UNIVERSITY OF MARYLAND

ENPM662(Robot Modelling)

Formal Report

Project

Spot Welding Robot

Author:
Akwasi A. Obeng

UID:
117000801

December 08, 2019

Contents

1 Motivation

Automated Welding is a practical necessity in companies today as it is much safer, cheaper and more robust. Different companies from Automobile to Delivery make of use welding robots and this has resulted in a boost in productivity.The surge in the development of welding robots not only lies in the safety it offers but rather in the precision and accuracy with which it is achieved.In the automation industry, welding robots are used for a variety of tasks, from resistance welding (join metal surfaces together through the application heat by resistance of electric current) to arc welding. Refer to `https://en.wikipedia.org/wiki/Arc_welding` for more information. The type of welding to be looked at is **Spot welding** and is a type of resistance welding which involves joining two thin metals that resist electric currents.

, 

Figure 1: Welding in Automotive industry

Welding Robots are designed to perform a specific task and in other for this to be achieved, the control system, sensing system and the different components of the robot must function well to achieve some desired result. Precision is of the essence as the robot is supposed to join(weld) two spots together at the some **specified location**, with the right **amount of force** whilst operating at some **optimum speed** for productivity.Should the robot be off in its measurement by even a few centimeters,then this could be detrimental to the whole operation. **Precision is of paramount importance**.This project looks at how this can be achieved.

2 Introduction

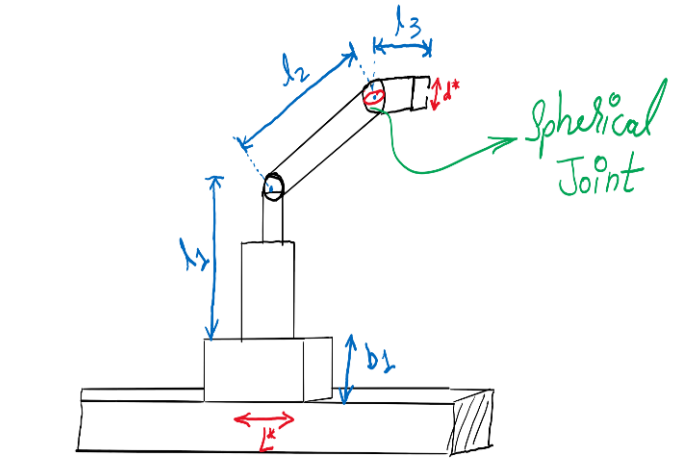
The focus of this project is on how a welding robot can be modelled with precision to achieve some desired task. The robot to be simulated should in principle be able to join metal plates together. For brevity, a specialized robotic arm (robotic arm modified to meet required task) is used. That is given some metal plates, the robot should efficiently navigate its environment, handle appropriate weights and operate within some required standards.

The robot to be simulated must satisfy the following criteria

- Have a good workspace range
- Have tapering ends for piercing metal sheets
- Must be able to position itself to weld some specified locations

3 Model Design

Based on the requirements, the design below was created



,

Figure 2: Robot Design

Robot Attributes

- 6 Degrees of Freedom(DOF) not including the end effector
- 1 prismatic joint(Rack and pinion) at the base and 5 revolute joints

- Link lengths $l1 > l2 > l3$
- End effector has tapering ends for piercing effect

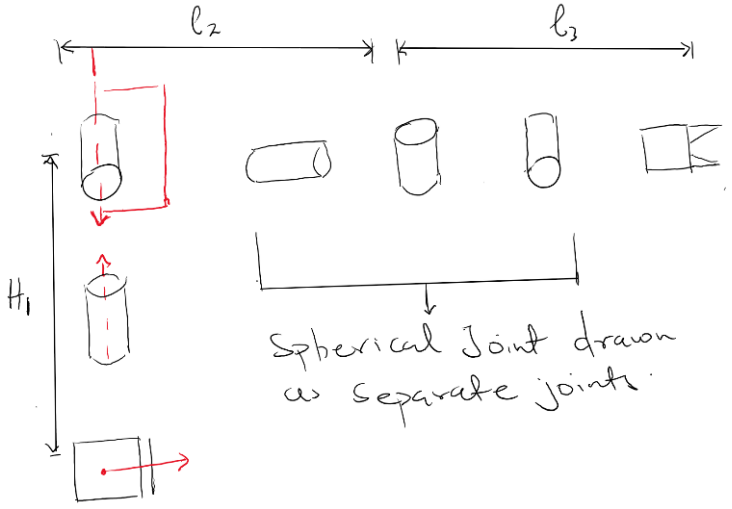


Figure 3: Joints of Robot

3.1 workspace

Due to the complexity of the workspace, the projected workpace is used.

