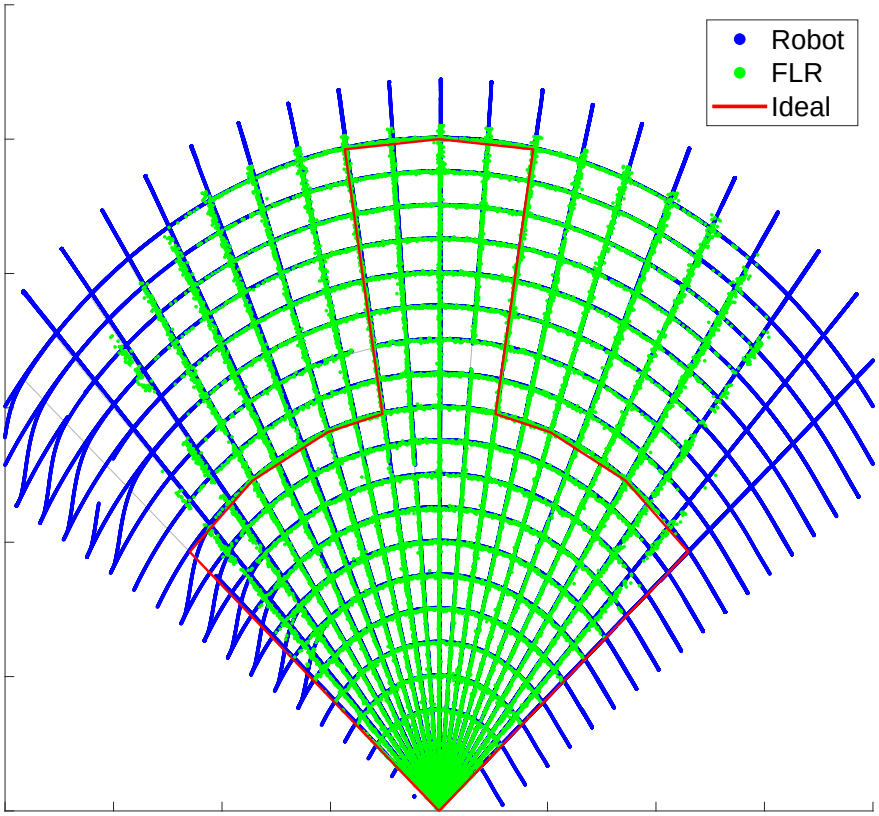


Design, Implementation, and Evaluation of Radar Sensor Models in a Virtual Environment

Master's thesis in Systems, Control, and Mechatronics

JACOB CARLBACK, SARA MJÖBERG

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Design, Implementation, and Evaluation of Radar Sensor Models in a Virtual Environment

JACOB CARLBACK, SARA MJÖBERG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2023

Design, Implementation, and Evaluation of Radar Sensor Models in a Virtual Environment

JACOB CARLBACK
SARA MJÖBERG

Supervisor: Carina Björnsson, Volvo Cars
Examiner: Jonas Sjöberg, Electrical Engineering

Master's Thesis 2023
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Measurements from the forward looking radar sensor (green) along with the trajectory of the robot (blue) and the idealistic sensor model (red).

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Design, Implementation, and Evaluation of Radar Sensor Models in a Virtual Environment

JACOB CARLBACK
SARA MJÖBERG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This project aims to design, implement, and evaluate radar sensor models to be used in a virtual environment as the current virtual sensor model lacks realistic radar properties such as measurement error and detection rate. Data driven models were designed using collected data measurements from testing of the forward looking radar. The testing was conducted with a robot mounted with a pedestrian child doll driving in a fan and circular pattern. New field of view models were designed as a concave and convex polygon as well as a polynomial consisting of three polynomial equations. Analyzing the fan pattern measurement error showed that small areas had a similar magnitude and direction. Therefore, a segment grid similar to the testing grid was constructed in which the mean and variance were calculated. The model samples an error from the normal distribution in each segment. Furthermore, the measurement error model calculates an error depending on the target trajectory. The detection rate model calculates the probability of detection from the number of detections and misses in each segment. Each model was evaluated based on their accuracy, execution time, and repeatability. The evaluation showed that the concave model was the most optimal when evaluating the execution time and field of view. The measurement error model did not affect the number of detections and correctly calculated the error in the fan and circle pattern, further data collection is needed to verify other driving trajectories. The detection rate model altered the field of view based on the probability of detections in each segment.

Keywords: radar; sensor model; virtual testing; measurement error; detection rate; ADAS functions; OSI; PIP

Acknowledgements

First and foremost we would like to thank Carina Björnsson for being our supervisor as well as Rickard Johansson for providing us with information, insight, and help within CSPAS. We would also like to thank Mikael Andersson for giving us a good introduction to CSPAS and giving us good feedback on our presentation. Furthermore, we would like to thank Karl Vanäs and William Milqvist from the radar team for giving us access to the collection of radar data as well as for presenting us with a good introduction to radar systems. We would also like to thank the whole team at Collision Avoidance for always being kind and being good people to discuss the thesis with. Of course, we also want to thank our examiner Jonas Sjöberg for giving us feedback as well as helping us with structuring the objectives for the thesis.

Jacob Carlback, Sara Mjöberg, Gothenburg, June, 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ACC	Adaptive Cruise Control
ADAS	Advanced Driver-Assisted Systems
AD	Autonomous Driving
AEB	Autonomous Emergency Braking
CPTA	Car-Pedestrian-Turning-Adult
CPTC	Car-Pedestrian-Turning-Child
CSPAS	Compiled Simulation Platform for Active Safety
CW	Collision Warning
EM	Electromagnetic
EuroNCAP	European New Car Assessment Programme
FLR	Forward Looking Radar
FMCW	Frequency-Modulated Continuous Wave
FOV	Field Of View
LRR	Long Range Radar
OSI	Open Simulation Interface
PIP	Point in Polygon
SRR	Short Range Radar
VCC	Volvo Car Corporation

Nomenclature

Below is the nomenclature of symbols that have been used throughout this thesis.

N_s	Number of samples
N_d	Number of detections
N_m	Number of missed samples
N_{si}	Number of samples inside
N_{so}	Number of samples outside
N_{op}	Number of outer points
N_v	Number of vertices
s	degree of polynomial equation
R	Range
θ	Azimuth angle
α	Rotational angle of object
β	Angle between global and sensor coordinate system
q	Distance
d	Distance error
W	Width
L	Length
μ	Expectation value
σ	Standard deviation
(x_c, y_c)	Center coordinates of object
(x_g, y_g)	Global coordinates
(x_s, y_s)	Sensor coordinates
(x_{sp}, y_{sp})	Sensor position coordinates
(x_r, y_r)	Radar GPS coordinates
ts	Timestamp

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

This chapter introduces the background, contributions, related work, a detailed overview of the Compiled Simulation Platform for Active Safety, and a thesis outline.

1.1 Background

When designing technology it is important to ensure safe operation for the end user. This is especially true for car technology where accidents can have fatal consequences. However, research has found that more than 90 % of road accidents are caused by driver and human error [4]. In a study presented in [5, p. 86] it was theoretically deduced that implementation of Advanced Driver-Assisted Systems (ADAS) functions could prevent up to 43 % of all car crashes. Therefore, car manufacturing companies, such as Volvo Cars Corporation (VCC), put a lot of effort into the development of ADAS functions, which rely heavily on perception sensors such as e.g. radars and cameras. Some ADAS functions are warning systems such as Collision Warning (CW), which triggers an alarm in order to alert the driver a collision is close and action is needed. ADAS also consists of action-taking systems such as Autonomous Emergency Braking (AEB), where the function brakes when the driver is not performing any action to prevent the collision.

To ensure the functions behave as expected, VCC performs an extensive amount of testing and verification. Traditionally, this is performed on a test track. However, properly setting up various scenarios on a test track can be time consuming, and more complex driving scenarios can be difficult to perform. Moreover, in order to fully validate ADAS functions, it is necessary to test for billions of kilometers [6]. This is infeasible to perform on a test track. Therefore, it is more suitable to perform testing and validation in a virtual environment.

Virtual environment testing and validation of ADAS functions is already conducted at VCC in their own virtual simulation environment, Compiled Simulation Platform for Active Safety (CSPAS). CSPAS is a closed-loop, software-in-the-loop platform, with the purpose to test and validate, among other things, ADAS functions [7]. Therefore, CSPAS simulates vehicle dynamics such as brake and steering, as well as the sensors within the car, to obtain realistic simulation conditions for testing virtually. Since ADAS functions rely heavily on sensors for perception, the sensors need to be simulated realistically in order to achieve proper function activation.

However, the Forward Looking Radar (FLR) model in CSPAS has an idealistic Field Of View (FOV), in the sense that objects that enter the FOV are detected instantaneously with exact coordinates, whereas a realistic radar sensor has some measurement error and detection rate which varies depending on the angle and range to an object. Furthermore, the idealistic FOV does not truly represent the visual field of the real radar since the FOV is defined as the area bounded by a circle sector with a radius range R and angle ϕ. As a result of this, testing and verification of ADAS functions conducted within the virtual environment of CSPAS will not accurately represent a real world scenario. Therefore a new FOV model to more accurately represent the visual field needs to be designed, along with a measurement error and detection rate model.

The current idealistic FOV of the FLR in the virtual environment is modeled by combining the FOV of two radar models represented as circle sectors [8], see Figure 1.1. One Short Range Radar (SRR) with a radius range R_{SRR} and angle $2\theta_{SRR}$, and one Long Range Radar (LRR) with a radius range R_{LRR} and angle $2\theta_{LRR}$. The angle θ, also known as the azimuth angle, is the horizontal angle from the center front of the car, where the sensor is placed. The azimuth angle θ is half the circle sector angle ϕ.

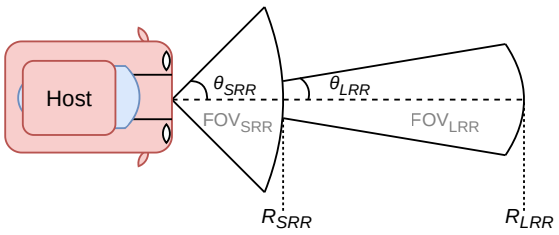


Figure 1.1: Current idealistic radar sensor model in CSPAS constructed with two ideal radar models.

Recently, real-world testing has been conducted at VCC where data from the FLR sensor has been collected. This enables data-driven radar sensor models to be developed based on the collected data as a more realistic alternative to the current FLR model in CSPAS. The model needs to be deterministic to ensure that a simulated scenario with specific initial conditions generates the same result for repeated simulations.

1.2 Contributions

This thesis improves the CSPAS framework by designing a more realistic radar sensor model. In order to improve the model some realistic radar properties are designed, such as a new FOV to more accurately represent the real FLR view, radar measurement error, and detection rate. This is done by using collected data from a real FLR sensor to design data-driven models. Three new FOV models are designed to more accurately represent the visual field of the real radar. The FOV is modeled with polygons, one convex and one concave, as well as one polynomial model constructed

of three polynomials. To achieve a deterministic model both the measurement error model and detection rate model are based on number generation with seed to achieve a pseudo-random distribution. When a specific number combination, also known as a seed, is used as input to a model the output should always be the same for the same seed, when changing the seed the output also changes.

The models are evaluated on their accuracy, execution time, and repeatability. If more than 95 % of the collected data points are inside the FOV, the model is deemed as accurate. However, while including as many data points as possible in the new shapes, they should not be overestimating the FOV by including too much area outside of the collected data points. The models are evaluated on their execution time compared to the idealistic model, however, since there is no official speed requirement from VCC, this is not a strict value, but rather a comparison between the new and idealistic models. The model is evaluated on its repeatability by executing a scenario in the simulation and comparing the result of each execution.

1.3 Related Work

Several projects have previously been made on the topic of designing radar sensor models in virtual environments. In [1], a radar sensor model was designed, implemented, and validated to be used for ADAS/Autonomous Drive (AD) functions. The simulation environment was built on an Open Simulation Interface (OSI) with GroundTruth references including, among others, class definitions, correct object positions, and bounding boxes.

They introduce three definitions of sensor models in a virtual environment: *ideal*, *probabilistic*, and *physical*. *Ideal* sensor models, typically used in two-dimensional (2D) environments, recognize all objects in the FOV without errors. *Probabilistic* sensors, mostly used in 2D environments, establish a probabilistic relationship between sensor output and GroundTruth such that sensor effects, error, and environmental conditions are modeled by means of observations and statistics. *Physical* sensors, often used in 3D environments, are based on physical laws such as electromagnetic wave propagation, interference, and diffraction.

Based on these definitions mentioned in the previous paragraph, [1] consider their model ideal with additional features to simulate sensor effects via probabilistic relationships. The model uses GroundTruth references as input while the output is a sensor-specific object list that includes, for example, class definition, and object position (x, y). The output is sent to ADAS/AD functions to influence vehicle dynamics, such as acceleration, braking, and steering. This is fed back to the virtual environment and a new loop will initiate.

To be able to use the sensor model a coordinate transformation was needed between the virtual environment global coordinate system and the sensor coordinate system. This was further discussed by [2] who used the mounting placement of the sensor

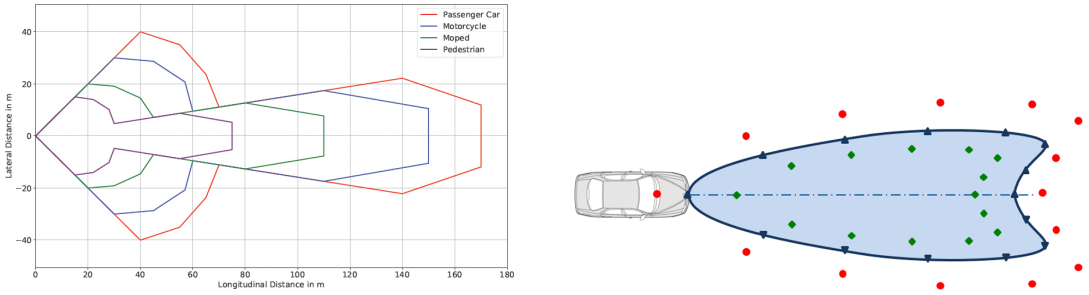
(s_x, s_y) in the global coordinate system as well as the mounting orientation φ

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix}_{\text{sensor}} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \left(\begin{bmatrix} d_x \\ d_y \end{bmatrix}_{\text{global}} - \begin{bmatrix} s_x \\ s_y \end{bmatrix}_{\text{global}} \right) \tag{1.1}$$

and applied a rotation and translation in order to calculate the placement of the car (d_x, d_y) in sensor coordinates.

In [1] they argue that modeling the FOV as a 2D polygon would be better than a circular segment as this allowed to vary the FOV shape based on object dependency, see Figure 1.2a. They further reference [2] that created the FOV with a set of points x_i from the data to make the circle sector border, see Figure 1.2b. Calculations to create the FOV used in [2] were made from an inverse matrix, and the FOV could be adjusted by tuning a parameter σ to adapt it for various target objects. However, according to [1], this resulted in an unrealistic and computationally complex model.

The sensor model in [1] uses the MATPLOTLIB module ***path*** and a set of points (x_i, y_i) to define a polygon shape. Using data from GroundTruth references as input into a function, a boolean output describes if an object is inside the FOV. They specify that if the center of an object is within the FOV, it is detected by the radar.



(a) FOV as polygon from [1]. **(b)** FOV as circle segment, from [2].

Figure 1.2: FOV represented as (a) a polygon with object dependency and (b) a circle segments from a set of points. Figures from [1], [2].

1.4 Compiled Simulation Platform for Active Safety

Similar to the virtual simulation used in [1], CPSAS follows the Open Simulation Interface (OSI) specification to ensure compatibility between the ADAS function and any simulator [9]. The OSI also provides structures where information about the virtual environment is stored. One such structure is the **GroundTruth** reference which is used as input to the sensor model in CSPAS, while the output is formatted according to the **SensorData** reference, see Figure 1.3.

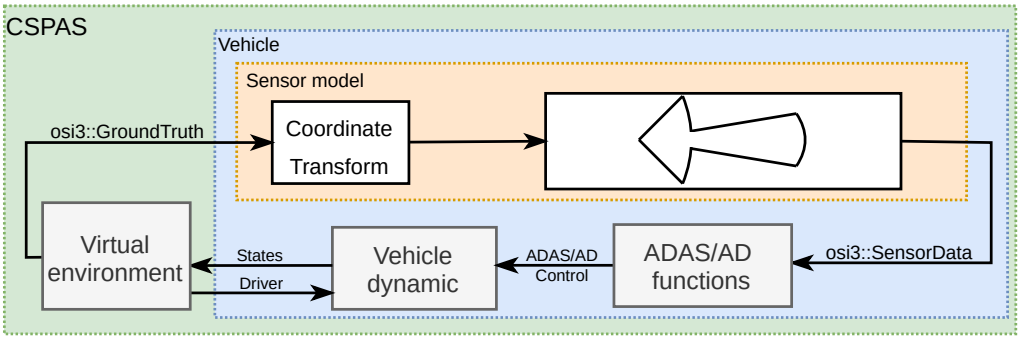


Figure 1.3: CSPAS closed-loop flowchart, coordinates are transformed from global to sensor coordinates before being sent to the FOV model.

The **GroundTruth** contains information about the simulated environment with regard to the global coordinate system (x_g, y_g) [10]. This includes information about the host vehicle, roads, lanes, objects, etc. The sensor model uses information stored in the **MovingObject** reference which refers to all moving objects in the simulated environment such as cars, motorcycles, and pedestrians. The structure of the **MovingObject** reference is illustrated in Figure 1.4.

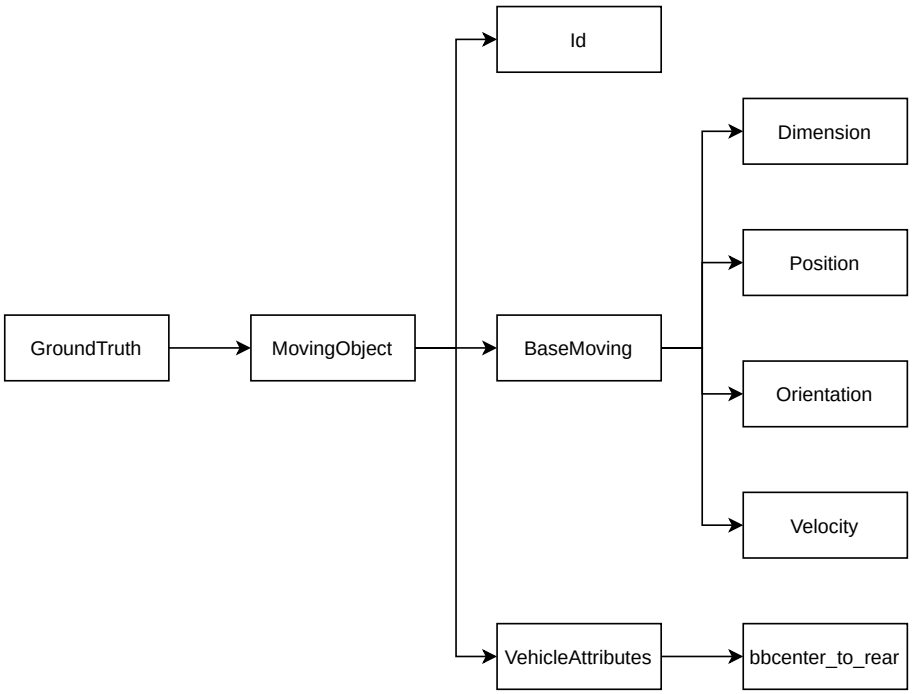


Figure 1.4: CSPAS GroundTruth references of interest.

The **Id** reference indicates which object the **MovingObject** data is referring to. All objects in the simulation are constructed as bounding boxes with a length (L) and width (W) which are included in the **Dimension** reference, see Figure 1.5.

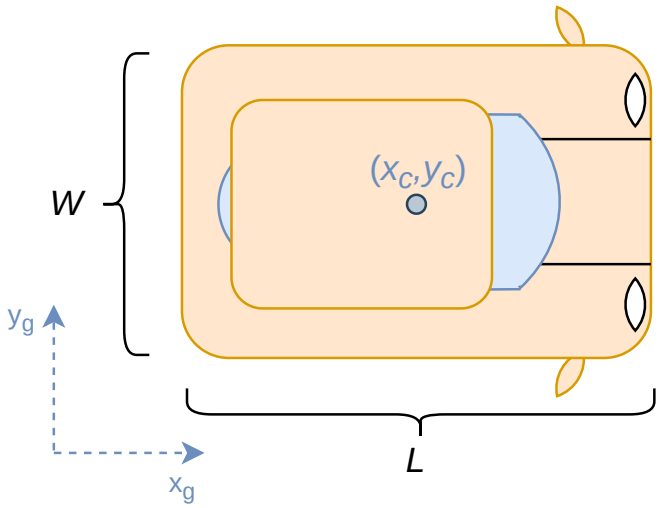


Figure 1.5: The bounding box of an object in CSPAS.

The **Position** reference refers to the center position of the bounding box (x_c, y_c) and the **Orientation** reference includes the center position's yaw orientation (α) around the z-axis. The **Velocity** reference includes the velocity of an object in x and y components (v_x, v_y). To get the origin of the vehicle coordinate system (x_v, y_v) the **bbcenter_to_rear** vector from the **VehicleAttribute** reference is used, see Figure 1.6.

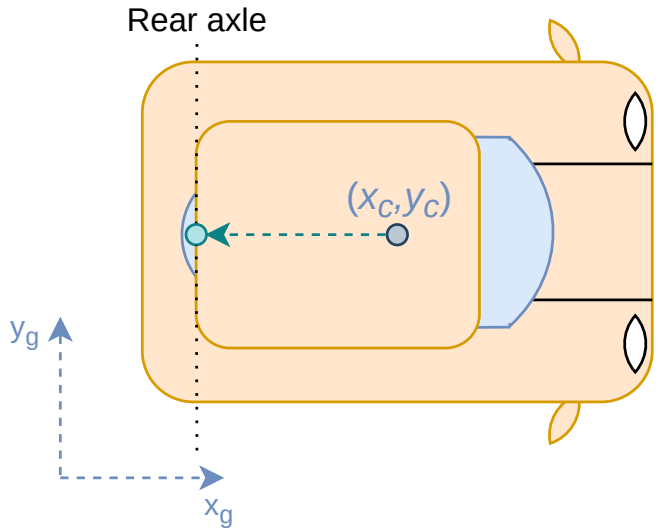


Figure 1.6: The **bbcenter_to_rear** vector (green) which gives the origin of the vehicle coordinate system (x_v, y_v).

The output from the sensor model is made according to the **SensorData** reference which includes information about the environment with regard to the sensor coordinate system (x_s, y_s) [11]. The **MountingPosition** represents the mounting

Position and **Orientation** (β) of the radar sensor with regard to the vehicle coordinate system. The mounting position is the origin of the sensor coordinate system which the **DetectedMovingObject** reference is with regard to.

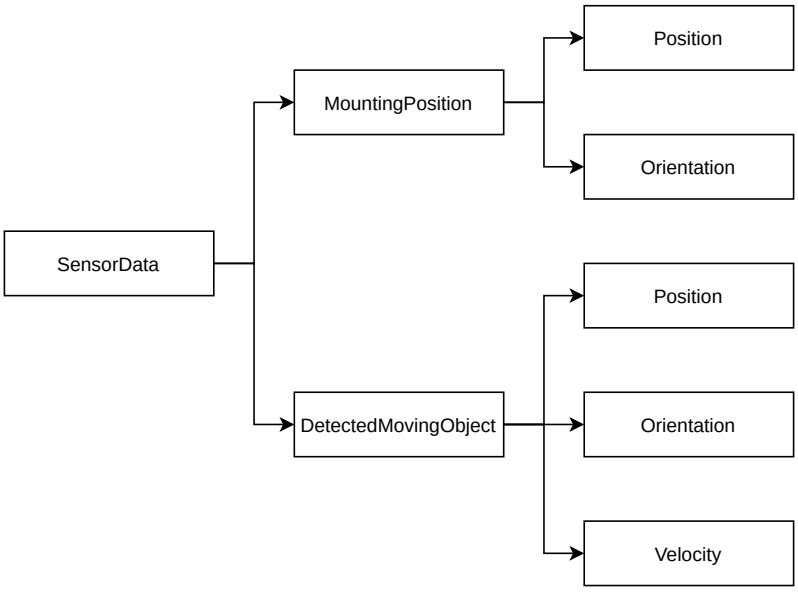


Figure 1.7: CSPAS SensorData references of interest.

The **DetectedMovingObject** reference includes the **Position**, **Orientation**, and **Velocity** of the moving objects detected by the radar in sensor coordinates. In Figure 1.8 all three coordinate systems are illustrated.

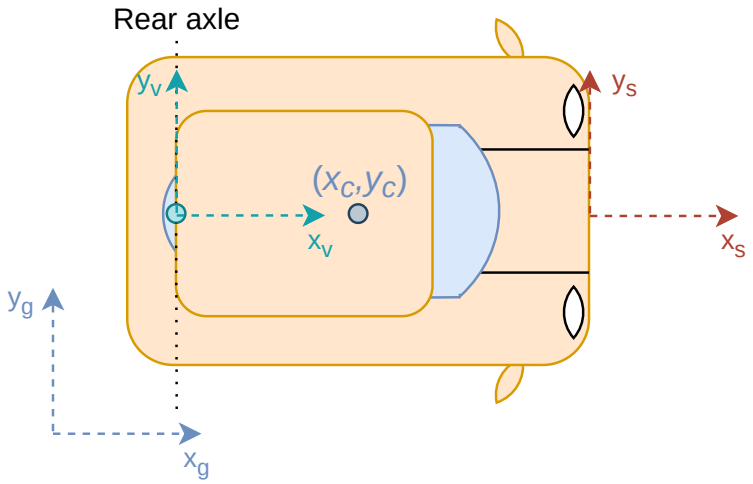


Figure 1.8: The global (x_g, y_g), vehicle (x_v, y_v), and sensor (x_s, y_s) coordinate system in CSPAS.

