Scientific Experimentation and Evaluation Assignment 4

Aaqib Parvez Mohammed
Chetan Sidnal
Mihir Patil
Sushma devaramani

June 28, 2018

Contents

List of Figures

List of Tables

1 Experiment 1: Manual Motion Observation

Aim: Manually measure the observable pose variation for three different constant velocity motions(Straight line, an arc to the left and an arc to the right) of LEGO NXT differential drive robot.

1.1 Setup

- The measurement system for the experiment includes a white sheet, a LEGO NXT differential drive robot, two pencils and a scale.

- The **device under test(DUT)** is the LEGO NXT differential drive robot.

- Clamp the sheet on a flat surface.

- Construct a robot equipped with two pencils in the front, such that the start and end positions can be marked. This constitutes the measurement facility.

- Install *Lejos* OS on the robot.

- Program the robot for it to run in three different constant velocity motions.
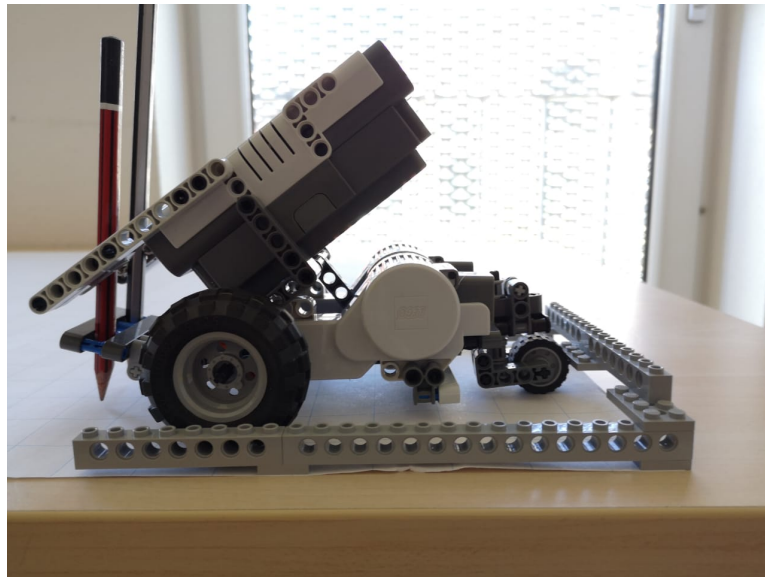

Figure 1: Side view

Figure 2: Isometric view

Figure 3: Top view

Figure 4: front view

1.2 Procedure

- The experimental setup is prepared.

- The starting position of the robot is marked with the help of the points drawn by the pencils.

- The center of the line joining the two points is recorded which will be the starting position of the robot.

- The robot is programmed to run in a straight line forward motion at a constant speed.

- Once the robot stops, the center of the line joining the last points drawn by the pencils will be the stop position.

- The **Measurand** i.e., the relative coordinates of the robot are measured by the difference in the coordinates of the starting position and the stopping position.

- **Measurement System:** Our measurement system consists of robot placed on a cardboard sheet. Two pencils are attached in the front to mark the position. As the robot is moved, the path is traced on the sheet. After reaching the end position, the final pose is marked. The variation in the path is observed and the error in pose is measured.

- The **Measurement result** i.e., the pose variation is obtained by finding the difference between the observed values and their mean.

- Repeat the above experiment with a program to run the robot with constant angular and translational velocities for a fixed time period, such that it describes an arc to the left.

- Repeat the above experiment with a program to run the robot with constant angular and translational velocities for a fixed time period, such that it describes an arc to the right.

1.3 Expected problems

- Parallax error.

- Positional errors.

- There might be slipping of wheels.

- The tip of the pencils might break during the motion of the robot.

- The pencils will have to be sharpened as the nibs might get blunt due to continuous drawing of lines which might result in measurement errors.

1.4   Expected performance

- The Rotation sensor measures the motor rotations with the accuracy of +/- one degree for one rotation(360 degree). Hence the accuracy in distance traveled for one rotation will be +/- 0.05 cm.

- Reference: LEGO NXT: Features & Limitations.

- Assuming experiment distance = 70.4 cm.

- Expected accuracy is ±2 and expected precision is 70.4±0.2cm.

1.4.1   Reasons

- The slippage of the wheels may cause poor accuracy and precision.

- The internal errors from the DUT may cause poor accuracy but the precision won't be affected significantly, since the DUT won't be changed while doing the measurement.

1.5   Execution of the experiment:

- The starting position of the robot was marked using two pencils which were placed on either side of the robot.

- The same points were used as the starting points while repeating the experiment.

- The position of the back wheel was also kept constant by drawing a line and using the same line as reference for future run of experiments.

- The position of the back wheel impacted the outcome of the experiment as a small deviation from the reference line would result in a considerable deviation of the outcome.

- The theta values were calculated by drawing a perpendicular to the line joining the two observed end points and calculating the angle using the formula $tan^{-1}(\frac{y}{x})$. As shown in the below figure.
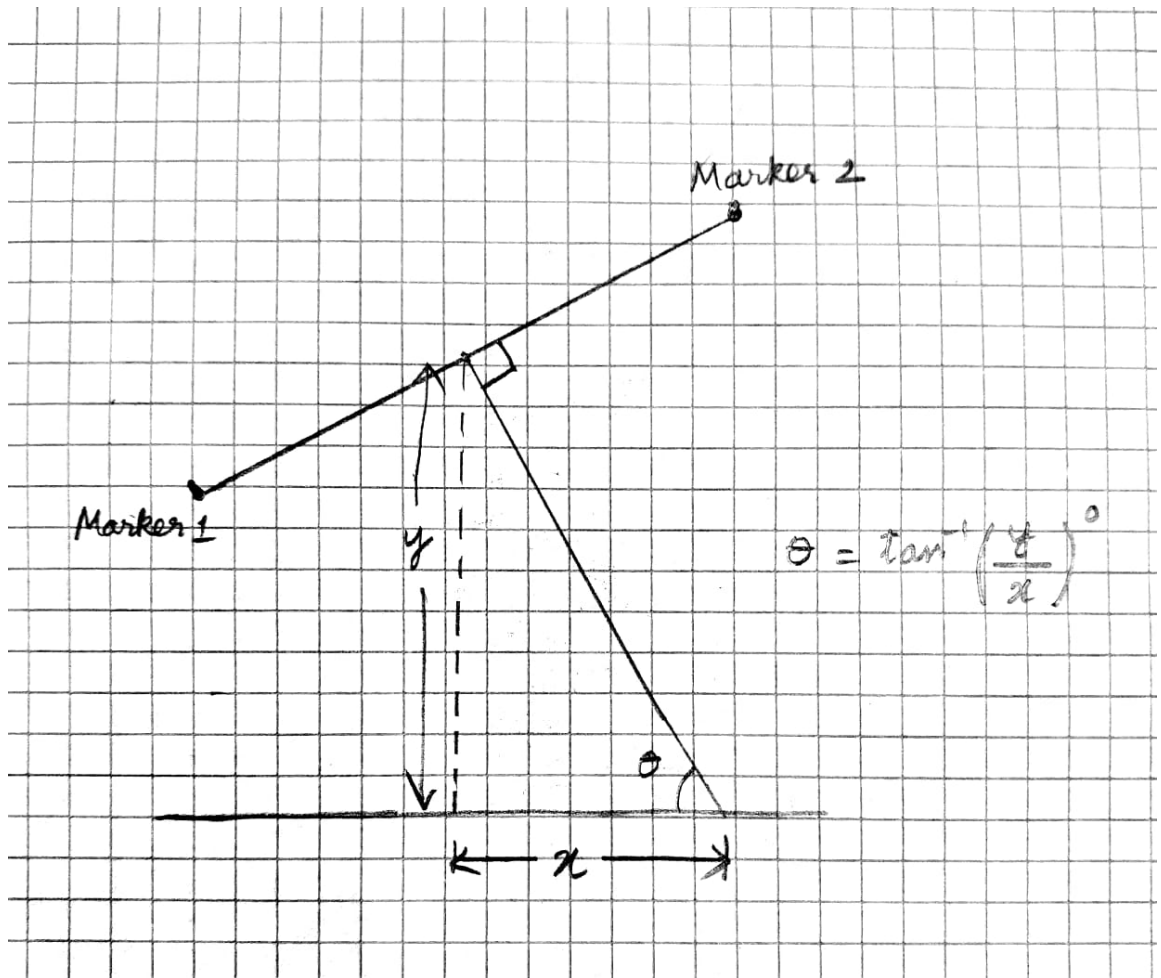
Figure 5: Finding out θ for robot orientation

- Lego NXT 2.0 software was used to program the robot with the following parameters.
  No. of rotations = 4
  Power = 40

- The battery voltage before the experiment = 8.2V.
  The battery voltage after the experimenter = 4.2V.

- The observed data has been recorded in the ".csv" format.

- There was no pre-processing of data required as no outliers were detected.

- Graph has been plotted for all three motions.

11

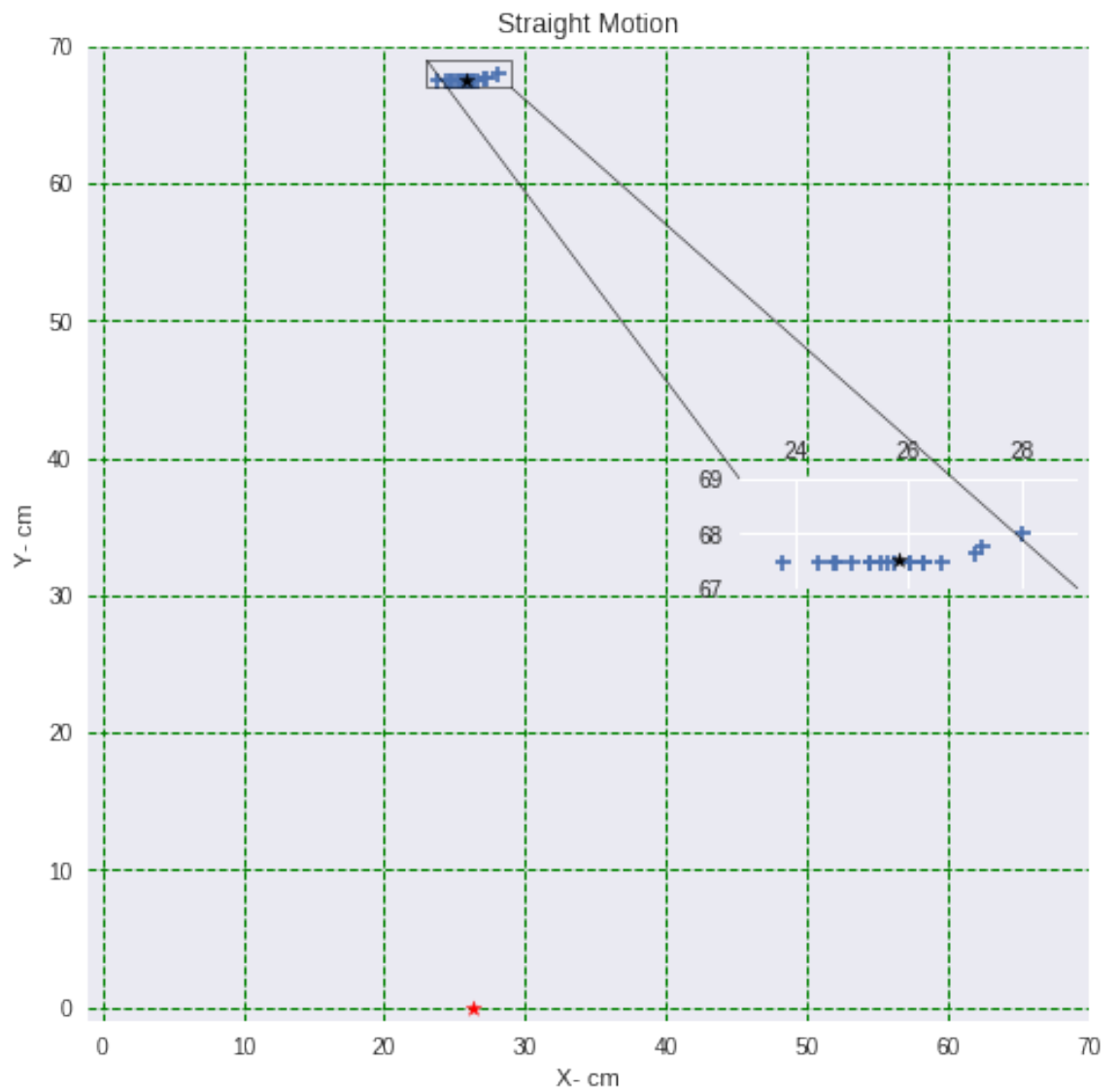Note : All x,y measurements are in centimeters (cm)