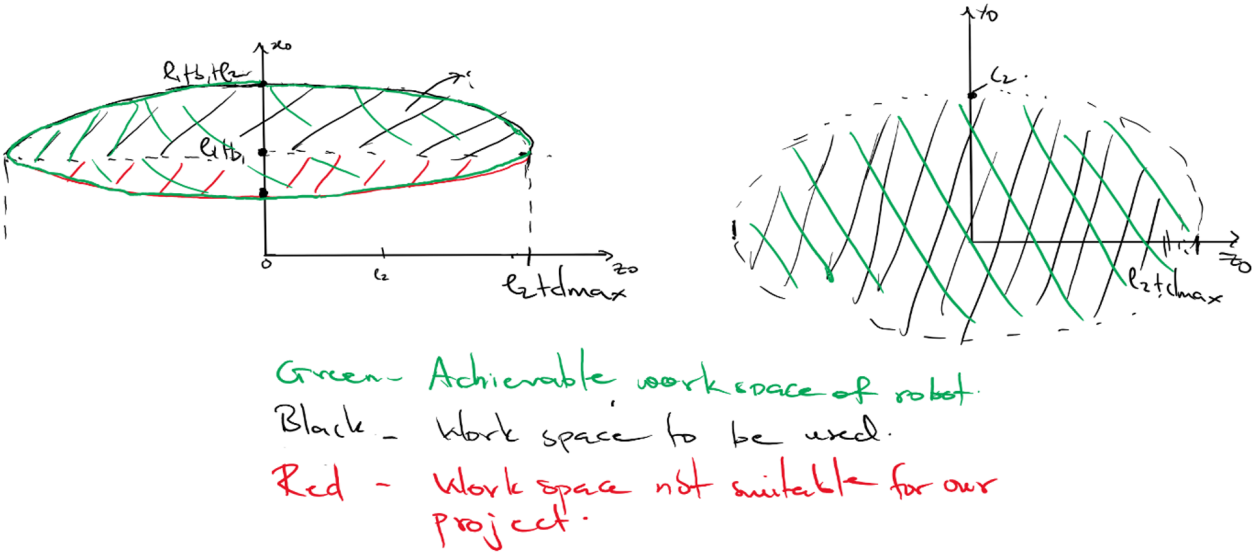


Figure 4: Workspace

4 Kinematics

In this section, the forward and inverse kinematics calculations are performed for the robot described above.

4.1 Foward Kinematics

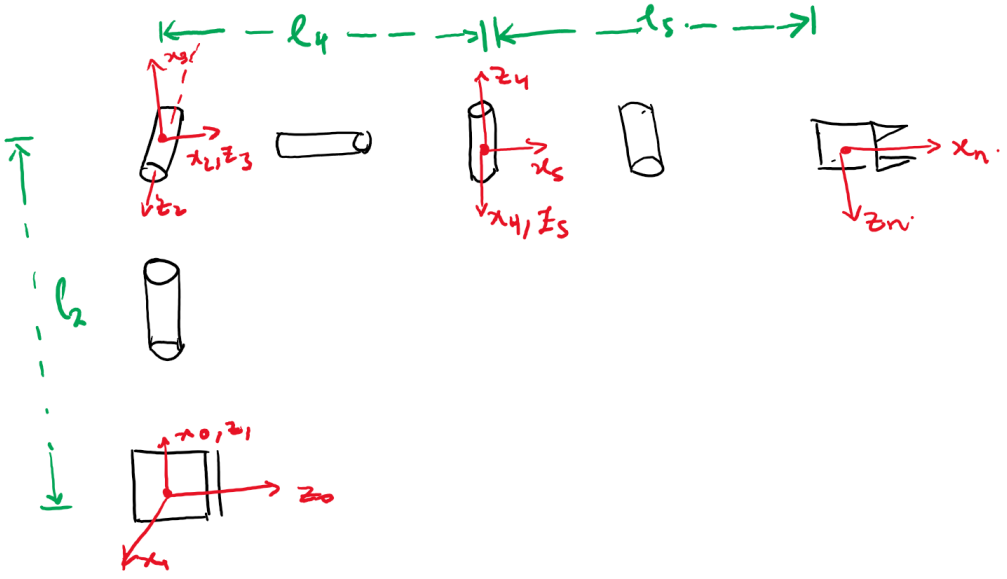


Figure 5: Assigning DH Frames

DH Table				
T	θ	d	a	α
$0->1$	90^0	d_1^*	0	90^0
$1->2$	$\theta_1^* + 90^0$	l_2	0	90^0
$2->3$	$\theta_2^* + 90^0$	0	0	90^0
$3->4$	$\theta_3^* + 90^0$	l_4	0	90^0
$4->5$	$\theta_3^* + 90^0$	0	0	90^0
$5->n$	0	0	l_5	0