With the **GroundTruth** and **SensorData** structures in mind, details within the sensor model can be further investigated. The virtual environment creates the

GroundTruth reference which is used as input to the sensor model, see Figure 1.3. An object is deemed detected if two corners of the object bounding box are within the FOV of the radar sensor. Given the center position (x_c, y_c) of an object from the **GroundTruth** reference, the corners are calculated using the length (L) and width (W) from the **Dimension** reference as well as the **Orientation** angle (α)

$$\begin{aligned} x_{fr} &= x_c + \tfrac{L}{2}\cos\alpha + \tfrac{W}{2}\sin\alpha & y_{fr} &= y_c + \tfrac{L}{2}\sin\alpha - \tfrac{W}{2}\cos\alpha \\ x_{rr} &= x_c - \tfrac{L}{2}\cos\alpha + \tfrac{W}{2}\sin\alpha & y_{rr} &= y_c - \tfrac{L}{2}\sin\alpha - \tfrac{W}{2}\cos\alpha \\ x_{rl} &= x_c - \tfrac{L}{2}\cos\alpha - \tfrac{W}{2}\sin\alpha & y_{rl} &= y_c - \tfrac{L}{2}\sin\alpha + \tfrac{W}{2}\cos\alpha \\ x_{fl} &= x_c + \tfrac{L}{2}\cos\alpha - \tfrac{W}{2}\sin\alpha & y_{fl} &= y_c + \tfrac{L}{2}\sin\alpha + \tfrac{W}{2}\cos\alpha. \end{aligned} \tag{1.2}$$

An illustration of the placement of the corners can be seen in Figure 1.9.

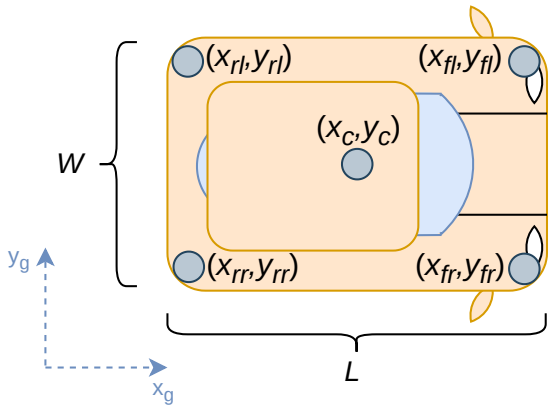


Figure 1.9: The corners are labeled according to their placement on the car, front (f) or rear (r) and right (r) or left (l). Given that the car is traveling along the x-axis, the orientation yaw angle $\alpha = 0$.

The GroundTruth reference is transformed to sensor coordinates (x_s, y_s) by the sensor position (x_{sp}, y_{sp}) and yaw orientation of the sensor coordinate system with regard to the global coordinate system (β)

$$\begin{aligned} x_s &= (x_g - x_{sp}) \cdot \cos(\beta) + (y_g - y_{sp}) \cdot \sin(\beta) \\ y_s &= (y_g - y_{sp}) \cdot \cos(\beta) - (x_g - x_{sp}) \cdot \sin(\beta) \end{aligned} \tag{1.3}$$

before being used as input to the sensor model. To determine if the corners are within the FOV the azimuth angle (θ) and range (R) to the corner points are calculated and compared to the azimuth angle and radius range of the idealistic FOV circle segments shown in Figure 1.1. If the azimuth angle and range to the point are smaller than the azimuth angle and range of either circle sector the point is inside the FOV, see Algorithm 1.

Algorithm 1: Determining if a point is inside the FOV for the idealistic model

Input: fov_check = 0
Input: detection = False
Data: p[i] = (x_i, y_i) $\forall i \in [0, C)$; /* Corner points in sensor coordinates */
for $(i := 0; i < C; i \mathrel{+}= 1)$ **do**
  $R_i = \sqrt{p[i].x^2 + p[i].y^2}$; /* Convert to polar coordinates */
  $\theta_i = \arctan\left(\frac{p[i].y}{p[i].x}\right)$;
  if $(|\theta_i| \leq \theta_{SRR}$ ***and*** $R_i \leq R_{SRR})$ ***or*** $(|\theta_i| \leq \theta_{LRR}$ ***and*** $R_i \leq R_{LRR})$ **then**
    fov_check += 1 ; /* Point is inside */
  end
end
if *fov_check* ≥ 2 **then**
  detection = True ; /* Object is inside */
end

However, this algorithm is limited to a FOV designed as circle sectors. The output from the sensor model is a **SensorData** structure which is used as input to the ADAS functions, similar to [1]. If an object has been detected by the radar, the ADAS functions apply control to the vehicle dynamics such as breaking or steering to avoid collision with the obstacle.

1.5 Thesis Outline

The thesis is organized as follows. Chapter 2 presents the theoretical background of radars as well as algorithms to solve the point in polygon problem and the equivalent theory needed for the point in polynomial solution. Chapter 3 presents the collection of measurement data, the data structure, pre-filtering, and an analysis of the collected data. Using the analysis of the collected data, chapter 4 discusses the design of the radar sensor models. The models to be designed consists of the FOV, measurement error, and detection rate models. Chapter 5 presents the implementation of each model into CSPAS. Chapter 6 presents the evaluation according to the evaluation requirements presented in Section 1.2 Contributions. Chapter 7 includes a discussion of the results from the previous chapters. Chapter 8 presents the conclusions drawn from the result and evaluation, the social, ethical, and ecological aspects, as well as what can be done in future work.

2

Theory

This chapter introduces the theoretical background of radars as well as the theory within the point in polygon problem and the equivalent for polynomials.

2.1 Radar Basics

Radio Detection and Ranging sensors, abbreviated to radars, can measure the range (R), radial velocity (v_r), and azimuth angle (θ) to an object [12], see Figure 2.1. Radars emit electromagnetic (EM) waves that are reflected back when they encounter an object [13]. Therefore, they work under any condition, unlike LiDAR and cameras whose performance might be affected by environmental factors such as rain, fog, and snow [13]. Radars are therefore very suitable to use for surrounding sensing technologies and have been used in various car applications since the early '90s for functions such as CW and Adaptive Cruise Control (ACC) [14].

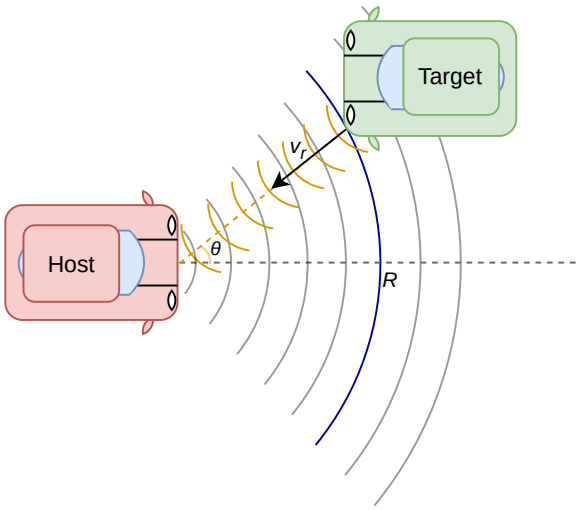


Figure 2.1: Radar emits EM waves (gray) that when reflected against an object (orange) can measure range (R), radial velocity (v_r), and azimuth angle (θ).

There are various kinds of radar sensors, the most common in the automotive industry is the Frequency-Modulated Continuous Wave (FMCW) radar since they radiate continuous power [13]. This improves the detectability of obstacles at small ranges

and ensures the target's range and relative velocity can be measured simultaneously [13]. The radar investigated in this thesis is the FLR sensor which is an FMCW radar operating with a signal frequency of 77 GHz.

2.2 Point in Field Of View

As previously presented in Section 1.4 Compiled Simulation Platform for Active Safety, an object is detected when two corners of the bounding box are inside the FOV. For the current idealistic FOV constructed as two circle sectors, as shown in Figure 1.1, the algorithm for determining if a corner point is inside the FOV is rather simplistic, see Algorithm 1. The algorithm checks if the corner points polar coordinates (R, θ) are smaller than either of the radius and azimuth angles of the circle sectors. This algorithm for determining if the object is inside the FOV only works for circle sectors. With the FOV designed as polygons and polynomial models, new algorithms for determining if an object is inside the FOV need to be designed.

2.2.1 Point In Polygon

With the FOV designed as polygons, the algorithm to determine if a point is inside the FOV shape is commonly known as the Point In Polygon (PIP) problem. In [1] the PIP problem was solved with a ray-casting algorithm. The algorithm is also known as ray tracing, ray intersection, and even-odd method. Ray-casting consists of counting the number of times a ray intersects with a polygon vertex when starting at the point of interest and going along the x-axis to infinity, see Figure 2.2. If the count is an odd number the point is inside the FOV, while if it is an even number it is outside the FOV. To ensure that zero is counted as an even number, thereby meaning outside the FOV, the even and odd count is performed using modulo two.

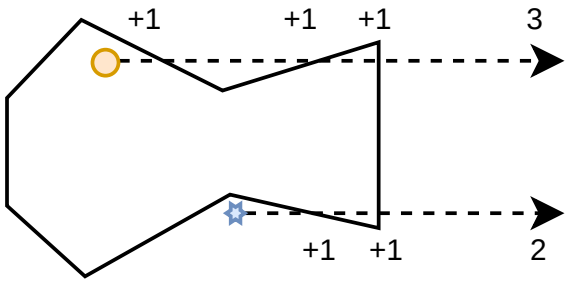


Figure 2.2: Ray-casting for the Point In Polygon problem. The orange point is inside the FOV with an odd ray-casting number while the blue start is outside with an even number.

Assuming a polygon has N_v vertices (x_i, y_i) for $i \in 0, ..., N_v - 1$ where $(x_{N_v}, y_{N_v}) = (x_0, y_0)$ to form a closed polygon. The ray casting method evaluates if the ray intersects with the vertices of the polygon, thereby the time complexity is $\mathcal{O}(N_v)$. Generally, ray-casting is the best alternative to solve the PIP problem. However, for points close to the polygon edge it might not be accurate. Other methods are more accurate for these scenarios, such as the winding number algorithm [15].

The winding number algorithm calculates the number of revolutions made around the point of interest while following the trajectory of the polygon, see Figure 2.3. If the number of revolutions is zero then the point is outside the FOV, any other whole number ensures the point is inside the FOV.

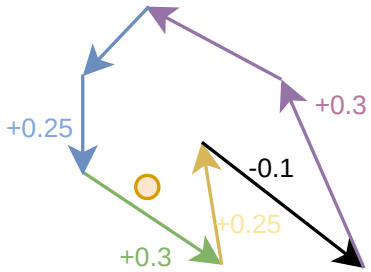


Figure 2.3: Winding number algorithm, the number signifies how much of a revolution around the point has been made, the orange point has winding number 1.

The winding number can be applied in many ways, one way is to calculate the sum of the angles between the point and all polygon vertices [16]. If the sum is equal to 2π the point is inside the FOV, if the sum is zero it is outside. Similar to ray-casting, this method evaluates all vertices of the polygon, meaning a time complexity of $\mathcal{O}(N_v)$. However, since angle calculations include trigonometric functions it is quite computationally complex [15]. Angle summation is also sensitive to truncation and rounding errors. Therefore it is not suitable for polygons with many vertices [17].

A quite simple solution to the PIP problem is to calculate the determinant between the vector of the polygon edges and the vector between the polygon vertices and the point [18]. This method is commonly known as the Sign-Off method. Assuming an anti-clockwise orientation of the polygon edge vectors, if the point lies to the left of all polygon edges it is inside the FOV. To verify that the point is to the left of all edges, the determinant of the polygon edge vector and point to polygon vertex vector is calculated. By defining the point to vertices vector with the arrowhead towards the point, see Figure 2.4, if the determinant is positive it is inside the FOV. This method also has a time complexity $\mathcal{O}(N_v)$, however, it is not sensitive to points close to the edge or on the edge as this results in the determinant being close to respectively equal to zero. This method only works for convex polygons.

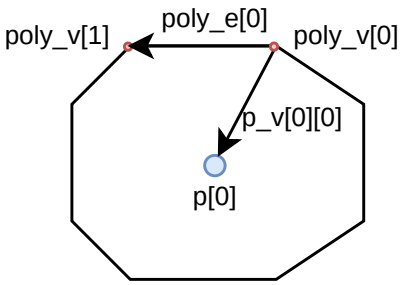


Figure 2.4: Sign-Off method where poly_v are the polygon vertices, poly_e is the polygon edge vector, and p_v is the vector between a vertex and point p.

2.2.1.1 Point In Convex Polygon

The Sign-Off method is used for the convex polygon, see Algorithm 2.

Algorithm 2: Determining if a point is inside convex FOV – Sign-Off method

Input: sum_dot[i] $= 0\ \forall i \in [0, C)$
Input: fov_check $= 0$
Input: detection = False
Data: p[i] $= (x_i, y_i)\ \forall i \in [0, C)$; `/* Corner points in sensor coordinates */`
Data: poly_v[i] $= (x_i, y_i)\ \forall i \in [0, N_v)$; `/* Polygon vertices */`
Data: poly_e[i] $=$ poly_v[$i+1$] $-$ poly_v[i] $\forall i \in [0, N_v - 1)$,
poly_e[$N_v - 1$] $=$ poly_v[0] $-$ poly_v[N_v] ; `/* Polygon edges */`

```
for (c := 0; c < C; c += 1) do
    for (n := 0; n < N_v; n += 1) do
        p_v[c][n] = ((p[c].x - poly_v[n].x), (p[c].y - poly_v[n].y));
        det[c][n] = poly_e[n].x · p_v[c][n].y - poly_e[n].y · p_v[c][n].x;
        if det[c][n] ≥ 0 then
            sum_dot[c] += 1;
            if sum_dot[c] == N_v then
                fov_check += 1 ;                    /* Point is inside */
            end
        end
    end
end
if fov_check ≥ 2 then
    detection = True ;                              /* Object is inside */
end
```

By combining the ray-casting and the Sign-Off method, one can use a similar strategy for concave polygons. Ray-casting is applied, however, now starting at minus infinity and going through the point of interest to infinity along the x-axis. The ray-casting checks if any vertices intersect with the ray to then apply the Sign-Off method to calculate the determinant for all vertices that intersect. If the determinant is equal to zero or positive the counter is increased by one, if the determinant is negative the counter is subtracted by one. If the final counter has a positive value, the point is inside the FOV. This method has time complexity $\mathcal{O}(N_v)$ and works for all polygons.

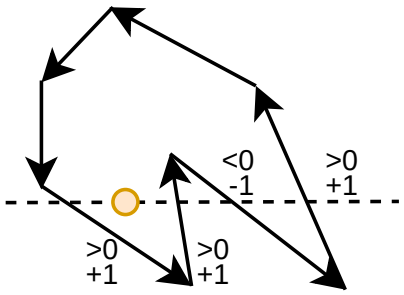


Figure 2.5: Combining ray-casting and Sign-Off.

2.2.1.2 Point In Concave Polygon

The algorithm chosen for the concave polygon was the combined ray-casting and Sign-Off method. From [19] the ray-casting algorithm is performed using the following equations. Given that a vector $\overrightarrow{v}$ can be described by two points P_1, P_2

$$\overrightarrow{v} = P_2 - P_1 \tag{2.1}$$

and a line described as a point P_1 with a vector $\overrightarrow{v}$

$$l_1 = P_1 + t\overrightarrow{v} \tag{2.2}$$

were $t \in [0, 1]$ is the interval where the line resides. If the cross product between the two vectors is zero

$$\overrightarrow{v} \times \overrightarrow{u} = 0 \tag{2.3}$$

the lines are parallel and have no intersection. The intersection of two lines is otherwise calculated as

$$\begin{aligned}
P_1 + t\overrightarrow{v} &= Q_1 + s\overrightarrow{u} \\
t\overrightarrow{v} &= (Q_1 - P_1) + s\overrightarrow{u} \\
t\overrightarrow{v} \times \overrightarrow{u} &= (Q_1 - P_1) \times \overrightarrow{u} + s\overrightarrow{u} \times \overrightarrow{u} \\
t\overrightarrow{v} \times \overrightarrow{u} &= (Q_1 - P_1) \times \overrightarrow{u} \qquad \text{[19]} \\
t(\overrightarrow{v} \times \overrightarrow{u}) \cdot (\overrightarrow{v} \times \overrightarrow{u}) &= (Q_1 - P_1) \times \overrightarrow{u} \cdot (\overrightarrow{v} \times \overrightarrow{u}) \\
t &= \frac{(Q_1 - P_1) \times \overrightarrow{u} \cdot (\overrightarrow{v} \times \overrightarrow{u})}{(\overrightarrow{v} \times \overrightarrow{u}) \cdot (\overrightarrow{v} \times \overrightarrow{u})}
\end{aligned} \tag{2.4}$$

if $t \in [0, 1]$ then l_2 intersects l_1 at point $P_1 + t\overrightarrow{v}$. If $t \notin [0, 1]$ then l_1 would intersect l_2 if l_1 was longer. Therefore the polygon vertices are defined as l_1 and the ray as l_2 and the ray-tracing is crossing the polygon vertices if $t \in [0, 1]$, see Figure 2.6.

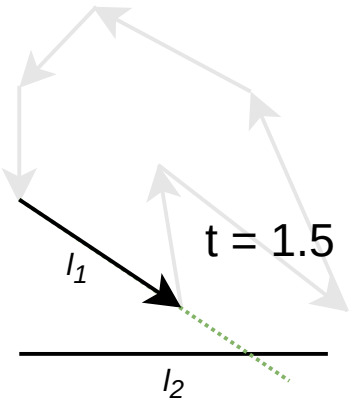


Figure 2.6: Defining the polygon vertex as l_1 as the ray tracing line as l_2 where t signifies at what position of l_1 the intersection occurs, if $t \notin [0, 1]$ then l_1 needs to be longer for intersection to occur with l_2.

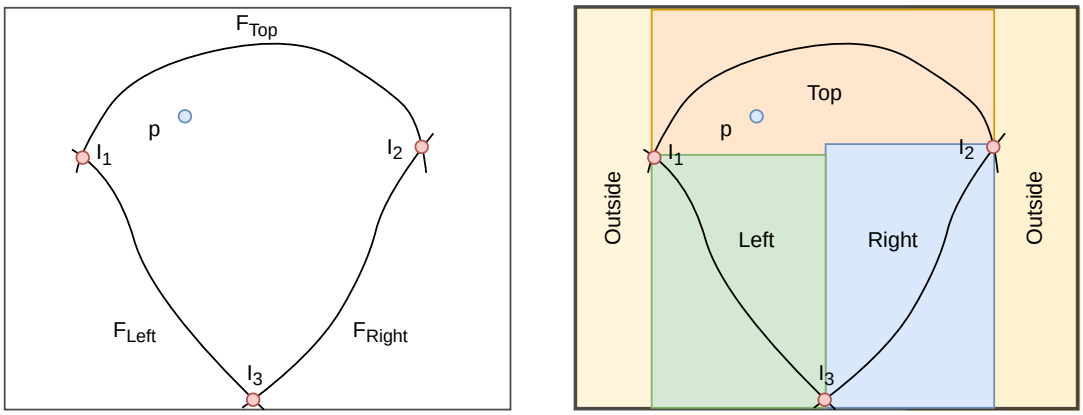
Algorithm 3: Determining if a point is inside concave FOV – ray-tracing and Sign-Off method

Input: start = 0, stop > R_{LRR}
Input: counter[i] = 0 $\forall i \in [0, C)$
Input: fov_check = 0
Input: detection = False
Data: p[i] = (x_i, y_i) $\forall i \in [0, C)$; `/* Corner point in sensor coordinates */`
Data: poly_v[i] = (x_i, y_i) $\forall i \in [0, N_v)$; `/* Polygon vertices */`
Data: poly_e[i] = poly_v[$i+1$] − poly_v[i] $\forall i \in [0, N_v - 1)$,
poly_e[$N_v - 1$] = poly_v[0] − poly_v[N_v] ; `/* Polygon edges */`

```
for (c := 0; c < C; c += 1) do
    ray[c] = ((start,p[c].y),(stop, p[c].y)) ;          /* Ray casting (Q1,Q2) */
    ray_v[c] = (stop, 0) ;                              /* Ray casting vector (u⃗) */
    for (n := 0; n < Nv; n += 1) do
        diff = ray[c].p1 - poly_v[n] ;                  /* Q1 - P1 */
        cross = poly_e[n].x · ray_v[c].y - ray_v[c].x · poly_e[n].y ; /* v⃗ × u⃗ */
        if cross != 0 then
            t = (diff ×u⃗· cross)/(cross · cross);
            if 0 ≤ t and t ≤ 1 then
                det = poly_e[n].x · p_v[c][n].y - p_v[c][n].x · poly_e[n].y ;
                if det ≥ 0 then
                    counter[c] += 1;
                else
                    counter[c] -= 1;
                end
            end
        end
    end
    if counter[c] > 0 then
        fov_check += 1 ;                                /* Point is inside */
    end
end
if fov_check ≥ 2 then
    detection = True ;                                  /* Object is inside */
end
```

2.2.2 Point In Polynomial

Similarly to the PIP problem for the polygon shapes, an algorithm for the polynomial FOV model is introduced. This is done by considering the intersections points $I \in \{I_1, I_2, I_3\}$ of the polynomial functions $F \in \{F_{Top}, F_{Left}, F_{Right}\}$, see Figure 2.7.



(a) Polynomials and intersection points.

(b) Validity domain of each polynomial.

Figure 2.7: Set up of polynomials FOV.

Constructing domains for each polynomial function, a point p in sensor coordinates is paired with a domain. The domain check is done with the following cases.

$$
\begin{array}{lllll}
p.x \in [\infty, I_1.x) \textbf{ or } (I_2.x, -\infty] & & \rightarrow & p \text{ outside} \\
p.x \in [I_1.x, I_3.x) \textbf{ and } p.y \in [I_3.y, I_1.y) & & \rightarrow & p \text{ in } F_{Left} \\
p.x \in [I_1.x, I_3.x) \textbf{ and } p.y \in [I_1.y, \infty] & & \rightarrow & p \text{ in } F_{Top} \\
p.x \in [I_3.x, I_2.x] \textbf{ and } p.y \in [I_2.y, \infty] & & \rightarrow & p \text{ in } F_{Top} \\
p.x \in [I_3.x, I_2.x] \textbf{ and } p.y \in [I_3.y, I_2.y) & & \rightarrow & p \text{ in } F_{Right}
\end{array}
$$

Given the polynomial domain of the point, the polynomial function is used to calculate $F(p.x)$ which is compared with $p.y$ to evaluate if p falls within the FOV boundary. The following cases are shown below.

$$
\begin{array}{llll}
p \text{ in } F_{Left} & p.y \geq F(p.x) & \rightarrow & \text{inside FOV} \\
p \text{ in } F_{Right} & p.y \geq F(p.x) & \rightarrow & \text{inside FOV} \\
p \text{ in } F_{Top} & p.y \leq F(p.x) & \rightarrow & \text{inside FOV}
\end{array}
$$

The theoretical computational complexity of evaluating a polynomial is $\mathcal{O}(s^2)$ where s is the degree of the polynomial. Algorithm 4 shows the method used for determining if a point is inside the polynomial FOV.

Algorithm 4: Is point in polynomials FOV

```
Data: p[i] = (x_i, y_i) ∀i ∈ [0, C)        /* Corner point in sensor coordinates */
Data: F = {F_Top, F_Left, F_Right}          /* Polynomial functions */
Data: I = [I_1, I_2, I_3]                   /* Intersection points list */
Function Get_domain(p, I):
    if p.x > I_1.x and p.x < I_2.x then
        return Outside
    else if p.x ≥ I_3.x and p.y ≤ I_1.y then
        return Left
    else if p.x ≤ I_3.x and p.y ≥ I_2.y then
        return Right
    else
        return Top
    end
for (i := 0; i < C; i += 1) do
    Domain = Get_domain(p[i], I)
    switch Domain do
        case Outside do
            nothing
        end
        case Top do
            if p[i].y ≤ F_Top(p[i].x) then
                fov_check += 1
            end
        end
        case Left do
            if p[i].y ≥ F_Left(p[i].x) then
                fov_check += 1
            end
        end
        case Right do
            if p[i].y ≥ F_Right(p[i].x) then
                fov_check += 1
            end
        end
    end
end
if fov_check ≥ 2 then
    detection = true
end
```

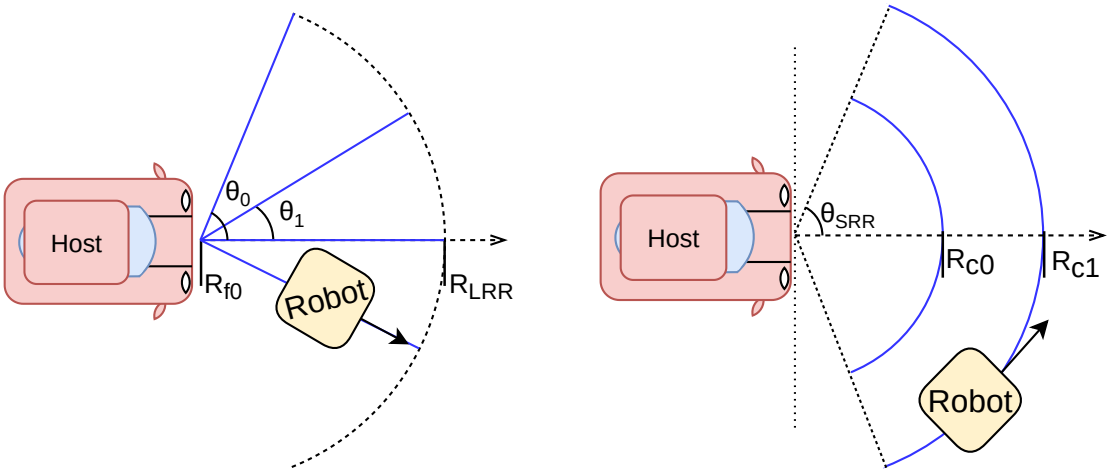
3

Collection and Analysis of Data Measurements

In this chapter, the collection of data measurements are presented along with the collected data, the data structure, the pre-filtering, and an analysis of the data measurements.