1.5.1 Straight

Figure 6: 20 iteration of moving straight

Fitting the distribution for the obtained data:
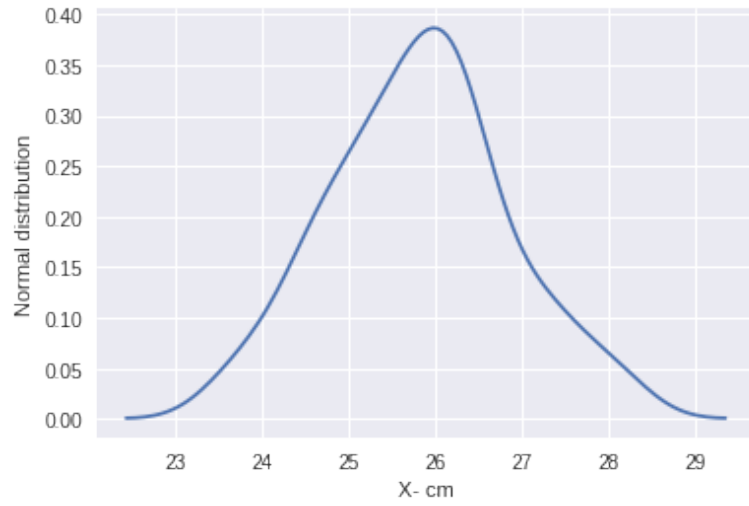**(X, Y) points for straight line**

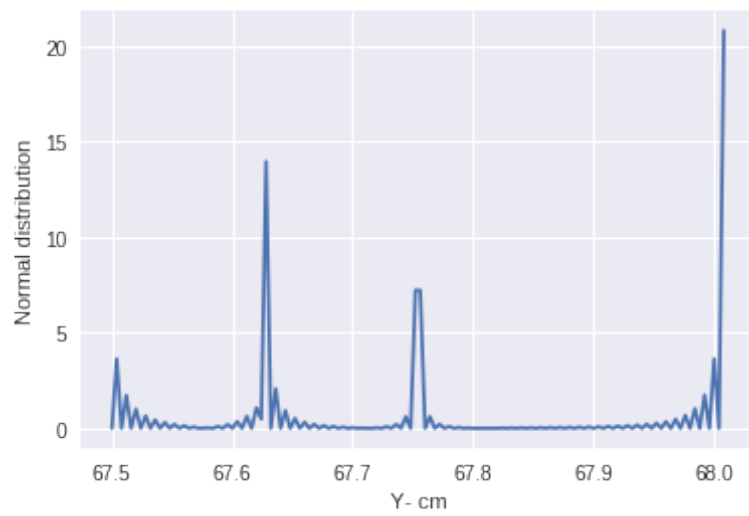Figure 7: Normal distribution for straight motion (X in cm)

Figure 8: Normal distribution for straight motion (Y in cm)

Figure 9: Normal distribution for left motion (X in cm)

Figure 10: Normal distribution for left motion (Y in cm)

Figure 11: Normal distribution for right motion (X in cm)
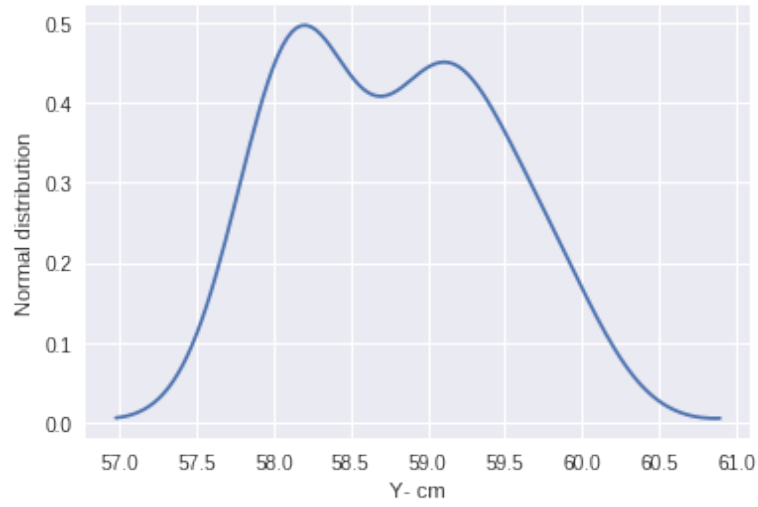
Figure 12: Normal distribution for right motion (Y in cm)

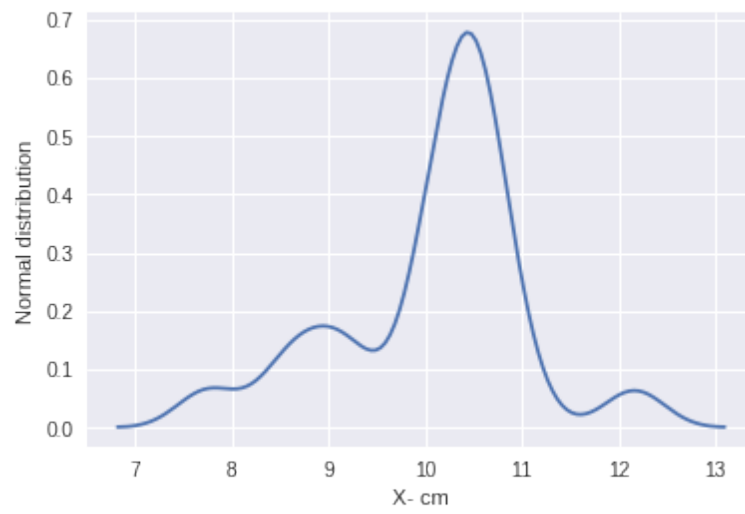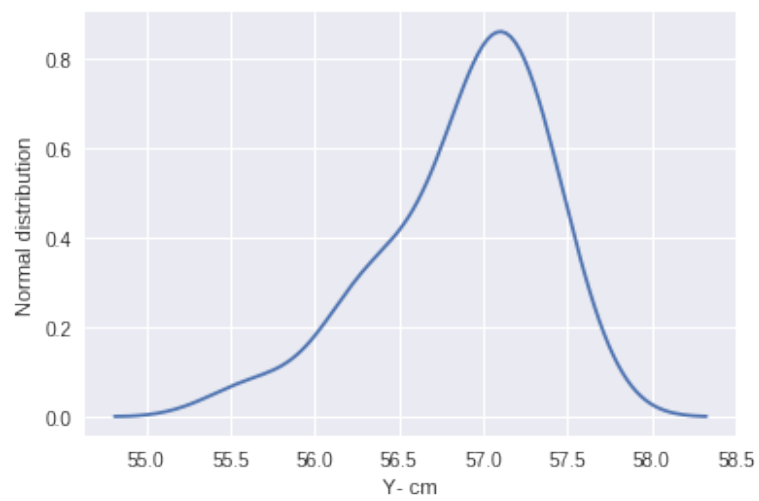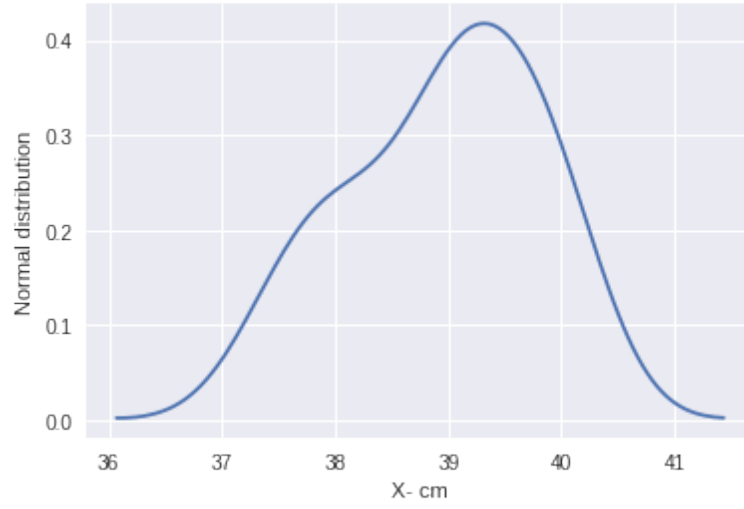- $Range(x, y) = ([23.8 - 28.1], [67.5 - 68.1])$
- $Mean(x, y) = 25.8, 67.5$
- $\sigma(x, y) = 1.1, 0.1$
- $Precision(x, y) = 25.8 \pm 1.1, 67.5 \pm 0.1$
- $Accuracy(x, y) = x \pm 0.9, y \pm 0.1$

1.5.2 Right
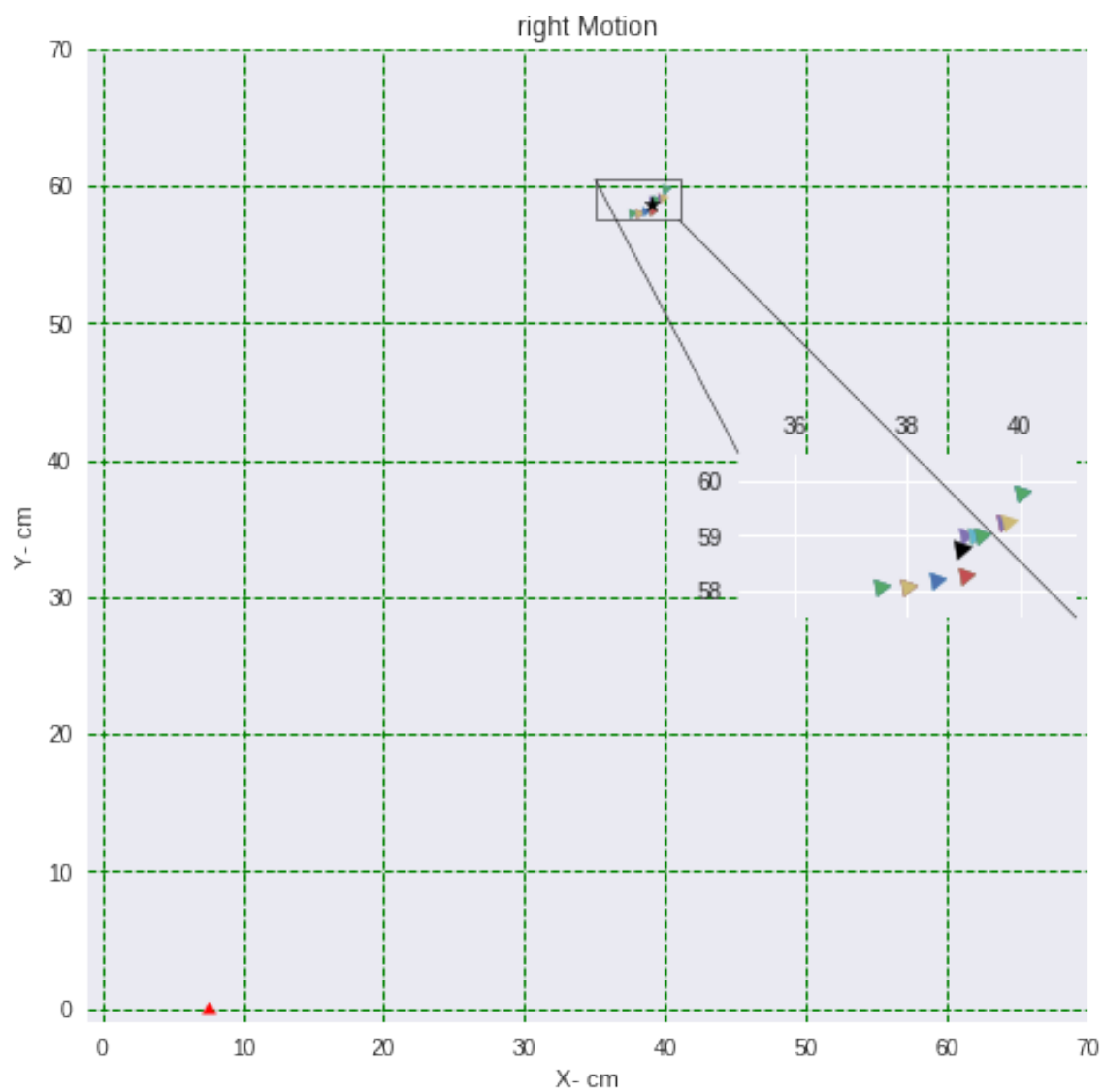
Figure 13: 20 observations of right arc motion
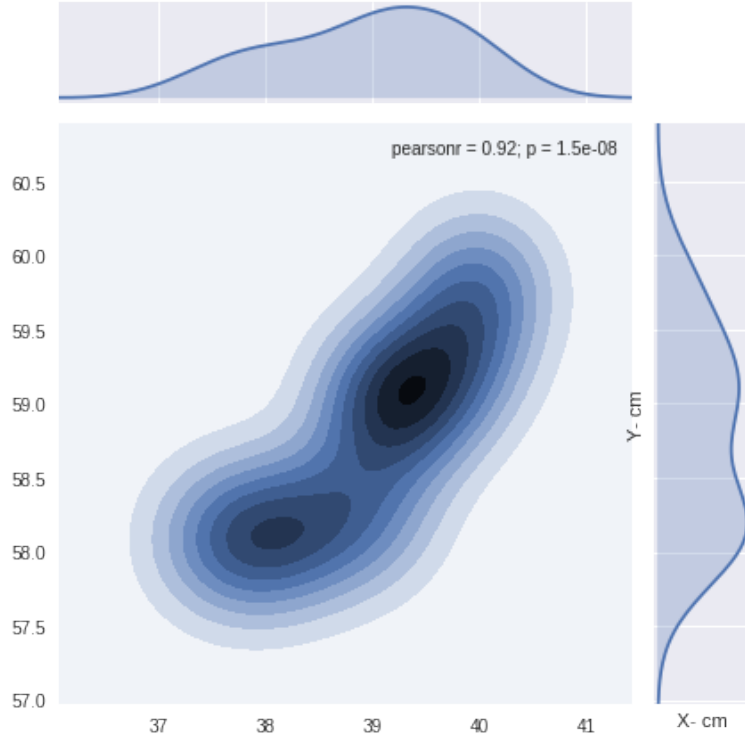
Figure 14: Density plot of (X, Y) points for right arc motion

In the above plot it can be observed that the distribution of X and Y positions for right arc motion is gaussian.

- $Range(x, y) = ([37.5 - 40.0], [58.0 - 59.8])$
- $Mean(x, y) = 38.9, 58.8$
- $Precision(x, y) = 38.9 \pm 0.8, 58.8 \pm 0.6$
- $Accuracy(x, y) = x \pm 0.5, y \pm 0.5$
- $\sigma(x, y) = 0.8, 0.6$
- $Mean(\theta) = 40.6 degree$
- $Accuracy(\theta) = \theta \pm 5 deg$
- $\sigma = 1.1 deg$
- $range(\theta) = 38.8 - 43.3$

1.5.3 Left

Figure 15: 20 iteration of moving left
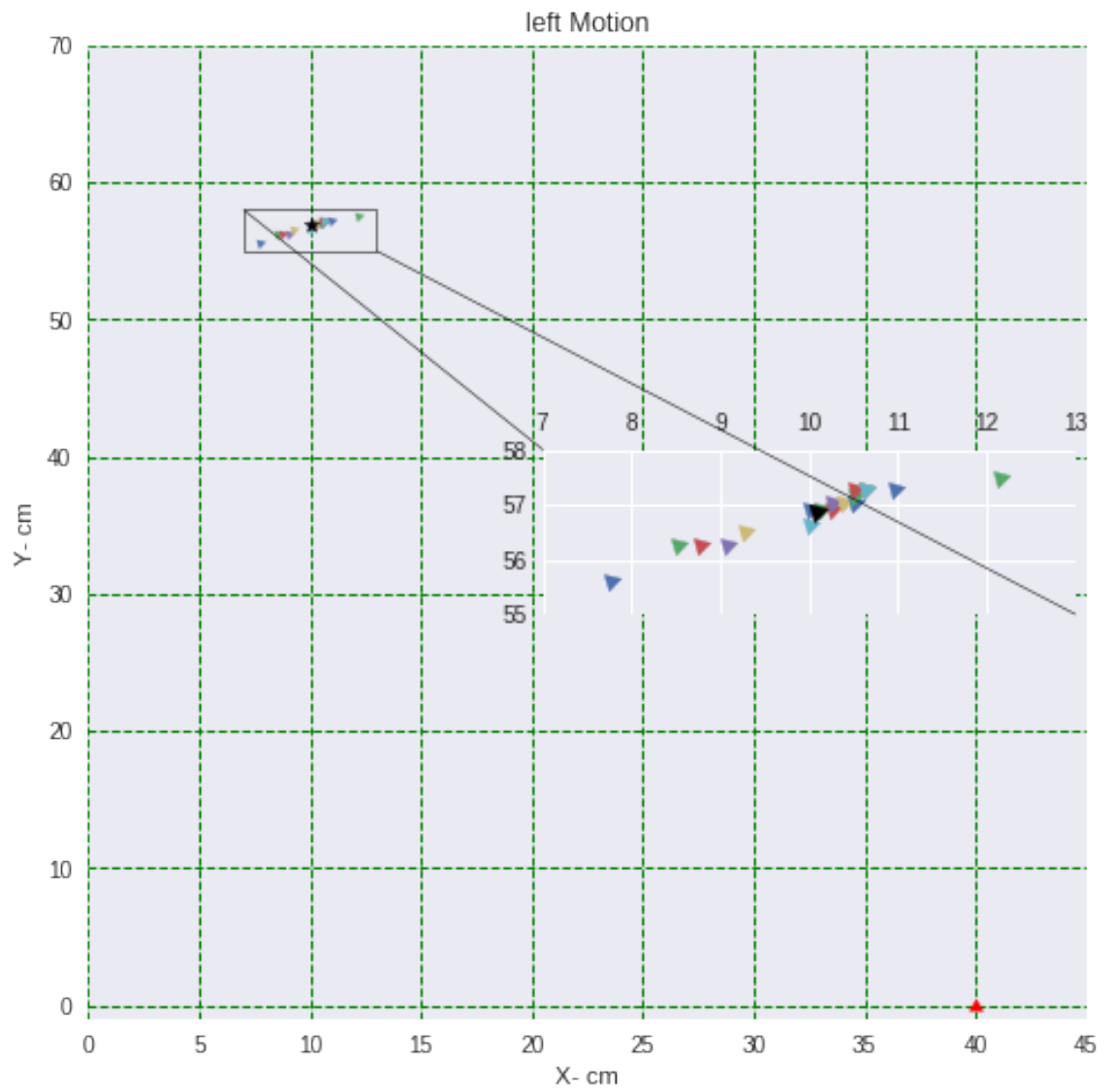
Figure 16: Density plot of (X, Y) for moving left

In the above figure it can be observed that the (X, Y) points obtained for left arc motion fit a gaussian distribution.

- $Range(x, y) = ([7.8 - 12.2], [55.6 - 57.5])$
- $Mean(x, y) = 10.1, 56.9$
- $Precision(x, y) = 10.1 \pm 2.0, 56.9 \pm 1.5$
- $Accuracy(x, y) = x \pm 0.5, y \pm 0.5$
- $\sigma(x, y) = 1.0, 0.5$
- $Mean(\theta) = 45.7 degree$
- $Accuracy(\theta) = \theta 7 \pm deg$
- $\sigma = 1.3 deg$
- $range(\theta) = 43.2 - 48.3$

Boxplots

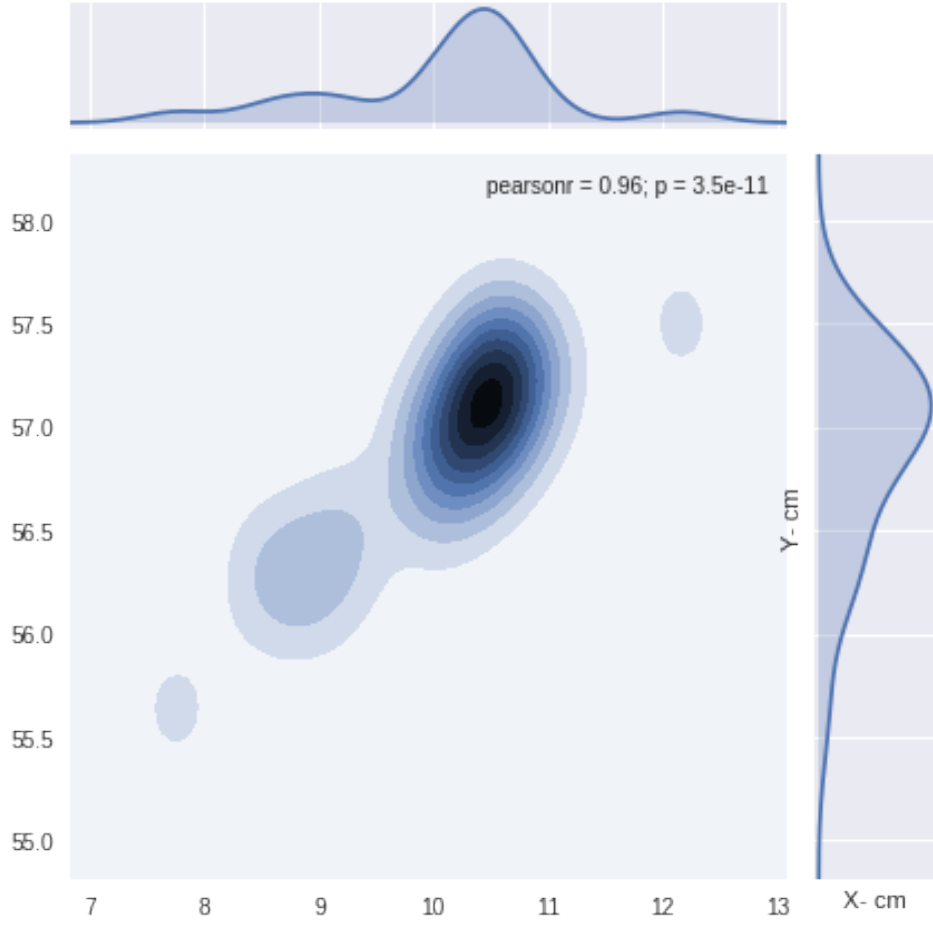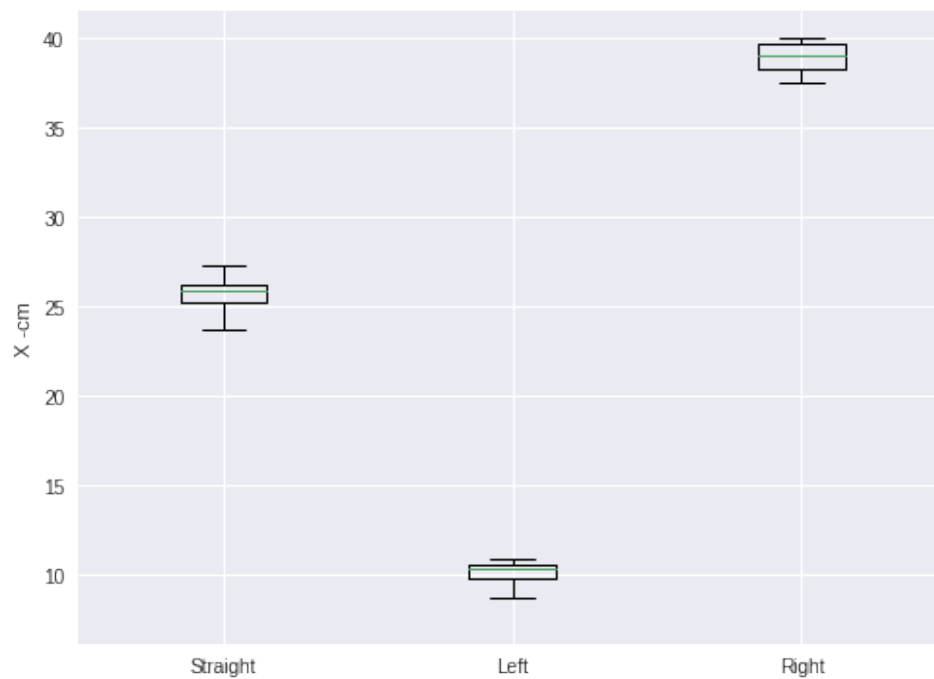Figure 17: Box plot of X-values for straight line motion, left and right turn
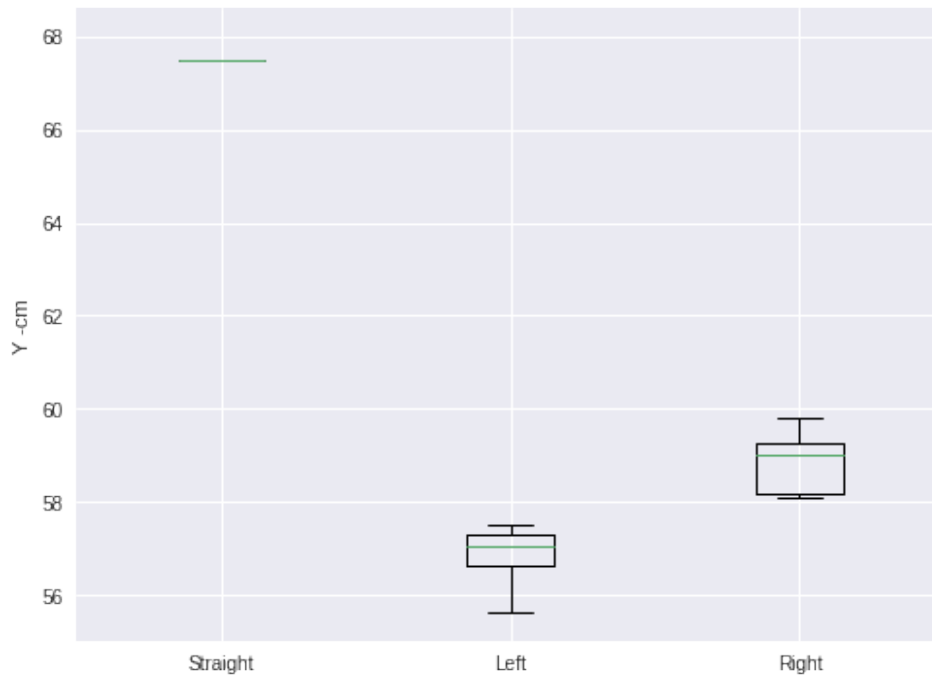
Figure 18: Box plot of Y-values for straight line motion, left and right turn

- 2.3 The chi-square test indicates that the observed data is a normal distribution.

```
In [9]: ks , p = stats.normaltest(straight_xy)
alpha = 1e−3
# null hypothesis: x comes from normal distribution
if p.all() < alpha:
print("The null hypothesis can be rejected")
else:
print("The null hypothesis cannot be rejected")

[Out]: The null hypothesis cannot be rejected

In [10]: ks , p = stats.normaltest(right_xy)
alpha = 1e−3
if p.all() < alpha:
print("The null hypothesis can be rejected")
else:
print("The null hypothesis cannot be rejected")
[Out]: The null hypothesis cannot be rejected
```

```
In [11]: ks , p = stats.normaltest(left_xy)
alpha = 1e-3
if p.all() < alpha:
print("The null hypothesis can be rejected")
else:
print("The null hypothesis cannot be rejected")
[Out]: The null hypothesis cannot be rejected
```

- Deliverable 2.2: The functions used for plotting are from the seaborn library and use a kernel density method to obtain a 2-D density plot of the x and y coordinates.