DH calculations

The DH table is represented as follows

$$T_0^n = \begin{bmatrix} c_4s_2 + c_2s_3s_4 & c_2c_3 & s_2s_4 - c_2c_4s_3 & l_2 + l_5(c_4s_2 + c_2s_3s_4) + l_4s_2 \\ -s_4c_1c_3 - s_1s_2s_3 - c_2c_4s_1 & c_1s_3 + c_3s_1s_2 & c_4(c_1c_3 - s_1s_2s_3) - c_2s_1s_4 & -l_5(s_4(c_1c_3 - s_1s_2s_3) + c_2c_4s_1) - l_4c_2s_1 \\ c_1c_2c_4 - s_4(c_3s_1 + c_1s_2s_3) & s_1s_3 - c_1c_3s_2 & c_4(c_3s_1 + c_1s_2s_3) + c_1c_2s_4 & d_1 - l_5(s_4(c_3s_1 + c_1s_2s_3) - c_1c_2c_4) + l_4c_1c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Refer to the Appendix(Code for DH Table) for the code used in generating the dh table.

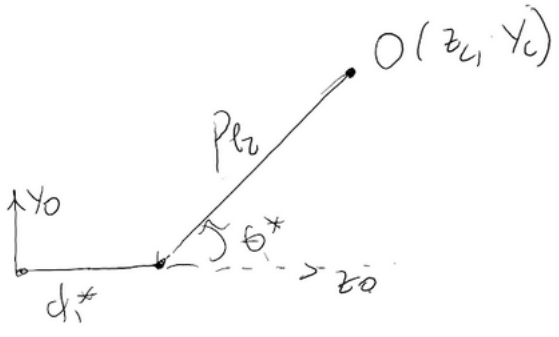
4.2 Inverse Kinematics

Since robot has six degree freedom and a spherical wrist, the problem is decoupled into two parts, that is

- Finding Wrist center
- Orientation of Wrist

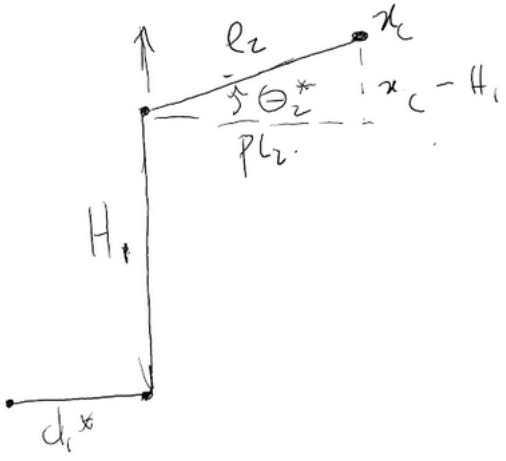
Finding Wrist Center Suppose the wrist center is located at the coordinates X_c, Y_c, Z_c the required joint angle parameters are obtained as follows