3.1 Collection of Data Measurements

The collection of measurement data was conducted by the VCC radar team by performing two tests on the real FLR sensor, according to specifications from the VCC internal tool *SystemWeaver*, [20], [21]. The two tests performed can be seen in Figure 3.1 below, where the blue line is the pedestrian child robot path. Both tests were conducted similarly. The host vehicle equipped with the FLR sensor is placed stationary at a known location while the robot with a mounted doll, to represent a child pedestrian, is driving in a fan and circle pattern respectively.



(a) Fan test pattern. **(b)** Circle test pattern.

Figure 3.1: Tests performed in the a) fan pattern and b) circle pattern according to *SystemWeaver* specification.

In the fan test, see Figure 3.1a, the robot starts at R_{f0} meters from the host car and drives radially away from the car until the distance R_{LRR} is surpassed. This is done for n angles between the azimuth angles $\pm\theta_{SRR}$ with the starting angle $\theta_0 = \theta_{SRR}$ decreasing with $\Delta\theta = 0.1\theta_{SRR}$

$$\theta_n = \theta_{SRR} - n\Delta\theta \tag{3.1}$$

until azimuth angle $-\theta_{SRR}$ is surpassed.

In the circle test, see Figure 3.1b, the robot trajectory drives with radius R_{c0} from the host car between the azimuth angles $\pm\theta_{SRR}$. This was done for m ranges with the distance increasing with $\Delta R = 0.05R_{LRR}$

$$R_{cm} = R_{c0} + m\Delta R \tag{3.2}$$

until radius R_{LRR} is surpassed.

This gives a testing grid according to Figure 3.2 consisting of 21 angles for the fan test and 19 range intervals for the circle test.

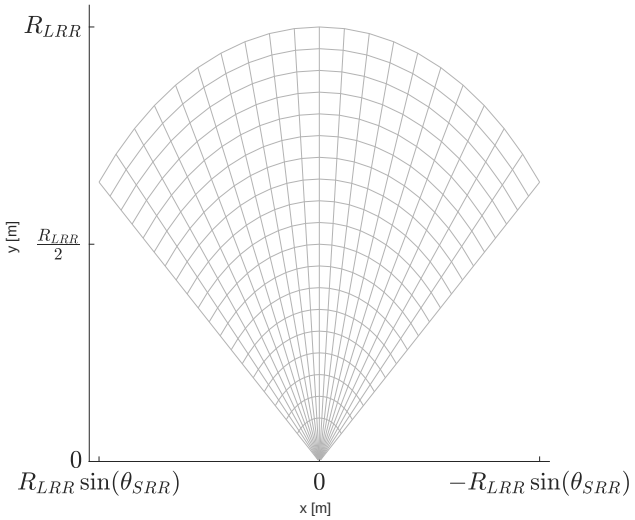


Figure 3.2: Testing grid constructed from the tests.

3.1.1 Data Structure

The collected data from both tests are stored in two subsets, data from the FLR sensor on the test car, and data from the robot child pedestrian. Both datasets have for each sample a timestamp (ts). The FLR sensor also logs the measured azimuth angle (θ) and range (R) to the robot while the robot logs its GPS position (x_r, y_r), see Table 3.1.

Table 3.1: Data parameters collected from the Forward Looking Radar and robot.

FLR		Robot	
Timestamp	(ts)	Timestamp	(ts)
Range	(R)	GPS latitude	(x_r)
Azimuth angle	(θ)	GPS longitude	(y_r)

The FLR sensor measured position was converted from the polar coordinates, range R and azimuth angle θ, to Cartesian coordinates

$$\begin{aligned} x_{FLR} &= R \cdot \sin\theta \\ y_{FLR} &= R \cdot \cos\theta \end{aligned} \tag{3.3}$$

to have the same coordinate system as the GPS position.

3.1.2 Pre-filtering of the Collected Data Measurements

As previously presented in Section 2.1 Radar Basics, the radar emits EM waves that when meeting an object are reflected back to the radar. At every sample, the FLR sensor can log up to 400 received data measurements. However, not all are detected objects. When the FLR sensor does not detect any object, i.e. no EM waves are reflected back to the antenna, the range is stored as a default value of $4R_{LRR}$. Knowing that the robot only travels up to a range of R_{LRR} it is safe to assume that data measurements with a range of $4R_{LRR}$ are not of the robot. By applying a filter to remove these large range values the samples kept are of the detect robot.

3.2 Analysis of Data Measurements

The number of FLR detection samples after pre-filtering is $N_s = 42125$, and can be seen in Figure 3.3 (green) along with the current ideal model in CSPAS (red) and the robot trajectory (blue). From this plot, one can see the importance of implementing a new FOV for the virtual simulation environment. The new FOV will incorporate the detected area (green) that is partially outside the ideal FOV.

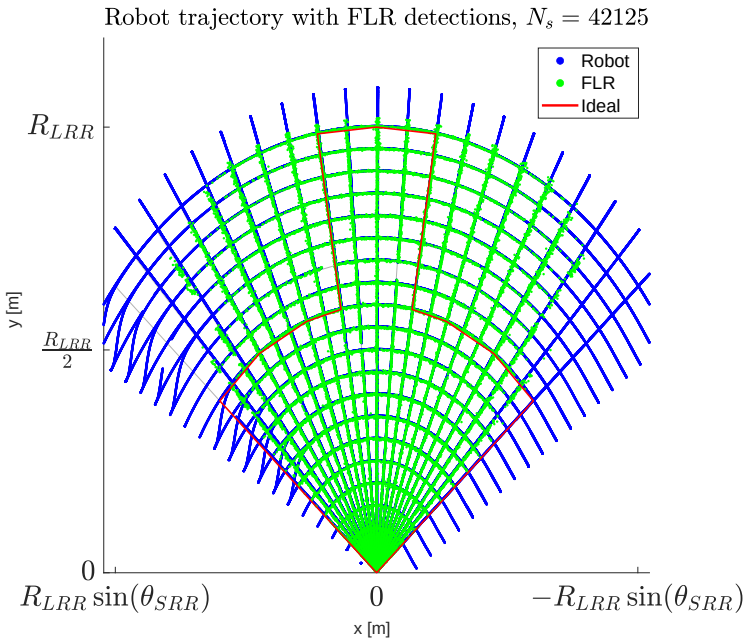


Figure 3.3: Robot trajectory (blue) along with the detections from the FLR (green) and the current ideal sensor in CSPAS (red).

3.2.1 Analysis of the Field Of View

The collected FLR data measurements show that detections are not made in the outskirts of the testing grid, see Figure 3.4.

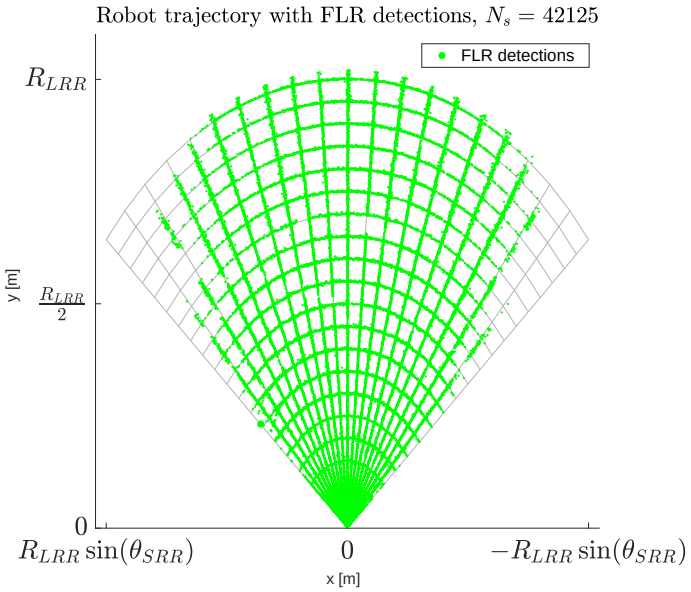


Figure 3.4: Testing grid (gray) and FLR detections (green).

From this one could argue what shape would be most suitable for the FOV. A polygon, used in [1], is described by a finite number of points, also called vertices, to form straight line segments, also called edges, which are closed to form a polygon chain. A polygon can be concave or convex, see Figure 3.5.

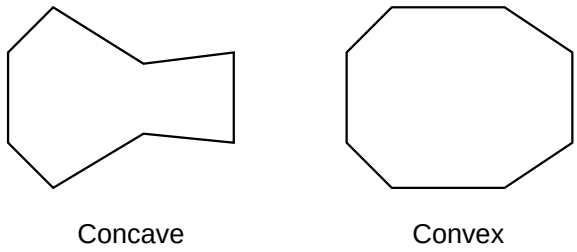


Figure 3.5: Polygon representation as concave and convex.

By more closely investigating the FLR detections it can be observed that there are some intervals within the FOV where detections are not made, see Figure 3.6.

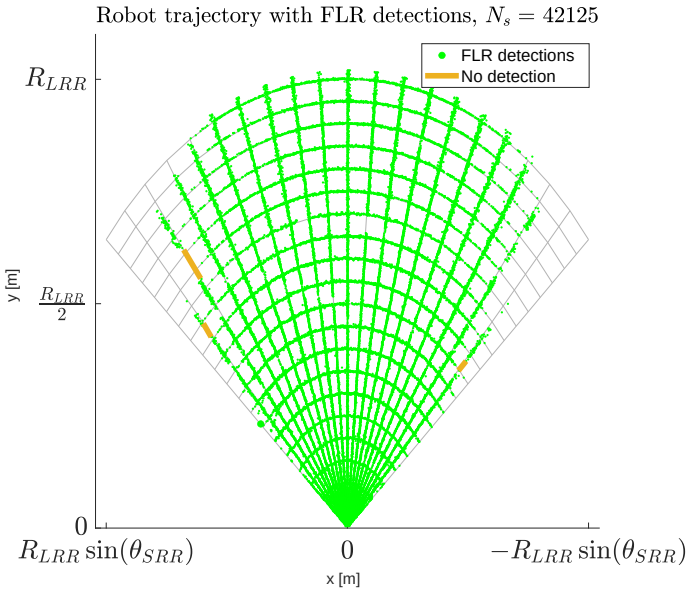


Figure 3.6: The FLR detections (green) and no detection intervals (orange).

Therefore it might be suitable to model the FOV as a concave polygon as this could more accurately shape the observed topology by excluding the intervals, shown in Figure 3.6, where detections are not made. However, a concave model will most likely consist of more vertices compared to a convex model, thus making it more computationally complex as presented in Section 2.2.1 Point In Polygon. Designing the FOV as a convex polygon might therefore be more optimal from an execution time standpoint but will also result in a less accurate model by including areas where detections have not been made. Overall challenges with the polygon approach involve finding appropriate points to design the boundary, this is further discussed in Section 4.1 Design of the Field Of View Models.

A polynomial approach might more accurately represent the observed topology, however, it could also be more computationally complex, as presented in Section 2.2.2 Point In Polynomial. A challenge with this approach is to find a good polynomial fit to the data, and how to choose the data points to be used which is further discussed in Section 4.1 Design of the Field Of View Models.

3.2.2 Analysis of the Measurement Error

To decide on a design approach for the measurement error model in Section 4.2 Design of the Measurement Error Model, the error is first analyzed. The measurement error from the FLR sensor is calculated using the polar coordinates (R, θ) from the FLR converted to Cartesian coordinates (x_{FLR}, y_{FLR}) according to Eq. (3.3). From these, the GPS coordinates (x_r, y_r) from the robot are subtracted

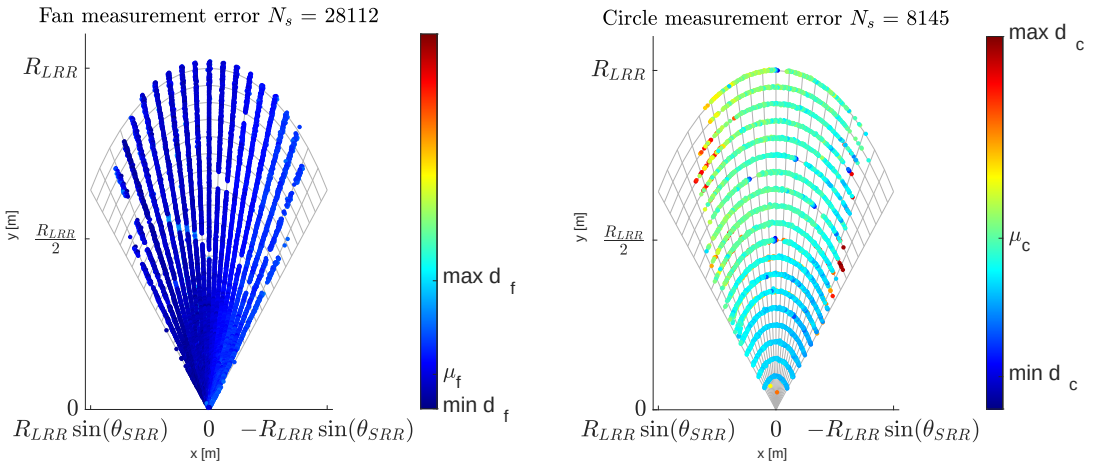
$$\begin{aligned} x_e &= x_{FLR} - x_r \\ y_e &= y_{FLR} - y_r \end{aligned} \tag{3.4}$$

the measurement error is then used to calculate the distance error

$$d = \sqrt{x_e^2 + y_e^2} \tag{3.5}$$

for both the fan and circle pattern test, presented in Section 3.1 Collection of Data Measurements. Outliers are removed by observing the spread of the error separately for both tests. The outlier filtering removed about 12.1 % of samples from the fan test and 19.7 % from the circle test. Meaning less than 14 % of all samples were removed, making the total number of samples $N_s = 36257$.

The distance errors d_f and d_c, where the indexation of the parameters signifies which test pattern the variable corresponds to, can be seen in Figure 3.7 below. Indexation f is the fan pattern test, and c is the circle pattern test. The figures visualize the location of the error where the color corresponds to the value of the error in the location. Both figures are displayed with the same color scale magnitude to be able to see the relationship between the errors from both tests.



(a) Distance error d_f in fan pattern. **(b)** Distance error d_c in circle pattern.

Figure 3.7: Distance error d in the a) fan pattern and b) circle pattern illustrated with color to show the magnitude of the error.

The distance error d is significantly smaller in the fan test, see Figure 3.7a, compared to the circle test, see Figure 3.7b. The largest error in the fan test is smaller than the mean distance error of the circle test, $\max d_f < \mu_c$. Furthermore, the mean in the fan test is closer to the min value than to the max, suggesting a small variance with some outliers of a larger error. Contrary to the fan test, the mean error of the circle test is almost exactly in the center of the min and max values, suggesting a relatively evenly spread of the error with a large variance.

Given the variation in error between the tests it seems necessary to design two models, one for each test. A radar generally has a higher accuracy when detecting objects with a small azimuth angle compared to those with a large azimuth angle, therefore the design of the measurement error model might be based on the angle.

However, because of the large number of samples in each of the previous figures, it is hard to distinguish a relationship between error magnitude and their location as the samples overlap each other. Therefore, in order to determine the magnitude and distribution of errors, the samples from each test are divided into figures depending on the magnitude of the distance error. Using the min, μ, and max values from each test separately,

$$\min d \leq d_i < \frac{\min d + \mu}{2} \qquad \text{interval 1} \tag{3.6a}$$

$$\frac{\min d + \mu}{2} \leq d_i < \mu \qquad \text{interval 2} \tag{3.6b}$$

$$\mu \leq d_i < \frac{\max d + \mu}{2} \qquad \text{interval 3} \tag{3.6c}$$

$$\frac{\max d + \mu}{2} \leq d_i \leq \max d \qquad \text{interval 4} \tag{3.6d}$$

all samples i in N_s are divided into four intervals based on their error magnitude to better observe the measurement error distribution for each test pattern. The samples in the four intervals are shown for both patterns in Figures 3.8-3.9, where the samples in interval 1, i.e. the smallest errors, are shown in the top left figure and samples in interval 4, i.e. the largest errors, are shown in the bottom right figure.

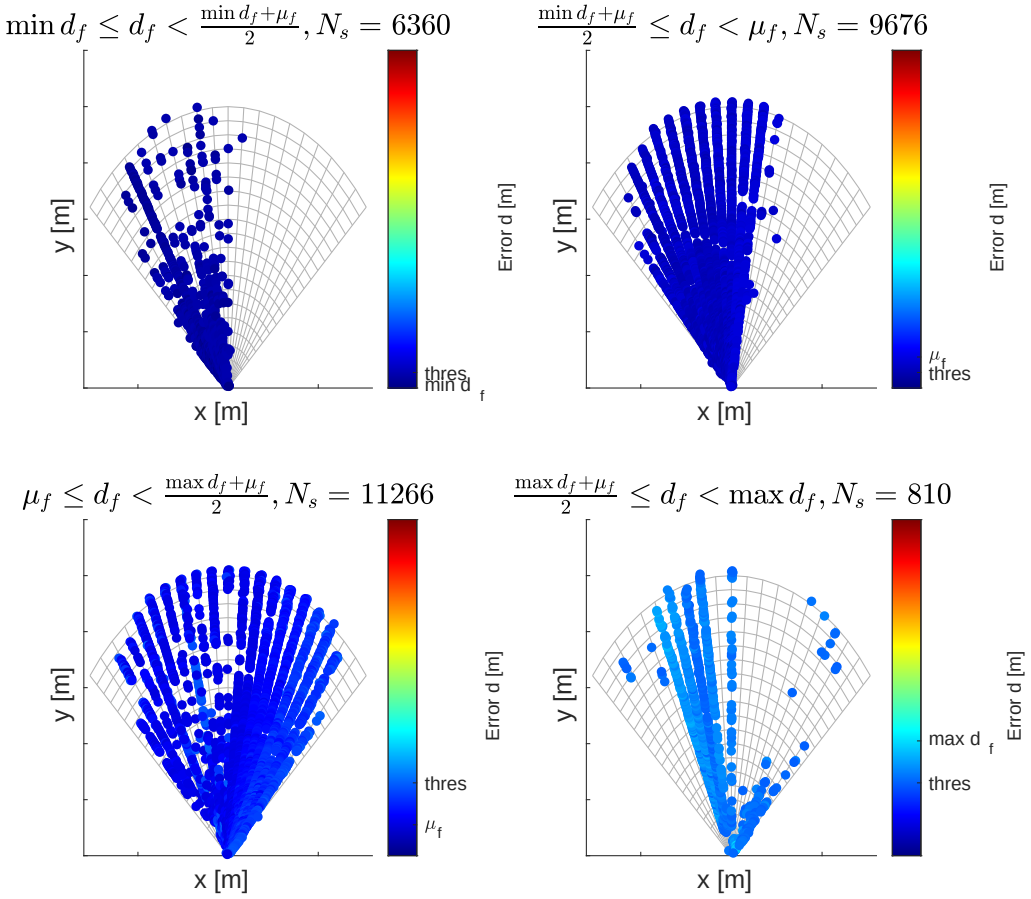


Figure 3.8: Error sections in the fan pattern to see distance error distribution with the smallest error in the top left and largest error in the bottom right corner.

In Figure 3.8 above it is observed that the larger errors are prone to occur on the left side of the FOV, mainly in angles θ_5, θ_6, θ_7, θ_8 and, θ_{10}, with some scattered on the right side. Furthermore, the left side has a wider variance compared to the right side which is fairly contained within the interval $\mu_f \leq d_f < \frac{\max d_f + \mu_f}{2}$. This does not show a connection between the magnitude of the error and the azimuth angle, but rather some systematical error for some angles in the test pattern. Therefore, it might not be reasonable to design the measurement error model depending on the azimuth angle in the FOV.

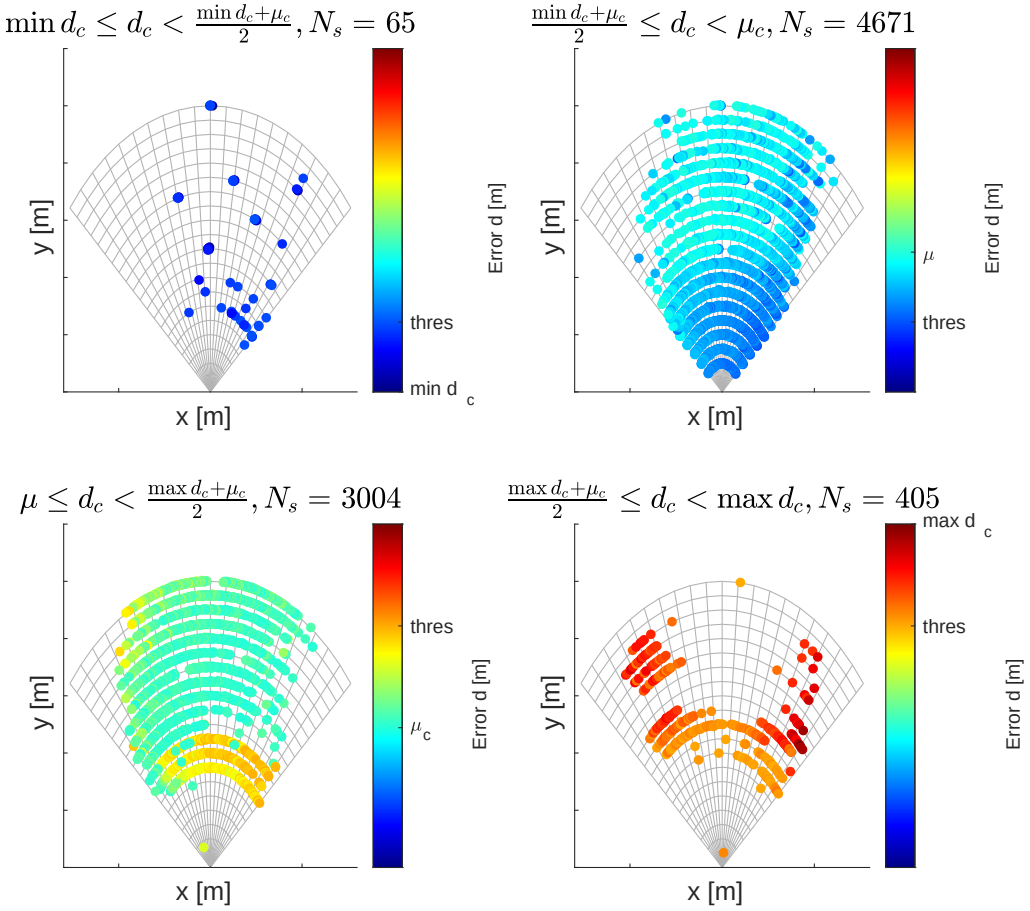
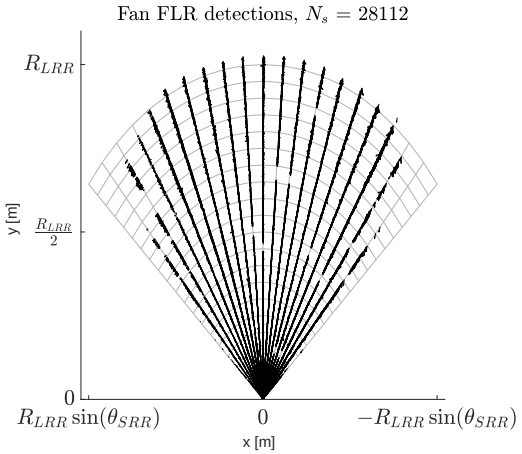


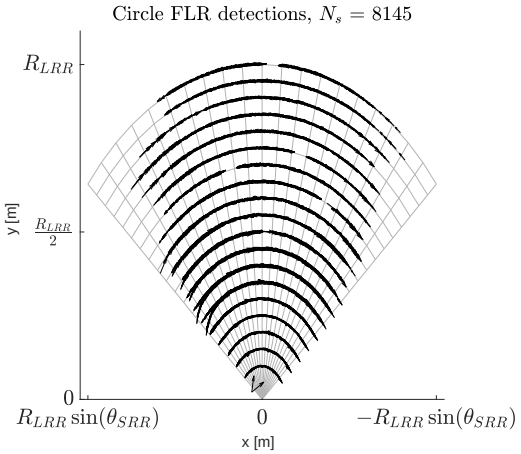
Figure 3.9: Error sections in the circle pattern to see distance error distribution with the smallest error in the top left and largest error in the bottom right corner.

In the circle pattern test, see Figure 3.9, large errors are located in the larger $|\theta|$ azimuth angle regions, which is excepted of a radar. However, similar to the fan test, the errors are not as consistent as previously thought. Therefore, the error model might not be accurate enough if one error is to be applied to an entire angle, this is further discussed in Section 4.2 Design of the Measurement Error Model.

The direction of the error is important to apply the error accurately in the simulation. In Figure 3.10 the direction of the error is illustrated with arrows. The arrowhead points to where the FLR measured the position of the robot, while the base of the arrow is at the position of the robot according to the GPS.



