UNIVERSITY OF MARYLAND,
COLLEGE PARK

Cable Driven Parallel Robot Manipulator for Pick and Place Applications

Done by - Saumil Shah - 116745338
Partner - Mahmoud Dahman
ENPM 662

December 14, 2019

Abstract

This report aims to describe the cable based parallel robots, which can help fill the gaps in larger autonomous warehouse managements. Inverse kinematics, quasi static analysis and gripping has been discussed in this report. All the theoretical work has been carried out and results are validated with the simulations. All the details about simulating this robot in gazebo software has been provided. Scope of this project is limited as of now, but it will surely help in building the real system.

Key words: Parallel robots, Cable based parallel robots, CDPR, Forward kinematic, Inverse kinematics, Velocity kinematics, Degree of Freedom(dof)

Contents

List of Figures

List of Tables

1 Introduction and Motivation

With increasing demand of automation in warehouses and mass manufacturing facilities, more and more robots are being used to do several tasks in such environments. These tasks include moving big packages or heavy loads in metal industry. In metal or manufacturing industry,usually hand type serial manipulators are used on larger scale. Because of their smaller workspace, number of such robots have to be incorporated. For moving packages, serial manipulators are being used with automatic guided vehicles which again complicates the whole process.

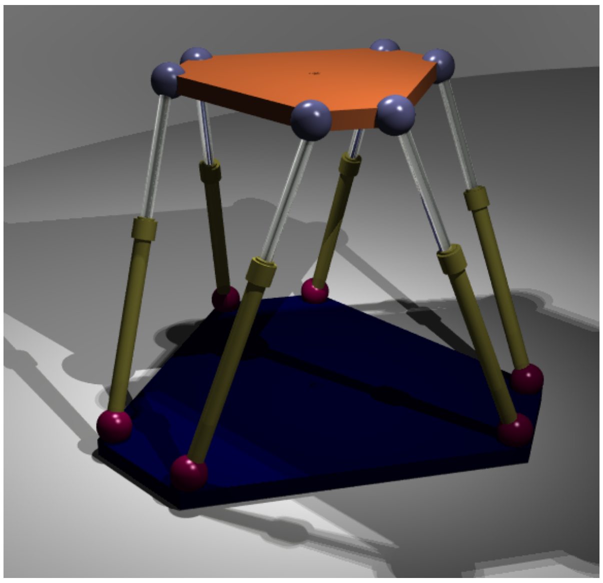


Figure 1.1: 6 DOF type Stewart-Gough platform

Parallel manipulators have some good advantages over serial manipulators because of their closed loop structures. Because of quick responses and good accuracy they are heavily used in medical robotics. They can carry high loads because of closed kinematic structures, which makes them a good candidate for industrial applications. 6 degree of freedom type Stewart gough platform, as shown in Figure 1.1, is the most common kind of parallel manipulator which can achieve six degrees of freedom. This platform is normally used in flight simulators or tire testing and their applications are not limited to only these tasks.

Cable based parallel manipulators are a modified version of parallel manipulators where prismatic joints are replaced with cables under tension. These cables give them an extra edge because of less complex structure which can be easily scaled. Though cables have their own disadvantages because they can pull but cannot push the end-effector, but this problem can be resolved with the inclusion of redundant joints.

2 Robot Description

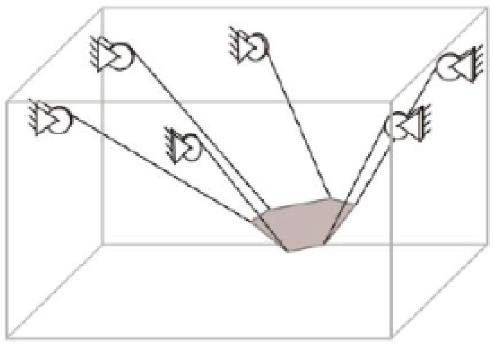


Figure 2.1: Robot

A cable driven parallel robot (CDPR) can be constructed with different architectures. Essentially an end effector should be connected with cables and other ends of cables should be connected to the actuator-which are usually servo motors. Cables pass through a fixed point which is a pulley in our case, over which cables bend and go to the servo motors. Joint between cables and end effector is usually taken as a ball joint. While the joint between a fixed point and cables is taken as universal joint, because of which pulley should have a degree of freedom and should be connected to a revolute joint rather than a fixed joint. Length of cables between this fixed point and end effector is changed in order to move the end effector in the workspace.

Figure 1 contains a simple 6 cable CDPR which can achieve 6 degrees of freedom, but it is not well suited for the task because of cables. Cables can only pull the object but cannot push,which will make end effector vibration prone and slow. To make the system more stable, two more cables are introduced, which will increase the efficacy of task performance.

We will use the architecture as shown in figure 3 as basis of this project. Workspace will be a square warehouse of size $3.5m \times 3.5m \times 3.5m$, each upper corner being the fix point for the joint.Because of square workspace each corner will lie on the same circle, and this symmetry will make the mathematics behind the problem somewhat easier to follow. End effector will be a square box of size $0.6m \times 0.6m \times 0.6m$ and at all the corners of the box are considered as a ball joint which establish connections with cables.

2.1 Assumptions

It is an enormous task to build the entire robotic system and requires lots of small considerations. Some assumptions are required to be taken in order to achieve the goal of this project, which are mentioned below.