Top View



(a) Inverse kinematics

Side View



(b) Inverse kinematics

$$\theta_2^* = \sin^{-1}(\frac{Y_c}{Pl_2}) \quad (1)$$

$$d_1^* = Z_c - Pl_2 cos(\theta_1^*) \quad (2)$$

$$\theta_3^* = \sin^{-1}(\frac{X_c - H}{l_2}) \quad (3)$$

$$Pl_2 = l_2 cos(\theta_3^* h) \quad (4)$$

Therefore,

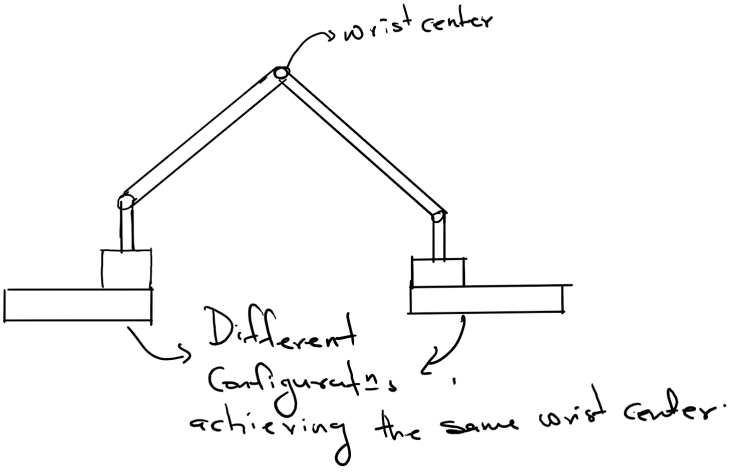
$$\theta_2^* = sin^{-1}(\frac{Y_c}{Pl_2}) \tag{5}$$

$$\theta_3^* = \sin^{-1}(\frac{X_c - H}{l_2}) \tag{6}$$

$$d_1^* = Z_c - l_2 * cos(h(\sin^{-1}(\frac{X_c - H}{l_2})))cos(sin^{-1}(\frac{Y_c}{Pl_2})) \tag{7}$$

The values of X_c, Y_c and Z_c can be gotten from O_4(Refer to Appendix(Code for Jacobian)).
NB Since there is a prismatic joint at the base, it is possible to get 2 or more solutions for the wrist center. To visualize this consider the case shown below

Multiple Solutions



Orientation of Wrist

The actual calculations needed to get the angles for orienting the sperical wrist is not provided but rather a cursory overview on how this can be achieved. The orientation of the wrist can be gotten as follows

$$R_6^0 = R_3^0 R_6^3 \tag{8}$$

$$R_6^3 = R_0^3 R_6^0 \tag{9}$$

R_0^3 is known from previous calcuations and R_6^0 is known from the DH table.

Representing R_6^3 as a zyz transformation, the following angles θ, ϕ, γ can be obtained by comparing matrix with the product of $R_0^3 R_6^0$ and solving for the angles.

NB. Infinite number of solutions when the robot is in an upright position

5 Piercing, and Singularity

The amount of force needed at the endeffector(gripper) to pierce the metal sheet is material dependent. Therefore, only the general form of the wrench is obtained.

$$T = J^T(q)F \tag{10}$$

where T is the joint torque,F is the endeffector wrench and J is the jacobian. Refer to **Appendix(Code for Jacobian)** for calculation of Jacobian

Singularity
Since the robot has a spherical joint, it has a singularity(wrist flip) when the robot is in an upright(projected) position. To get all the values of q for which the Jacobian has a singular configuration, we find

$$Det(J) = 0 \tag{11}$$

The calculation of the determinant of J is quite cumbersome since J is a 6x6 matrix. The calculation can be decoupled into two components, ie

$$Det(J_{3:6}^{1:3}) = 0 \tag{12}$$
$$Det(J_{1:3}^{3:6}) = 0 \tag{13}$$

That is the determinant of the upper right and lower left blocks of the jacobian can be calculated and the values of q for which a singularity occurs can then be obtained. However, for the purporse of this project, all the possible values for which a singular configuration occurs is not of interest. We only need to check if there is some singularity along some predetermined path and this can be done by careful inspection.

6 Simulation

For convenience of simulation, the original design is modified to yield the figure shown below.

Modified Design

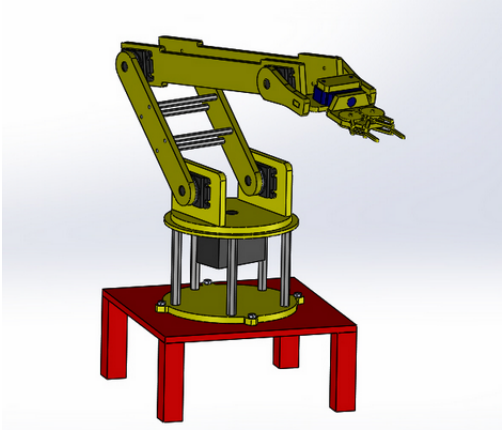


Figure 7: CAD Model

Clearly the new model above differs from the original design. However, this new model can be considered as a restricted version of the old design. Futher emphasis is provided in the next section.

6.1 Constraints and Design Modifications

- **Modified base**
  Instead of a movable base, a fixed base is used in the simulation.