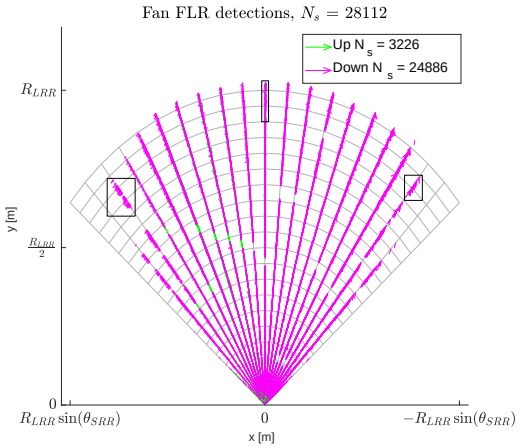
(a) Error direction in fan pattern.



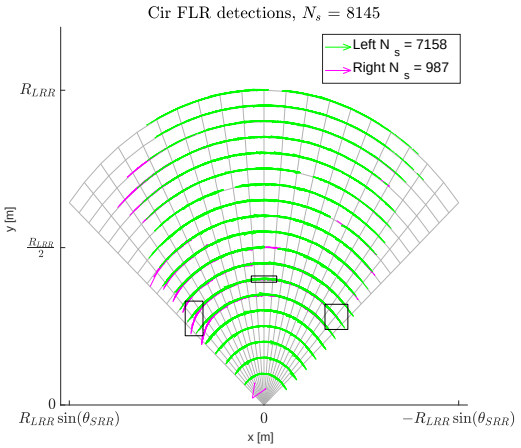
(b) Error diretion in circle pattern.

Figure 3.10: Error direction illustrated with arrows to show the direction of error.

It is hard to distinguish the direction of the arrows because the number of samples causes the arrows to overlap. Therefore, colorized versions are made, where the color signifies the direction of the arrow, see Figure 3.11.



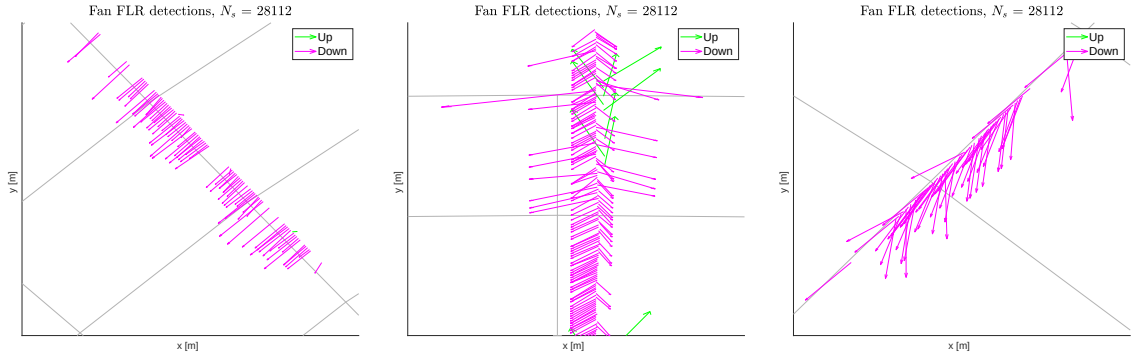
(a) Error direction in fan pattern with colorized direction.



(b) Error direction in circle pattern with colorized direction.

Figure 3.11: Colorized direction of the arrow in a) fan pattern and b) circle pattern with black boxes to illustrate where enhancements have occurred.

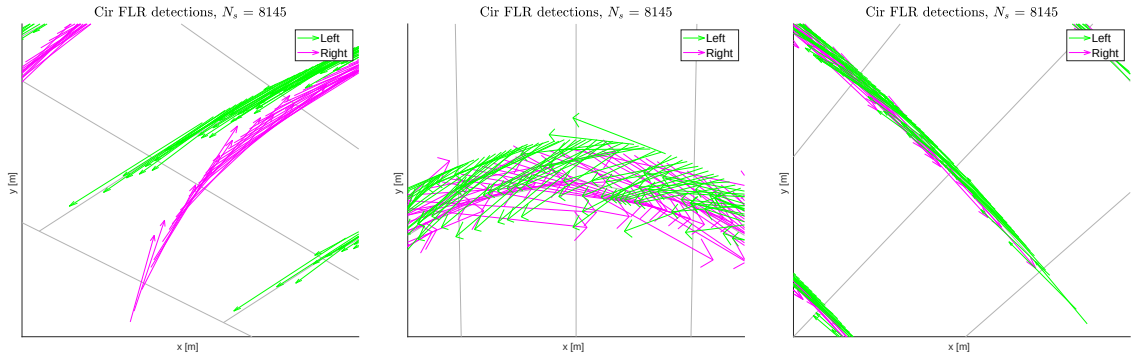
In the fan test, see Figure 3.11a the green arrows signify an upwards direction and the purple arrow a downward direction. The majority of the samples are directed down toward the host car. The circle test, see Figure 3.11b has the majority of the errors directed to the left, i.e. a green arrow while the purple arrow shows a right direction. However, it is still difficult to distinguish the arrows, therefore enhanced versions are made where the black boxes are placed, see Figures 3.12 - 3.13.



(a) Error in fan right. **(b)** Error in fan center. **(c)** Error in fan left.

Figure 3.12: Enhanced segments in the fan test to show the direction of the error.

In the fan pattern enhancements, it can be seen that the direction and magnitude of the error are relatively consistent in each small segment. However, not when comparing the segments with each other.



(a) Error in circle right. **(b)** Error in circle center. **(c)** Error in circle left.

Figure 3.13: Enhanced segments in the circle test to show the direction of the error.

The circle pattern enhancements do not show the same result, each segment has a rather conflicting error direction. With regard to the consistency seen in the enhanced segments of the fan test, the FOV is divided into small segments in a similar grid pattern as the testing grid made from Eq. (3.1)-(3.2). However, the segment grid should encompass the pattern of the fan and circle test. Therefore, the segment grid is shifted half the increment in both azimuth angle ($\Delta\theta$) and range (ΔR) to make the testing grid centered in the segment grid,

$$\begin{aligned} \phi_n &= \theta_{SRR} - n\Delta\theta + \tfrac{\Delta\theta}{2} \\ R_m &= m\Delta R + \tfrac{\Delta R}{2} \end{aligned} \tag{3.7}$$

this is done for n angles and m ranges. The segment grid is constructed with 22 angles and 21 range intervals which compose 441 segments. The segment grid can be seen in Figure 3.14.

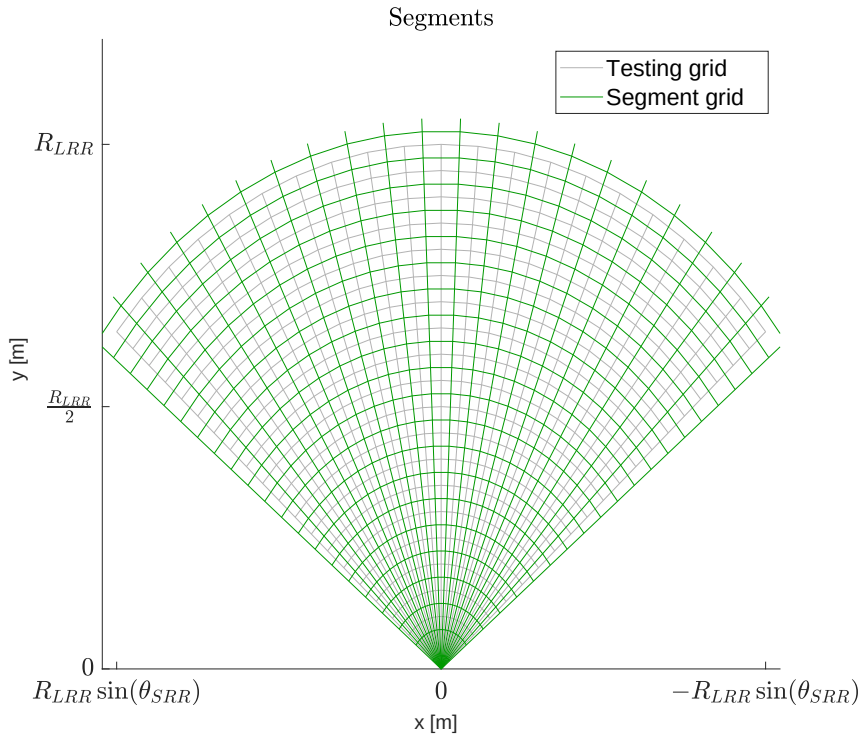


Figure 3.14: The testing grid (gray) along with the segment grid (green).

The segment grid in polar coordinates is shown in Figure 3.15 where range segment 1 is the segment closest to the host car.

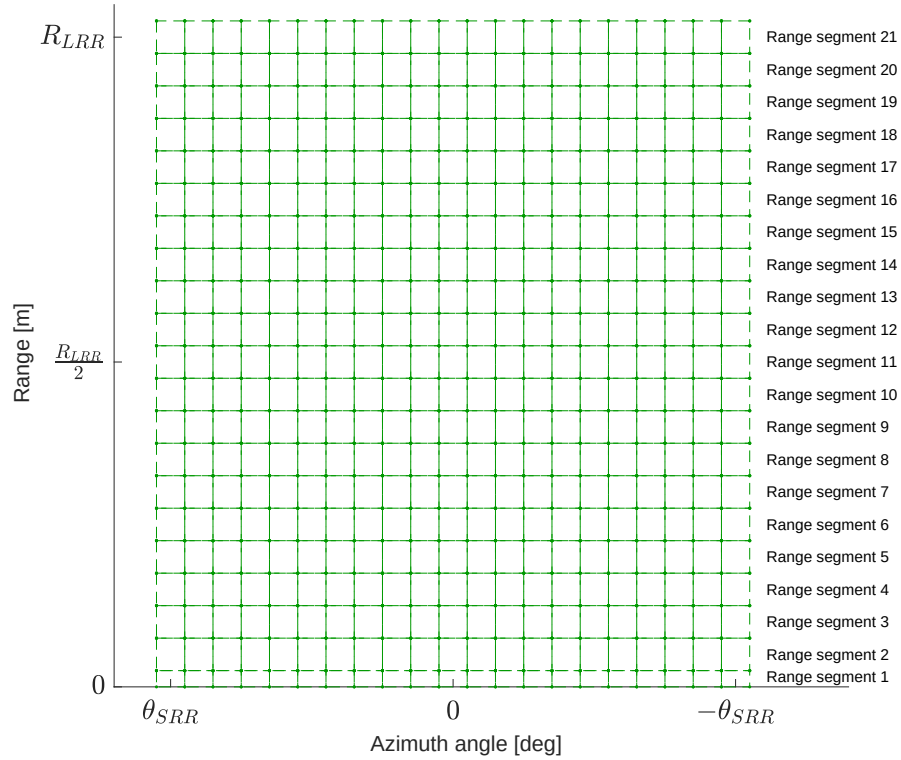


Figure 3.15: The segment grid (green) in polar coordinates.

The fan test pattern consists of many samples, $N_s = 28112$, and if evenly distributed there would be approximately 64 samples in each segment. There are 46 segments with zero samples, but the majority (267) of the segments had above 60 samples. The circle test pattern consists of fewer samples, $N_s = 8145$, and if evenly distributed each segment would have approximately 18 samples. Since the testing grid for the circle pattern was composed of 19 range intervals the two closest range segments include zero samples for the circle test, in total there are 88 segments with zero samples. The majority (261) have above 10 samples in each segment. To further observe the distribution of samples in each segment for both the fan and circle test, see Appendix A.1 Number of samples in each segment.

To further investigate the distribution and variance of the error, box plots were made for each range segment. A box plot is a standardized way of displaying the distribution of data with five components, the minimum, and maximum values, the median, and the lower and upper quantile [22]. The box is bounded by the lower and upper quantile of the data, meaning that the box includes 50 % of the observations with the median illustrated as a red line, see Figure 3.16. The black lines extending from the box are the whiskers which show the minimum and maximum values calculated as being one and a half box length away from the lower and upper quantiles. Data points that are further away than the whiskers are outliers [23], here illustrated as a red plus sign.

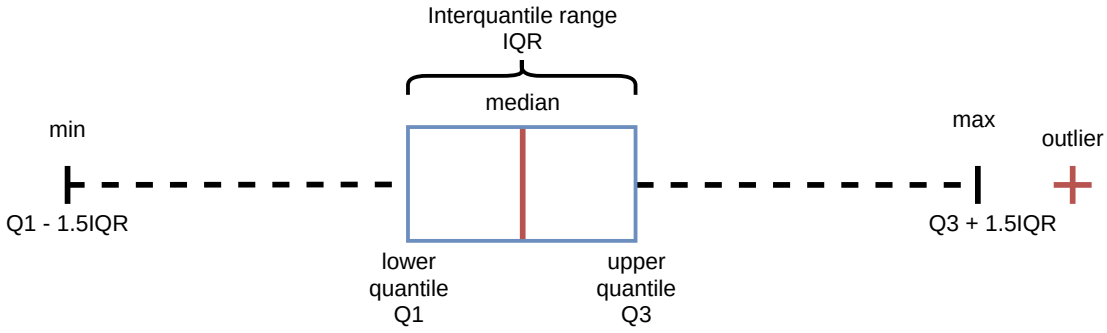


Figure 3.16: Illustration of a box plot made up of the median (red line) value, the lower and upper quantiles, the minimum and maximum values, and outliers (red plus).

In Figures 3.17-3.20 two of the range segments for each test are shown, while all box plots can be seen in Appendix A.2 Box plots for measurement error. Added to the box plots is the mean value illustrated as a green dot as well as the number of samples and outliers for each azimuth angle at the top of the plot in black and red respectively. As the error is significantly smaller for the fan test, the y-axis scale is therefore smaller for the fan box plots than for the circle box plots.

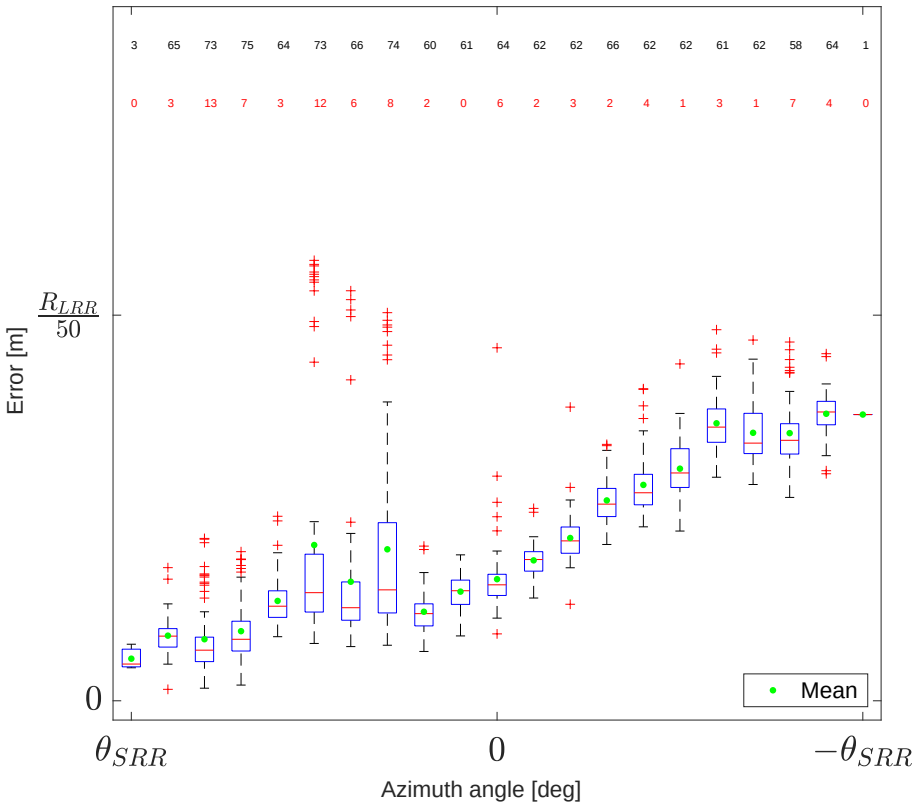


Figure 3.17: Box error plot in range segment 5 for the fan test.

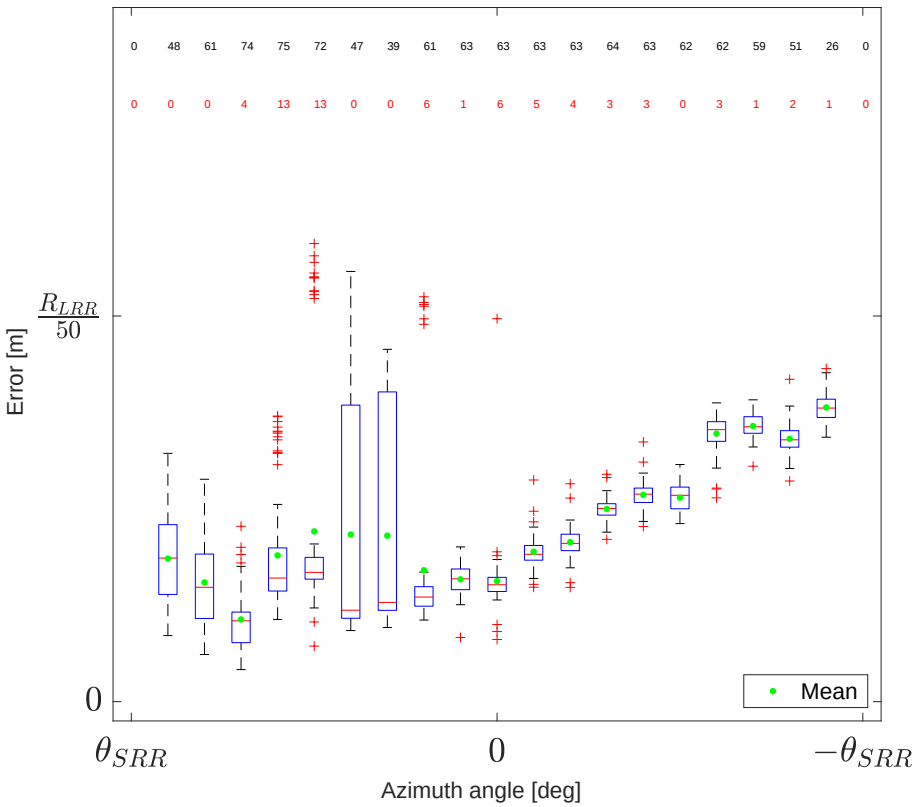


Figure 3.18: Box error plot in range segment 12 for the fan test.

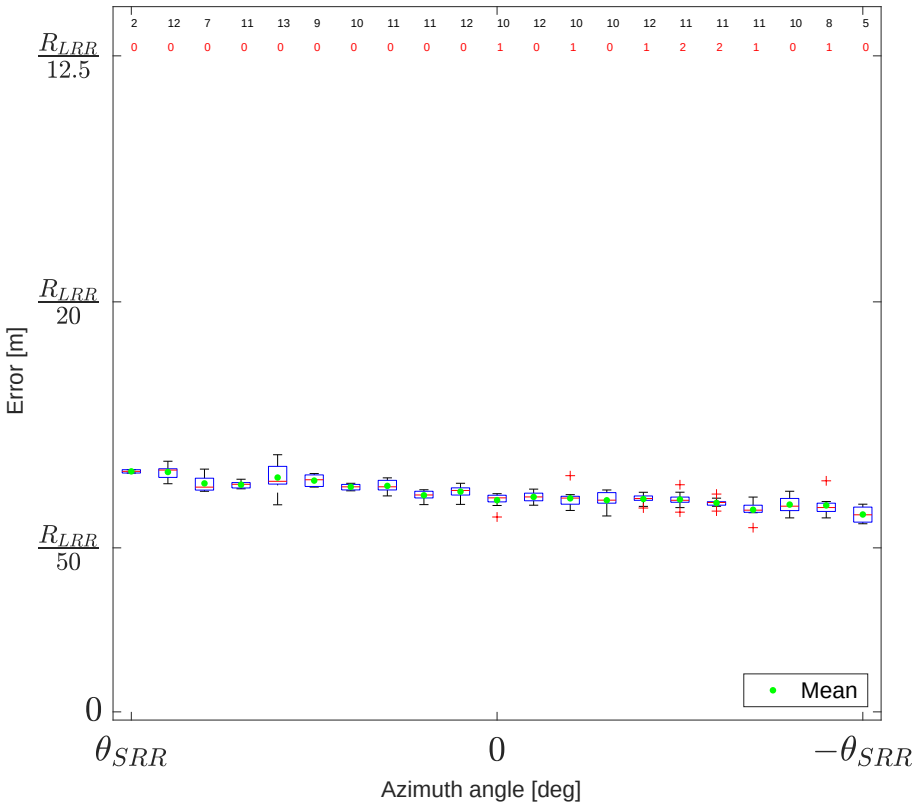


Figure 3.19: Box error plot in range segment 5 for the circle test.

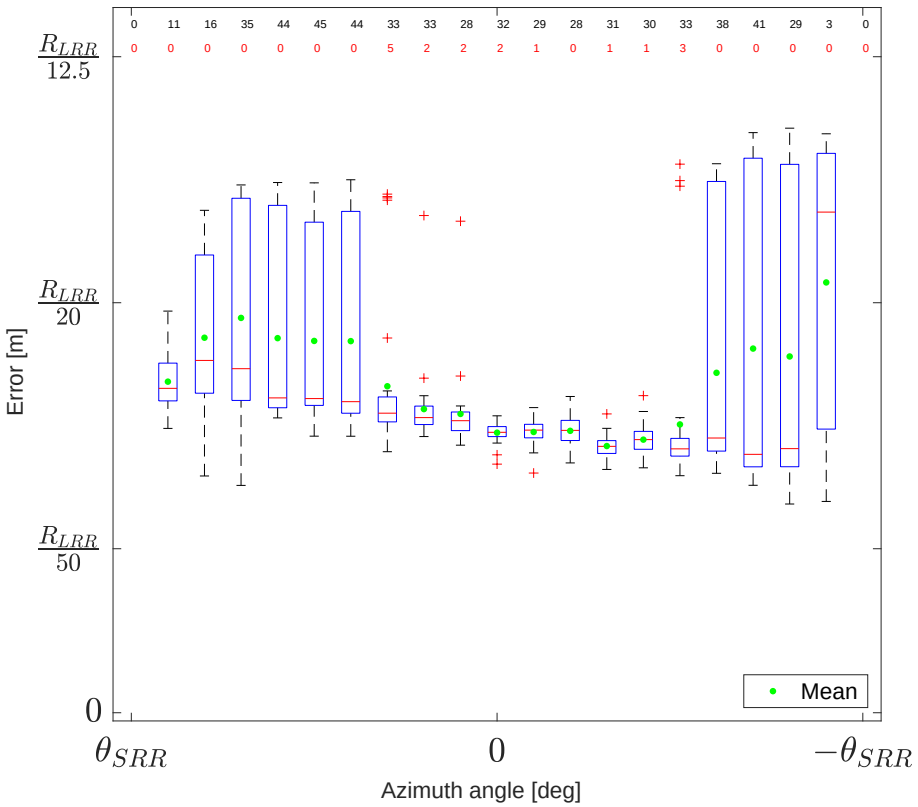


Figure 3.20: Box error plot in range segment 12 for the circle test.

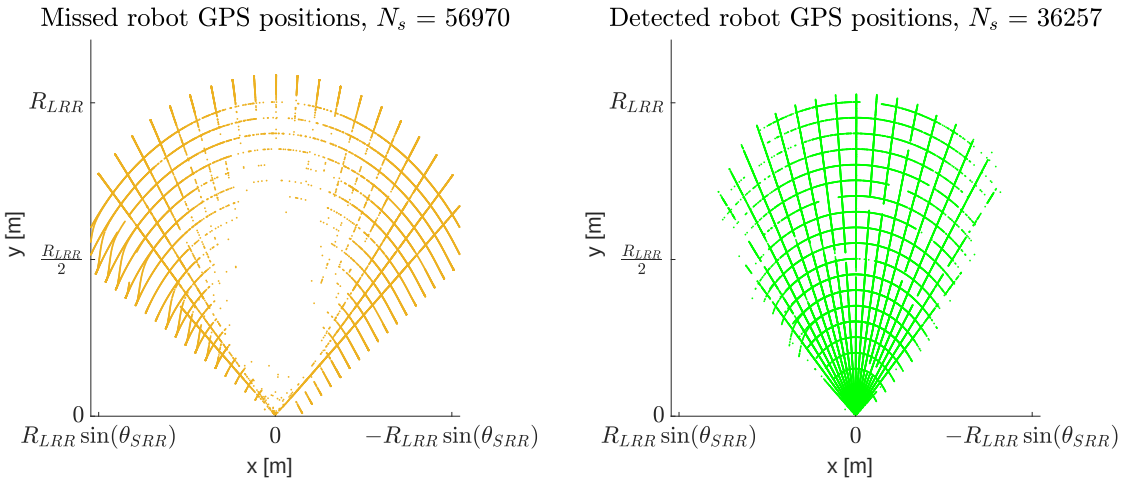
In Figure 3.8 it was observed that the largest error was mostly concentrated on the left side of the FOV in azimuth angles $\theta_5, \theta_6, \theta_7, \theta_8, \theta_{10}$, this again is shown in the box plot above. Furthermore, the box plots show the fan test pattern having an increasing distance error for a decreasing azimuth angle. From this, a simplistic model could be to calculate the error based on the azimuth angle. The box plots for the circle test show the same relationship as previously seen, with a larger variance in distance error for larger ($|\theta|$) azimuth angles in some range segments. However, this is not consistent for all range segments. The design of the measurement error model is further discussed in Section 4.2 Design of the Measurement Error Model.

3.2.3 Analysis of the Detection Rate

The detection rate is calculated by comparing the number of detected robot samples (N_d) and the number of missed robot samples (N_m). Considering the segment grid previously introduced in Figure 3.15, the probability of detection

$$P_d = \frac{N_d}{N_d + N_m} \tag{3.8}$$

is calculated for each segment. In Figure 3.21a, all missed robot GPS positions can be seen, and in Figure 3.21b all detected robot GPS positions can be seen.



