, JVO2, JVO3] = jacobianscriptMOD(CurrQ(1),CurrQ(2),CurrQ(3));

%Determine attractive torques
TattO1 = transpose(JVO1)*FattO1;
TattO2 = transpose(JVO2)*FattO2;
TattO3 = transpose(JVO3)*FattO3;

%Determine repulsive torques
TrepO1 = transpose(JVO1)*FrepO1;
TrepO2 = transpose(JVO2)*FrepO2;
TrepO3 = transpose(JVO3)*FrepO3;

%Determine total torques for O1,O2,O3
TtotalO1 = TattO1 + TrepO1;
TtotalO2 = TattO2 + TrepO2;
TtotalO3 = TattO3 + TrepO3;

%return result
torqueVec = TtotalO1 + TtotalO2 + TtotalO3;


end
```

Appendix 4 – attractive.m:

```
function [FattO1, FattO2, FattO3] =
attractive(CurrQ,EndQ,zeta1,zeta2,zeta3,d)
```

```
%function that takes current configuratin, goal configuration and
%calculates attractive forces on O1,O2,O3. Can be tweaked by zeta value.
%Parabolic or Conic well used depends on d

%Transformation matrix for final configuration
[T01, T02, T03] = tmatrixscriptMOD(EndQ(1),EndQ(2),EndQ(3));

%Get OF (O for final configuration
O1F = T01(1:3,4);
O2F = T02(1:3,4);
O3F = T03(1:3,4);

OFMat = [O1F,O2F,O3F];

%calculate attractive for for each joint

%Transformation matricies for each joint at curr position
[T01C, T02C, T03C] = tmatrixscriptMOD(CurrQ(1),CurrQ(2),CurrQ(3));

%Frame origins for each joint at current config
O1C = T01C(1:3,4);
O2C = T02C(1:3,4);
O3C = T03C(1:3,4);

OCMat = [O1C,O2C,O3C];

FattMatrix = [];

%calculate attractive forces for each frame using appropriate attractive
%field
for i = 1:3
    %calculate norm between current frame and goal frame
    goalDist = norm(OCMat(:,i)-OFMat(:,i));

    %set zeta value depending on which frame being assessed
    if i == 1
        zeta = zeta1;
    elseif i == 2
        zeta = zeta2;
    else
        zeta = zeta3;
    end

    %calculate attractive force depending on goaldistance and d
    if goalDist <= d
        %Use parabolic potential
        f = -zeta*(OCMat(:,i)-OFMat(:,i));
        FattMatrix = [FattMatrix,f];
    else
        %Use conic potential
        f = -d*zeta*((OCMat(:,i)-OFMat(:,i))/goalDist);
        FattMatrix = [FattMatrix,f];
    end
```

```
end
%Seperate FattMatrix into attractive forces for each frame
FattO1 = FattMatrix(:,1);
FattO2 = FattMatrix(:,2);
FattO3 = FattMatrix(:,3);


end
```

Appendix 5 –tmatrixscriptMOD.m:

```
function [T01 T02 T03] = tmatrixscriptMOD(theta1,theta2,D3)
%units in cm and deg

%Function that returns resultant transformation matrix T03 when given
%joint variables theta1,theta2, and D3.
%-------------------------------------------------------------------------%
%inputs for DH parameters
L1 = 0.6;
L2 = 0.4;
L3 = 0.1;
L4 = 0.2;

a1 = L2;
a2 = L4;
a3 = 0;

d1 = L1;
d2 = L3;
d3 = D3;

alpha1 = 0;
alpha2 = 180;
alpha3 = 0;

t1 = theta1; %theta value
t2 = theta2;
t3 = 0;


%-------------------------------------------------------------------------%
%A matrix calculation

A1 = amatrixcalc(a1,d1,alpha1,t1);

A2 = amatrixcalc(a2,d2,alpha2,t2);

A3 = amatrixcalc(a3,d3,alpha3,t3);

%-------------------------------------------------------------------------%
%Intermediate Transformation Matrix Calculation
T01 = A1;
T02 = A1*A2;
```

```
T03 = A1*A2*A3;