Changing Base

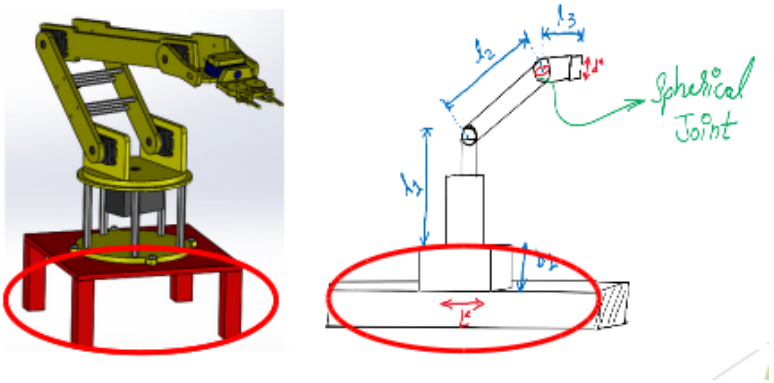


Figure 8: Modified Base

This can however be viewed as the original design constrained to some fixed location. That is

$$d_1^* = constant(c) \quad - \quad \text{Holonomic Constraint} \tag{14}$$

- **Modified Spherical Joint**
  Instead of a spherical joint at the wrist, only 2 revolute joints are used

Changing Spherical joint

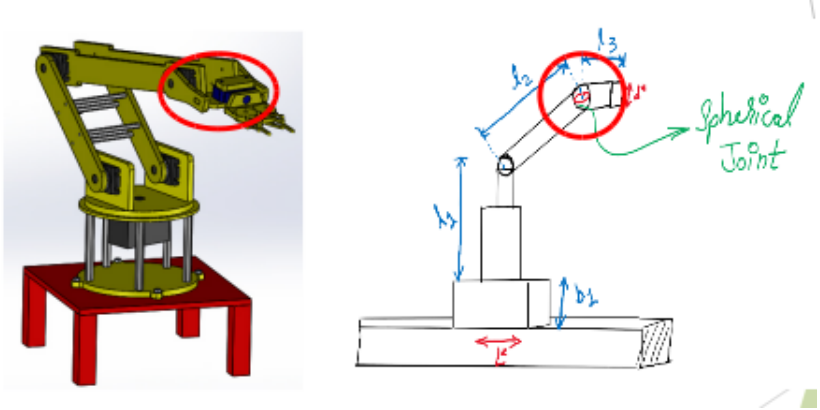


Figure 9: Modified Spherical joint

This can be viewed as the original design with the constraint as follows

$$\theta_5^* = constant(c) \quad - \quad \text{Holonomic Constraint} \tag{15}$$

- **Increase Workspace Range**
  To increase workspace range, of the robot, one additional link and revolute joint is added

Changing Spherical joint

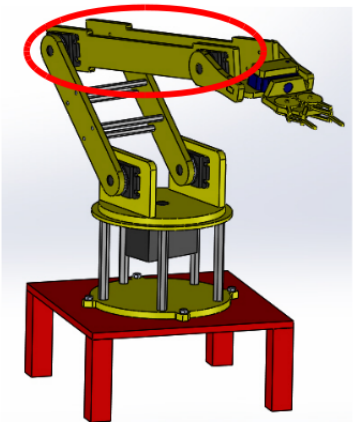


Figure 10: Added link and revolute joint

Changing Endeffector
For ease of simulation, an inbuilt Baxter gripper is used.

6.2 Assumptions

The following assumptions were made for simulation

- Singularity - The robot is never in a singular configuration.
- Dynamical Forces -Friction and other dynamical forces associated with the moving object are not accounted for
- Structure- The robot will operate within some structured environment.Objects(metal sheets) to be welded together are placed on a **conveyor belt** moving at a **constant speed**. The conveyor belt is equipped with sensors to determine when an object is present and enough time is given to the robot to weld metal plates together.

- Conveyor belt does not accelerate when already in motion
- There is enough gap between adjacent metal sheets been transported
- Metal sheet placed on conveyor belt is within reacheable workspace of robot

6.3 Scope of Simulation

- Forward and Inverse kinematics - This was accounted for in the simulation
- Contact and Piercing- This was partially accounted for in the simulation. The baxter gripper used does not have a piercing end and so although the calculations were made, the simulation didn't include this effect
- Dynamics - Not Simulated. The (Inverse/Forward) mode was used in vrep and not the (torque/dynamics) mode.
- Constraints - This is accounted for by the modified model used in the simulation

7 Vrep Scene

Components used for scene in Vrep. **Refer to Appendix(Simulation Components for Vrep)** for the full scene and code.