(a) Missed robot positions. **(b)** Detected robot positions.

Figure 3.21: The missed and detected robot GPS positions.

In Figure 3.22 the polar coordinates of the missed and detected robot position is plotted along with the segment grid.

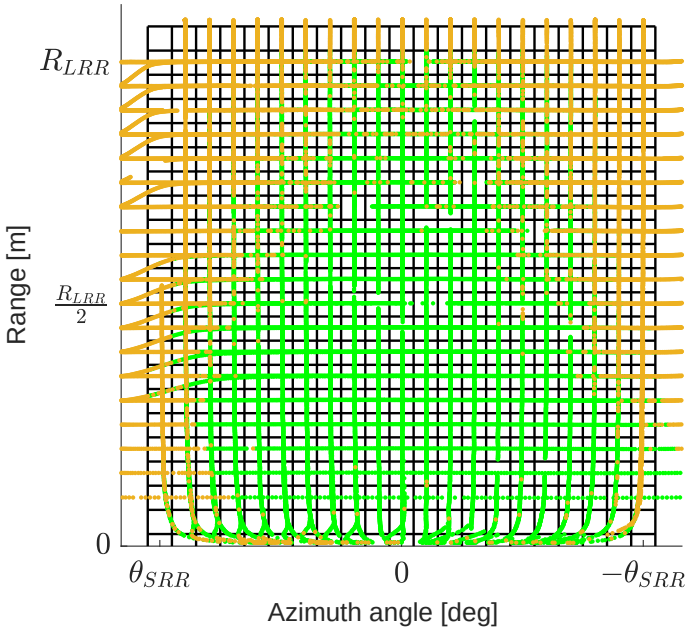


Figure 3.22: FLR detections (green) and misses (orange) of the robot position in the segment grid (black).

The calculated probability of detection in each segment can be seen in Figure 3.23, where the color of each segment corresponds to its probability of detection.

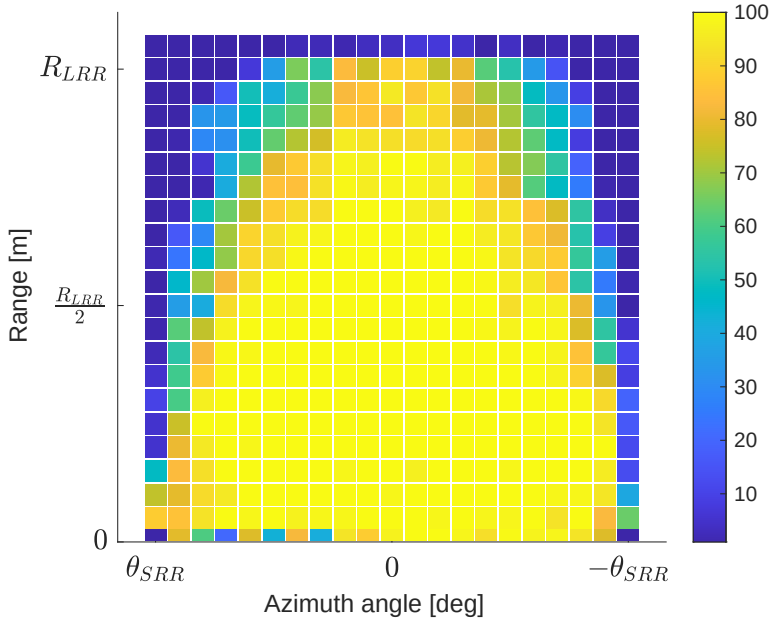


Figure 3.23: Probability of detection P_d along with the number of samples in each segment.

This is further discussed in Section 4.3 Design of the Detection Rate Model.

4

Design of Radar Sensor Models

In this chapter, the design of the FOV models, measurement error, and detection rate based on the analysis of measurement data in the previous chapter is presented.

4.1 Design of the Field Of View Models

From the analysis of the collected measurement data in Section 3.2.1 Analysis of the Field Of View it was found suitable to design the FOV using two approaches, as 2D polygon and polynomial equations. However, whether it is more accurate to design the polygon shape as convex or concave can be debated, therefore both are designed and evaluated separately.

4.1.1 Design of the Polygon Field Of View Models

As previously mentioned in Section 3.2.1 Analysis of the Field Of View, a polygon is described by a finite number of vertices to form straight line segments, called edges, which are closed to form a polygon chain, see Figure 4.1.

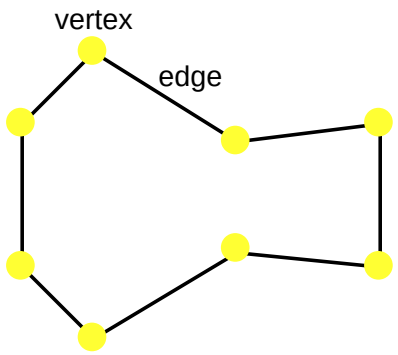
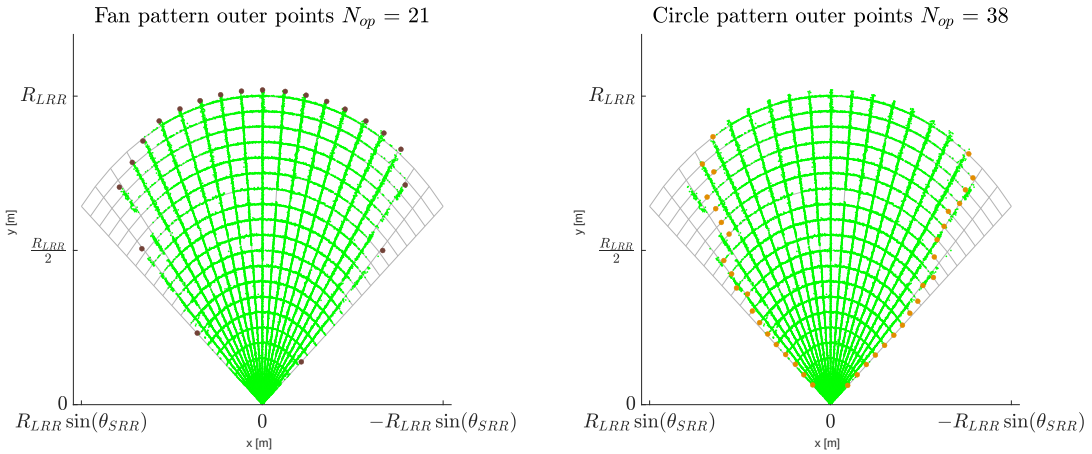


Figure 4.1: A polygon with vertices (yellow) which form the polygon edges (black).

To get the vertices for a polygon, the collected FLR measurements for the fan and circle test, presented in Section 3.1 Collection of Data Measurements, are viewed separately. For the fan test, the FLR sample with the maximum radius was taken

for each angle in the testing grid. With a total of 21 angles this generated 21 outer points, (N_{op}), see Figure 4.2a. For the circle test, the sample with the largest and smallest angle was taken for each range increment. With a total of 19 range segments for the circle test, this generated 38 outer points, see Figure 4.2b.



(a) Fan outer points for each angle (brown).

(b) Circle outer points for each range (orange).

Figure 4.2: The testing grid (gray) and FLR detections (green) together with the number of outer points N_{op} from the a) fan test (brown) and b) circle test (orange).

Together the fan and circle test has 59 outer points and with the origin, the total number of outer points is 60, see Figure 4.3.

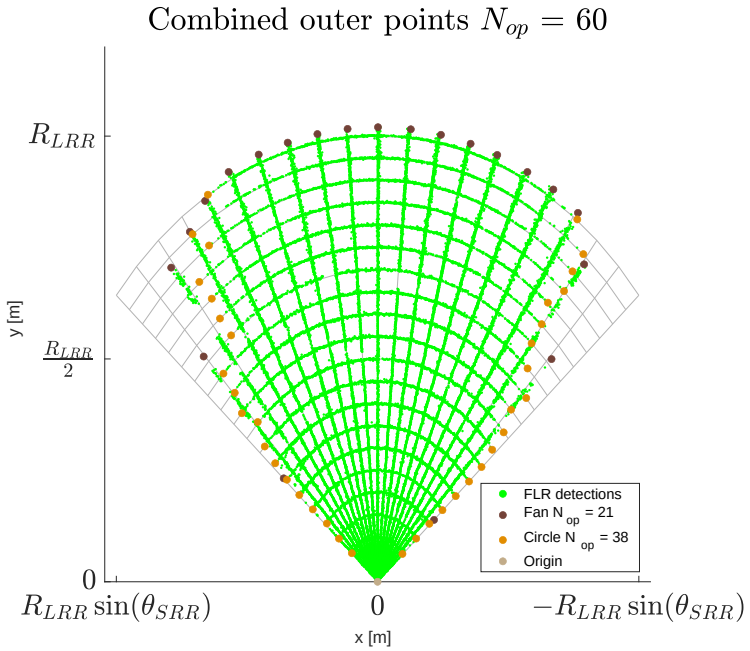


Figure 4.3: FLR detections (green) together the number of outer points N_{op} of the fan (brown) and circle (orange) test as well as the origin (beige).

With the selected outer points, a concave polygon was formed while removing the collinear points. Points are collinear if they belong to the same straight line segment, two points are described as trivially collinear as they always form a straight line [24]. Therefore, the outer points are evaluated in sets of three points in sequential order. If collinear, the middle point (P_2) is removed, see Figure 4.4. The collinear points are removed to ensure the polygon contains the least amount of vertices to make it less computationally complex since the complexity of the polygon increases with the number of polygon vertices (N_v), as presented in Section 2.2.1 Point In Polygon.

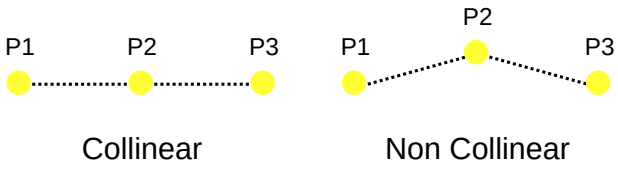


Figure 4.4: Illustration of a set of three points as collinear and non-collinear.

The outer points removed are the two circle points on the right side closest to the origin. For the concave polygon, seen in Figure 4.5, the number of samples inside the FOV was $N_{si} = 41731$, meaning 99.06 % of all detections are inside the FOV.

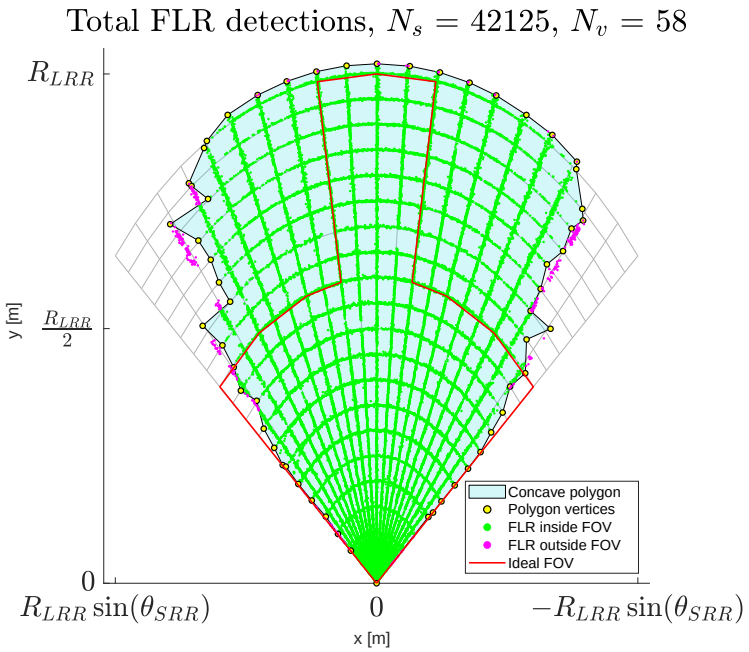


Figure 4.5: The testing grid (gray) along with the FOV designed as a concave polygon (blue) constructed from the polygon vertices (yellow). The FLR detections inside (green) and outside the FOV (magenta), as well as the ideal FOV (red).

The convex hull of the concave polygon was calculated using the MATLAB function ***convhull***. The convex polygon was constructed with 23 of the outer points, see Figure 4.6, with $N_{si} = 42099$, meaning 99.94 % of all detections are inside the FOV.

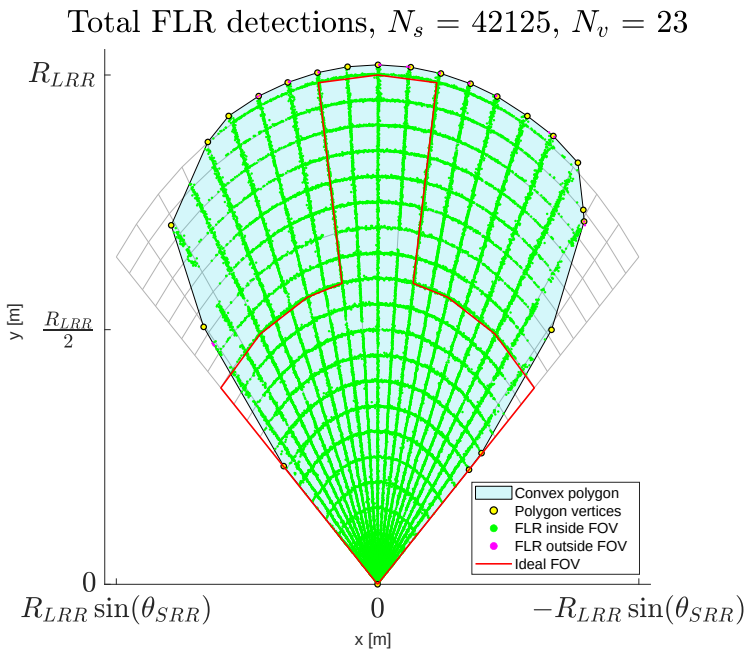


Figure 4.6: The testing grid (gray) along with the FOV designed as a convex polygon (blue) constructed from the polygon vertices (yellow). The FLR detections inside (green) and outside the FOV (magenta), as well as the ideal FOV (red).

4.1.2 Design of the Polynomial Field Of View Model

By analyzing the FLR detections, in Section 3.2.1 Analysis of the Field Of View, three polynomial equations were the least amount found feasible to design the FOV boundary. The polynomial equations intersect at the intersection points which were chosen as the extreme points of the FLR detections, see Figure 4.7.

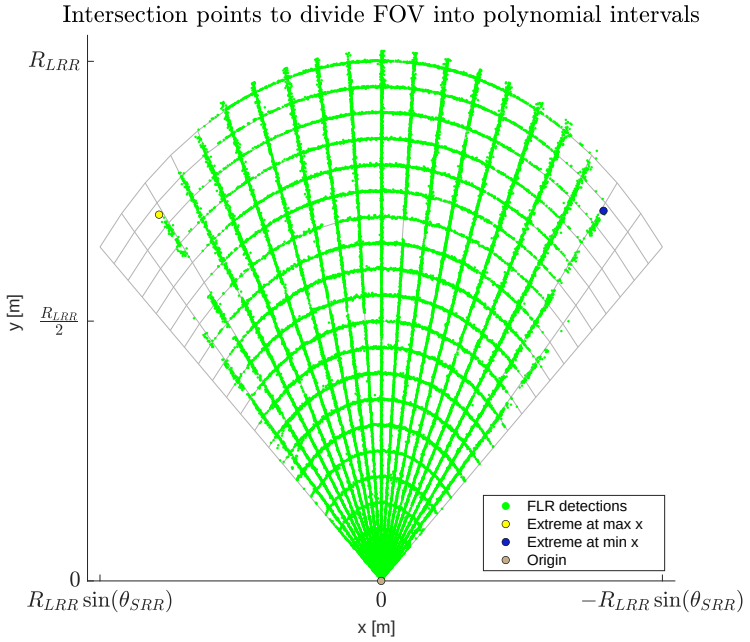


Figure 4.7: Intersection points (I) marking the intervals of the polynomials.

The polynomials were designed using the previously obtained outer points for the polynomial regression. More data points provide more accuracy in the polynomial fit, hence all N_{op} was used. As presented in Section 2.2.2 Point In Polynomial the complexity increases quadratically with the order of the polynomial equation, therefore the order needs to be as low as possible while still including enough FLR detections within the FOV boundary. The least degree that was found to include as many FLR detections as possible while not overestimating the FOV boundary was with a degree of four for all polynomials. The three polynomials and the corresponding N_{op} data points for each polynomial can be seen in Figure 4.8. The design had $N_{si} = 41273$, meaning 97.98 % of the FLR detections are inside the FOV.

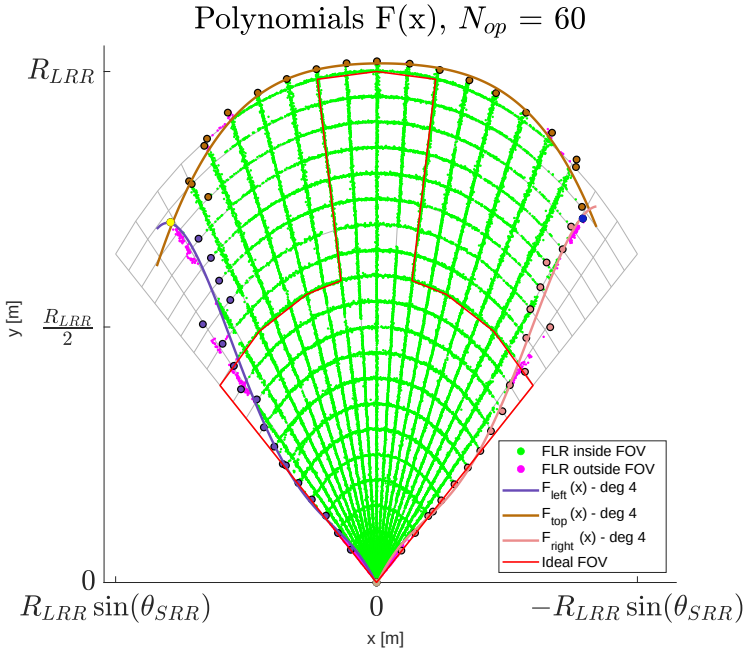


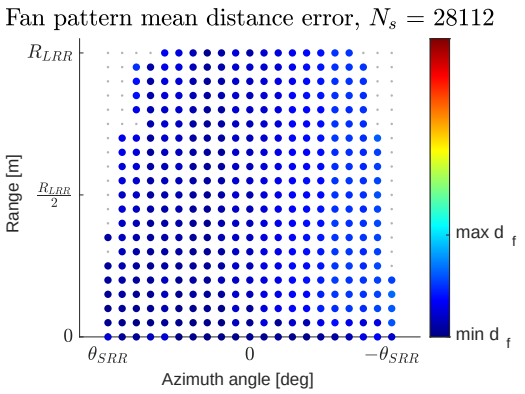
Figure 4.8: The testing grid (gray) along with the FOV design as three polynomial equations of the fourth degree. The FLR detections inside (green) and outside the FOV (magenta) as well as the ideal FOV (red).

Some additional polynomial combinations can be seen in Appendix A.3 Polynomial along with the percentage of FLR detections inside the FOV and a short summary of why they were not found suitable.

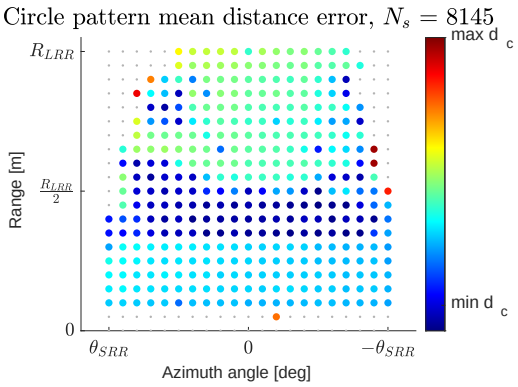
4.2 Design of the Measurement Error Model

From the analysis of the measurement error presented in Section 3.2.2 Analysis of the Measurement Error, it was found to be a considerable variation in error between the fan and circle tests. Therefore, two separate error models were designed, one according to the fan test data and one according to the circle test data. A simplistic model to represent the average performance of the radar is a model that applies the mean error from the segment that the object lies within. The mean distance error

d, from Eq. (3.5), can be seen in Figure 4.9 below, where the color corresponds to the magnitude of the error.



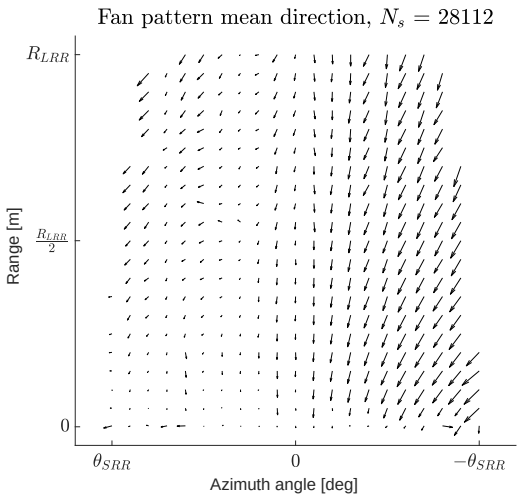
(a) Fan pattern mean distance error.



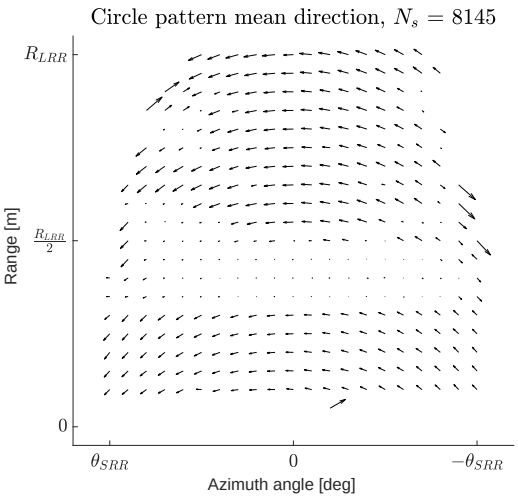
(b) Circle pattern mean distance error.

Figure 4.9: The mean distance error d in each segment for the a) fan pattern and b) circle pattern.

The distance error is composed of the error in x and y coordinates, x_e and y_e respectively from Eq. (3.4). The direction in which the error should be applied is therefore the mean in x and y coordinates, $\bar{x}_e$ and $\bar{y}_e$, see Figure 4.10.



(a) Fan pattern direction of the error.



(b) Circle pattern direction of the error.

Figure 4.10: Direction of the error structured from $\bar{x}_e$ and $\bar{y}_e$ in the a) fan pattern and b) circle pattern illustrated as black arrows.

For the simplistic mean error model, the mean error in x and y coordinates $\bar{x}_e$ and $\bar{y}_e$ is structured in a mean error vector

$$\boldsymbol{\mu} = [\bar{x}_e, \bar{y}_e]^\top \tag{4.1}$$

for each segment which structured the mean error matrix

$$E_{\boldsymbol{\mu}} = \begin{bmatrix} \boldsymbol{\mu}_{11} & \dots & \boldsymbol{\mu}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\mu}_{m1} & \dots & \boldsymbol{\mu}_{mn} \end{bmatrix} \tag{4.2}$$

where $\boldsymbol{\mu}_{m,n}$ is the m:th range segment and n:th azimuth angle.

The design of the simplistic mean error model consists of two matrices, $E_{\boldsymbol{\mu},fan}$ and $E_{\boldsymbol{\mu},circle}$. However, from analyzing each segment in Appendix A.2 Box plots for measurement error it was found that segments had diverse variances. Using the mean value in each segment might be too simplistic since it does not represent the variance that occurs within each segment. Investigating the distribution of error in each segment, it was found that the normal distribution was a good fit for the majority of the fan pattern segments. For the circle pattern segments there are mostly too few samples in each segments to draw any good conclusions of what distribution would be a good fit, therefore, a normal distribution is used there as well. The probability density function

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.3}$$

is used to draw an error sample from the normal distribution according to the mean (μ) and standard deviation (σ) in each segment. Therefore, the standard deviation was calculated in both x and y coordinates and saved to a standard deviation error vector

$$\boldsymbol{\sigma} = [\sigma_x, \sigma_y]^\top \tag{4.4}$$

for each segment that constructed the standard deviation error matrix

$$E_{\boldsymbol{\sigma}} = \begin{bmatrix} \boldsymbol{\sigma}_{11} & \dots & \boldsymbol{\sigma}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\sigma}_{m1} & \dots & \boldsymbol{\sigma}_{mn} \end{bmatrix} \tag{4.5}$$

The implementation is further discussed in Section 5.3 Implementation of the Measurement Error Model.

4.3 Design of the Detection Rate Model

The probability of detection P_d, as introduced in Section 3.2.3 Analysis of the Detection Rate, was used as model parameters for the detection rate model. The model parameter matrix

$$\boldsymbol{P} = \begin{bmatrix} P_{d,11} & \dots & P_{d,1n} \\ \vdots & \ddots & \vdots \\ P_{d,m1} & \dots & P_{d,mn} \end{bmatrix} \tag{4.6}$$

structure the probability of detection of each segment. For segments that contain zero samples, the probability of detection is 0 %. The design of the detection rate model assumes that the radar's tracking algorithm is perfect, meaning that once a target is detected the radar will continue to detect it until it leaves the FOV. This is further discussed in Section 5.4 Implementation of the Detection Rate Model.

5

Implementation of Radar Sensor Models in the Simulation Platform

In this chapter, an overview of how the designed models are implemented is given. The chapter starts by describing how the CSPAS structure will be altered followed by a more detailed presentation of how each model design is implemented.

5.1 Implementation structure

The sensor model consists of three separate modules, one for each model, namely the FOV, measurement error, and detection rate. The models are implemented into CSPAS according to Figure 5.1. Implementing the models as three separate modules provides flexibility, in the sense that models can be added, removed, and easily modified. A more in-depth introduction to CSPAS is presented in Section 1.4 Compiled Simulation Platform for Active Safety.