- Deliverable 2.5:

  Straight: In the case of straight line motion the readings obtained for Y coordinates appear to be both precise and accurate, whereas the X coordinates appear to be precise but not very accurate.

  Right: The readings obtained for Y coordinates appear to be both highly precise and accurate, whereas the X coordinates appear to be precise but not very accurate.

  Left: The readings obtained for X coordinates appear to have low precision and accuracy, whereas the Y coordinates appear to be precise with a low accuracy.

- **Comparison of Actual performance with expected:**

  - The expected distance to be traveled was 70.4 cm.
  - The mean actual distance travelled was calculated by applying distance formula on initial and end points, which is equal to 67.5 cm.
  - Estimated error = 2 cm
  - Actual error = 2.9 cm
  - The difference in the actual and expected error can be due to systematic or/and random errors.

2 Calibration of an Optical Tracking System

2.1 Setup for camera calibration

- The camera calibration process is done on a checkerboard (calibration tool).

- The *MATLAB support package for USB webcam* for camera calibration is installed.

- The camera calibration toolbox for MATLAB is installed.

- The webcam (Microsoft Lifecam) is connected to the laptop and the connectivity is checked by running *webcam list* command in MATLAB, which shows the list of all webcams connected to PC.

- The webcam is mounted on a static paper punch, while capturing the images.

- The Auto-focus is turned on and the camera is made to adjust to the distance of the checkerboard.

- Then the Auto-focus is turned *off* by running the following command in MATLAB,

  cam.FocalMode = manual

- The images of checkerboard are captured from different perspectives (different position and orientation in the camera's field of view). A total of 25 images are captured and saved for the calibration process.

2.2 Estimation of number of images

- Since the object in the world is 3-dimensional and the captured images are 2D (co-planar), we loose one dimension. In order to regain the actual dimensionality during the calibration, various number of images are captured in different positions and orientations.

- As a rule of thumb, 20-40 images is quite enough for the process. For our calibration we have captured 25 images in different position and orientations.

- The idea behind the capturing enough images is to cover the field of view, and to have a good distribution of 3D orientations of the board.

2.3 Description of the camera parameters

After the calibration, the list of parameters obtained are of two types.

- **Intrinsic parameters**

  - **Focal length:** Distance between the lens and the image sensor when the object is in focus, usually stated in millimeters. The focal length in pixels is stored in the 2x1 vector fc.

  - **Principal point:** A point at the intersection of the optical axis and the image plane. The principal point coordinates are stored in the 2x1 vector cc.

  - **Skew coefficient:** Defines the angle between x and y pixels axes and is stored in $alpha_c$

  - **Radial Distortions:** Symmetric distortion caused by the lens due to imperfections in curvature when the lens was ground.

- **Extrinsic parameters**

  - **Rotational Vectors:** Vector describing the rotation of camera plane with respect to the world frame.

  - **Translational vectors:** Vector describing the translation of camera plane with respect to the world frame.

2.4 Possible problems or error sources that can disturb the calibration process

- If the naming conventions of the images are different, then the calibration process gives error.

- The computer running the calibration toolbox must have enough RAM(128 MB or less). If this condition is not met, then it gives *OUT OF MEMORY* error.

- Enough number of images must be provided to get better calibration results.

- The images captured with same orientations will not provide enough inputs for the toolbox to calibrate efficiently.

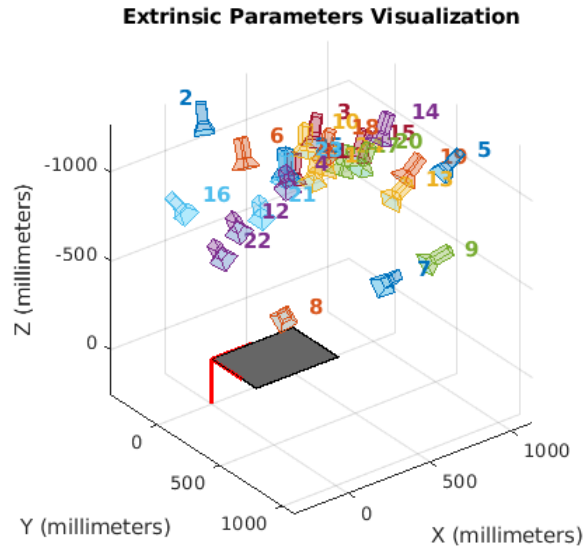## 2.5 Image poses used for calibration
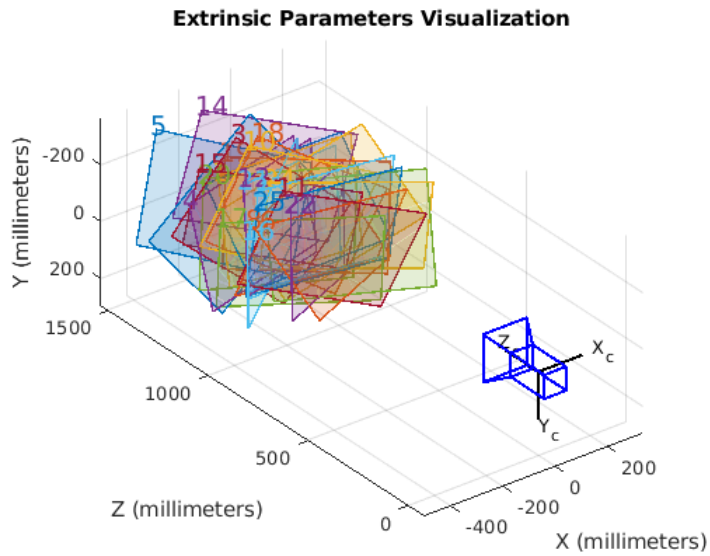

Figure 19: Pattern centric view

Figure 20: Camera centric view

The image poses can be approximated using the figures presented above which are obtained using the MATLAB camera calibration toolbox. How-

ever the exact values of each of these images poses is not possible to obtain due to a lack of a suitable measuring metric.

2.6   Camera parameters obtained from calibration

$$IntrinsicMatrix = \begin{bmatrix} 650.1601 & 0 & 0 \\ -0.4441 & 649.8506 & 0 \\ 417.1918 & 288.0535 & 1 \end{bmatrix}$$

$$FocalLength = \begin{bmatrix} 650.1601 & 649.8506 \end{bmatrix}$$

$$Principalpoint = \begin{bmatrix} 417.1918 & 288.0535 \end{bmatrix}$$

$$Skew = -0.4441$$

$$Radialdistortion = \begin{bmatrix} -0.0215 & 0.1424 \end{bmatrix}$$

$$TangentialDistortion = \begin{bmatrix} -9.82e - 04 & 0.0010 \end{bmatrix}$$

$$Meanprojectionerror = 0.1031$$

2.7   Estimation errors

2.7.1   Intrinsic errors

$$Skewerror = 0.0641$$

$$Focallengtherror = \begin{bmatrix} 0.3066 & 0.2922 \end{bmatrix}$$

$$Principalpointerror = \begin{bmatrix} 0.4705 & 0.3823 \end{bmatrix}$$

$$Radialdistortionerror = \begin{bmatrix} 0.0025 & 0.0167 \end{bmatrix}$$

$$Tangentialdistortionerror \begin{bmatrix} 2.1093e - 04 & 2.5468e - 04 \end{bmatrix}$$

2.7.2   Extrinsic errors

They are available in the *estimationerrors.mat* file

2.8   Arguments for which camera model best fits

If the lens distortions are really too severe (for fish-eye lenses for example), the simple guiding tool based on a single distortion coefficient kc may not be sufficient to provide good enough initial guesses for the corner locations. then corner extraction has to be done manually.

3 Measurement of the accuracy and precision of a KUKA youBot arm

3.1 Description of the experiment

- The experiment involves a **KUKA youBot arm 1** performing different pick and place tasks.

- Three objects with different masses are to be placed in three different poses.

- The object poses (x, y, θ) are determined by tracking ArUco markers that are placed on top of the objects by an external vision system (Microsoft LifeCam) placed above the arm's workspace.

- The experiment is performed 20 times for each object-pose combination resulting in a total of 180 experimental trials.
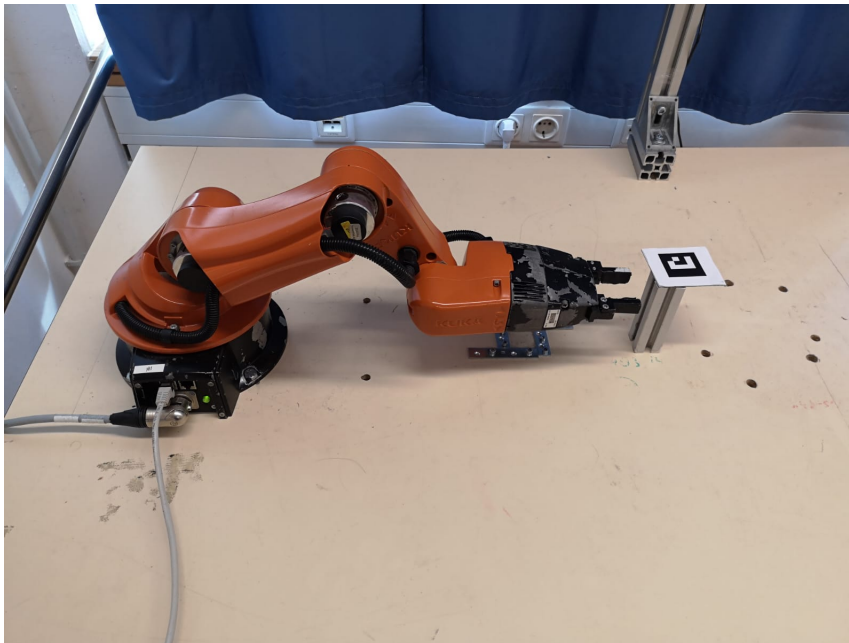
Figure 21: KUKA youBot arm after placing the object
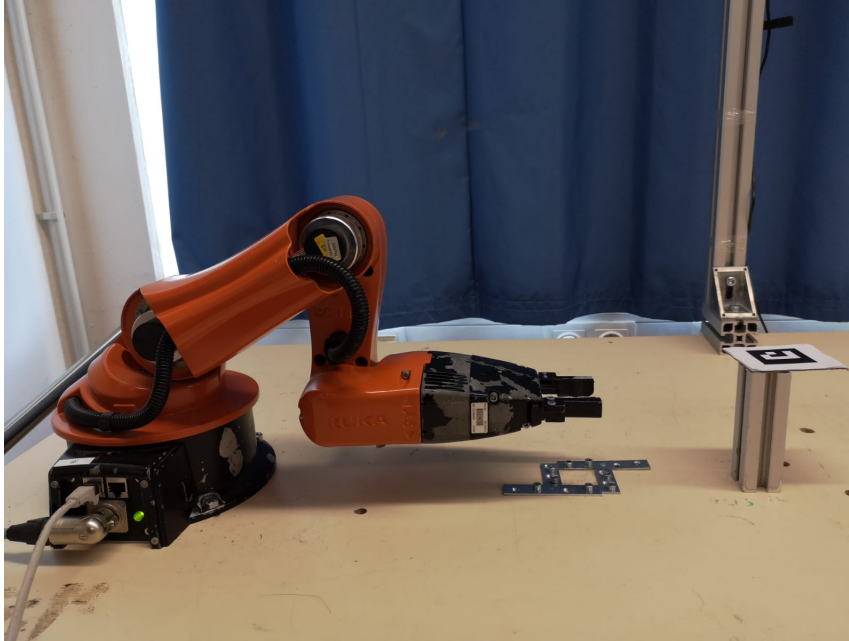
Figure 22: KUKA youBot arm at the initial position

3.2 Observations

3.2.1 Sources of error

- Changes in the position of the small/ medium object while being placed on the table.

- Changes in the position of the object while the arm retracts after placing the object.

- Small shakes in the table can cause changes in the pose readings.

3.2.2 Pose filtering procedure

- For every experiment the final pose is obtained by taking an average of 50 readings.

- For the obtained pose values in each experiment we have fitted a Gaussian distribution.

- It can be seen that there are some outliers in the data.

- We have removed these outliers by using Chebyshev algorithm and the final data is plotted.

- On average, 3 outliers were detected for the entire experiment.

3.3 Scattered plots

1. **Experiment: Small Marker-** The below plot shows the combined plot (X and Y coordinates) of pick and place of small object in three different places(straight, left and right).

   The mean of the orientation values for each experiments (straight, left and right) is stored. The triangle in the plot is oriented by the mean orientation values.
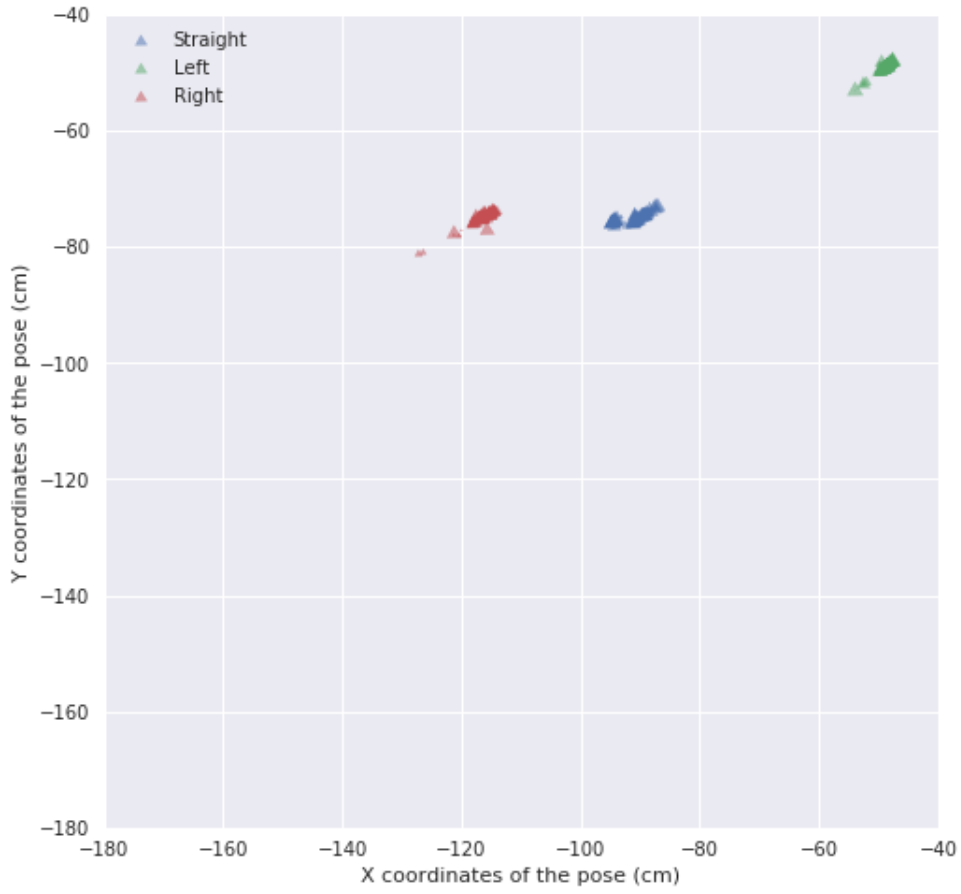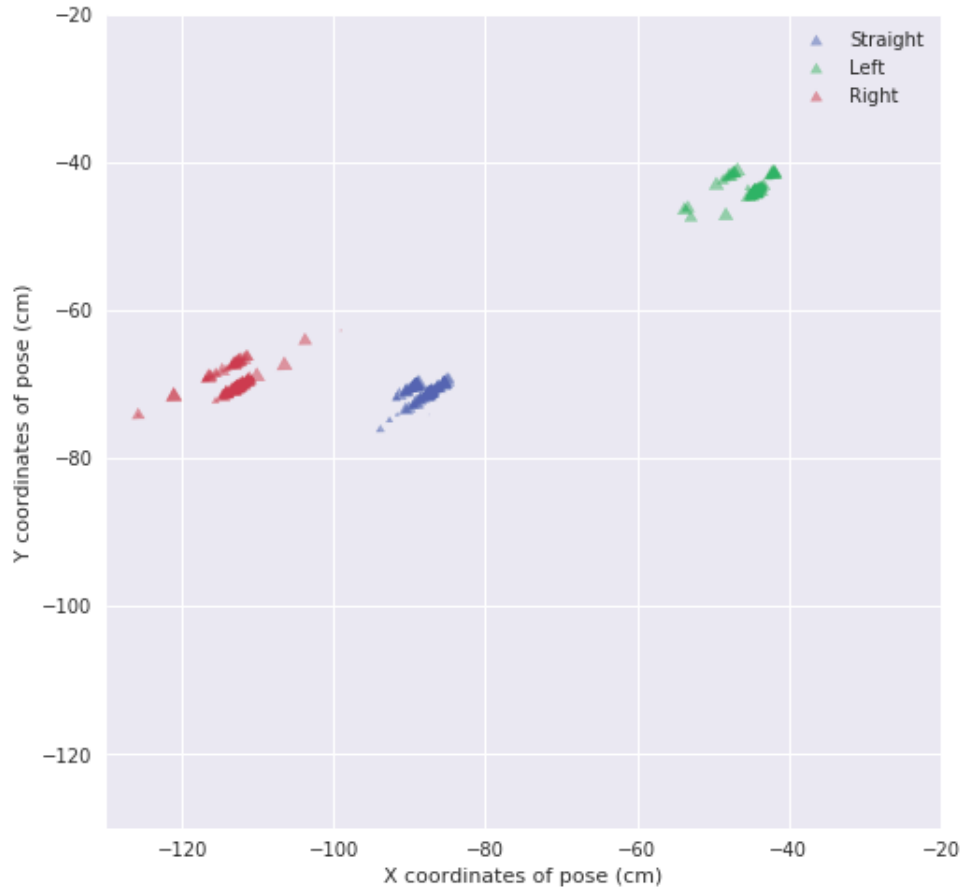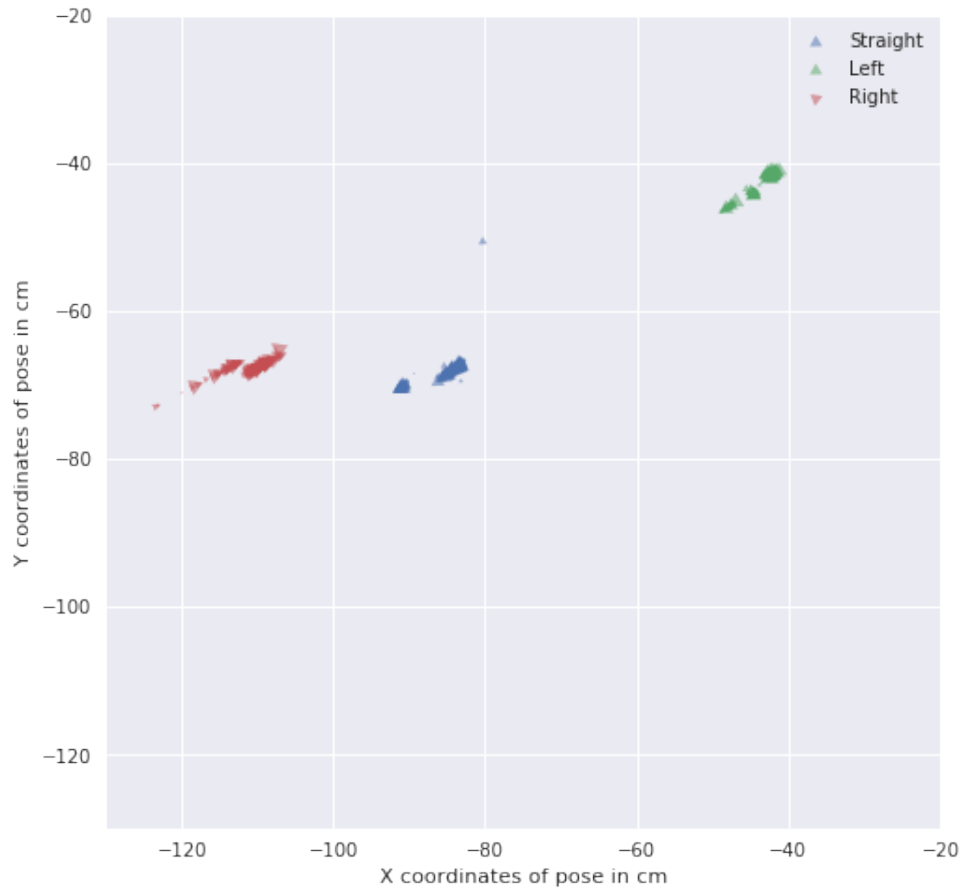

Figure 23: Scattered plot of small marker experiment

2. **Experiment: Medium Marker-**The below plot shows the combined plot (X and Y coordinates) of pick and place of medium mass object in three different places(straight, left and right).

   The mean of the orientation values for each experiments (straight, left and right) is stored. The triangle in the plot is oriented by the mean orientation values.

Figure 24: Scattered plot of medium marker experiment

3. **Experiment: Large Marker-**The below plot shows the combined plot (X and Y coordinates) of pick and place of large mass object in three different places(straight, left and right).

The mean of the orientation values for each experiments (straight, left and right) is stored. The triangle in the plot is oriented by the mean orientation values.

Figure 25: Scattered plot of large marker experiment

3.3.1 Distribution plots

For the obtained experimental results, a Gaussian distribution is plotted. It is observed that there are many outliers that do not belong to the distribution.

Therefore, we have used the **Chebyshev's outliers algorithm** to remove the outliers.

The algorithm requires to set two likelihoods (pp1 and pp2).

pp1 – likelihood of expected outliers (e.g. 0.1, 0.05 , 0.01)

pp2 – final likelihood of real outliers (e.g. 0.01, 0.001 , 0.0001)

The pp1 and pp2 are equal to (1/sigma).

Sometimes, setting pp1 and pp2 as (1/sigma) do not completely remove the outliers. Therefore, we have increased the pp1 and pp2 according to the observed outliers.

1. Experiment: Pick and place of Small marker - Straight
   X coordinate: pp1 : 0.22, pp2 : 0.22
   Y coordinate: pp1 : 0.27, pp2 : 0.22
   Orientation: pp1 : 0.25, pp2 : 0.25
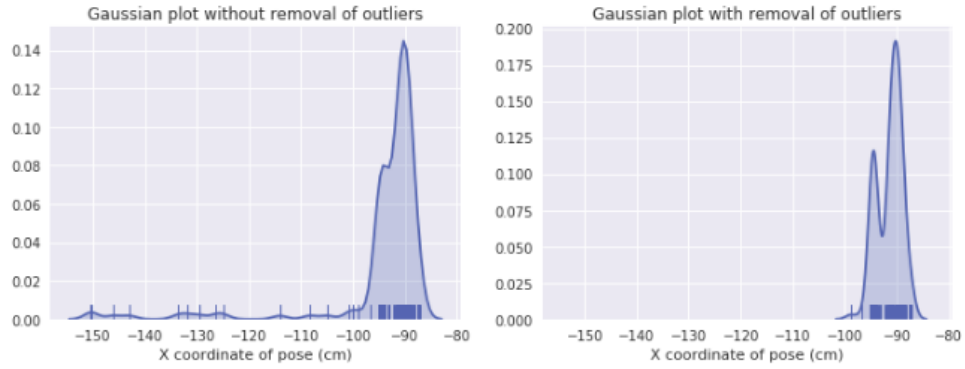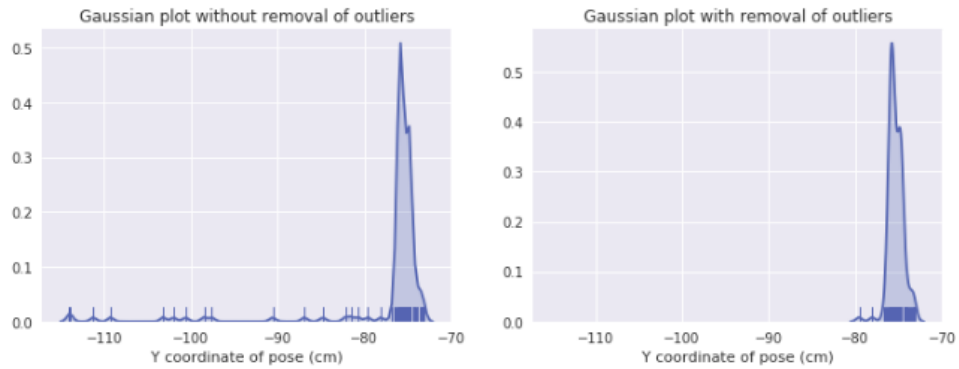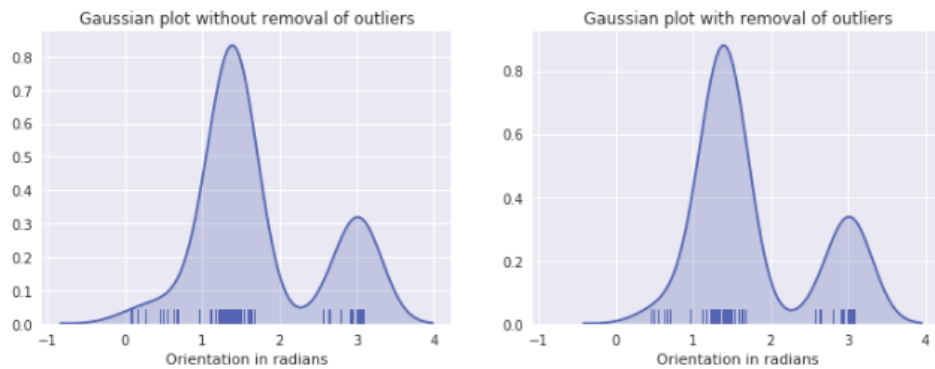
Figure 26: Distribution plot for Small marker-Straight

2. Experiment: Pick and place of Small marker - Left
   X coordinate: pp1 : 0.1, pp2 : 0.1
   Y coordinate: pp1 : 0.12, pp2 : 0.12
   Orientation: pp1 : 0.27, pp2 : 0.27

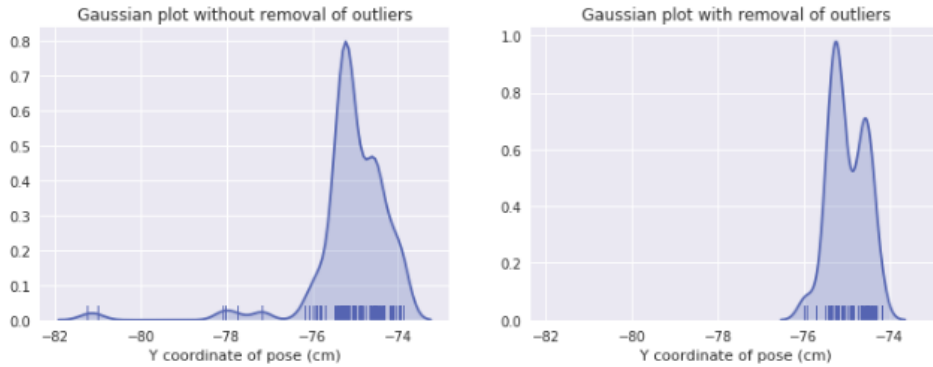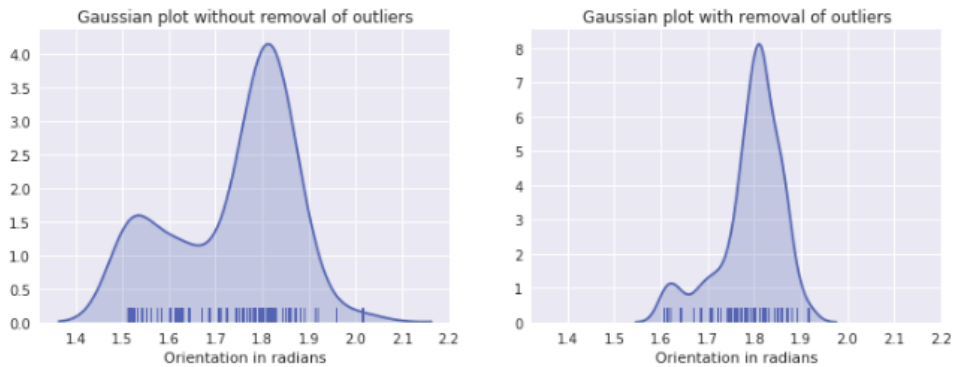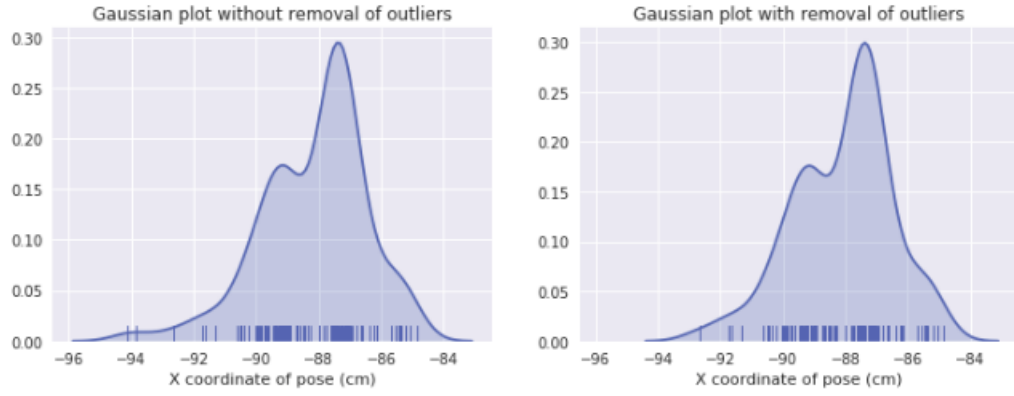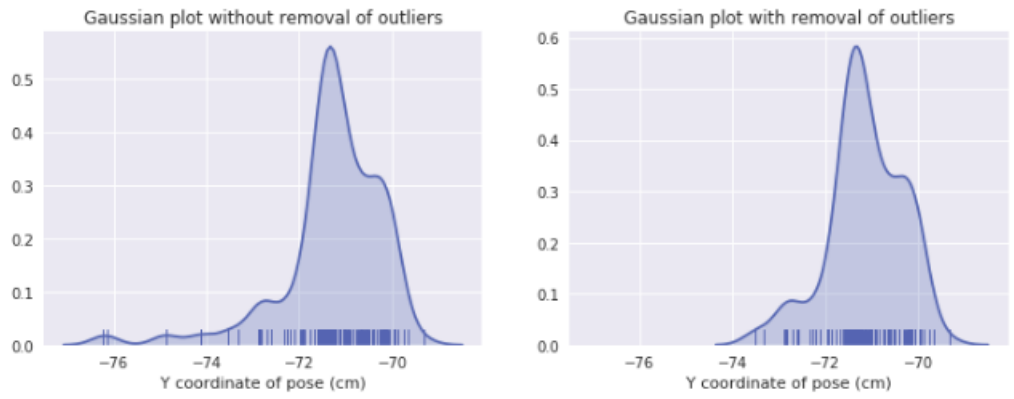Removed outliers: 6
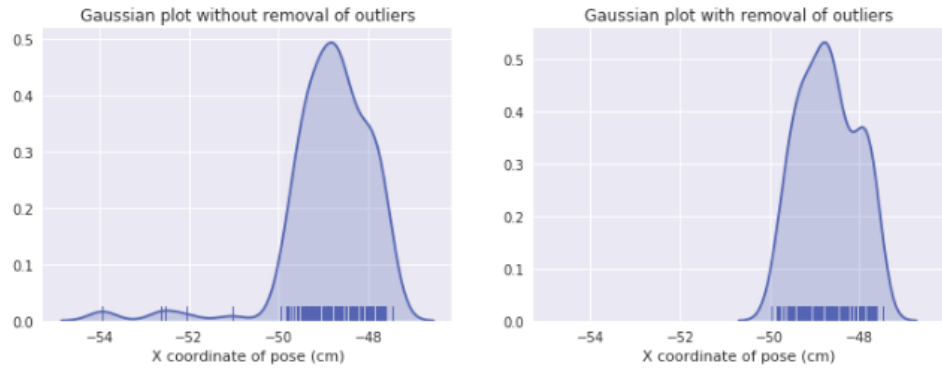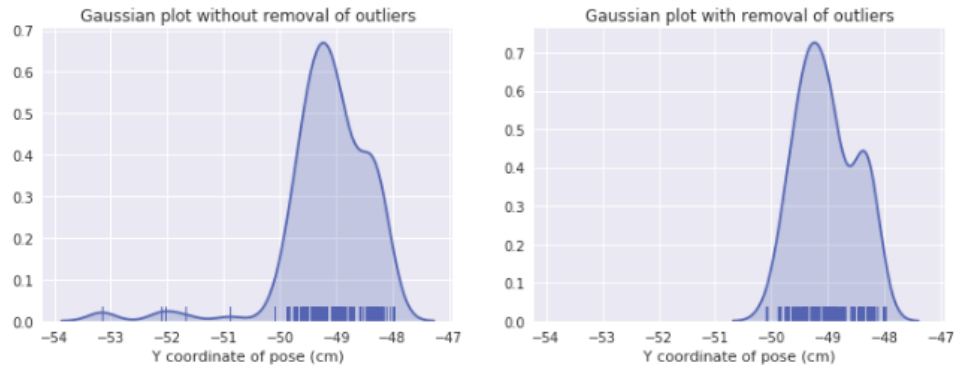
Removed outliers: 6
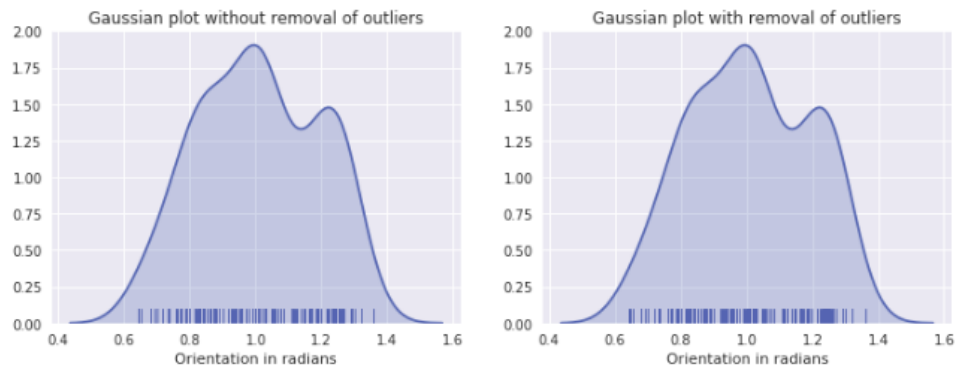
Removed outliers: 0

Figure 27: Distribution plot for Small marker-Left

3. Experiment: Pick and place of Small marker - Right
   X coordinate: pp1 : 0.61, pp2 : 0.61
   Y coordinate: pp1 : 1.0, pp2 : 1.0
   Orientation: pp1 : 0.39, pp2 : 0.39
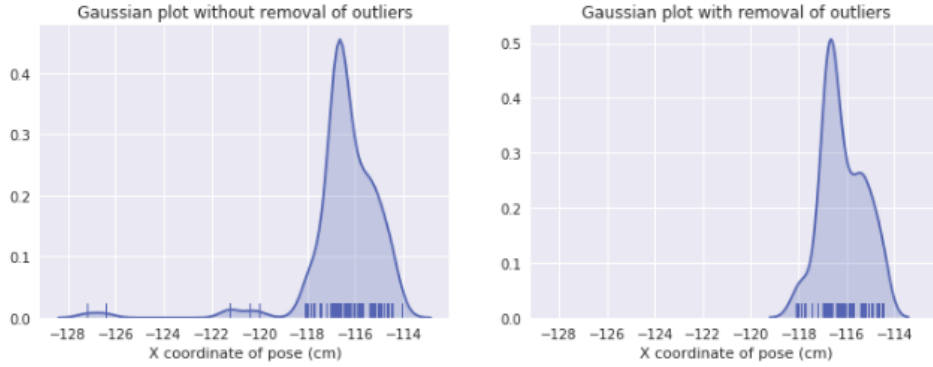
Figure 28: Distribution plot for Small marker-Right
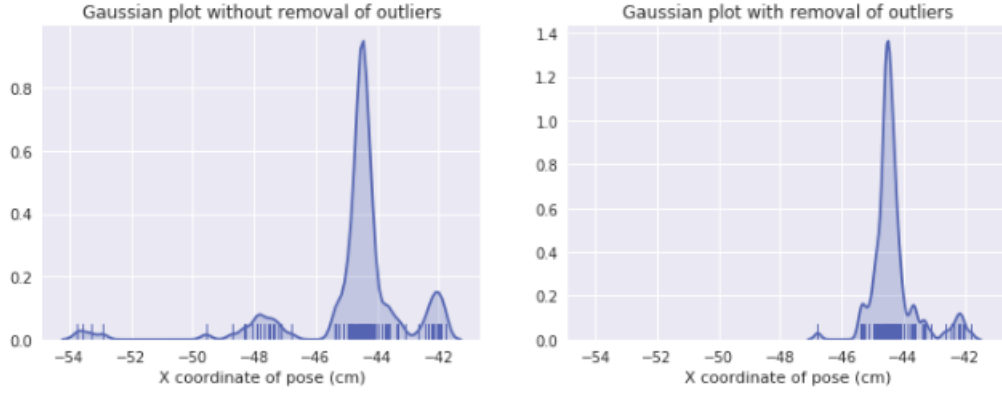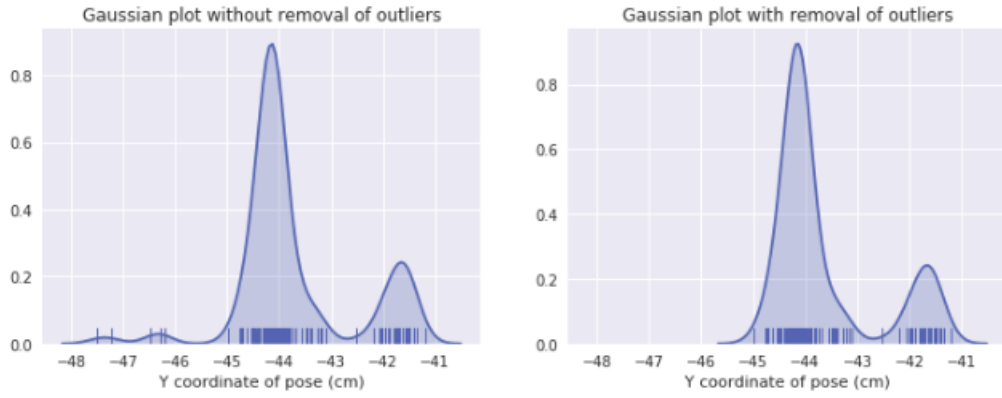
4. Experiment: Pick and place of Medium marker - Straight

Figure 29: Distribution plot for medium marker-Straight

5. Experiment: Pick and place of Medium marker - Left

Figure 30: Distribution plot for medium marker-Left

6. Experiment: Pick and place of Medium marker - Right

Figure 31: Distribution plot for medium marker-Right

7. Experiment: Pick and place of large marker - Straight

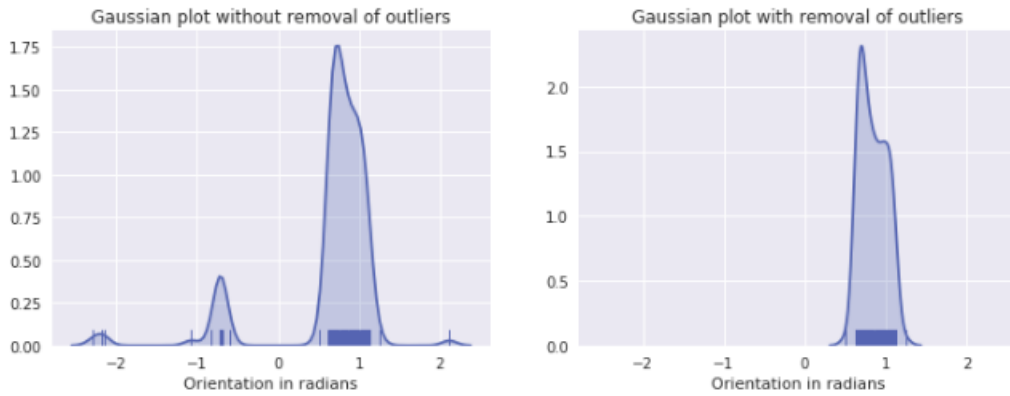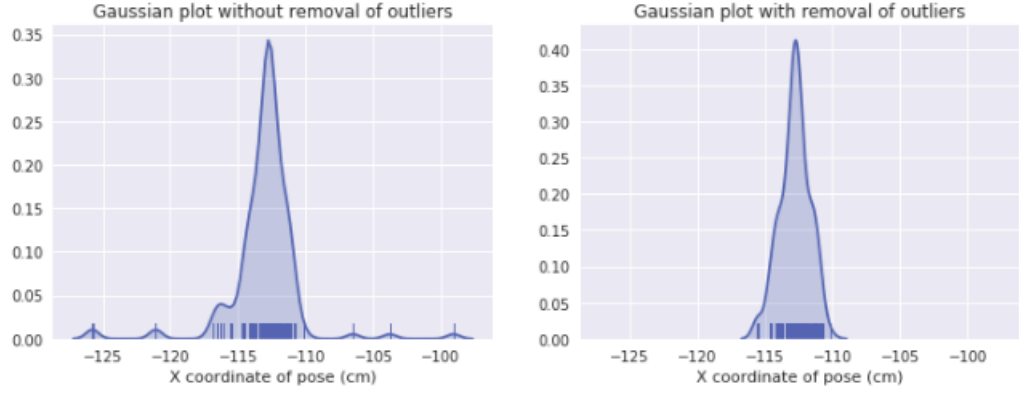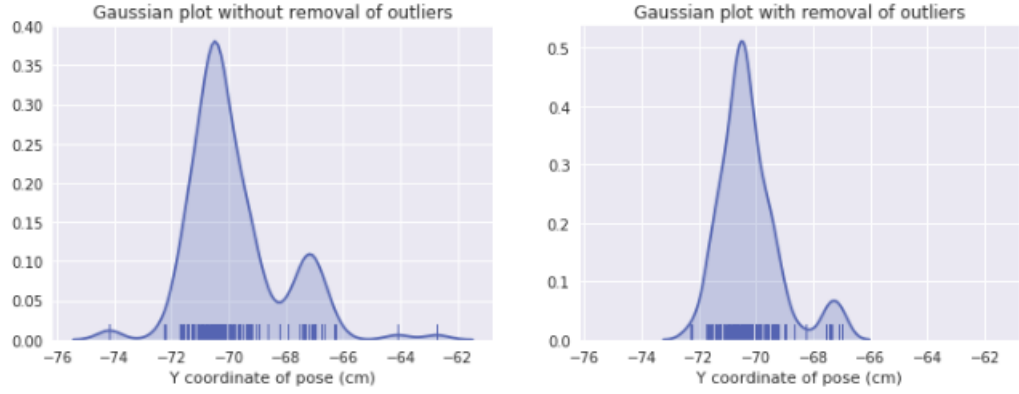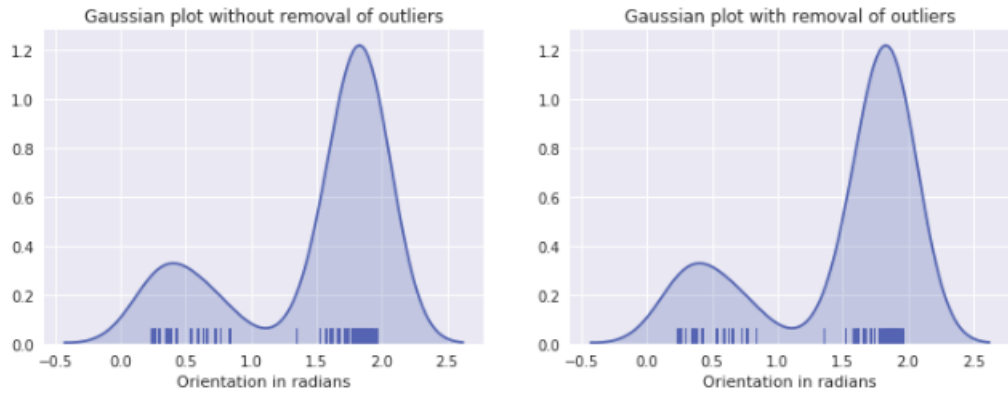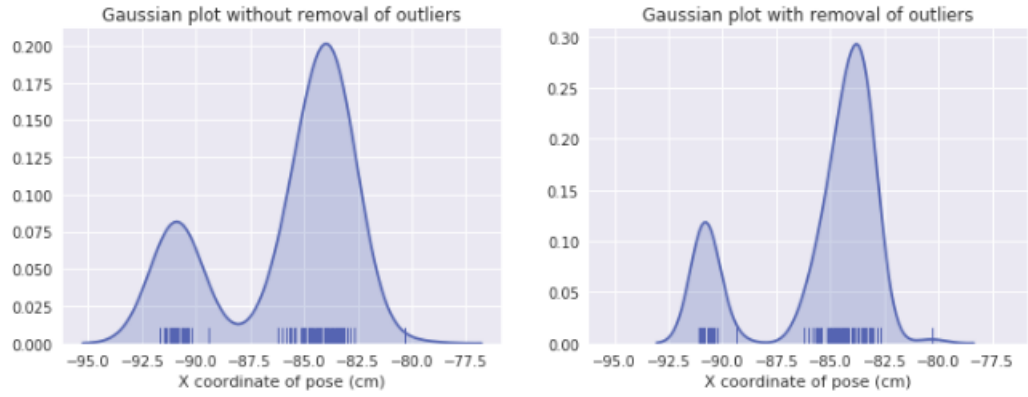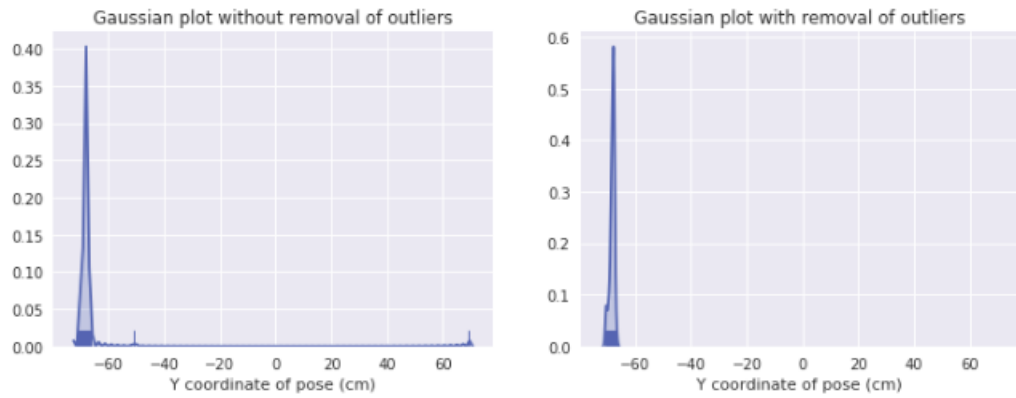Removed outliers: 9

Removed outliers: 36

Removed outliers: 35

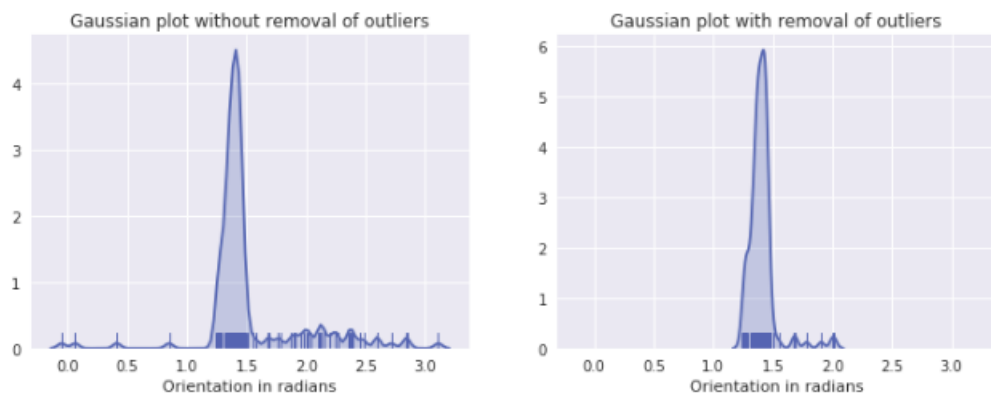Figure 32: Distribution plot for large marker-Straight

8. Experiment: Pick and place of large marker - Left

Removed outliers: 22
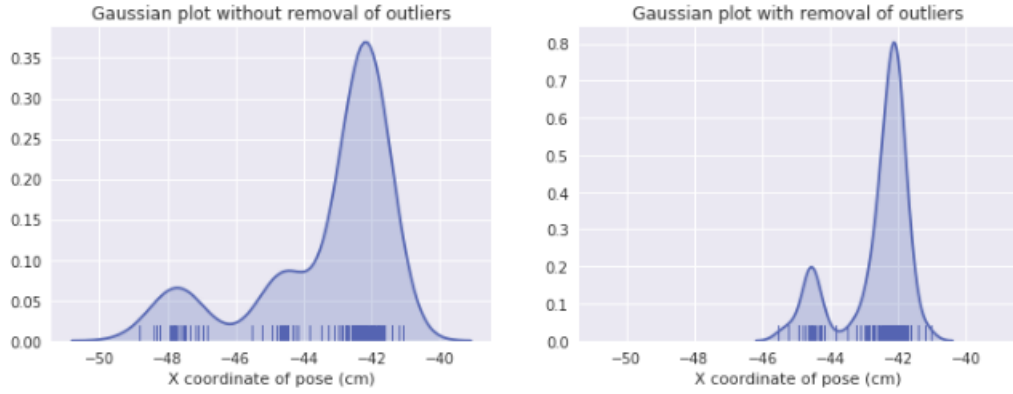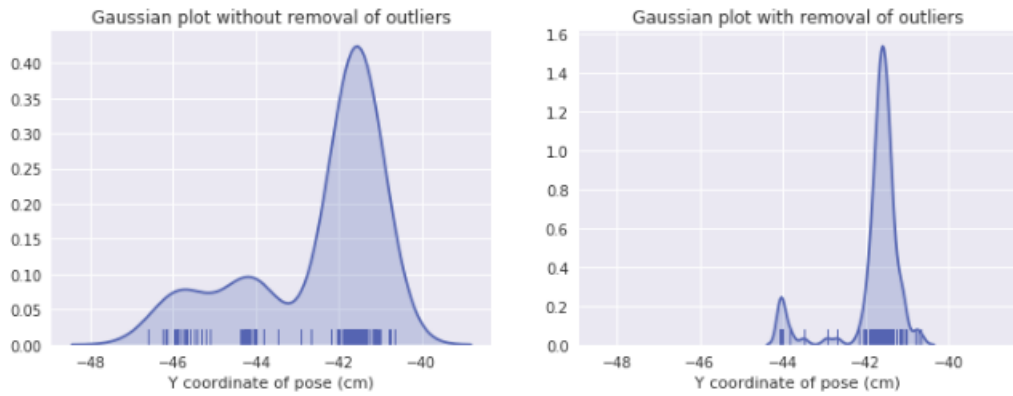

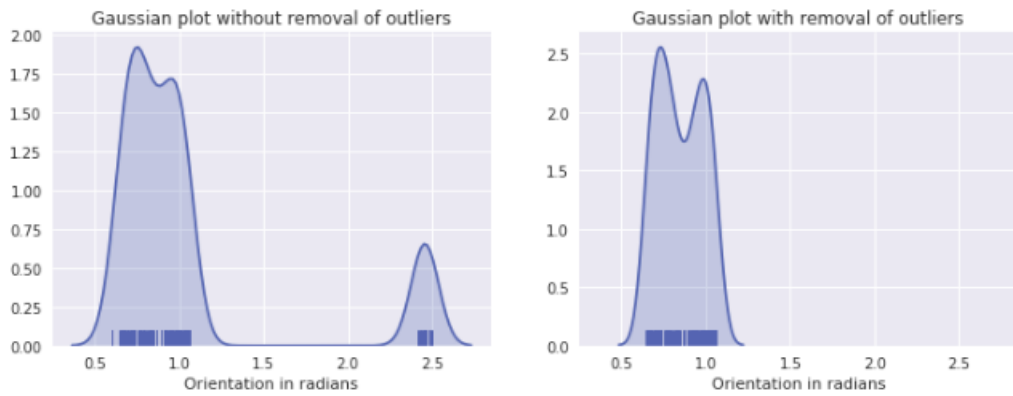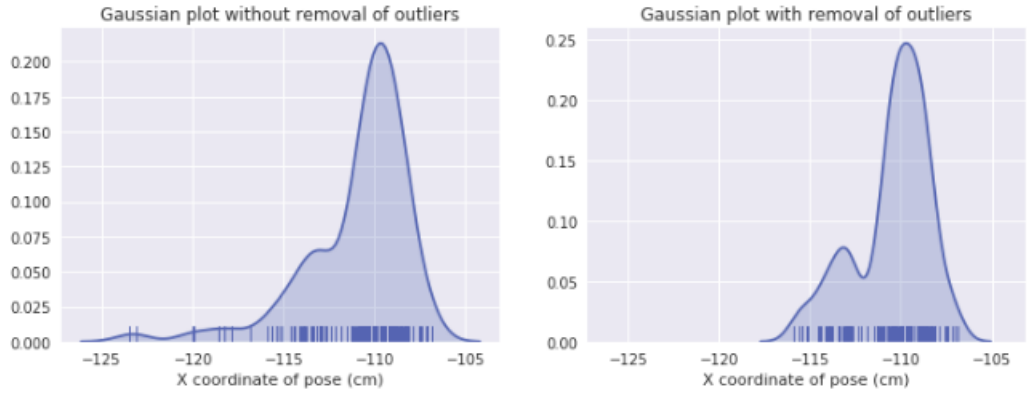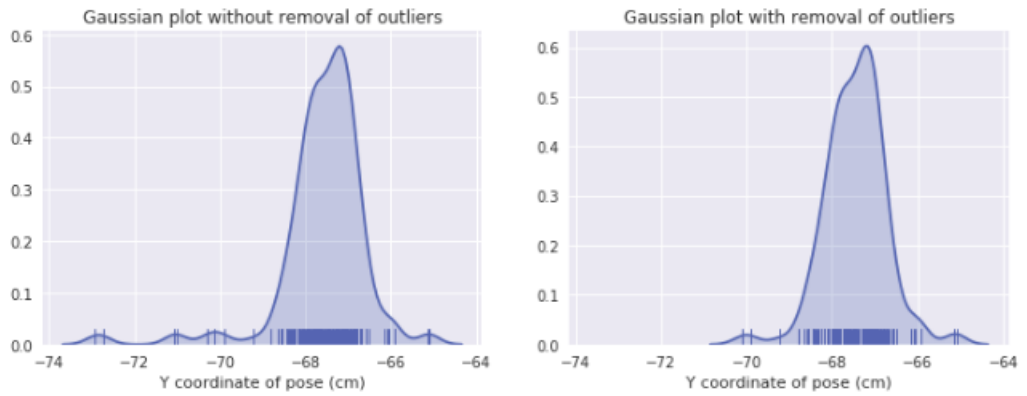**Removed outliers: 35**

Removed outliers: 24

Figure 33: Distribution plot for large marker-Left

9. Experiment: Pick and place of large marker - Right

Removed outliers: 8

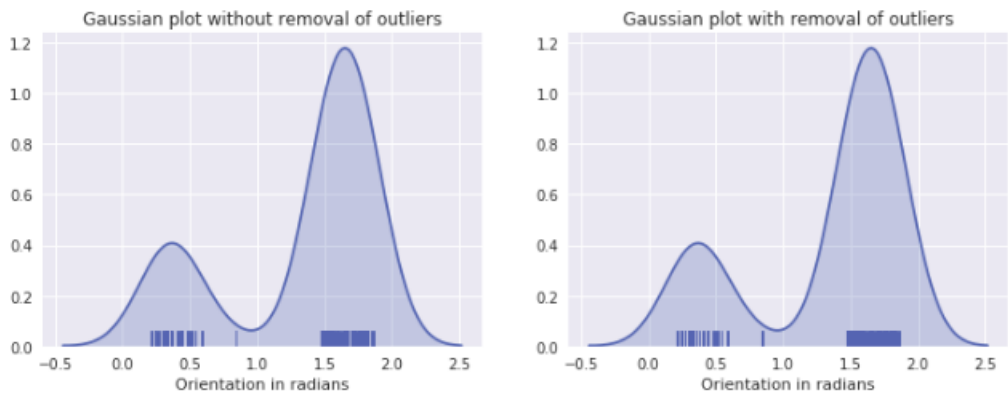Removed outliers: 9

Removed outliers: 0

Figure 34: Distribution plot for large marker-Right

3.3.2 Analyzing the experimental results

The obtained experimental data is analyzed by calculating accuracy and precision of the used youbot arm.
Ground truth: Youbot Arm 1
The ground-truth initial and final poses for each of the three motions with respect to the camera frame are given below.

Pose	$x(cm)$	$y(cm)$	$\theta(rad)$
Pick	-85.41	-55.60	1.38
Straight	-86.89	-71.31	1.45
Left	-45.47	-45.00	0.88
Right	-112.13	-70.58	1.78

Figure 35: Distribution plot for large marker-Right

The Accuracy is calculated as follows,

$$Accuracy = (desired - actual)/desired$$

Where,
Desired = ground truth values
Actual = mean of the experiment values

Table 1: Accuracy

Small marker	X(cm)	Y(cm)	θ(radians)
Straight	-0.05	-0.05	-0.22
Left	-0.07	-0.08	-0.13
Right	-0.03	-0.06	0.024
Medium marker			
Straight	-0.01	0.00	0.16
Left	0.01	0.02	0.38
Right	0.00	0.01	0.20
Large marker			
Straight	0.01	0.05	-0.05
Left	0.05	0.05	-0.17
Right	0.01	0.04	0.33

Precision of the experimental values The following table displays the precision values for different objects.

Table 2: Precision(Mean ± variance)

Small marker	X(cm)	Y(cm)	θ(radians)
Straight	-91.28 ± 2.22	-75.28 ± 0.73	1.86 ± 0.73
Left	-48.84 ± 0.99	-49.14 ± 0.78	1.00 ±0.17
Right	-116.44±1.61	-75.09 ± 0.96	1.73 ±0.12
Medium marker			
Straight	-88.15±1.68	-71.27±1.09	1.24±1.12
Left	-44.76±1.95	-43.70±1.14	0.63±0.64
Right	-113.04±2.54	-69.81±1.64	1.47±0.60
Large marker			
Straight	-85.81±3.07	-67.58± 10.84	1.53±0.42
Left	-43.30±1.98	-42.50±1.60	1.06±0.56
Right	-110.90±2.84	-67.58±1.00	1.33±0.56

Observations:

- From the above table it is observed that the precision has decreased greatly with the increase in the mass of the object.

- The precision for small marker (Y-coordinate) is ± 0.73 cm, for the medium mass object it is ± 1.09 cm and for the large mass object it has increased to ± 10.84 cm.

- The possible cause for this significant decrease in precision is; the arm grasps the object at certain height (say h) which is same for all the object sizes. Hence the object with greater length wobbles during the placement.

4 Appendix

4.1 Softwares used

- Python 2.7

- Jupyter Notebook

4.2 Packages used

- Matplotlib

- Numpy

- Scipy

- Seaborn

- Pylab

4.3 Tools used

- Onscreen ruler is used to mark scale on the sheet (metrics in inches)
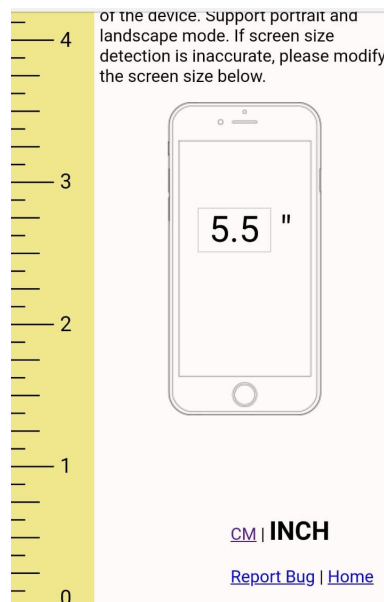
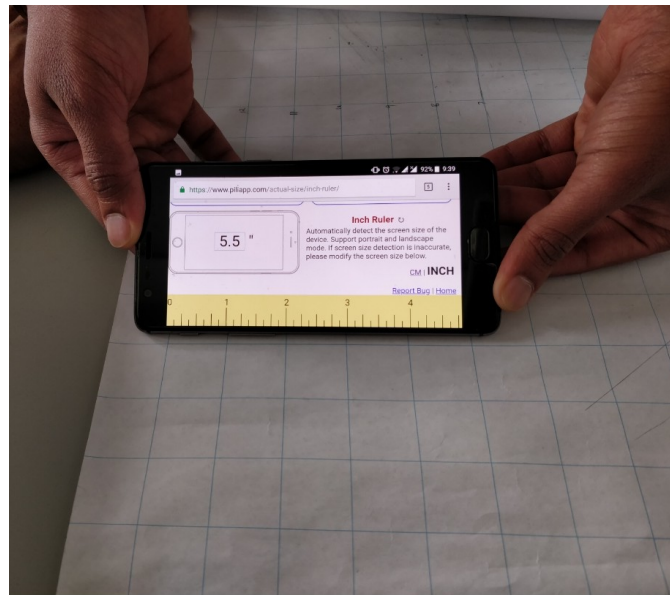- https://www.piliapp.com/actual-size/inch-ruler/

Figure 36: Online ruler

44

Figure 37: Marking scale on the sheet using onscreen ruler