- Robot - The robot utilized is the cad Model(welding robot) that was descibed above.
- Conveyor Belt - This is used to transport the objects(metal sheets) to be welded together
- Sensors - These are used to indicate whether an object(metal sheet to be welded) is present and also to determine whether the object has reached the end of the the conveyor belt.
- Metal Sheet - Metal sheets are depicted as a plane sandwiched between two blocks.

Simulation Enviroment

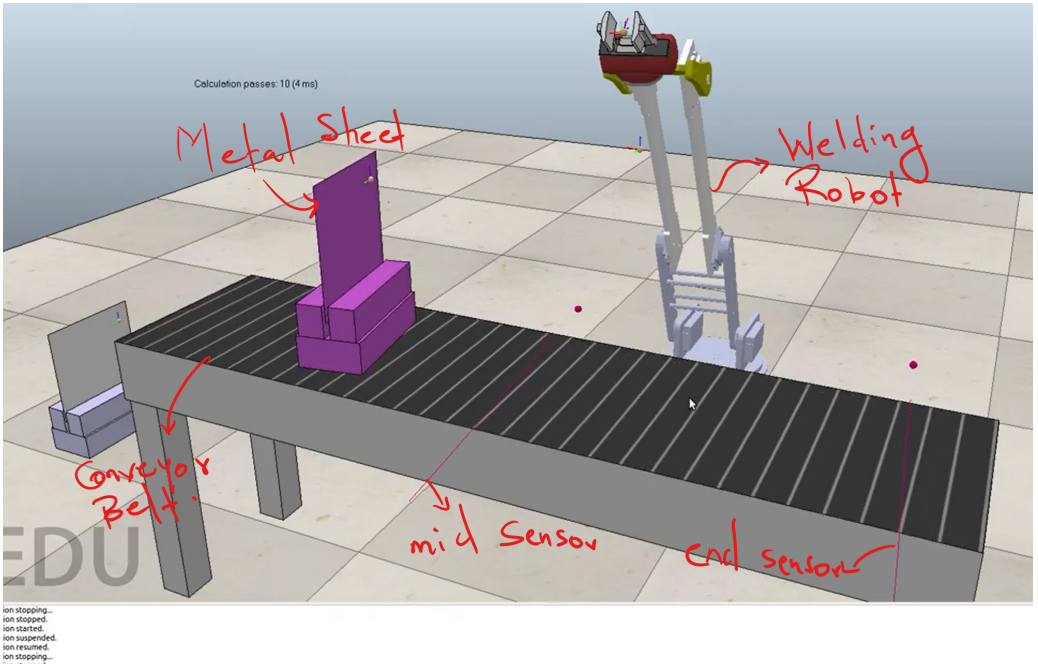


Figure 11: Components displayed in Simulation Environment

7.1 How the Code Works

A brief overview on how the code works is given below.

The code can be broken down into two key components

Conveyor Belt – Non Threaded

1. Continue to transport object on the conveyor belt
2. If an item is sensed by mid sensor(Sensor located at the middle of the conveyor belt)
   (a) Stop Conveyor belt
   (b) Wait for object to be processed(metal sheet to be welded)
      i. If item has been processed, start the conveyor and continue transporting objects
3. If an item is sensed at the end sensor(Sensor located at the end of the convyeor belt)

(a) Delete Object

Welding Robot – Threaded

1. Wait for an object to be sensed by the sensor in the middle of the conveyor belt
   (a) If an object is sensed, follow a pickup path(path to object)
   (b) Weld the object(metal sheet)
   (c) Send a signal to indicate object has been welded
   (d) Follow a release path(path to return the object to its initial position)
   (e) Send a signal to indicate that it is okay for the conveyor belt to resume transporting objects
   (f) Wait for an object to detected again and repeat the process

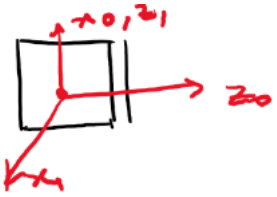
8 Testing and Validation

Visit the following url to view the simulation
`https://www.youtube.com/watch?v=xeCUFldfl0U&feature=youtu.be`
`https://www.youtube.com/watch?v=jMd6bSmziDo&feature=youtu.be`

Testing Code and Parameters Used in simulation
Based on the simulation videos, it can be seen that the welding robot is able to position itself in a suitable orientation for welding metal sheets even when the metal sheet is slightly tilted. However, the piercing effect of the gripper is not properly done and this is partly due to the fact that the baxter gripper is not appropriate for application at hand.

Forward and Inverse Verification
Forward Kinematics

Base Frame



(a) Frame of Base

EndEffector Frame

(b) Frame of EndEffector

Output Checking if the rotation matrix needed to transform the orientation of Base frame matches what we expect for EndEffector frame when robot has not been rotated. Clearly this implies the

- x_n must point in the z_0 direction
- z_n must point in the y_0 direction
- y_n must point in the x_0 direction

Base Frame

```
ak@ubuntu16:~$ python3 Desktop/proj.py
[[-0.  1.  0.  0.]
 [ 0. -0.  1.  0.]
 [ 1.  0. -0.  0.]
 [ 0.  0.  0.  1.]]
ak@ubuntu16:~$
```

Figure 13: Frame of Base

Clearly this matches with the matrix obtained. Refer to **Appendix Section(Code for Forward Kinematics Test)**

Inverse Kinematics Calculations

The verification of the inverse kinematics was done by manually inputing values and checking to see if the results match what was expected. This is not the ideal way to verify inverse kinematics calculations. Trying to calculations to vrep was not trivial as it was difficult to track frames.

9 Results

The results of the simulation can be listed as follows:

1. Simulation of Forward and Inverse Kinematics - **Successful**
   (a) Robot is able to get to the metal sheet - **Successful**
   (b) No singular configurations encounted from robot movements - **Successful**
   (c) Robot is able to get to initial configuration - **Successful**
   (d) Robot is able to follow some specified path - **Successful**
2. Simulation of Gripper - **partially successful**
   (a) Endeffector is placed at right location to weld metal sheets -**Successful**
   (b) Endeffector pierces(close and open)metal sheet - **Not successful**
3. Object dynamics - **Not successful**
   (a) Metal sheet dynamics(force needed to pierce metal sheet) -**Not simulated**
   (b) Included robot dynamics in simulation-**Not simulated**(Most of the dynamics of the welding robot was turned off)
4. Mimicking Real World Scenario - **Partially successful**
   (a) Introducing probabilitic effects in the goods produced, that is defects in production -**Not simulated**
   (b) Making the environment more elaborate -**Partially successful**(Simulation environment could be better)
5. Testing and Verifying - **Partially successful**
   (a) Trying to match calculations with vrep calculations - **Partially successful** (In order to match the calculations outline above together vrep's, the frames utilized in vrep was hard to track. Need to learn how to use vrep and obtain such information)

10 Conclusion and Future Work

Based on the simulation, it could be noted that the endeffector is able to get to the desired location to weld the metalsheets. However, the gripper(baxter gripper) used

for the project is not appropriate for the application at hand. Future simulations would rectify this by using a proper cad model for the endeffector(with a piering effect) including necessary code and calcuations for the visualization effect. Also, the dynamics of the robot model can be included to make the model more realistic.

11 Appendix

This section has all the necessary codes used in calcuations and simulation

11.1 Code for DH Table

```
%Matlab Code
syms alpha theta a d

A(theta,d,a,alpha)= [cos(theta), -sin(theta)*cos(alpha) ...
                     sin(theta)*sin(alpha), a*cos(theta);
                     sin(theta),  cos(theta)*cos(alpha), ...
                     -cos(theta)*sin(alpha), a*sin(theta);
                     0 sin(alpha) cos(alpha) d;
                     0 0 0 1];

syms d1 theta1 theta2 theta3 theta4 l2 l4 l5
A_0_1 = simplify(A(pi/2,d1,0,pi/2));
A_1_2 = simplify(A(theta1+pi/2, l2, 0, pi/2));
A_2_3 = simplify(A(theta2+pi/2, 0 , 0 ,pi/2));
A_3_4 = simplify(A(theta3+pi/2, l4, 0 ,pi/2));
A_4_5 = simplify(A(theta4+pi/2, 0, 0, pi/2));
A_5_n = simplify(A(0, 0, l5, 0));

T_0_n=A_0_1*A_1_2*A_2_3*A_3_4*A_4_5*A_5_n;

simplify(T_0_n)
```