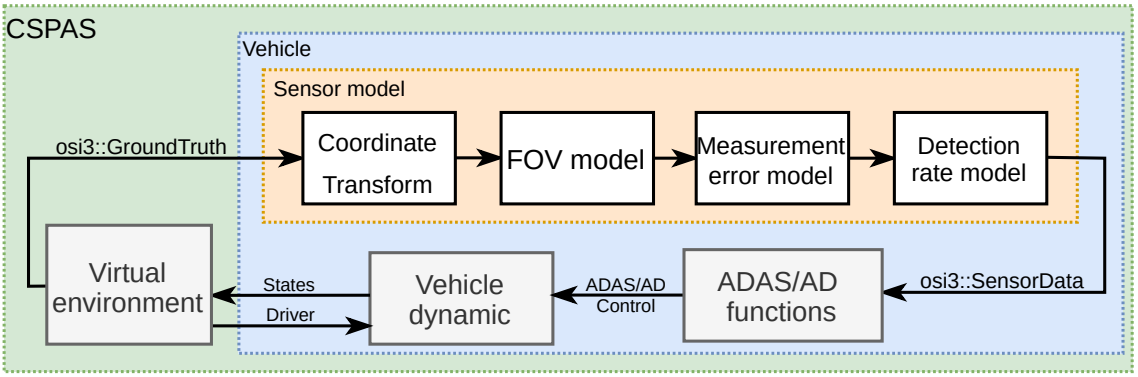


Figure 5.1: Overview of CSPAS with the implemented sensor model.

The input to the FOV models is the corner positions in sensor coordinates (x_s, y_s). The FOV model evaluates if the object is detected by calculating if two points are inside the FOV shape, as presented in Section 1.4 Compiled Simulation Platform for Active Safety. The algorithms used for calculating this are presented in Section 2.2 Point in Field Of View.

Having the FOV model first in the sensor model reduces the computational complexity, since only if a target is detected it is passed on to the measurement error model. From the measurement error model the error (x_e, y_e) is added to the true position in sensor coordinates based on which segment the target is within, as presented in Section 4.2 Design of the Measurement Error Model. Lastly, the detection rate model is placed, where the detection probability is calculated for the detections made in the FOV model.

5.2 Implementation of the Field Of View Models

The implementation for the FOV models mostly consists of incorporating the algorithms presented in Section 2.2 Point in Field Of View. When executing a simulation in CSPAS the FOV type is chosen in the configuration as either *Convex*, *Concave*, or *Polynomial* which calls the corresponding algorithm. For the polygon models, the PIP algorithms only need the polygon vertices, introduced in Section 4.1 Design of the Field Of View Models, as model parameters.

For *Polynomials* the model parameters are the coefficients and intersections of the three polynomials. However, the coefficients were numerically small, meaning that the evaluation of the polynomials can be affected by truncation. Therefore, the model instead takes the centered and scaled x coordinate position

$$\hat{x} = \frac{x - \bar{x}}{\sigma_x} \tag{5.1}$$

where $\bar{x}$ is the mean and σ_x is the standard deviation of the x value of the outer points (N_{op}) corresponding to each polynomial as seen in Figure 4.8. The equation to evaluate the polynomials is now

$$f(\hat{x}) = \sum_{k=0}^{s} a_k \hat{x}^k \tag{5.2}$$

where s is the polynomial degree and a_k is the s:th coefficient calculated from polynomial regression with the scaled and centered outer point x values.

5.3 Implementation of the Measurement Error Model

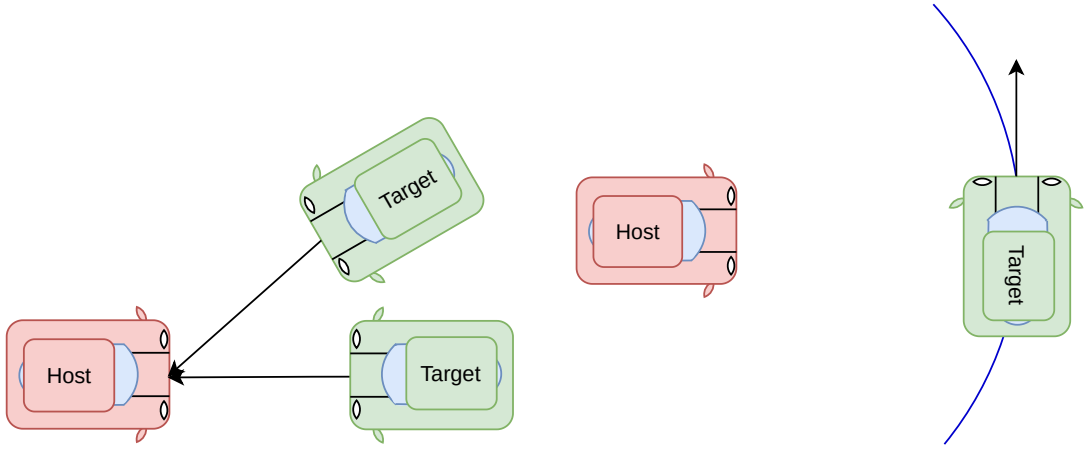
With the significant variation in error between the fan and circle test as described in Section 3.2.2 Analysis of the Measurement Error, it was found suitable to apply the error based on the direction the target is traveling. Therefore, if the object is driving radially toward the host vehicle the fan error will be applied, while if the target is driving in a circular pattern, for instance performing a turn in front of the host vehicle, the circle error will be applied. To determine whether the target is traveling in a fan or circular pattern the difference in range and angle is calculated

$$\begin{aligned} \Delta R &= R_{ts} - R_{ts-1} \\ \Delta\theta &= \theta_{ts} - \theta_{ts-1} \end{aligned} \tag{5.3}$$

between each time step (ts). By calculating the velocity in radius and angle

$$\begin{aligned} v_R &= \Delta R \\ v_\theta &= \Delta\theta \cdot R_{ts-1} \end{aligned} \tag{5.4}$$

the fan pattern has a difference in radius (ΔR) but not in angle ($\Delta\theta$), while the circle pattern has a difference in angle ($\Delta\theta$) but not in radius (ΔR), see Figure 5.2.



(a) Target driving in a fan pattern.

(b) Target driving in a circular pattern.

Figure 5.2: Target driving in a a) fan and b) circle pattern.

A ratio between the velocity in radius and angle is calculated

$$\mathbb{R} = \frac{|v_R|}{|v_R| + |v_\theta|} \tag{5.5}$$

which gives a fan pattern direction $\mathbb{R} = 1$, and a circle pattern direction $\mathbb{R} = 0$. Therefore the error is calculated as

$$\begin{aligned} E_x &= e_{x,f} \cdot (\mathbb{R}) + e_{x,c} \cdot (1 - \mathbb{R}) \\ E_y &= e_{y,f} \cdot (\mathbb{R}) + e_{y,c} \cdot (1 - \mathbb{R}) \end{aligned} \tag{5.6}$$

in Cartesian coordinates. The error applied to the target is therefore calculated from to the velocity vector, with the error (e_x, e_y) separately for the fan and circle pattern in Cartesian coordinates. The error is chosen in the simulation from either the mean value $\boldsymbol{\mu}$ in each segment, or as a sample drawn from the normal distribution in each segment, as previously discussed in Section 4.2 Design of the Measurement Error Model.

5.4 Implementation of the Detection Rate Model

With the detection rate model as the last model, the object sensor coordinates (x_s, y_s) are detected. The random variable

$$X \sim U(0, 100) \tag{5.7}$$

where U is a uniform distribution is used as a condition for detection

$$X < P_d \tag{5.8}$$

where P_d is the probability of detection. Since the model assumes a perfect tracking of the object, as soon as the condition 5.8 is met, the target is detected and the model internally marks it detected. This internal detection will stay true until the target leaves the FOV, as presented in Section 4.3 Design of the Detection Rate Model. In Algorithm 5, the implementation is described in detail.

Algorithm 5: Check if visible targets are detected.

```
Input:  p[i] = (x[i], y[i]) ∀i ∈ [0, N_d)                    /* Detected corner */
Input:  P_d[i]                                               /* Probability of detection */
Input:  detection[i] = True                                  /* Detection is true (inside FOV) */
Data:   detection_internal[i] = False                        /* Initially false  */
for (i := 0; i < N_d; i += 1) do
    if detection_internal[i] == False then
        X ~ U(0, 100) ;
        if X < P_d[i] then
            detection[i] = False  ;                          /* Object missed */
            detection_internal[i] = False  ;                 /* Mark internally false */
        else
            detection[i] = True  ;                           /* Object detected */
            detection_internal[i] = True  ;                  /* Mark internally true */
        end
    end
    if p[i]_{ts-1} ∉ p_ts then
        detection_internal[i] = False ;                      /* Reset if p[i] is outside FOV.  */
    end
end
```

6

Evaluation of the Radar Sensor Models

In this chapter, the radar sensor models are evaluated. The scenarios used for evaluation along with the result according to specifications made in Section 1.2 Contributions is presented. The idealistic FOV model is compared with the new FOV models based on their execution time and repeatability.

6.1 Scenarios used for Evaluation

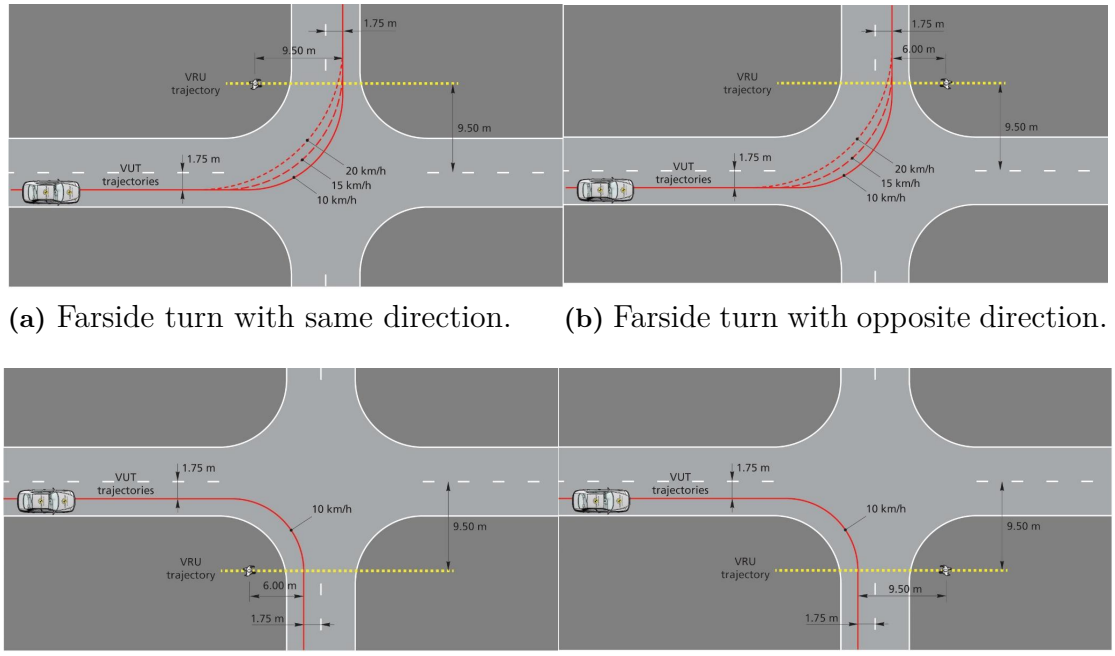
The scenario used for the evaluation of the models is the Car-Pedestrian-Turning-Adult (CPTA) scenario from the European New Car Assessment Programme (EuroNCAP). However, since the data measurements presented in Section 3.1 Collection of Data Measurements were collected with a robot mounted with a child pedestrian doll, the simulated pedestrian has therefore been altered from the adult dimensions to a child. Thereby, changing the name of the simulation to Car-Pedestrian-Turning-Child (CPTC). Since the new FOV models have increased the visual field at large azimuth angles $|\theta|$, a turning scenario was found most suitable for evaluating the models. Other scenarios have also been created to evaluate the measurement error and detection rate models. The scenarios are presented further below.

6.1.1 The Car-Pedestrian-Turning-Child Scenario

The CPTC scenario is composed of eight scenarios to perform a full range of testing of the radar [3]. The host is varying the speed according to specifications seen in Table 6.1. The host car is nearing a four-way crossing and makes a farside and nearside turn, see Figure 6.1. The child pedestrian crosses the road that the car approaches, after the turn is performed. The direction of the pedestrian alternates between being the same as the host car, and the opposite.

Table 6.1: Simulation specifications for the Car-Pedestrian-Turning-Child scenario.

	Farside			Nearside	
Host speed	10	15	20	10	km/h
Target Speed	5				km/h



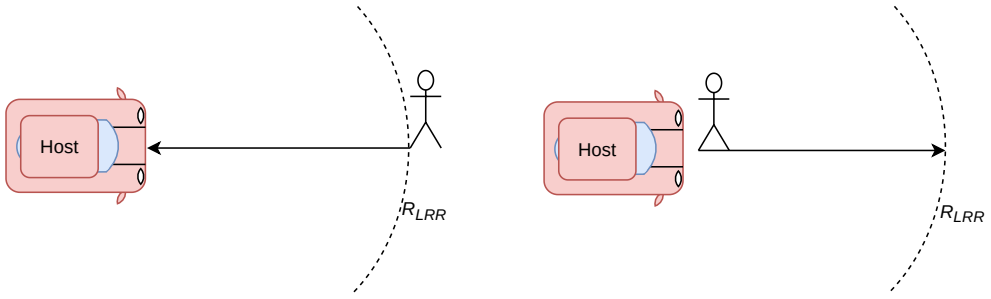
(a) Farside turn with same direction. **(b)** Farside turn with opposite direction.

(c) Nearside turn with same direction. **(d)** Nearside turn with opp direction.

Figure 6.1: Car-Pedstrian-Turning-Child, figures from EuroNCAP [3].

6.1.2 The Fan Pattern Scenario

The fan pattern scenario is designed to replicate the fan testing performed when the collection of data measurements was conducted, introduced in Section 3.1 Collection of Data Measurements. Similarly to the CPTC scenario, the child pedestrian is traveling at 5 km/h while the host car is stationary. There are two fan scenarios, one starting at R_{LRR} traveling towards the host car with azimuth angle 0° and another starting directly in front of the car, traveling away from the host until R_{LRR} is surpassed, see Figure 6.2.



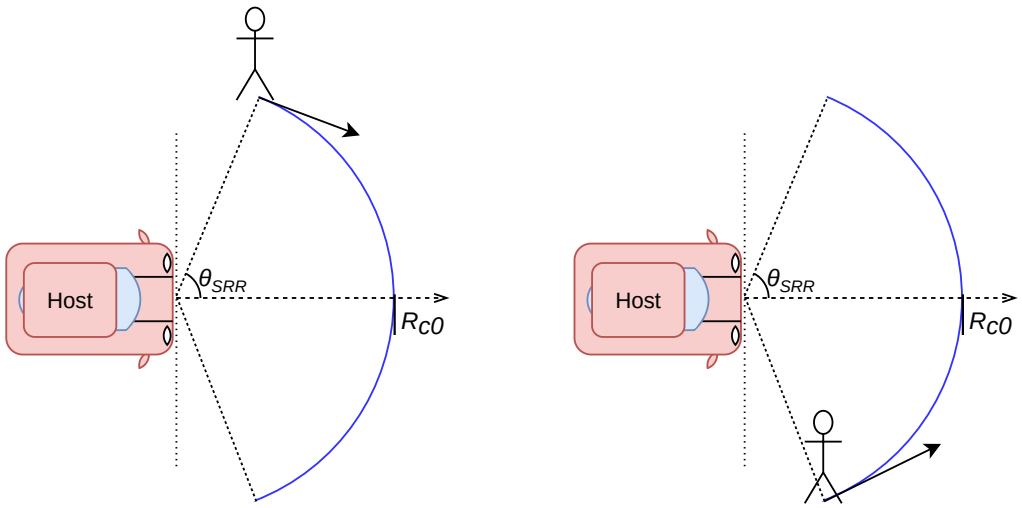
(a) Fan scenario with pedestrian traveling towards the host car.

(b) Fan scenario with pedestrian traveling away from the host car.

Figure 6.2: Fan pattern scenario for evaluating the measurement error model.

6.1.3 The Circle Pattern Scenario

Similarly to the fan pattern scenario, the circle pattern scenario is designed to replicate the circle testing performed when the collection of data measurement was conducted, as presented in Section 3.1 Collection of Data Measurements. The child pedestrian is traveling at 5 km/h while the host car is stationary. There are two circle scenarios, one starting at azimuth angle θ_{SRR} traveling in an arc with range R_{c0} until the azimuth angle θ_{SRR} is surpassed. The other is starting at azimuth angle $-\theta_{SRR}$ traveling in the arc with range R_{c0} until the azimuth angle θ_{SRR} is surpassed, see Figure 6.2.



(a) Circle scenario with pedestrian travelling from θ_{SRR} to $-\theta_{SRR}$.

(b) Circle scenario with pedestrian travelling from $-\theta_{SRR}$ to θ_{SRR}.

Figure 6.3: Circle pattern scenario with a pedestrian traveling between azimuth angle $\pm\theta_{SRR}$ in an arc with radius R_{c0} for evaluating the measurement error model.

6.1.4 The Detection Rate Scenario

To evaluate the detection rate model, the following five scenarios are designed, see Figure 6.4. The child pedestrian travels at a speed of 5 km/h while the host car is stationary. Scenario one and two starts outside the FOV and travels in front of the host car at a narrow range. In scenarios three and four, the target starts at radius $\frac{1}{2}R_{LRR}$ and travels towards the host at an azimuth angle slightly smaller than θ_{SRR} for scenario three, and at an angle slightly larger than $-\theta_{SRR}$ for scenario 4. Scenario five starts at a range slightly larger than R_{LRR} and travels radially toward the host at an azimuth angle of 0°.

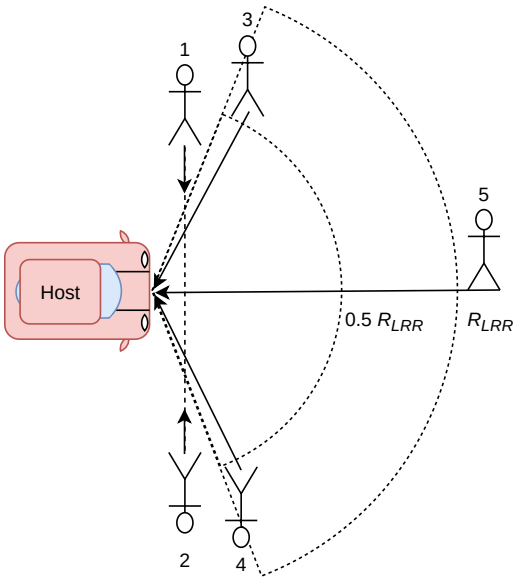


Figure 6.4: Detection rate scenarios for evaluating the detection rate model.

6.2 Evaluation of the Field of View Models

The new FOV models have been constructed to more accurately represent the real FLR FOV compared to the idealistic FOV in CSPAS. All three FOV models include more than 95 % of the detected FLR points and are therefore deemed accurate, according to the objectives stated in Section 1.2 Contributions, see Table 6.2

Table 6.2: Number of samples inside (N_{si}) and outside (N_{so}) the FOV models, the percentage % inside the FOV, and the symbolical variable for their areas A, with the total number of FLR detections $N_s = 42125$.

	Ideal	Convex	Concave	Polynomial
N_{si}	31748	42099	41731	41273
N_{so}	10377	26	394	852
%	75.37	99.94	99.06	97.98
A	A_{ideal}	A_{convex}	$A_{concave}$	$A_{polynomial}$

The convex model included more samples inside the FOV than both the concave and polynomial models. However, the convex model can not be assumed most optimal by only observing the number of detections inside the FOV. Given that the area of the models has the following relationship

$$A_{convex} > A_{concave} > A_{polynomial} > A_{ideal} \tag{6.1}$$

the convex model has the largest area and could therefore be overestimating the FOV by including area where detections have not been made. To evaluate which model that has the most accurate FOV shape the following method is used.

Assuming the samples are evenly distributed and using the area and number of samples inside the ideal model the density of samples (A_{ideal}/N_{ideal}) is calculated. With the density of samples, the expected number of samples N_e over an area A is calculated

$$N_e = \frac{A}{A_{ideal}} \cdot N_{si,ideal} \tag{6.2}$$

for the designed FOV models. The expected number of samples are presented in Table 6.3.

Table 6.3: Expected number of samples in FOV.

	Convex	Concave	Polynomial
N_e	59878.14	57220.10	57151.60

Calculating the ratio between the expected number of samples to the number of samples in the FOV models, one can see that the designed models include fewer samples than what is expected in Table 6.4.

Table 6.4: Ratio of samples inside the FOV and the expected number of samples.

	Convex	Concave	Polynomial
$\frac{N_{si}}{N_e}$	0.70	0.73	0.72

This is because the number of samples are not evenly distributed, instead, the FOV has a higher density of samples in the center of the FOV while the angular measurements and large ranges have a lower density. Therefore, the ideal model has a higher density of samples inside the FOV. However, from the ratio it can be observed that the concave FOV has the highest ratio closely followed by the polynomial model, meaning the area most likely has been decreased where samples are unlikely to be while still including a suitable amount of samples. From this, the concave could be deemed the most suitable FOV model.

6.2.1 Execution Time for the Field Of View Models

To evaluate the execution time the CPTC scenario, presented in Section 6.1.1 The Car-Pedestrian-Turning-Child Scenario, was simulated eleven times. As the execution time varied between executions, the mean μ_t value and standard deviation σ_t were calculated for each FOV model, see Table 6.5

Table 6.5: Execution time for the FOV models with the CTPC scenario.

	Farside						Nearside		
	10		15		20		10		v_h [km/h]
	same	opp	same	opp	same	opp	same	opp	direction
Ideal	5.690	5.474	3.928	4.611	3.123	3.024	5.698	6.016	μ_t [s]
	0.879	1.432	1.053	1.610	0.851	0.690	1.137	1.212	σ_t [s]
Convex	6.765	6.037	4.316	4.513	3.603	3.253	5.841	5.631	μ_t [s]
	1.018	1.498	1.148	1.286	0.874	0.556	0.869	0.610	σ_t [s]
Concave	6.254	5.298	4.090	3.748	2.939	3.035	5.583	5.918	μ_t [s]
	1.531	0.533	1.215	0.584	0.531	0.746	0.667	0.977	σ_t [s]
Polynomial	6.354	6.269	4.330	4.988	3.670	3.755	6.429	6.893	μ_t [s]
	1.381	1.716	1.046	1.886	1.082	0.792	1.272	2.701	σ_t [s]

The execution time is quite similar for all FOV models. The polynomial model has a larger variance for the majority of the scenarios compared to the other models. However, the mean values are fairly similar for all models. The concave model has a smaller execution time mean value compared to the convex model for all scenarios except one, which is not as expected. Knowing that the concave model is composed of more vertices (N_v) than the convex model, it should be more computationally complex, as presented in Section 2.2.1 Point In Polygon, and therefore have a longer execution time. The two models use their own PIP algorithms. The convex model uses the Sign-Off method which only works for convex polygons, while the concave model uses a combination of ray tracing and the Sign-Off method. Both solutions should have a computational complexity linear to the number of polygon vertices ($\mathcal{O}(N_v)$). However, to verify that the difference in PIP algorithms were not the factor affecting the execution time, another simulation was made with both polygon models using the concave PIP algorithm. This did not result in any noticeable change in execution time, and the concave model was still faster than the convex.

6.2.2 Repeatability for the Field Of View Models

To evaluate the repeatability of the FOV models, the number of detections for the CPTC scenarios made by each model was documented, see Table 6.6. All models resulted in the same number of detections for each simulation.

Table 6.6: Number of detections (N_d) made by the FOV models for the CPTC scenario during a total number of samples N_s.

	Farside						Nearside		
	10		15		20		10		v_h [km/h]
	same	opp	same	opp	same	opp	same	opp	direction
	1201	1201	801	801	601	601	1201	1201	N_s
Ideal	631	888	466	579	430	422	666	852	N_d
Convex	669	947	509	618	454	452	697	853	N_d
Concave	666	947	506	618	454	452	692	853	N_d
Polynomial	629	946	463	618	454	452	646	852	N_d

From this it is observed that the convex model has more detections in some scenarios of the CPTC, naturally because it has a larger area. However, in most scenarios, the convex model makes as many detections as the concave model and sometimes even all three designed models have the same amount of detections. However, the polynomial model sometimes has fewer detections compared to the ideal model.

6.3 Evaluation of the Measurement Error Model

As described in Section 4.2 Design of the Measurement Error Model, the measurement error model was designed to apply an error depending on the direction the target is traveling. If the target is traveling in a fan pattern, then the fan error should be applied, and if the target is traveling in a circle pattern, the circle error should be applied. To verify this, the fan and circle pattern scenarios, presented in Section 6.1.2 The Fan Pattern Scenario and Section 6.1.3 The Circle Pattern Scenario, respectively, was used. Both scenarios showed the measurement error model gave the correct error in each segment. The measurement error model was then evaluated using the CPTC scenario. The number of detections was not affected and the position of the object was moved according to Eq. (5.6). Whether this is correct or not can not be evaluated.

6.4 Evaluation of the Detection Rate Model

The detection rate model was evaluated using the detection rate scenario presented in Section 6.1.4 The Detection Rate Scenario. To determine repeatability nine randomly generated seeds were used, see Table 6.7.

Table 6.7: Randomly generated seeds used in repeated simulations of detection rate scenario.

