DELIVERABLE D4.1

MEASUREMENTS/SIMULATIONS DOCUMENTATION

JULY 15, 2023

MARIOS RASPOPOULOS, STELIOS IOANNOU, ANDREY SESYUK

INTERDISCIPLINARY SCIENCE PROMOTION & INNOVATIVE RESEARCH EXPLORATION (INSPIRE)

"This work was co-funded by the European Union under the programme of social cohesion "THALIA 2021-2027", through Research and Innovation Foundation (Project: CONCEPT/0722/0031)".

Abstract

This report Includes the details about the data produced in T4.1 either through simulations or measurements, the description about the data structure as well as a description about the use of the software functions that access the data structure.

Table of Contents

1 Precision Measurements/Analysis

This chapter describes the measurement methodology to evaluate the precision of typical mmWave radar sensors. The current market availability of mmWave radar sensors has steered this investigation in mainly two directions: one using 2-DoF (Degrees of Freedom) sensors that support ranging and azimuth measurements and one using 3-DoF sensors that additionally measure the elevation of targets. For each of these cases we performed a precision analysis of the most predominantly-used mmWave ranging sensors currently in the market and thereafter used the ranging/angular information to conduct positioning using various methods. For the 2DOF case we consider sensors that have the ability to measure range and azimuth (the Texas Instruments IWR1642Boost and the Infineon Distance2Go) while for the 3DOF that additionally measures elevation we used the Texas Instruments IWR1843.

The importance of these measurements is first of all to evaluate the measurement precision under different orientations and ranges, to evaluate their range and field of view and their potential to perform signle achnor positioning (for the IWR1843 sensor). This precision analysis will provide valuable input for the generation of the the 3D positioning algorithms (see Deliverable 3.2) either for the quantification of the reliability of a measurement or the removal of possible outliers from measurements.

1.1 Equipment Used

1.1.1 2-DOF Sensors

The two mmWwave radar sensors that were used for the 2-DOF precision analysis were the Texas Instruments (TI) IWR1642BOOST and Infineon Distance2Go. The TI sensor is equipped with 4 receiving (Rx) and 2 transmitting (Tx) antennas operating at frequencies between 76-81*GHz* with a 120-degree field of view and ranging capabilities of up to 72 meters. In contrast, the Infineon Distance2Go mmWave sensor is equipped with 1 Rx and 1 Tx antenna and operates between 24-26*GHz* with a field of view of 20 degrees and a maximum detection range of around 20 meters. While the TI sensor performs range and angle measurements, the Infineon one can only measure range. The experimental setup involved utilizing a DJI Air 2S drone as the target for ranging and angular measurements. It is a compact drone with dimensions of $183.0\times77.0\times253.0mm$.

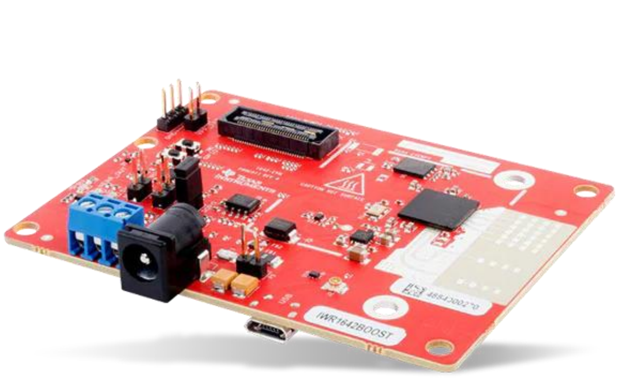


Figure 1: 2-DOF mmWave Sensors: TI's IWR1642BOOST (left) and Infenion's Distance2Go (right)

1.1.1 3-DOF Sensors

Similarly to IWR1642BOOST, the IWR1843BOOST possesses a Frequency Modulated Continuous Wave (FMCW) transceiver which enables the measurement of range, azimuth angle and velocity of the target. However, due to an additional TX antenna, in addition to the azimuth angling information, it is also able to provide the elevation data of the target. A similar system setup was used for this sensor like the one used for the IWR1642BOOST in which each sensor is connected to a Raspberry PI, through which the collected context is parsed and then sent to a central PC through a UDP connection.

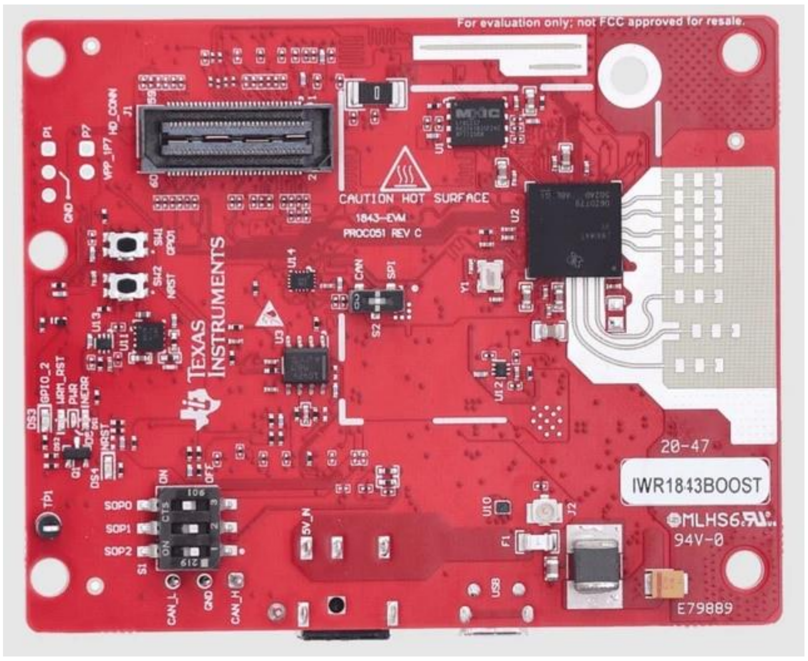


Figure 2: IWR1843BOOST Sensor

1.2 Experimental Setup

1.2.1 2DOF Experimental Setup

Both the precision analysis and the 3D positioning accuracy experimentation (see next chapter) using 2-DOF sensors were carried out in an 8.85×6.85*m* engineering laboratory the top-view of which is shown in Figure 1

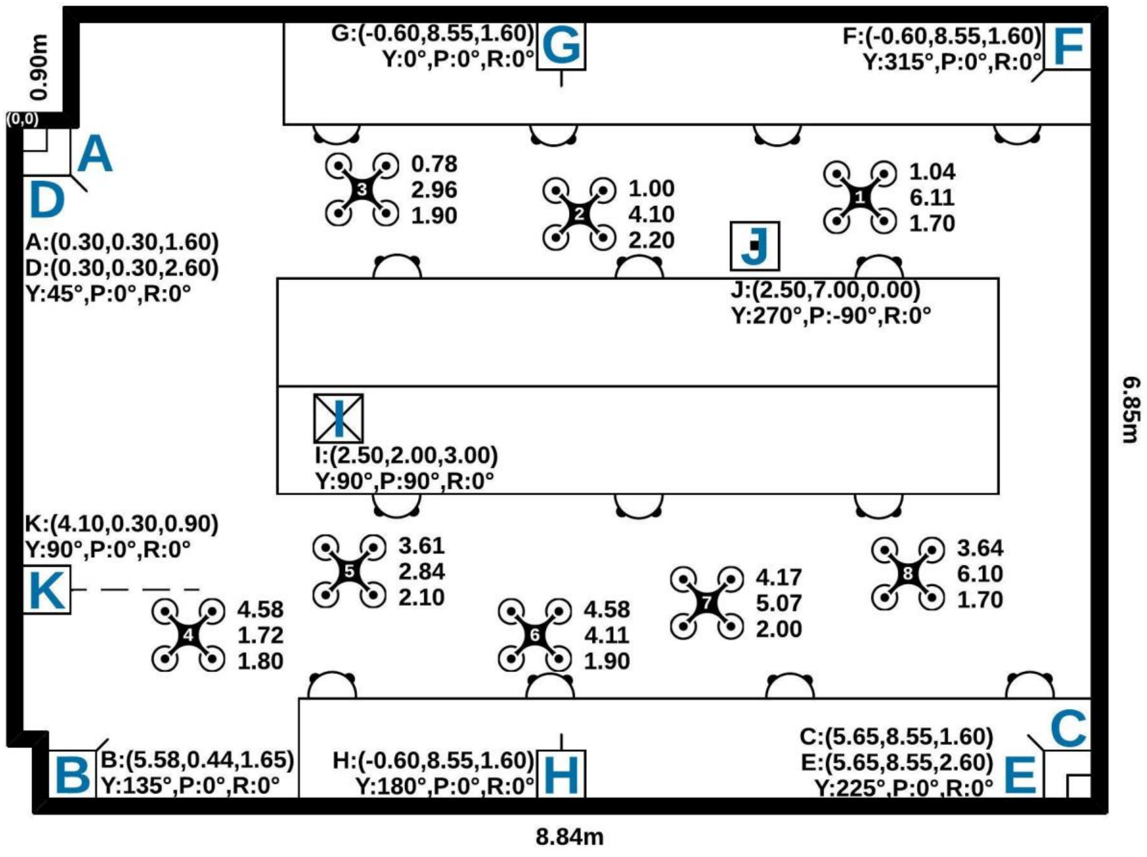


Figure 3: mmWave 3D Positioning Experimental Setup using 2-DOF mmWave Sensors (Y: Yaw, P: Pitch, R: Roll)

The precision analysis was conducted to compare the ranging and angular capabilities of the two mmWave sensors. The sensor under test was placed in location K as shown in Figure 5 and range measurements were collected every 0.5*m* while the drone was flying in a straight line in front of the sensor (0.5 to 8*m*). To assess the ability of the sensors to conduct range measurements at different angles, the orientation of the sensor was systematically varied from 0 to 60 degrees (15-degree step). This comprehensive analysis aimed to gather precise data on the sensors' precision,

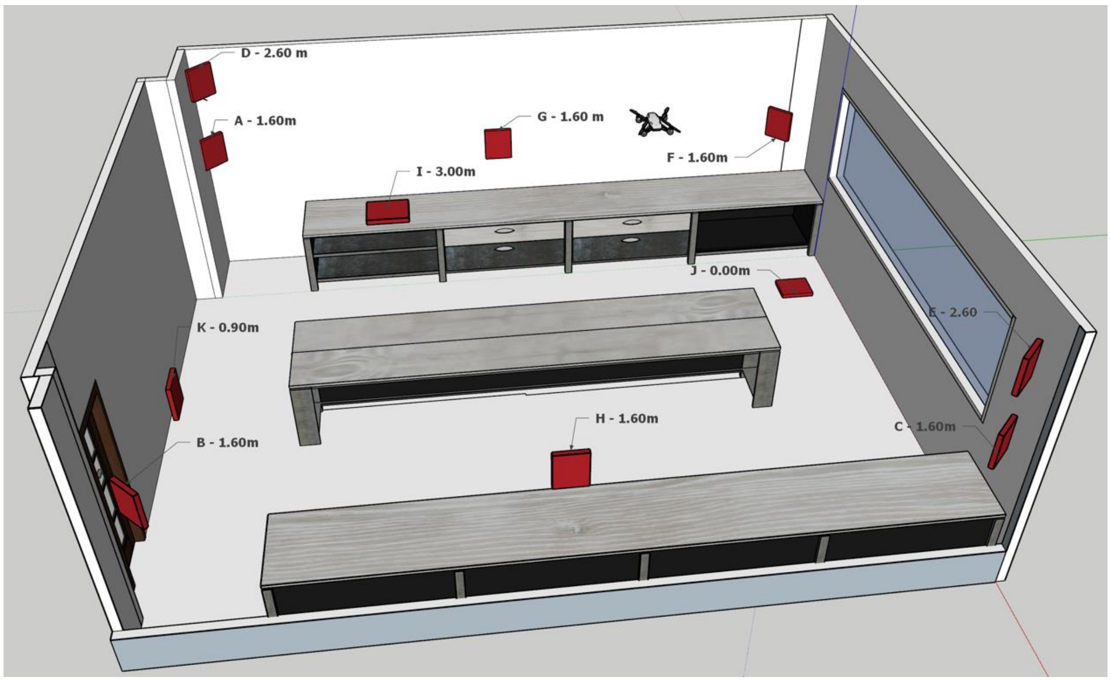


Figure 4 - mmWave 3D Positioning Experimental Setup using 2-DOF mmWave Sensors (3D View)

resolution, and reliability at different distances and angles. Also, the precision of the TI sensor in measuring the angle of departure was evaluated using the same setup.

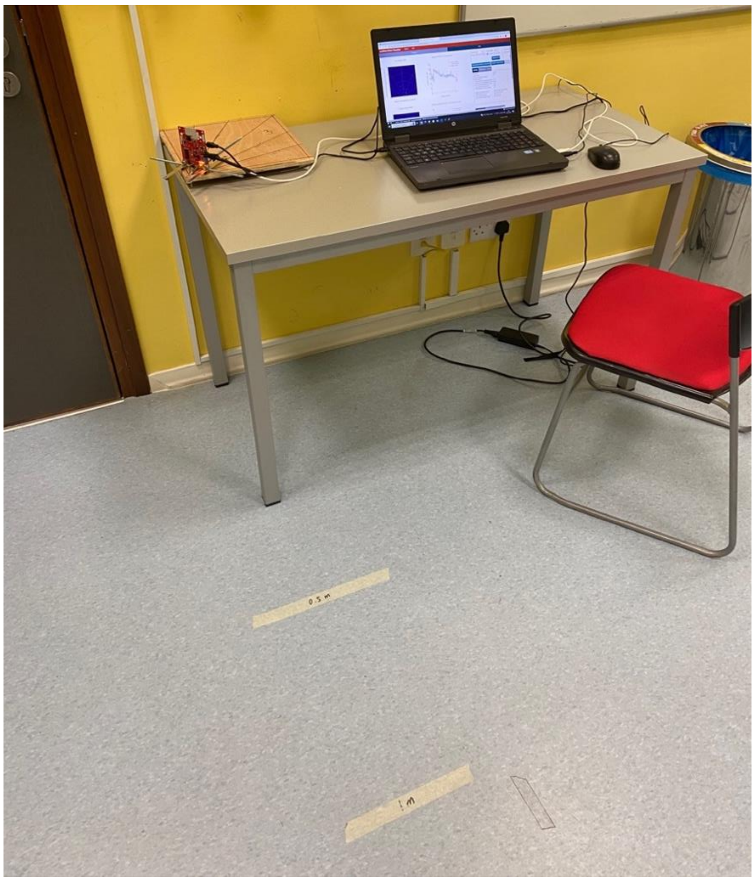


Figure 5: 2DoF Presision Analysis - Sensor Placement

1.2.2 *3-DOF Experimental Setup*

For the evaluation of the precision of the IWR1843BOOST the mmWave sensor was mounted on a versatile tripod that allowed both vertical and horizontal adjustments, provided an ideal platform for manipulating the sensor's orientation. This experiment sought to evaluate the precision of the sensor in scenarios mimicking the dynamic movement of a flying drone. To emulate the drone's movement, a basketball of a similar size of the drone was suspended on a string and set into a spinning motion as shown in Figure 6. This dynamic motion was crucial as mmWave sensors, due to the Doppler shift requirement, struggle to detect stationary targets effectively. By employing the spinning basketball, the movement patterns of a drone in flight could be emulated, providing a real-world scenario for the precision analysis. This experimental setup allowed to systematically vary both the azimuth and elevation angles of the sensor using the adjustable tripod. The azimuth angle, representing the horizontal orientation, and the elevationangle, representing the vertical orientation, were adjusted to different degrees to test the sensor's precision under diverse angling conditions while the tripod used also moved at different distances away from the target (0 to 6.5m). This comprehensive approach aimed to uncover any potential limitations or strengths of the mmWave sensor in different spatial configurations.

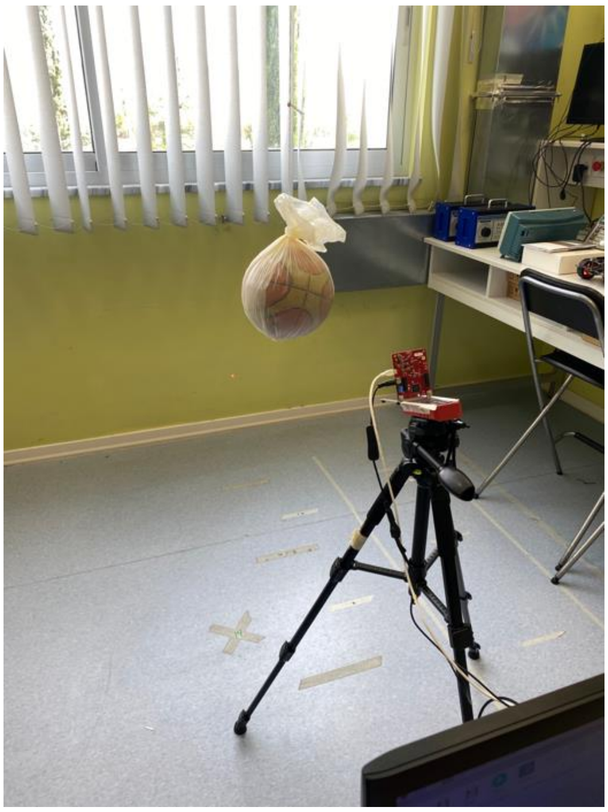


Figure 6: mmWave 3-DOF Precision Analysis Setup using 3-DOF mmWave Sensor

1.3 2-DOF Sensor Precision Analysis

To evaluate the accuracy and sensing quality of the IWR1642BOOST and the Infineon sensors a range/angle precision analysis experimentation was carried out using the setup described in 1.2.1. A drone was flown along a straight line, while a mmWave sensor was placed at different orientations at location K as shown in Figure 3. Given that the Infineon sensor has a relatively narrow field of view (around 20 degrees), the analysis of distance accuracy in comparison to the TI sensor was conducted up to 15 degrees. The results of this comparison are shown in Figure 7 and a notable observation is the difference in distance errors between the two sensors. Both at 0 and 15 degrees, the TI sensor outperforms the Infineon sensor. Specifically, the TI sensor demonstrates an average distance error of around $0.32m$, whereas the Infineon sensor exhibits a higher error of $0.46m$. While the error remains relatively consistent as the distance increases for both sensors, the analysis indicates a decrease in accuracy with larger angles. At 15 degrees, there is a slight increase in error, approximately $0.05m$, compared to the error at 0 degrees.

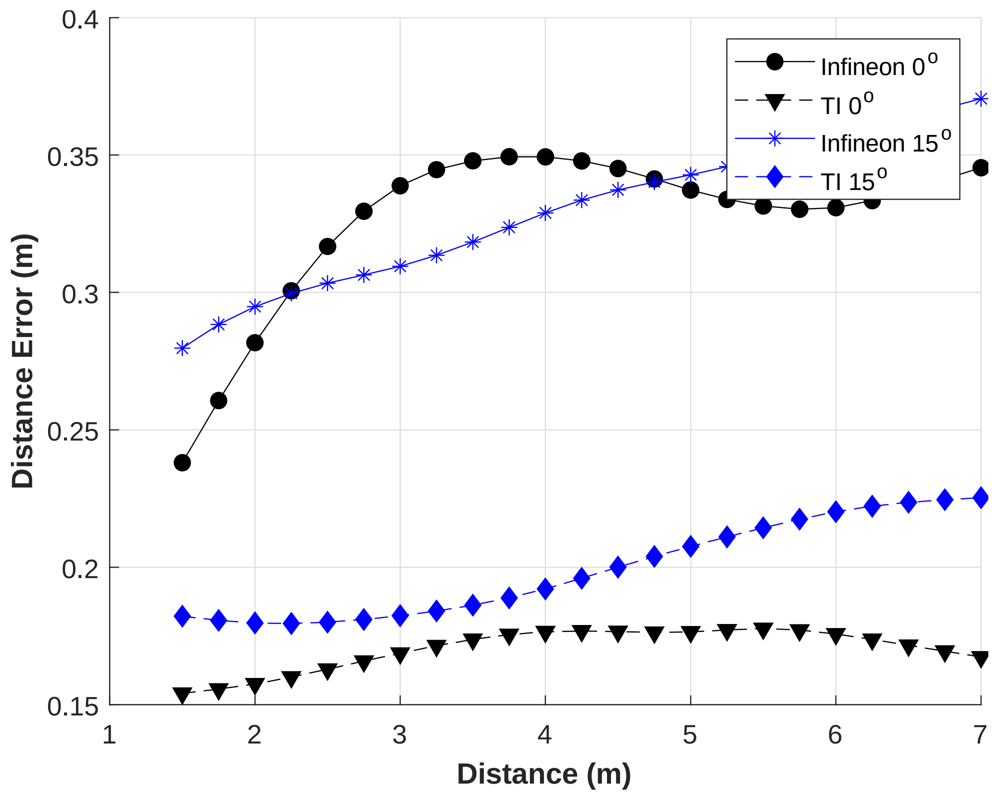


Figure 7: Infenion vs TI Distance Measurements

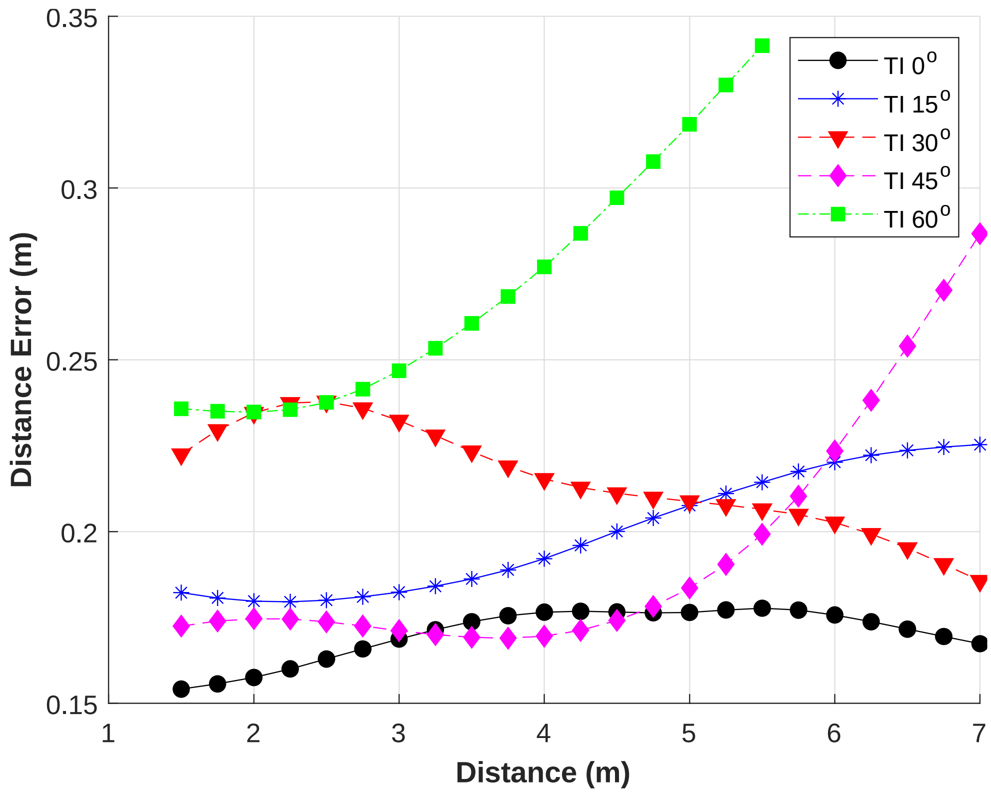


Figure 8: IWR1642BOOST Distance Accuracy

Following the comparison between the two sensors, the distance and azimuth angle accuracy of the TI sensor were further tested beyond 15 degrees. Figure 8 specifically illustrates the distance error of the TI mmWave sensor across angles ranging from 0 to 60 degrees. Upon closer inspection, it becomes evident that while the error remains consistent for each analyzed angle, there is a noticeable and constant increase in error. At 0 degrees, the average distance error stands at 0.32*m*, gradually rising to approximately 0.43*m* at 60 degrees. It is worth noting that considering the wide field of view spanning 60 degrees, an error of 0.43*m* may not appear excessively large. However, a limitation is encountered as the sensor ceases to detect objects beyond a range of 6 meters.

Table 1: Infineon Distance2Go Vs TI IWR1642BOOST

Actual Distance *(m)*	Distance Error *(m) @ Azimuth Angle (degrees)*						
	Infenion Distance2Go		Texas Instruments IWR1642BOOST				
	0	15	0	15	30	45	60
1	0.31	0.32	0.3	0.35	0.3	0.3	0.39
1.5	0.38	0.55	0.33	0.33	0.41	0.33	0.41
2	0.43	0.44	0.26	0.31	0.39	0.35	0.39
2.5	0.5	0.48	0.33	0.33	0.46	0.33	0.33
3	0.56	0.39	0.31	0.35	0.35	0.31	0.4
3.5	0.44	0.44	0.33	0.33	0.38	0.33	0.46
4	0.5	0.51	0.36	0.31	0.3	0.3	0.36
4.5	0.55	0.55	0.33	0.38	0.38	0.33	0.51
5	0.44	0.43	0.23	0.36	0.36	0.36	0.4
5.5	0.5	0.5	0.42	0.34	0.34	0.29	0.56
6	0.38	0.52	0.32	0.4	0.4	0.36	0.49
6.5	0.55	0.51	0.3	0.38	0.34	0.43	-
7	0.5	-	0.32	0.36	0.32	0.45	-
Average	***0.46***	***0.47***	***0.32***	***0.35***	***0.36***	***0.34***	***0.43***

Following the range-precision analysis, an experiment was conducted to evaluate the azimuth angle precision of the TI sensor. Similar experimental methodology was used, with the object moving away from the sensor while adjusting the sensor angle from 0 to 60 degrees. The results can be seen in Figure 9, During the experiment, the azimuth angle error exhibited variations ranging from 0.5 to 3.5 degrees. Notably, it was observed that the error improved with increasing distance. This improvement can be attributed to the fact that as the object moves farther away, its target size diminishes, making it relatively easier to identify accurately.

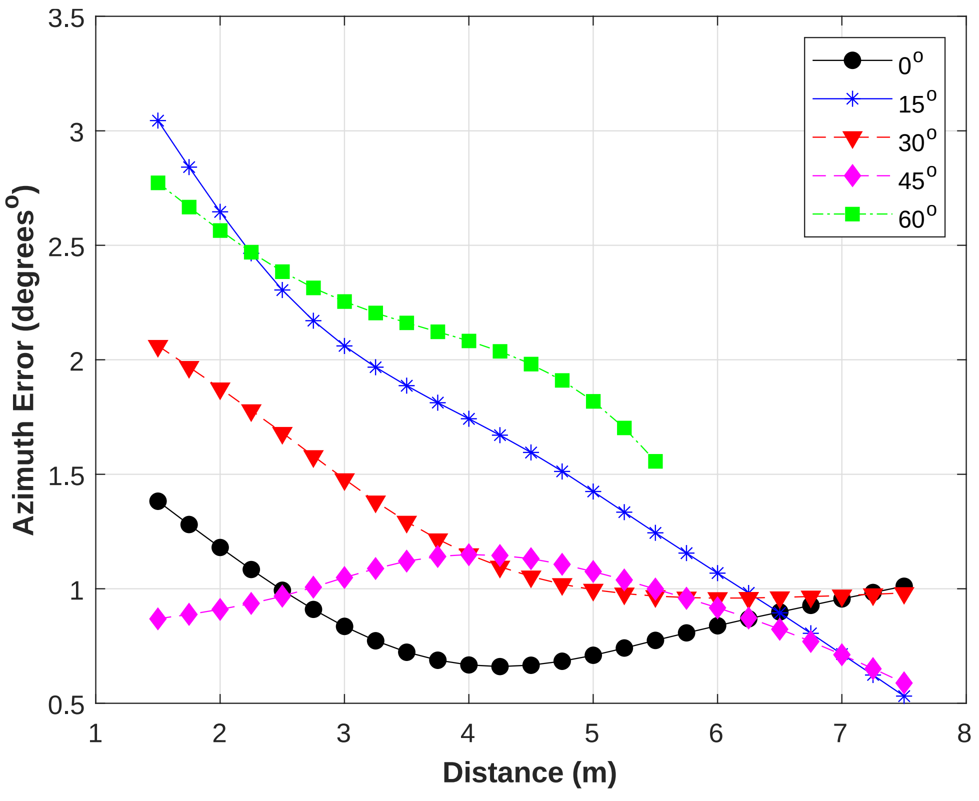


Figure 9: IWR1642BOOST Azimuth Accuracy

Table 2: Precision Analysis of the IWR1642 under various azimuth angles

Distance *(m)*	Azimuth Error *(m) @ Azimuth Angle (degrees)*				
0	**0**	**15**	**30**	**45**	**60**
1.5	1.5	3.5	2	1	3
2.5	1	1.5	2	0.5	2
3.5	0.55	2	1	1.5	2
4.5	0.5	2	1	1.25	2
5.5	1	1	1	0.75	2.5
6.5	0.9	1	1	1	0
7.5	1	0.5	1	0.5	0

1.3.1 Data and Code for 2DOF precision analysis

The following files include the aforementioned results in matlab .mat format:

- `angSensitivity.mat`
- `infineon.mat`
- `TexasInstruments.mat`

The MATLAB function `loadAndPlot2DOFPrecisionAnalysis.m` provided in Appendix 4.1 loads and plots the results.

1.3.2 Discussion on precision results for 2DOF Presicion Analysis

The presented evaluation of the IWR1642BOOST and Infineon sensors provides valuable insights into their range and angle precision capabilities. The experimental setup, involving a drone flying along a straight line with sensors at different orientations, offers a comprehensive analysis of their performance. One key observation is the comparison of distance accuracy between the two sensors, particularly within the narrow field of view of the Infineon sensor (around 20 degrees). The results, depicted in Figure 7, reveal that the TI sensor consistently outperforms the Infineon sensor at both 0 and 15 degrees. The average distance error of 0.32m for the TI sensor contrasts with the higher error of 0.46m exhibited by the Infineon sensor. This discrepancy in accuracy is noteworthy and may impact the suitability of each sensor for specific applications. Moreover, the analysis indicates a general decrease in accuracy with larger angles, as evidenced by the slight increase in error at 15 degrees for both sensors. The limitations become more apparent when examining the TI sensor's performance beyond 15 degrees. Figure 8 shows a constant increase in distance error from 0.32m at 0 degrees to approximately 0.43m at 60 degrees. While the wide 60-degree field of view compensates for the seemingly acceptable error, a critical limitation arises as the sensor fails to detect objects beyond 6 meters.

The assessment of azimuth angle precision, as illustrated in Figure 9, provides additional insights into the TI sensor's performance. The experiment, involving the adjustment of the sensor angle from 0 to 60 degrees as the object moves away, reveals variations in azimuth angle error ranging from 0.5 to 3.5 degrees. Notably, the error improves with increasing distance, attributed to the diminishing target size. This observation underscores the interplay between distance, angle, and object characteristics in influencing sensor precision. In summary, while the TI sensor demonstrates superior performance in distance accuracy within a limited angle range, its limitations become apparent beyond 15 degrees. The azimuth angle precision analysis sheds light on the sensor's ability to adapt to changing distances but highlights variations in accuracy. These findings underscore the importance of considering the specific requirements of an application and the trade-offs between sensor specifications when selecting the most suitable sensor for a given task.

1.4 3-DOF Sensor Precision Analysis

Texas Instrument's IWR1843BOOST is considered as a newer upgrade to the IWR1642BOOST sensor used in the previous experiments. Due to an additional antenna, which provides the user with additional elevation data, allows for 3-DOF. This, in theory, means that 3D positioning could now be achieved using only one sensor. In this section a precision analysis is conducted using the experimental setup described in 1.2.2 as well as 3D positioning experiment results using single- and multi-sensor configuration are showcased.

The aim of this analysis is to evaluate the elevation accuracy of the IWR1843BOOST sensor at different azimuths and distances. By inspecting Figure 10 it can be observed that the elevation error across different azimuth angles follows a similar pattern where the error gradually increases with an increase of the azimuth angle and the distance. At boresight (0 degrees azimuth - Figure 10 a) the sensor measures the elevation quite accurately with the error ranging between 1-2 degrees up to an elevation angle of 30 degrees. Beyond an elevation angle of 30 degrees the sensor fails to provide elevation measurements at distance beyond 2.5*m*. Increasing the azimuth angle has negative effect on the elevation measurement accuracy as shown in Figure 10 b, c, d. It can then be concluded from these plots that the sensor demonstrates a fairly acceptable elevation measurement accuracy (averaging at 3 degrees) within a vertical field of view of around 60 degrees (-30 to +30). That combined with the 90-degree azimuth field-of-view, illustrates that the sensor could cover the majority of the room if placed in the corner.

Table 3: IWR1843BOOST Precision Analysis (0 Azimuth)

Actual Range	Actual Elevation	Measured Range	Measured Azimuth	Measured Elevation	Range Error	Azimuth Error	Elevation Error
0.5	0	0.57	-0.36	-0.39	0.07	0.36	**0.39**
0.5	15	0.52	-1.50	15.60	0.02	1.50	**0.60**
0.5	30	0.54	1.00	30.44	0.04	1.00	**0.44**
0.5	45	0.55	1.43	40.26	0.05	1.43	**4.74**
1.5	0	1.61	-1.43	1.26	0.11	1.43	**1.26**
1.5	15	1.54	0.56	12.46	0.04	0.56	**2.54**
1.5	30	1.57	0.63	27.63	0.07	0.63	**2.37**
1.5	45	1.54	1.63	41.95	0.04	1.63	**3.05**
1.5	60	1.46	-0.97	74.84	0.04	0.97	**14.84**
2.5	0	2.57	-2.15	0.35	0.07	2.15	**0.35**
2.5	15	2.53	-1.32	16.48	0.03	1.32	**1.48**
2.5	30	2.49	-0.20	29.17	0.01	0.20	**0.83**
2.5	45	2.61	0.30	25.60	0.11	0.30	**19.40**
3.5	0	3.84	-1.44	1.29	0.34	1.44	**1.29**
3.5	15	3.66	-1.32	17.26	0.16	1.32	**2.26**
3.5	30	3.62	-0.87	31.27	0.12	0.87	**1.27**
4.5	0	4.83	-1.80	1.91	0.33	1.80	**1.91**
4.5	15	4.73	-0.57	17.19	0.23	0.57	**2.19**
4.5	30	4.67	-1.21	31.81	0.17	1.21	**1.81**
5.5	0	5.81	-2.87	0.03	0.31	2.87	**0.03**
5.5	15	5.72	-0.57	17.54	0.22	0.57	**2.54**
5.5	30	5.67	-0.80	32.39	0.17	0.80	**2.39**
6.5	0	6.72	-2.51	1.00	0.22	2.51	**1.00**
6.5	15	6.74	-0.37	17.65	0.24	0.37	**2.65**
Average					*0.13*	*1.16*	*2.98*
Standart Deviation					*0.10*	*0.70*	*4.53*

Table 4: IWR1843BOOST Precision Analysis (15 Azimuth)

Actual Range	Actual Elevation	Measured Range	Measured Azimuth	Measured Elevation	Range Error	Azimuth Error	Elevation Error
0.5	0	0.58	15.07	0.41	0.08	0.07	**0.41**
0.5	15	0.56	16.52	13.79	0.06	1.52	**1.21**
0.5	30	0.58	16.69	27.95	0.08	1.69	**2.05**
0.5	45	0.59	16.79	48.03	0.09	1.79	**3.03**
1.5	0	1.69	17.10	0.78	0.19	2.10	**0.78**
1.5	15	1.69	17.64	13.71	0.19	2.64	**1.29**
1.5	30	1.66	18.43	28.35	0.16	3.43	**1.65**
2.5	0	2.67	17.47	-0.08	0.17	2.47	**0.08**
2.5	15	2.72	18.16	15.64	0.22	3.16	**0.64**
2.5	30	2.60	19.12	30.76	0.10	4.12	**0.76**
3.5	0	3.66	16.77	3.99	0.16	1.77	**3.99**
3.5	15	3.73	17.28	16.71	0.23	2.28	**1.71**
3.5	30	3.57	18.65	30.58	0.07	3.65	**0.58**
4.5	0	4.81	17.90	3.59	0.31	2.90	**3.59**
4.5	15	4.79	18.35	17.53	0.29	3.35	**2.53**
4.5	30	4.76	18.48	31.11	0.26	3.48	**1.11**
5.5	0	5.76	18.06	3.20	0.26	3.06	**3.20**
5.5	15	5.67	18.27	16.79	0.17	3.27	**1.79**
5.5	30	5.72	18.48	32.34	0.22	3.48	**2.34**
6.5	0	6.82	18.30	5.02	0.32	3.30	**5.02**
6.5	15	6.78	17.97	19.31	0.28	2.97	**4.31**
Average					*0.19*	*2.69*	*2.00*
Standart Deviation					*0.08*	*0.94*	*1.40*

Table 5: IWR1843BOOST Precision Analysis (30 Azimuth)

Actual Range	Actual Elevation	Measured Range	Measured Azimuth	Measured Elevation	Range Error	Azimuth Error	Elevation Error
0.5	0	0.59	30.90	0.34	0.09	0.90	**0.34**
0.5	15	0.58	30.98	14.66	0.08	0.98	**0.34**
0.5	30	0.59	30.85	26.95	0.09	0.85	**3.05**
0.5	45	0.58	32.67	37.13	0.08	2.67	**7.87**
1.5	0	1.67	32.96	0.55	0.17	2.96	**0.55**
1.5	15	1.68	32.18	15.68	0.18	2.18	**0.68**
1.5	30	1.59	33.35	29.46	0.09	3.35	**0.54**
2.5	0	2.70	30.92	1.33	0.20	0.92	**1.33**
2.5	15	2.68	33.45	16.56	0.18	3.45	**1.56**
2.5	30	2.57	33.99	27.08	0.07	3.99	**2.92**
3.5	0	3.77	32.18	3.94	0.27	2.18	**3.94**
3.5	15	3.68	33.56	17.02	0.18	3.56	**2.02**
4.5	0	4.63	34.28	6.57	0.13	4.28	**6.57**
4.5	15	4.63	32.28	17.45	0.13	2.28	**2.45**
4.5	30	4.68	33.73	30.69	0.18	3.73	**0.69**
5.5	0	5.70	34.37	4.24	0.20	4.37	**4.24**
5.5	15	5.70	35.95	16.55	0.20	5.95	**1.55**
5.5	30	5.67	37.79	32.39	0.17	7.79	**2.39**
6.5	0	6.72	33.33	5.23	0.22	3.33	**5.23**
6.5	15	6.80	33.47	16.65	0.30	3.47	**1.65**
Average					***0.16***	***3.16***	***2.50***
Standart Deviation					***0.06***	***1.73***	***2.13***

Table 6: IWR1843BOOST Precision Analysis (45 Azimuth)

Actual Range	Actual Elevation	Measured Range	Measured Azimuth	Measured Elevation	Range Error	Azimuth Error	Elevation Error
0.5	0	0.58	45.88	2.54	0.08	0.88	**2.54**
0.5	15	0.88	46.90	18.25	0.38	1.90	**3.25**
0.5	30	0.68	47.51	14.42	0.18	2.51	**15.58**
1.5	0	1.54	45.25	2.28	0.04	0.25	**2.28**
1.5	15	1.62	48.61	10.42	0.12	3.61	**4.58**
1.5	30	1.58	48.61	19.88	0.08	3.61	**10.12**
2.5	0	2.68	47.67	1.83	0.18	2.67	**1.83**
2.5	15	2.60	48.33	10.36	0.10	3.33	**4.64**
3.5	0	3.80	47.86	1.80	0.30	2.86	**1.80**
3.5	15	3.77	48.17	11.98	0.27	3.17	**3.02**
4.5	0	4.82	47.35	2.46	0.32	2.35	**2.46**
4.5	15	4.79	48.31	12.68	0.29	3.31	**2.32**
5.5	0	5.67	51.80	2.47	0.17	6.80	**2.47**
5.5	15	5.79	52.92	12.55	0.29	7.92	**2.45**
5.5	30	5.79	52.29	21.75	0.29	7.29	**8.25**
6.5	0	6.82	48.64	1.51	0.32	3.64	**1.51**
6.5	15	6.89	62.69	1.17	0.39	17.69	**13.83**
Average					***0.22***	***4.34***	***4.88***
Standart Deviation					***0.11***	***4.01***	***4.37***

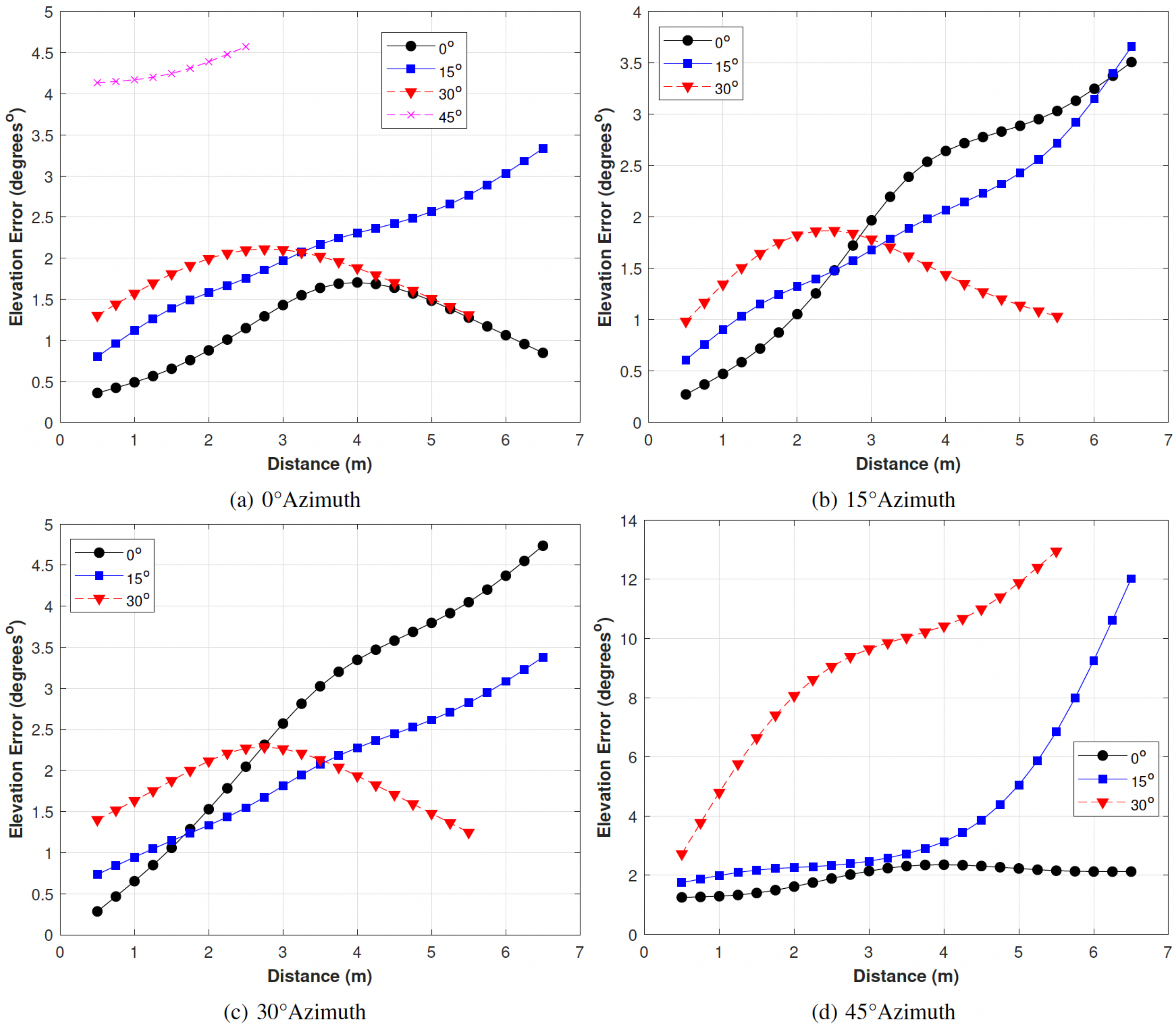
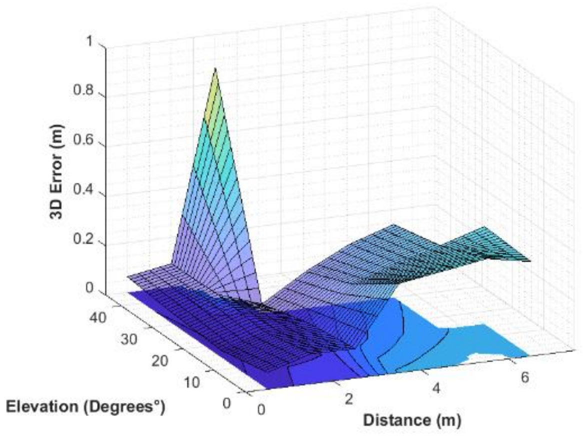
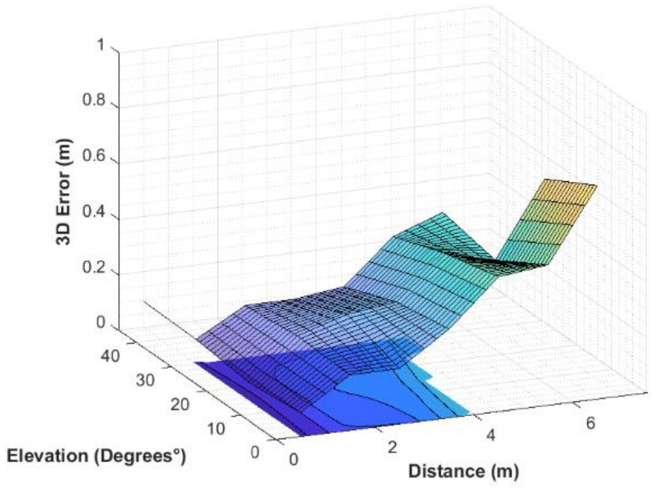


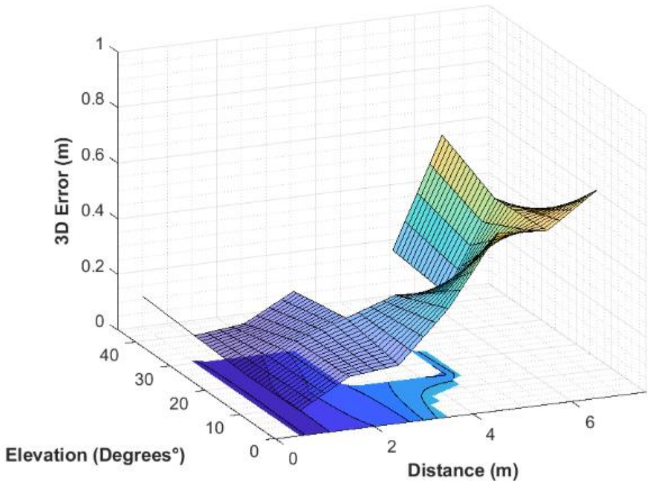
Figure 10: IWR1843BOOST Elevation Accuracy



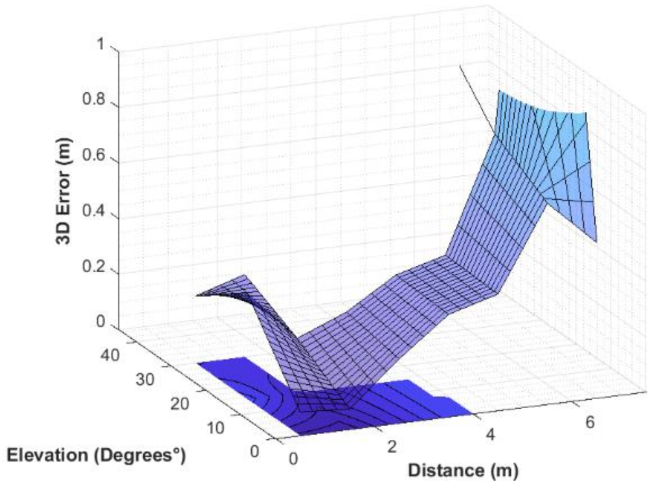
(a) 0°Azimuth



(b) 15°Azimuth



(c) 30°Azimuth



(d) 45°Azimuth

Figure 11: IWR1843BOOST Single Sensor Positioning Accuracy

1.4.1 Data and Code for the 3DOF precision analysis

The following files include the aforementioned 3DOF results in matlab .mat format:

- `precisionIWR1843XYZ.mat`

The MATLAB function `precisionIWR1843XYZ.mat` provided in Appendix 4.2 loads and plots the results.

The function loads 5 AzXX variables. The XX represents the azimuth under which measurements were done (0, 15,30, 45 and 60 degrees). Each of these are arrays that have 19 columns. Each one of these columns represents:

1. The true range of the measurement (meters)
2. The true Azimuth Angle (degrees)

3. The true Elevation Angle (degrees)
4. The Measured distance (meters)
5. The measured Azimuth Angle (degrees)
6. The Measured Elevation Angle (degrees)
7. The range error (meters)
8. The azimuth error (degrees)
9. The elevation error (degrees)
10. The True X (meters)
11. The True Y (meters)
12. The True Z (meters)
13. The Estimated X (meters)
14. The Estimated Y (meters)
15. The Estimated Z (meters)
16. DX (meters)
17. DY (meters)
18. DZ (meters)
19. The Euclidean Error (meters)

1.4.2 Discussion on precision results for 3DOF Presicion Analysis

The precision analysis results for the 3-DOF sensor, particularly the IWR1843BOOST, is similar to the 2-DOF sensor, however, the additional antenna and the extended ability to measure the z-axis introduce a new dimension of analysis, focusing on the elevation error. This expansion enhances the sensor's overall utility by enabling a more comprehensive assessment of spatial accuracy. One notable observation from the precision analysis is the elevation error's tendency to increase as the distance from the sensor to the object grows. This trend is expected, as longer distances introduce more variables and complexities in signal propagation. However, the analysis underscores a critical point regarding close-range measurements at 0.5m. In such scenarios, the elevation errors are relatively higher due to the sensor waves bouncing off different points of the object. This behavior mirrors the azimuth accuracy analysis conducted on the IWR1642BOOST sensor, indicating a common challenge in radar-based systems when dealing with close proximity. The reported elevation error values provide quantitative insights into the sensor's performance. Specifically, the error ranges from 0.4 to 4 degrees for elevation angles of 30 degrees and below, averaging around 2 degrees in less extreme cases. This suggests a reasonable level of accuracy for scenarios involving moderate elevation angles. However, as the azimuth angle increases to 45 degrees and beyond, the error averages at around 4.5 degrees, indicating a degradation in performance. Beyond 45 degrees azimuth, the error exhibits a substantial increase, reaching very high levels.

2 Positioning Measurements

This section describes the measurement methodology and setup to collect position-specific measurements (e.g. distances, azimuths, elevations) for specific points in the environment for the purpose of evaluating the 3D Positioning accuracy of various positioning algorithms (see derivable D3.2).

2.1 2-DOF Positioning Measurements Setup

For positioning accuracy experimentation, the positioning system comprises mainly of a number of TI mmWave sensors each of which is connected to a Raspberry Pi 4 that serves as a gateway collecting the data from each TI sensor and sending it to the central PC for processing. Each sensor has its own Raspberry Pi 4 where the data string is sent through a UDP connection and parsed. A number of TI sensors were deployed in various locations within the lab (indicated with different capital letters in Figure 13 while position estimation was done using 3 approaches: (1) 3D Algebraic Multilateration, (2) 3D Recursive Multilateration and (3) an improved 3D-triangulation approach. Different combinations of sensors were used for each case to investigate the effects of Dilution of Precision (DoP). Eight ground-truth points **(1-8)** were randomly selected across the lab space as shown in Figure 3. Each point was meticulously marked, and their corresponding coordinates were recorded. The drone was positioned precisely on these marked points and subsequently lifted to hover over them at various heights. These heights were also carefully noted down for subsequent analysis. While the drone hovered over each point, the range and angle measurements from each sensor were sent to a central PC that produces the metadata needed to perform 3D positioning calculations using the two approaches mentioned above. This setup allowed for a direct comparison of the accuracy and performance of the two methods for real-time 3D positioning, providing valuable insights into capabilities and suitability for both the methods and the technology for practical applications.

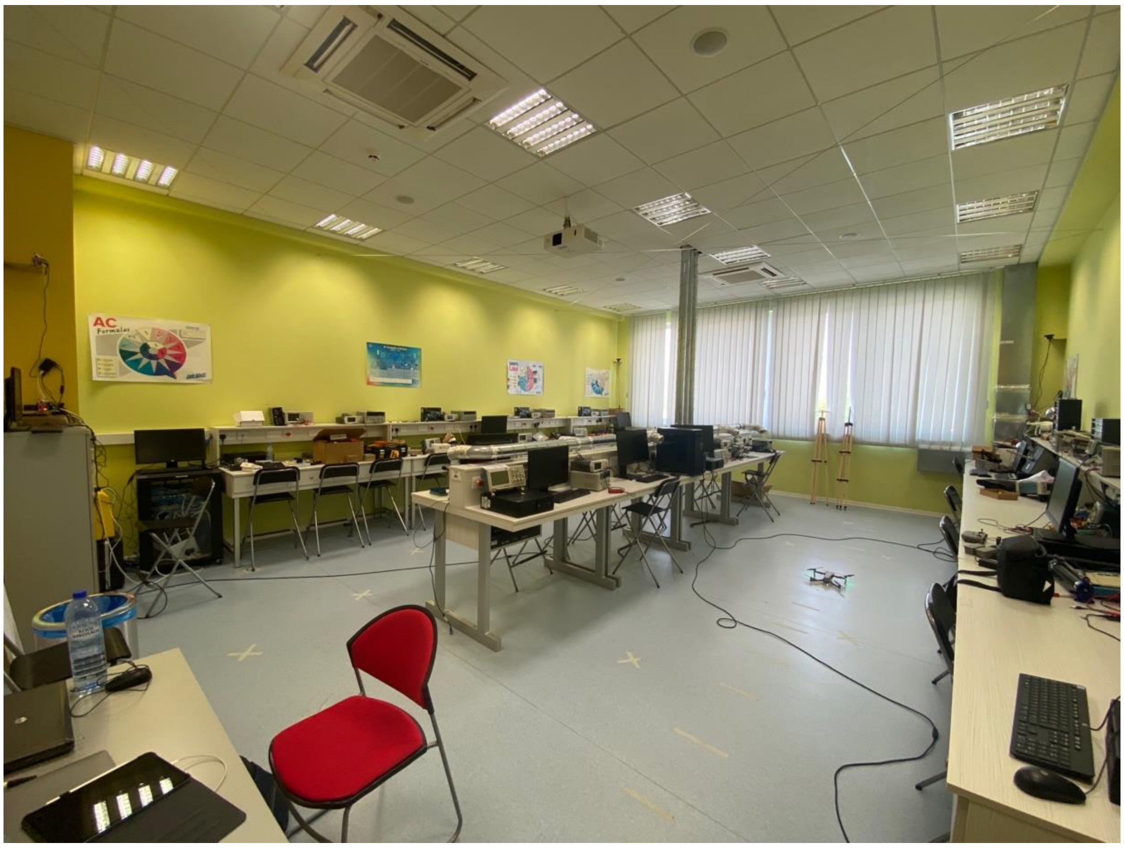


Figure 12: Laboratory Setup for 2DOF Positioning Measurements

2.2 3-DOF Sensor Setup

2.2.1 Equipment

Similarly to IWR1642BOOST, the IWR1843BOOST possesses a Frequency Modulated Continuous Wave (FMCW) transceiver which enables the measurement of range, azimuth angle and velocity of the target. However, due to an additional TX antenna, in addition to the azimuth angling information, it is also able to provide the elevation data of the target. A similar system setup was used for this setup like the one used for the IWR1642BOOST in which each sensor is connected to a Raspberry PI that parses the collected context and sends it to a central PC through a UDP connection.

2.2.2 Experimental Setup

In the pursuit of advancing positioning experiments, a 3-DOF positioning accuracy experiment was conducted utilizing the IWR1843BOOST, similar to 2-DOF experiment that employed the IWR1642BOOST mmWave sensors where the

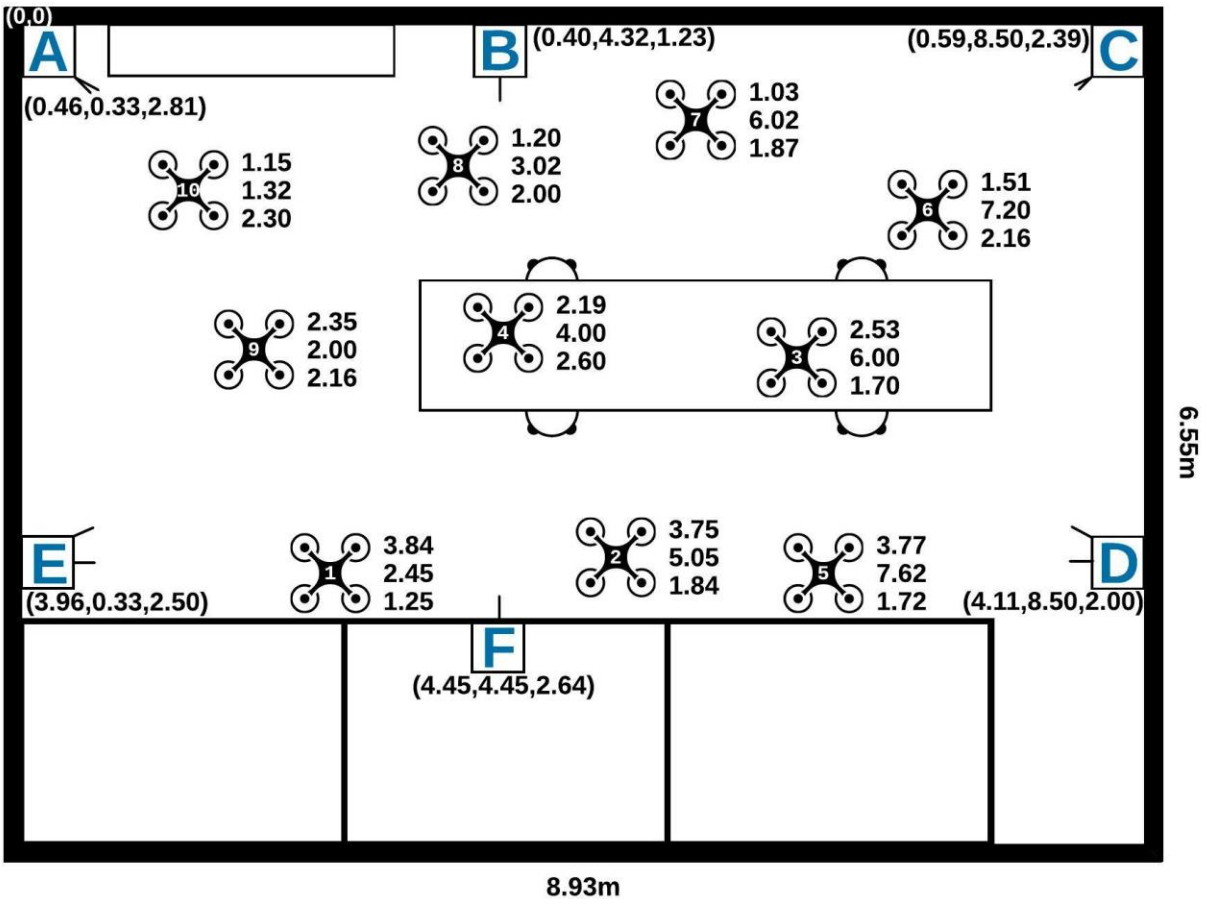


Figure 13 - mmWave 3D Positioning Experimental Setup using 3-DOF mmWave Sensors

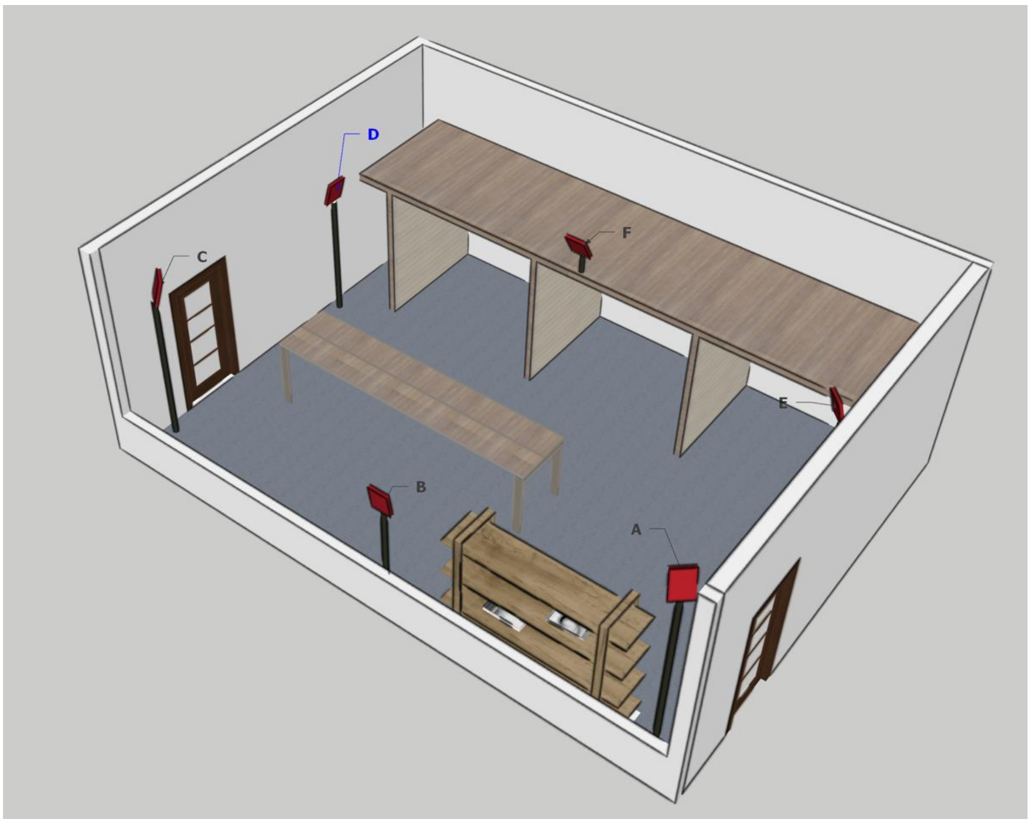


Figure 14 - mmWave 3D Positioning Experimental Setup using 3-DOF mmWave Sensors (3D View)

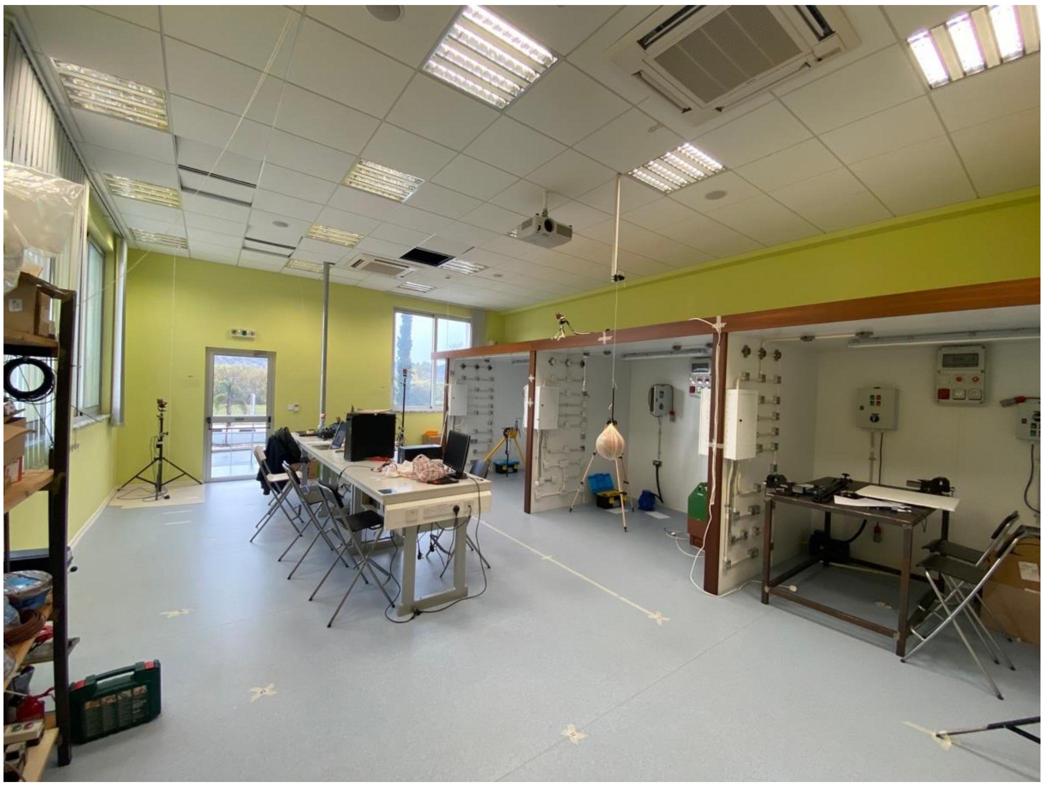


Figure 15: Laboratory Setup for 3DOF Positioning Measurements

drone was hovered across multiple scattered points across the room. This exploration took place in a separate laboratory setting (see Figure 13), emphasizing the versatility and adaptability of the sensor systems across different environments. To ensure a comprehensive assessment of the system's capabilities, the sensors were strategically mounted on adaptable tripods, allowing for flexibility in placement and orientation. Two distinctive sensor setups, denoted as Setup A and Setup B, were meticulously devised to examine varying anchor configurations and orientations. In Setup A, two sensors were strategically positioned in the corners, facing diagonally toward the center of the room, while two additional sensors faced each other in parallel to the wall. This configuration aimed to optimize room coverage, with a slight downward tilt applied to enhance the spatial perception of the environment. Additionally, a fifth sensor was centrally placed along one of the walls, oriented upward to capture data from an alternative perspective. Setup B featured four sensors situated in the corners and oriented diagonally toward the central point of the room. This arrangement was specifically designed to enhance coverage of the central area, with a deliberate tilt to maximize the effectiveness of the system. In this setup, a fifth sensor was strategically elevated and directed downward, compared to the upward orientation in the previous arrangement.

Table 7: Single Anchor mmWave Positioning Accuracy (0 Azimuth)

Actual Range	Actual Elevation	X Error	Y Error	Z Error	3D Error
0.5	0	0.00	-0.07	0.00	0.07
0.5	15	0.01	-0.02	-0.01	0.03
0.5	30	-0.01	-0.03	-0.02	0.04
0.5	45	-0.01	-0.06	0.00	0.06
1.5	0	0.04	-0.11	-0.04	0.12
1.5	15	-0.01	-0.05	0.06	0.08
1.5	30	-0.02	-0.10	0.02	0.10
1.5	45	-0.03	-0.08	0.03	0.09
1.5	60	0.01	0.37	-0.11	0.38
2.5	0	0.10	-0.07	-0.02	0.12
2.5	15	0.06	-0.02	-0.07	0.09
2.5	30	0.01	-0.01	0.04	0.04
2.5	45	-0.01	-0.59	0.64	0.87
3.5	0	0.10	-0.34	-0.09	0.36
3.5	15	0.08	-0.11	-0.18	0.23
3.5	30	0.05	-0.06	-0.13	0.15
4.5	0	0.15	-0.32	-0.16	0.39
4.5	15	0.04	-0.17	-0.23	0.29
4.5	30	0.08	-0.07	-0.21	0.24
5.5	0	0.29	-0.30	0.00	0.42
5.5	15	0.05	-0.15	-0.30	0.34
5.5	30	0.07	-0.02	-0.29	0.29
6.5	0	0.29	-0.22	-0.12	0.38
6.5	15	0.04	-0.15	-0.36	0.39
Average		***0.07***	***0.14***	***0.13***	***0.23***
Standard Deviation		***0.08***	***0.14***	***0.15***	***0.19***

Table 8: Single Anchor mmWave Positioning Accuracy (15 Azimuth)

Actual Range	Actual Elevation	X Error	Y Error	Z Error	3D Error
0.5	0	-0.15	-0.06	0.00	0.17
0.5	15	-0.16	-0.04	0.00	0.16
0.5	30	-0.15	-0.06	-0.02	0.16
0.5	45	-0.11	-0.02	-0.08	0.14
1.5	0	-0.50	-0.11	-0.02	0.51
1.5	15	-0.50	-0.12	-0.01	0.51
1.5	30	-0.46	-0.09	-0.04	0.47
2.5	0	-0.80	-0.04	0.00	0.80
2.5	15	-0.82	-0.08	-0.09	0.83
2.5	30	-0.73	0.06	-0.08	0.74
3.5	0	-1.06	0.00	-0.26	1.09
3.5	15	-1.06	-0.03	-0.17	1.07
3.5	30	-0.98	0.12	-0.07	0.99
4.5	0	-1.47	-0.07	-0.30	1.51
4.5	15	-1.44	0.01	-0.28	1.46
4.5	30	-1.29	0.03	-0.21	1.31
5.5	0	-1.78	0.03	-0.32	1.81
5.5	15	-1.70	0.16	-0.21	1.72
5.5	30	-1.53	0.18	-0.31	1.57
6.5	0	-2.13	0.05	-0.60	2.22
6.5	15	-1.98	0.19	-0.56	2.06
Average		*0.07*	*0.99*	*0.07*	*0.17*
Standard Deviation		*0.08*	*0.63*	*0.05*	*0.18*

Table 9: Single Anchor mmWave Positioning Accuracy (30 Azimuth)

Actual Range	Actual Elevation	X Error	Y Error	Z Error	3D Error
0.5	0	-0.05	-0.07	0.00	0.09
0.5	15	-0.05	-0.07	-0.02	0.09
0.5	30	-0.05	-0.08	-0.02	0.10
0.5	45	-0.07	-0.08	0.00	0.11
1.5	0	-0.16	-0.10	-0.02	0.19
1.5	15	-0.13	-0.11	-0.06	0.19
1.5	30	-0.11	-0.03	-0.03	0.12
2.5	0	-0.14	-0.15	-0.06	0.21
2.5	15	-0.21	-0.05	-0.12	0.25
2.5	30	-0.20	-0.02	0.08	0.21
3.5	0	-0.25	-0.15	-0.26	0.39
3.5	15	-0.26	-0.01	-0.17	0.31
4.5	0	-0.34	0.09	-0.53	0.64
4.5	15	-0.19	0.03	-0.22	0.29
4.5	30	-0.28	0.03	-0.14	0.32
5.5	0	-0.46	0.07	-0.42	0.63
5.5	15	-0.55	0.18	-0.20	0.61
5.5	30	-0.55	0.34	-0.29	0.71
6.5	0	-0.43	0.04	-0.61	0.75
6.5	15	-0.45	0.01	-0.27	0.52
Average		***0.25***	***0.09***	***0.18***	***0.34***
Standard Deviation		***0.17***	***0.08***	***0.18***	***0.23***

Table 10: Single Anchor mmWave Positioning Accuracy (45 Azimuth)

Actual Range	Actual Elevation	X Error	Y Error	Z Error	3D Error
0.5	0	-0.06	-0.05	-0.03	0.08
0.5	15	-0.27	-0.23	-0.15	0.38
0.5	30	-0.18	-0.14	0.08	0.24
1.5	0	-0.03	-0.02	-0.06	0.07
1.5	15	-0.17	-0.03	0.10	0.20
1.5	30	-0.20	-0.06	0.21	0.30
2.5	0	-0.21	-0.04	-0.09	0.23
2.5	15	-0.20	0.01	0.18	0.27
3.5	0	-0.34	-0.07	-0.12	0.37
3.5	15	-0.35	-0.07	0.12	0.38
4.5	0	-0.36	-0.08	-0.21	0.42
4.5	15	-0.41	-0.03	0.11	0.43
5.5	0	-0.56	0.38	-0.24	0.72
5.5	15	-0.75	0.35	0.17	0.84
5.5	30	-0.89	0.08	0.60	1.08
6.5	0	-0.52	0.09	-0.18	0.56
6.5	15	-1.68	1.28	1.54	2.61
Average		***0.42***	***0.18***	***0.25***	***0.54***
Standard Deviation		***0.40***	***0.30***	***0.36***	***0.60***

3 Positioning Context Simulations

To facilitate the generation of positioning context that will support the more extensive investigation of positioning methods for a variety of scenarios a positioning simulator was implemented in matlab utilizing the information collected from the precision analyses of the sensors.

The simulator takes as input the a predefined route of ground truth points, the location (x,y,z) and orientation (yaw, pitch, roll) of the millimeter wave sensors and generates milimeter wave measurements using two error modelling options:

- The user can define the mean and standard deviation of the error in the range, azimuth and elevation measurements and returns an estimated "erroneous" measurement based on a normal distribution.
- The user can import the precision analysis results of the sensor for different ranges, azimuths and elevations and the simulator will generate "errorneous" measurements using linear interpolation of the the precision analysis measurements.

Using the erroneous measurements of the range, azimuth and elevation from the target the simulator then estimates the 3D location (x,y,z) of the target with respect to the room local ENU (East-North-Up) coordinate system. Having available, the range (*r*), the azimuth (θ) and the elevation (φ) measurements from the anchor to the target one can estimate the coordinates of the target with respect to the body-frame coordinate system of the anchor using standard spherical to Cartesian coordinate conversion according the Equation below:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = r \begin{bmatrix} cos(\theta)\, cos\,(\phi) \\ cos(\theta)\, sin\,(\phi) \\ sin\,(\theta) \end{bmatrix}$$

To properly determine the coordinates of the target within the room's coordinate plane, it was imperative to align the coordinate system of the sensor (body frame coordinate system) to that of the room (Local Coordinate System). Achieving this alignment involves a series of calculations that account for the sensor's yaw, pitch, and roll. These adjustments were critical in ensuring that the sensor's data correspond accurately to the room's coordinate plane, allowing for reliable 3D positioning. Assuming that the anchor is first rotated by an angle ψ around the *z*-axis (yaw),

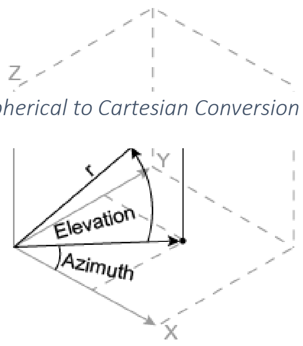


Figure 16: Spherical to Cartesian Conversion

then by an angle θ around *y*-axis (pitch) and finally by an angle ϕ around the *x*-axis (roll) the 3*x*3 rotation matrix is given by:

$$R = R_z + R_y + R_x$$

where,

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\varphi) & -sin(\varphi) \\ 0 & sin(\varphi) & cos(\varphi) \end{bmatrix},\ R_y = \begin{bmatrix} cos(\varphi) & 0 & sin(\varphi) \\ 0 & 1 & 0 \\ -sin(\varphi) & 0 & cos(\varphi) \end{bmatrix},\ R_z = \begin{bmatrix} cos(\varphi) & -sin(\varphi) & 0 \\ sin(\varphi) & cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

With reference to **Error! Reference source not found.** and considering that body-frame measurement from a sensor positioned at $A = [x_a y_a z_a]$ is $P' = [x'y'z']$ then the local coordinates $P = [xyz]$ of the target can be calculated using:

$$P = [R \cdot P'^T]^T + A$$

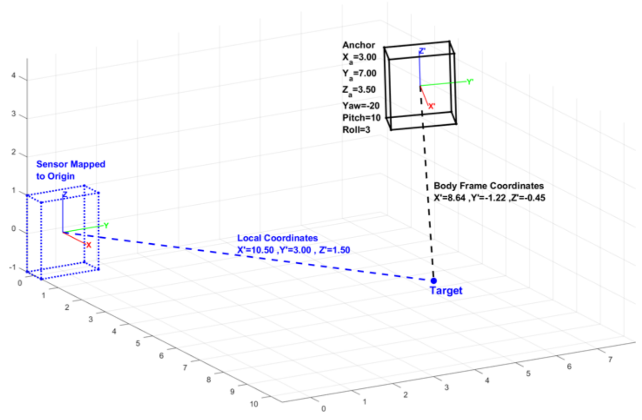


Figure 17:Body frame to Room (ENU) coordinate conversion

The simulator produces one position estimation from each sensor which are then averaged estimate the final position. There is also the option to choose the averaging method, including the removal of outlier measurements which have been taken either at long distances and/or high azimuth, elevation angles.

Figure below shows an indicative result of the simulator. The red dotted line represents the ground trough locations whereas the green solid line represents the stimation, while the black dots at the end of the route indicate the estimates of the 5 sensors which are averaged to estimate the final position. The simulation displays these black dots at real time while the simulation runs. (here we see only line frame at the end of the simuation).

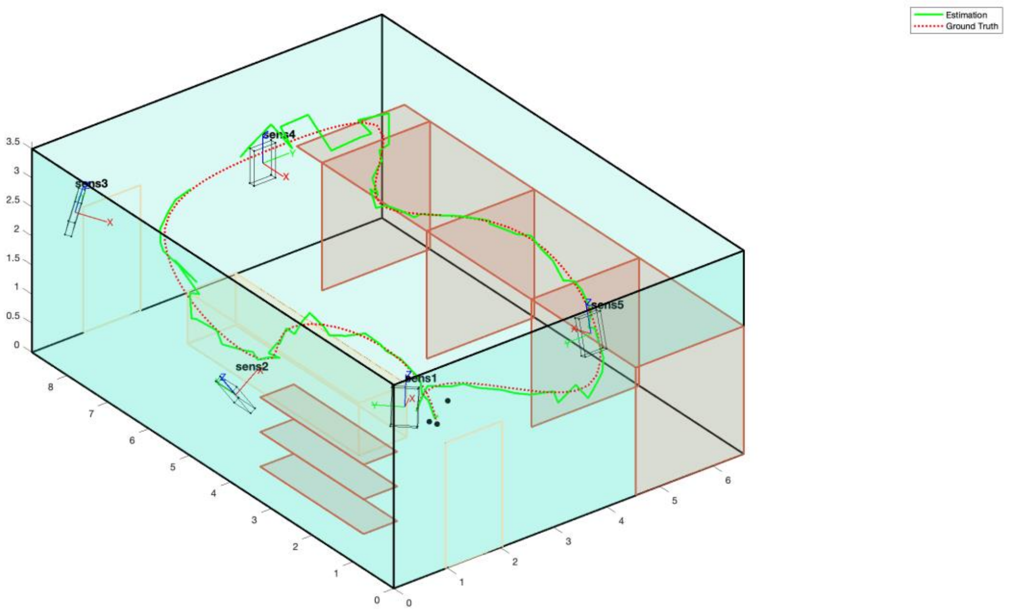


Figure 18: 3D Positioning Simulator Output.

3.1.1 Code and Data for the Simulator

The data of the precision analysis have been interpolated/processed and placed in the file

- `precisionIWR1843_3DMatrices_interpolated.mat`

This file has formed the input of the previously described simulator the code of which is made available in the Appendix 4.3

4 Appendix

4.1 loadAndPlot2DOFPrecisionAnalysis.m

```
% This is a function to load and plot the results of the 2DOF precision Analysis

clear all
close all
load("angSensitivity.mat");
load("infineon.mat");
load("TexasInstruments.mat");

correction=0.15;
all_marks = {'+','o','*','v','x','s','d','^','v','>','<','p','h'};
linestyles= {'-','-','--','--','-.','-.'};
figure
for i=2:size(angSensitivity,2)
    leg(i-1)=cellstr([num2str(angSensitivity(1,i)) '^{o}']);
    xx = 1.5:.25:7.5;
    if i==6
        xx=1.5:0.25:5.5;
    end
    yy=spline(angSensitivity(2:end,1),angSensitivity(2:end,i),xx);
plot(xx,yy,'LineStyle',linestyles{i-1},'Marker',all_marks{mod(i,13)},'Color','black');
hold on
end
legend(leg)
grid on
xlabel('Distance (m)','FontWeight','bold');
ylabel('Azimuth Error (degrees^{o})','FontWeight','bold');

savefig('AzimuthSensitivityTexas.fig');
saveas(gcf,'AzimuthSensitivityTexas.png','png');

figure
hold on
count=1;
for i=2:size(Infineon,2)
    leg(count)=cellstr(['Infineon ' num2str(Infineon(1,i)) '^{o}']);
    count=count+1;
    leg(count)=cellstr(['TI ' num2str(TexasInstruments(1,i)) '^{o}']);
    count=count+1;
    xx = 1.5:.25:7;
    if i==6
        xx=1.5:0.25:5.5;
    end
    yy=spline(Infineon(2:end,1),Infineon(2:end,i)-Infineon(2:end,1)-correction,xx);
    yy2=spline(TexasInstruments(2:end,1),TexasInstruments(2:end,i)-TexasInstruments(2:end,1)-
correction,xx);
plot(xx,yy,'LineStyle',linestyles{i-1},'Marker',all_marks{mod(i,13)},'Color','black');

plot(xx,yy2,'LineStyle',linestyles{i+1},'Marker',all_marks{mod(i+2,13)},'Color','black');
end
legend(leg)
grid on
xlabel('Distance (m)','FontWeight','bold');
ylabel('Distance Error (m)','FontWeight','bold');
savefig('TexasVsInfeneonDistanceAccuracy.fig');
saveas(gcf,'TexasVsInfeneonDistanceAccuracy.png','png');
```

```
figure
hold on
count=1;
for i=2:size(TexasInstruments,2)
    leg(count)=cellstr(['TI ' num2str(TexasInstruments(1,i)) '^{o}']);
    count=count+1;
    xx = 1.5:.25:7;
    if i==6
        xx=1.5:0.25:5.5;
    end
    yy2=spline(TexasInstruments(2:end,1),TexasInstruments(2:end,i)-TexasInstruments(2:end,1)-
correction,xx);
plot(xx,yy2,'LineStyle',linestyles{i-1},'Marker',all_marks{mod(i,13)},'Color','black');
end
legend(leg)
grid on
xlabel('Distance (m)','FontWeight','bold');
ylabel('Distance Error (m)','FontWeight','bold');
savefig('TexasDistanceSensitivity.fig');
saveas(gcf,'TexasDistanceSensitivity.png','png');
```

4.2 precisionIWR1843XYZ.m

```
% This is a function to load and plot the results of the 2DOF precision Analysis
clear all
load("precisionIWR1843XYZ.mat")




[dd(:,:,1),elev(:,:,1),azim(:,:,1),err3d(:,:,1),rangeErr(:,:,1),azErr(:,:,1),elErr(:,:,1)]=plotPrecisio
nAccuracy(AZ00xyz,0);
[dd(:,:,end+1),elev(:,:,end+1),azim(:,:,end+1),err3d(:,:,end+1),rangeErr(:,:,end+1),azErr(:,:,end+1),el
Err(:,:,end+1)]=plotPrecisionAccuracy(AZ15xyz,15);
[dd(:,:,end+1),elev(:,:,end+1),azim(:,:,end+1),err3d(:,:,end+1),rangeErr(:,:,end+1),azErr(:,:,end+1),el
Err(:,:,end+1)]=plotPrecisionAccuracy(AZ30xyz,30);
[dd(:,:,end+1),elev(:,:,end+1),azim(:,:,end+1),err3d(:,:,end+1),rangeErr(:,:,end+1),azErr(:,:,end+1),el
Err(:,:,end+1)]=plotPrecisionAccuracy(AZ45xyz,45);
[dd(:,:,end+1),elev(:,:,end+1),azim(:,:,end+1),err3d(:,:,end+1),rangeErr(:,:,end+1),azErr(:,:,end+1),el
Err(:,:,end+1)]=plotPrecisionAccuracy(AZ60xyz,60);


function
[dnew,elnew,aznew,error3dnew,rangeErrornew,azimuthErrornew,elevationErrornew]=plotPrecisionAccuracy(dat
a,azimuth)
    azimuthString=num2str(azimuth);
    filename=['precisionXYZ_azz' azimuthString];

    [d,el]=meshgrid(0.5:1:6.5,0:15:45);
    error3d=zeros(size(d));
    rangeError=zeros(size(d));
    azimuthError=zeros(size(d));
    elevationError=zeros(size(d));
    for i=1:size(d,1)
        for j=1:size(d,2)
            ind=intersect(find(data(:,1)==d(i,j)),find(data(:,3)==el(i,j)));
            if isempty(ind)
                error3d(i,j)=NaN;
                rangeError(i,j)=NaN;
                azimuthError(i,j)=NaN;
                elevationError(i,j)=NaN;
            else
                error3d(i,j)=data(ind,19);
                rangeError(i,j)=data(ind,7);
                azimuthError(i,j)=data(ind,8);
                elevationError(i,j)=data(ind,9);
            end

        end
    end


    [dnew,elnew]=meshgrid(0.5:.1:6.5,0:1:45);
    aznew=ones(size(elnew))*azimuth;
    rangeErrornew = interp2(d,el,rangeError,dnew,elnew);
    azimuthErrornew = interp2(d,el,azimuthError,dnew,elnew);
    elevationErrornew = interp2(d,el,elevationError,dnew,elnew);
    error3dnew = interp2(d,el,error3d,dnew,elnew);
    fig=figure;
    surf(dnew, elnew, error3dnew,'FaceAlpha',0.5);
    xlabel("Distance (m)",FontWeight="bold");
    ylabel(['Elevation (Degrees' char(176) ')' ],FontWeight="bold",FontSmoothing="on");
    zlabel("3D Error (m)",FontWeight="bold");
    % title(['Azimuth=' azimuthString ' degrees' char(176)])
    % colormap("gray")

    zlim([0 1.5]);
    xlim([0 6.5]);
    ylim([0 45]);
    view(-23,23);
    grid minor;
    threshold=0.4;
    lessThanThresholdIndexes=find(error3dnew>threshold);
    error3DbelowThreshoold=error3dnew;
    error3DbelowThreshoold(lessThanThresholdIndexes)=NaN;
    hold on
    %
scatter3(dnew(lessThanThresholdIndexes),elnew(lessThanThresholdIndexes),zeros(size(dnew(lessThanThresho
ldIndexes))),'o','filled','MarkerFaceAlpha',0.2,'MarkerFaceColor','k')
    contourf(dnew,elnew,error3DbelowThreshoold);
    saveas(fig,filename,'epsc')
end
```

4.3 ThreeDOFMultiSensorPositioningSimulator

```
% Download and place in the same folder the following function
%interparc.m
% https://www.mathworks.com/matlabcentral/fileexchange/34874-interparc

clear all
close all
numberOfElecmentsToInterpolate=100;
maxNumberofScatteredPoints=50;
sigma=0.2;

load('precisionIWR1843_3DMatrices_interpolated.mat');
```

```
%Define the Route
x=[1.2 2.7 4.2 4.8 4.1 5.0 3.0 1.3 1.1 0.9 1.8 2.7 3.1];
y=[1.0, 1.2 0.7 3.0 5.8 6.8 7.1 7.3 7.0 4.2 5.0 5.2 3.0];
z=[2.2 2.0 1.7 3.0 2.7 3.0 3.2 2.4 2.2 1.7 1.5 1.2 0.4];

%Define the room specifications
height=3.5;
roomWidth=6.58;
roomLength=8.866;
workingSpaceStartingX=4.551;
workingSpaceLength=7.692;
workingSpaceLenghtExtra=0.62;
workingSpaceHeight=2.2;
benchStartingX=2;
benchStartingY=3.46;
benchWidth=0.9;
benchLength=4.2;
benchHeight=0.9;
doorStartingX=0.97;
doorWidth=1.07;
doorHeight=2.16;
shelfStartingY=0.74;
shelfWidth=0.63;
shelfLenght=2.53;
shelfHeight=1.79;
numberOfSelves=3;

%Create the environment
building.facet(1)=createVerFacet([0 roomWidth],[0 0],0,height,'external_wall');
building.facet(end+1)=createVerFacet([roomWidth roomWidth],[0 roomLength],0,height,'external_wall');
building.facet(end+1)=createVerFacet([roomWidth 0],[roomLength roomLength],0,height,'external_wall');
building.facet(end+1)=createVerFacet([0 0],[roomLength 0],0,height,'external_wall');
building.facet(end+1)=createHorFacet([0 roomWidth roomWidth 0],[0 0 roomLength roomLength],0,'floor');
for i=0:3
    building.facet(end+1)=createVerFacet([workingSpaceStartingX roomWidth],[i*workingSpaceLength/3
i*workingSpaceLength/3],0,workingSpaceHeight,'wood');
end
building.facet(end+1)=createHorFacet([workingSpaceStartingX roomWidth roomWidth
workingSpaceStartingX],[0 0 workingSpaceLength+workingSpaceLenghtExtra
workingSpaceLength+workingSpaceLenghtExtra],workingSpaceHeight,'wood');
building.facet(end+1)=createVerFacet([benchStartingX benchStartingX],[benchStartingY
benchStartingY+benchLength],0,benchHeight,'bench');
building.facet(end+1)=createVerFacet([benchStartingX+benchWidth
benchStartingX+benchWidth],[benchStartingY benchStartingY+benchLength],0,benchHeight,'bench');
building.facet(end+1)=createVerFacet([benchStartingX benchStartingX+benchWidth],[benchStartingY
benchStartingY],0,benchHeight,'bench');
building.facet(end+1)=createVerFacet([benchStartingX
benchStartingX+benchWidth],[benchStartingY+benchLength
benchStartingY+benchLength],0,benchHeight,'bench');
building.facet(end+1)=createHorFacet([benchStartingX benchStartingX+benchWidth
benchStartingX+benchWidth benchStartingX],[benchStartingY benchStartingY benchStartingY+benchLength
benchStartingY+benchLength],benchHeight,'bench');
building.facet(end+1)=createVerFacet([doorStartingX doorStartingX+doorWidth],[0
0],0,doorHeight,'bench');
building.facet(end+1)=createVerFacet([doorStartingX doorStartingX+doorWidth],[roomLength
roomLength],0,doorHeight,'bench');
for i=1:numberOfSelves
    building.facet(end+1)=createHorFacet([0 shelfWidth shelfWidth 0],[shelfStartingY shelfStartingY
shelfStartingY+shelfLenght shelfStartingY+shelfLenght],i*shelfHeight/numberOfSelves,'wood');
end

figure
plot_environment(building); view(3)
rotate3d on; hold on; axis equal
```

```
%Define the location and orientation of the sensors
sens(1)=createSensor(45,15,0,0.46,0.33,2.81,1,'sens1');
sens(2)=createSensor(0,-35,0,0.34,4.32,1.23,1,'sens2');
sens(3)=createSensor(-45,15,0,0.53,8.5,2.39,1,'sens3');
sens(4)=createSensor(-90,0,0,4.06,8.5,1.995,1,'sens4');
sens(5)=createSensor(90,15,0,3.96,0.33,2.84,1,'sens5');

%Interpolate points
pt = interparc(numberOfElecmentsToInterpolate,x,y,z)
xData=pt(:,1);
yData=pt(:,2);
zData=pt(:,3);

numberOfscatteredPoints=randi(maxNumberofScatteredPoints);
scatterX=normrnd(xData(1),sigma,[1 numberOfscatteredPoints]);
scatterY=normrnd(yData(1),sigma,[1 numberOfscatteredPoints]);
scatterZ=normrnd(zData(1),sigma,[1 numberOfscatteredPoints]);

scatterNewMeasurements=scatter3([],[],[],'k','filled','HandleVisibility','off')
estimatedPlot=plot3(1,1,1,'-g','LineWidth',2);
groundTruth=plot3(xData,yData,zData,':r','LineWidth',2)
legend('Estimation', 'Ground Truth')

for i=1:length(xData)
    numberOfscatteredPoints=randi(maxNumberofScatteredPoints);
    scatterX=normrnd(xData(i),sigma,[1 numberOfscatteredPoints]);
    scatterY=normrnd(yData(i),sigma,[1 numberOfscatteredPoints]);
    scatterZ=normrnd(zData(i),sigma,[1 numberOfscatteredPoints]);
    for iSens=1:length(sens)

tempXYZ(iSens,:)=rotateToSensorFrame(xData(i),yData(i),zData(i),sens(iSens).yaw,sens(iSens).pitch,sens(
iSens).roll,sens(iSens).x, sens(iSens).y, sens(iSens).z)
        [AZ, EL, R]=cart2sph(tempXYZ(iSens,1),tempXYZ(iSens,2),tempXYZ(iSens,3));
        polarMeasAZ_EL_R(iSens,:)=[-AZ*180/pi EL*180/pi R];

        %Interpollate the errors based on the loaded precision analysis
        %results calculated using the precisionIWR1843XYZ.m file
        azError=interp3(dd,elev,azim,azErr,R,abs(EL*180/pi),abs(AZ*180/pi));
        elError=interp3(dd,elev,azim,elErr,R,abs(EL*180/pi),abs(AZ*180/pi));
        dError=interp3(dd,elev,azim,rangeErr,R,abs(EL*180/pi),abs(AZ*180/pi));
        AZnew=polarMeasAZ_EL_R(iSens,1)+unifrnd(-azError,azError,size(AZ));
        ELnew=polarMeasAZ_EL_R(iSens,2)+unifrnd(-elError,elError,size(EL));
        Rnew=polarMeasAZ_EL_R(iSens,3)+unifrnd(-dError,dError,size(R));
        [XnewM, YnewM, ZnewM]=sph2cart(-AZnew*pi/180,ELnew*pi/180,Rnew);

room_pts=rotateToRoomFrame(XnewM,YnewM,ZnewM,sens(iSens).yaw,sens(iSens).pitch,sens(iSens).roll,sens(iS
ens).x, sens(iSens).y, sens(iSens).z);
        newMeasurementXYZ(iSens,:)=room_pts';
    end

    set(scatterNewMeasurements,'XData',newMeasurementXYZ(:,1),'YData',newMeasurementXYZ(:,2),'ZData',
newMeasurementXYZ(:,3))
    estimatedX(i)=nanmean(newMeasurementXYZ(:,1));
    estimatedY(i)=nanmean(newMeasurementXYZ(:,2));
    estimatedZ(i)=nanmean(newMeasurementXYZ(:,3));
    absAZ=abs(AZnew);
    absEL=abs(ELnew);
    absR=abs(Rnew);
    Dx(i)=xData(i)-estimatedX(i);
    Dy(i)=yData(i)-estimatedY(i);
    Dz(i)=zData(i)-estimatedZ(i);
    Dxyz(i)=norm([Dx(i) Dy(i) Dz(i)]);
    set(estimatedPlot,'XData',estimatedX,'YData',estimatedY,'ZData',estimatedZ)
    pause(0.1)
end

%NEEDED Functions
function room_pts=rotateToRoomFrame(X,Y,Z,yaw,pitch,roll,anchorX, anchorY, anchorZ)
    pts=[X; Y; Z];
    anchor=[anchorX; anchorY; anchorZ];
    rot_z = [ cosd(yaw), -sind(yaw), 0; ...
              sind(yaw),  cosd(yaw), 0; ...
              0, 0, 1];
    rot_y = [cosd(pitch), 0,   sind(pitch); ...
                         0, 1,              0; ...
            -sind(pitch), 0,  cosd(pitch)];
    rot_x = [1, 0, 0; ...
              0, cosd(roll), -sind(roll); ...
              0, sind(roll),  cosd(roll)];
    rotMatrix=rot_z * rot_y * rot_x;
    room_pts = rotMatrix * pts+repmat(anchor,[1 size(pts,2)]);
end

function sensor_pts=rotateToSensorFrame(X,Y,Z,yaw,pitch,roll,anchorX, anchorY, anchorZ)
    pts=[X; Y; Z];
    anchor=[anchorX; anchorY; anchorZ];
    rot_z = [ cosd(yaw), -sind(yaw), 0; ...
              sind(yaw),  cosd(yaw), 0; ...
              0, 0, 1];
    rot_y = [cosd(pitch), 0,   sind(pitch); ...
                         0, 1,              0; ...
            -sind(pitch), 0,  cosd(pitch)];
    rot_x = [1, 0, 0; ...
              0, cosd(roll), -sind(roll); ...
              0, sind(roll),  cosd(roll)];
    rotMatrix=rot_z * rot_y * rot_x;
    sensor_pts = rotMatrix\(pts-repmat(anchor,[1 size(pts,2)]));
end

function facet=createVerFacet(X,Y,ground,ceiling,type)
    facet.coordinates=[X(1) Y(1) ground; X(2) Y(2) ground;X(2) Y(2) ceiling;X(1) Y(1) ceiling];
    facet.type=type;
    facet.child=[];
end

function facet=createHorFacet(X,Y,Elevation,type)
    for i=1:length(X)
        facet.coordinates(i,:)=[X(i) Y(i) Elevation];
    end
    facet.type=type;
    facet.child=[];
end

%Function to plot the environment
function plottedEnv=plot_environment( building)
plottedEnv=[];
alph=0.2;
%Plot the Building
for i=1:length(building.facet)
  switch building.facet(i).type
        case 'external_wall'
            building.facet(i).coordinates=[building.facet(i).coordinates;...
                    building.facet(i).coordinates(1,:)];
            plottedEnv(i)=line(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),...
                building.facet(i).coordinates(:,3),'Linewidth',2,'Color','k','HandleVisibility','off');
            patch(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),building.facet(i).coordinates(:,3
),[0.3 0.9 0.8],'FaceAlpha',alph,'HandleVisibility','off');
        case 'internal_wall'
            building.facet(i).coordinates=[building.facet(i).coordinates;...
                    building.facet(i).coordinates(1,:)];
            plottedEnv(i)=line(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),...
                building.facet(i).coordinates(:,3),'Linewidth',1,'Color',[0.88 0.75
0.46],'HandleVisibility','off');
            patch(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),building.facet(i).coordinates(:,3
),[0.88 0.75 0.46],'FaceAlpha',alph,'HandleVisibility','off');
                    case 'floor'
            building.facet(i).coordinates=[building.facet(i).coordinates;...
                    building.facet(i).coordinates(1,:)];
            % plottedEnv(i)=line(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),...
            %     building.facet(i).coordinates(:,3),'Linewidth',2,'Color','b');
            patch(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),building.facet(i).coordinates(:,3
),[0.3 0.9 0.8],'FaceAlpha',alph,'HandleVisibility','off');
        case 'wood'
%
patch(building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),building.facet(i).coordinat
es(:,3),[0.3 0.9 0.8]);
building.facet(i).coordinates=[building.facet(i).coordinates;...
                    building.facet(i).coordinates(1,:)];
            plottedEnv(i)=line(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),...
                building.facet(i).coordinates(:,3),'Linewidth',2,'Color',[0.8 0.4
0.3],'HandleVisibility','off');
            patch(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),building.facet(i).coordinates(:,3
),[0.8 0.4 0.3],'FaceAlpha',alph,'HandleVisibility','off');

      case 'bench'
%
patch(building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),building.facet(i).coordinat
es(:,3),[0.3 0.9 0.8]);
building.facet(i).coordinates=[building.facet(i).coordinates;...
                    building.facet(i).coordinates(1,:)];
            plottedEnv(i)=line(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),...
                building.facet(i).coordinates(:,3),'Linewidth',2,'Color',[0.93 0.88
0.73],'HandleVisibility','off');
            patch(
building.facet(i).coordinates(:,1),building.facet(i).coordinates(:,2),building.facet(i).coordinates(:,3
),[0.93 0.88 0.73],'FaceAlpha',alph,'HandleVisibility','off');
      otherwise
    end

  if (~isempty(building.facet(i).child))
     for (k=1:1:length(building.facet(i).child))
         switch building.facet(i).child(k).type

             case 'window'
                 patch(
building.facet(i).child(k).coordinates(:,1),building.facet(i).child(k).coordinates(:,2),building.facet(
i).child(k).coordinates(:,3),[0.3 0.9 0.8],'HandleVisibility','off');
             case 'wooden_door'
                 patch(
building.facet(i).child(k).coordinates(:,1),building.facet(i).child(k).coordinates(:,2),building.facet(
i).child(k).coordinates(:,3),[0.8 0.4 0.3],'HandleVisibility','off');
```

```
              case 'metal_door'
                  patch(
building.facet(i).child(k).coordinates(:,1),building.facet(i).child(k).coordinates(:,2),building.facet(
i).child(k).coordinates(:,3),[0.43 0.38 0.38],'FaceAlpha',alph,'HandleVisibility','off');
              otherwise
          end
       end
  end
end

end

function sensor=createSensor(yaw,pitch,roll,x,y,z,plotSensor,sensorName)
sensor.height=0.3;sensor.width=0.2;sensor.depth=0.05;
sensor.yaw=yaw;sensor.pitch=pitch;sensor.roll=roll;
sensor.x=x;sensor.y=y;sensor.z=z;
sensor.name=sensorName;
location=[x y z]';
if plotSensor
    % Set up an array of points for a cube
    pts = [1 1 1 1 -1 -1 -1 -1 ;
        1 1 -1 -1 1 1 -1 -1 ;
        1 -1 1 -1 1 -1 1 -1 ];
    pts=pts.*[sensor.depth;sensor.width;sensor.height];
    rot_z = [ cosd(yaw), -sind(yaw), 0; ...
          sind(yaw),  cosd(yaw), 0; ...
          0, 0, 1];
    rot_y = [cosd(pitch), 0,   sind(pitch); ...
                   0, 1,                0; ...
            -sind(pitch), 0,  cosd(pitch)];
    rot_x = [1, 0, 0; ...
              0, cosd(roll), -sind(roll); ...
              0, sind(roll),  cosd(roll)];
    % Rotate the points
    rot_pts = rot_z * rot_y * rot_x * pts+repmat(location,[1 size(pts,2)]);
    % Plot the axis marker
    axisSize=0.5;
    axesPoints=[0 axisSize  0 0 0 0;
        0 0 0 axisSize 0 0;
        0 0 0 0 0 axisSize];
    rot_axesPoints=rot_z * rot_y * rot_x * axesPoints+repmat(location,[1 6]);
    plot3([rot_axesPoints(1,1) rot_axesPoints(1,2)],[rot_axesPoints(2,1) rot_axesPoints(2,2)],
[rot_axesPoints(3,1) rot_axesPoints(3,2)],'r','linewidth',1,'HandleVisibility','off');
    hold on;
    plot3([rot_axesPoints(1,3) rot_axesPoints(1,4)],[rot_axesPoints(2,3) rot_axesPoints(2,4)],
[rot_axesPoints(3,3) rot_axesPoints(3,4)],'g','linewidth',1,'HandleVisibility','off');
    plot3([rot_axesPoints(1,5) rot_axesPoints(1,6)],[rot_axesPoints(2,5) rot_axesPoints(2,6)],
[rot_axesPoints(3,5) rot_axesPoints(3,6)],'b','linewidth',1,'HandleVisibility','off');
    plot3(rot_pts(1,:), rot_pts(2,:), rot_pts(3,:),'.k','MarkerSize',5,'HandleVisibility','off');

    % Plot the edges (original then rotated)
    segment_order = [1,2;3,4;5,6;7,8;1,3;2,4;5,7;6,8;1,5;2,6;3,7;4,8];

    for i = 1:size(segment_order, 1)
       % plot3(pts(1,segment_order(i,:)), pts(2,segment_order(i,:)), pts(3,segment_order(i,:)), ':b');
       plot3(rot_pts(1,segment_order(i,:)), rot_pts(2,segment_order(i,:)),
rot_pts(3,segment_order(i,:)), '-k','HandleVisibility','off');
    end
    text(rot_axesPoints(1,2),rot_axesPoints(2,2),rot_axesPoints(3,2),'X','Color','red')
    text(rot_axesPoints(1,4),rot_axesPoints(2,4),rot_axesPoints(3,4),'Y','Color','green')
    text(rot_axesPoints(1,6),rot_axesPoints(2,6),rot_axesPoints(3,6),'Z','Color','blue')
    text(x,y,z+0.5,sensorName,"FontSize",12,"FontWeight","bold")
end
end
```

5 References

[1] D. Wang, M. Fattouche and X. Zhan, “Pursuance of mm-Level Accuracy: Ranging and Positioning in mmWave Systems,” *IEEE Systems Journal,* vol. 13, no. 2, pp. 1169-1180, 2019.

[2] C. Laoudias, A. Moreira, S. Kim, S. Lee, A. Wirola and C. Fischione, “A Survey of Enabling Technologies for Network Localization, Tracking, and Navigation,” *IEEE Communications Surveys & Tutorials,* vol. 20, no. 4, pp. 3607-3644, 2018.

[3] F. Zafari, A. Gkelias and K. Leung, “A Survey of Indoor Localization Systems and Technologies,” *IEEE Communications Surveys & Tutorials,* vol. 21, no. 3, pp. 2568-2599, 2019.

[4] Y. Han, Y. Shen, X. Zhang, M. Z. Win and H. Meng, “ "Performance Limits and Geometric Properties of Array Localization," in IEEE Transactions on Information Theory, vol. 62, no. 2, pp. 1054-1075, Feb. 2016, doi: 10.1109/TIT.2015.2511778.”.

[5] Y. Zhao, J. C. Y. Liang, X. Sha and W. Li, “Adaptive 3D Position Estimation of Pedestrians by Wearing One Ankle Sensor,” *IEEE Sensors Journa,* vol. 20, no. 19, pp. 11642-11651, 2020.

[6] S. Yu and e. Al., “A Low-Complexity Autonomous 3D Localization Method for Unmanned Aerial Vehicles by Binocular Stereovision Technology,” in *10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), ,* 2018.

[7] T, Wild; et. al, ““Joint design of communication and sensing for beyond 5g and 6g systems,” *IEEE Access,* vol. 9, no. 30, pp. 845-857, 2021.

[8] Hao, Z; et. al, “Millimetre-wave radar localization using indoor multipath effect,,” *Sensors,* vol. 22, p. 5671, 2022.

[9] Saily, M; et. al, “Positioning technology trends and solutions toward 6g,” in *IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2021.

[10] A. Bourdoux; et al., “6g white paper on localization and sensing,” 2020. [Online]. Available: https://arxiv.org/abs/2006.01779. [Accessed May 2023].