11.2 Code for Jacobian

```
%Matlab Code
syms alpha theta a d

A(theta,d,a,alpha)= [cos(theta), -sin(theta)*cos(alpha) ...
                     sin(theta)*sin(alpha), a*cos(theta);
                     sin(theta),  cos(theta)*cos(alpha), ...
                     -cos(theta)*sin(alpha), a*sin(theta);
                     0 sin(alpha) cos(alpha) d;
                     0 0 0 1];
```

```
10
11 syms d1 theta1 theta2 theta3 theta4 l2 l4 l5
12 A_0_1 = simplify(A(pi/2,d1,0,pi/2));
13 A_1_2 = simplify(A(theta1+pi/2, l2, 0, pi/2));
14 A_2_3 = simplify(A(theta2+pi/2, 0 , 0 ,pi/2));
15 A_3_4 = simplify(A(theta3+pi/2, l4, 0 ,pi/2));
16 A_4_5 = simplify(A(theta4+pi/2, 0, 0, pi/2));
17 A_5_n = simplify(A(0, 0, l5, 0));
18
19 T_0_1 = A_0_1;
20 T_0_2 = T_0_1*A_1_2;
21 T_0_3 = T_0_2*A_2_3;
22 T_0_4 = T_0_3*A_3_4;
23 T_0_5 = T_0_4*A_4_5;
24 T_0_n = T_0_5*A_5_n;
25
26 z0 = [0 0 1]';
27 z1 = T_0_1(1:3,3);
28 z2 = T_0_2(1:3,3);
29 z3 = T_0_3(1:3,3);
30 z4 = T_0_4(1:3,3);
31 z5 = T_0_5(1:3,3);
32 zn = T_0_n(1:3,3);
33
34
35 O0 = [0 0 0]';
36 O1 = T_0_1(1:3,4);
37 O2 = T_0_2(1:3,4);
38 O3 = T_0_3(1:3,4);
39 O4 = T_0_4(1:3,4);
40 O5 = T_0_5(1:3,4);
41 On = T_0_n(1:3,4);
42
43 J = [z0 cross(z1,On-O1) cross(z2,On-O2) ...
44      cross(z3,On-O3) cross(z4,On-O4) cross(z5,On-O5);
45       O0 z1 z2 z3 z4 z5;
46      ];
47
48 simplify(J)
```

11.3 Simulation components used in Vrep

Refer to `https://github.com/mesneym/RobotModelling` for the scene, code , and URDF files, used in vrep

11.4 Code for Forward Kinematics Test

```
import numpy as np

def transZ(n):
    a = np.identity(4)
    return a

def transX(n):
    a=np.identity(4)
    a[0,3]=n
    return a

def transY(n):
    a=np.identity(4)
    a[1,3]=n
    return a

def rotX(n):
    x=np.deg2rad(n)
    a=np.array([[1,0,0,0],
                [0,np.cos(x),-np.sin(x),0],
                [0,np.sin(x),np.cos(x),0 ],
                [0, 0,0,1]
               ])
    return a

def rotY(n):
    x=np.deg2rad(n)
    a=np.array([[np.cos(x),0,np.sin(x),0],
                [0,1,0,0],
                [-np.sin(x),0,np.cos(x),0 ],
                [0, 0,0,1]
               ])
    return a
```

```
34
35 def rotZ(n):
36     x=np.deg2rad(n)
37     a=np.array([[np.cos(x),-np.sin(x),0,0],
38                 [np.sin(x),np.cos(x),0,0],
39                 [0,0,1,0 ],
40                 [0,0,0,1]
41                ])
42     return a
43
44
45 def A_dh(theta,d,a,alpha):
46    return  np.dot(rotZ(theta),
47            np.dot(transZ(d),
48            np.dot(transX(a),
49            rotX(alpha))))
50
51
52
53 #Question Project
54 ##########################################################################
55 ##
56 ##########################################################################

57 q = [0, 0, 0, 0, 0]
58 l2 = l4 = 2
59 A1=A_dh(90,q[0],0,90)
60 A2=A_dh(q[1]+90,l2,0,90)
61 A3=A_dh(q[2]+90,0,0,90)
62 A4=A_dh(q[3]+90,l4,0,90)
63 A5=A_dh(q[4]+90,0,0,90)
64
65 T0_1 = A1
66 T0_2 = np.dot(T0_1,A2)
67 T0_3 = np.dot(T0_2,A3)
68 T0_4 = np.dot(T0_3,A4)
69 T0_5 = np.dot(T0_4,A5)
70
71 print(np.round(T0_5,3))
```

12 References

[1] https://www.youtube.com/watch?v=N5AYZxsnDuM

[2] https://en.wikipedia.org/wiki/Arc_welding

[3] https://en.wikipedia.org/wiki/Robot_welding