Random seeds		
5718087	2565081	15285
5844280	4753345	6821664
1535973	8700903	3120416

The detection rate model was evaluated by comparing the difference in the parameters time t, distance q to the target at first detection, and number of detections N_d in the simulation. With the model on, the parameters varies depending on the seed. Therefore, the average parameter values were calculated and the difference between the parameters with the detection rate model on and off were calculated as Δt, Δq, and ΔN_d, see Table 6.8-6.9.

Table 6.8: Difference in time t, distance q and number of detections N_d between model on and off, for detection rate scenario 1 and 2.

	Scenario 1			Scenario 2		
	Δt	Δq	ΔN_d	Δt	Δq	ΔN_d
Ideal	0	0	0	0.011	0.015	1
Convex	0	0	0	0.011	0.015	1
Concave	0	0	0	0.014	0.019	1
Polynomial	0	0	0	0.089	0.123	4

For scenario 1, there is no difference since P_d is high for the position of first detection. For scenario 2 there is a slight difference, where the model misses one sample on average for *Ideal*, *Convex* and *Polynomial*, and 4 samples for *Polynomial*.

Table 6.9: Difference in time t, distance q and number of detections N_d between model on and off, for detection rate scenario 3, 4, and 5.

	Scenario 3			Scenario 4			Scenario 5		
	Δt	Δq	ΔN_d	Δt	Δq	ΔN_d	Δt	Δq	ΔN_d
Ideal	2.989	4.040	120	2.219	2.944	89	0	0	0
Convex	1.208	1.679	49	0.395	0.548	16	0	0	0
Concave	0.953	1.324	62	0.322	0.448	13	0	0	0
Polynomial	0.700	0.972	38	0.247	0.343	10	0	0	0

In scenarios 3 and 4, the model misses a lot of samples. This is expected, since the probability of detection is low close to the edges of the FOV. There is also a difference between the FOV models. *Ideal* has the greatest difference between model on and model off. For scenario 5, there is no difference since, just as for scenario 1, P_d is high for the position of first detection. Looking more specifically at scenarios 3 and 4, in Table 6.10, the range of first detection with model off is presented.

Table 6.10: Range first detection for scenarios 3 and 4 with model off, as factor of R_{LRR}.

Model off	Ideal	Convex	Concave	Polynomial
Scenario 3	0.50	0.41	0.39	0.38
Scenario 4	0.50	0.45	0.44	0.43

Since scenario 3 and 4 starts from $\frac{1}{2}R_{LRR}$, it can be seen that the ideal model instantly detects the target, this because the ideal model incorporates more visual field for angles close to $\pm\theta_{SRR}$ compared to the designed models. Therefore, the target needs to travel some distance before being detected with the other FOV models. This means that the starting probability of detection varies between the FOV models as the target is detected in different segments. The range of fist detection with the detection rate model on can be seen in Table 6.11.

Table 6.11: Range first detection for scenarios 3 and 4 with the model on, as factor of R_{LRR}.

Model on	Ideal	Convex	Concave	Polynomial
Scenario 3:	0.46	0.39	0.38	0.37
Scenario 4:	0.47	0.44	0.44	0.43

From this it can be seen that the model most affected by the detection rate model is the ideal model. However, with the detection rate model on the ideal model is able to detect the object further away than the designed models. The designed models give a similar result with and without the detection rate model.

7

Discussion

The following chapter discusses the radar sensor models and their evaluation. The collection of data measurements was already conducted at VCC. Therefore, an in-depth conclusion of how testing was conducted and how it can have affected the collected measurements can not be discussed.

7.1 Discussion of the Field Of View Models

Evaluation of the FOV models showed that the concave model was more accurate when regarding the ratio between the number of samples inside the FOV and the expected number of samples. However, given that data collection was only performed once, further testing on the radar might prove that the radar is able to make detections in areas where there currently is a low probability. The concave model was also faster when comparing the execution time between the designed models. This was not expected since the concave polynomial consists of more vertices and should therefore be more computationally complex. The ideal model should be deterministic and the execution time should therefore not have any variance between simulations. Yet, it was found that even the ideal model included some variance in execution time between simulations as well. Furthermore, in some scenarios the mean execution time and variance for the ideal model were larger than the designed FOV models, suggesting something else could be affecting the execution time, such as background updates and computer capacity. However, since the variance in execution time is quite small compared to the mean execution time, and did not increase drastically for the FOV models compared to the ideal model, all the designed FOV models are deemed as having passed the execution time criteria set in Section 1.2 Contributions. When evaluating the number of detections made by the models for the CPTC scenario, the convex model was slightly better than the concave, which is to be excepted as it has a larger area.

7.2 Discussion of the Measurement Error Model

The measurement error was calculated by subtracting the FLR detected position with the robot position from the GPS, Eq. (3.4). This means that the measurement error calculation is greatly affected by the accuracy of the GPS. The accuracy of the GPS has not been taken into consideration when designing the measurement

error model. As was shown in Section 3.2.2 Analysis of the Measurement Error the fan test pattern had an increasing distance error for a decreasing azimuth angle. Knowing that testing started at azimuth angle θ_{SRR} and was decreased according to Eq. (3.1), the increased error could be from a systematical addition of error over the measurements, meaning the radar might need re-calibration after some time period. The circle pattern had a small variance in error for measurements close to the host car, and an increased variance for large azimuth angles $|\theta|$. However, no direct connection between the magnitude or variance of the error and the angle or radius could be seen.

Since the collection of data measurements was only conducted by two tests, the fan and circle pattern as presented in Section 3.1 Collection of Data Measurements, the measurement error model was designed as two models. The error was then calculated depending on the direction the target is traveling. If a target is traveling in a direction dissimilar to the fan or circle pattern it is unknown what the FLR sensor error would be. Therefore, this model approximates the error based on the fan and circle data measurements.

7.3 Discussion of the Detection Rate Model

Evaluation of the detection rate model showed that the model did not make any difference for detection rate scenarios 1 and 5. Considering that the FOV model limits what the detection rate model can detect, a larger or unlimited FOV might give a different outcome. This behavior was also observed for scenarios 3 and 4, where with the detection rate model turned off, the ideal FOV had the first detection at a range larger than any of the other FOV models with the model turned on. This means that the FOV model can in some sense limit the detection rate model, in an undesirable way. It could therefore be considered to use the detection rate model in place of any of the FOV models, to avoid this limitation of the detection rate model caused by the FOV model.

However, the detection probability is 0 % if either there are no detections in the segment, or if there are no samples at all. When data is missing completely, no conclusions can be drawn. Either the model can ignore this segment, regarding it as always detected, and trusting the FOV model, or it can be kept at 0 % so that the FOV is effectively reduced by a small amount. In case data for many segments are missing, this is not desirable, since it would create undetectable gaps. By always detecting those segments, it leaves the decision completely to the FOV model, which means such a lacking detection rate model should not be used in place of the FOV model.

Furthermore, the detection rate model assumes that the radar has perfect tracking, which is not true. There is the possibility that the radar loses its tracking of the target, and the probability and conditions for when this can happen are currently unknown.

8

Conclusion

In this chapter, the conclusions that could be drawn from the result are presented along with the social, ethical, and ecological aspects of the thesis, as well as what can be done in future work.

8.1 Summary of results

To summarize the findings from the evaluation and discussion, the concave FOV model was found to be the most optimal when considering its execution time and the FOV area. The convex model had more detections as a result of a larger area which could be consisting of areas where detections have not been made. The convex model was also more computationally complex compared to the concave. The polynomial model had fewer detections and for some scenarios, even fewer detections than the ideal model, making it not as optimal. The measurement error model did not affect the number of detections made in the CPTC scenario but did apply an accurate error when evaluating the fan and circle pattern scenarios. For other scenarios, the measurement error model can not be validated without further collection of data measurements from the radar. The detection rate model produces a simulation result that is repeatable given the same seed as input. The model made the largest difference for the fan test patterns close to the FOV edges. Finally, it was found that the FOV model can limit the detection rate model.

8.2 Social, Ethical, and Ecological Aspects

As presented in Section 1.1 Background, more than 90% of all road accidents are caused by the driver [4]. In [5, p. 86] was theoretically deduced that implementation of ADAS functions could prevent these accidents and can therefore have a great social benefit by reducing car crashes caused by human error. However, having these functions in the car could also lead to the driver relying more on the functions alerting when a collision is near. The traditional way of verifying ADAS functions on a test track is as previously mentioned in Section 1.1 Background infeasible as one would need to test drive for billions of kilometers [6]. Verifying the functions in a virtual environment is therefore the only option. Performing testing and verification in a virtual environment will inevitably save fuel and the number of cars produced for testing on a test track, which will reduce the amount of greenhouse gas emissions.

However, relying more on the simulations than real-world testing puts pressure on the simulation to be valid, where faults within the simulation environment will have serious consequences for the function performance. If the virtual simulation does not provide a trustworthy simulation, the functions might not work as intended when introduced into the systems of a real car. Developing a realistic sensor model using data-driven models will provide the virtual simulation with a more trustworthy simulation.

Developing a realistic sensor model can have further ethical aspects. A car application of this sort has high-performance requirements, meaning that it must be valid in highly dynamic environments. Other applications can utilize radar sensors, such as drones, missiles, aircraft, and other weapons that are operating in highly dynamic conditions. These applications do not have the same moral and ethical purpose as a collision avoidance car application does. Although not developed for military applications, the model could still be vulnerable as it can be stolen and/or re-purposed for non-intended applications. Another aspect of vulnerability is that the sensor model can be exploited, such as inducing a malfunction. A perpetrator that has access to the model and knows scenarios where the model fails, can study the model and learn its weaknesses.

8.3 Future work

The models presented in this thesis were designed from data that only contains measurements from one conducted test of the FLR radar according to the specifications presented in Section 3.1 Collection of Data Measurements. Having a larger data set, with a higher resolution grid, could further improve the accuracy of the radar sensor models. Including more data measurements can also help validate or alter the measurement error model by providing more samples. Furthermore, by introducing other driving patterns than the fan and circle, the calculation of the measurement error as presented in Section 4.2 Design of the Measurement Error Model can be evaluated or altered to a more suitable equation. For example, data collection of a diagonal or cross pattern test can be carried out, to be able to both validate the findings in this work and further improve upon the presented models.

The testing conducted at VCC was of a robot mounted with a child pedestrian doll as presented in Section 3.1 Collection of Data Measurements. In [1] it was suggested that the radar can be affected by the object. Performing data collection with other objects can further improve the radar model and object-dependent models can be derived, which will further improve the accuracy of the radar model.

The detection rate model assumes a perfect tracking algorithm which is not accurate for a real radar. To further improve the radar model it would be needed to further investigate the tracking of the real radar and what causes a radar to lose the detected object while it is still inside the FOV. As CSPAS mostly consisted of 2D algorithms the radar models presented in this thesis were designed in 2D, to further improve the CSPAS framework the models should be designed in 3D.

Bibliography

[1] S. Muckenhuber, H. Holzer, J. Rübsam, and G. Stettinger, "Object-based sensor model for virtual testing of ADAS/AD functions," in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 1–6, 2019.

[2] M. Stolz and G. Nestlinger, "Fast generic sensor models for testing highly automated vehicles in simulation," *e & i Elektrotechnik und Informationstechnik*, vol. 135, pp. 365–369, 2018.

[3] "European New Car Assessment Programme." `https://cdn.euroncap.com/media/75436/euro-ncap-aeb-lss-vru-test-protocol-v43.pdf`, 2022. (Accessed 12-04-2023).

[4] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," Traffic Safety Facts Crash • Stats Report No. DOT HS 812 506, Washington, DC: National Highway Traffic Safety Administration, March 2018. `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812506` (Accessed: 01-03-2023).

[5] H. Winner, S. Hakuli, F. Lotz, and C. Singer, *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Springer Reference, Springer, 2016. `https://doi.org/10.1007/978-3-319-12352-3` (Accessed: 01-03-2023).

[6] B. Schlager, S. Muckenhuber, S. Schmidt, H. Holzer, R. Rott, F. Maier, K. Saad, M. Kirchengast, G. Stettinger, D. Watzenig, and J. Ruebsam, "State-of-the-art sensor models for virtual testing of advanced driver assistance systems/autonomous driving functions," *SAE International Journal of Connected and Automated Vehicles*, vol. 3, pp. 233–261, 2020. `https://doi.org/10.4271/12-03-03-0018` (Accessed: 01-03-2023).

[7] Volvo Cars, "CSPAS Overview." (Accessed: 31-01-2023).

[8] Volvo Cars, "Specification of radar in CarWeaver." (Accessed: 30-01-2023).

[9] "open_simulation_interface: Open simulation interface (osi)." `https://opensimulationinterface.github.io/open-simulation-interface/index.html`. (Accessed 01-06-2023).

[10] "open_simulation_interface: osi3::GroundTruth Struct Reference — opensimulationinterface.github.io." https://opensimulationinterface.github.io/open-simulation-interface/structosi3_1_1GroundTruth.html. (Accessed 31-01-2023).

[11] "open_simulation_interface: osi3::SensorData Struct Reference." https://opensimulationinterface.github.io/open-simulation-interface/structosi3_1_1SensorData.html.

[12] "Radar - Wikipedia — en.wikipedia.org." https://en.wikipedia.org/wiki/Radar. (Accessed 02-03-2023).

[13] "How RADARs work — thinkautonomous.ai." https://www.thinkautonomous.ai/blog/how-radars-work/ (Accessed 02-03-2023).

[14] M. Schneider, "Automotive Radar – Status and Trends," pp. 144–147, 2005. (Accessed 02-03-2023).

[15] A. Topiwala, "Is the Point Inside the Polygon? — towardsdatascience.com." https://towardsdatascience.com/is-the-point-inside-the-polygon-574b86472119, 2020. (Accessed 06-04-2023).

[16] C.-W. Huang and T.-Y. Shih, "On the complexity of point-in-polygon algorithms," *Computers & Geosciences*, vol. 23, no. 1, pp. 109–118, 1997.

[17] M. El-Salamony and A. Guaily, "Enhanced modified-polygon method for point-in-polygon problem," in *Recent Advances in Engineering Mathematics and Physics* (M. H. Farouk and M. A. Hassanein, eds.), (Cham), pp. 47–61, Springer International Publishing, 2020.

[18] "Geometry - Point in convex polygon - Competitive Programming | INGInious — inginious.org." https://inginious.org/course/competitive-programming/geometry-pointinconvex. (Accessed 14-04-2023).

[19] S. H. Ahn, "Line Equation — songho.ca." http://www.songho.ca/math/line/line.html. (Accessed 13-04-2023).

[20] Volvo Cars, "Specification in SystemWeaver Fan test." (Accessed: 30-01-2023).

[21] Volvo Cars, "Specification in SystemWeaver Circle test." (Accessed: 30-01-2023).

[22] "Box plot - Wikipedia — en.wikipedia.org." https://en.wikipedia.org/wiki/Box_plot. (Accessed 27-05-2023).

[23] J. Ryden, *Stokastik för ingenjörer*. 2015. (Accessed: 25-04-2023).

[24] "Collinear Points - Definition, Formula, Examples — cuemath.com." https://www.cuemath.com/geometry/collinear-points/. (Accessed 27-05-2023).

A

Appendix

A.1 Number of samples in each segment

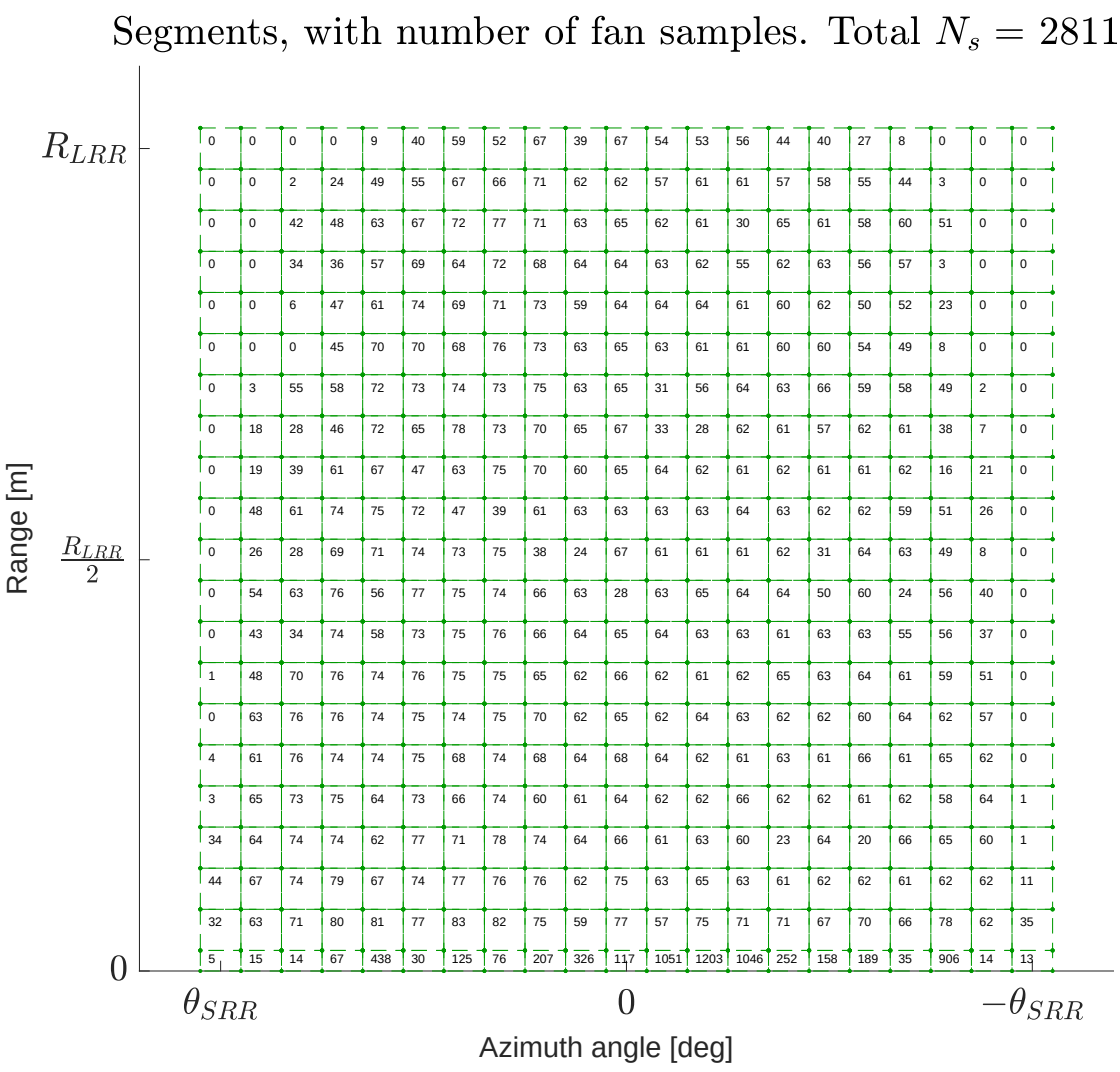


Figure A.1: Number of samples in each segment of the fan test.

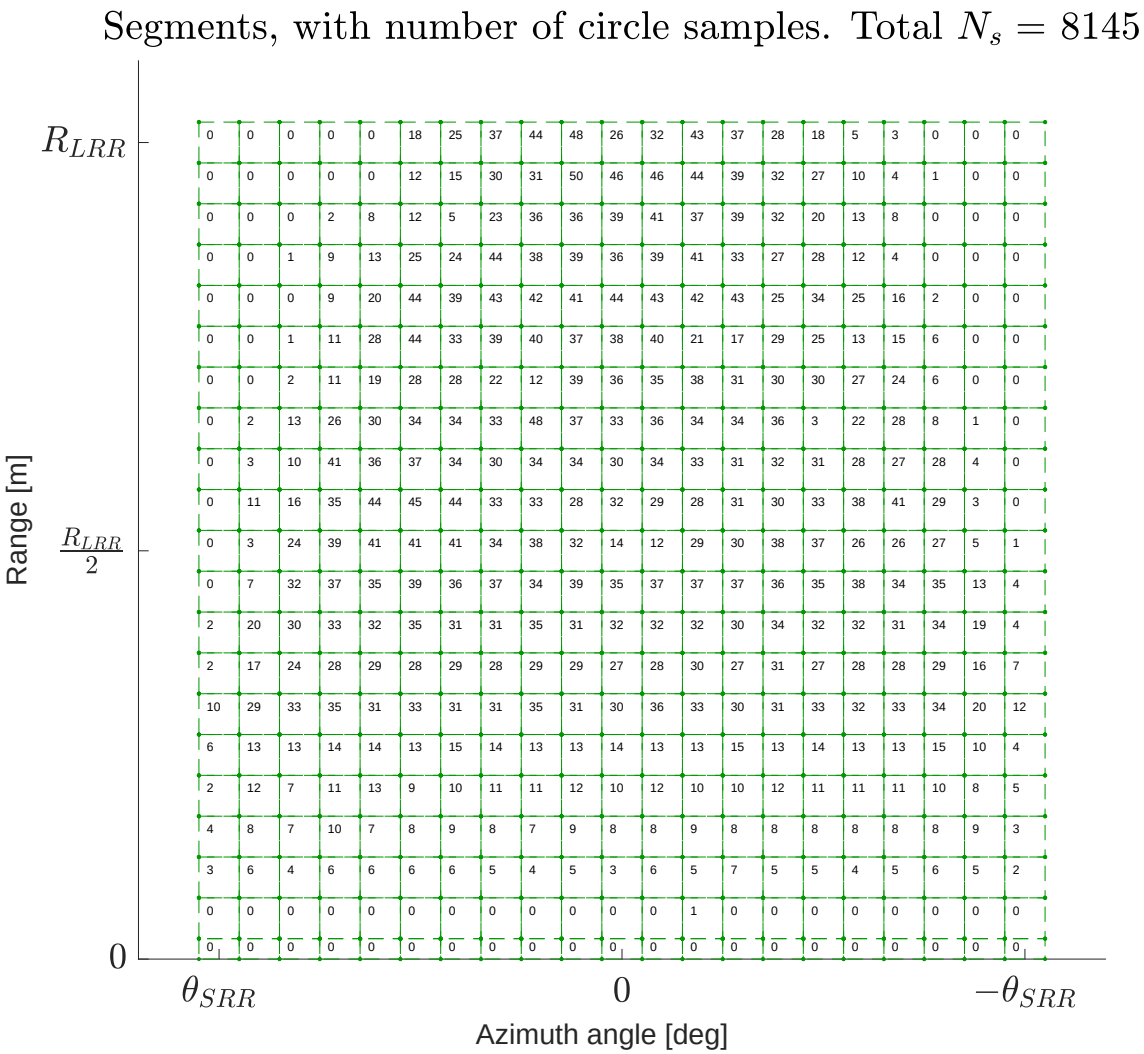
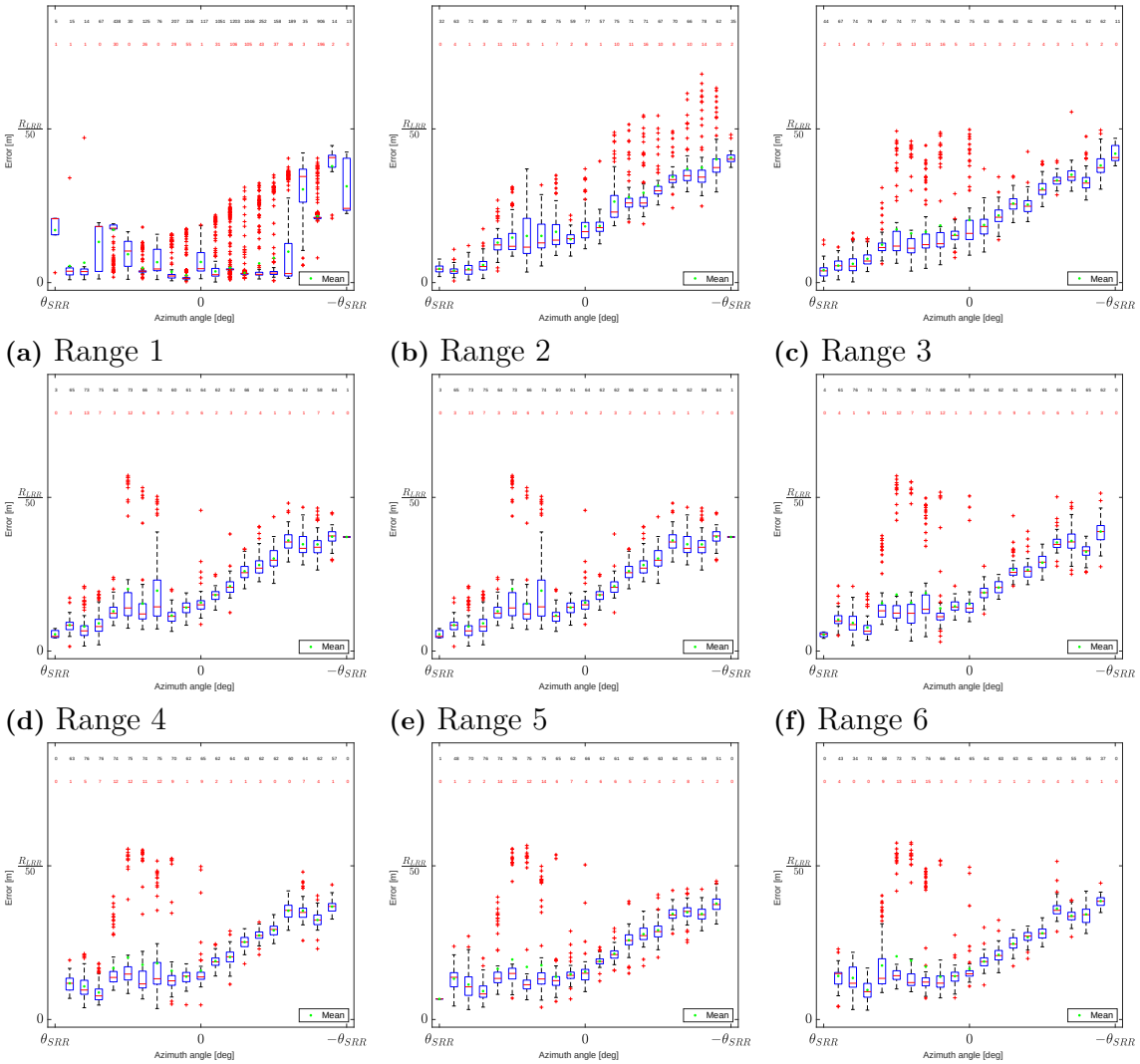


Figure A.2: Number of samples in each segment of the circle test.