- The entire warehouse is assumed to be rigid.
- The robot utilizes its parallel gripper to grasp a metallic cubic object that is rigid in nature and cannot be deformed. Position of the object is known and no sensing elements is present which can sense the location of the object.
- All the cables are massless, rigid and no collision is taking place between cables and between cables and other environment object available.
- All the spherical and universal joints does not have any joint limitations and all the joints are massless and frictionless.
- Motors can generate any required amount torque to grip and move the object.
- The workspace is obstacle-free which excludes the task of path planning.

3 Robot Modelling

3.1 Inverse Position Kinematics

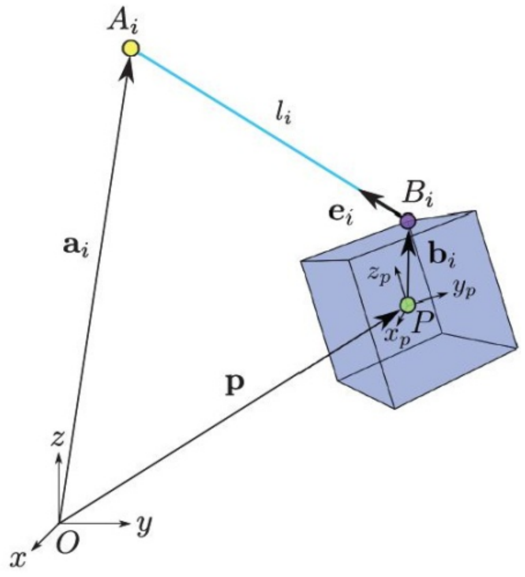


Figure 3.1: Geometric description of the robot

To move out end effector in the workspace and to position it at certain defined state, we change the length of cables and it is necessary to have an idea about what length should we keep to reach a particular state. World frame is assumed to be at the at the centre of the warehouse, but at the ground level. Let's assume that O_d is the new position of the end-effector in the world coordinate frame and θ_x, θ_y and θ_Z are the XYZ euler angles of the new position. A_is are the coordinates of the outer frames from where cables pass through to get to motors. This joints will not move and distance between these joints and end-effector joints will actually give the length of the cable, which we are trying to compute. b_is are the positions of joints in the end-effector frame, on the end-effector, where the cables are connected. B_is are the positions of those spherical joints in world frame.

$$\vec{P} = \vec{O}_p^0 = \begin{bmatrix} x_d & y_d & z_d \end{bmatrix}^T$$

$$R_d = R_n^0 = R_x(\theta_x) * R_y(\theta_y) * R_z(\theta_z)$$

The final rotation matrix will look like this,

$$R_d = \begin{bmatrix} cos\theta_y cos\theta_z & -cos\theta_y sin\theta_z & sin\theta_y \\ cos\theta_x sin\theta_z + sin\theta_x sin\theta_y cos\theta_z & cos\theta_x cos\theta_z - sin\theta_x sin\theta_y sin\theta_z & -sin\theta_x cos\theta_y \\ sin\theta_x sin\theta_z - cos\theta_x sin\theta_y cos\theta_z & sin\theta_x cos\theta_z + cos\theta_x sin\theta_y sin\theta_z & cos\theta_x cos\theta_y \end{bmatrix}$$

A 4×4 transformation matrix can be described as below. With these transformation matrix we can get the position and orientation (in the world frame) of the objects, whose position and orientation is known in end effector frame.

$$T_d = T_n^0 = \begin{bmatrix} R_d & \vec{P} \\ O_{1\times 3} & 1 \end{bmatrix}$$

Length of each cable is our main goal and thus all the length vectors can be described as below,

$$\vec{q} = \vec{l} = \begin{bmatrix} l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 & l_8 \end{bmatrix}^T$$

Length vectors can be calculated with the following formula,

$$\vec{l}_i = \vec{B}_i - \vec{A}_i$$

$$\vec{B}_i = [R_n^0]\vec{b}_i + \vec{P}$$

Python code written below, calculates the lengths and length vectors required.

```
#################################################################
import numpy as np
from math import cos, sin, radians
from sympy import Symbol, cos, sin, Matrix, zeros
#################################################################
frame = [[-3.5, -3.5, 3.5],
         [-3.5, -3.5, 3.5],
         [3.5, -3.5, 3.5],
         [3.5, -3.5, 3.5],
         [-3.5, 3.5, 3.5],
         [-3.5, 3.5, 3.5],
         [3.5, 3.5, 3.5],
         [3.5, 3.5, 3.5]]
platform = [[0.3, -0.3, -0.3],
            [-0.3, 0.3, 0.3],
            [-0.3, -0.3, 0.3],
            [0.3, 0.3, -0.3],
            [-0.3, -0.3, -0.3],
            [0.3, 0.3, 0.3],
            [0.3, -0.3, 0.3],
            [-0.3, 0.3, -0.3]]
platform = np.array(platform)
frames = np.array(frame)

def rotation(ax,ay,az):
    tx = radians(ax)
    ty = radians(ay)
    tz = radians(az)
    rx = np.array([[1,0,0],[0,cos(tx),-sin(tx)],[0,sin(tx),cos(tx)]])
    ry = np.array([[cos(ty),0,sin(ty)],[0,1,0],[-sin(ty),0,cos(ty)]])
    rz = np.array([[cos(tz),-sin(tz),0],[sin(tz),cos(tz),0],[0,0,1]])
    return np.matmul(rx, np.matmul(ry,rz))

pose = [1,1,1,10,10,10]  ### Position and euler angles (x, y, z, rx, ry, rz)

for i in range(8):
    ll = frames[i,:] - np.array(pose[:3]) +
    np.matmul(rotatio(pose[3], pose[4], pose[5])
    ,np.matrix.transpose(platform[i,:]))      ###### Length vectors
    print(ll)

for i in range(8):
    ### Prints the lengths of cables
    print(np.sqrt(ll[i,0]**2 + ll[i,1]**2 +ll[i,2]**2))
######################################################################
```

3.2 Quasi Static Analysis

Quasi static analysis gives us the forces with which each cable should be pulled to keep the end effector in at the desired position. This force can be used to calculate the torques required by the motor to achieve a static position. Tensions can be found with the help of virtual work method and for that we need to complete inverse velocity kinematics. Velocity jacobian can be computed with the equation given below.

$$\dot{L} = J\dot{x}$$

$$J^T = \begin{bmatrix} -d_1^T & (d_1 \times Rb_1)^T \\ -d_2^T & (d_2 \times Rb_2)^T \\ -d_3^T & (d_3 \times Rb_3)^T \\ -d_4^T & (d_4 \times Rb_4)^T \\ -d_5^T & (d_5 \times Rb_5)^T \\ -d_6^T & (d_6 \times Rb_6)^T \\ -d_7^T & (d_7 \times Rb_7)^T \\ -d_8^T & (d_8 \times Rb_8)^T \end{bmatrix}$$

$$d_i = \frac{l_i}{||l_i||}$$

Here d_is are the unit vectors in the direction of the length of the cable.

From virtual work theorem, we can write,

$$\vec{\vec{q}} = J(q)\vec{\epsilon_d}$$

Where $\vec{\vec{q}}$ is the applied wrench on end effector and $\vec{\epsilon_d}$ is the cable tension matrix. Forces applied at the the spherical joints, due to the tensions in the cables, are in direction of the the length vector of the respective cable. Moments can be can be taken as the cross product of the force in the direction of length vectors and the position vector of a spherical joint in world frame. This forces and moments should be equal to the wrench provided on the end effector, in order to keep it stable at the desired pose.

So, tensions in cables can be computed with,

$$\vec{\epsilon_d} = J^{-1}(q)\vec{\vec{q}}$$

Here $J(q)$ is not a square matrix and thus we have to find the pseudo inverse of the matrix. Below is the python code which computes the tension values,

```
###################################################################
import numpy as np
from math import cos, sin, radians
from sympy import Symbol, cos, sin, Matrix, zeros
###################################################################

D = Matrix(ll)  ##### ll -- length matrix calculated from inverse kinematics
for i in range(8):
    D[:,i] = D[:,i]/(L[:,i].norm())  ## Unit vector

J = zeros(8,6)
for i in range(8):
    J[i,:3] = -D[:,i].transpose()  ### Direction of negative length unit vector
    J[i,3:] = (D[:,i].transpose()).cross(R*B[:,i])  ### Cross product

f = Matrix([0,0,-150*9.8,0,0,0])  ### External wrench

print(J.transpose().pinv()*f) ### Tension values
###################################################################
```

3.3 Grasping

Grasping modelling determines the set of contact forces and torques exerted by the end-effector to to resist the external wrench applied by the object to be held.

With the soft-finger contact model,

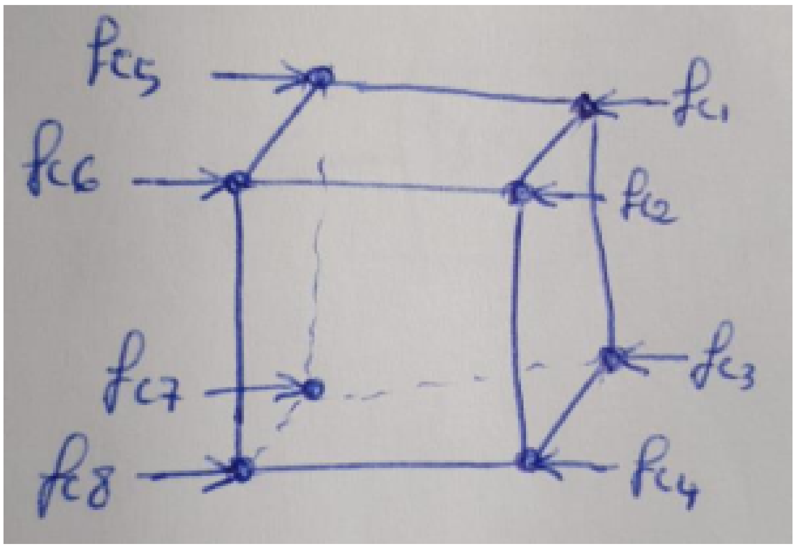


Figure 3.2: Grasp diagram of parallel gripper

$$\begin{cases} \text{wrench basis} = B_{c_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \text{contact force} = \vec{f}_{c_i} = \left[f_{i_1}\ f_{i_2}\ f_{i_3}\ f_{i_4}\right]^T \in FC_{c_i}; \\ FC_{c_i} = \left\{\vec{f} \in \mathbb{R}^4 : \sqrt{{f_1}^2 + {f_2}^2} \leq \mu f_3, f_3 \geq 0, |f_4| \leq \gamma f_3\right\} \end{cases}$$

where the static coefficient of friction $\mu = 1$ and the coefficient of torsional friction $\gamma = 1$, contact map can be defined as,

The contact forces f_c can be alternatively computed by formulating the problem as a quadratic mathematical program as follows:

$$\min \ \|\vec{f}_c\|^2$$

$$\text{subject to} \begin{cases} \vec{F}_g = G\,\vec{f}_c \\ f_{i_3} \geq 0;\ 1 \leq i \leq 8 \end{cases}$$

4 Validation

All the required mathematics to solve inverse kinematics, inverse velocity kinematics, quasi static analysis and grasping. But it is necessary to validate all those by comparing them with the actual robot's data or simulating the robot in the virtual world, which could be done with any simulation software. We have chosen ROS and Gazebo to validate our results.

4.1 ROS and Gazebo

Robotic operating system (ROS) is widely used middle ware to communicate between different modules of a robot. Gazebo is a physics engine software, which can simulate dynamics, contact and kinematics of robot, and which is quite reliable and gives results which are very close to the actual robot moving around in the real world.

4.2 Model Generation

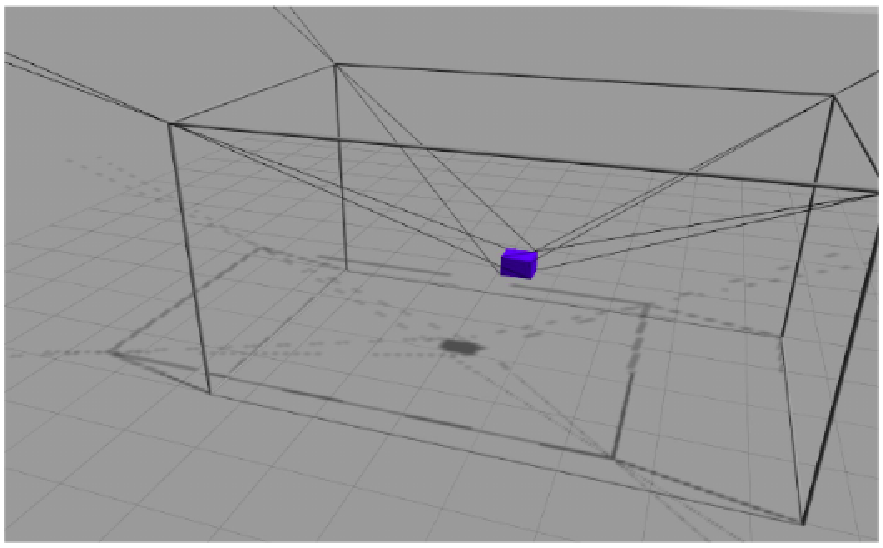


Figure 4.1: Model in gazebo

Assuming warehouse to be the outer frame of our model, we will create a virtual world with mesh files created with the Solidworks. Outer frame is assumed to be an empty square of dimension $3.5m$. Platform which moves in the entire workspace has been considered as a rigid square box of $0.6m$, which is shown as blue box in Figure 4.1. Gazebo doesn't allow to create massless and flexible links, so cables are taken as rods of mass $10g$ and of $10mmdiameter$. Joints between rods and frame corners are taken as universal joints and joints between rods and platform are taken as spherical ball joints without any joint limitations. As shown in Figure 4.2 rods and universal joints are connected with prismatic joints which allows the movement.

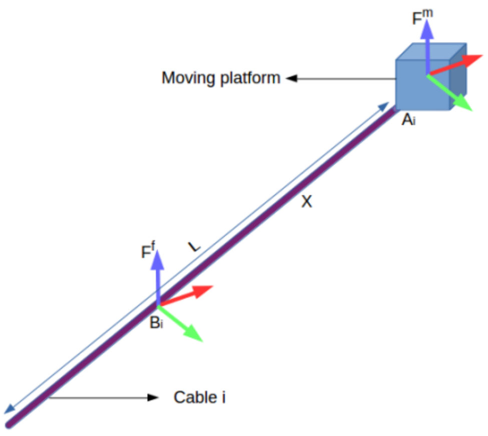
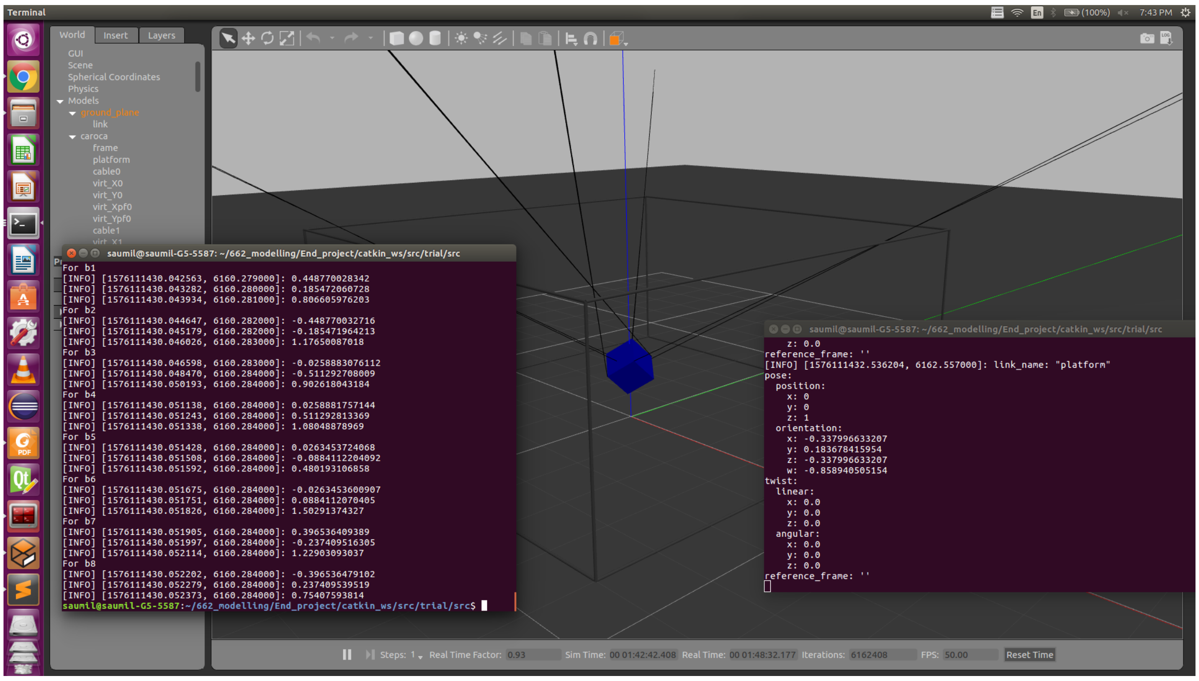


Figure 4.2: Platform and cables

4.2.1 Movement in World



```
saumil@saumil-G5-5587: ~/662_modelling/End_project/catkin_ws/src/trial/src
For b1
[INFO] [1576111430.042563, 6160.279000]: 0.448770028342
[INFO] [1576111430.043282, 6160.280000]: 0.185472060728
[INFO] [1576111430.043934, 6160.281000]: 0.806605976203
For b2
[INFO] [1576111430.044647, 6160.282000]: -0.448770032716
[INFO] [1576111430.045179, 6160.282000]: -0.185471964213
[INFO] [1576111430.046026, 6160.283000]: 1.17650087018
For b3
[INFO] [1576111430.046598, 6160.283000]: -0.0258883076112
[INFO] [1576111430.048470, 6160.284000]: -0.511292708009
[INFO] [1576111430.050193, 6160.284000]: 0.902618043184
For b4
[INFO] [1576111430.051138, 6160.284000]: 0.0258881757144
[INFO] [1576111430.051243, 6160.284000]: 0.511292813369
[INFO] [1576111430.051338, 6160.284000]: 1.08048878969
For b5
[INFO] [1576111430.051428, 6160.284000]: 0.0263453724068
[INFO] [1576111430.051508, 6160.284000]: -0.0884112204092
[INFO] [1576111430.051592, 6160.284000]: 0.480193106858
For b6
[INFO] [1576111430.051675, 6160.284000]: -0.0263453600907
[INFO] [1576111430.051751, 6160.284000]: 0.0884112070405
[INFO] [1576111430.051826, 6160.284000]: 1.50291374327
For b7
[INFO] [1576111430.051905, 6160.284000]: 0.396536409389
[INFO] [1576111430.051997, 6160.284000]: -0.237409516305
[INFO] [1576111430.052114, 6160.284000]: 1.22903093037
For b8
[INFO] [1576111430.052202, 6160.284000]: -0.396536479102
[INFO] [1576111430.052279, 6160.284000]: 0.237409539519
[INFO] [1576111430.052373, 6160.284000]: 0.75407593814
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws/src/trial/src$ 
```

```
saumil@saumil-G5-5587: ~/662_modelling/End_project/catkin_ws/src/trial/src
    z: 0.0
reference_frame: ''
[INFO] [1576111432.536204, 6162.557000]: link_name: "platform"
pose:
  position:
    x: 0
    y: 0
    z: 1
  orientation:
    x: -0.337996633207
    y: 0.183678415954
    z: -0.337996633207
    w: -0.858940505154
twist:
  linear:
    x: 0.0
    y: 0.0
    z: 0.0
  angular:
    x: 0.0
    y: 0.0
    z: 0.0
reference_frame: ''
```

Figure 4.3: Service call to set platform at [0,0,1,10,10,10] pose

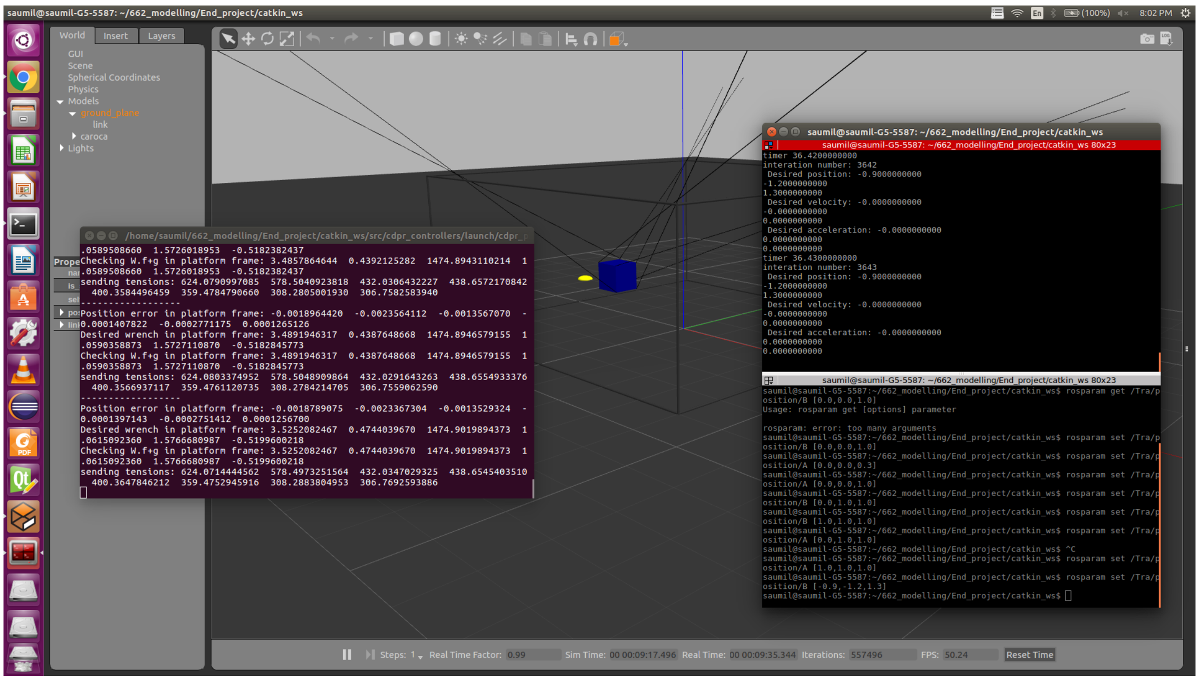
This gazebo model was connected with ROS and ros-services allow to change the platform position and orientations. With defining certain state of the platform we can use ros-services to get the positions of spherical joints in the world coordinate frame. As position of universal joints are fixed at the outer frame, we can get the lengths of every cables with this data. For different positions and orientations data points were taken and errors were computed. Computed errors for inverse kinematics are shown in Table 4.1.

Platform state $X, Y, Z, \theta_x, \theta_y, \theta_z$ (meters and degrees)	Lengths Theoretical (meters)	Lengths Simulation (meters)	Error Theoretical - practical (%)
(0, 0, 1, 0, 0, 0)	5.43323	5.42123	0.221
	5.70263	5.72631	-0.415
	5.70263	5.61263	1.578
	5.43323	5.42311	0.186
	5.43323	5.33231	1.857
	5.70263	5.76310	-1.060
	5.70263	5.70310	-0.008
	5.43323	5.40829	0.459
(1, 1, 1, 0, 0, 0)	6.74685	6.85112	-1.545
	6.96563	6.92991	0.513
	5.97662	5.91190	1.083
	5.50636	5.51600	-0.175
	5.72014	5.71399	0.108
	5.77235	5.73479	0.651
	4.52990	4.54066	-0.238
	4.18569	4.15690	0.688
(1, 1, 1, 0, 0, 10)	6.67700	6.69782	-0.312
	7.03262	7.02169	0.155
	6.03582	6.02740	0.140
	5.44141	5.44068	0.013
	5.65442	5.64184	0.223
	5.83674	5.81942	0.297
	4.58704	4.59409	-0.154
	4.12299	4.12911	-0.148
(1, 1, 1, 0, 10, 10)	6.63673	6.64264	-0.089
	7.07063	7.06338	0.103
	6.08699	6.09858	-0.191
	5.38411	5.37066	0.250
	5.71015	5.71496	-0.084
	5.78223	5.78306	-0.014
	4.53163	4.51634	0.337
	4.18381	4.19130	-0.179
(0, 0, 1, 10, 10, 10)	5.31702	5.29223	0.466
	5.81113	5.82569	-0.250
	5.82018	5.82776	-0.130
	5.30712	5.31712	-0.188
	5.42455	5.44927	-0.456
	5.71089	5.70578	0.089
	5.72394	5.71640	0.132
	5.41078	5.39976	0.204

Table 4.1: Errors in lengths of cables

4.3 Quasi Static Analysis

It is not possible to directly measure the tensions in the cable, as we have assumed them to be rigid rod with prismatic joint. It is easy to measure the forces required to pull the rods while sliding over prismatic joint. Gazebo allows to include force sensors on joints and we can use those values to validate out model. Friction at all the joints is taken as zero in order to reduce any loss of energy. Figure 4.4 shows the output of sensor values with velocity and acceleration of platform at the instance. Simulation values and errors computed are included in Table 4.2.



```
/home/saumil/662_modelling/End_project/catkin_ws/src/cdpr_controllers/launch/cdpr_
.0589508660  1.5726018953  -0.5182382437
Checking W.f+g in platform frame: 3.4857864644  0.4392125282  1474.8943110214  1
.0589508660  1.5726018953  -0.5182382437
sending tensions: 624.0790997085  578.5040923818  432.0306432227  438.6572170842
  400.3584496459  359.4784790660  308.2805001930  306.7582583940
------------------
Position error in platform frame: -0.0018964420  -0.0023564112  -0.0013567070  -
0.0001407822  -0.0002771175  0.0001265126
Desired wrench in platform frame: 3.4891946317  0.4387648668  1474.8946579155  1
.0590358873  1.5727110870  -0.5182845773
Checking W.f+g in platform frame: 3.4891946317  0.4387648668  1474.8946579155  1
.0590358873  1.5727110870  -0.5182845773
sending tensions: 624.0803374952  578.5048909864  432.0291643263  438.6554933376
  400.3566937117  359.4761120735  308.2784214705  306.7559062590
------------------
Position error in platform frame: -0.0018789075  -0.0023367304  -0.0013529324  -
0.0001397143  -0.0002751412  0.0001256700
Desired wrench in platform frame: 3.5252082467  0.4744039670  1474.9019894373  1
.0615092360  1.5766680987  -0.5199600218
Checking W.f+g in platform frame: 3.5252082467  0.4744039670  1474.9019894373  1
.0615092360  1.5766680987  -0.5199600218
sending tensions: 624.0714444562  578.4973251564  432.0347029325  438.6545403510
  400.3647846212  359.4752945916  308.2883804953  306.7692593886
```

```
saumil@saumil-G5-5587: ~/662_modelling/End_project/catkin_ws
saumil@saumil-G5-5587: ~/662_modelling/End_project/catkin_ws 80x23
timer 36.4200000000
interation number: 3642
 Desired position: -0.9000000000
-1.2000000000
1.3000000000
 Desired velocity: -0.0000000000
-0.0000000000
0.0000000000
 Desired acceleration: -0.0000000000
0.0000000000
0.0000000000
timer 36.4300000000
interation number: 3643
 Desired position: -0.9000000000
-1.2000000000
1.3000000000
 Desired velocity: -0.0000000000
-0.0000000000
0.0000000000
 Desired acceleration: -0.0000000000
0.0000000000
0.0000000000
```

```
saumil@saumil-G5-5587: ~/662_modelling/End_project/catkin_ws 80x23
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam get /Tra/p
osition/B [0.0,0.0,1.0]
Usage: rosparam get [options] parameter

rosparam: error: too many arguments
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam set /Tra/p
osition/B [0.0,0.0,1.0]
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam set /Tra/p
osition/A [0.0,0.0,0.3]
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam set /Tra/p
osition/A [0.0,0.0,1.0]
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam set /Tra/p
osition/B [0.0,1.0,1.0]
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam set /Tra/p
osition/B [1.0,1.0,1.0]
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam set /Tra/p
osition/A [0.0,1.0,1.0]
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ ^C
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam set /Tra/p
osition/A [1.0,1.0,1.0]
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ rosparam set /Tra/p
osition/B [-0.9,-1.2,1.3]
saumil@saumil-G5-5587:~/662_modelling/End_project/catkin_ws$ 
```

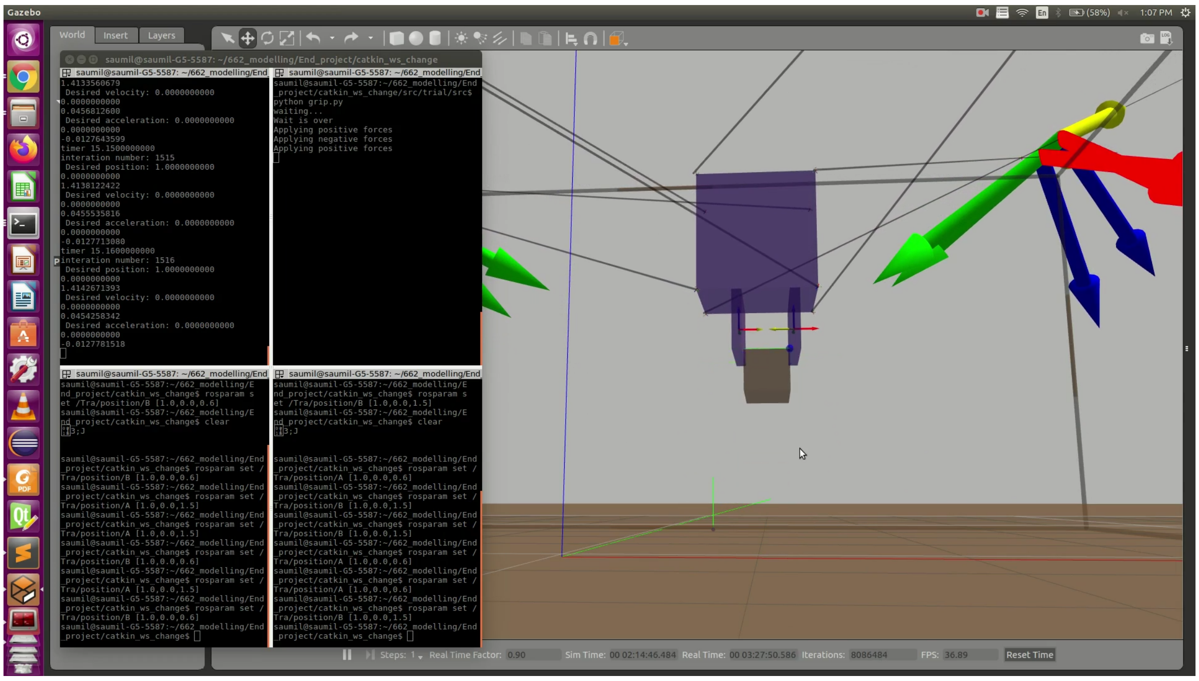
Figure 4.4: Forces at prismatic joint when platform at [0.9 , 1.2 , 1.3] pose

Platform state X , Y , Z , θ_x,θ_y,θ_z (Meters and Degrees)	Tensions Theoretical (Newtons)	Tensions Simulation (Newtons)	Error Theoretical - practical (%)
(0, 0, 1, 0, 0, 0)	419.143	418.41	0.175
	399.342	398.606	0.185
	399.342	398.606	0.185
	419.143	418.41	0.175
	419.143	418.41	0.175
	399.342	398.606	0.185
	399.342	398.606	0.185
	419.143	418.41	0.175
(0, 1, 1, 0, 0, 0)	329.332	329.766	-0.132
	326.691	326.958	-0.082
	316.491	316.6973	-0.065
	339.066	339.465	-0.118
	483.029	483.254	-0.047
	462.082	462.497	-0.090
	454.394	454.726	-0.073
	490.169	490.36	-0.039
(1, 1, 1, 0, 0, 0)	278.742	280.269	-0.545
	275.457	276.355	-0.325
	353.357	353.167	0.054
	388.380	389.811	-0.367
	375.106	375.858	-0.200
	367.076	368.9	-0.494
	522.489	523.4	-0.174
	561.784	562.426	-0.114
(0.9, 1.2, 1.3, 0, 0, 0)	623.073	624.071	-0.160
	576.756	578.497	-0.301
	428.198	432.035	-0.888
	437.294	438.654	-0.310
	397.514	400.364	-0.712
	359.935	359.475	0.128
	306.773	308.288	-0.491
	303.968	306.769	-0.913

Table 4.2: Errors between theoretical and simulated results for tensions in cable

4.4 Grasping

A simple parallel gripper with two plates of $0.3m \times 0.6m \times 0.1m$ were attached to the moving platform with the prismatic joint. Surfaces of both the plates were rough and contains friction coefficient of $\mu = 1$. "ApplyJointEffort" service from ROS was used to apply force at the prismatic joint. A rigid rectangular block with mass $1kg$ was introduced in the virtual world. As per calculations gripper would require 5 N force to pick up the block. After applying 5 N force to the prismatic joint, gripper was unable to grip the block while moving, because of acceleration of the end effector. With the application of 6 N force it was able to pick up the block as shown in Figure 4.5.


Gazebo
World
Insert
Layers
saumil@saumil-G5-5587: ~/662_modelling/End_project/catkin_ws_change
Reset Time

```
saumil@saumil-G5-5587: ~/662_modelling/End_
1.4133560679
 Desired velocity: 0.0000000000
0.0000000000
0.0456812600
 Desired acceleration: 0.0000000000
0.0000000000
-0.0127643599
timer 15.1500000000
interation number: 1515
 Desired position: 1.0000000000
0.0000000000
1.4138122422
 Desired velocity: 0.0000000000
0.0000000000
0.0455535816
 Desired acceleration: 0.0000000000
0.0000000000
-0.0127713080
timer 15.1600000000
interation number: 1516
 Desired position: 1.0000000000
0.0000000000
1.4142671393
 Desired velocity: 0.0000000000
0.0000000000
0.0454258342
 Desired acceleration: 0.0000000000
0.0000000000
-0.0127781518
```

```
saumil@saumil-G5-5587: ~/662_modelling/End_
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change/src/trial/src$
python grip.py
waiting...
Wait is over
Applying positive forces
Applying negative forces
Applying positive forces
```

```
saumil@saumil-G5-5587: ~/662_modelling/End_
saumil@saumil-G5-5587:~/662_modelling/E
nd_project/catkin_ws_change$ rosparam s
et /Tra/position/B [1.0,0.0,0.6]
saumil@saumil-G5-5587:~/662_modelling/E
nd_project/catkin_ws_change$ clear
3;J


saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/B [1.0,0.0,0.6]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/A [1.0,0.0,1.5]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/A [1.0,0.0,1.5]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/B [1.0,0.0,0.6]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/A [1.0,0.0,1.5]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/B [1.0,0.0,0.6]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$
```

```
saumil@saumil-G5-5587: ~/662_modelling/End_
saumil@saumil-G5-5587:~/662_modelling/E
nd_project/catkin_ws_change$ rosparam s
et /Tra/position/B [1.0,0.0,1.5]
saumil@saumil-G5-5587:~/662_modelling/E
nd_project/catkin_ws_change$ clear
3;J


saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/A [1.0,0.0,0.6]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/B [1.0,0.0,1.5]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/B [1.0,0.0,1.5]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/A [1.0,0.0,0.6]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/A [1.0,0.0,0.6]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$ rosparam set /
Tra/position/B [1.0,0.0,1.5]
saumil@saumil-G5-5587:~/662_modelling/End
_project/catkin_ws_change$
```

Steps: 1 Real Time Factor: 0.90 Sim Time: 00 02:14:46.484 Real Time: 00 03:27:50.586 Iterations: 8086484 FPS: 36.89

Figure 4.5: Gripping the block of one kg

5 Conclusion and Future Work

In this report, several important aspects of 8-cable driven parallel robot have been modeled, containing inverse kinematics, inverse velocity kinematics (though not included in the proposal), quasi static analysis and grasping. All those models were verified with the simulation using Gazebo and ROS. Simulation also proves that CDPR is faster in terms of pick and place task and be used for bigger warehouses.

The next step would be to model dynamic analysis of the robot and stiffness analysis of cables. These models will help to create suitable control system for the real time application. In the real world, robot must be able to avoid obstacles and shall contain the capabilities to generate and follow trajectories that avoids all the obstacles.

Entire code base, robot models, ROS and Gazebo files and videos can be found on the following link.
https://github.com/SaumilShah66/CDPR-Robot-modelling-project

References

[1] Okoli, Franklin Lang, Yuchuan Kermorgant, Olivier Caro, Stéphane "CableDriven Parallel Robot Simulation Using Gazebo and ROS, 2019. doi: 10.1007/978-3-319-78963-737

[2] https://www.ros.org/

[3] http://gazebosim.org/