%-------------------------------------------------------------------------%
%sub function used to calculate A matrix
function a_matrix_output = amatrixcalc(a,d,alpha,theta)
    a_matrix_output = [cosd(theta),-
sind(theta)*cosd(alpha),sind(theta)*sind(alpha),a*cosd(theta);
    sind(theta),cosd(theta)*cosd(alpha),-
cosd(theta)*sind(alpha),a*sind(theta);
    0,sind(alpha),cosd(alpha),d;
    0,0,0,1];
end
%-------------------------------------------------------------------------%
end
```

Appendix 6 – repulsive.m:

```
function [FrepO1, FrepO2, FrepO3] = repulsive(CurrQ,aeta1,aeta2,aeta3)
%function that takes current configuration and aeta values for O1,O2,O3
%then computes the repulisive forces of applicable obstacles on O1,O2,O3
%returns the repulsive force for O1,O2,O3
%aeta1 for O1
%aeta2 for O2
%aeta3 for O3
%Determine joint origins in current configuration
[T01C, T02C, T03C] = tmatrixscriptMOD(CurrQ(1),CurrQ(2),CurrQ(3));

O1C = T01C(1:3,4);
O2C = T02C(1:3,4);
O3C = T03C(1:3,4);
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%repulsive forces for left conveyor rail (LR) (Quadrant 2,3)

%O1
row0LR = 0.15; %region of influence for Left Rail

%treating rail as line segment, finding shortest distance to rail
PLR = [-0.38;0.7;0.275];
QLR = [-0.38;-0.8;0.275];

%Finding nearest point on line segment
dist = dot((PLR-QLR),(QLR-O1C))/dot((PLR-QLR),(PLR-QLR));
NP = QLR - dist*(PLR-QLR); %Nearest Point

%calculate distance of O1 to nearest point on left rail line segment
RowOfO1CLR = norm(O1C - NP);

%determine whether O1 falls in the region of influnce or not
if RowOfO1CLR <= row0LR
    GradOfO1CLR = (O1C - NP)/(norm(O1C-NP));
    FrepLeftRailO1 = aeta1*((1/RowOfO1CLR) -
(1/row0LR))*(1/RowOfO1CLR^2)*GradOfO1CLR;
```

```
else
    %if outside of region of influence set force to 0
    FrepLeftRailO1 = [0;0;0];
end

%-------------------------------------------------------------------------%
%Repeat calculation for repulsive force for O2

%Finding nearest point on line segment
dist = dot((PLR-QLR),(QLR-O2C))/dot((PLR-QLR),(PLR-QLR));
NP = QLR - dist*(PLR-QLR);

%calculate distance of O2 to nearest point on left rail line segment
RowOfO2CLR = norm(O2C - NP);

%determine whether O2 falls in the region of influnce or not
if RowOfO2CLR <= row0LR
    GradOfO2CLR = (O2C - NP)/(norm(O2C-NP));
    FrepLeftRailO2 = aeta1*((1/RowOfO2CLR) -
(1/row0LR))*(1/(RowOfO2CLR^2))*GradOfO2CLR;
else
    %if outside of region of influence set force to 0
    FrepLeftRailO2 = [0;0;0];
end

%-------------------------------------------------------------------------%
%Repeat calculation for repulsive force for O3

%Finding nearest point on line segment
dist = dot((PLR-QLR),(QLR-O3C))/dot((PLR-QLR),(PLR-QLR));
NP = QLR - dist*(PLR-QLR);

%calculate distance of O3 to nearest point on left rail line segment
RowOfO3CLR = norm(O3C - NP);

%determine whether O3 falls in the region of influnce or not
if RowOfO3CLR <= row0LR
    GradOfO3CLR = (O3C - NP)/(norm(O3C-NP));
    FrepLeftRailO3 = aeta1*((1/RowOfO3CLR) -
(1/row0LR))*(1/(RowOfO3CLR^2))*GradOfO3CLR;
else
    %if outside of region of influence set force to 0
    FrepLeftRailO3 = [0;0;0];
end

%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%Determine Repulsive forces for Right Rail (RR) (Quadrant 1,4)

%O1
row0RR = 0.15; %region of influence for Left Rail

%treating rail as line segment, finding shortest distance to right rail
P = [0.38;0.7;0.275];
Q = [0.38;-0.8;0.275];
```

```
%Finding nearest point on line segment
dist = dot((P-Q),(Q-O1C))/dot((P-Q),(P-Q));
NP = Q - dist*(P-Q);

%calculate distance of O1 to nearest point on right rail line segment
RowOfO1CRR = norm(O1C - NP);

%determine whether O1 falls in the region of influnce or not
if RowOfO1CRR <= row0RR
    GradOfO1CRR = (O1C - NP)/(norm(O1C-NP));
    FrepRightRailO1 = aeta2*((1/RowOfO1CRR) -
(1/row0RR))*(1/RowOfO1CRR^2)*GradOfO1CRR;
else
    %if outside of region of influence set force to 0
    FrepRightRailO1 = [0;0;0];
end

%-------------------------------------------------------------------------%
%Repeat calculation for repulsive force for O2

%Finding nearest point on line segment
dist = dot((P-Q),(Q-O2C))/dot((P-Q),(P-Q));
NP = Q - dist*(P-Q);

%calculate distance of O2 to nearest point on right rail line segment
RowOfO2CRR = norm(O2C - NP);

%determine whether O2 falls in the region of influnce or not
if RowOfO2CRR <= row0RR
    GradOfO2CRR = (O2C - NP)/(norm(O2C-NP));
    FrepRightRailO2 = aeta2*((1/RowOfO2CRR) -
(1/row0RR))*(1/(RowOfO2CRR^2))*GradOfO2CRR;
else
    %if outside of region of influence set force to 0
    FrepRightRailO2 = [0;0;0];
end

%-------------------------------------------------------------------------%
%Repeat calculation for repulsive force for O3

%Finding nearest point on line segment
dist = dot((P-Q),(Q-O3C))/dot((P-Q),(P-Q));
NP = Q - dist*(P-Q);

%calculate distance of O3 to nearest point on right rail line segment
RowOfO3CRR = norm(O3C - NP);

%determine whether O3 falls in the region of influnce or not
if RowOfO3CRR <= row0RR
    GradOfO3CRR = (O3C - NP)/(norm(O3C-NP));
    FrepRightRailO3 = aeta2*((1/RowOfO3CRR) -
(1/row0RR))*(1/(RowOfO3CRR^2))*GradOfO3CRR;
else
    %if outside of region of influence set force to 0
```

```
    FrepRightRailO3 = [0;0;0];
end

%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%Determine Repulsive forces for Structural Pole

%O1
row0P = 0.15; %region of influence for pole (taken from center of pole)

%treating pole as line segment through center of pole
P = [0;0.65;1];
Q = [0;0.65;-0.8];

%Finding nearest point on line segment
dist = dot((P-Q),(Q-O1C))/dot((P-Q),(P-Q));
NP = Q - dist*(P-Q);

%calculate distance of O1 to nearest point on pole line segment
RowOfO1CP = norm(O1C - NP);

%determine whether O1 falls in the region of influence or not
if RowOfO1CP <= row0P
    GradOfO1CP = (O1C - NP)/(norm(O1C-NP));
    FrepPoleO1 = aeta3*((1/RowOfO1CP) -
(1/row0P))*(1/RowOfO1CP^2)*GradOfO1CP;
else
    %if outside of region of influence set force to 0
    FrepPoleO1 = [0;0;0];
end

%-------------------------------------------------------------------------%
%Repeat calculation for repulsive force for O2

%Finding nearest point on line segment
dist = dot((P-Q),(Q-O2C))/dot((P-Q),(P-Q));
NP = Q - dist*(P-Q);

%calculate distance of O2 to nearest point on pole line segment
RowOfO2CP = norm(O2C - NP);

%determine whether O2 falls in the region of influnce or not
if RowOfO2CP <= row0P

    GradOfO2CP = (O2C - NP)/(norm(O2C-NP));
    FrepPoleO2 = aeta3*((1/RowOfO2CP) -
(1/row0P))*(1/RowOfO2CP^2)*GradOfO2CP;
else
    %if outside of region of influence set force to 0
    FrepPoleO2 = [0;0;0];
end

%-------------------------------------------------------------------------%
%Repeat calculation for repulsive force for O3
```

```
%Finding nearest point on line segment
dist = dot((P-Q),(Q-O3C))/dot((P-Q),(P-Q));
NP = Q - dist*(P-Q);

%calculate distance of O3 to nearest point on pole line segment
RowOfO3CP = norm(O3C - NP);

%determine whether O3 falls in the region of influnce or not
if RowOfO3CP <= row0P
    GradOfO3CP = (O3C - NP)/(norm(O3C-NP));
    FrepPoleO3 = aeta3*((1/RowOfO3CP) -
(1/row0P))*(1/(RowOfO3CP^2))*GradOfO3CP;
else
    %if outside of region of influence set force to 0
    FrepPoleO3 = [0;0;0];
end

%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%Sum the repulsive forces from each of the obstacles and return total
%repulsive forces on O1,O2,O3

FrepO1 = FrepLeftRailO1 + FrepRightRailO1 + FrepPoleO1;
FrepO2 = FrepLeftRailO2 + FrepRightRailO2 + FrepPoleO2;
FrepO3 = FrepLeftRailO3 + FrepRightRailO3 + FrepPoleO3;

end
```

Appendix 7 – jacobianscriptMOD.m:

```
function [JVO1, JVO2, JVO3] = jacobianscriptMOD(theta1,theta2,D3)
%function that takes joint variables(theta1, theta2, d3) input and outputs
%jacobian matrix

%-------------------------------------------------------------------------%
%inputs for DH parameters
L1 = 0.6;
L2 = 0.4;
L3 = 0.1;
L4 = 0.2;

a1 = L2;
a2 = L4;
a3 = 0;

d1 = L1;
d2 = L3;
d3 = D3;

alpha1 = 0;
```

```
alpha2 = 180;
alpha3 = 0;

t1 = theta1; %theta value
t2 = theta2;
t3 = 0;
%-------------------------------------------------------------------------%
%A matrix calculation

A1 = amatrixcalc(a1,d1,alpha1,t1);

A2 = amatrixcalc(a2,d2,alpha2,t2);

A3 = amatrixcalc(a3,d3,alpha3,t3);
%-------------------------------------------------------------------------%
%Intermediate Transformation Matrix Calculation
T1 = A1;
T2 = T1*A2;
T3 = T2*A3;

%-------------------------------------------------------------------------%
%Calculating Parameters needed for Jacobian Matrix
%(Z0,Z1,Z2,Z3,Z4,Z5,O0,O1,O2,O3,O4,O5,O6)

%extracting z column vectors from respective Tmatrix
Z0 = [0;0;1];
Z1 = T1(1:3,3);
Z2 = T2(1:3,3);

%extracting O vectors from respective Tmatrix
O0 = [0;0;0];
O1 = T1(1:3,4);
O2 = T2(1:3,4);
O3 = T3(1:3,4);

OHAT = [0;0;0];

%-------------------------------------------------------------------------%
%This section from HW2 not used in HW3
%Calculate J1,J2,J3
% J1 = [cross(Z0,O3-O0);Z0];
% J2 = [cross(Z1,O3-O1);Z1];
% J3 = [Z2;OHAT];

%-------------------------------------------------------------------------%
%calculate Jacobian matrices for each frame

JVO1 = [cross(Z0,O1-O0),OHAT,OHAT];
JVO2 = [cross(Z0,O2-O0),cross(Z1,O2-O1),OHAT];
JVO3 = [cross(Z0,O3-O0),cross(Z1,O3-O1),Z2];
%-------------------------------------------------------------------------%
%sub function used to calculate A matrix
function a_matrix_output = amatrixcalc(a,d,alpha,theta)
    a_matrix_output = [cosd(theta),-
sind(theta)*cosd(alpha),sind(theta)*sind(alpha),a*cosd(theta);
```

```
    sind(theta),cosd(theta)*cosd(alpha),-
cosd(theta)*sind(alpha),a*sind(theta);
    0,sind(alpha),cosd(alpha),d;
    0,0,0,1];
end

%-------------------------------------------------------------------------%
end
```

Appendix 8 – Path1 Configuration Summary (A = Θ_1, B = Θ_2, C = d3):

	A	B	C
1	45	0	0.2
2	47.95894	0.494669	0.2
3	50.91657	0.997063	0.2
4	53.87302	1.506368	0.2
5	56.8284	2.021864	0.2
6	59.78281	2.542911	0.2
7	62.73633	3.068935	0.2
24	72.33889	1.726067	0.2
25	69.49263	0.778035	0.2
26	72.44884	1.288727	0.2
27	69.60278	0.340069	0.2
28	72.55961	0.847154	0.2
29	69.71376	-0.10213	0.2
30	72.67122	0.401308	0.2
31	69.82558	-0.54859	0.2
185	87.12407	-51.0845	0.2
186	90.12314	-51.01	0.2
187	87.32978	-52.1042	0.2
188	90.32907	-52.0391	0.2
189	87.56288	-53.2002	0.2
190	90.56237	-53.1451	0.2
191	93.56142	-53.0693	0.2

241	115.5226	-88.5269	0.2
242	115.4032	-91.5246	0.2
243	118.4013	-91.6303	0.2
244	121.4002	-91.7114	0.2
245	124.3997	-91.768	0.2
246	127.3995	-91.8003	0.2
247	130.3995	-91.8083	0.2
248	133.3994	-91.792	0.2
249	136.3992	-91.7513	0.2

260	169.3147	-89.5903	0.2
261	168.5035	-86.702	0.2
262	171.2487	-85.4921	0.2
263	169.6824	-82.9334	0.2
264	168.2678	-80.2879	0.2
265	171.1105	-79.3291	0.2
266	169.9167	-76.5768	0.2

335	170.0997	16.15465	0.2
336	167.1416	15.65514	0.2
337	170.0741	16.28805	0.2
338	167.1161	15.78772	0.2
339	167.1161	15.78772	0.2
340			

Appendix 9 – Path2 Configuration Summary (A = Θ_1, B = Θ_2, C = d3):

	A	B	C
1	10	0	0.4
2	12.96728	0.441844	0.4
3	15.93049	0.910276	0.4
4	18.89034	1.399436	0.4
5	21.8474	1.905208	0.4
6	24.8021	2.424578	0.4
7	27.75479	2.955258	0.4
8	25.29393	2.079744	0.35081
9	28.2421	2.60177	0.357118

17	43.39212	4.977315	0.251481
18	46.31367	5.53128	0.264717
19	49.24027	6.090751	0.27636
20	49.3076	6.111349	0.176387
21	52.2046	6.667153	0.194598
22	55.11066	7.228266	0.210926
23	58.02366	7.793799	0.225632
24	60.942	8.363004	0.238935
25	63.86452	8.935246	0.25102

94	76.48265	-4.12023	0.391807
95	73.63887	-5.07568	0.391807
96	76.60266	-4.61132	0.392399
97	73.7591	-5.56745	0.392399
98	76.72352	-5.10709	0.392948
99	73.88019	-6.06389	0.392948
100	76.84525	-5.60757	0.393457
101	74.00214	-6.56504	0.393458

285	113.3366	-90.2217	0.399995
286	116.3345	-90.3332	0.399996
287	119.3333	-90.4202	0.399996
288	122.3326	-90.4829	0.399997
289	125.3324	-90.5214	0.399997
290	128.3323	-90.5359	0.399997
291	131.3323	-90.5263	0.399997

297	149.3216	-89.9589	0.399999
298	146.8277	-91.1063	0.359668
299	149.821	-90.9519	0.363935
300	152.8132	-90.7716	0.367912
301	151.1003	-91.6158	0.290789
302	154.0729	-91.4306	0.302758
303	157.0465	-91.2183	0.313957
304	156.2513	-91.6566	0.218647

322	169.7935	-82.1081	0.314342
323	169.8344	-82.0777	0.214356
324	168.8349	-80.8947	0.3
325	168.8701	-80.8693	0.200011
326	169.9828	-79.7606	0.285207
327	170.0671	-79.6966	0.18527

360	175.5323	-46.5357	0.31548
361	175.3466	-44.8731	0.398488
362	175.7246	-41.8983	0.4
363	176.1828	-38.9335	4.00E-01
364	176.7112	-35.9804	0.4
365	177.3012	-33.039	0.4
366	177.9449	-30.1089	0.4
367	178.6349	-27.1893	0.4

431	169.8959	16.13841	0.4
432	166.9372	15.64195	0.4
433	169.8794	16.22803	0.4
434	169.8794	16.22803	0.4
435			

Appendix 10 – Path3 Configuration Summary (A = Θ_1, B = Θ_2, C = d3):

	A	B	C
1	10	0	0.4
2	8.067727	-0.64471	0.326586
3	7.543188	-0.81043	0.228281
4	7.312558	-0.88491	0.128608
5	9.734577	-0.57401	0.1867
6	12.40266	-0.20177	0.230705
7	12.00901	-0.33355	0.131667
8	14.61523	0.047661	0.179536
9	17.36731	0.472026	0.216743
10	16.88646	0.306704	0.11819
11	19.57385	0.732179	0.160313
12	22.35838	1.189296	0.194264

33	66.54649	9.078443	0.14631
34	69.44286	9.642499	0.164347
35	66.59288	8.705722	0.164437
36	69.49594	9.268758	0.181276
37	66.64636	8.330749	0.181375
38	67.88467	8.731374	0.091276
39	70.76221	9.285536	0.112689
40	67.91106	8.352299	0.112715
41	70.79731	8.906123	0.132793
42	67.9465	7.971818	0.132819
43	70.84063	8.524987	0.151618
44	67.99016	7.589664	0.151643

281	90.9234	-59.2332	0.152391
282	93.87663	-59.2182	0.169973
283	91.12086	-60.4038	0.170072
284	94.07992	-60.3994	0.186535

297	97.51249	-67.9375	0.192587
298	94.82856	-69.2779	0.192783
299	97.79316	-69.3364	0.207974
300	95.21059	-70.8615	0.210171
301	98.17998	-70.9334	0.224216
302	101.154	-70.9824	0.237242
303	98.57428	-72.5138	0.237271
304	101.5514	-72.5757	0.249422
305	98.98893	-74.1357	0.249458
306	101.9687	-74.2107	0.260783
307	99.426	-75.8027	0.260832
308	102.408	-75.891	0.27138

345	168.3858	-91.0212	0.240861
346	170.6574	-90.1422	0.299239
347	170.543	-90.2277	0.199353
348	169.3193	-89.168	0.283546
349	169.1973	-89.255	0.18367
350	168.9216	-88.165	0.276383
351	168.9077	-88.131	0.17639
352	169.6708	-87.0935	0.266705

469	169.896	16.08202	0.4
470	166.9372	15.58624	0.4
471	169.8794	16.17241	0.4
472	169.8794	16.17241	0.4
473			

Appendix 11 – Path4 Configuration Summary (A = Θ_1, B = Θ_2, C = d3):

	A	B	C
1	10	0	0.4
2	12.99253	0.21153	0.4
3	15.98422	0.434697	0.4
4	18.97521	0.667016	0.4
5	21.96562	0.90673	0.4
6	24.95553	1.152541	0.4
7	27.94502	1.403466	0.4
8	25.48189	0.547715	0.350551
9	28.47043	0.791688	0.353725
10	31.45894	1.040477	0.356494
14	38.88938	1.273749	0.291787
15	41.87409	1.53082	0.2971
16	41.28839	1.331673	0.199249
17	44.26413	1.587957	0.208636
18	47.24173	1.847097	0.217246
19	50.22078	2.108769	0.225189
20	53.20097	2.372694	0.232555
112	83.38234	-27.3213	0.374235
113	80.55058	-28.3118	0.374235
114	83.54746	-28.1778	0.375152
115	80.71608	-29.1695	0.375153
116	83.71314	-29.0394	0.376037
117	80.88217	-30.0322	0.376037
118	83.87941	-29.9061	0.37689
119	81.04884	-30.9	0.37689
120	84.04626	-30.7779	0.377712
121	81.21612	-31.773	0.377713
122	84.2137	-31.655	0.378505
123	81.38401	-32.6514	0.378506
124	84.38176	-32.5374	0.37927
125	81.55253	-33.5351	0.37927
161	90.7476	-49.8635	0.389614
162	87.95395	-50.9569	0.389614
163	90.95368	-50.9176	0.389982
164	88.16194	-52.0159	0.389982
165	91.16172	-51.9814	0.390338
166	88.37201	-53.0848	0.390338

524	170.0277	15.88695	0.4
525	167.0387	15.63037	0.4
526	170.0184	15.97847	0.4
527	167.0294	15.72137	0.4
528	167.0294	15.72137	0.4
529			