A.2 Box plots for measurement error

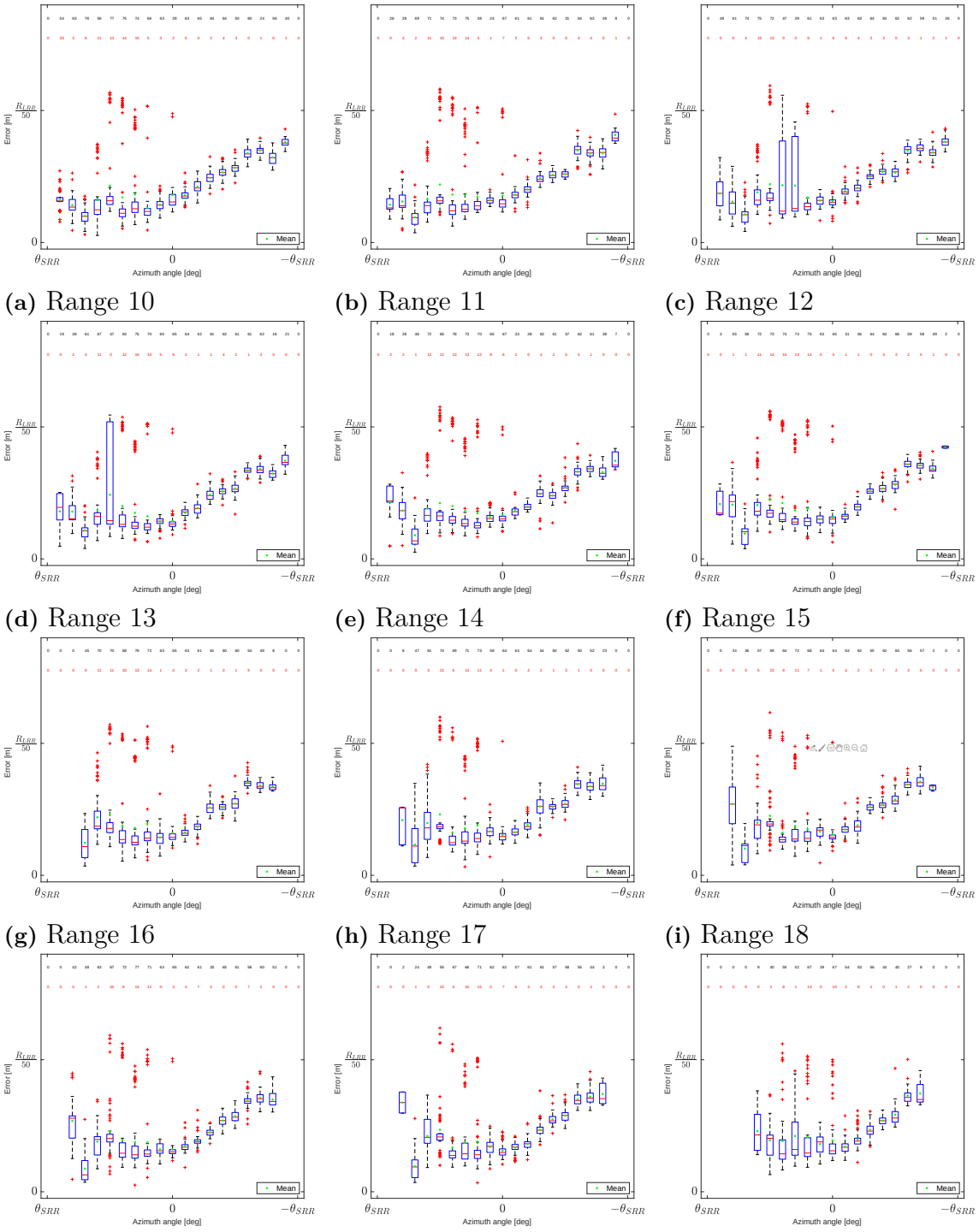


(a) Range 1 **(b)** Range 2 **(c)** Range 3

(d) Range 4 **(e)** Range 5 **(f)** Range 6

(g) Range 7 **(h)** Range 8 **(i)** Range 9

Figure A.3: Fan to show the spread of the error for each range segment (1-9), the number of samples for each angle can be seen at the top of the images.



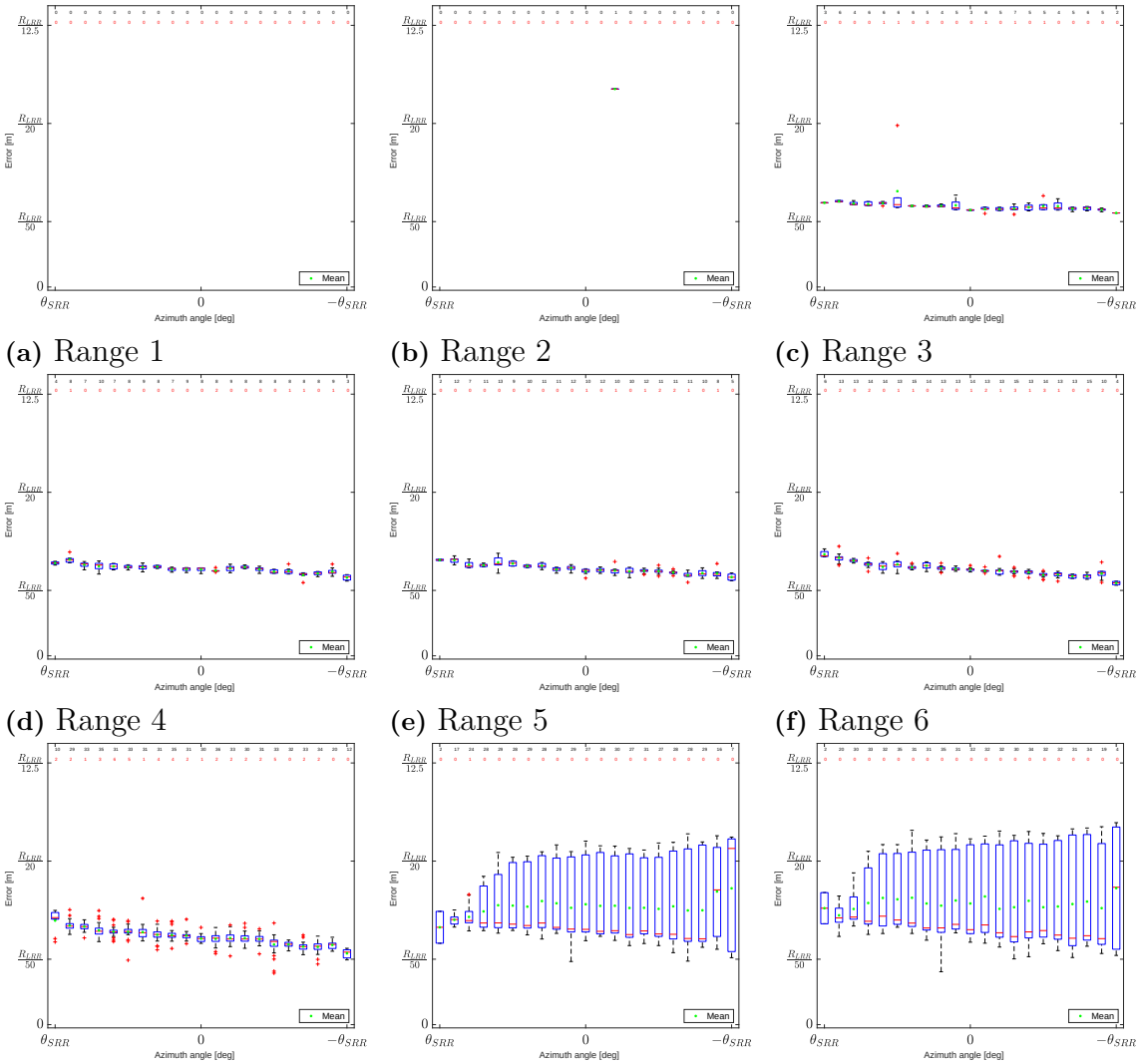
(a) Range 10 **(b)** Range 11 **(c)** Range 12

(d) Range 13 **(e)** Range 14 **(f)** Range 15

(g) Range 16 **(h)** Range 17 **(i)** Range 18

(j) Range 19 **(k)** Range 20 **(l)** Range 21

Figure A.4: Fan to show the spread of the error for each range segment (10-21), the number of samples for each angle can be seen at the top of the images.

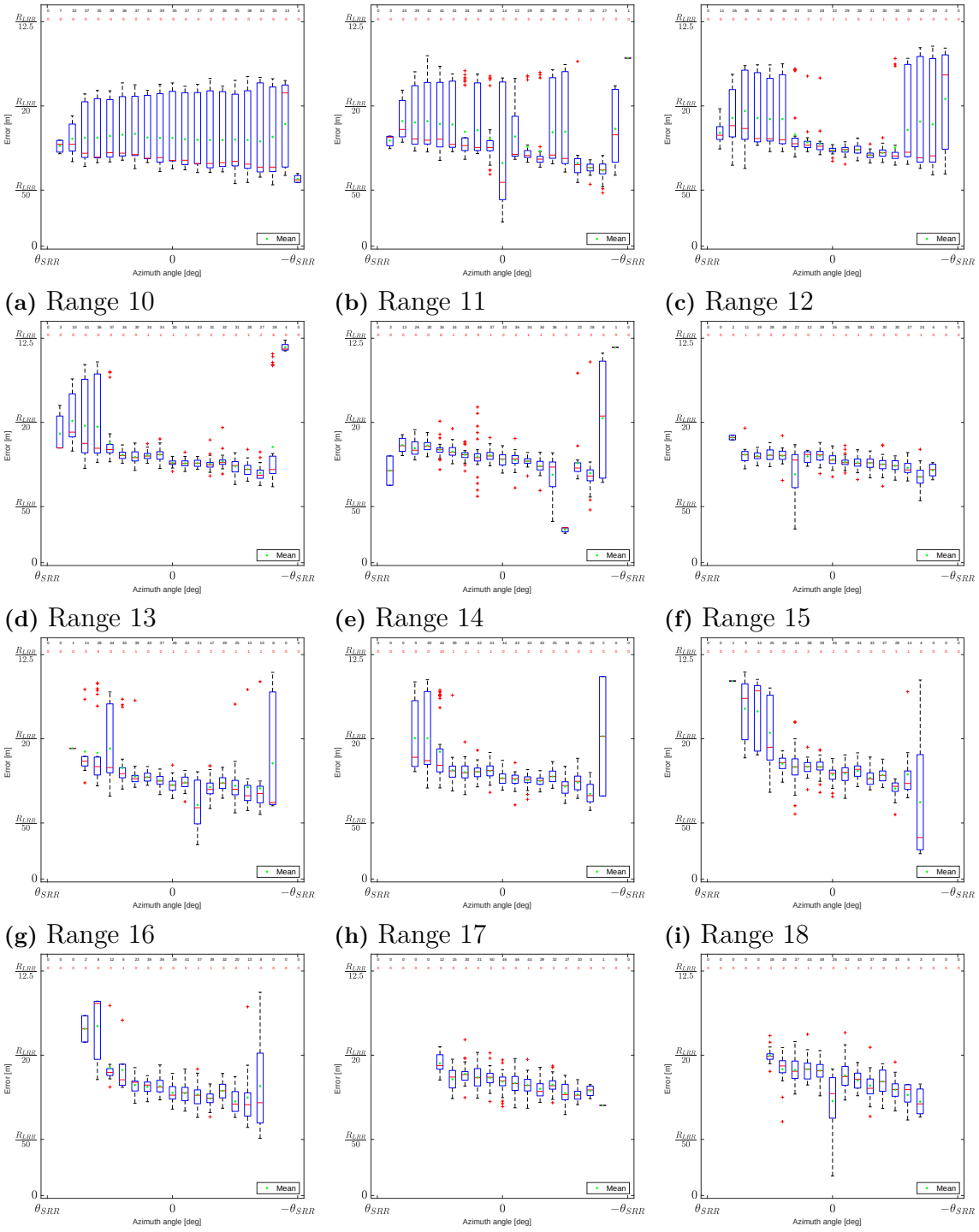


(a) Range 1 **(b)** Range 2 **(c)** Range 3

(d) Range 4 **(e)** Range 5 **(f)** Range 6

(g) Range 7 **(h)** Range 8 **(i)** Range 9

Figure A.5: Cir to show the spread of the error for each range segment (1-9), the number of samples for each angle can be seen at the top of the images.



(a) Range 10 (b) Range 11 (c) Range 12

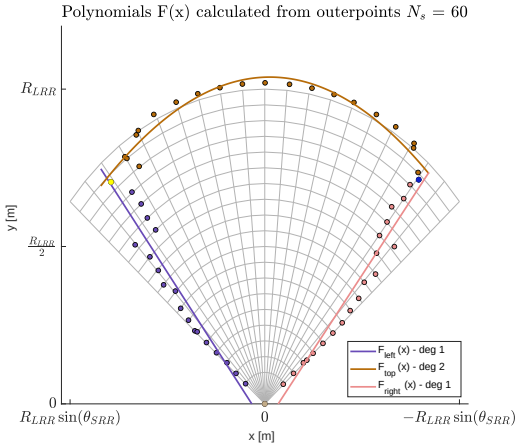
(d) Range 13 (e) Range 14 (f) Range 15

(g) Range 16 (h) Range 17 (i) Range 18

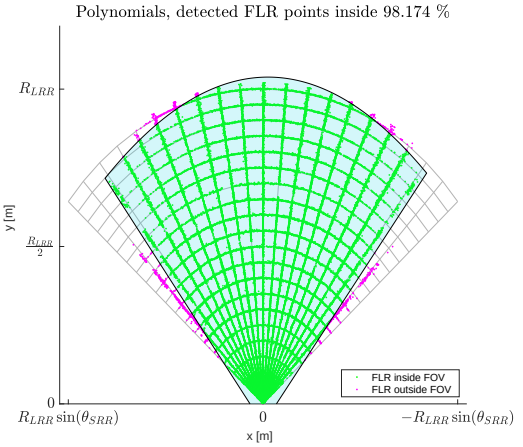
(j) Range 19 (k) Range 20 (l) Range 21

Figure A.6: Cir to show the spread of the error for each range segment (10-21), the number of samples for each angle can be seen at the top of the images.

A.3 Polynomial

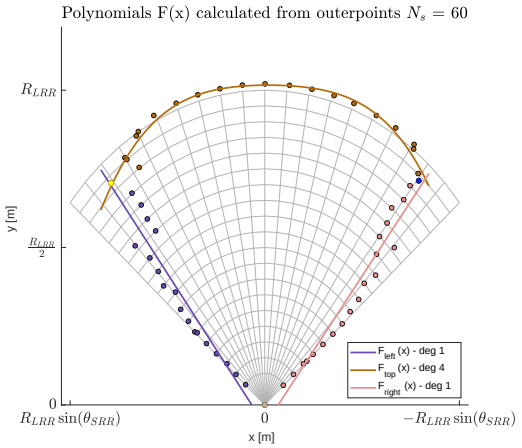


(a) Polynomial.

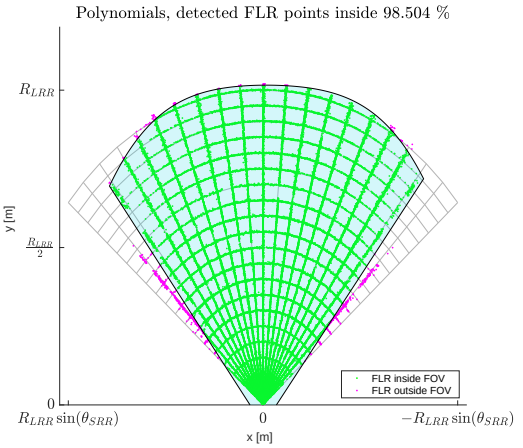


(b) Polynomial performance.

Figure A.7: Polynomial of degree left 1, top 2, right 1 was not found accurate for the origin point as it includes to much outside the detected FOV, the top expression also includes too much at the center.

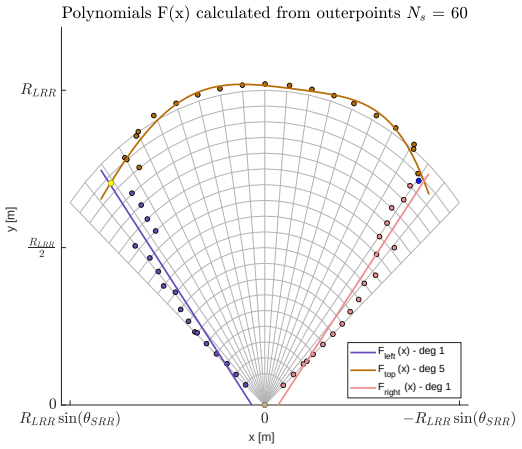


(a) Polynomial.

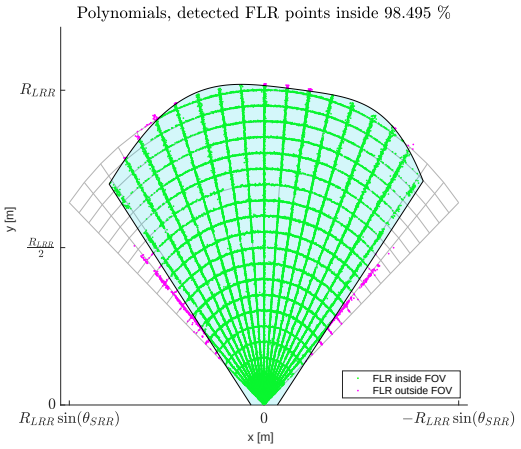


(b) Polynomial performance.

Figure A.8: Polynomial of degree left 1, top 4, right 1 was found to be a better fit than the previous top function while the percentage of points inside the FOV has not been changed.

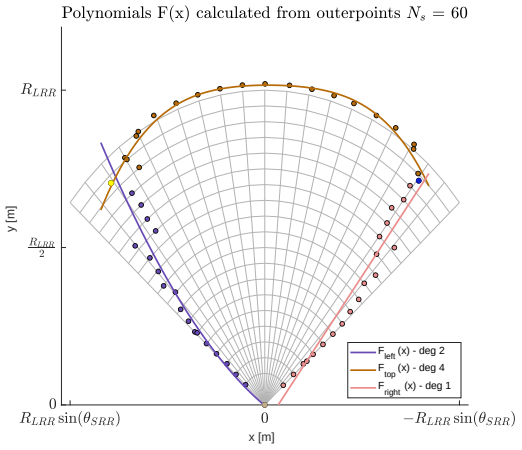


(a) Polynomial.

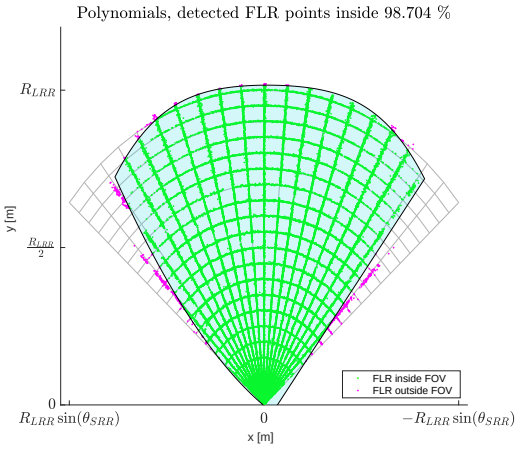


(b) Polynomial performance.

Figure A.9: Polynomial of degree left 1, top 5, right 1 was found to be a good choice as the shape of the top function is not realistic and did not affect the percentage which is why the top function set as 4 was found adequate.

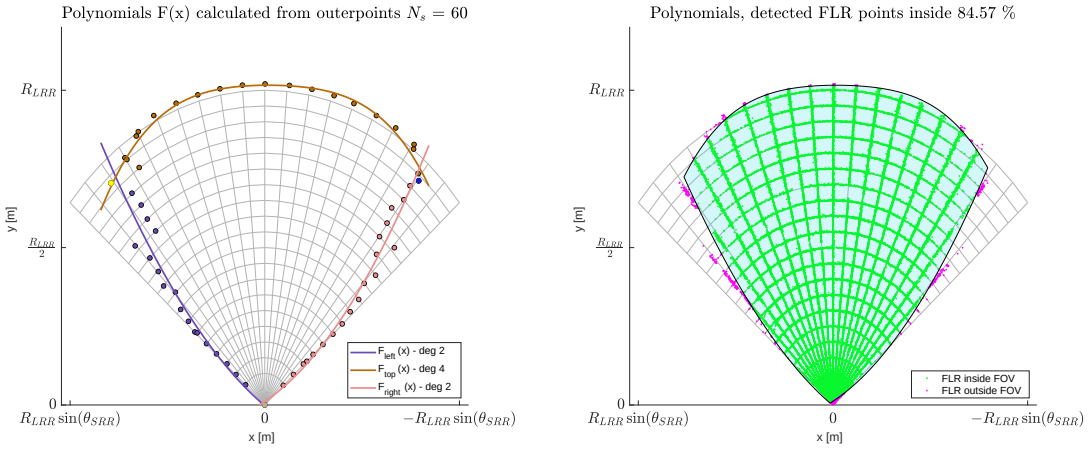


(a) Polynomial.



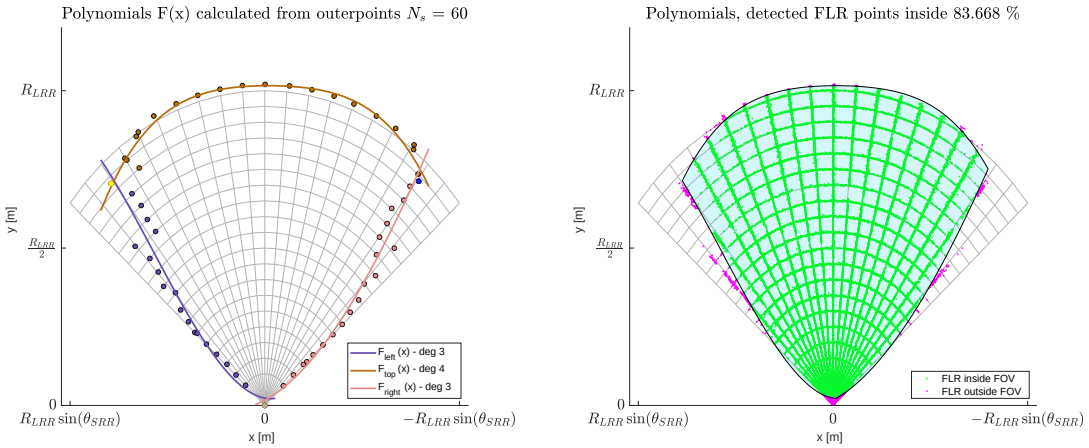
(b) Polynomial performance.

Figure A.10: Polynomial of degree left 2, top 4, right 1 was found to generate a better FOV than 141 as it does not include too much space outside the detected area at the origin.



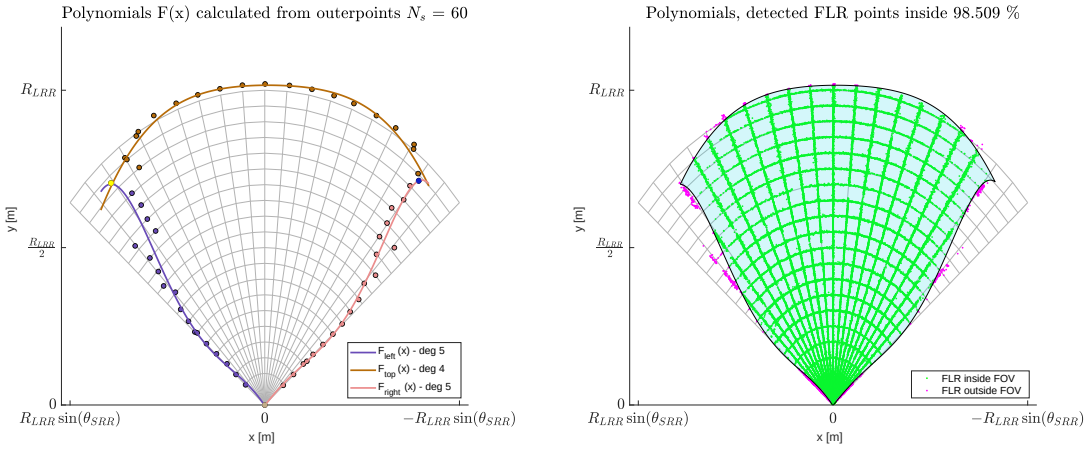
(a) Polynomial. **(b)** Polynomial performance.

Figure A.11: Polynomial of degree left 2, top 4, right 2 was found to be more accurate at the origin, however, it was found to exclude too many detected points.



(a) Polynomial. **(b)** Polynomial performance.

Figure A.12: Polynomial of degree left 3, top 4, right 3 was found to exclude too many detected points at the origin.



(a) Polynomial. **(b)** Polynomial performance.

Figure A.13: Polynomial of degree left 5, top 4, right 5 was found to be accurate while excluding some points at the origin it still includes enough overall.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY