Scientic Experimentation and Evaluation

Alex Jude, Madhumetha Ramesh, Yogeshkarna Govindaraj

December 2020

1 Manual motion observation of differential drive robot

The aim of this experiment is to analyse the motion of the designed differential robot using the Lego EV3 kit. The pose variations of the differential drive robot is also analysed.

1.1 Measurement Facility

Materials used to perform this experiment is listed below,

- Lego EV3 kit
- Two pen refills
- Grid sheet of 25mm each (A1 size)
- 30cm Ruler
- Rubber bands

1.1.1 Robot Design

- The robot is designed using Lego Mindstorms EV3(evolution 3) kit which is used for educational purposes. Figure 1 shows the robot designed using the Lego EV3 kit.

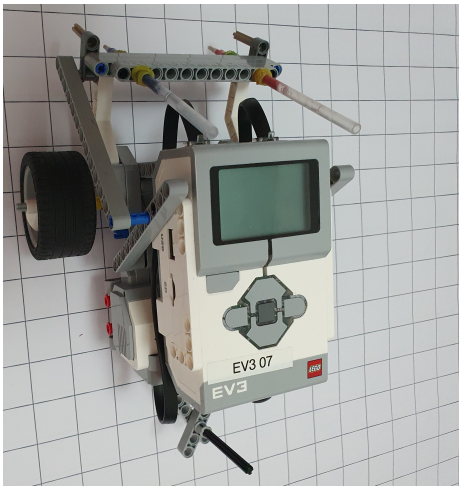

Figure 1: *Top view of the robot*

- The robot designed follows the differential drive configuration in which the motion analysis is done.
- Differential drive robot has two wheels which has two independent actuators for each wheel and has a castor wheel which is non-driven. This robot design is showed in figure 2 and the red highlighted box represents the castor wheel.

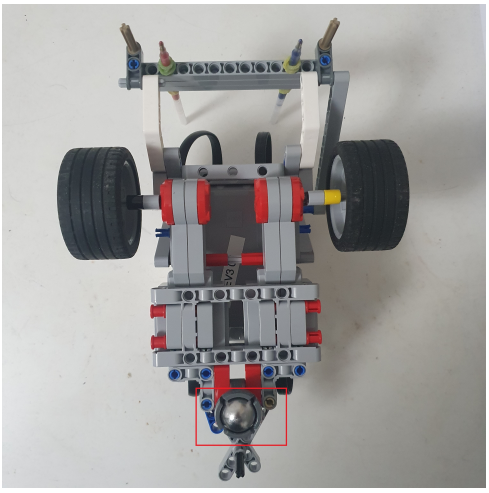

Figure 2: *Bottom view of the robot*

- EV3 controller provides command to the motor for the motion purposes and it is placed in the base of the robot between the two pillars in order to achieve the rigidity while taking the measurements during the robot motion. (Figure 1)
- EV3 controller has options to save and run the program which is done using the Mindstorms software. Also Bluetooth and WiFi option is available for the connectivity between the PC and the robot controller. Program can be done in the software and can be transferred to the controller.
- The robot built is placed on the top of the A1 size grid sheet of 25mm each for the measurement process. (Figure 1)
- For the measurement process, two pen refills are fixed in the rear side of the robot inline and parallel which is represented in Figure 3 with the green box. Rubber bands is used to tighten the refills which provides friction.

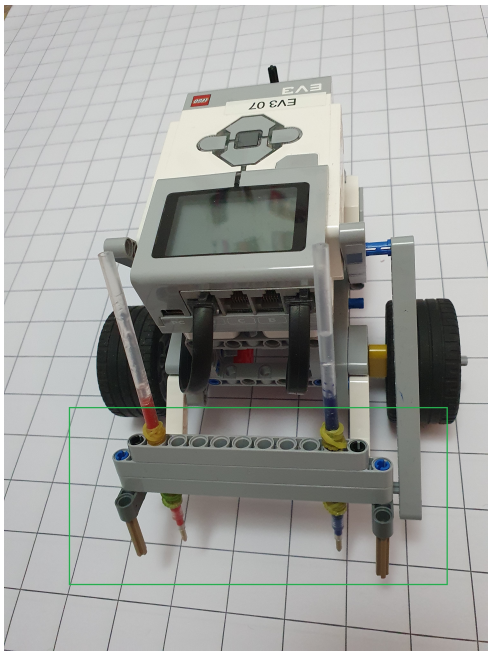

Figure 3: *Pen refills for the measurement process*

- A calibration marker is also fixed at the front of the robot other than the two pen refills in the rear in order to ensure the position from where the robot starts (Figure 8). The calibration marker is highlighted with red color rectangle in Figure 4.
- A manual mechanism is built for the two pen refills to lock them when the robot is moving and it can be unlocked when it is needed to mark the measurement. This mechanism makes sure that the pen refills is not in contact with the grid sheet during robot motion. (Figure 6)
- The measurand is the pose of the robot $[x, y, \theta]^T$ where the translation $[x; y]^T$ is measured in mm and the orientation θ is measured in degree.
- Distance between the two pen refills, distance between the calibration marker and the pen refills centre and distance of the wheel centres are represented in Figure 8.

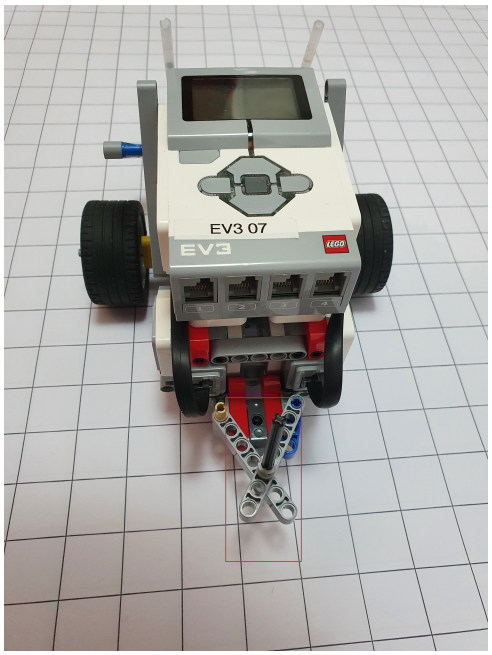

Figure 4: *Calibration marker in the rear side*

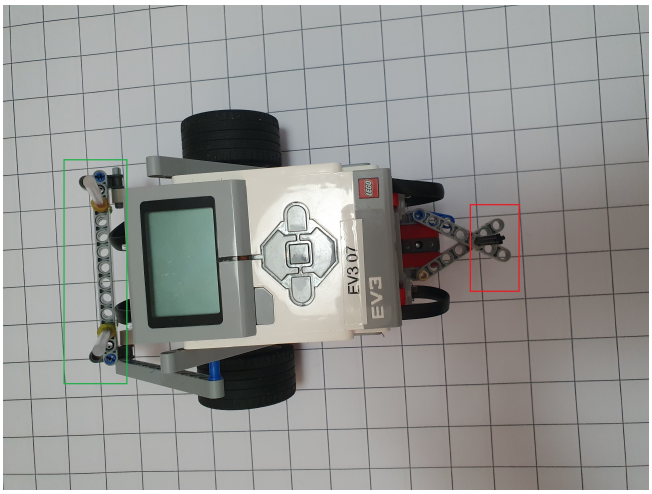

Figure 5: *Top view of two pen refills and calibration marker*

- Robot is initially placed in the centre of one side of the grid sheet and using calibration marker the initial position is marked. (Figure 7)

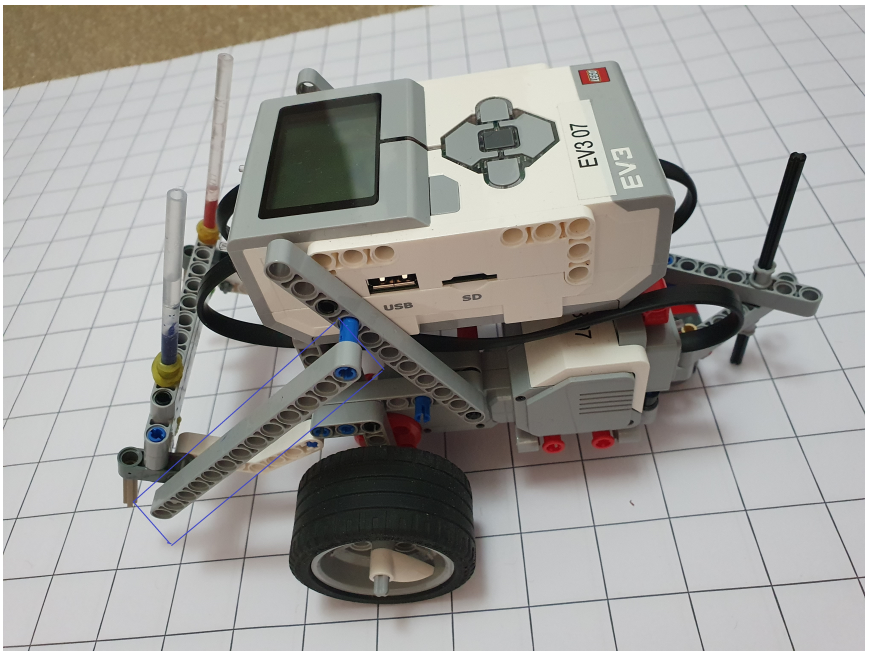

Figure 6: *Mechanism for two pen refills*

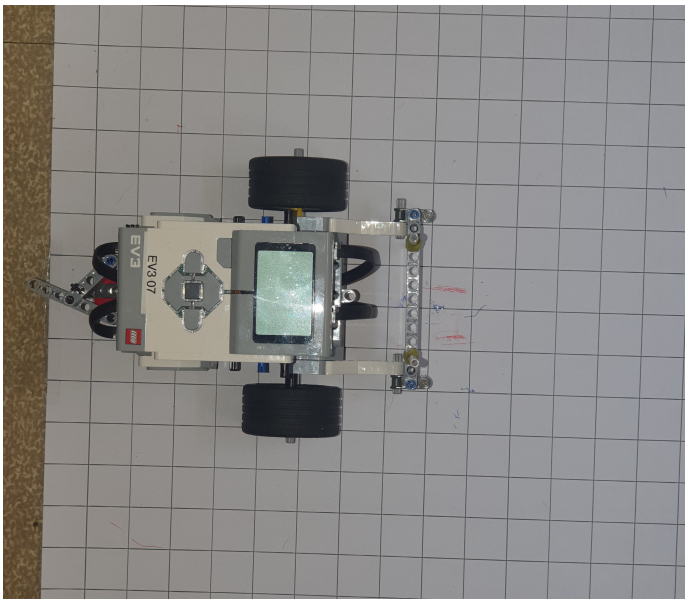

Figure 7: *Initial position of the robot*

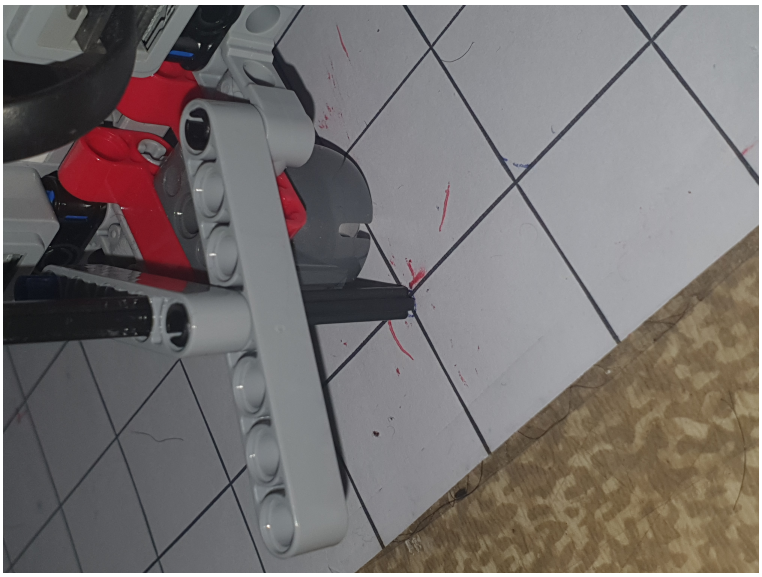

Figure 8: *Marking the initial position of the robot*

- The initial point of the robot in the grid sheet is marked by lowering the two pen refills for the reference, then the markers are lifted up and locked when the robot goes along its path.
- Once it reaches the end position the pen refills are again lower down and those point is marked (Figure 6). Additionally, a 30cm is ruler is used to measure the grid co.ordinates (Figure 9).
- Figure 10 represents the distance between two pen refills, distance between the centre of both the ends and also the distance between wheel centres.

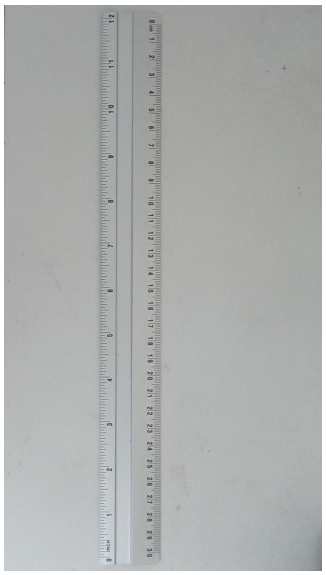

Figure 9: *Ruler used for measurement*

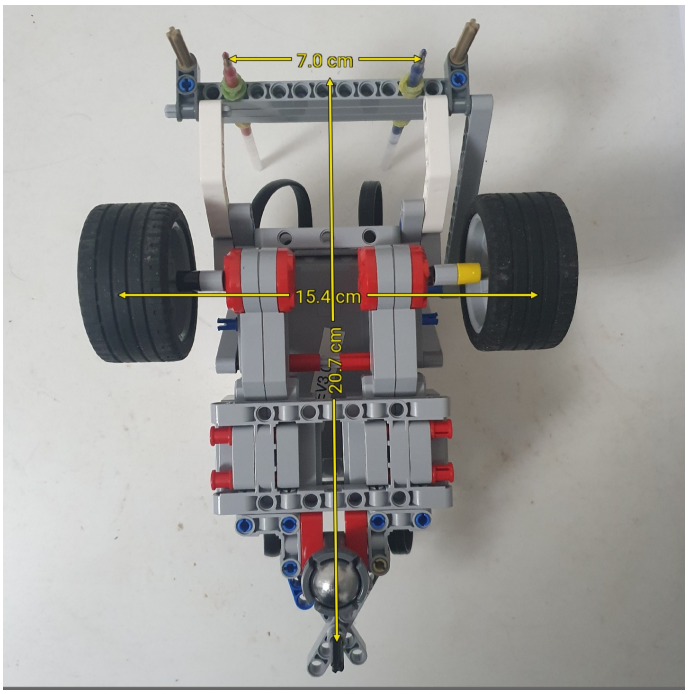

Figure 10: *Measurements of the robot*

- For taking the reading, reference design given is constructed which has one marker at the front and one marker at the back. To verify the robots initial position for the each run both the markers were aligned collinear to the x-axis and back marker is aligned with origin.[Figure 11,12]

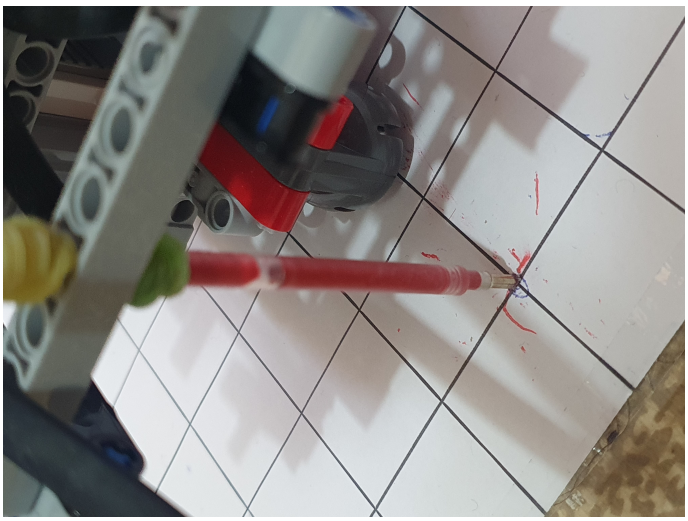

Figure 11: *Initial position of the reference design robot*

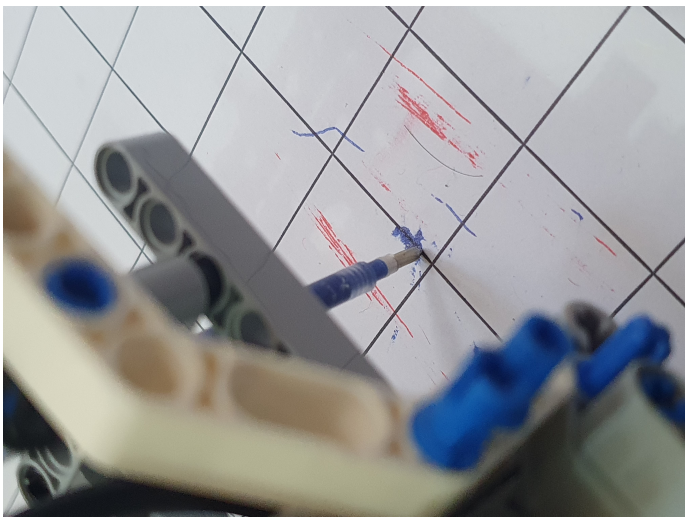

Figure 12: *Initial position of the reference design robot*

- Figure 13 represents the initial position of the reference design robot which is used for taking the readings.

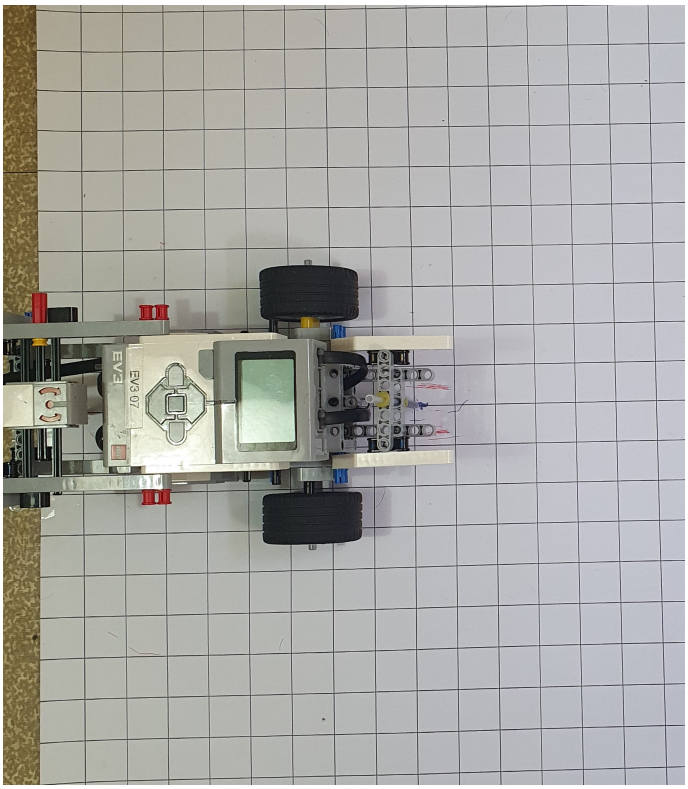

Figure 13: *Initial position of the reference design robot*

1.1.2 Robot pose calculation

- Once we marked the end-poses $M1'(x_1, y_1)$, $M2'(x_2, y_2)$ the mid-pint of that two co-ordinates(B') is found.
- The vector from the center of the axle at the start position to this calculated using mid-point is equal to the summation of vector from the center of the axle (A) at the start position to the center of the axle at the end position (A') and vector from center of the axle at the end position to the calculated mid-point.
- From the above method we can find the position of the end pose of the robot with respect to the center of the axle. Once we got the vector the angle is calculated between the basis vector and the computed vector.

$$\theta = arccos(unitvector(u) \times unitvector(v')) \tag{1}$$

- The two vectors are solved to obtain the orientation θ using equation 1.
- The position of the wheel-axis center after motion commands A' will represent end position of the robot. This process is repeated to obtain poses of all motion.
- Figure 14 represents the calculation of the robot end poses.

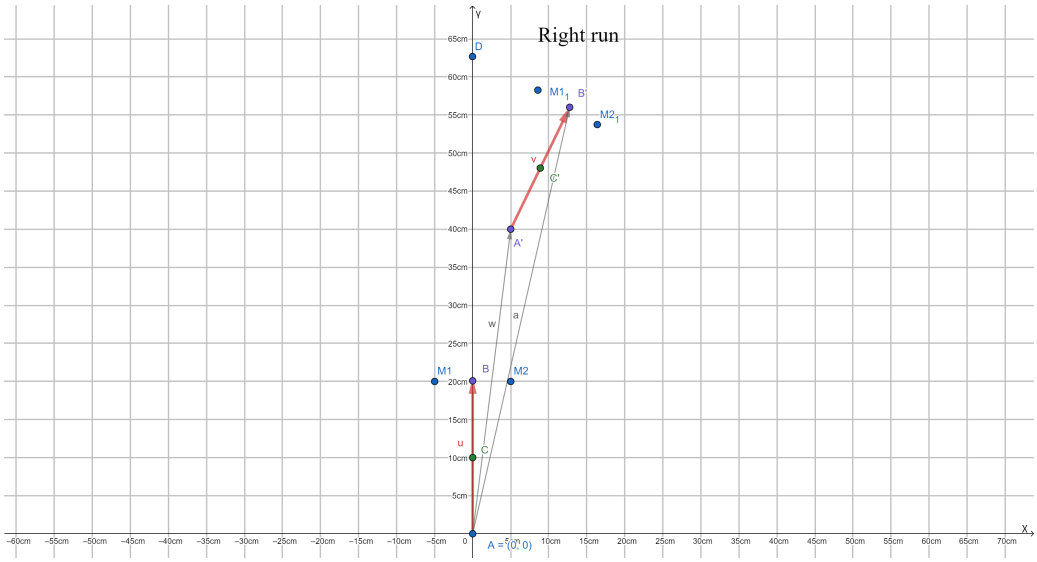

Figure 14: *Robot pose calculation*

1.1.3 Problems expected to occur during taking measurement

- Least count of a measuring instrument is the smallest value that can be measured by it. The least count of a standard ruler or a line gauge is 1mm.
- The grids present in the grid paper are of the dimension 25mm x 25 mm and while taking measurements and marking on the grid paper, there can be an error of +/- 1mm
- While taking the measurement, if we observe the measurement markings at an angle, then the readings will either be less or more than the actual value. This is called parallax error. Care should be taken to align our line of sight exactly on top of the measurement marking to avoid parallax error.
- The BRIC battery should be charged to an optimal level so as to ensure that proper amount of voltage is supplied to various ports. A low battery charge would result in inefficiency in robot performance and in extension, would hinder the measurement process.

- All the connections of the robot should be checked. Scenarios like loose connections of jumper cables or various lego blocks or misalignment of the wheel etc would lead to irregular robot behavior.
- Other plausible cause of error would be the friction between the grid paper and the robot wheels which might either cause slippage or restricted motion. When the robot motion starts, there might be a slight moment of slippage of wheels off the paper.
- In our design, the extension which holds two markers are lowered down to mark its end position after the robot finishes its motion. Constant raising and lowering motion of this robot extension would lead to it being flimsy and lose its friction.
- Lastly, to move the robot, we have to select "SEE Motion" option and further press the arrow key to specify the direction of motion. Force applied on pressing the buttons may result in slightly moving the robot from its starting point.

1.1.4 Rough estimate of expected precision

- Measurement error (= Least count) = +/- 1mm
- Error while marking on the grid = +/- 1mm
- Parallax error = +/- 2mm
- Error caused by slippage = +/- 3mm
- Error while marking (lowering the markers on the grid paper) = +/- 3mm
- Error contributed by pressing the buttons on BRIC = +/- 1mm
- Our estimate on the expected precision of measurement would be +/- 11mm.

1.1.5 Measurement Error propagation estimation

The uncertainties or error 'δ' associated with the measurement system which may occur because of measuring principle, measuring method or sensitivity of measuring method grows faster than the

sum of individual errors. This phenomenon is called **error propagation**. This can be modelled by,

$$\delta Q = Jac(Q) \cdot \delta(x) \tag{2}$$

Where, 'δ' is error associated with each measurement and 'Q' is function of 'x'.

- The error in measurement caused while measuring co-ordinates (x, y) will propagate throughout orientation estimation process. This can be generalized using equation 2.

$$\theta = arctan2(\frac{y_2 - y_1}{x_2 - x_1}) \tag{3}$$

  where (x_1, y_1) are starting coordinates and (x_2, y_2) are end coordinates From equation 3 we can imply that Orientation $\theta = f(x_1, y_1, x_2, y_2)$

- So the error propagation along orientation estimation is,

$$\delta\theta = Jac(\theta) \cdot \delta \begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \end{bmatrix} \tag{4}$$

$$\delta\theta = \begin{bmatrix} -\frac{y_1 - y_2}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2} & \frac{y_1 - y_2}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2} \\ -\frac{-x_1 + x_2}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2} & \frac{-x_1 + x_2}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2} \end{bmatrix} \cdot \delta \begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \end{bmatrix} \tag{5}$$

$$\delta\theta = \frac{(y_1 - y_2) \times (\delta x_2 - \delta x_1)}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2} + \frac{(x_1 - x_2) \times (\delta y_1 - \delta y_2)}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2} \tag{6}$$

$$\delta\theta = \frac{((y_1 - y_2) \times (\delta x_2 - \delta x_1) + (x_1 - x_2) \times (\delta y_1 - \delta y_2))}{(-x_1 + x_2)^2 + (-y_1 + y_2)^2} \tag{7}$$

	X_1 (cm)	Y_1 (cm)	X_2 (cm)	Y_2 (cm)	Lowerbound error (deg)	Upperbound error (deg)
Straight run	0.9	42.3	1.5	62.7	-0.95	0.95
Left run	-5.1	38.4	-16.2	54.0	-0.80	0.80
Right run	5.4	38.7	13.6	54.9	-1.49	1.49

Table 1: *Error propagation in orientation calculation*

- The rough expected precision is estimated as 1.1 cm. The Upper-bound and Lower-bound of orientation error estimation can be calculated by substituting all combinations of $(\delta x_1, \delta y_1, \delta x_2, \delta y_2)$, For example $(1.1, -1.1, -1.1, 1.1)$ in equation 7. The maximum and minimum values are taken as upperbound and lowerbound errors in orientation calculation. Table 1 shows the example of lowerbound and upperbound for approximate measurements.

- Python code for the error propagation is mentioned below,

```
import numpy as np
import sympy as sp
from itertools import combinations
from IPython.display import display

# Computing the Jacobian
def jacobian():
    x1, y1, x2, y2, theta = sp.symbols('x1, y1, x2,
    ↪  y2, theta')
    equation_2 = sp.atan2((y2-y1),(x2-x1))
    theta = sp.Matrix([equation_2])
    coordinates = sp.Matrix([x1, x2, y1, y2])
    jac = theta.jacobian(coordinates)
    return jac

#Calculating the upper bound error estimate
def calculate_upperbound(jac,val):
    x1, x2, y1, y2, theta = sp.symbols('x1, x2, y1,
    ↪  y2, theta')
    # Using +/- 15mm as the estimated error
    ↪  associated with each measurements
    #delta_xy_upper = np.array([1.1, 1.1, 1.1, 1.1])
```

```
    comb = list(combinations( [-1.1, -1.1, -1.1,
    ↪  -1.1,1.1,1.1,1.1,1.1],4))
    comb_list = []
    for i in comb:
        delta_xy_upper = np.asarray(i)
        jac_subs =
        ↪  jac.subs({x1:val[0],x2:val[1],y1:val[2],y2:val[3]})
        upper_bound_error = np.dot(np.array(jac_subs),
        ↪  delta_xy_upper)
        comb_list.append(upper_bound_error[0])
    upper_bound_error = max(comb_list)
    #upper_bound_error =
    ↪  round(float(upper_bound_error[0]),4)
    result = np.rad2deg(float(upper_bound_error))
    print('Upper bound error estimation',result)

#Calculating the lower bound error estimate
def calculate_lowerbound(jac,val):
    x1, x2, y1, y2, theta = sp.symbols('x1, x2, y1,
    ↪  y2, theta')
    # Using +/- 15mm as the estimated error
    ↪  associated with each measurements
    #delta_xy_lower = np.array([-1.1, 1.1, 1.1,
    ↪  -1.1])
    comb = list(combinations( [-1.1, -1.1, -1.1,
    ↪  -1.1,1.1,1.1,1.1,1.1],4))
    comb_list = []
    for i in comb:
        delta_xy_lower = np.asarray(i)
        jac_subs =
        ↪  jac.subs({x1:val[0],x2:val[1],y1:val[2],y2:val[3]})
        lower_bound_error = np.dot(jac_subs,
        ↪  delta_xy_lower)
        comb_list.append(lower_bound_error[0])
    upper_bound_error = min(comb_list)
    result = np.rad2deg(float(upper_bound_error))
    print('lower bound error estimation',result)

jac = jacobian()
```

```
calculate_upperbound(jac,[-5.1,38.4,-16.2,54])
calculate_lowerbound(jac,[-5.1,38.4,-16.2,54])
```

1.2 Programs and Parameters

1.2.1 Program

- The robot is reconstructed based on the reference design provided and then the readings are taken.

- Figure 15 shows the program done using Lego Mindstorms EV3 software to run the robot.

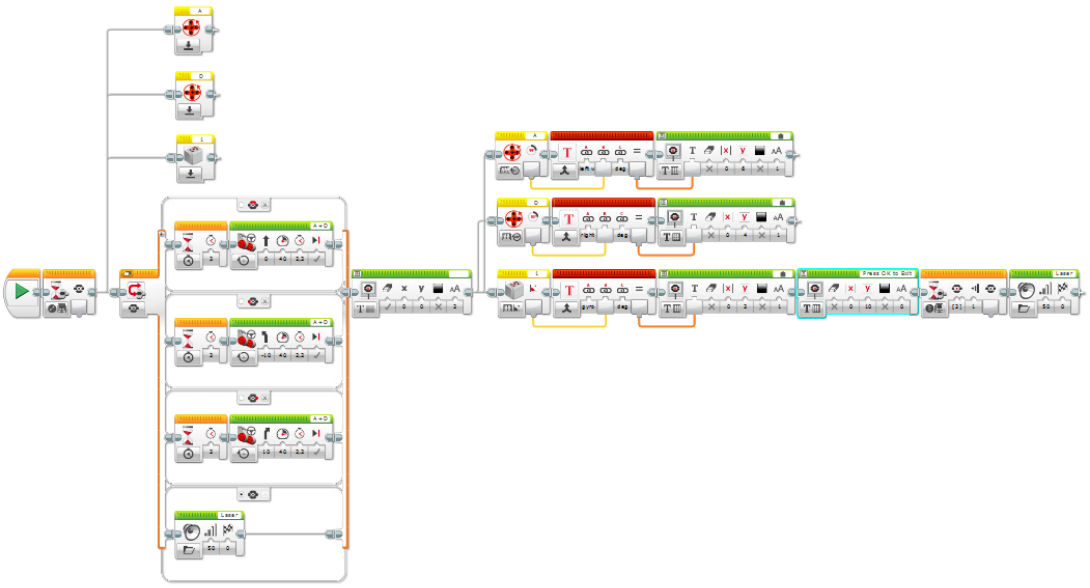

Figure 15: *Program to run the robot*

- The code used in Lego Mindstorm EV3 software is given below

```
button_left,button_right,button_up = False,False,False
speed = 40 #rpm
time = 2.2 #seconds
steering_left = -10 # degrees
steering_right = 10 # degrees
steering_straight = 0 # degrees
```

```
if button_left == True:
    move_left(ports(A,D), steering_left,speed,time)
    while time=0:
        time -= change_in_time
        rot_a = encoder.motor_A.count()
        rot_b = encoder.motor_D.count()
        gyro_reading = gyro.measure()

    return print(rot_a, rot_b, gyro_reading)

if button_right == True:
    move_right(ports(A,D), steering_right,speed,time)
    while time =0:
        time -= change_in_time
        rot_a = encoder.motor_A.count()
        rot_b = encoder.motor_D.count()
        gyro_reading = gyro.measure()
    return print(rot_a, rot_b, gyro_reading)

if button_straight == True:
    move_straight(ports(A,D),
     ↪  steering_straight,speed,time)
    while time=0:
        time -= change_in_time
        rot_a = encoder.motor_A.count()
        rot_b = encoder.motor_D.count()
        gyro_reading = gyro.measure()
    return print(rot_a, rot_b, gyro_reading)
```

1.2.2 Parameters

- Wheel Diameter = 5.6 cm +/- 0.2 cm
- Distance between centre of the wheel = 16.3 cm +/- 0.1 cm
- Distance between two markers = 20.5 cm +/- 0.2 cm
- Distance between marker A and wheel axle centre = 14.5 cm +/- 0.2 cm

1.3 Experiment

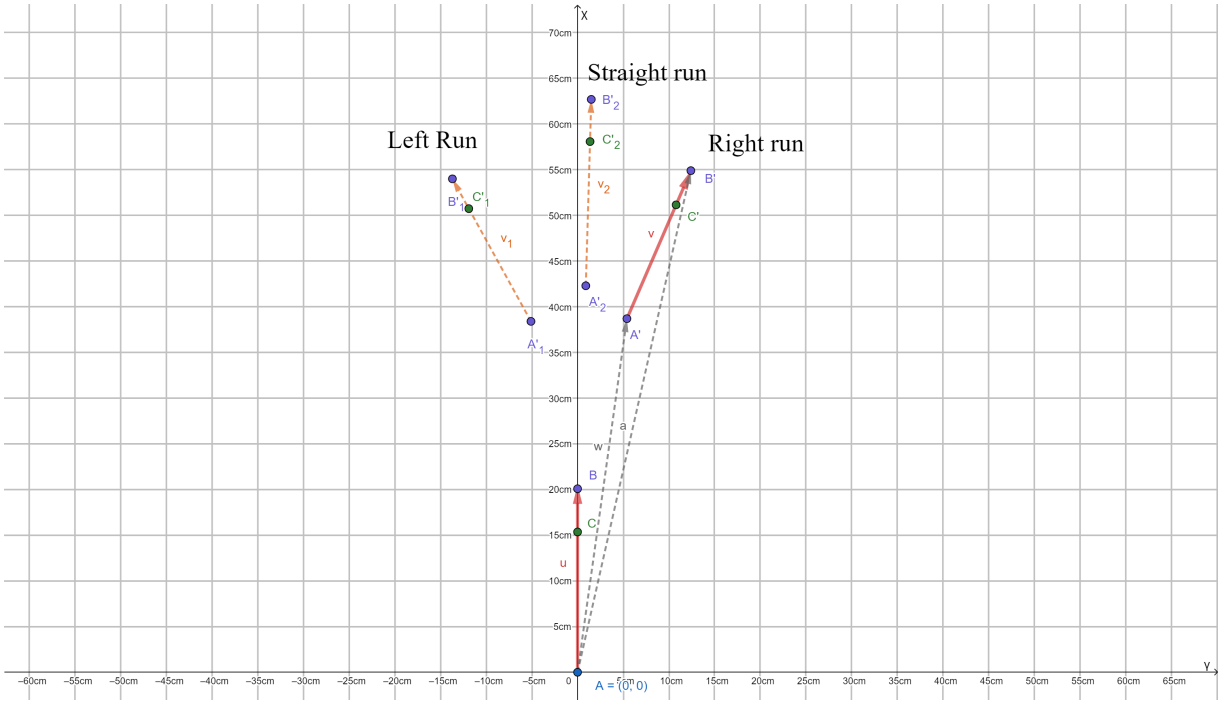

Figure 16: *End poses calculation*

- The origin is taken as start position (0,0), In figure 16 A and B represents two markers, markers are aligned to the Y-axis and A marker is aligned with origin such that robot starts from same point for all test runs.

- The point 'C' represents the robot-axle midpoint and it is collinear to the two markers and the distance from two markers are known C to A = -14.5 cm and C to B = 6 cm.
- Given motion commands for right motion, The two markers A and B will move to end position A' and B'. The position of two markers are measured using ruler and points are noted as A' and B' respectively.
- The process is repeated for all motions Left run, Straight run and Right run and approximate end poses of these runs are measured as (A'_1, B'_1), (A'_2, B'_2)and (A', B') respectively.
- Robot end pose is represented as (x, y, θ) with respect to the center of the axle. Since the measurements are taken with 'A' marker as origin, the measurements are transformed to the robot center 'C' frame which is at (0,14.5).
- After transformation the orientation of robot is estimated using start pose(A, B) and end pose (A', B').
- A vector u represents vector from point A to B and a vector v represents vector from point A' to B'. Dividing the vectors by their respective norms will result in unit vectors u' and v' respectively.

$$\theta = arccos(u' \times v') \tag{8}$$

- The two vectors are solved to obtain the orientation θ using equation 8.
- The position of the wheel-axis center after motion commands C' will represent end position of the robot.

$$(x, y) = v + (d \times v') \tag{9}$$

  where d is distance between marker A and wheel-axle center, $d = 14.5$
- The end position of the robot with respect to wheel-axle center can be obtained using equation 9. The processes is repeated for all motion measurements to obtain end pose of the robot. End Poses are Tabulated in section 1.4.

- Python program for the pose estimation is mentioned below,

```
import numpy as np
import math
import pandas as pd
import matplotlib.pyplot as plt


 # Calculating the translation
def vectorCentreOfAxle(points):

    diff = [points[1][0]-points[0][0],
     ↪  points[1][1]-points[0][1]]
    norm_ed = diff/np.linalg.norm(diff)
    ed = points[0] + (norm_ed * 16.5)
    return round(float(ed[0]),1),
     ↪  round(float(ed[1]),1)

 # Calculating the orientation
def calculate_angle(points):
    diff = [[points[1][0]-points[0][0]],
     ↪  [points[1][1]-points[0][1]]]
    x = np.array([0, 20.5])
    y = np.array(diff)
    norm_x = np.linalg.norm(x)
    norm_y = np.linalg.norm(y)
    x1 = x/norm_x
    y1 = y/norm_y
    theta = np.arctan(np.dot(x1,y1))
    return round(float(np.rad2deg(theta)),1)

# Proessing the measurements
def process(file,dest, Right = False):
    print(Right)
    x_list = list()
    y_list = list()
    angle_list = list()
    file_pd = pd.read_csv(file)
    for i in range(len(file_pd)):
```

```
        x,y =
         ↪  vectorCentreOfAxle([(file_pd['x1'][i],
        file_pd['y1'][i]),
        (file_pd['x2'][i],file_pd['y2'][i])])
        angle =
         ↪  calculate_angle([(file_pd['x1'][i],
        file_pd['y1'][i]),
        (file_pd['x2'][i],file_pd['y2'][i])])
        x_list.append(x)
        y_list.append(y)
        angle_list.append((angle))
    return x_list, y_list, angle_list
```

1.4 Observed Data

1.4.1 Manual Readings

- Tables 2,3,4 shows the manual End pose readings for the left, right and straight motion of the robot respectively using ruler.

LEFT RUN			
S.no	X(cm)	Y(cm)	Angle(degrees)
1	50.8	-13.4	41.1
2	51.5	-13.3	38.6
3	51.8	-12.7	38.2
4	51.4	-13.2	39.2
5	51.8	-12.8	38.4
6	51.1	-14.3	41.1
7	51.4	-13.1	38.4
8	51.3	-13.9	40.1
9	51.6	-13.1	38.8
10	51.6	-13.4	39.2
11	51.3	-12.7	37.6
12	51.3	-13.3	39.9
13	51.3	-13.3	39.1
14	51.3	-13.7	39.7
15	51.5	-13.3	39.1
16	51.5	-12.9	38.7
17	51.4	-13.9	39.6
18	51.4	-13.2	38.6
19	51.4	-13.6	39.4
20	51.1	-13.1	38.1

Table 2: *End pose readings for arc at left*

RIGHT RUN			
S.no	X(cm)	Y(cm)	Angle(degrees)
1	51.9	12.8	-37.1
2	51.9	12.7	-36.4
3	51.6	12.8	-36.6
4	51.4	13.2	-37.8
5	51.6	12.7	-36.4
6	51.6	13.6	-38.1
7	51.3	12.8	-37.4
8	51.3	14.1	-39.1
9	51.4	13.2	-37.4
10	51.2	14.1	-39.1
11	51.2	14.2	-39.2
12	51.5	13.6	-38.1
13	51.5	13.1	-36.8
14	51.5	13.1	-37.1
15	51.2	13.2	-37.3
16	51.1	14.2	-39.6
17	51.3	13.2	-37.2
18	51.5	12.8	-36.6
19	51.2	13.1	-37.3
20	51.3	13.3	-37.3

Table 3: *End pose readings for arc at right*

STRAIGHT RUN			
S.no	X(cm)	Y(cm)	Angle(degrees)
1	58.8	1.4	1.7
2	58.9	0.7	0.0
3	58.8	0.1	0.6
4	58.7	0.2	0.3
5	59.1	0.3	0.6
6	59.1	-0.6	2.3
7	58.8	0.0	1.1
8	58.6	-0.6	2.1
9	58.6	2.5	7.1
10	58.7	-0.3	1.1
11	58.8	-0.3	1.1
12	58.9	-0.7	2.3
13	58.8	-0.6	2.3
14	58.8	0.0	0.9
15	58.9	0.5	0.0
16	59.1	0.7	0.0
17	59.1	0.8	0.3
18	58.7	0.5	0.0
19	58.5	0.9	0.6
20	59.1	-0.1	0.8

Table 4: *End pose readings for straight motion*

1.4.2 Sensor Readings

- Tables 5,6,7 shows the sensor readings of the left encoders, right encoders and gyro from the robot for the straight, left and right motion of the robot respectively.

STRAIGHT RUN			
S.no	Left Wheel(degrees)	Right Wheel(degrees)	Gyro(degrees)
1	891	889	0
2	892	892	0
3	889	894	0
4	891	892	0
5	891	893	0
6	890	895	-1
7	890	892	0
8	890	892	-1
9	891	892	0
10	891	892	0
11	891	892	-1
12	889	894	-1
13	890	892	-1
14	891	892	0
15	890	891	0
16	891	893	0
17	893	892	0
18	892	894	0
19	889	890	0
20	890	892	0

Table 5: *End pose readings for straight motion*

LEFT RUN			
S.no	Left Wheel(degrees)	Right Wheel(degrees)	Gyro(degrees)
1	705	880	-32
2	713	889	-32
3	713	888	-31
4	712	890	-32
5	713	889	-31
6	712	890	-34
7	713	889	-32
8	713	888	-33
9	713	889	-32
10	712	888	-33
11	714	888	-33
12	711	889	-32
13	712	888	-32
14	713	888	-32
15	714	889	-32
16	713	887	-32
17	712	890	-33
18	714	890	-32
19	713	890	-33
20	711	889	-33

Table 6: *End pose readings for arc at left*

RIGHT RUN			
S.no	Left Wheel(degrees)	Right Wheel(degrees)	Gyro(degrees)
1	889	719	29
2	890	717	29
3	891	717	29
4	890	716	30
5	889	715	29
6	892	716	31
7	890	715	29
8	891	715	31
9	890	716	30
10	893	716	30
11	891	715	31
12	890	715	30
13	891	716	29
14	890	716	29
15	889	714	29
16	890	714	30
17	891	715	29
18	891	716	28
19	889	715	29
20	889	714	30

Table 7: *End pose readings for arc at right*

1.4.3 End pose estimation from wheel encoders

- We find the end poses from the encoder readings using differential drive kinematics which we studied in AMR.
  $\Delta U_l = \frac{\pi * \text{Diameter of the wheel} * \text{Degrees of rotation of left wheel}}{360}$
  where ΔU_l = Distance travelled by left wheel
  $\Delta U_r = \frac{\pi * \text{Diameter of the wheel} * \text{Degrees of rotation of right wheel}}{360}$
  where ΔU_r = Distance travelled by right wheel
  $\Delta U = \frac{\Delta U_l + \Delta U_r}{2}$ = Center translation
  $\Delta\theta = \frac{\Delta U_r - \Delta U_l}{b}$ = Center rotation
  where b = distance between the wheels.
  New orientation:
  $$\theta_i = \theta_{i-1} + \Delta\theta \tag{10}$$
  New position:
  $$x_i = x_{i-1} + \Delta U_i cos\theta_i \tag{11}$$
  $$y_i = y_{i-1} + \Delta U_i sin\theta_i \tag{12}$$
- Tables 8, 9, 10 shows the sensor end pose readings for the left, right and straight motion of the robot respectively

STRAIGHT RUN			
S.no	X(cm)	Y(cm)	Angle(degrees)
1	43.5	-0.2	-0.3
2	43.6	0.1	0.1
3	43.5	-0.6	-0.8
4	43.5	-0.1	-0.1
5	43.6	-0.2	-0.3
6	43.6	-0.6	-0.8
7	43.5	0.2	0.3
8	43.5	0.2	0.3
9	43.6	0.1	0.1
10	43.5	0.1	0.1
11	43.5	0.1	0.1
12	43.5	-0.6	-0.8
13	43.5	-0.3	-0.3
14	43.5	-0.1	-0.1
15	43.5	-0.1	-0.1
16	43.6	-0.2	-0.3
17	43.6	0.1	0.1
18	43.6	0.3	0.3
19	43.4	0.1	0.1
20	43.3	-0.3	-0.3

Table 8: *End pose sensor readings for straight motion*

LEFT RUN			
S.no	X(cm)	Y(cm)	Angle(degrees)
1	33.5	-19.4	30.1
2	33.8	-19.7	30.2
3	33.8	-19.5	30.1
4	33.7	-19.9	30.5
5	33.8	-19.7	30.2
6	33.7	-19.9	30.5
7	33.8	-19.7	30.2
8	33.8	-19.5	30.1
9	33.8	-19.7	30.2
10	33.7	-19.6	30.2
11	33.9	-19.5	29.8
12	33.6	-19.8	30.5
13	33.7	-19.6	30.2
14	33.8	-19.6	30.1
15	33.9	-19.6	29.9
16	33.8	-19.4	29.8
17	33.7	-19.9	30.5
18	33.8	-19.7	30.2
19	33.7	-19.8	30.4
20	33.8	-19.7	30.2

Table 9: *End pose sensor readings for arc at left*

RIGHT RUN			
S.no	X(cm)	Y(cm)	Angle(degrees)
1	34.3	19.2	-29.2
2	34.1	19.5	-29.7
3	34.2	19.5	-29.9
4	34.3	19.6	-29.8
5	33.9	19.5	-29.8
6	33.9	19.7	-30.2
7	33.9	19.6	-30.1
8	33.9	19.7	-30.2
9	34.1	19.5	-29.8
10	33.8	19.9	-30.4
11	33.9	19.7	-30.2
12	33.9	19.6	-29.9
13	33.9	19.6	-29.7
14	34.1	19.5	-29.8
15	33.9	19.6	-30.1
16	33.8	19.7	-30.2
17	33.9	19.7	-30.2
18	33.7	19.6	-30.1
19	33.6	19.5	-29.8
20	33.9	19.6	-30.4

Table 10: *End pose sensor readings for arc at right*

1.5 Visualization of End Poses

- Figure 15 resemble a cluster of the multiple readings measured by ruler of each motion which shows that the readings are precise.

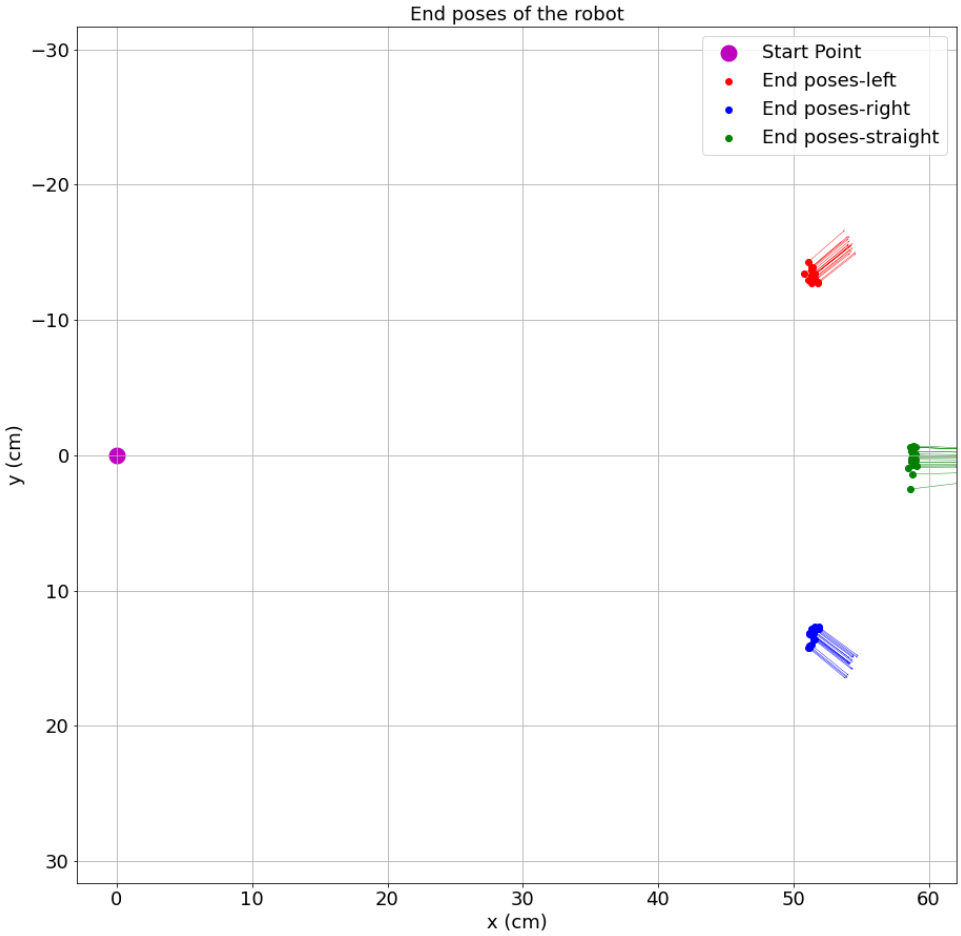

Figure 17: *End pose reading for the left, right and straight run of the robot for ruler measurements*

- Figure 16,17,18 shows the individual end poses of the left, right and straight motion respectively for ruler measurement.

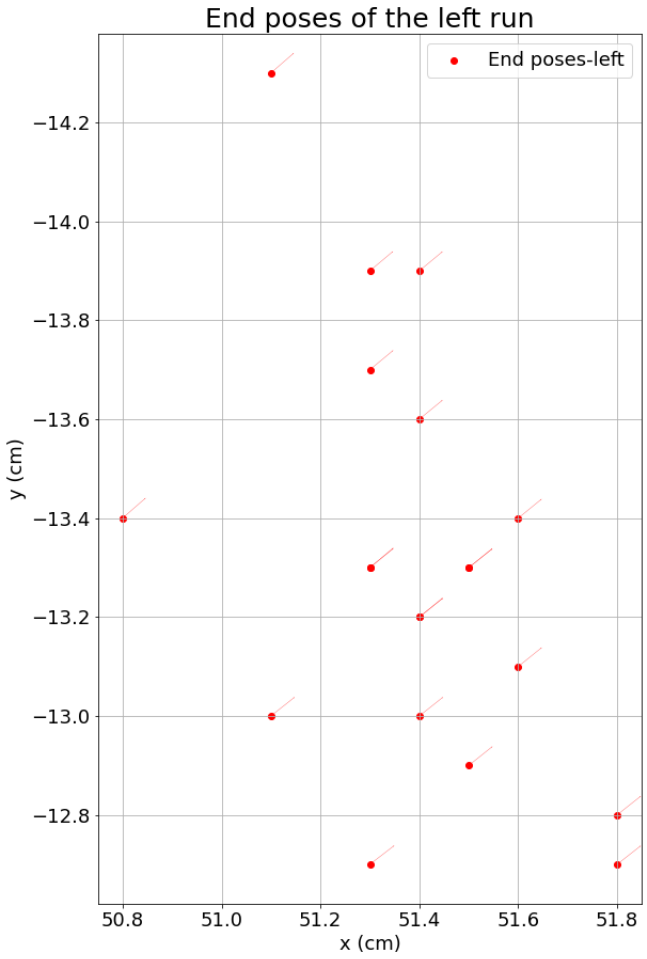

Figure 18: *End poses of the left run for ruler measurements*

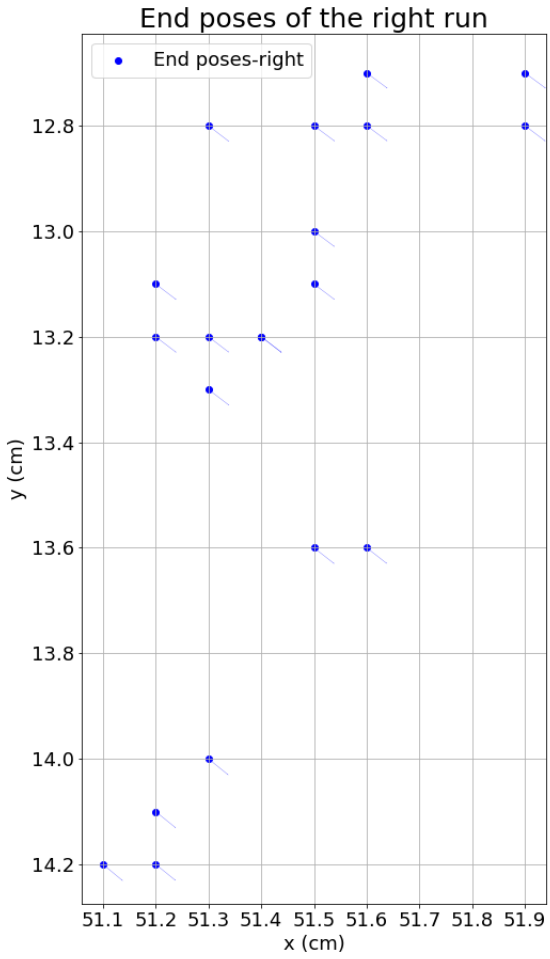

Figure 19: *End poses of the right run for ruler measurements*

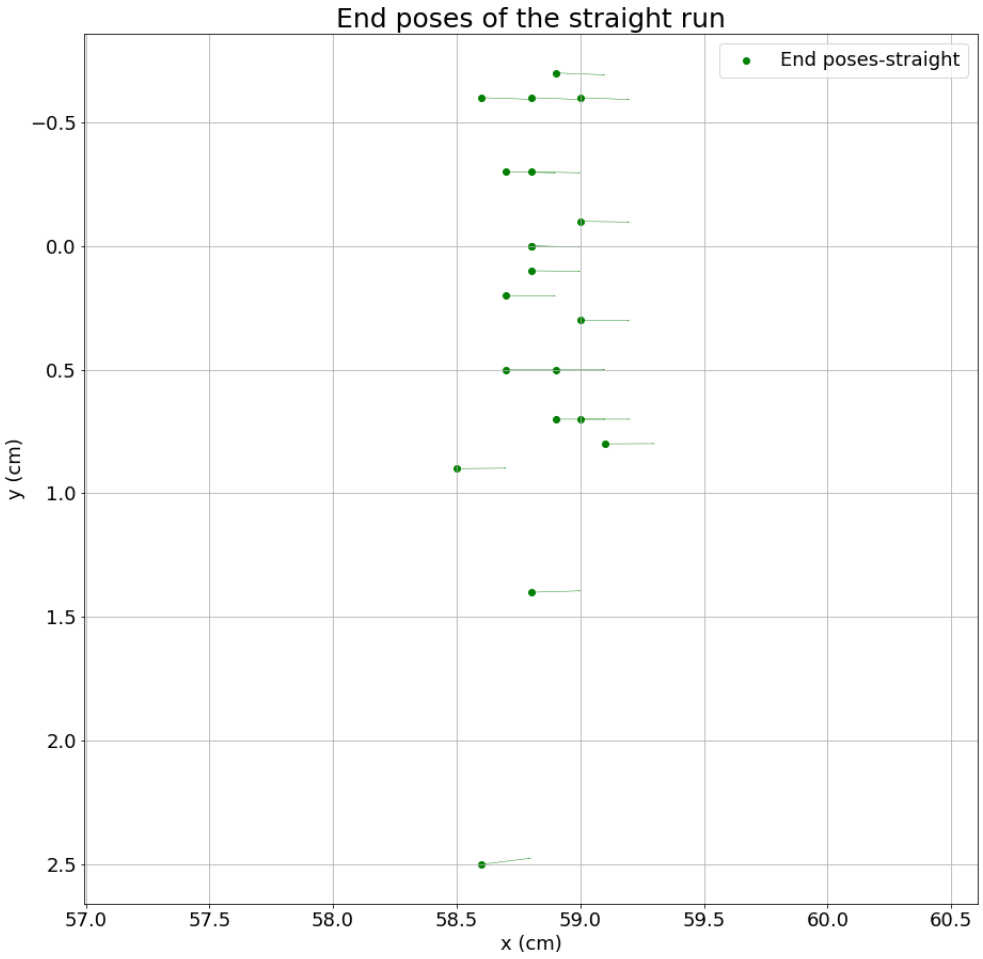

Figure 20: *End poses of the straight run for ruler measurements*

- Figure 19 resemble a cluster of the multiple readings measured by encoder of each motion.

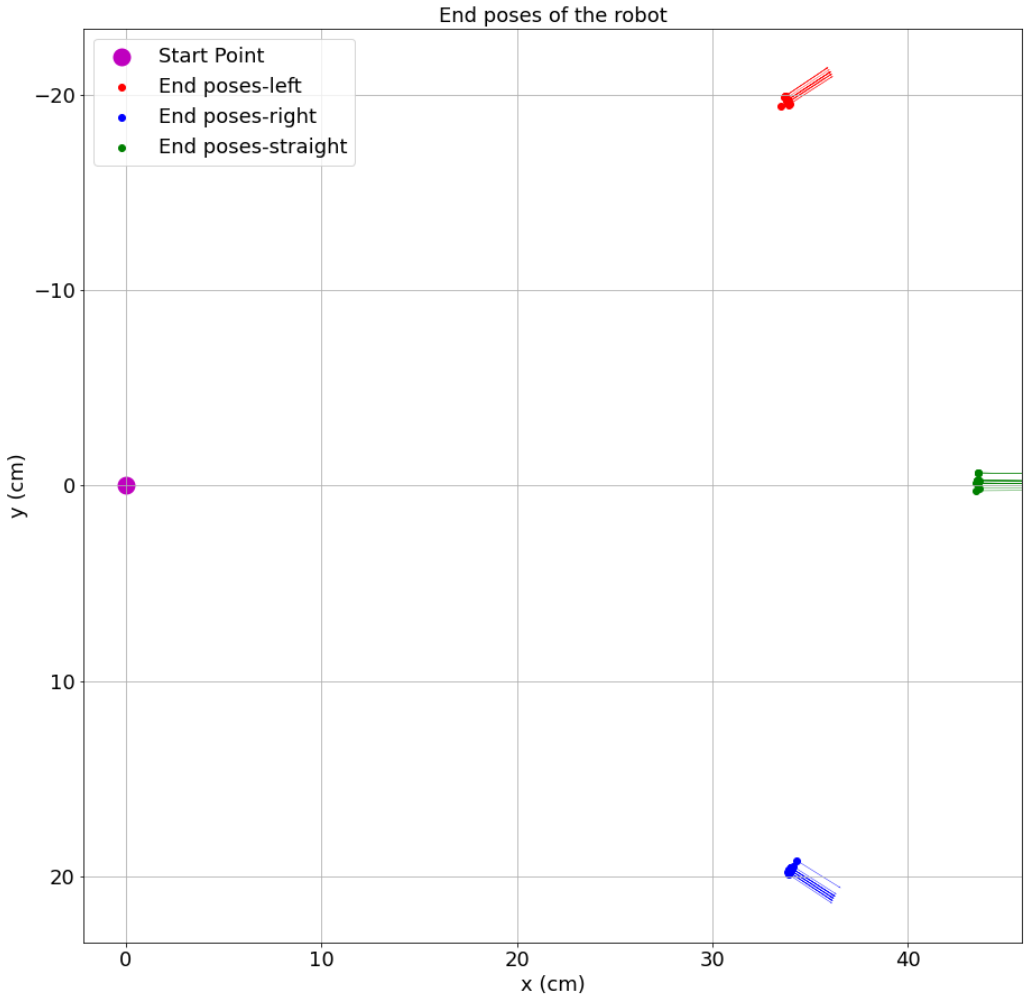

Figure 21: *End pose reading for the left, right and straight run of the robot for encoder measurements*

- Figure 20,21,22 shows the individual end poses of the left, right and straight motion respectively for encoder measurement.

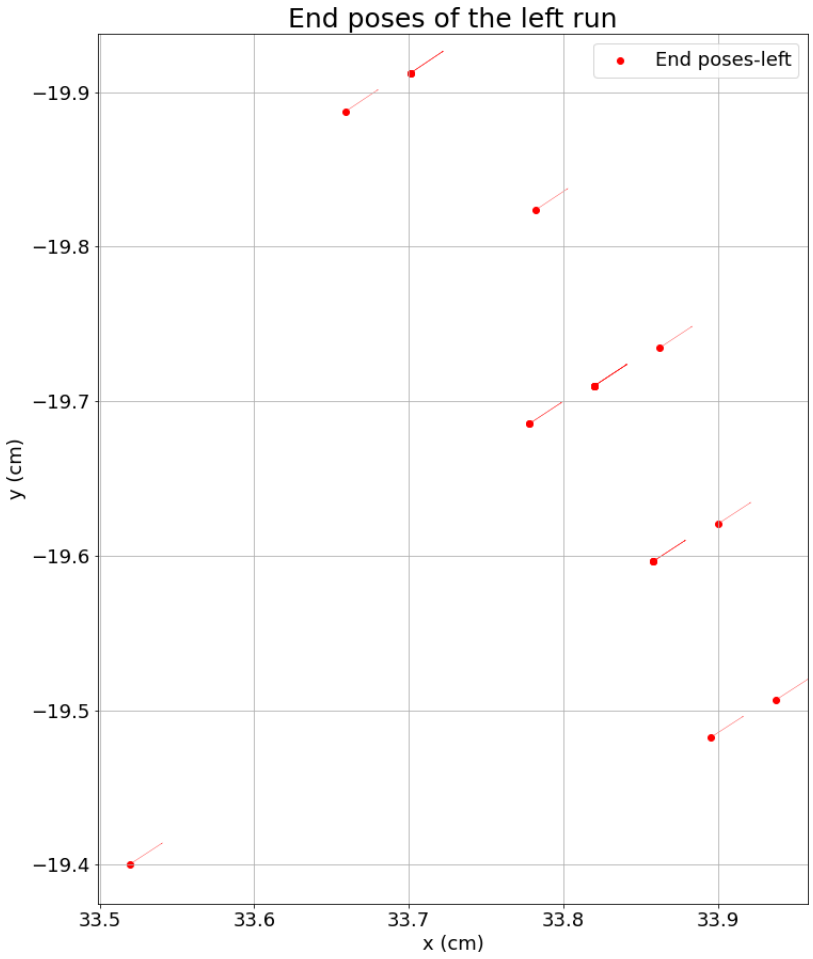

Figure 22: *End poses of the left run for encoder measurements*

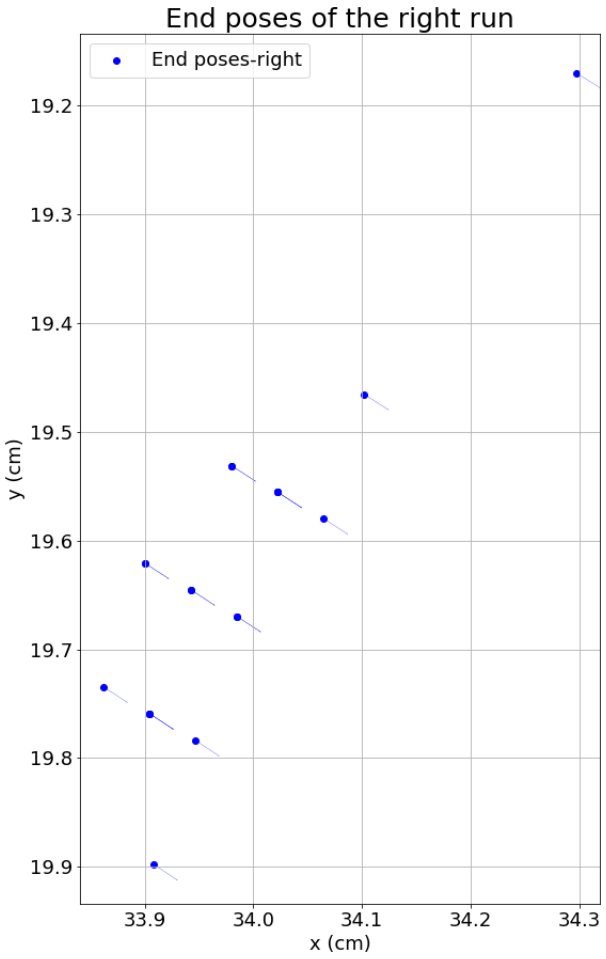

Figure 23: *End poses of the right run for encoder measurements*

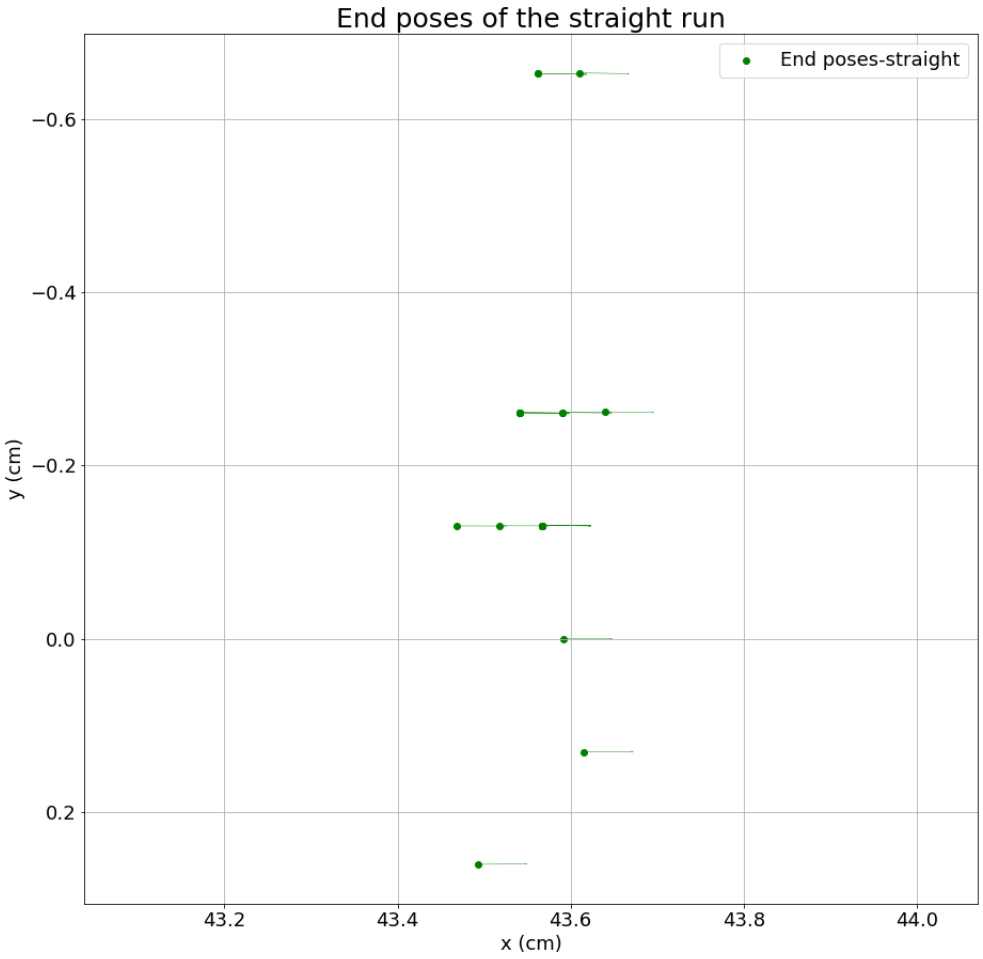

Figure 24: *End poses of the straight run for encoder measurements*

1.6 Data visualization for combined ruler data

- Figure 23 resembles a cluster of the multiple readings for the combined data measured by ruler for each motion.

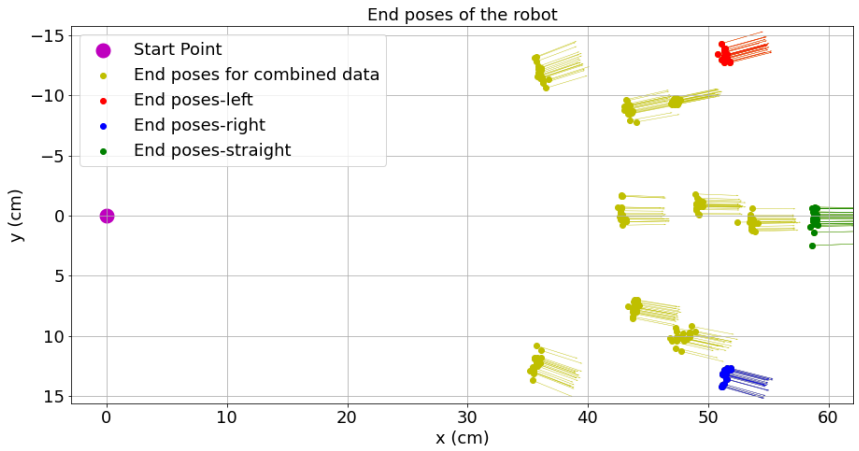

Figure 25: *End poses of the robot as a cluster for combined ruler data*

- Figure 24,25,26 shows the individual end poses of the left, right and straight motion respectively for combined data for ruler measurement.

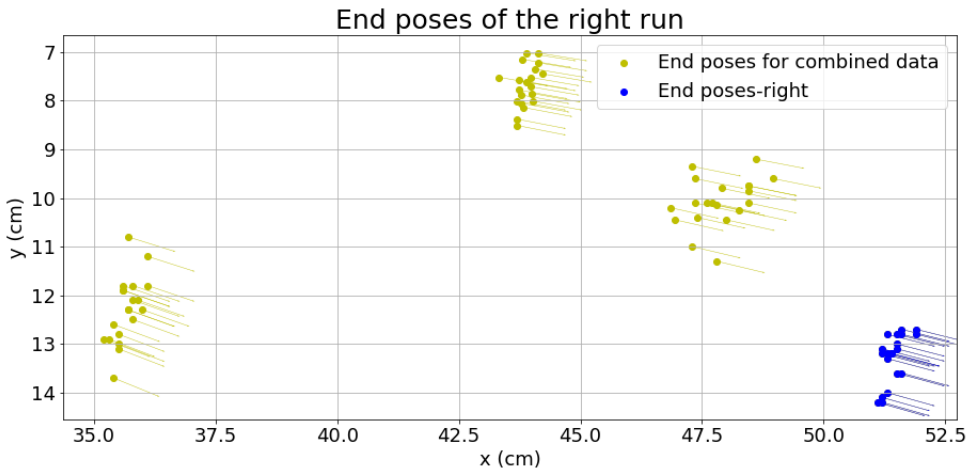

Figure 26: *End poses due to right side arc for combined data*

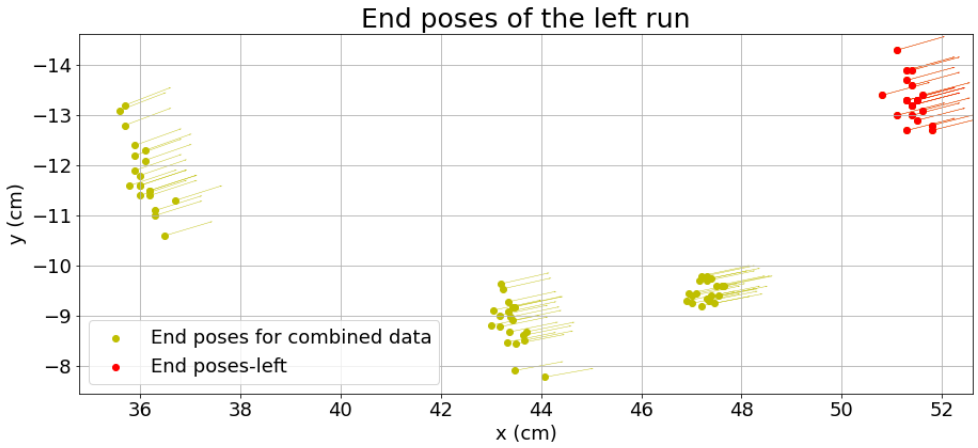

Figure 27: *End poses due to left side arc for combined data*

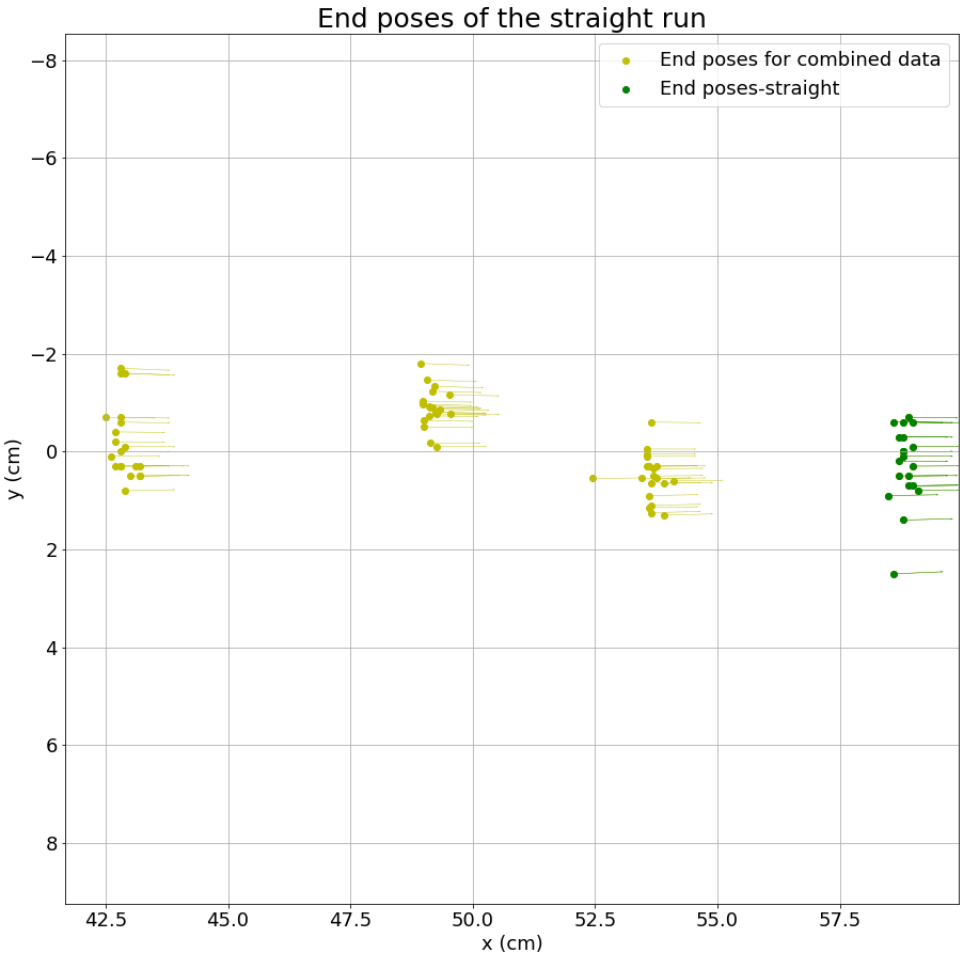

Figure 28: *End poses due to straight motion for combined data*

1.7 Data visualization for combined encoder data

- Figure 29 resembles a cluster of the multiple readings for the combined data measured by encoder for each motion.

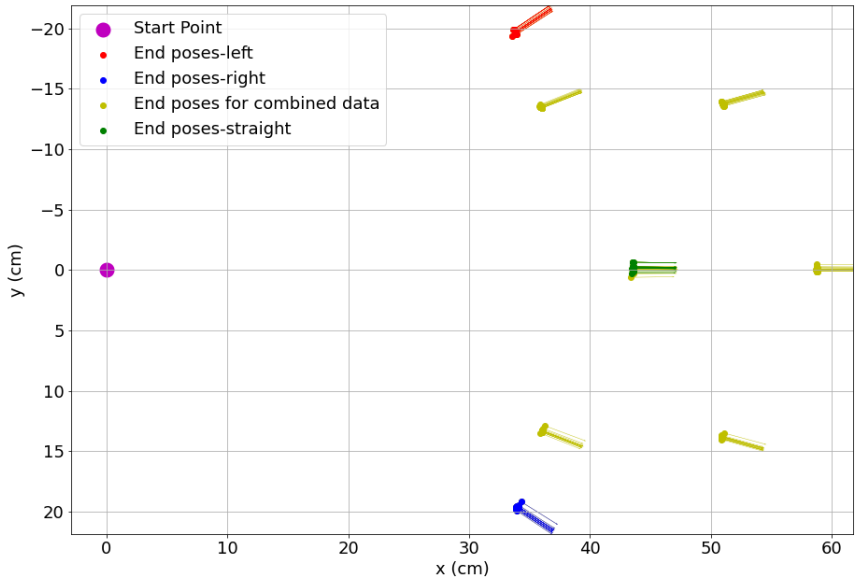

Figure 29: *End poses of the robot as a cluster for combined encoder data*

- Figure 30,31,32shows the individual end poses of the left, right and straight motion respectively for combined data for encoder measurement.

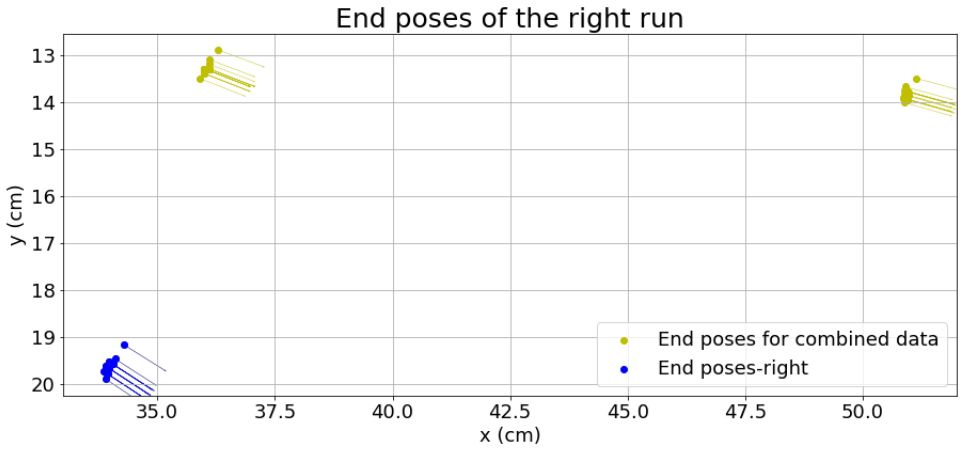

Figure 30: *End poses due to right side arc for combined encoder data*

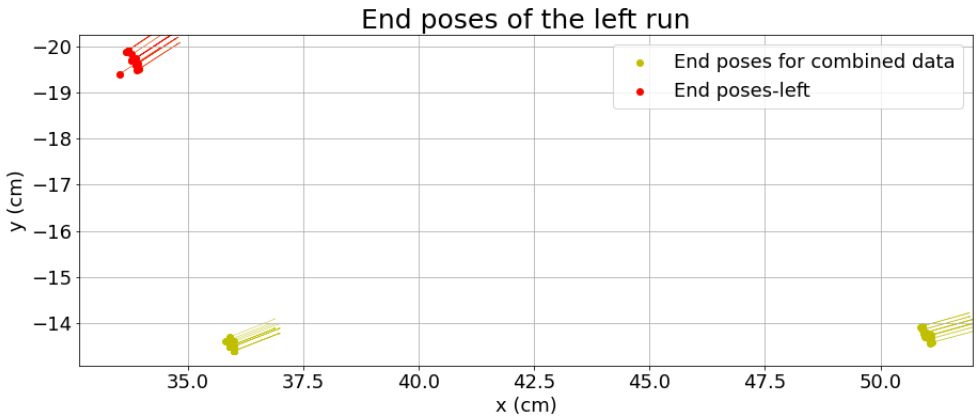

Figure 31: *End poses due to left side arc for combined encoder data*

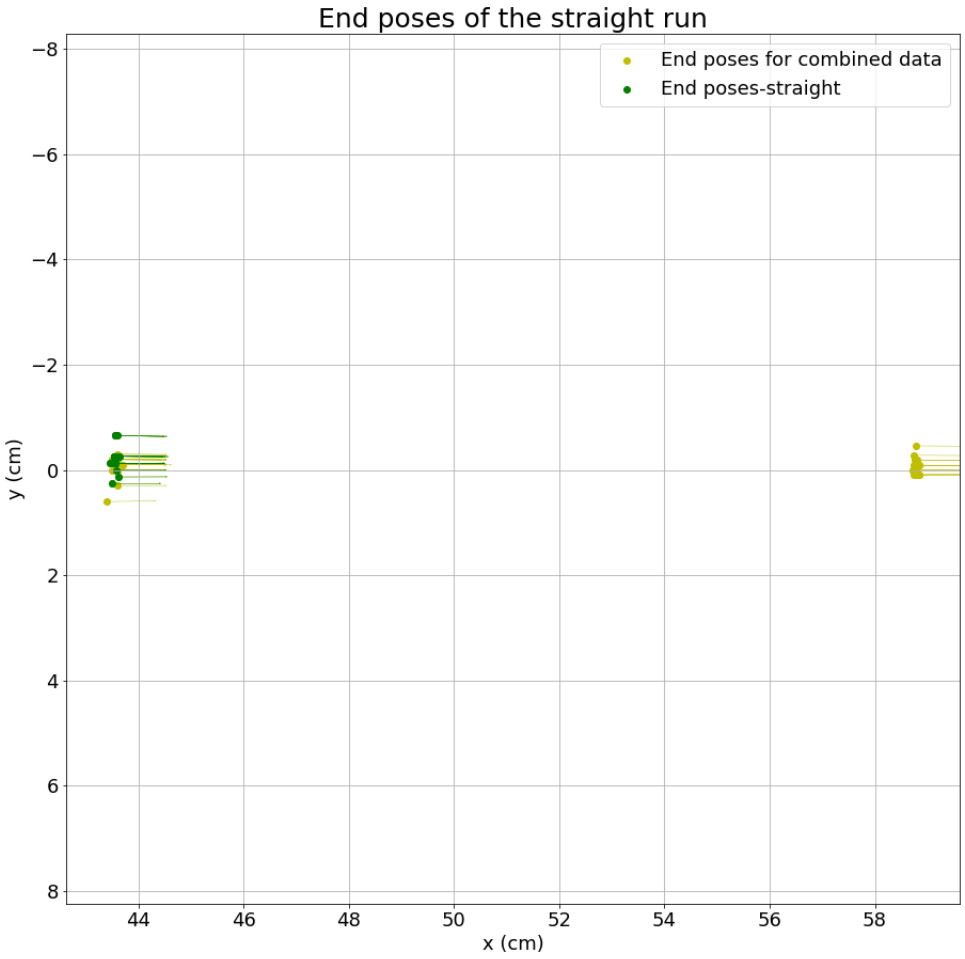

Figure 32: *End poses due to straight motion for combined encoder data*

- As seen from the above graphs, measured data of each group tend to be different from each other. We do not have specifications of the robot parts that they used. Possible reasons for this difference in measurement would be due to:
  - The distance between the two markers in their robot would

be different for different teams based on the way they constructed the robot.

- Wheel diameter for each group might be different, which would lead to different degrees of rotations.
- Internal clock frequency of the BRIC might have been set differently for each groups.
- Based on the type of grid sheet used, friction between paper and the wheel would induce different levels of slippage for different teams

2 Statistical Evaluation

2.1 Data pre-processing for combined ruler measurements dataset

- Initially We had combined the ruler measurements datasets provided by 4 groups and hence we had a total of 80 samples (20 from each group) for each run.
- However, the measurement data of each group was very dissimilar. Hence, we have taken only our data (20 samples) for all the operations.
- Bounds of Chebyshev theorem is used to remove the outliers.
- Chebyshev Theorem states that only a certain amount of data points in a probability distribution can be present from a particular distance from the mean of the distribution. and is given by:

$$P(|X - \mu| \leq K\sigma) \geq (1 - \frac{1}{K^2}) \tag{13}$$

$$P(\mu - 2\sigma < X < \mu + 2\sigma) \geq \frac{3}{4} = 0.75 \tag{14}$$

- We have taken the threshold as 2 standard deviations from the mean. Points outside the threshold are taken as outliers.
- We estimate the random variable X between upper and lower bounds having mean μ and standard deviation σ. We choose the random variable to be end-pose co-ordinates and orientation for each motion.
- Table 25 represents the outliers values for our ruler measurement data.

S.No	Direction	Random Variable	Total no.of data	No.of outliers
1	Left	X	20	1
2	Left	Y	20	1
3	Left	Theta	20	2
4	Right	X	20	2
5	Right	Y	20	0
6	Right	Theta	20	1
7	Straight	X	20	1
8	Straight	Y	20	1
9	Straight	Theta	20	1

Table 11: *Outliers for our ruler measurement data*

2.2 Fitting Gaussian

- Figure 33,34,35 represents the Gaussian fitting plot for left, right and straight motion respectively for our ruler measurement dataset.

Gaussian fit for the left motion

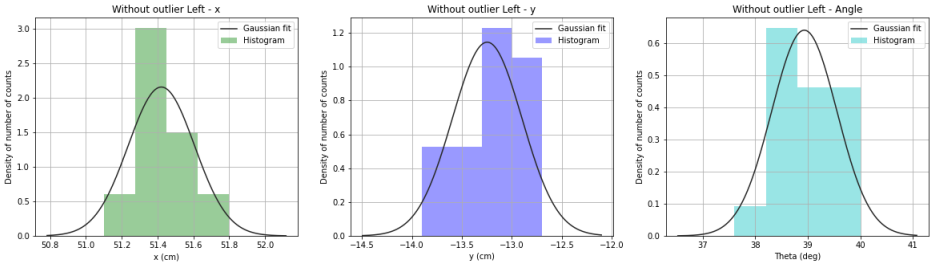

Figure 33: *Gaussian fit for left arc*

Gaussian fit for the right motion

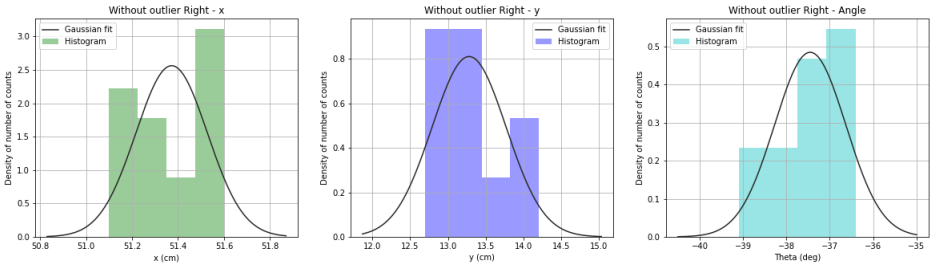

Figure 34: *Gaussian fit for right arc*

Gaussian fit for the straight motion

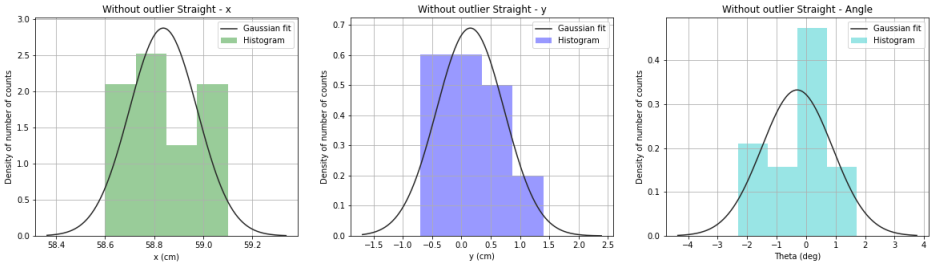

Figure 35: *Gaussian fit for straight motion*

2.3 Fitting Gaussian after PCA

- PCA is performed for dimensionality reduction.
- Figure 36,37,38 represents the Gaussian fitting plot for left, right and straight motion respectively after PCA.

Gaussian fit for the left motion

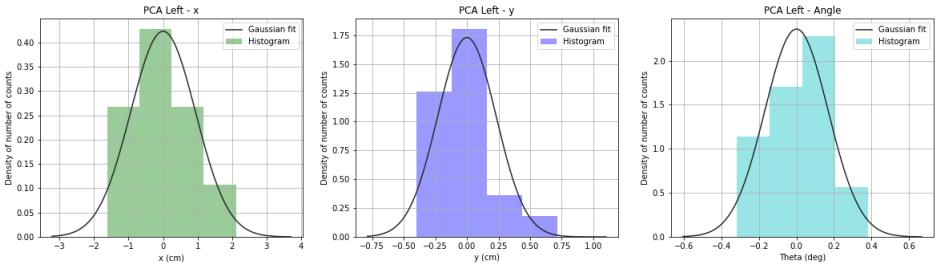

Figure 36: *Gaussian fit after PCA for left arc*

Gaussian fit for the right motion

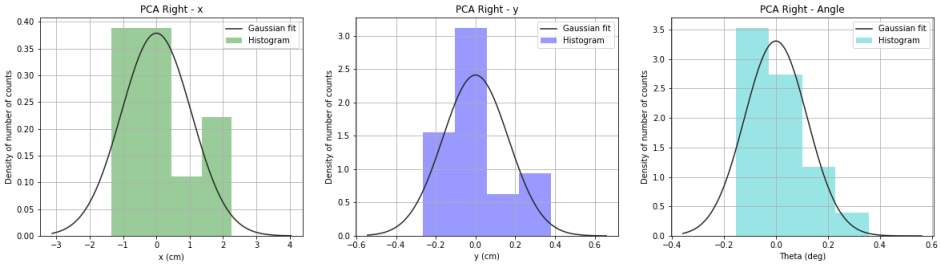

Figure 37: *Gaussian fit after PCA for right arc*

Gaussian fit for the straight motion

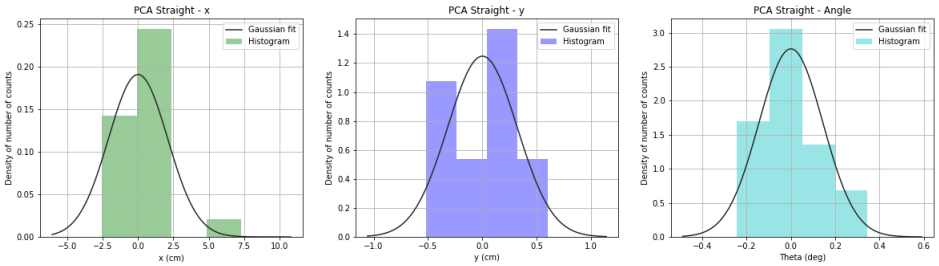

Figure 38: *Gaussian fit after PCA for straight motion*

2.4 Uncertainty ellipse

- Figure 39,40,41 represents the uncertainty ellipse of the robots left, right and straight motion after PCA respectively for our ruler dataset.

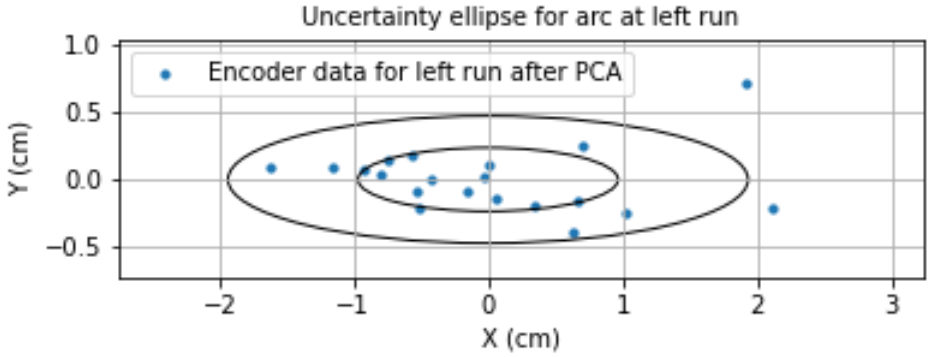

Figure 39: *Uncertainty ellipse for arc at left*

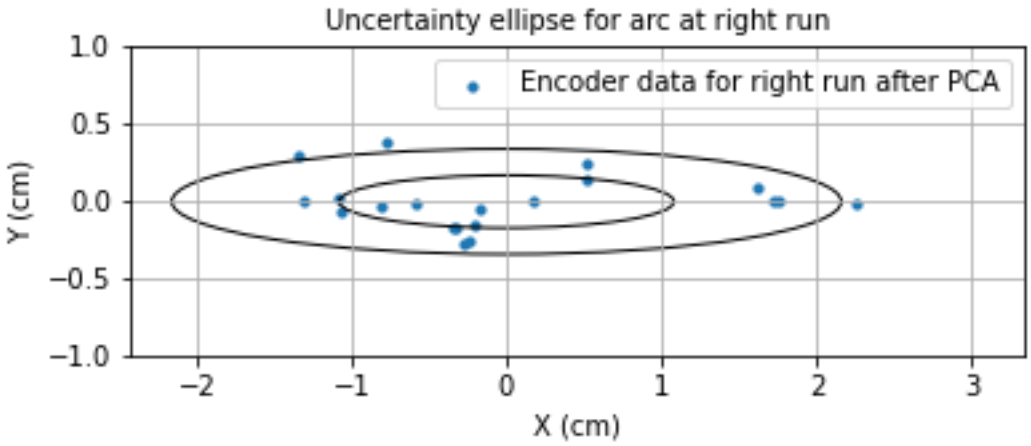

Figure 40: *Uncertainty ellipse for arc at right*

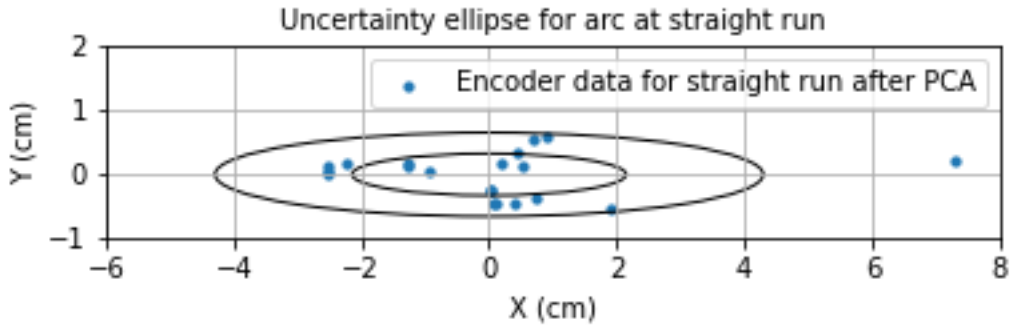

Figure 41: *Uncertainty ellipse for arc straight*

2.5 Softwares used

- Numpy
- Pandas
- Matplotlib
- Seaborn
- sklearn

2.6 Analysing the data

- Accuracy and Precision of measured data will determines the robot's consistency and accuracy.

2.7 Statistical parameters

- Chi-square is performed using python and scipy library.
- Chi square test is used to obtain information whether the data fits the gaussian distribution.
- The Chi square function returns two parameters which are chi sqaure test statistics and P-value.
- P-value provides the strength with which null hypothesis is supported.
- Chi sqaure test statistics measure the deviation of actual results from the expected results.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{15}$$

  χ^2 = chi squared

  O_i = observed value

  E_i = expected value

- According to Fisher's test significance level is taken as 0.05. This parameter is used to determine whether the dataset fits the Gaussian distribution.
- We have measured the accuracy by finding the average of absolute deviation of the measurements from the mean value.
- We estimated precision as mean plus or minus average absolute deviation of measurements from the mean value.
- Table 29 shows the statistical parameter values for our ruler measurement data.

```
import scipy.stats as stats

def chi_square(data):
    obs_data = np.array(data, dtype = np.float)
    obs_freq,_ = np.histogram(obs_data, bins=5)
    mean = np.mean(obs_data)
    std = np.std(obs_data)
```

```
    exp_data = np.random.normal(mean,std,len(data))
    exp_freq,_ = np.histogram(exp_data,
     ↪  bins=len(obs_freq))
    return stats.chisquare(obs_freq,exp_freq)
```

S.No	Direction	Random Variable	Mean (cm)	Variance (cm)	Precision (cm)	Accuracy (cm)	Chi value	P value	Is Gaussian
1	Left	X	51.42	0.43	51.42 +/- 0.14	0.14	4.05	0.39	Yes
2	Left	Y	-13.24	0.59	-13.24 +/- 0.27	0.27	6.83	0.14	Yes
3	Left	Theta	38.93	0.78	38.93 +/- 0.52	0.52	1.91	0.75	Yes
4	Right	X	51.37	0.39	51.37 +/- 0.13	0.13	57.37	0.00	No
5	Right	Y	13.27	0.71	13.27 +/- 0.41	0.41	6.25	0.18	Yes
6	Right	Theta	-37.45	0.91	-37.45 +/- 0.65	0.65	inf	0.00	No
7	Straight	X	58.83	0.37	58.83 +/- 0.11	0.11	27.23	0.00	No
8	Straight	Y	0.15	0.76	0.15 +/- 0.48	0.48	28.41	0.00	No
9	Straight	Theta	-0.31	1.09	-0.31 +/- 1.02	1.02	5.22	0.26	Yes

Table 12: *Statistical parameters for our ruler measurement data*

2.8 Data pre-processing for our encoder measurements dataset

- Initially we had combined the encoder measurements datasets provided by 3 groups and had a total of 60 samples (20 from each group) for each run.
- However, due to dissimilarity present in each groups data, we have decided to go ahead with just 20 samples of our encoder data.
- We find and remove the outliers like we did for the ruler dataset using lower bounds of Chebyshev Theorem.
- Table 13 represents the outliers values for our encoder measurement data.

S.No	Direction	Random Variable	Total no.of data	No.of outliers
1	Left	X	20	1
2	Left	Y	20	1
3	Left	Theta	20	0
4	Right	X	20	1
5	Right	Y	20	1
6	Right	Theta	20	1
7	Straight	X	20	1
8	Straight	Y	20	1
9	Straight	Theta	20	1

Table 13: *Outliers for the our encoder measurement data*

2.9 Fitting Gaussian

- Figure 88,89,90 represents the Gaussian fitting plot for left, right and straight motion respectively for our encoder measurement dataset.

Gaussian fit for the left motion

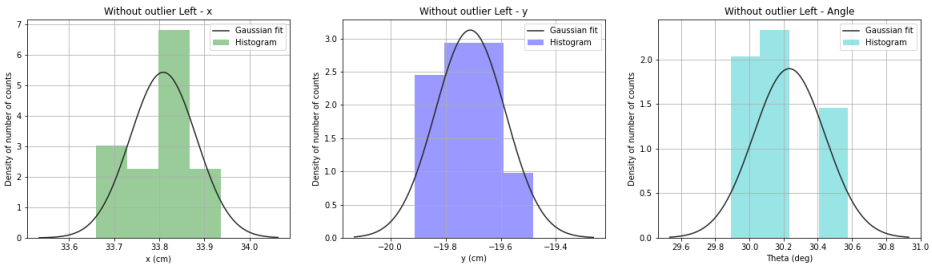

Figure 42: *Gaussian fit for left arc*

Gaussian fit for the right motion

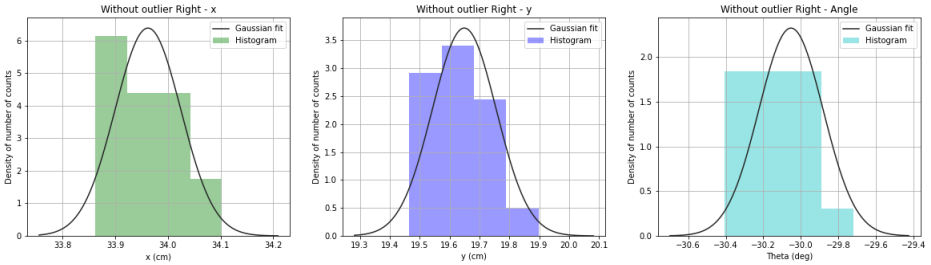

Figure 43: *Gaussian fit for right arc*

Gaussian fit for the straight motion

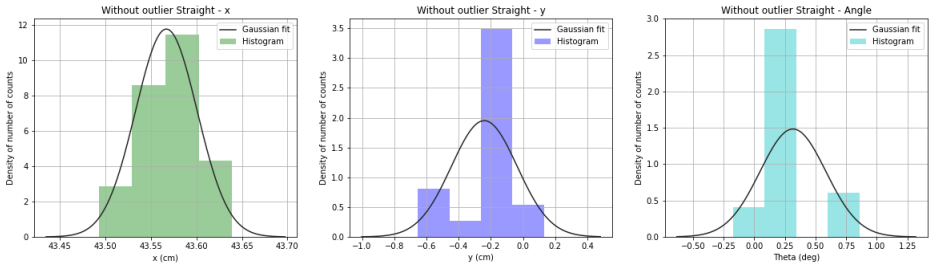

Figure 44: *Gaussian fit for straight motion*

2.10 Fitting Gaussian after PCA

- PCA is performed for dimensionality reduction.
- Figure 25,26 and 27 represents the Gaussian fitting plot for left, right and straight motion respectively after PCA for our encoder measurement data.

Gaussian fit for the left motion

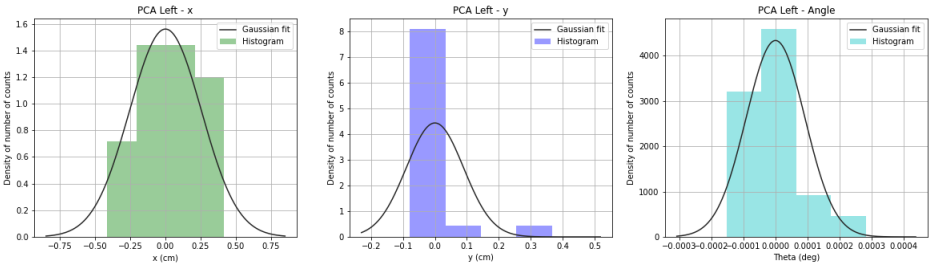

Figure 45: *Gaussian fit after PCA for left arc*

Gaussian fit for the right motion

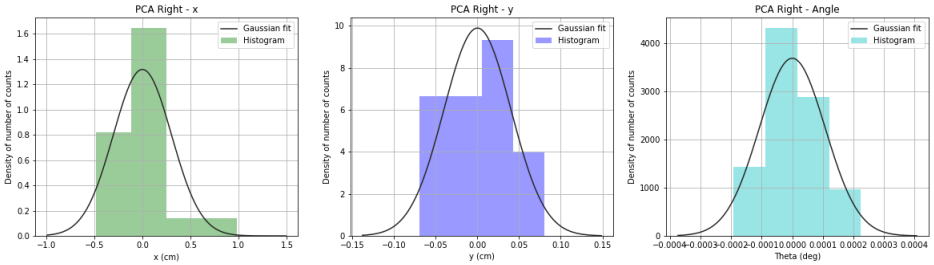

Figure 46: *Gaussian fit after PCA for right arc*

Gaussian fit for the straight motion

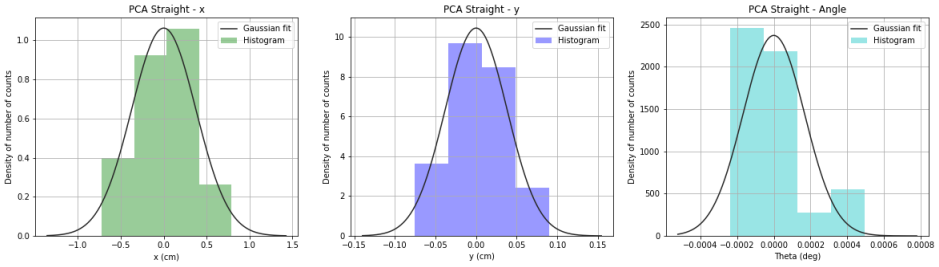

Figure 47: *Gaussian fit after PCA for straight motion*

2.11 Uncertainty ellipse

- Figure 106,107,108 represents the uncertainty ellipse of the robots left, right and straight motion after PCA respectively for our encoder measurement dataset.

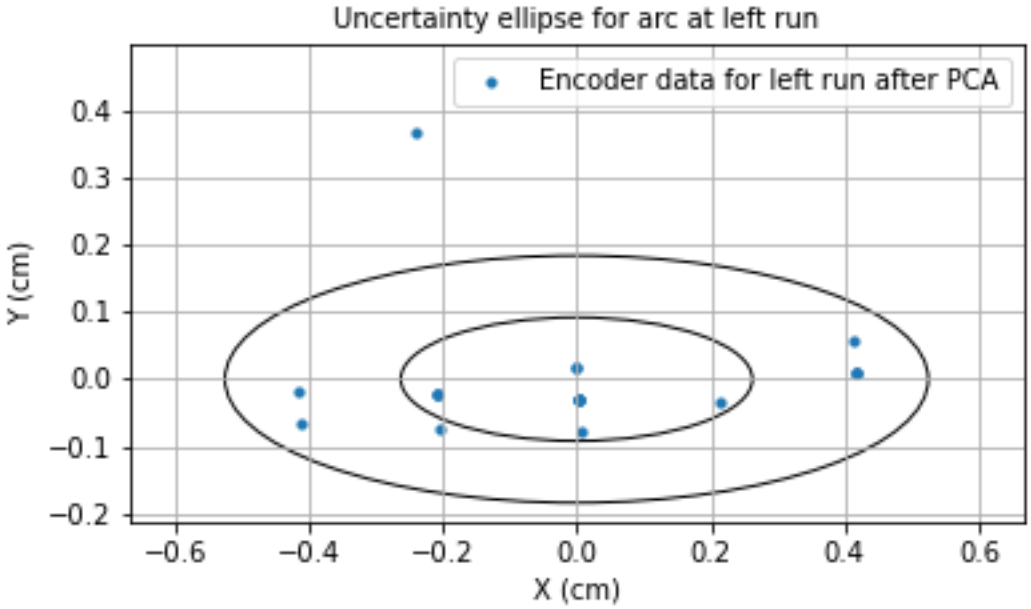

Figure 48: *Uncertainty ellipse for arc at left - Encoder*

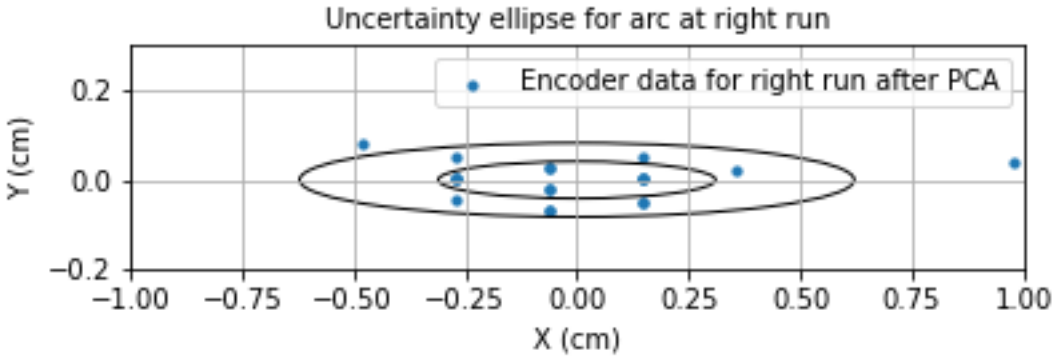

Figure 49: *Uncertainty ellipse for arc at right - Encoder*

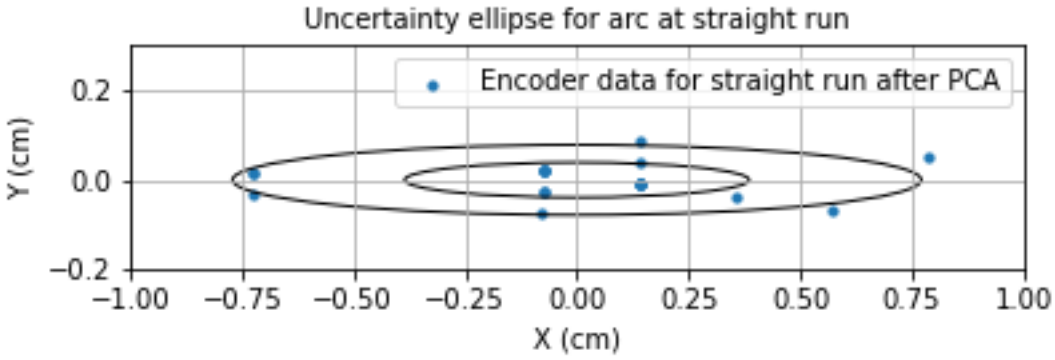

Figure 50: *Uncertainty ellipse for arc straight - Encoder*

2.12 Statistical parameters of our encoder measurement data

- Chi square test is used to obtain information whether the data fits the gaussian distribution.
- Table 14 shows the statistical parameter values for our encoder measurement data.

S.No	Direction	Random Variable	Mean (cm)	Variance (cm)	Precision (cm)	Accuracy (cm)	Chi value	P value	Is Gaussian
1	Left	X	33.8	0.27	33.8 +/- 0.05	0.05	13.33	0.00	No
2	Left	Y	-19.71	0.35	-19.71 +/- 0.09	0.09	23.16	0.00	No
3	Left	Theta	30.23	0.45	30.23 +/- 0.15	0.15	18.97	0.00	No
4	Right	X	33.96	0.24	33.96 +/- 0.05	0.05	14.58	0.00	No
5	Right	Y	19.64	0.32	19.64 +/- 0.08	0.08	2.83	0.58	Yes
6	Right	Theta	-30.05	0.41	-30.05 +/- 0.13	0.13	19.78	0.00	No
7	Straight	X	43.56	0.18	43.56 +/- 0.02	0.02	3.2	0.52	Yes
8	Straight	Y	-0.24	0.45	-0.24 +/- 0.14	0.14	8.88	0.06	Yes
9	Straight	Theta	0.31	0.51	0.31 +/- 0.19	0.19	13.38	0.00	No

Table 14: *Statistical parameters for our encoder measurement data*

3 Camera Calibration

3.1 Aim

The aim of this experiment is to calibrate the given Microsoft Lifecam HD camera using MATLAB camera calibration toolbox. The camera parameters will also be obtained from this experiment.

3.2 Calibration Setup

- Hardware components required
  - Microsoft lifecam studio
  - Tripod with camera mounting mechanism
  - Checkerboard with known square dimension and grid shape
    - Square dimension: 25 mm
    - Grid shape: 7×5
- Software components required
  - MATLAB R2020b
  - Camera Calibration Toolbox for MATLAB by Jean-Yves Bouguet, Caltech
  - GTK UVC video viewer software

3.3 Introduction

- Camera calibration estimates the parameters of a lens and image sensor of an image or video camera. These parameters can be used to correct the distortion of the lens, determine the camera location in the scene or to measure the size of an object in world units.
- Pinhole cameras are the cameras which will have tiny aperture without any lens. This type of camera is used in this experiment.
- Camera parameters obtained are intrinsic and extrinsic parameters which is represented in figure 51. The extrinsic parameter maps the world co-ordinate to camera co-ordinate whereas the

intrinsic parameter maps the camera co-ordinate to image co-ordinate.

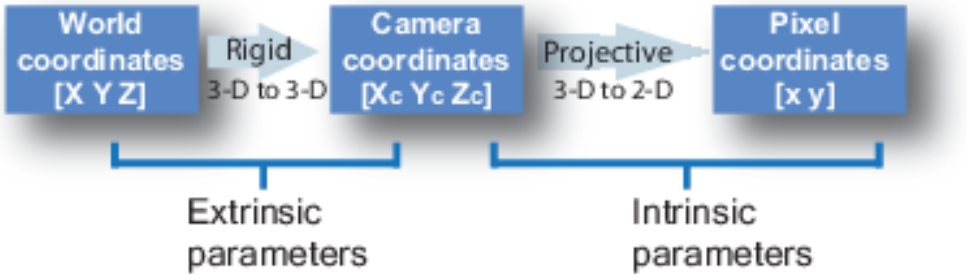

Figure 51: *Camera parameters (source: https://de.mathworks.com/help/vision/ug/camera-calibration.html)*

- Intrinsic parameters consists of internal parameters such as focal length, principal point, skew co.efficient and distortions. The camera intrinsic matrix is given as, K= $\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$
- There are two types of distortions. They are radial distortions and tangential distortions.
- Radial distortions seems to be occurred when the rays of the light bend more near the edges of a lens compared to the optical center. The smaller the size of the lens, the greater the distortion. Radial distortion is further classified as negative, no and positive radial distortion. [Figure 52]

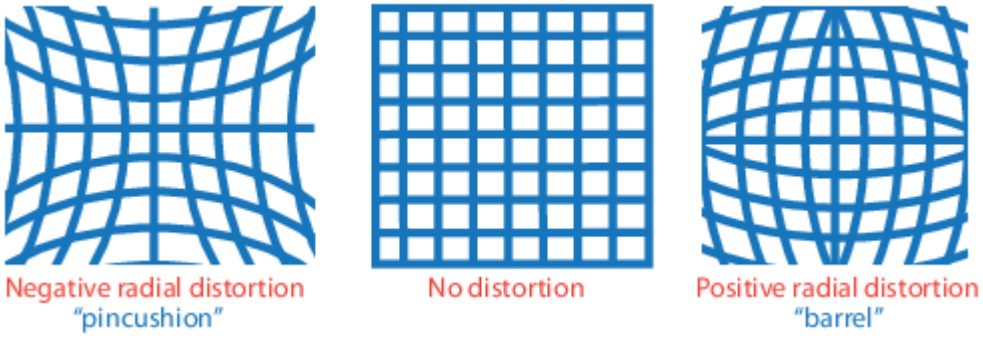

Figure 52: *Radial distortions in camera calibration (source: https://de.mathworks.com/help/vision/ug/camera-calibration.html)*

- The radial distortion coefficients is formulated as,

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r_4 + k_3 r^6)$$
$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

- Tangential distortion seems to be occurred when lens and image plane are not in parallel. Tangential distortion is further classified as zero tangential distortion and tangential distortion. [Figure 53]

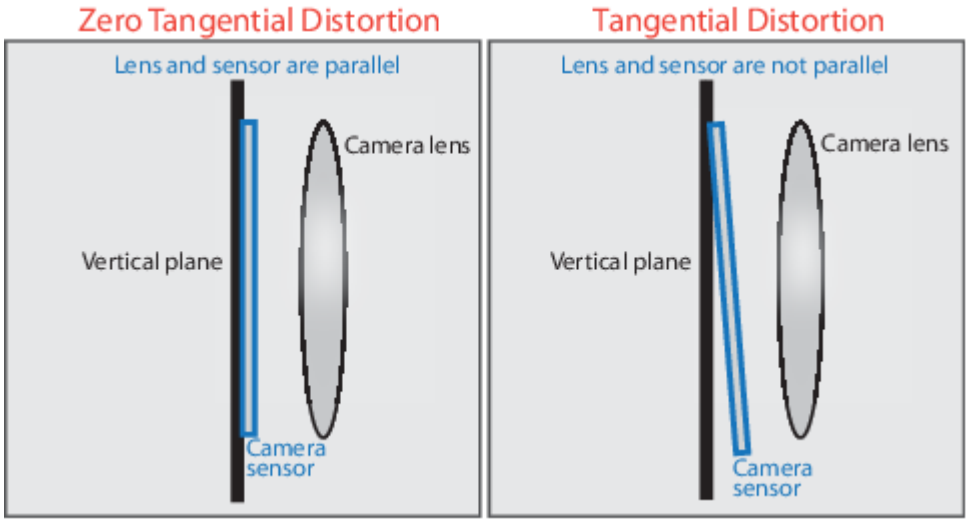

Figure 53: *Tangential distortions in camera calibration (source: https://de.mathworks.com/help/vision/ug/camera-calibration.html)*

- The tangential distortion coefficients is formulated as,

$$x_{distorted} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$
$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

- Extrinsic parameters corresponds to rotation and translation. The origin is at optical centre. The formulation of the parameters is mentioned below,

$$b = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \approx \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{K} \underbrace{\begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix}}_{B} \underbrace{\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}}_{C} \tag{16}$$

where, A is intrinsic parameter, B is extrinsic parameter and C is scene coordinate system.

- In equation 12, $B \cdot C$ represents the camera co.ordinate system. [Figure 54]

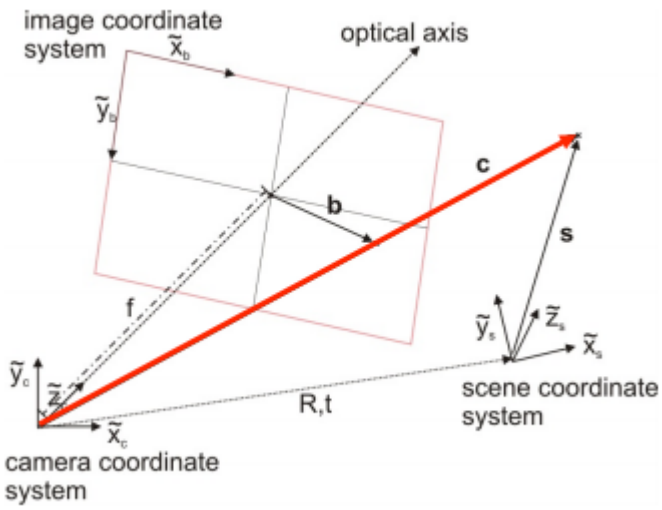

Figure 54: *Camera coordination system (source: http://ais.informatik.uni-freiburg.de/teaching/ws10/robotics2/pdfs/rob2-10-camera-calibration.pdf)*

- In equation 12, $A{\cdot}B{\cdot}C$ represents the image co.ordinate system. [Figure 55]

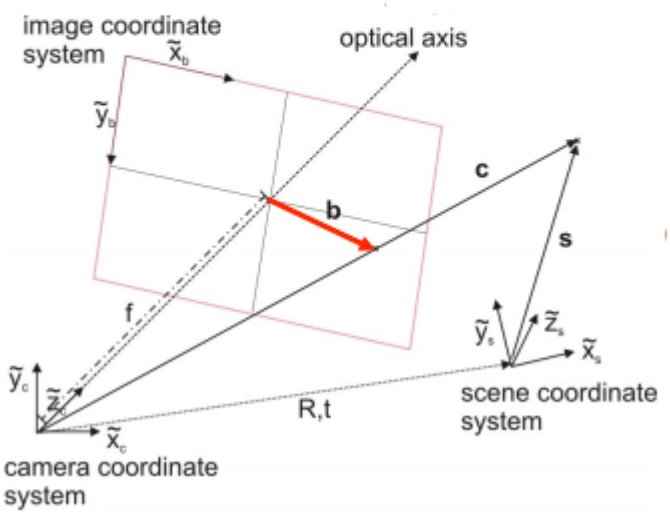

Figure 55: *Image coordinate system (source: http://ais.informatik.uni-freiburg.de/teaching/ws10/robotics2/pdfs/rob2-10-camera-calibration.pdf)*

3.4 Procedure- Description of calibration, images number and position

- For camera calibration, the camera setup is done using a tripod with camera attachment mechanism and the checkerboard is made to display on laptop. Figure 56,58, 57 shows the setup done for the experiment.

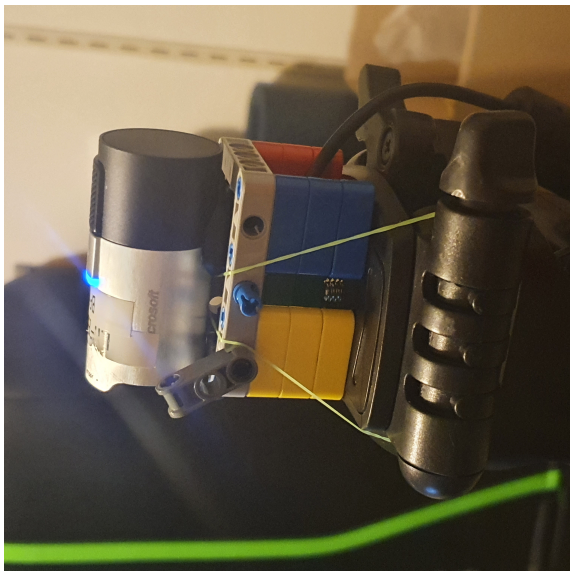

Figure 56: *Camera setup - side view*

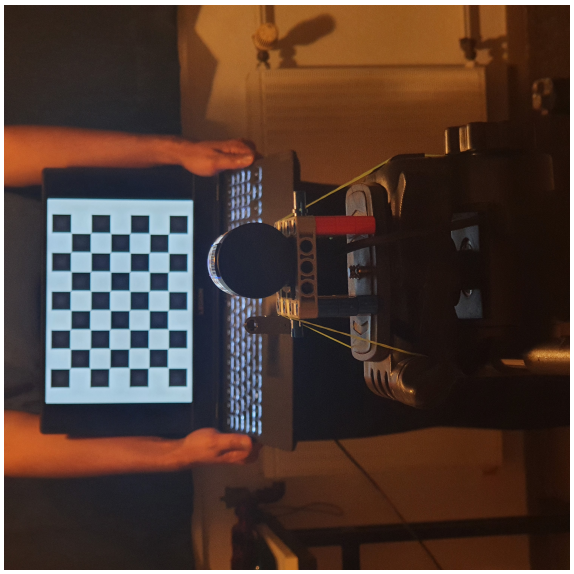

Figure 57: *Checkerboard and camera setup*

Figure 58: *Tripod with attached camera*

- Given Microsoft lifecam studio webcam is connected to the system to take the images. Autofocus is turned off during the experiment.
- GTK UVC software is installed in the system in which the camera is connected and this software is used to take the images.
- Checkerboard is displayed 15.6inch lenovo laptop of shape 7*5. The side of the each square is 2.5cm which is measured using scale.

- The camera is placed stationary using the tripod and then image is clicked from the software trigger. The monitor is moved in various orientations to take images. The checkerboard is kept constant.
- Around 30 to 40 raw images of the checkerboard in different orientations using GTK UVC software is taken. In our experiment we have taken 34 images in total.
- The position of the laptop is moved in order to cover various angle such as the top view, side view, oriented towards left,right, up, down.
- The camera calibration is done using MATLAB toolbox shown in figure 59, from which the camera parameters are derived.

Figure 59: *MATLAB toolbox used for camera calibration*

3.4.1 Calibration Images

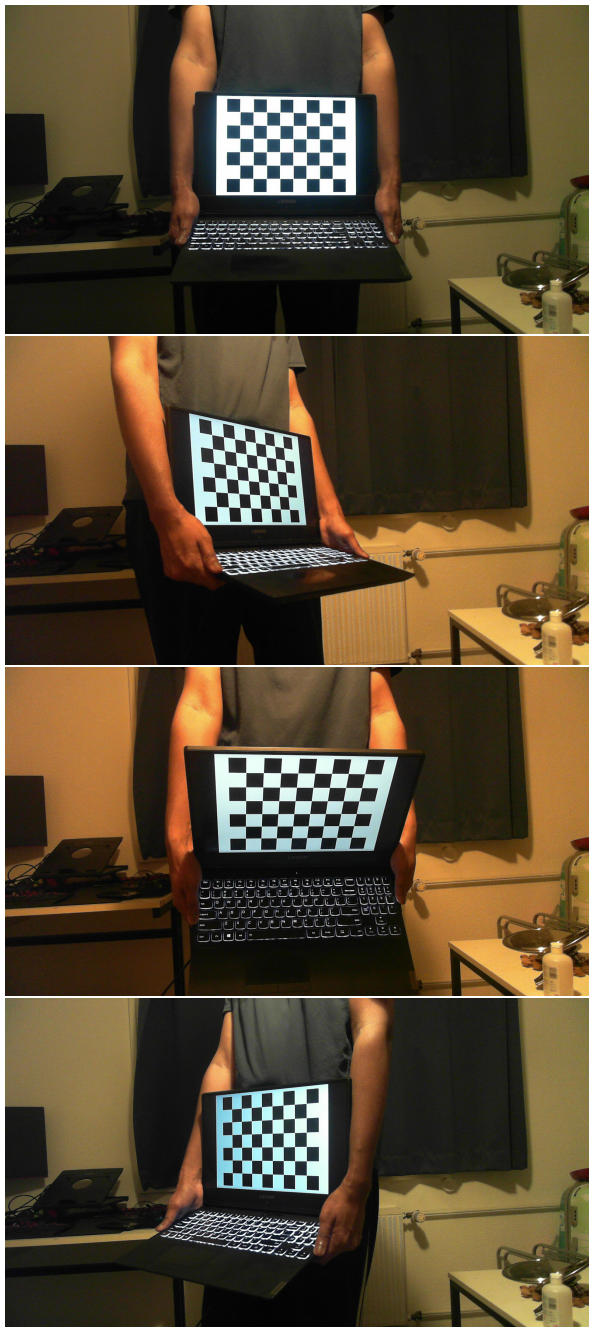

Figure 60: *Images taken from different orientations for calibration*

3.4.2 Results

- The size of input image is 1920 × 1080 px
- Intrinsic parameters: All units are in pixels

$$K = \begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} = \begin{bmatrix} 1456.0 & 0.0 & 0.0 \\ 0.0 & 1456.0 & 0.0 \\ 959.5 & 539.5 & 1.0 \end{bmatrix} \pm \begin{bmatrix} 5.7 & 0.0 & 0.0 \\ 0.0 & 5.5 & 0.0 \\ 10.2 & 8.7 & 0.0 \end{bmatrix}$$

- Distortion coefficients:

$$\begin{bmatrix} -0.03041 & 0.38110 & 0.00041 & 0.00144 & 0.00000 \end{bmatrix} \pm \begin{bmatrix} 0.02215 & 0.17283 & 0.00235 & 0.00257 & 0.00000 \end{bmatrix}$$

- From the calculated intrinsic parameters, Image 61 is undistorted and the extrinsic parameters of that image are given below.
- The figure 62 shows the image after corner extraction of the input image.
- The figure 63 shows the re-projection of points(red) using computed intrinsic parameters.
- Computed extrinsic parameters: All the units are in mm
  - Translation vector $P_0 = \begin{bmatrix} -19.614408 & -128.840927 & 706.23532 \end{bmatrix}$
  - Rotation matrix $R = \begin{bmatrix} -0.358341 & 0.723229 & 0.590366 \\ 0.924558 & 0.187161 & 0.331908 \\ 0.129552 & 0.664764 & -0.735735 \end{bmatrix}$
  - P_0 and R are of parameters rigid motion equation 17,

$$P_2 = P_1 \times R + P_0 \tag{17}$$

Where P_1 is coordinate vector of any point **P** in grid(world) reference frame (can be seen in figure 63) and P_2 is camera frame coordinate vector of point **P**.

3.4.3 Output images

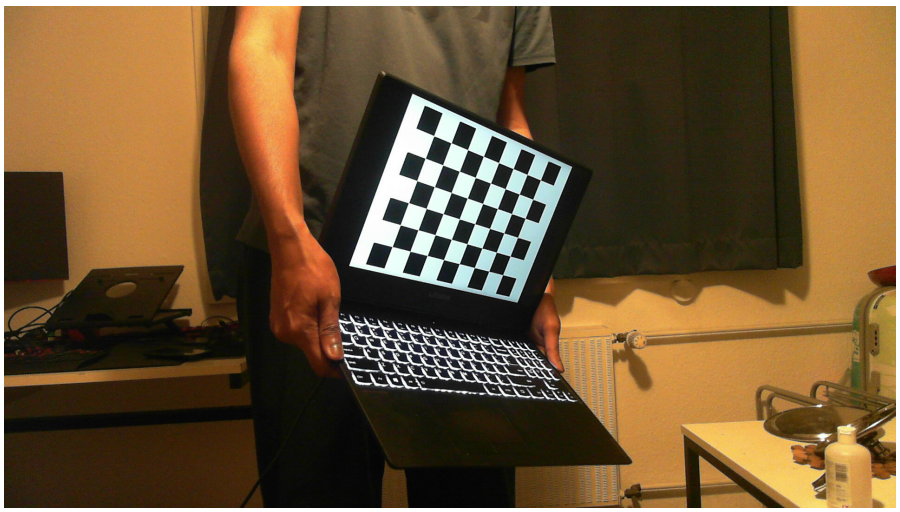

Figure 61: *Input image for undistortion*

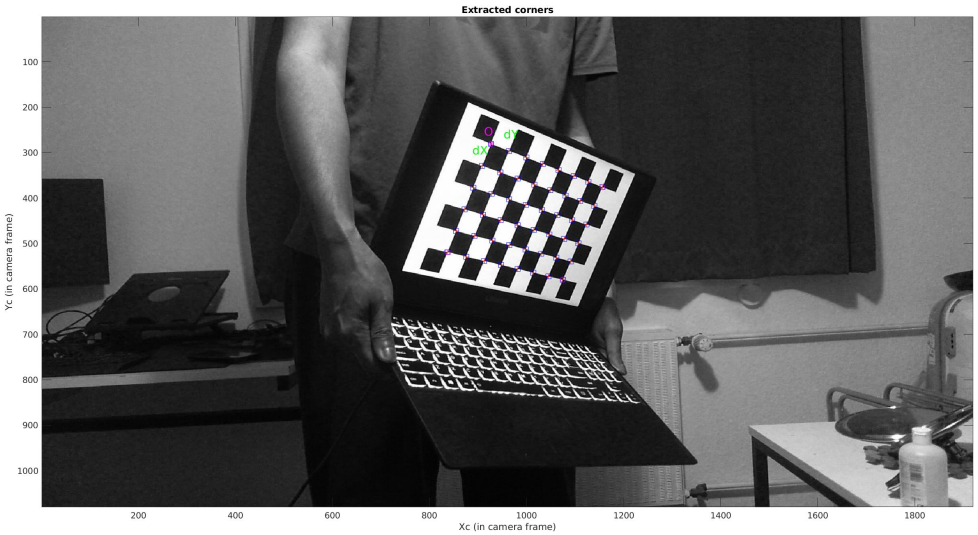

Figure 62: *Image after corner extraction*

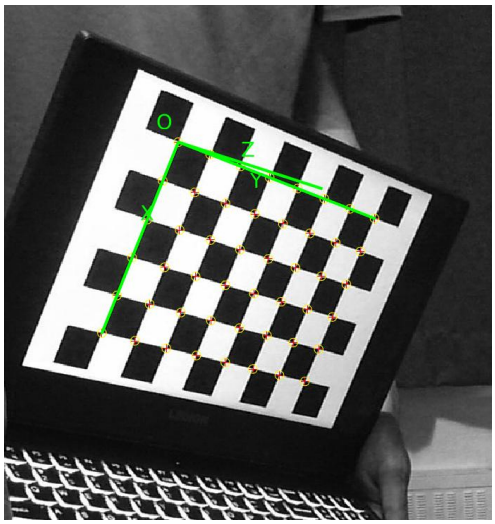

Figure 63: *Re-projection using computed intrinsic parameters*

- Figure 64 represents the extrinsic parameter of the camera model of all input images in camera-centred view.

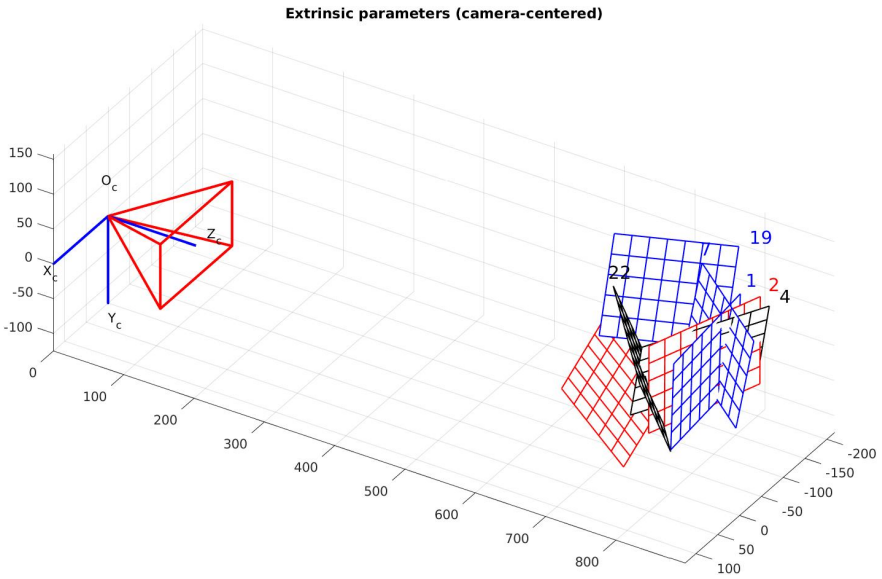

Figure 64: *Extrinsic parameter of calibration images in camera-centred view (All units are in mm)*

3.4.4 Possible error sources

- Daylight and Nightlight setup may affect the experiment.
- The camera can go out of focus.
- Since laptop is used to display the checkerboard, the refresh rate might of display may affect the image quality.
- Brightness of the laptop screen and keyboard backlight may affect the experiment.
- Shadow of the tripod or human can occur which may cause difficult in corner detection.

3.4.5 Best fit camera model

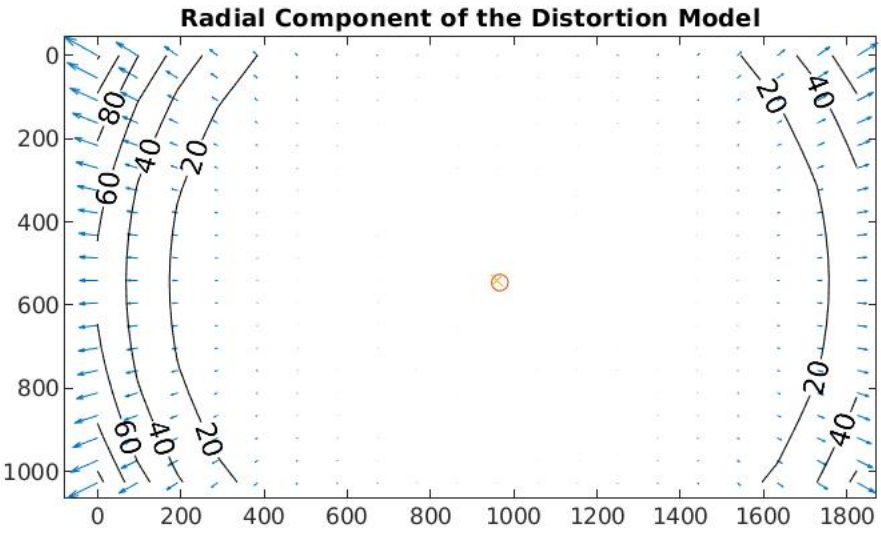

Figure 65: *Radial component of the distortion model*

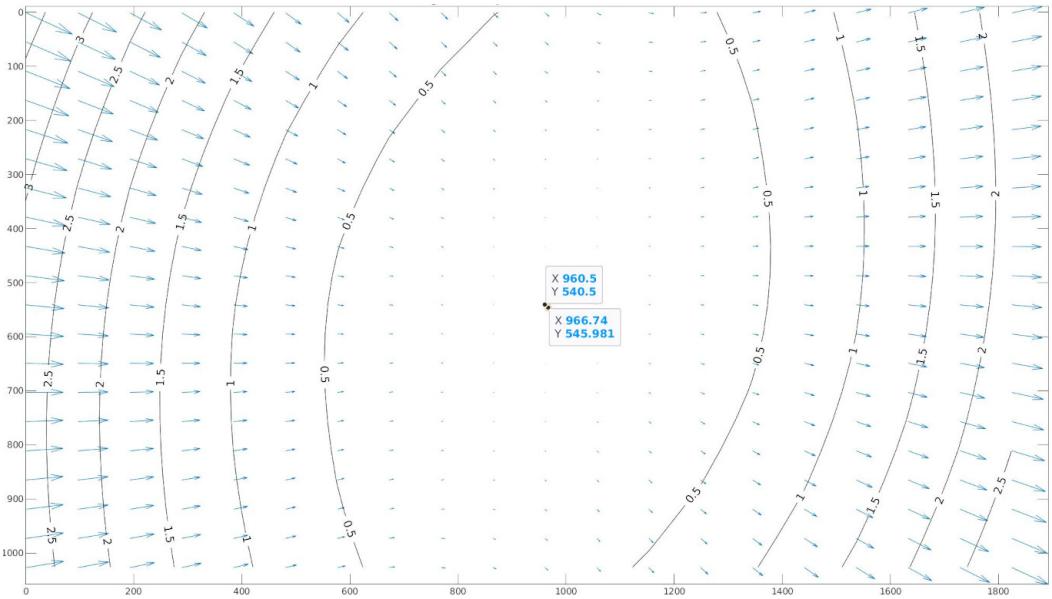

Figure 66: *Tangential component of the distortion model*

4 Measuring the Accuracy and Precision of a KUKA youBot Arm

4.1 Aim

The aim of this experiment is to estimate the accuracy and precision of the KUKA youBot robotic arm by performing the pick and place task with different mass of the objects and with different poses.

4.2 Experimental Setup

- KUKA youBot robot arm.
- Three aluminum hex channel of different weights with ArUco markers attached on top of it.
- A camera which is placed above the robot's workspace.
- Fixed holder on the table in order to ensure the placement position is the same for each trials.
- Workstation with necessary interface and packages to control the robot.
- Additional laptop/computer to subscribe, to the values from the camera to get the end pose of the object. The subscriber script is available in LEA.

4.3 Experiment Procedure

- Necessary ROS launch files should be opened in the terminal of the computer in workstation as described in the SEE manual.[Figure 67]
- The value subscription can be checked from the other computer. [Figure 68]

Figure 67: *Laptop used to run ROS launch files*

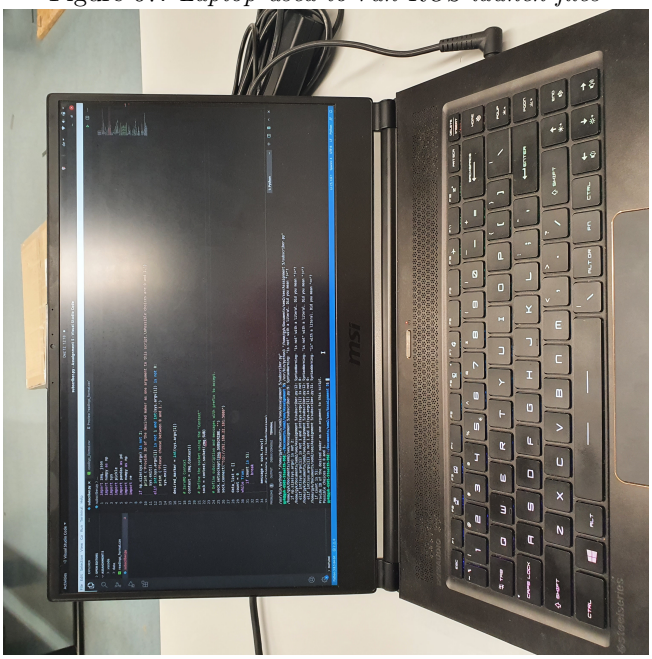

Figure 68: *Laptop used to run Subscriber.py file*

- Before working with the youBot, the home position and the stability of the robot should be ensured. Then the robot can be operated from the workstation. [Figure 69]

Figure 69: *YouBot used in this experiment*

- Once the youBot comes to the initial position, the object should be placed in the fixed holder of the table and ArUco markers is used in order to ensure the initial position for each trials. [Figure 70]

Figure 70: *Experiment Setup*

- Initially the object with less weight is placed. After the specific number of runs, the weight of the object can be selected in the increasing order (small, medium and large). [Figure 71]

Figure 71: *Object used for pick and place*

- Once the object is placed, the command to grasp the object can be given from the workstation and then the arm picks the object. [Figure 72]

Figure 72: *YouBot picking the object*

- In order to place the object, the position can be chosen from the workstation. The position can either be left, right or straight.
- After choosing the position, the robot places the object in that position and comes back to its position.
- The initial position of the object should be maintained same for each run and also there should not be any vibrations.
- Once the robot places the object in the desired position, the subscriber script can be started on the computer in order to obtain the end pose of the object from the camera.
- This steps should be performed for minimum 20 times each in all three positions(left, right and straight) and also with different object mass (small, medium and large).
- Finally, 180 observations should be made in the end of this experiment.

4.4 Estimated Error Sources

- When center of the gripper is not matched with the center of the object, then the object experiences some movement.
- When the object is placed on the specific position, it is not smoothly landed. Because of the vibration, there might be a small shift in the position.
- Vibration of the object increases with the increase in the height of the object. This might also cause the drift in the placing position.
- Conditions of the lighting might affect the experiment. Hence same lighting conditions should be maintained throughout the experiment.
- While subscribing the values from the camera, robot or object should not be moved. Even small deviations can affect the values of the experiment.
- Few times the gripper rubs the object as it retracted back.

4.5 Pose Filtering

- We have combined the measurements datasets provided by all the groups and hence we had a total of 150 samples (20 from each group, but some group took more samples apparently) for each run for each configurations.
- Bounds of Chebyshev theorem is used to remove the outliers.
- Chebyshev Theorem states that only a certain amount of data points in a probability distribution can be present from a particular distance from the mean of the distribution. and is given by:

$$P(|X - \mu| \leq K\sigma) \geq (1 - \frac{1}{K^2}) \tag{18}$$

$$P(\mu - 2\sigma < X < \mu + 2\sigma) \geq \frac{3}{4} = 0.75 \tag{19}$$

- We have taken the threshold as 2 standard deviations from the mean. Points outside the threshold are taken as outliers.
- We estimate the random variable X between upper and lower bounds having mean μ and standard deviation σ. We choose the random variable to be end-pose co-ordinates and orientation for each motion.
- Tables below represent the outlier values for our measurement data and also combined data.

S.No	Object Size	Direction	Random Variable	Total no.of data	No.of outliers
1	Small	Left	X	20	2
2	Small	Left	Y	20	1
3	Small	Left	Theta	20	2
4	Small	Right	X	20	1
5	Small	Right	Y	20	1
6	Small	Right	Theta	20	2
7	Small	Straight	X	20	1
8	Small	Straight	Y	20	1
9	Small	Straight	Theta	20	0
10	Medium	Left	X	20	1
11	Medium	Left	Y	20	1
12	Medium	Left	Theta	20	1
13	Medium	Right	X	20	0
14	Medium	Right	Y	20	0
15	Medium	Right	Theta	20	2
16	Medium	Straight	X	20	1
17	Medium	Straight	Y	20	1
18	Medium	Straight	Theta	20	0
19	Large	Left	X	20	0
20	Large	Left	Y	20	0
21	Large	Left	Theta	20	0
22	Large	Right	X	20	1
23	Large	Right	Y	20	1
24	Large	Right	Theta	20	0
25	Large	Straight	X	20	1
26	Large	Straight	Y	20	1
27	Large	Straight	Theta	20	2

Table 15: *Pose filtering for our individual measurement data*

S.No	Object Size	Direction	Random Variable	Total no.of data	No.of outliers
1	Small	Left	X	150	3
2	Small	Left	Y	150	1
3	Small	Left	Theta	150	3
4	Small	Right	X	150	9
5	Small	Right	Y	150	7
6	Small	Right	Theta	150	6
7	Small	Straight	X	150	11
8	Small	Straight	Y	150	9
9	Small	Straight	Theta	150	13
10	Medium	Left	X	150	9
11	Medium	Left	Y	150	8
12	Medium	Left	Theta	150	5
13	Medium	Right	X	150	3
14	Medium	Right	Y	150	8
15	Medium	Right	Theta	150	7
16	Medium	Straight	X	150	8
17	Medium	Straight	Y	150	7
18	Medium	Straight	Theta	150	2
19	Large	Left	X	150	1
20	Large	Left	Y	150	14
21	Large	Left	Theta	150	6
22	Large	Right	X	150	6
23	Large	Right	Y	150	5
24	Large	Right	Theta	150	4
25	Large	Straight	X	150	15
26	Large	Straight	Y	150	12
27	Large	Straight	Theta	150	10

Table 16: *Pose filtering for the combined measurement data*

4.6 Pre-processed Data

Small Object - Left motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	54.83	-42.33	-116.53
2	53.75	-41.48	-120.85
3	54.12	-41.66	-117.35
4	53.69	-40.81	-119.98
5	54.33	-42.03	-117.24
6	54.33	-42.03	-117.27
7	63.17	-56.51	-93.78
8	24.52	-40.46	-96.69
9	54.33	-42.03	-117.21
10	53.88	-41.37	-117.15
11	25.06	-41.13	-89.89
12	53.45	-40.99	-118.75
13	53.55	-41.08	-118.07
14	53.63	-41.16	-118.03
15	53.68	-41.27	-117.98
16	53.84	-41.34	-116.95
17	54.33	-42.03	-117.23
18	54.77	-42.37	-121.35
19	54.33	-42.03	-117.23
20	51.99	-39.64	-120.46
21	52.66	-40.29	-121.09

Table 17: *Measurement reading of End poses for left motion for small object configuration*

Small Object - Straight motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	27.34	-28.11	-89.01
2	27.19	-28.23	-88.43
3	27.23	-28.27	-93.86
4	27.27	-28.34	-88.75
5	27.34	-28.49	-88.04
6	27.07	-28.09	-104.19
7	25.91	-26.72	-105.86
8	27.26	-28.34	-89.18
9	27.17	-28.24	-98.06
10	26.06	-26.91	-106.19
11	26.69	-27.66	-106.66
12	26.69	-27.65	-107.04
13	27.34	-28.38	-89.13
14	27.32	-28.44	-89.92
15	26.17	-27.23	-91.55
16	26.19	-27.22	-90.76
17	26.98	-27.98	-102.05
18	27.16	-28.19	-98.38
19	27.29	-28.22	-88.95
20	26.97	-27.76	-101.11
21	27.33	-28.31	-90.07

Table 18: *Measurement reading of End poses for straight motion for small object configuration*

Small Object - Right motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	-4.14	-28.62	-53.75
2	-3.66	-29.72	-57.55
3	-3.84	-29.24	-60.14
4	-3.84	-29.25	-60.15
5	-3.81	-29.35	-60.07
6	-4.58	-27.73	-59.58
7	-3.93	-29.04	-59.36
8	-4.08	-28.68	-58.85
9	-4.14	-28.61	-58.87
10	-4.25	-28.33	-58.67
11	-3.61	-29.75	-56.81
12	-4.45	-28.89	-59.13
13	-4.08	-28.75	-58.26
14	-3.64	-29.74	-55.17
15	-3.94	-29.03	-58.53
16	-3.75	-29.52	-57.88
17	-3.95	-29.02	-60.36
18	-4.21	-28.42	-58.69
19	-4.18	-28.52	-58.34
20	-3.89	-29.13	-58.52
21	-4.17	-28.56	-58.97

Table 19: *Measurement reading of End poses for right motion for small object configuration*

Medium Object - Left motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	51.17	-36.93	-118.87
2	54.86	-40.17	-118.57
3	51.78	-37.61	-116.67
4	51.16	-36.78	-117.35
5	52.67	-38.18	-120.86
6	52.74	-38.43	-118.19
7	51.82	-37.64	-116.71
8	53.18	-38.72	-118.25
9	52.13	-37.77	-116.85
10	52.87	-38.41	-117.97
11	50.58	-36.27	-119.52
12	55.34	-40.67	-116.25
13	50.64	-36.33	-119.38
14	56.74	-42.17	-118.17
15	50.56	-36.25	-119.43
16	51.82	-37.64	-116.71
17	51.44	-37.05	-119.33
18	51.23	-36.91	-118.88
19	51.82	-37.64	-116.71
20	51.26	-36.94	-118.86
21	51.12	-36.93	-115.36

Table 20: *Measurement reading of End poses for left motion for medium object configuration*

Medium Object - Straight motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	25.27	-24.56	-88.66
2	25.19	-24.37	-97.34
3	26.16	-25.51	-95.99
4	25.13	-24.44	-102.51
5	24.77	-23.91	-97.32
6	25.19	-24.36	-93.92
7	25.24	-24.36	-99.74
8	25.25	-24.63	-104.16
9	25.23	-24.39	-99.19
10	25.16	-24.29	-90.32
11	26.56	-25.99	-92.15
12	25.23	-24.41	-91.98
13	25.59	-24.83	-90.46
14	26.18	-25.56	-92.07
15	25.74	-25.15	-91.96
16	25.23	-24.66	-91.05
17	25.45	-24.66	-99.64
18	25.27	-24.56	-102.85
19	26.39	-25.81	-92.02
20	25.29	-24.51	-96.39
21	26.48	-25.94	-92.05

Table 21: *Measurement reading of End poses for straight motion for medium object configuration*

Medium Object - Right motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	-5.25	-26.49	-62.66
2	-5.21	-26.58	-62.72
3	-5.66	-25.74	-62.81
4	-5.94	-25.18	-61.09
5	-5.66	-25.56	-64.21
6	-5.35	-26.45	-62.86
7	-5.73	-25.68	-60.64
8	-5.05	-26.69	-65.64
9	-5.38	-26.22	-63.18
10	-5.39	-26.16	-63.28
11	-5.05	-26.98	-59.23
12	-5.73	-25.69	-60.45
13	-4.94	-27.02	-65.29
14	-4.94	-26.91	-65.29
15	-5.45	-26.18	-62.99
16	-5.29	-25.97	-66.85
17	-5.66	-25.41	-62.95
18	-5.46	-26.08	-63.82
19	-5.24	-26.42	-63.47
20	-5.69	-25.62	-63.44
21	25.02	-23.28	-96.01

Table 22: *Measurement reading of End poses for right motion for medium object configuration*

Large Object - Left motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	49.46	-34.64	-118.15
2	50.09	-35.23	-116.42
3	50.22	-35.32	-117.59
4	50.72	-35.83	-116.35
5	50.26	-35.32	-118.96
6	49.64	-34.57	-118.08
7	49.64	-34.79	-118.62
8	49.89	-35.04	-117.94
9	51.16	-36.18	-116.49
10	49.43	-34.54	-116.55
11	49.66	-34.82	-118.06
12	51.28	-36.25	-116.46
13	50.63	-35.69	-117.31
14	51.38	-36.42	-116.57
15	49.35	-34.37	-118.85
16	48.93	-33.98	-118.59
17	49.13	-34.21	-118.26
18	48.91	-34.59	-118.65
19	50.92	-35.96	-117.34
20	49.34	-34.37	-118.65
21	51.64	-36.59	-116.67

Table 23: *Measurement reading of End poses for left motion for large object configuration*

Large Object - Straight motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	24.46	-22.86	-97.06
2	24.53	-22.64	-98.76
3	24.38	-22.88	-95.86
4	23.76	-22.14	-94.01
5	24.97	-23.36	-97.52
6	23.61	-21.99	-93.87
7	24.63	-23.37	-89.53
8	24.38	-22.4	-98.65
9	23.26	-21.65	-105.12
10	23.29	-21.69	-89.91
11	24.12	-22.51	-91.94
12	23.99	-22.35	-96.81
13	23.81	-21.64	-106.56
14	24.17	-22.44	-93.96
15	24.34	-22.63	-92.26
16	24.68	-22.87	-98.02
17	26.11	-24.38	-98.06
18	23.18	-21.46	-91.99
19	24.83	-23.37	-94.18
20	24.45	-22.38	-96.76
21	25.02	-23.28	-96.01

Table 24: *Measurement reading of End poses for straight motion for large object configuration*

Large Object - Left motion			
S.no	X axis (cm)	Y axis (cm)	Orientation (deg)
1	-7.26	-21.89	-63.14
2	-6.07	-24.97	-58.51
3	-6.85	-22.29	-59.32
4	-6.29	-23.94	-62.73
5	-6.77	-23.24	-58.97
6	-7.04	-22.35	-61.44
7	-6.51	-23.99	-59.23
8	-7.17	-22.45	-58.72
9	-6.54	-23.68	-59.75
10	-6.79	-23.54	-57.02
11	-6.41	-24.16	-58.12
12	-7.08	-22.23	-62.72
13	-6.86	-22.61	-61.74
14	-6.62	-23.4	-59.36
15	-6.71	-23.13	-60.48
16	-6.34	-23.82	-62.63
17	-6.72	-23.26	-59.66
18	-6.68	-23.54	-57.98
19	-6.29	-23.94	-62.73
20	-6.69	-23.02	-62.04
21	-6.11	-24.41	-62.11

Table 25: *Measurement reading of End poses for right motion for large object configuration*

4.7 Data Visualization

We have kept the camera frame as the reference frame while taking the measurement readings of the KUKA youbot positions as shown in the figure.

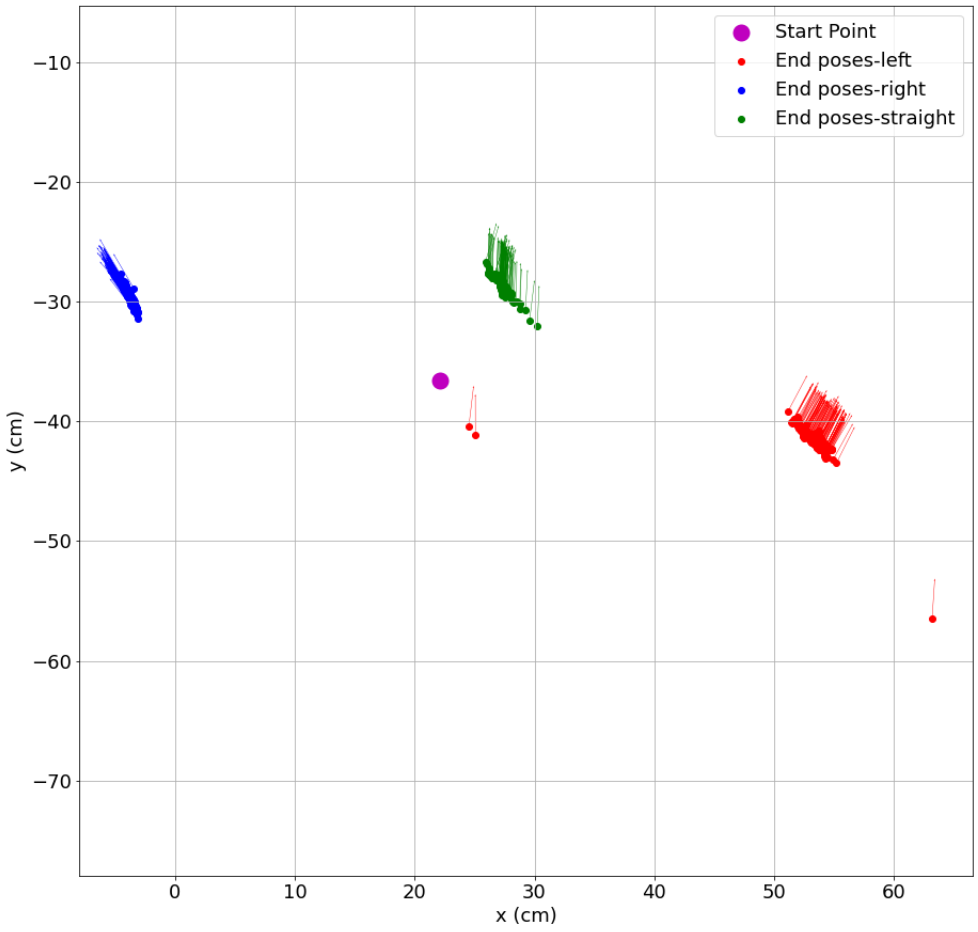

Figure 73: *End poses with respect to the camera frame for small object configuration with outliers*

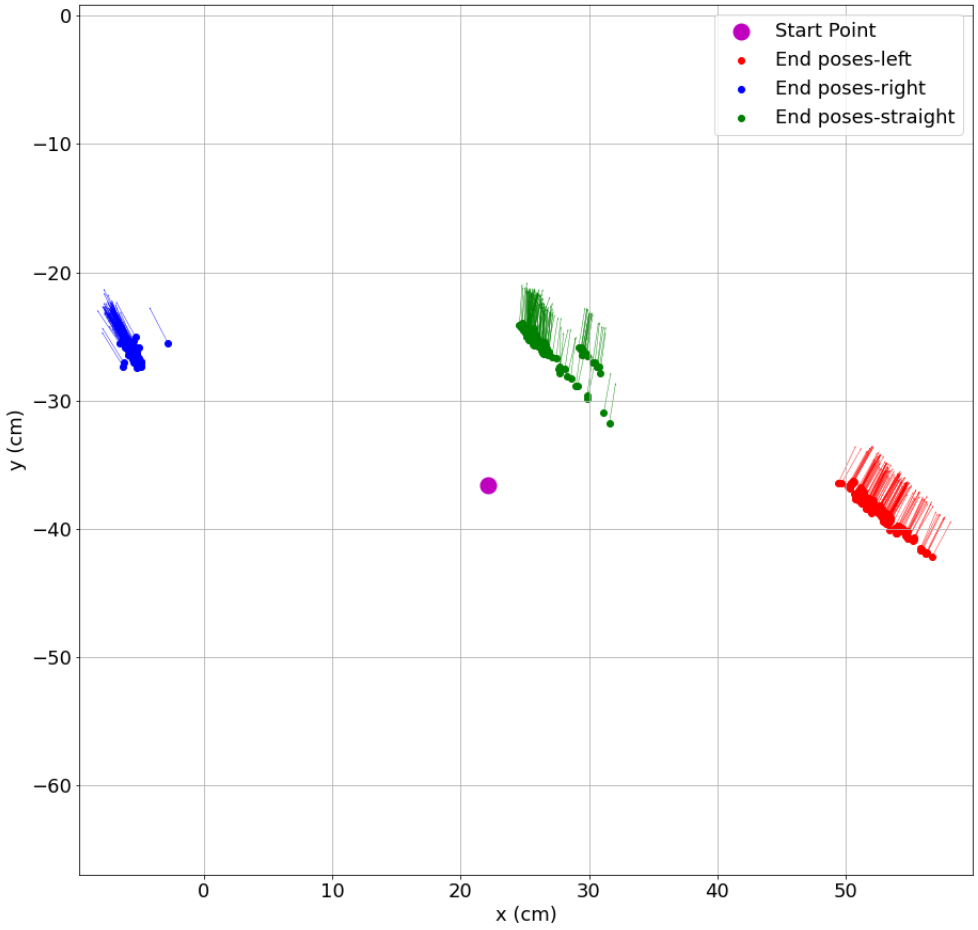

Figure 74: *End poses with respect to the camera frame for medium object configuration*

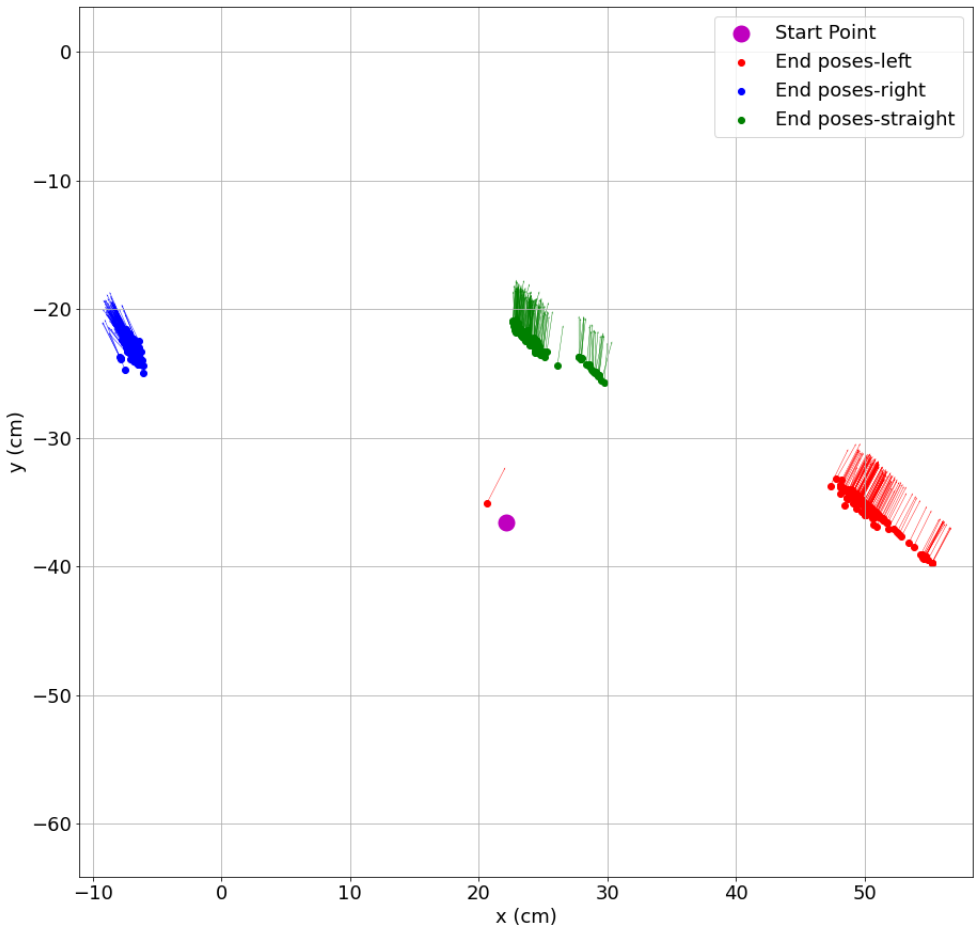

Figure 75: *End poses with respect to the camera frame for large object configuration*

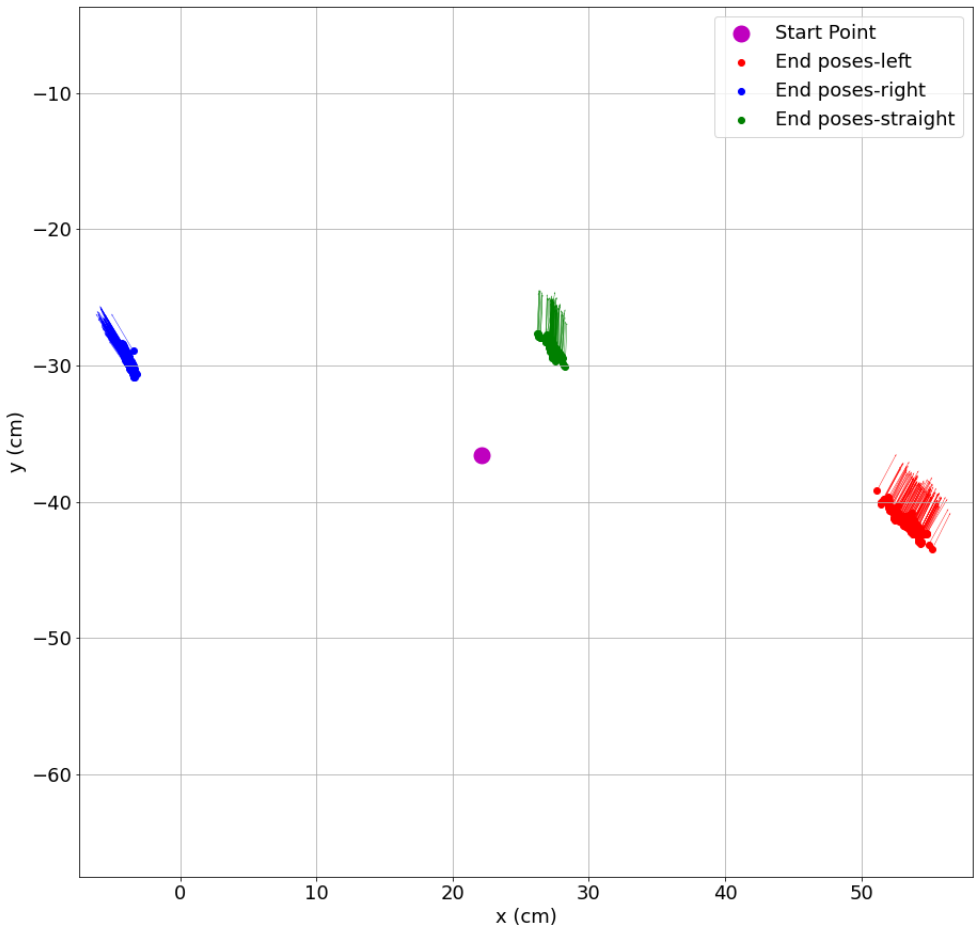

Figure 76: *End poses with respect to the camera frame for small object configuration without outliers*

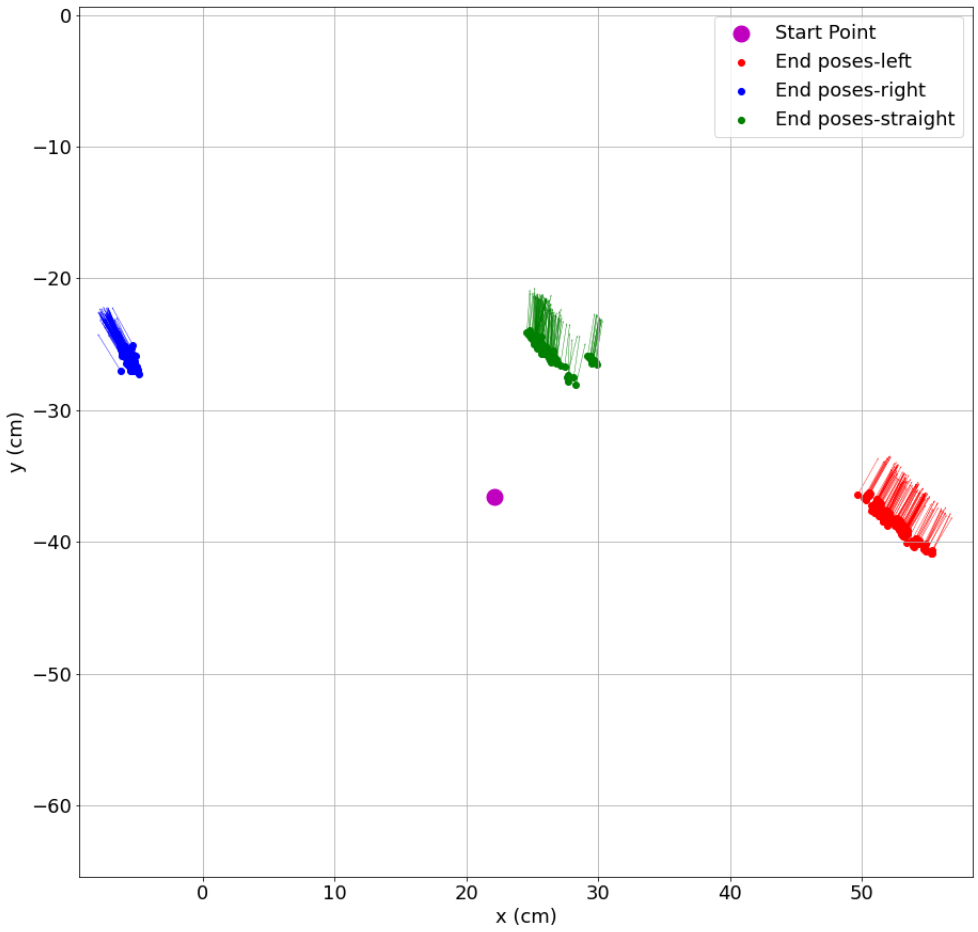

Figure 77: *End poses with respect to the camera frame for medium object configuration without outliers*

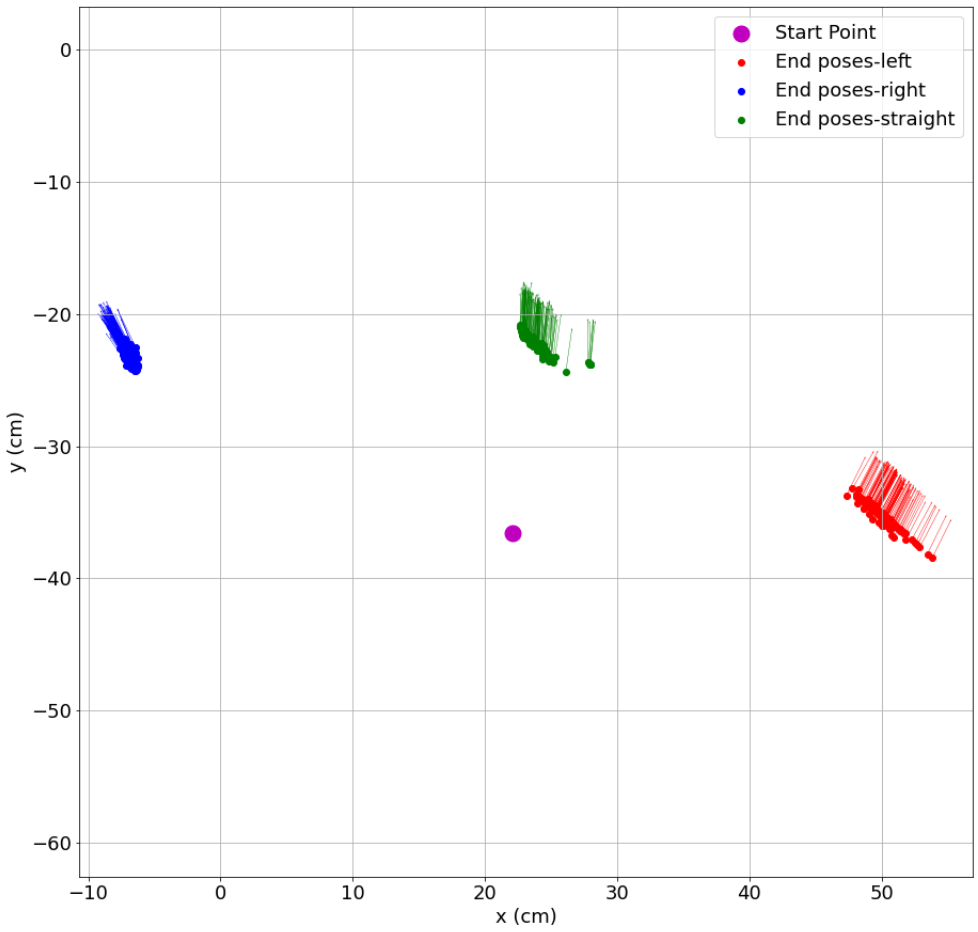

Figure 78: *End poses with respect to the camera frame for large object configuration without outliers*

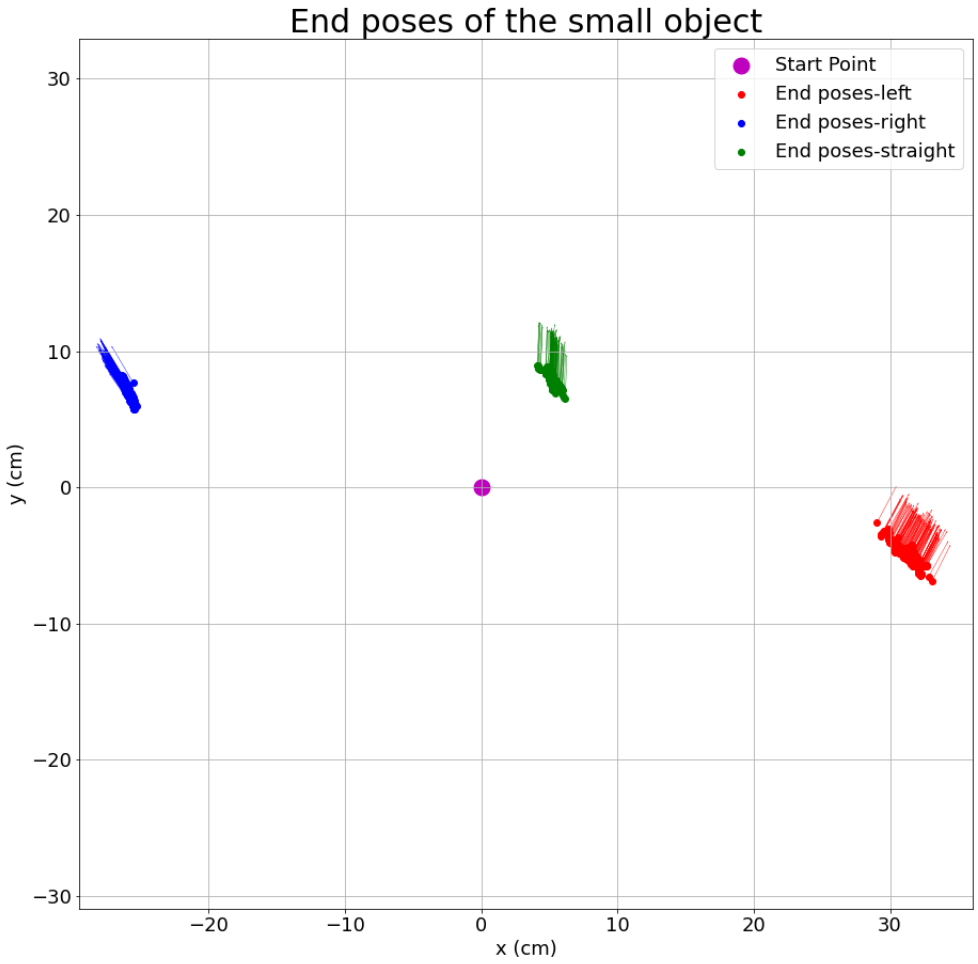

Figure 79: *End poses for small object configuration without outliers*

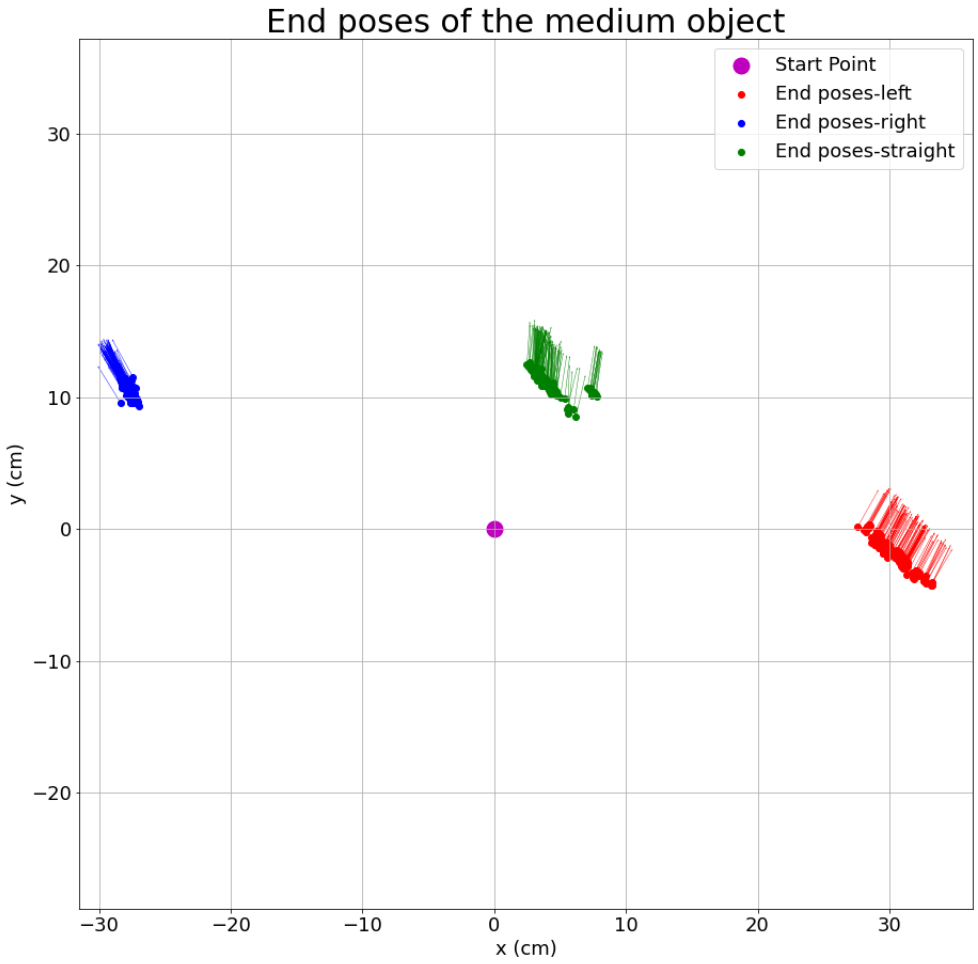

Figure 80: *End poses for medium object configuration without outliers*

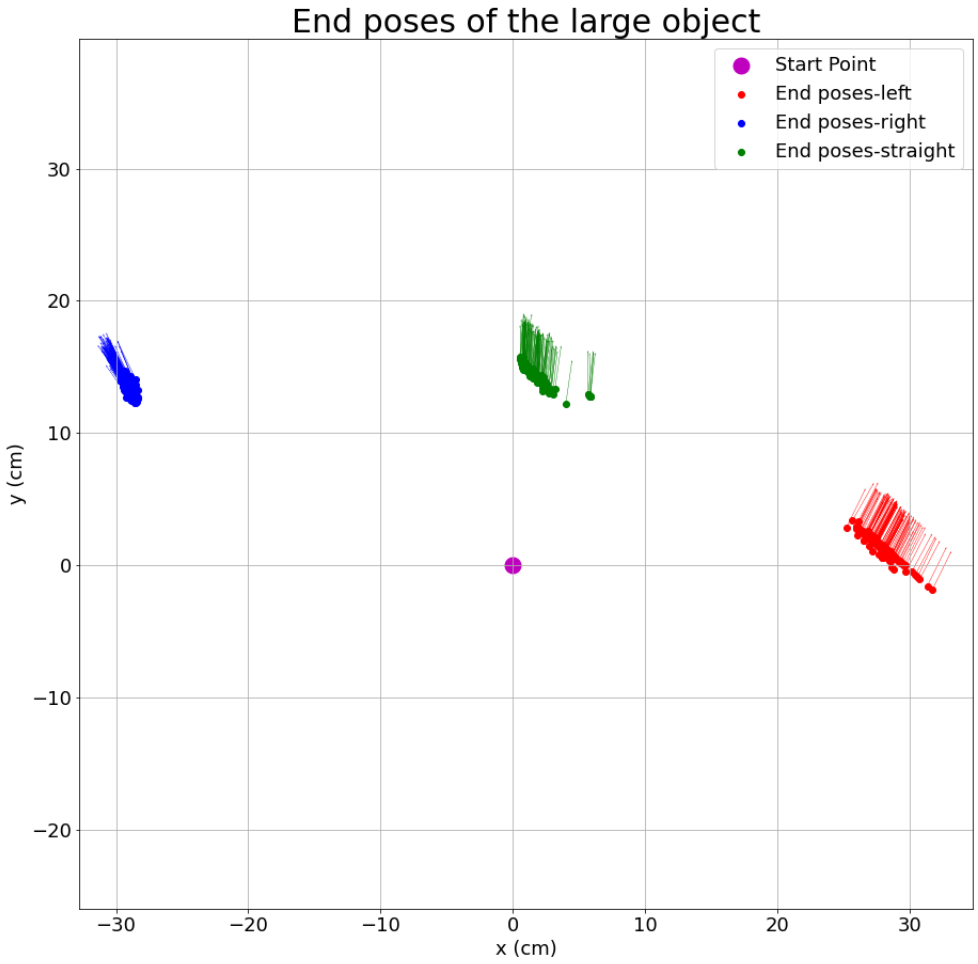

Figure 81: *End poses for large object configuration without outliers*

5 Statistical Evaluation of a KUKA youBot Arm measurement data

5.1 Fitting Gaussian

- The below figures represent the Gaussian fitting plot for left, right and straight motion for each weight of size small, medium and large.

Weight: Small
Gaussian fit for the left motion

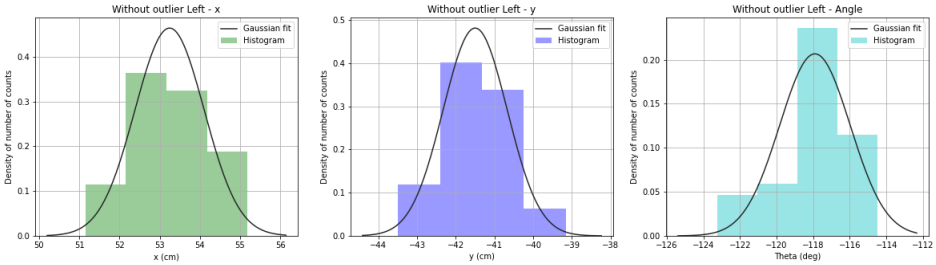

Figure 82: *Gaussian fit for left position for small object*

Gaussian fit for the right motion

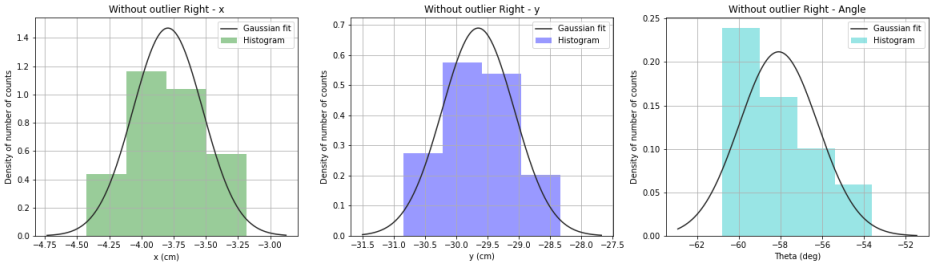

Figure 83: *Gaussian fit for right position for small object*

Gaussian fit for the straight motion

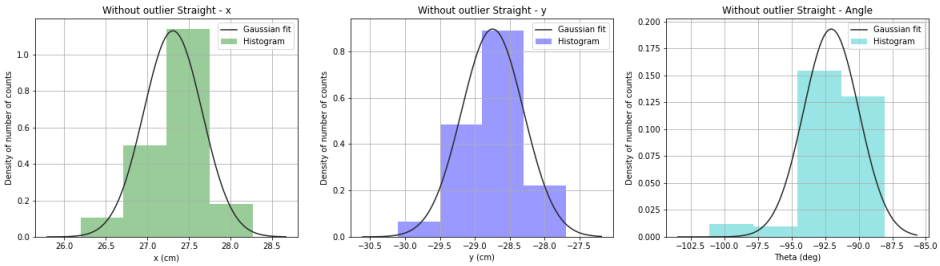

Figure 84: *Gaussian fit for straight position for small object*

Weight: Medium

Gaussian fit for the left motion

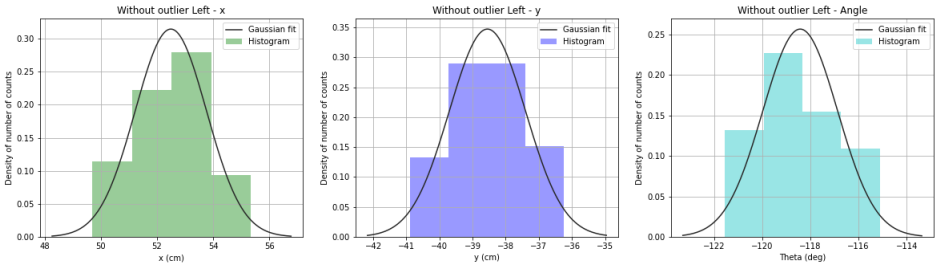

Figure 85: *Gaussian fit for left position for medium object*

Gaussian fit for the right motion

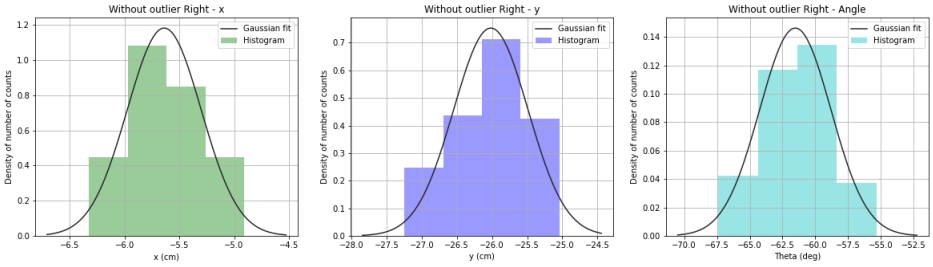

Figure 86: *Gaussian fit for right position for medium object*

Gaussian fit for the straight motion

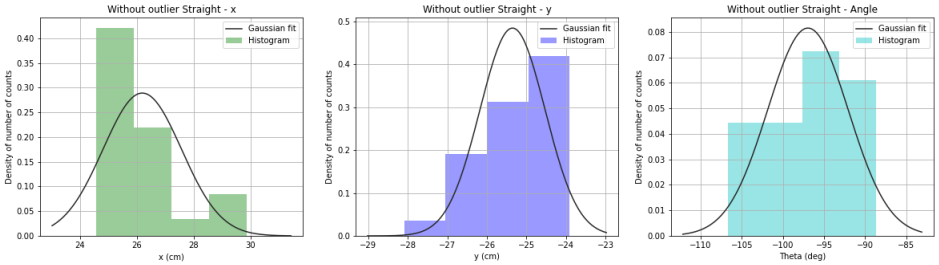

Figure 87: *Gaussian fit for straight position for medium object*

Weight: Large

Gaussian fit for the left motion

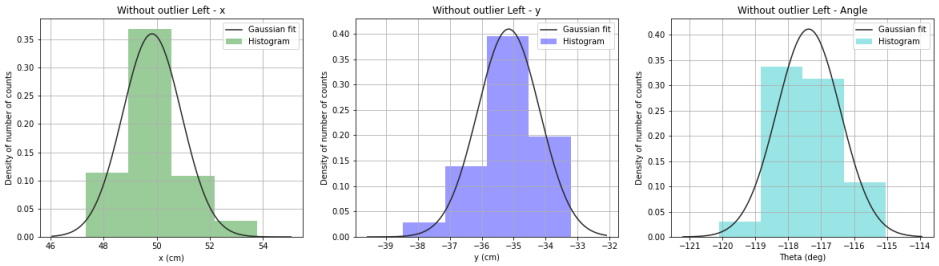

Figure 88: *Gaussian fit for left position for large object*

Gaussian fit for the right motion

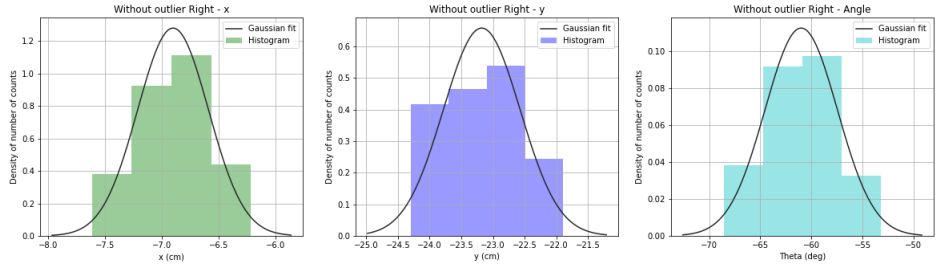

Figure 89: *Gaussian fit for right position for large object*

Gaussian fit for the straight motion

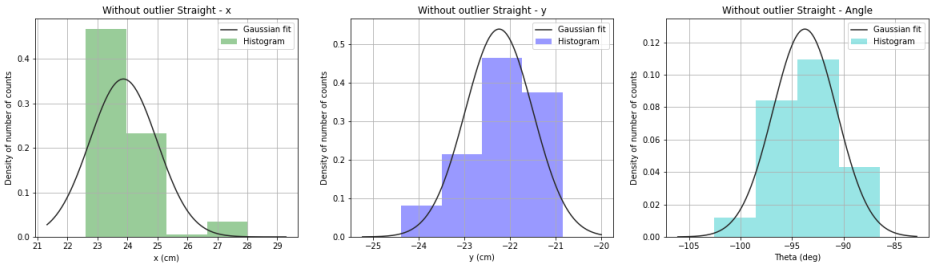

Figure 90: *Gaussian fit for straight position for large object*

5.2 Fitting Gaussian after PCA

- PCA is performed for dimensionality reduction.
- The below figures represent the Gaussian fitting plot for left, right and straight motion for each weight of size small, medium and large.

Weight: Small
Gaussian fit for the left motion

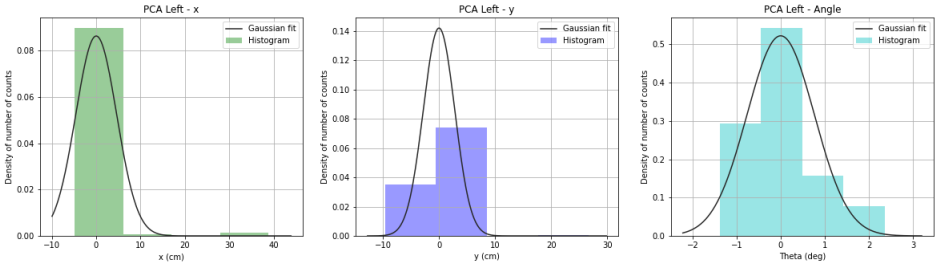

Figure 91: *Gaussian fit after PCA for left position for small object*

Gaussian fit for the right motion

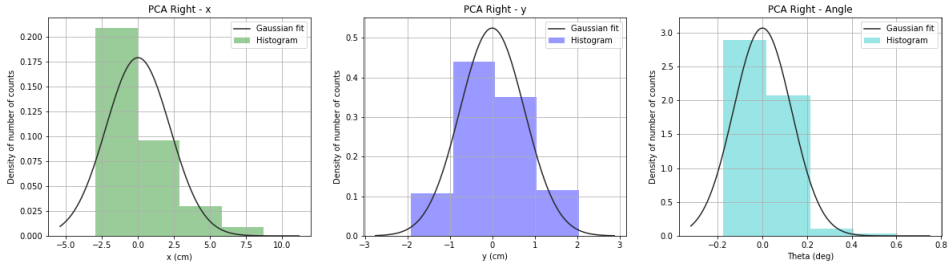

Figure 92: *Gaussian fit after PCA for right position for small object*

Gaussian fit for the straight motion

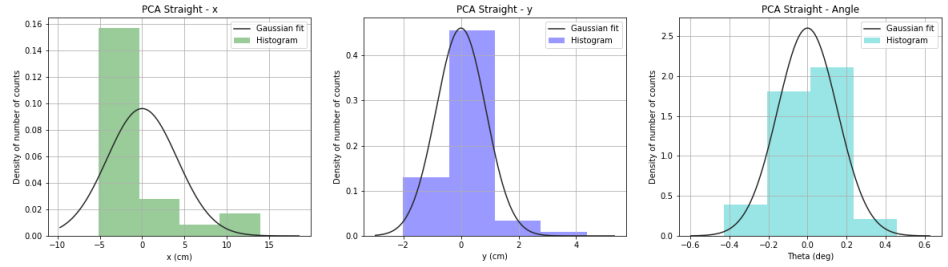

Figure 93: *Gaussian fit after PCA for straight position for small object*

Weight: Medium

Gaussian fit for the left motion

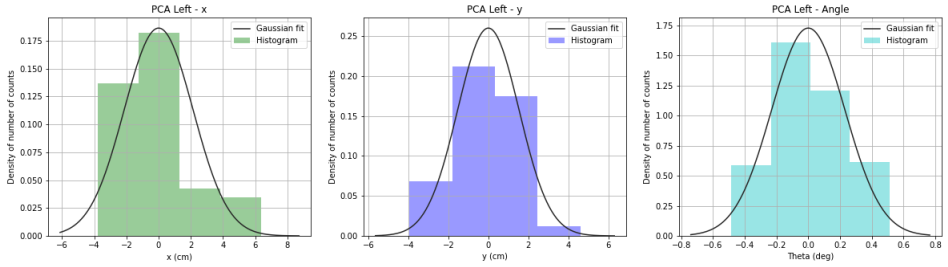

Figure 94: *Gaussian fit after PCA for left position for medium object*

Gaussian fit for the right motion

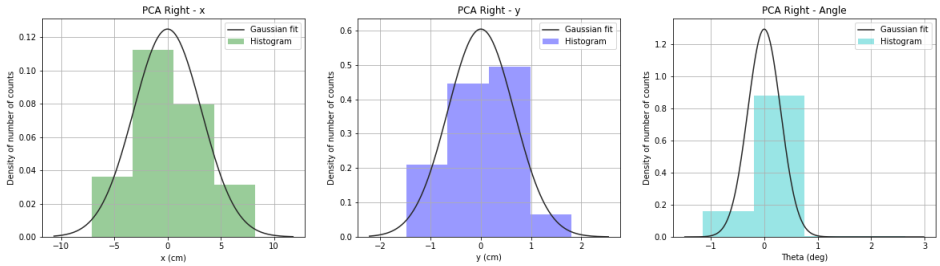

Figure 95: *Gaussian fit after PCA for right position for medium object*

Gaussian fit for the straight motion

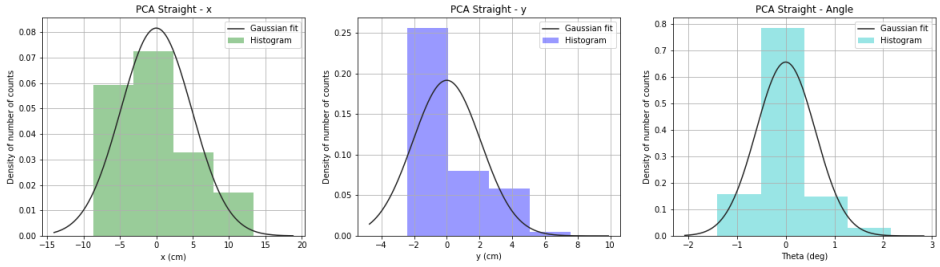

Figure 96: *Gaussian fit after PCA for straight position for medium object*

Weight: Large

Gaussian fit for the left motion

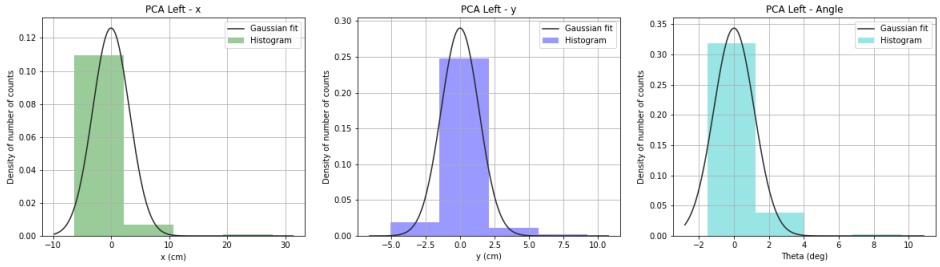

Figure 97: *Gaussian fit after PCA for left position for large object*

Gaussian fit for the right motion

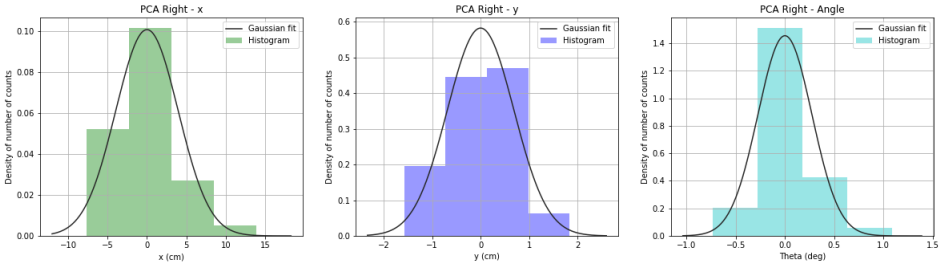

Figure 98: *Gaussian fit after PCA for right position for large object*

Gaussian fit for the straight motion

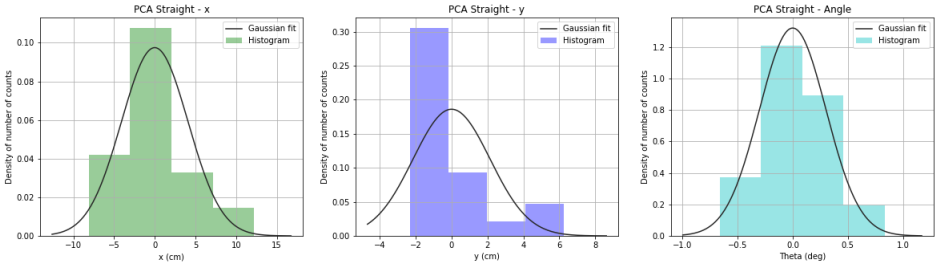

Figure 99: *Gaussian fit after PCA for straight position for large object*

5.3 Uncertainty ellipse

- The below figures represent the uncertainty ellipse of the robots left, right and straight motion after PCA respectively for each weight of size small, medium and large.

Weight: Small

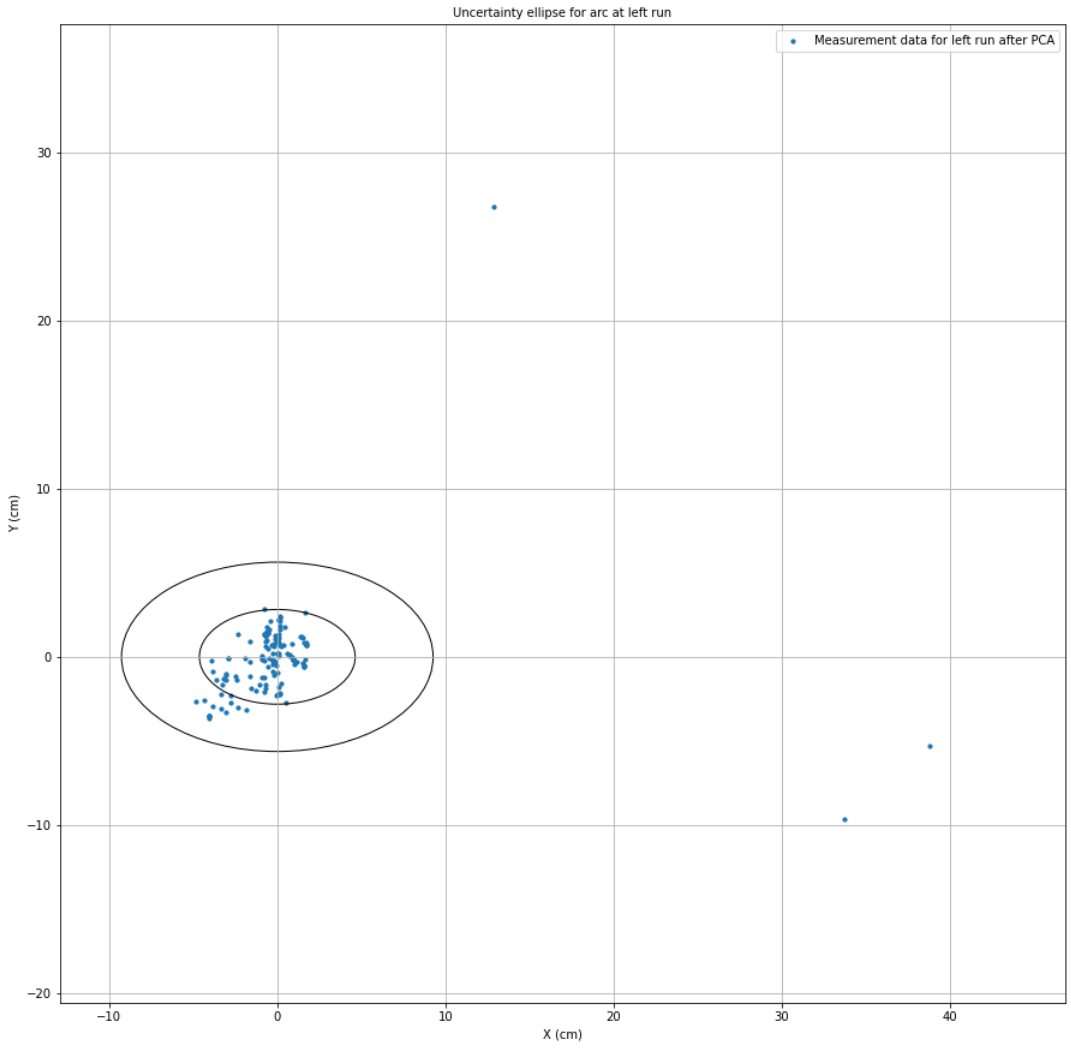

Figure 100: *Uncertainty ellipse for left position - small object configuration*

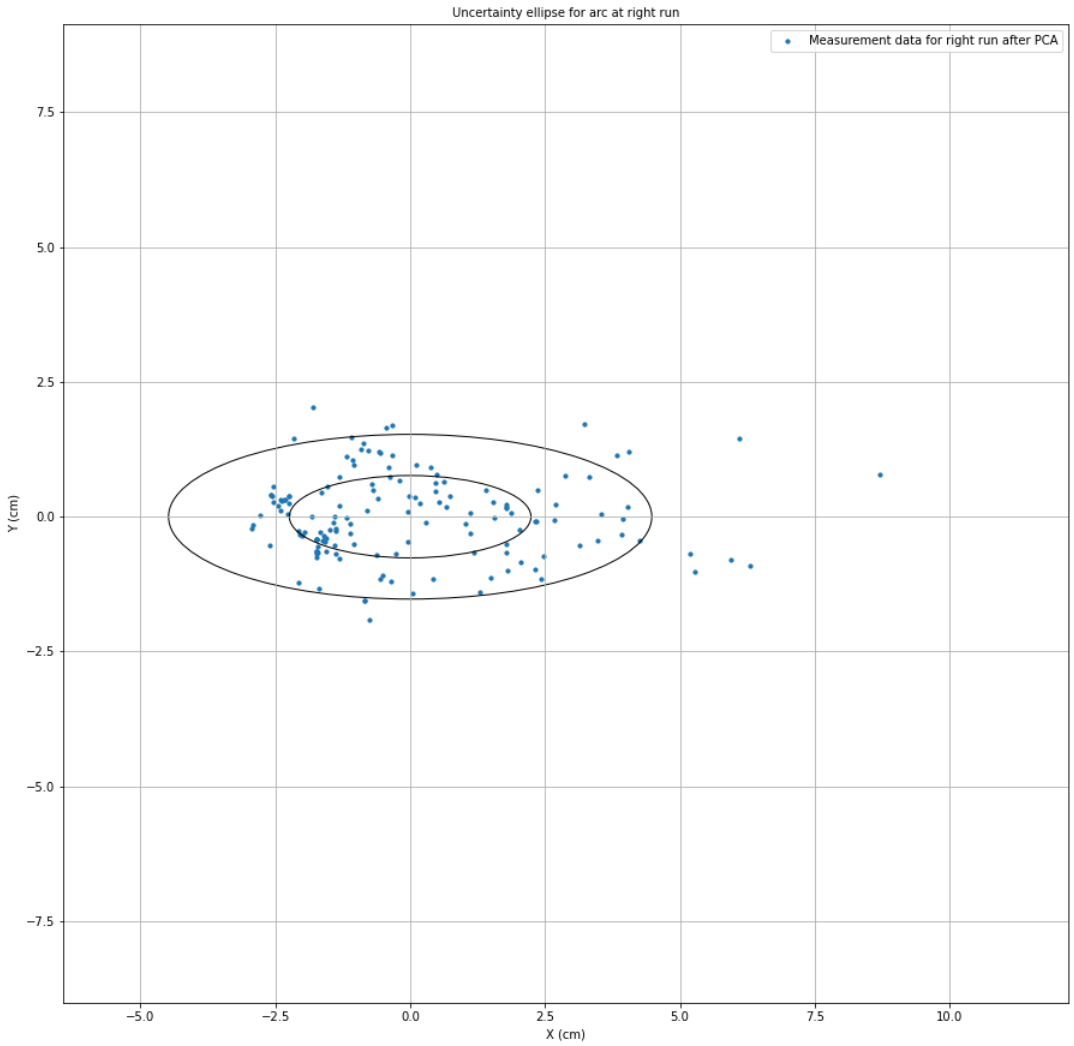

Figure 101: *Uncertainty ellipse for arc at right position - small object configuration*

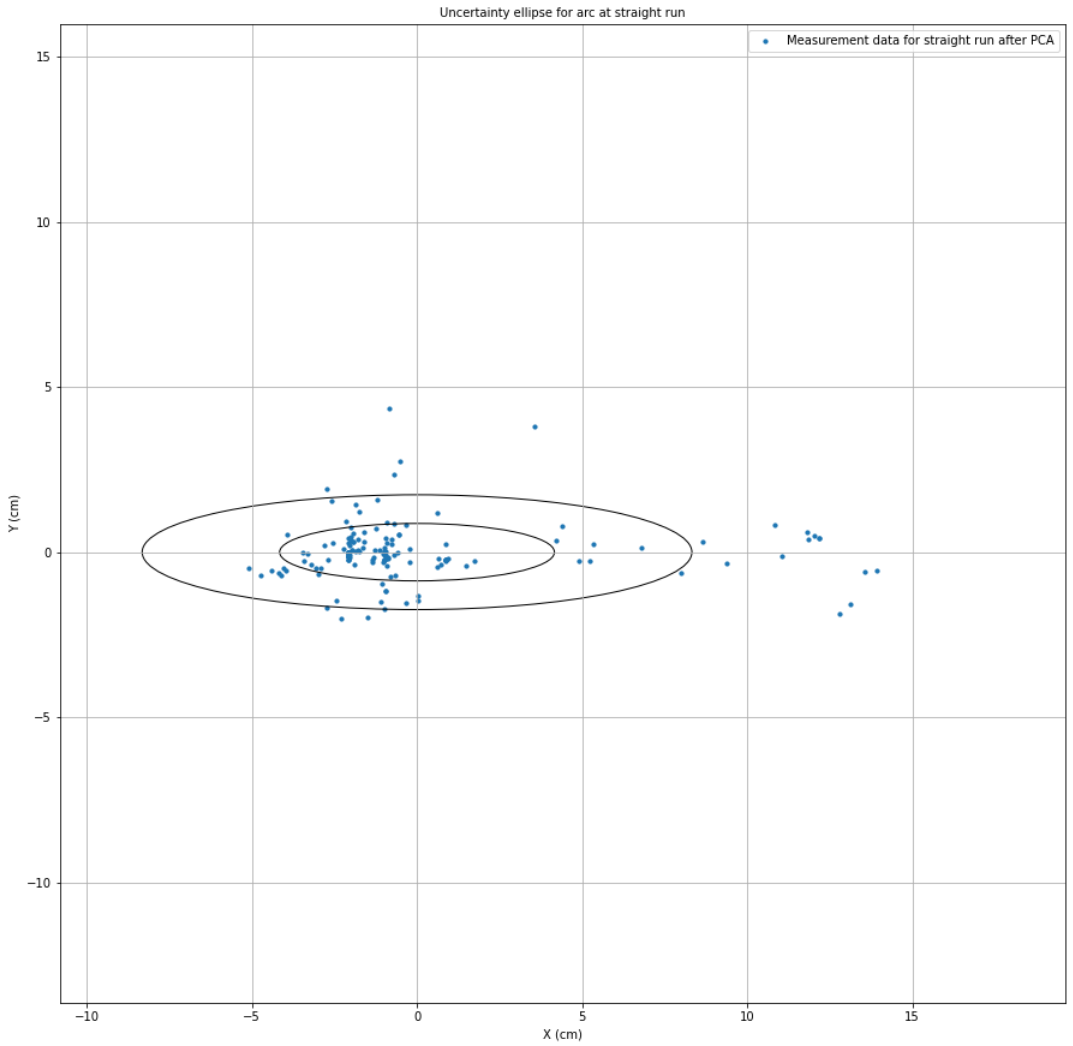

Figure 102: *Uncertainty ellipse for straight position - small object configuration*

Weight: Medium

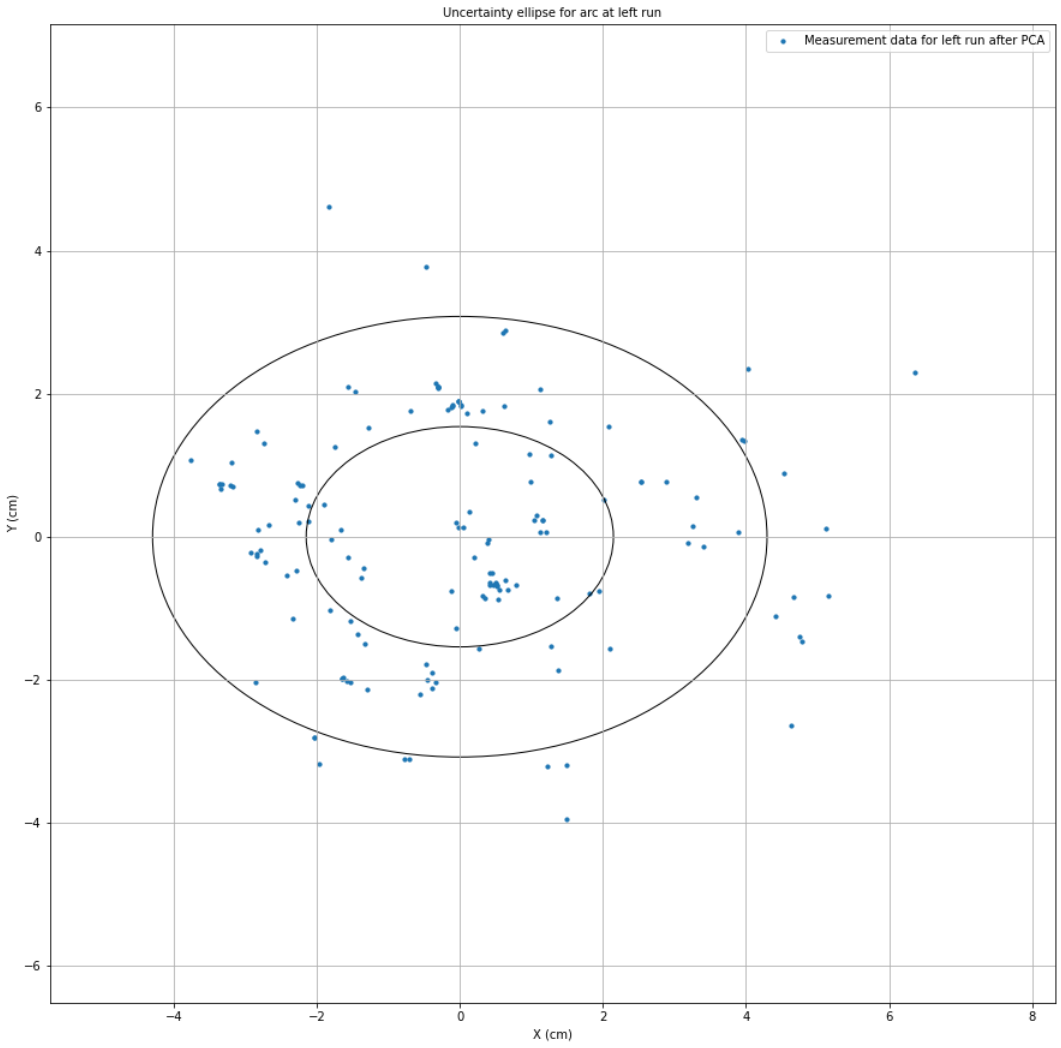

Figure 103: *Uncertainty ellipse for left position - medium object configuration*

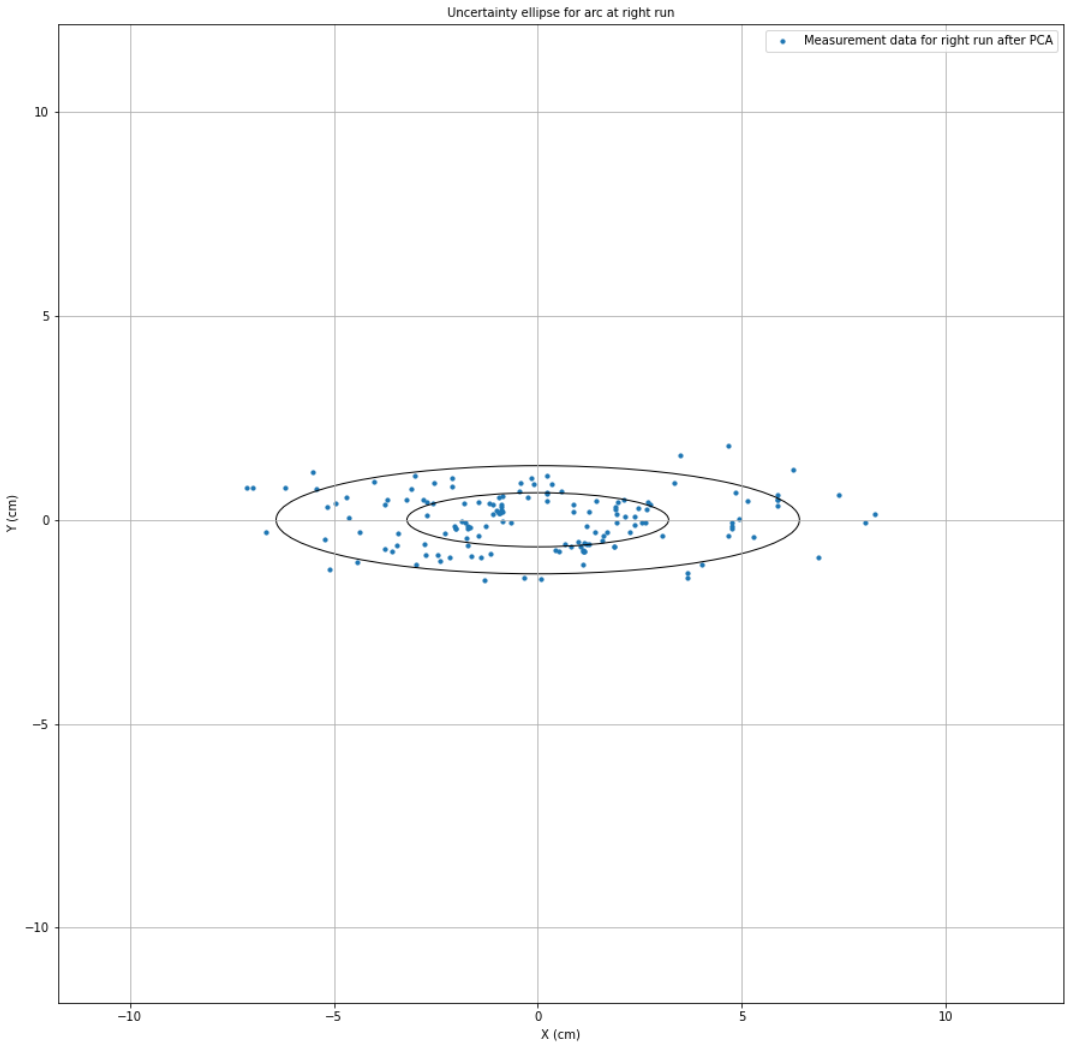

Figure 104: *Uncertainty ellipse for arc at right position - medium object configuration*

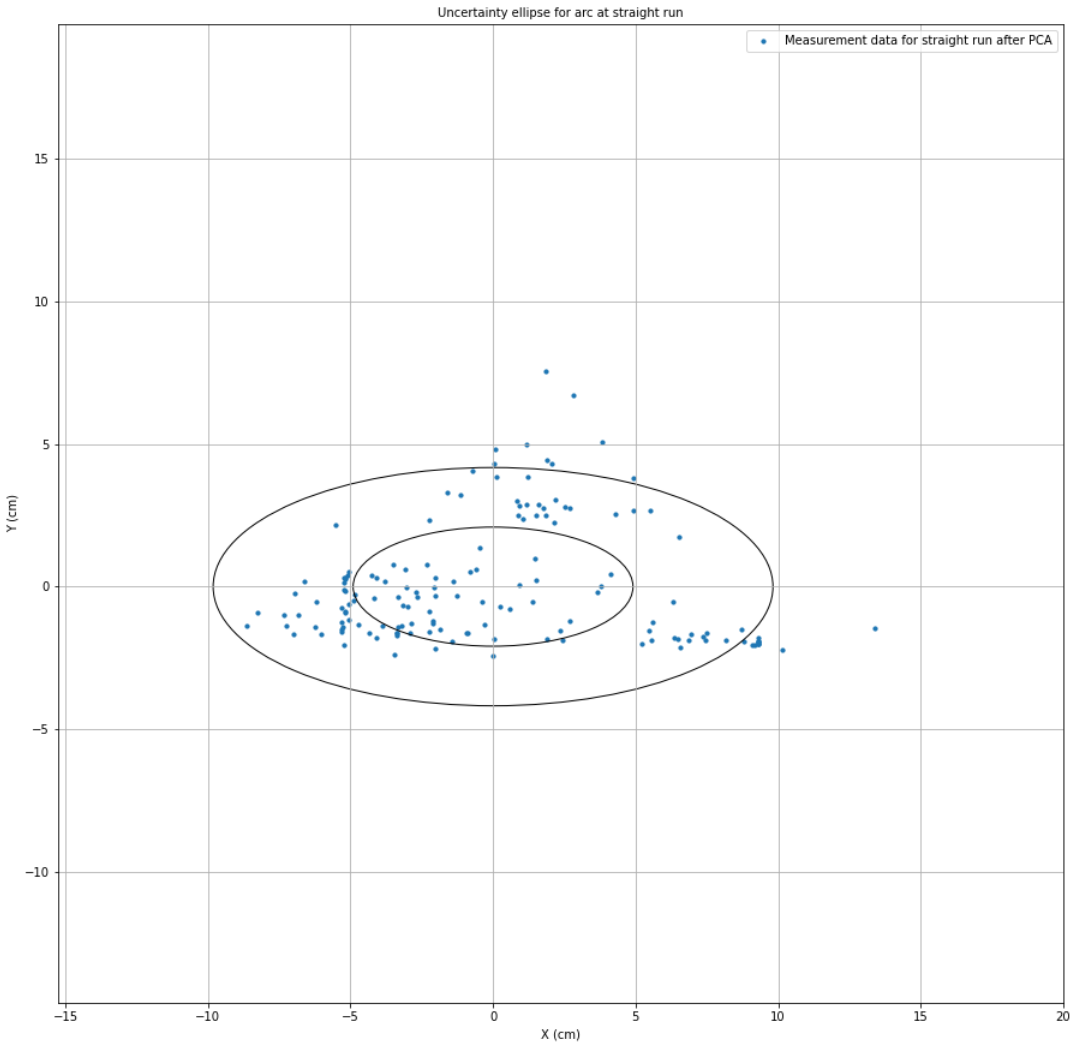

Figure 105: *Uncertainty ellipse for straight position - medium object configuration*

Weight: Large

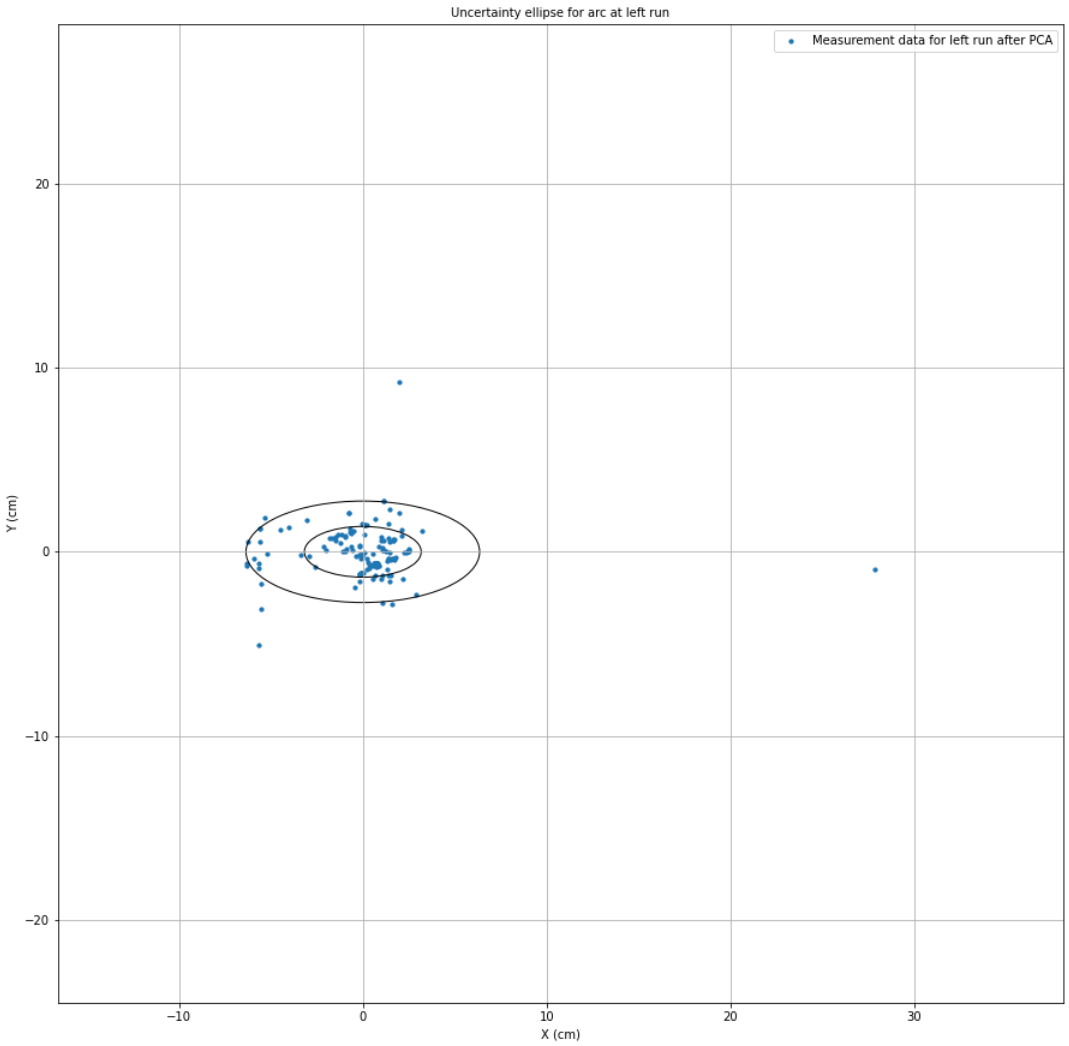

Figure 106: *Uncertainty ellipse for left position - large object configuration*

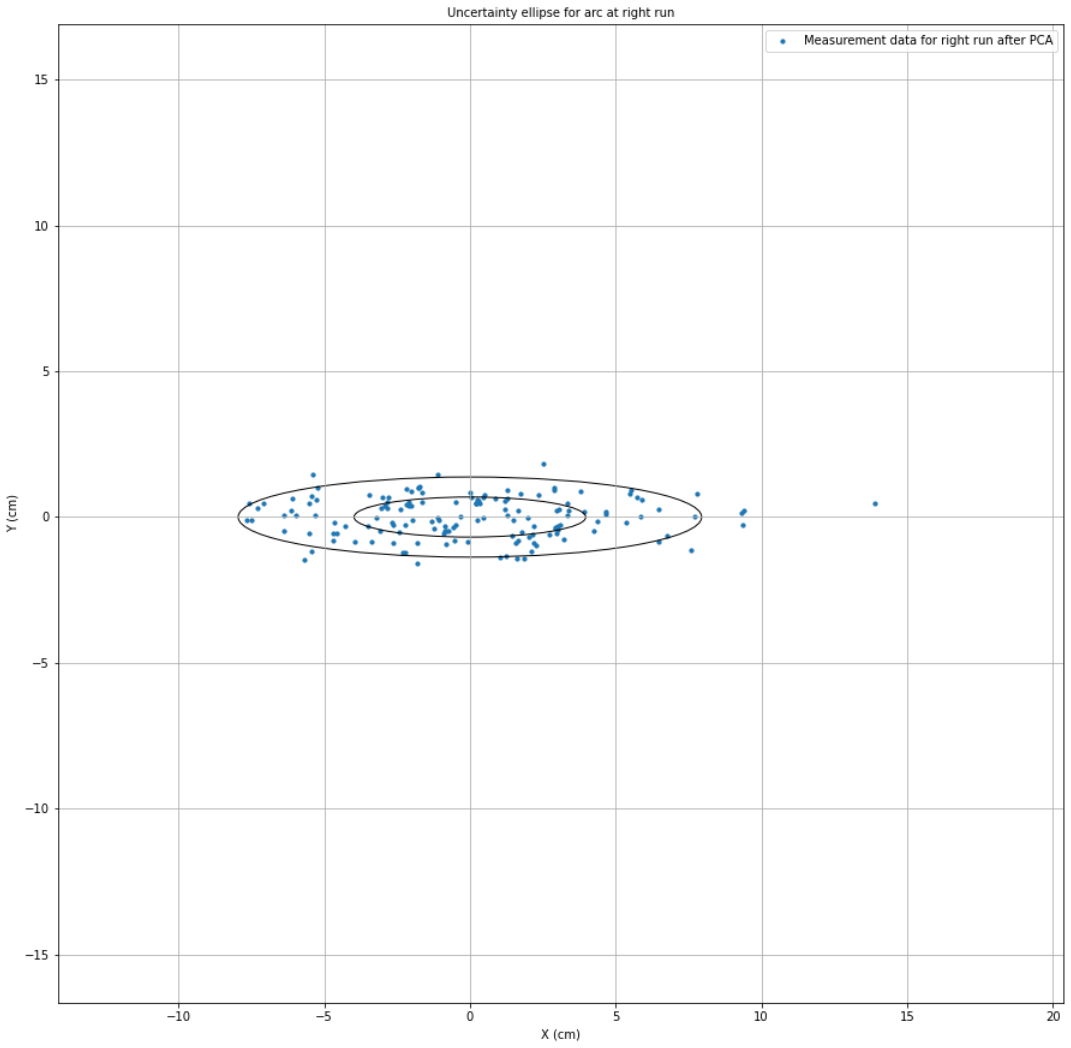

Figure 107: *Uncertainty ellipse for arc at right position - large object configuration*

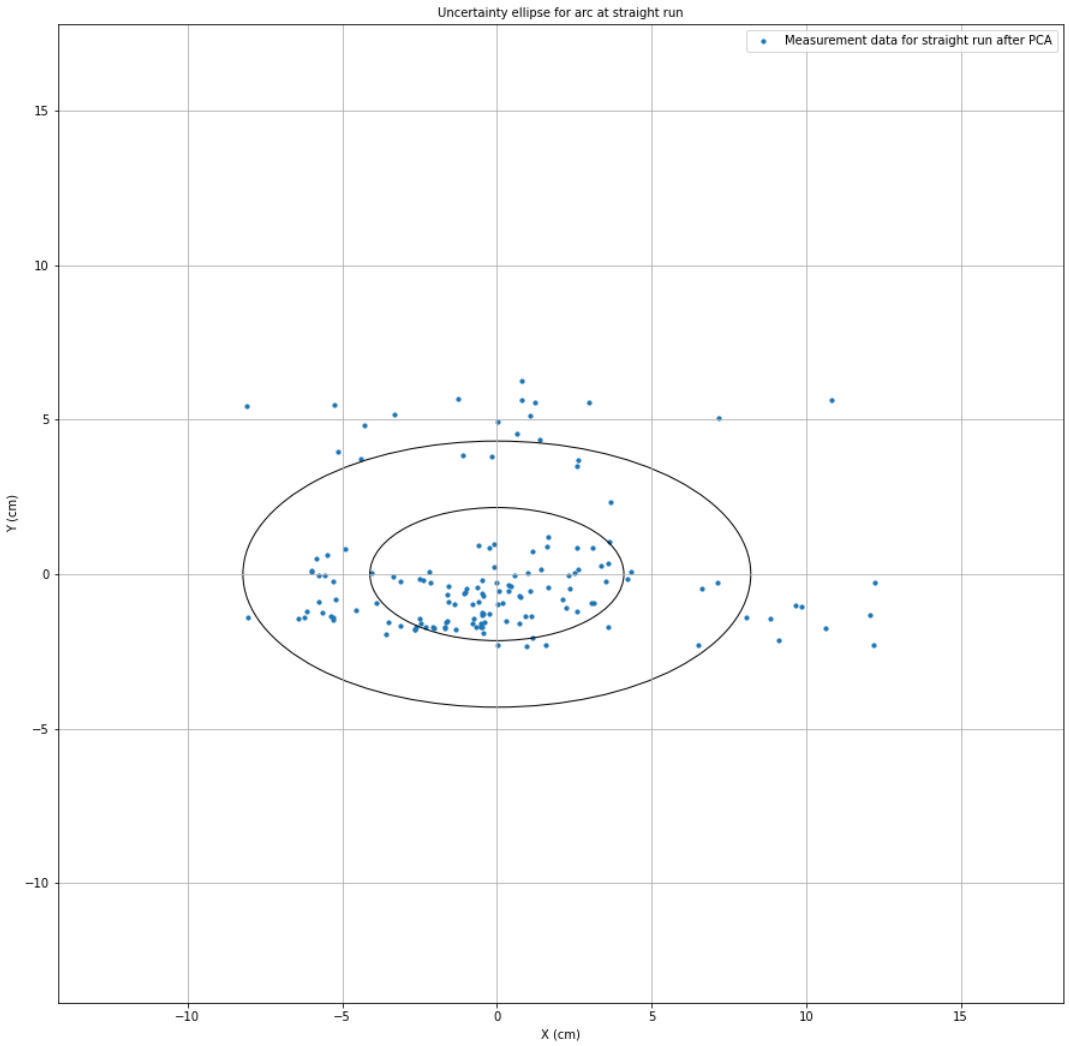

Figure 108: *Uncertainty ellipse for straight position - large object configuration*

5.4 Statistical parameters of our encoder measurement data

- Chi-square is performed using python and scipy library.
- Chi square test is used to obtain information whether the data fits the gaussian distribution.
- The Chi square function returns two parameters which are chi sqaure test statistics and P-value.
- P-value provides the strength with which null hypothesis is supported.
- Chi sqaure test statistics measure the deviation of actual results from the expected results.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{20}$$

  χ^2 = chi squared

  O_i = observed value

  E_i = expected value

- According to Fisher's test significance level is taken as 0.05. This parameter is used to determine whether the dataset fits the Gaussian distribution.
- We have measured the accuracy by finding the average of absolute deviation of the measurements from the ground truth.
- We estimated precision as the ground truth plus or minus average absolute deviation of measurements from the ground truth.
- Table 29 shows the statistical parameter values for the combined youbot measurement data.

```
import scipy.stats as stats

def chi_square(data):
    obs_data = np.array(data, dtype = np.float)
    obs_freq,_ = np.histogram(obs_data, bins=5)
    mean = np.mean(obs_data)
```

```
    std = np.std(obs_data)
    exp_data = np.random.normal(mean,std,len(data))
    exp_freq,_ = np.histogram(exp_data,
     ↪  bins=len(obs_freq))
    return stats.chisquare(obs_freq,exp_freq)
```

S.No	Object Size	Direction	Random Variable	Mean (cm)	Variance (cm^2)	Precision (cm)	Accuracy (cm)	Chi value	P value	Is Gaussian?
1	Small	Left	X	53.25	0.93	52.40+/-0.99	0.99	22.54	0.00	Yes
2	Small	Left	Y	-41.51	0.91	-38.30+/-3.20	3.20	30.02	0.00	Yes
3	Small	Left	Theta	-117.92	1.39	-116.88+/-1.48	1.48	35.28	0.00	Yes
4	Small	Right	X	-3.79	0.52	-5.50+/-1.71	1.71	14.15	0.01	Yes
5	Small	Right	Y	-29.65	0.76	-25.90+/-3.75	3.75	41.63	0.00	Yes
6	Small	Right	Theta	-58.09	1.37	-58.38+/-1.54	1.54	198.77	0.00	Yes
7	Small	Straight	X	27.31	0.59	25.40+/-1.91	1.91	56.92	0.00	Yes
8	Small	Straight	Y	-28.74	0.67	-24.90+/-3.84	3.84	26.53	0.00	Yes
9	Small	Straight	Theta	-92.04	1.44	-98.66+/-6.68	6.68	95.22	0.00	Yes
10	Medium	Left	X	52.49	1.13	52.40+/-1.08	1.08	37.58	0.00	Yes
11	Medium	Left	Y	-38.55	1.07	-38.30+/-0.97	0.97	15.93	0.00	Yes
12	Medium	Left	Theta	-118.43	1.25	-116.88+/-1.81	1.81	32.69	0.00	Yes
13	Medium	Right	X	-5.64	0.58	-5.50+/-0.30	0.30	139.8	0.00	Yes
14	Medium	Right	Y	-26.02	0.73	-25.90+/-0.44	0.44	39.49	0.00	Yes
15	Medium	Right	Theta	-61.52	1.65	-58.38+/-3.42	3.42	24.17	0.00	Yes
16	Medium	Straight	X	26.20	1.17	25.40+/-0.98	0.98	244.37	0.00	Yes
17	Medium	Straight	Y	-25.35	0.91	-24.90+/-0.71	0.71	82.22	0.00	Yes
18	Medium	Straight	Theta	-96.96	2.21	-98.66+/-4.51	4.51	70.64	0.00	Yes
19	Large	Left	X	49.83	1.05	52.40+/-2.63	2.63	123.24	0.00	Yes
20	Large	Left	Y	-35.16	0.99	-38.30+/-3.15	3.15	24.18	0.00	Yes
21	Large	Left	Theta	-117.38	0.98	-116.88+/-0.92	0.92	11.66	0.02	Yes
22	Large	Right	X	-6.90	0.56	-5.50+/-1.40	1.40	78.92	0.00	Yes
23	Large	Right	Y	-23.18	0.78	-25.90+/-2.72	2.72	157.31	0.00	Yes
24	Large	Right	Theta	-60.99	1.88	-58.38+/-3.55	3.55	34.70	0.00	Yes
25	Large	Straight	X	23.86	1.06	25.40+/-1.78	1.78	235.66	0.00	Yes
26	Large	Straight	Y	-22.24	0.86	-24.90+/-2.66	2.66	26.39	0.00	Yes
27	Large	Straight	Theta	-93.76	1.76	-98.66+/-5.09	5.09	6.24	0.18	Yes

Table 26: *Statistical parameters for the combined youbot measurement data*

5.5 Softwares used

- Numpy
- Pandas
- Matplotlib
- Seaborn
- sklearn

5.6 F-test

- F test is carried out so that we understand whether a pair of variables jointly are statistical significant or not.
- In order to find the joint statistical significance of a group of variables, we perform the f test to compare its variances.
- F test makes the following assumptions:
  - Both the variables or population follow gaussian distribution.
  - Samples in each of the variables are independent to each other.
- Null hypothesis : The variances of the two variables are equal.
- Alternate hypothesis : The variances of the two variables are not equal.
- The significance level is taken as 0.05, so if the P value is greater than the significance level 0.05, we accept the null hypothesis otherwise we reject it.
- If the variables are statistically significant, it means that the samples in those variables did not happen by chance, then we reject the null hypothesis.

```
import scipy.stats as stats

def ftest(x,y):
    F = np.var(x) / np.var(y)
    df1 = len(x) - 1,
    df2 = len(y) - 1
    alpha = 0.05
    p_value = stats.f.cdf(F, df1, df2)
    print("F value: ",F)
    print("p value: ",p_value)
    if p_value < alpha:
        print("Hypothesis Rejected\n")
    else:
        print("Hypothesis Accepted\n")
```

S.No	Variance 1	Variance 2	F-Value	P-Value	Hypothesis
1	Left X Small	Left X Medium	0.82	0.12	Accepted
2	Left X Small	Left X Large	0.89	0.24	Accepted
3	Left X Medium	Left X Small	1.22	0.88	Accepted
4	Left X Medium	Left X Large	1.08	0.68	Accepted
5	Left X Large	Left X Small	1.13	0.77	Accepted
6	Left X Large	Left X Medium	0.93	0.33	Accepted
7	Left Y Small	Left Y Medium	0.85	0.16	Accepted
8	Left Y Small	Left Y Large	0.92	0.31	Accepted
9	Left Y Medium	Left Y Small	1.18	0.84	Accepted
10	Left Y Medium	Left Y Large	1.08	0.67	Accepted
11	Left Y Large	Left Y Small	1.09	0.70	Accepted
12	Left Y Large	Left Y Medium	0.93	0.34	Accepted
13	Left Theta Small	Left Theta Medium	1.11	0.74	Accepted
14	Left Theta Small	Left Theta Large	1.42	0.98	Accepted
15	Left Theta Medium	Left Theta Small	0.91	0.26	Accepted
16	Left Theta Medium	Left Theta Large	1.28	0.93	Accepted
17	Left Theta Large	Left Theta Small	0.71	0.02	Rejected
18	Left Theta Large	Left Theta Medium	0.78	0.07	Accepted
19	Right X Small	Right X Medium	0.92	0.26	Accepted
20	Right X Small	Right X Large	0.93	0.33	Accepted
21	Right X Medium	Right X Small	1.12	0.75	Accepted
22	Right X Medium	Right X Large	1.04	0.59	Accepted
23	Right X Large	Right X Small	1.08	0.68	Accepted
24	Right X Large	Right X Medium	0.97	0.43	Accepted
25	Right Y Small	Right Y Medium	1.04	0.59	Accepted
26	Right Y Small	Right Y Large	0.97	0.43	Accepted
27	Right Y Medium	Right Y Small	0.96	0.40	Accepted
28	Right Y Medium	Right Y Large	0.94	0.36	Accepted
29	Right Y Large	Right Y Small	1.03	0.57	Accepted
30	Right Y Large	Right Y Medium	1.07	0.66	Accepted

Table 27: *F-test Table (1/2)*

S.No	Variance 1	Variance 2	F-Value	P-Value	Hypothesis
31	Right Theta Small	Right Theta Medium	0.83	0.13	Accepted
32	Right Theta Small	Right Theta Large	0.73	0.03	Rejected
33	Right Theta Medium	Right Theta Small	1.2	0.86	Accepted
34	Right Theta Medium	Right Theta Large	0.88	0.22	Accepted
35	Right Theta Large	Right Theta Small	1.37	0.97	Accepted
36	Right Theta Large	Right Theta Medium	1.14	0.78	Accepted
37	Straight X Small	Straight X Medium	0.51	0.00	Rejected
38	Straight X Small	Straight X Large	0.56	0.00	Rejected
39	Straight X Medium	Straight X Small	1.98	1.00	Accepted
40	Straight X Medium	Straight X Large	1.12	0.71	Accepted
41	Straight X Large	Straight X Small	1.84	1.00	Accepted
42	Straight X Large	Straight X Medium	0.91	0.29	Accepted
43	Straight Y Small	Straight Y Medium	0.74	0.04	Rejected
44	Straight Y Small	Straight Y Large	0.78	0.07	Accepted
45	Straight Y Medium	Straight Y Small	1.36	0.97	Accepted
46	Straight Y Medium	Straight Y Large	1.06	0.63	Accepted
47	Straight Y Large	Straight Y Small	1.28	0.93	Accepted
48	Straight Y Large	Straight Y Medium	0.95	0.38	Accepted
49	Straight Theta Small	Straight Theta Medium	0.65	0.01	Rejected
50	Straight Theta Small	Straight Theta Large	0.82	0.12	Accepted
51	Straight Theta Medium	Straight Theta Small	1.53	0.99	Accepted
52	Straight Theta Medium	Straight Theta Large	1.26	0.92	Accepted
53	Straight Theta Large	Straight Theta Small	1.22	0.88	Accepted
54	Straight Theta Large	Straight Theta Medium	0.83	0.09	Accepted

Table 28: *F-test Table (2/2)*

The above table proves that object's weight indeed plays a significant role in determining the final placing pose because null hypothesis was rejected for for various configurations of weights.

5.7 Single Measurement vs filtered measurement

In order to compare the filtered measurement with the single camera measurement, we perform the following steps:

- We take the first reading out of the 50 readings (from which we supposedly take the average.) for 20 measurements.

- Find the mean and variance of those 20 measurements and compare it with the filtered measurement.

S.No	Object Size	Direction	Random Variable	Mean (cm)	Variance (cm^2)
1	Small	Left	X filtered	53.25	0.93
2	Small	Left	Y filtered	-41.51	0.91
3	Small	Left	Theta filtered	-117.92	1.39
4	Small	Left	X unfiltered	53.93	0.79
5	Small	Left	Y unfiltered	-41.5	0.81
6	Small	Left	Theta unfiltered	-118.13	1.19

Table 29: *Comparison of filtered and unfiltered measurement data using Mean and variance*

- Observations:
  - We have chosen the "Small object - Left motion" as reference for comparison.
  - It can be seen from the above table that there are variations in the variance in the filtered and unfiltered measurement data.