

Kalle Paasio

APPLICATION OF RADAR SENSING IN KOVILTA OY SENSOR FRAMEWORK

Faculty of Information Technology and Communication Sciences
Master's thesis
Jukka Vanhala
Karri Palovuori
January 2024

ABSTRACT

Kalle Paasio: Application of radar sensing in Kovilta oy sensor framework
Master's thesis
Tampere University
Electrical Engineering
January 2024

This work explores radar sensing for sensor fusion applications with a camera. It addresses the fundamentals of radar, sensor fusion, and filtering. These fundamentals are put into practice in the implementation phase, where a physical system is constructed. The purpose of this work is to explore for Kovilta Oy a previously uncharted territory of radar sensing. The demonstration platform provides a real-world view into data formats, algorithms, radio-frequency related phenomena, and communication interfaces. It also serves as a basis for further in-house development being the first, and thus a foundational element, of Kovilta Oy's radar environment.

The implemented system is capable of tracking multiple humans in a room while continuously calculating their speed and position vectors. When applied to vehicular tracking, it can distinguish people, cars, cyclists and stop signs. In the conducted tests, the system was able to track cars from a distance of up to 50 meters. The novel sensor fusion algorithm also has the ability to scale its reliability with better radar platforms and with increasing computation power. Systems like this can be used as guidance for simultaneous localization and mapping, advanced driver assistance, or as standalone object-level monitoring systems. In particular, Kovilta Oy is interested in applying these systems to autonomous vehicles such as agricultural drones and self-driving cars.

In the final chapters, avenues for further exploration are presented including alternative approaches for sensor fusion, new radar modulation schemes, and ways to utilize machine learning on radar data. The proceedings of this work are made Open Source and are available on the author's GitHub under the MIT license.

Keywords: Radar, camera, sensor fusion, projection-based, ADAS, SLAM

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TIIVISTELMÄ

Kalle Paasio: Katsaus tutkan käyttömahdollisuuksiin Kovilta oy:n sensoriympäristössä
Maisterin tutkinto
Tampereen yliopisto
Sähkötekniikka
Tammikuu 2024

Tämä työ tutkii tutka havainnointia sekä tutkan yhteensopivuutta sensorifuusioon kameran kanssa. Työssä käydään läpi tutkan, sensori fuusion sekä filteröinnin perusteet. Tutkimuksen ohessa rakennetaan laite, jolla tekniikoita havainnoidaan. Tämän työn tavoite on selvittää Kovilta Oy:lle aikaisemmin tutkimatonta sensori ratkaisua. Rakennetun laitteen tarkoitus on ohjata kehitystä esille tuomalla systeemin toteutuskohtaisia ominaisuuksia, kuten data formaatteja, algoritmeja, radio etenemisen vaikutuksia sekä kommunikaatio rajapintoja. Koska työ on ensimmäinen Kovilta Oy:n tutka implementaatio, toimii se myös pohjana tulevaisuuden tutka kehitykselle.

Toteutettu laite voi saman aikaisesti seurata useita ihmisiä toimisto-olosuhteissa laskien heidän paikka- ja nopeusvektorinsa. Ajoneuvo ympäristössä laite erottaa toisistaan autot, ihmiset, pyöräilijät ja stop-merkit. Suoritetuissa mittauksissa laiteella voitiin seurata autoa 50 metrin etäisyyteen asti. Työssä muodostettu sensori fuusio algoritmi kykenee lisäksi kasvattamaan luotettavuuttaan sekä suorituskykyään, kun sille annetaan suorituskykyisempiä sensoreita tai prosessoreita. Tämän kaltaisia laitteita voidaan käyttää apuna autonomisessa navigoinnissa, kuljettaja-avustus järjestelmissä tai erillisissä seuranta tarkoituksissa. Erityisesti Kovilta Oy on kiinnostunut laitteen käytettävyydestä maatalous dronien sekä itse ajavien autojen osa-alueilla.

Tulosten esittämisen jälkeen työ käy läpi tulevaisuuden kehitys mahdollisuuksia sensori fuusion, tutka modulaation ja teko älyn hyödyntämisen osa-alueilla. Työn tulokset jaetaan avoimen lähdekoodin periaatteiden mukaan GitHub:ssa MIT lisenssin alaisina.

Avainsanat: Tutka, kamera, sensori fuusio, projektio, tekoäly

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

PREFACE

I would like to thank Kovilta Oy for the chance to conduct this hands-on exploratory survey of this previously uncharted sensor configuration.

Special thanks go to Eero Lehtonen who was influential in figuring out projection mathematics.

This work was performed autonomously by the author with guidance regarding next milestones from Kovilta Oy. The work had an exploratory nature due to the lack of previous knowledge about radar and had therefore a very wide scope. Eero Lehtonen and Mika Laiho directed this work. They gave goals that were to be completed before moving to next goals. We called these milestones, and I directed my work efforts with them. A total of 4 milestones were achieved. The university did not interfere with the work and gave only final remarks on the writing. As the only coder and newly made radar expert, I had a strong influence also on setting the milestones. The mathematics of projection were the only large part in which I had to ask for help. The idea of using projection as a basis of fusion was also from Eero. All code, filtering, hardware, mechanicals, research, and measurements were done by the author.

Tampere, 3.1.2024

Kalle Paasio

CONTENTS

1. INTRODUCTION.....	1
2. THEORY	2
2.1 FMCW MIMO Radar.....	2
2.2 Frequency Modulated Continuous Wave	4
2.3 Continuous False Alarm Rate	13
2.4 Angle of arrival.....	15
2.5 Sensor fusion	18
2.6 Kalman filtering.....	21
2.7 Simultaneous Localization And Mapping.....	26
2.7.1 Function of SLAM in vehicle context.....	26
2.7.2 How SLAM systems work	27
2.7.3 Challenges of SLAM.....	29
2.7.4 Examples of SLAM systems	30
3. PLATFORM	31
3.1 Architecture of radar SoC	31
3.2 Code resources	32
3.2.1 Embedded system code	32
3.2.2 PC side code	35
4. IMPLEMENTATION	37
4.1 Dense data	37
4.2 Sparse data	43
4.3 Object detection, tracking, and sensor fusion	46
4.4 3D representation using complementary sensor aspects.....	49
5. RESULTS	56
5.1 System performance.....	56
5.2 Sensor fusion	57
5.3 Kalman filtering.....	62
5.4 Calibration	65
5.5 Vehicular tracking	67
5.6 Inherent features of radar sensing	70
6. WHAT COULD BE IMPROVED	74
6.1 Optimization of chirp parameters.....	74
6.2 Alternative fusion approach	74
6.3 Digital Code Modulation	76
6.4 Radar based classification.....	78
6.5 Attention.....	79
6.6 Adaptive waveform	80
7. CONCLUSIONS.....	82
8. REFERENCES.....	84

LIST OF SYMBOLS AND ABBREVIATIONS

ADAS	Advanced Driver-Assistance System
ADC	Analog to digital converter
AI	Artificial intelligence
AoA	Angle of arrival
AOP	Antennas on package
API	Application programming interface
ASCII	American Standard Code for Information Interchange
BEV	Bird's eye view
CAN	Controller Area Network
CCS	Code composer studio
CEO	Chief executive officer
CFAR	Constant false alarm rate
CLI	Command line interface
COCO	Common objects in context
COM	Communication
CPU	Central processing unit
DCM	Digital code modulation
DPC	Data path chain
DPM	Data path manager
DSP	Digital signal processor
DSS	DSP subsystem
EDMA	Enhanced direct memory access
EKF	Extended Kalman filter
EVM	Evaluation module
FFT	Fast Fourier transform
FIFO	First in first out
FMCW	Frequency modulated continuous wave
FoV	Field of view
fps	Frames per second
FW	Front view
GPS	Global positioning system
GPU	Graphics processing unit
HCR	High Contrast Resolution
HWA	Hardware accelerator
ID	Identification
IDE	Integrated development environment
IEEE	Institute of Electrical and Electronics Engineers
IF	Intermediate frequency
IPC	Interprocess communication
I-Q	In-phase - quadrature modulation
IR	Infra-red
ISF	Interference Susceptibility Factor
LIDAR	Light Detection and Ranging
LoC	Lines of code
LVDS	Low voltage differential signaling
MIMO	Multiple input multiple output
ML	Machine learning
MSS	Master subsystem
ORB	Oriented fast and rotated brief

PA	Power amplifier
PC	Personal computer
PCB	Printed circuit board
PLA	Polylactic acid
PMCW	Phase modulated continuous wave
RF	Radiofrequency
RGB	Red-green-blue pixelated camera
RGBD	Red-green-blue-depth camera
RX	Receiver
SDK	Software development kit
SIFT	Scale-invariant feature transform
SLAM	simultaneous localization and mapping
SOC	System on a chip
STFT	Short-time Fourier transform
SURF	Speeded up robust features
TI	Texas instruments
TLV	Type Length Value
TX	Transmitter
UART	Universal asynchronous receiver-transmitter
UKF	Unscented Kalman Filter
USB	Universal serial bus
YOLO	You only look once

1. INTRODUCTION

Radar sensing is becoming more prominent in the automotive industry by the year. The number of radar sensors installed on vehicles is expected to rise to cover more than just adaptive cruise control, short-range collision avoidance, and automatic emergency braking applications employed in many systems today. The charm of radar has to do with its robustness to adverse weather conditions. Where camera and lidar sensors can be covered by heavy rain, snow, or fog, radar will see through these light barriers and continue to produce reliable data. Radar is therefore often used to provide sensor redundancy. Radar measurements inherently associate both range and velocity components which separates it from lidar in which only the range is measured. This allows more measurement inputs to be fed to the following state approximation filters and thus makes the system more observable.

In this work, we cover first the theoretical operation of the subsystems associated with the implementation. We then take a look at the hardware, software, and physical resources used in the making of the system. The decision process leading to the design choices is explained in detail. With the platform locked in the work moves to the implementation phase which was split into 4 milestones that made incremental improvements to the overall system performance and understanding of both sensor fusion algorithms and radar as a sensor. The results section covers the performance, accuracy, and successfulness of the different parts implemented in this system. At the end of this section, the author addresses the often-silent form of knowledge obtained by working with the chosen radar sensor. After the results obtained inside the scope of this master's thesis, the work proposes avenues of further exploration that address the newest approaches and algorithms employed in state-of-the-art radar systems.

2. THEORY

In this section, we have a look at the constituent parts needed to make sense of the system implemented later on. The subchapters cover the main physics, algorithms, and ways to think about the underlying problems found in radar-to-camera fusion. The intent is to provide enough information for the reader to comprehend the basics of the sensors used and the intended end application.

2.1 FMCW MIMO Radar

Radar is a sensor that can measure the distance to an object as well as its velocity perpendicular to the sensor. It achieves this by calculating the time-of-flight that it takes a signal to propagate from the transmit antenna to the object and back to the receive antenna. In automotive radar applications, the transmitter and receiver are often in close proximity to one another. The speed of the wave traveling in the medium of air with a dielectric constant of 1.0006 is approximated well with the speed of light in a vacuum. The roundtrip of the propagation is equal to 2 times the length of the trip and thus the time-of-flight can be calculated as

$$\tau = \frac{2d}{c}, \quad (1)$$

where d is the distance to the target and c is the speed of light in vacuum [1].

The signal used in transmission is often a sine wave pulse of known frequency. When the pulse bounces back from a stationary subject the pulse shape returning from the subject is the same and allows only the calculation of distance. When the subject is moving in a direction perpendicular to the transmitter however the pulse shape will have a phase shift due to the Doppler effect. The perceived frequency at the receiver will be

$$f = \left(\frac{c + v_t}{c + v_s} \right) f_0, \quad (2)$$

where v_t is the velocity of the receiver relative to the medium, v_s is the velocity of the source, and f_0 is the transmit frequency. The movement of the subject is therefore encoded in the modulation of frequency and can be determined. For speeds that are small compared to the speed of light, we can approximate the speed of the transmitter relative to the receiver as

$$\Delta f = f - f_0 \approx \left(\frac{\Delta v}{c} \right) f_0, \quad (3)$$

where $\Delta v = v_t - v_s$. [2] Solving this equation, we get the speed of the radar relative to the subject as

$$\Delta v \approx \Delta f \cdot \frac{c}{f_0}, \quad (4)$$

where the only unknown quantity is the frequency shift, Δf that is measured at the receiver. Thus, the radar sensor in its most basic form encompasses two separate measurements: the range and the velocity. This output format is also commonly referred to as the range-Doppler dimension. It must be noted that the radar sensor can only measure the component of motion radial to the sensor. The tangential speed component seen as sideways motion is ambiguous which makes the total ground speed of the detected object nondeterminable. The radar measurement gives the minimum ground speed, but no upper limit can be obtained as the second component of the speed vector cannot be measured. [3]

The frequencies at which a radar operates are divided into separate bands meant for different applications. The longer wavelength radar reaches further distances without attenuation as indicated in the free space radio frequency (RF) signal path loss function

$$PathLoss(r) = 20 \log \left(\frac{4\pi r}{\lambda} \right), \quad (5)$$

where r is the distance traveled by the wave, λ is the wavelength of the signal, and path loss is measured in decibels [4]. Therefore, longer wavelengths are used for longer-range radar. The shorter wavelengths are used for short-range radars and their benefit is that they can utilize more bandwidth as more band is available in the higher frequencies. The common operational radar frequency bands are described in table 1.

Table 1 Radar frequency bands [5].

Name	Frequency (GHz)	Wavelength (cm)
Millimeter	40–100	0.75–0.3
Ka	26.5–40	1.1–0.75
K	18–26.5	1.7–1.1
Ku	12.5–18	2.4–1.7
X	8–12.5	3.75–2.4
C	4–8	7.5–3.75
S	2–4	15–7.5
L	1–2	30–15
UHF	0.3–1	100–30

The wavelength dictates the length of the antennas and shorter wavelengths are deemed more useful in commercial use cases due to less required space and lesser material usage. The choice

of operation band also affects the Doppler frequency shift as formulated above. This is due to the choice of f_0 in formula (3). Large antennas for long distances are utilized for example in over-the-horizon radars whereas the higher frequency bands are used in targeting radars for aircraft. [5]

The formulas defined in the next chapter regarding FMCW modulation are equations for Multiple Input Multiple Output (MiMo) radar. The advantage of MiMo is that by keeping transmitted signals separate from each other we can later on in the receiver discern the different transmitters used to send them. For example, when 4 receive antennas each can determine 3 transmit antennas from their receive signal we can make a virtual antenna array of size 12. This allows the resolution to rise above the number of real antennas in a multiplicative fashion. The principle of MiMo is often mixed with the concept of beamforming. In beamforming, the goal is to use many antennas to transmit not many, but a single strengthened and directed beam [6] [7]. Beamforming can be applied also on the receiver side and can be used to change the radiation pattern of the antenna group. Beamforming techniques can be used at the same time as MiMo techniques [7]. This work focuses on MiMo radar and thus wants the antenna group to transmit separate MiMo signals instead of a single directed beam. While this work doesn't implement beamforming, it can be added to the system later on.

2.2 Frequency Modulated Continuous Wave

In automotive applications, the radar functionality is often implemented with a frequency-modulated continuous-wave modulation (FMCW). The longest distances viewable from the perspective of a sensor mounted in the bumper of a vehicle are in the range of 150 meters. For this shorter-range radar, the millimeter band is often utilized together with the FMCW modulation in the range of 76 to 81 GHz to achieve a maximum resolution as small as a fraction of a millimeter [1].§

The FMCW modulation differs from a standard pulsed radar in that it transmits a signal continuously as opposed to in pulses. Radar systems implementing FMCW also frequently employ more than 1 antenna for transmit and receive creating a multiple input multiple output (MiMo) transmit system. The addition of MiMo allows FMCW systems to detect the angle of arrival (AoA) of a reflected pulse. With additional processing and antenna placements, the AoA can also be split into separate azimuth and elevation angles to provide what is regarded in marketing as 4D-radar, encompassing in its output format the range, Doppler, azimuth, and elevation [8].

The signal used in FMCW changes its frequency linearly with time. The ramping of the frequency is displayed in figure 1 as a function of time. The resulting time domain signal is shown in figure 2. The signal format is referred to as a chirp because of the resemblance of the frequency response to the sound of birds chirping. The characteristics of the chirp are defined by the start frequency f_c , bandwidth B , slope S , and duration T_c illustrated in figure 1.

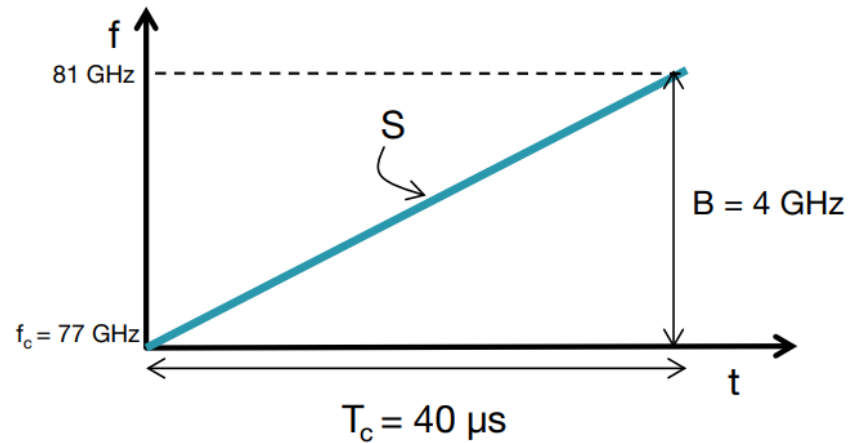


Figure 1 Frequency of the chirp signal as a function of time [1].

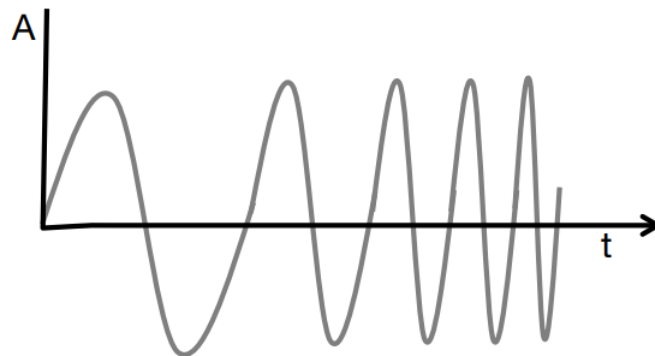


Figure 2 Amplitude of the chirp signal as a function of time [1].

To understand the function of the chirp in the system context we need to look at the constituent parts of the system and their functions. A simplified block diagram of the radar frequency (RF) subsystem is shown in figure 3.

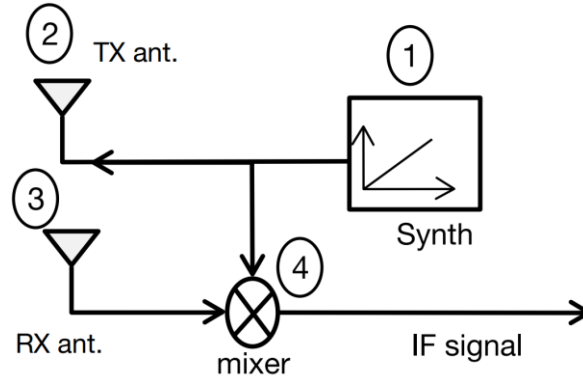


Figure 3 Simplified block diagram of the RF processing chain in an FMCW radar [1].

The radar operates by creating a chirp signal in the synthesizer. This chirp signal is transmitted to space via the transmit (TX) antenna. An object in the space to which the signal was sent reflects the signal back to the system. The receive (RX) antenna picks up this signal. The received signal is mixed together with the transmitted signal in order to obtain a lower intermediate frequency (IF) signal.

The mixer is an analog component that implements the time domain multiplication operation. The IF frequency created by the mixer can be represented by a sine wave with a frequency equal to the difference between the two inputs of the mixer. Mathematically this is due to trigonometric identity for sine wave multiplication

$$\sin(a) \cdot \sin(b) = \frac{\sin(a + b) + \sin(a - b)}{2}, \quad (6)$$

where we neglect the summation frequency and take only the difference frequency [9]. The neglecting of the summation frequency is implemented with low pass filtering. The filtering is trivial because the difference frequencies are orders of magnitude smaller in frequency than the summation frequencies. For two sine wave inputs of the form

$$x_1 = \sin(\omega_1 t + \phi_1) \quad (7)$$

$$x_2 = \sin(\omega_2 t + \phi_2), \quad (8)$$

where ω_1 and ω_2 represent the angular frequencies of the signals and ϕ_1 and ϕ_2 the phase of the sine waves, we get as the output of the mixer-lowpass sub-block a signal represented as

$$x_{out} = \sin[(\omega_1 - \omega_2)t + (\phi_1 - \phi_2)]. \quad (9)$$

When we couple equation (9) with knowledge of the input signals of the mixer we can visually represent the output of the mixer as shown in figure 4. The inputs of the mixer are again the TX chirp and the received delayed chirp that has a delay time equal to the time-of-flight between the object and the radar. In the illustration, we see that for a stationary object, the IF signal will

be a constant frequency sine wave. The frequency of the IF signal can be calculated from the slope, distance to the object, and speed of light as

$$f_0 = \frac{S2d}{c}. \quad (10)$$

The frequency difference amounting to f_0 is illustrated as S_τ in figure 4 to avoid confusion between the signals.

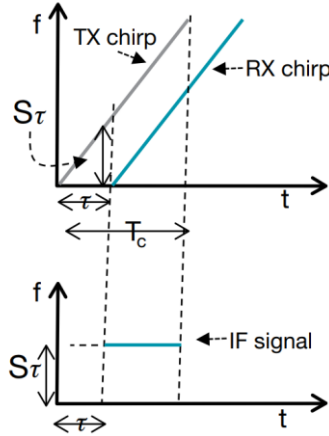


Figure 4 Mixer output (below) calculated as the difference of the mixer inputs (above) assuming a singular static object placed in front of the radar [1].

It must be noted that the IF signal is not valid after the duration, T_c of the chirp. This is because the mixer input from the synthesizer will go out of the desired chirp range after T_c and issue a reset. The IF signal is also valid only after τ seconds from the beginning of the chirp, which accounts for the time the signal took in time-of-flight. Figure 4 focused on the frequency of the signal. In addition to this, the resulting sine wave also has phase difference resulting from mixing. The phase can be derived from the start frequency and the time-of-flight as

$$\phi_0 = 2\pi f_c \tau, \quad (11)$$

and can be made to a useable approximation of the distance when the slope and distance are sufficiently small as

$$\phi_0 \approx \frac{4\pi d}{\lambda}. \quad (12)$$

For a radar with only one object in front of it, the total IF signal can then be represented as

$$x_{IF}(t) = A \sin(2\pi f_0 t + \phi_0), \quad (13)$$

where A is the amplitude of the signal. The calculation of distance can now be made from either the frequency in equation (10) or the phase in equation (12). The frequency option is often

used as it can be performed with a simple FFT. The distance can thus be calculated from the IF signal with all other variables being constant as

$$d = \frac{cf_0}{2S}. \quad (14)$$

This concludes the function of the RF chain before moving to the digital domain via an analog to digital conversion (ADC) for a simple radar path where only a single object exists in front of the radar. This configuration however is nearly a corner case of real-world conditions as multiple objects often reside in the radar's field of view (FoV). In figure 5 three different objects are placed in front of the radar at different depths. The objects result in 3 separate IF tones, the frequency of which correspond to the distance of the objects in question. The IF signal containing these 3 tones is then fed to the ADC.

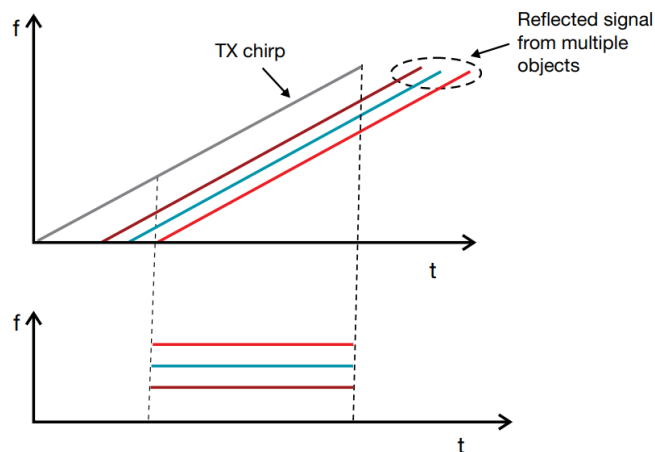


Figure 5 Three objects reflecting the TX chirp simultaneously result in 3 separate IF tones in the receiver [1].

The signal now enters the digital domain where a fast Fourier transform (FFT) is calculated. This FFT aims to gain access to the frequency information that the RF frontend decoded the distance data into. The resolution of an FFT is dependent on how long the FFT window is. The resolution of the FFT will in turn dictate the resolution for the measurement of range. The range resolution is defined as the smallest distance that two objects can be apart so that they can still be detected as two different objects. While the theoretical resolution limit is determined by radar parameters, the digitization must make enough samples of the signals for the resulting FFT bin sizes to be smaller or equal to the resolution if no information is to be lost. Fourier theory states that in order to increase the resolution (make more, but smaller bins), the duration of the chirp, T_c must be increased. According to Fourier, the bin width is the reciprocal of the sample time and therefore the frequency resolution of FFT can be defined as

$$\Delta f > \frac{1}{T_c}, \quad (15)$$

where T_c is the observation interval. In order to relate the Fourier parameters to the chirp parameters we note that the observation interval and chirp duration both constitute the same T_c . Given the slope we can then calculate the bandwidth of the chirp as

$$B = ST_c. \quad (16)$$

When we feed equations (10) and (16) into equation (15) we obtain the two different forms of the range resolution formula

$$d_{res} > \frac{c}{2ST_c} = \frac{c}{2B}. \quad (17)$$

We observe from these formulas that the only way to get better range resolution is to increase the bandwidth used by the radar. This formula gives rise to the different radar band use cases that were described in section 2.1. For a millimeter band radar working in a bandwidth of 4 GHz the maximum range resolution will be 3.75cm. [1]

Next, we derive the speed of the detected object from the IF data. In order to do this a second chirp needs to be sent out. The FFT is once again computed. The distance of the object may have been enough to switch from one FFT bin to another indicating a fast rate of speed but more commonly the peaks of the FFT will remain in the same spot. Here the reader is reminded that FFT produces as its output not a real number but a complex number instead with both amplitude and phase information. It turns out that even a smaller rate of speed of an object can be determined by looking at the phase difference between two peaks of consecutive FFT calculations. The two-chirp velocity measurement is illustrated in figure 6.

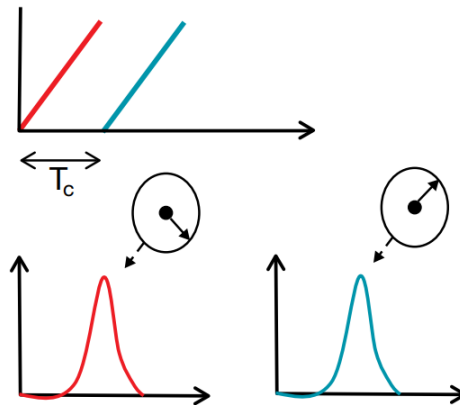


Figure 6 Consecutive chirps can be processed to the FFT domain, and the phase of the peaks compared to each other [1].

The phase difference between these two peaks is

$$\Delta\phi = \frac{4\pi v T_c}{\lambda}, \quad (18)$$

from which we can solve the velocity of the object, v using the chirp duration and wavelength of the f_0 frequency as

$$v = \frac{\lambda \Delta\phi}{4\pi T_c}. \quad (19)$$

As the velocity measurement is made using phase information which is periodical with a period of 2π , we must constrain the phase difference to $|\Delta\phi| < \pi$ in order to make the velocity measurement unambiguous. When this condition is applied to equation (19), we obtain the maximum relative speed that we can distinguish by sending 2 frames T_c apart from each other as

$$v_{max} > \frac{\lambda}{4T_c}. \quad (20)$$

Again, we have now calculated the results for a singular item in the view of the radar. In case there are multiple objects in front of the radar, moving at different speeds but at the same distance as shown in figure 7a, we need to resort to using more than two chirps. A total number of N chirps are collected to form a chirp frame. The range FFT is calculated for each chirp resulting in peaks in the exact same location. The phases of the chirps are rotating as per the velocity of the objects. We can now perform a second FFT on the results of the first FFT:s which we refer to as the Doppler FFT to indicate that this is done in order to resolve the speed. From complex input FFT theory, we remark that two input peaks with different phase rotation speeds will result in two peaks in the output. The input and output of the Doppler FFT are illustrated in figure 7b.

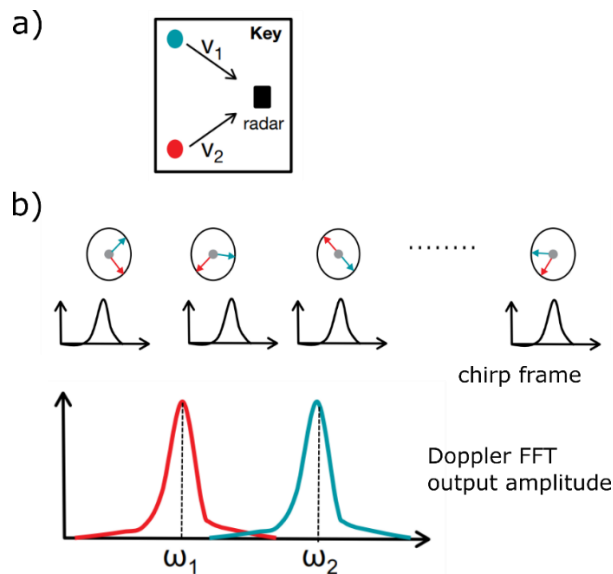


Figure 7a) The input scenario. b) The formation of N range FFT results into the input of the Doppler FFT calculation and the resulting amplitude output in Doppler space [1].

The velocities of the two moving bodies can now be determined from the location of the peaks of the Doppler FFT amplitude spectrum as

$$v_1 = \frac{\lambda \omega_1}{4\pi T_c} \quad (21)$$

$$v_2 = \frac{\lambda \omega_2}{4\pi T_c}, \quad (22)$$

where ω_1 and ω_2 correspond to the angular frequencies of the peaks. Again, the nature of the FFT calculation creates discrete bins, the size of which depends on the length of the input. The input size now is the number of chirps in a chirp frame, and this makes the bin size of the Doppler FFT $\Delta\omega = \frac{2\pi}{N}$. Formula (18) which originally applied only for 2 chirps also applies for the case of N frames when we extend the time period to the whole of the frame period $T_f = NT_c$. Doing this allows us to find the Doppler FFT bin size in the unit of speed. This is also referred to as the velocity resolution and is represented as

$$v_{res} = \frac{\lambda}{2T_f}. \quad (23)$$

This is the smallest change in speed that the Doppler FFT can discern. The speed resolution can be made better by increasing the frame time by sending more frames.

With the speed and velocity measurements now complete we move to the final measurement a radar sensor provides. The angle of arrival (AoA) is determined as the horizontal rotation angle between the centerline of the radar and the point of reflection. The angle of arrival in the horizontal plane is illustrated in figure 8.

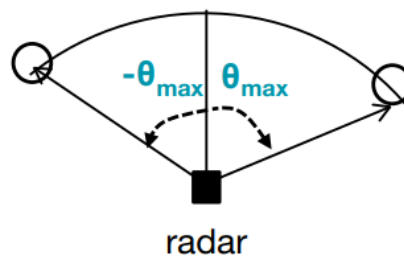


Figure 8 Horizontal angle of arrival. Sign of angle θ separates left and right [1].

In order to make the AoA calculation at least two antennas are needed. The angle of arrival can be obtained from the phase shift observed between two or more receiving antennas separated from each other by a distance of l . This is due to the length of the path a signal has to take. The length will increase by Δd by going to antenna 2 instead of antenna 1 as illustrated in figure 9.

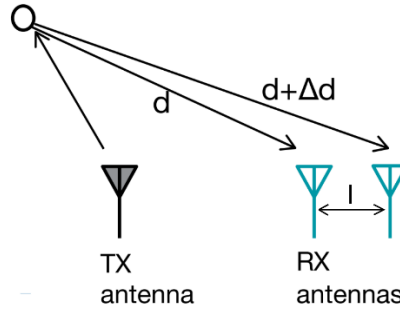


Figure 9 Geometry of the signal propagation with multiple receive antennas [1].

The phase shift generated by an extra distance of Δd is

$$\Delta\phi = \frac{2\pi\Delta d}{\lambda}. \quad (24)$$

We can assume the incoming wavefront is a plane wave and thus the straight arrow propagation also depicted in figure 9 holds. We can then use geometry to express the excess propagation length as $\Delta d = l \cdot \sin(\theta)$, where θ is the angle of arrival. Equation (24) can then be substituted with the alternate definition of Δd to obtain the angle of arrival by using only measured and previously known quantities as

$$\theta = \sin^{-1}\left(\frac{\lambda\Delta\phi}{2\pi l}\right), \quad (25)$$

which unlike the other radar outputs is inherently nonlinear. In some scenarios, we may want to approximate the nonlinearity using the small angle approximation $\sin(x) \approx x$. This approximation will have more error the bigger we make x . For $x < 1$ the approximation error remains under 20% [10]. Therefore, the equation used in practical devices becomes

$$\theta = \frac{\lambda\Delta\phi}{2\pi l}, \quad (26)$$

where we keep in mind that the detections at the edge of the field of view are distorted. Just like in previous measurements the phase shift must be limited to $|\Delta\phi| < \pi$ in order for the measurement to remain unambiguous. This restriction coupled with equation (26) gives us the expression for the maximum obtainable angle of arrival

$$\theta_{max} = \sin^{-1}\left(\frac{\lambda}{2l}\right), \quad (27)$$

which was also illustrated in figure 8. The largest obtainable field of view of $\pm 90^\circ$ is achieved with an antenna spacing of $l = \frac{\lambda}{2}$. [1]

The angular resolution of a radar system depends on the number of virtual receive antennas, N . A virtual antenna can be made by combining multiple transmit antennas. For a singular antenna, the formula of angular resolution becomes

$$\theta_{res} > \frac{\lambda}{Nd \cdot \cos \theta}, \quad (28)$$

where the presence of θ indicates that the angular resolution is worse at the edges of the beam and best at $\theta = 0^\circ$ also referred to as boresight direction. For antennas spaced at $l = \frac{\lambda}{2}$ and the beam at boresight, the angular resolution becomes dependent only on the number of virtual antennas as

$$\theta_{res} > \frac{2}{N}, \quad (29)$$

from which we can make out that in order to improve angular resolution the system needs more antennas. [11]

In this radar section, we have used a fundamentals-based approach to radar equations. For a more mathematical approach see [12]. With these equations, we can form the dense data representation. Next, we look at what can be done for this raw data in order to find and locate the real-world objects within.

2.3 Continuous False Alarm Rate

Continuous false alarm rate (CFAR) is an algorithm that is used to find peaks in a spectrum using a variable threshold. An illustration of the algorithm is presented in figure 10. The CFAR algorithm can be compared to the radar equivalent of feature detection in a computer vision context. In the radar application, the input data is the range FFT slice of the radar cube which is an array of complex numbers. The aim of CFAR is to separate objects reflecting radar waves from the noise floor. Together with angle of arrival (AoA) calculation, this algorithm turns the complex number radar cube into the form of an (x,y,z,v) point cloud, where the coordinates are measured from the radar sensor origin and speed is the radial component of the object in front of the radar.

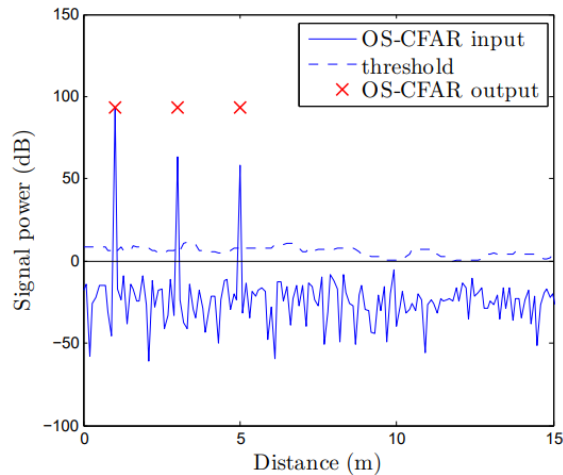


Figure 10 CFAR algorithm, in which the signal input, the algorithm-determined threshold, and the true false output mask are shown [13].

The CFAR algorithm determines its output by thresholding. The thresholding operation gives a true/false answer and the range bin index in which the positive occurred then gives the radial depth. An adaptive threshold is calculated in the range domain that ensures that the rate of false positive detections remains under a specified value. CFAR utilizes a window of the chosen size under which it operates. The threshold is calculated on a per-bin basis in a window around each range bin separately. Two common variations for threshold calculation exist. The cell average approach (CA-CFAR) takes the average of the window and uses that as a threshold, whereas the order statistic (OS-CFAR) approach sorts the values inside the window into ascending order after which one is chosen. The n :th number that is to be chosen from the sorted list is referred to as the order of the OS-CFAR algorithm. The output of both calculation methods is then multiplied by a constant T_{OS} which sets the false alarm rate. [13] The false alarm rate is a user-tweakable parameter that often fit on a use-case basis.

The CA-CFAR algorithm is highly dependent on the choice of window size, n because of the averaging. OS-CFAR does not have such a high dependency on window size. CA-CFAR is also known for masking nearby peaks because peaks in the same window will raise the threshold value and mask each other. OS-CFAR is, therefore, better suited for multiple object detection, but it also comes with higher computational costs. [13] There also exist other alternatives to fill the role of CFAR in a peak detection task such as DRD and RPDNet. These algorithms, although more complex, can offer more detections with also a higher accuracy. [14]

The purpose of the CFAR algorithm, or any other peak detection algorithm, is to decipher peaks emanating from real-world reflections from the radar clutter around them. The thresholding operation however discards large parts of the inherent semantical information from such reflections [15] [14]. With the help of machine learning, more information such as object classification

and confidence maps for the classified objects can also be attained from the raw RF data, also sometimes referred to as the radar tensor [14]. This can be done by not processing all data with CFAR and by leaving the raw data for a later stage. A 3D convolutional neural network based on an hourglass architecture with skip connections called RODNet is presented in [16]. It can achieve 86% average precision and 88% average recall for the CRUW dataset, in which, it separates pedestrians from cyclists and cars. In using a machine learning-based approach, special sensor-specific knowledge such as type of reflection and Doppler semantics can be extracted. This allows radar sensing to extract more classification information about the reflection than would be possible from a point cloud provided by Lidar [16]. This is crucial as the point cloud density made by CFAR processing is very sparse in comparison to Lidar sensing [14] [17]. Radar detections with CFAR do not scan the whole environment but instead, result only from places that reflect back radar waves. The average points per second with the CFAR algorithm is often under 1000, whereas other more advanced radar methods such as [18] can achieve 50 000+ points per second. As points do not emanate from all directions and the speed of their occurrence is slower than lidar or other feature detectors I refer to the CFAR data as being sparse. The sparse nature of the CFAR output is good for communication bandwidth, computational load, and memory footprint, but it discards the semantic information of the points.

NVIDIA CEO Jensen Huang claims in Computex 2023 that all information has an inherent structure that can be learned like a language [19]. The language of radar can be learned by implementing machine learning algorithms. We can then translate the language into other modalities such as classification and detection tasks. The language of radar is more complex than Lidar because of the inherent velocity associated with measurement and may therefore give a more robust description of the object. With AI having increased computer graphics performance a 1000-fold in just 5 years due to better AI hardware accelerators the idea of running AI for the massive amounts of raw data directly on the sensor is becoming a feasible option. [19]

2.4 Angle of arrival

Once potential objects are detected with the CFAR algorithm in the range-space the next step of radar processing is to calculate the point cloud. This is done via angle of arrival (AoA) calculations. In essence, this operation depicts moving the data from a spherical 3D space into an Euclidean 3D space. This is a general operation but here I will focus on the implementation specific to the hardware-accelerated AoA in the AWR1843AOP device used in this work. The implementation-specific AoA approach is depicted in figure 11.

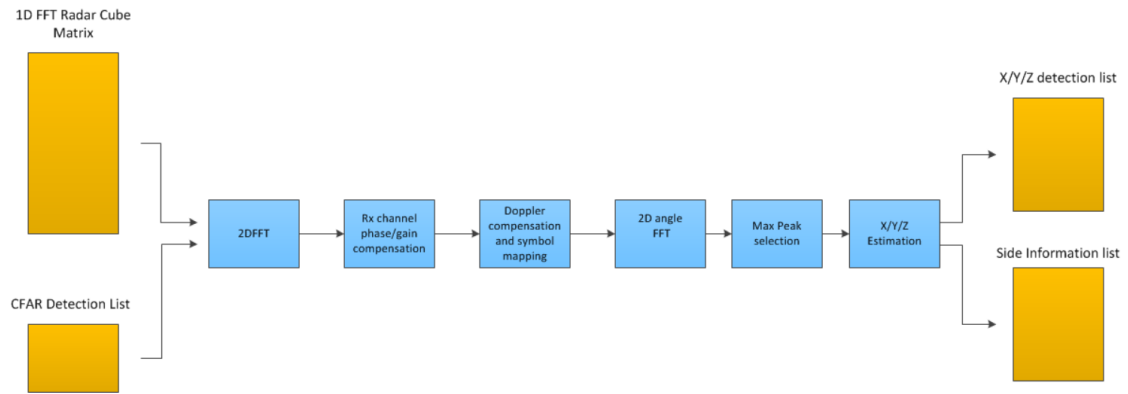


Figure 11 Angle of arrival calculation in the AWR1843AOP device [20].

The AoA calculation takes the range FFT and CFAR output as its inputs and calculates Doppler FFT only for the slices of range FFT that contain a CFAR detection. Receive channel phase and gain compensation are then performed. The compensation is done via pre-measured calibration values given to the device via a configuration file. The compensation aims to calibrate the phases and gains so that AoA output values have no rotation or translation compared to the 3D space they are fit into. Doppler compensation is also run which aims to adjust the gain of 2D-FFT symbols to increase output levels of weak targets.

Angle-FFT is then calculated in both azimuth and elevation direction. On this 2D plane, a new run of CFAR peak detection is performed. As an output, we have a 2D True-False map. On this map, we run peak grouping. Peak grouping uses a 5x5 kernel to go through the space and select only the center of the object. The result of peak grouping gives us object-specific dots on the azimuth-elevation plane. Each dot corresponds to the center of the incoming reflection. Each peak-grouped point is regarded as a detection.

For each detection, we then run x/y/z estimation which means calculating the cartesian coordinate of this detection. In the case of AWR1843AOP, we do this by using geometry. The coordinate geometry in regard to the virtual and physical antenna configuration is defined in figure 12.

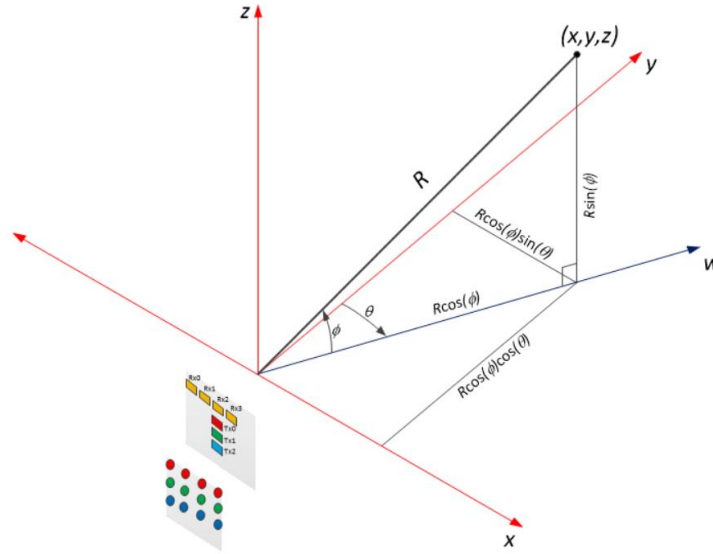


Figure 12 Coordinate geometry of AWR1843AOP in relation to the used physical and virtual antenna configuration [20]. The virtual array pattern is determined by the convolution of the physical transmit and receive antenna patterns [21].

From this geometry, we can decipher the formula for each coordinate as

$$x = R \cos(\phi) \sin(\theta) \quad (30)$$

$$z = R \sin(\phi) \quad (31)$$

$$y = \sqrt{R^2 - x^2 - z^2}, \quad (32)$$

where x, y , and z are the Euclidean coordinates, R is the range, θ the azimuth, and ϕ the elevation. Note that in this work we use the x - z -plane as the sensor plane and y -coordinate as the distance away from the plane. Inside the device, this calculation is done using indexes of the FFT bins. We have now arrived at the point cloud representation of radar data which is the final data format used in most radar applications.

Other methods for determining the angle of arrival of a radar beam also exist such as Bartlett beamforming, maximum-likelihood, MVDR, MUSIC, ESPRIT, min-norm, compressive sensing, and machine learning-based models of estimation [12], [22]. Radar data can also be preprocessed in order to remove unwanted components. A simple but very important preprocessing step called static clutter removal is used extensively in the following work. The static clutter removal algorithm calculates the mean value of samples for each range bin per antenna and subtracts it from the samples.

2.5 Sensor fusion

Sensor fusion means taking information from many sensor modalities and combining them together into a single representation. The motivation for this is that we can gain more detailed information by using many sensors. Using multiple sensors also brings with it redundancy and the ability to use more modality-dependent data representations to gather information about the environment. For example, cameras are great at sensing motion in their 2D image plane but lack depth information. Radar deals inherently in-depth and velocities. Mixing these two modalities can give richer knowledge of the world at hand. The sensors can also be used to cross-validate each other. The camera can be occluded by snow, fog, or rain in adverse weather conditions, which leads to camera data becoming degraded. Radar however is resilient to adverse weather conditions. In addition to these motivations, the radar-to-camera sensor fusion is of particular interest because of the cheap price of the sensor combination. Compared to lidar sensing this sensor combination can be an order of magnitude cheaper and still lead to comparable and sometimes even better end results [23] [17].

A key function in autonomous driving is planning. The more time a car has to observe and plan its actions the safer the safer its passage. The problem with cameras is that they cannot see objects that are far away. [15] Present-day radar sensors can however sense cars from a distance further than 350m and pedestrians from over 100m even if they are occluded from the direct line of sight [18]. Radar sensing thus gives the system more time to plan around dangerous situations. Now that the question of why sensor fusion is useful is answered we move to look at ways to implement radar-to-camera fusion. To illustrate the coming discussion, figure 13 is provided in advance of the discussion to aid in visualization.

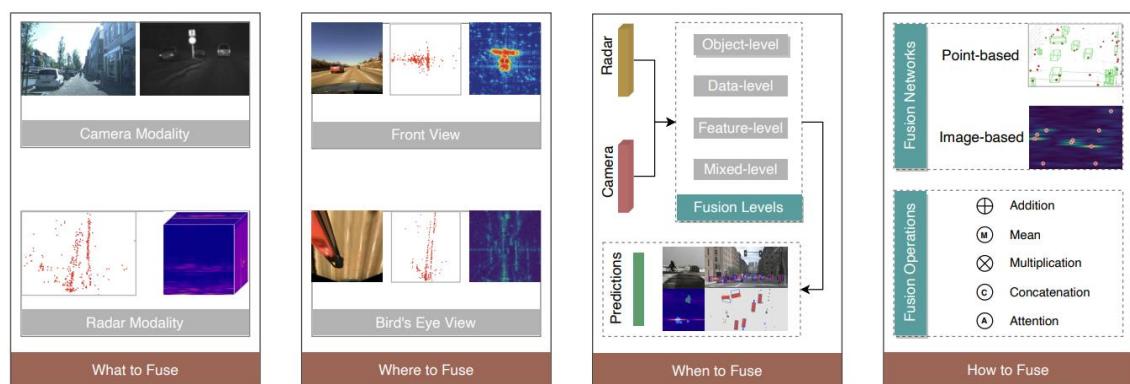


Figure 13 Core aspects of radar-camera fusion [15].

The first design question regards data. In what form do we gather the data from sensor modalities? On the camera side, standard RGB cameras are the cheap option. Their main design parameters include shutter speed, frames per second (fps), dynamic range, field of view, and noise. Infra-red (IR) cameras can also be used to gather camera data even in darkness. This may require an illumination source based on the IR wavelength used. In addition to these standard cameras, RGBD cameras also exist which give additional information about the depth (-D) of objects such as in [24]. The depth information is often gathered by employing multiple camera sensors in a single package. An event camera can also be utilized to simultaneously increase the fps to more than 1000 and to gain insight into the spatiotemporal changes of the scene. Event cameras such as the ones developed by [25] can give huge advantages for camera sensing because of their fast reaction time and frame rate. In this work, we utilize a standard RGB camera.

On the radar side, the AWR18430AOP data packets come in the form of point clouds defined in the previous chapters. Additionally, we can also output the whole radar cube with the dimensions being range, radial velocity, and azimuth (elevation is added in the case of 4D radar, but the form of data is still referred to as a radar cube). This radar cube is often sliced into range-azimuth and range-velocity slices. This is because a 2D slice is a common form that most neural networks intended for image data already accept. It is also common to take the velocity measurement from many consecutive frames by doing short-time Fourier transforms (STFT) on range slices of the radar cube [15]. This reveals what are called micro-Doppler signatures, which are temporal movement patterns. These dense data formats are mostly intended for machine learning approaches.

Next, we must decide which view system we are going to be using. There are two main viewpoints a fusion algorithm can use. The two approaches, Front view (FW) and Bird's eye view (BEV), were illustrated in figure 13. The front view is best approached as a 2D plane that is oriented perpendicular to the direction of movement. The BEV approach is also a 2D plane but looking at the vehicle from the top down. FW is often easy to integrate with present-day computer vision algorithms whereas BEV is good for mapping around the whole vehicle in SLAM applications and when using many sensors facing in different directions around the vehicle. [15]

Next on the path to fusion, you must choose when to fuse the data. There are many levels in a data hierarchy for fusion starting from low-level RF symbols for radar all the way to object-level detections. Figure 14 illustrates the different points at which the camera and radar modalities can be fused together. The level definitions arise from how much processing has been done to the data in its own branch before mixing it together with the second branch. As most modern-

day fusion approaches employ Machine learning for feature extraction and classification the stages are often referred to as layers.

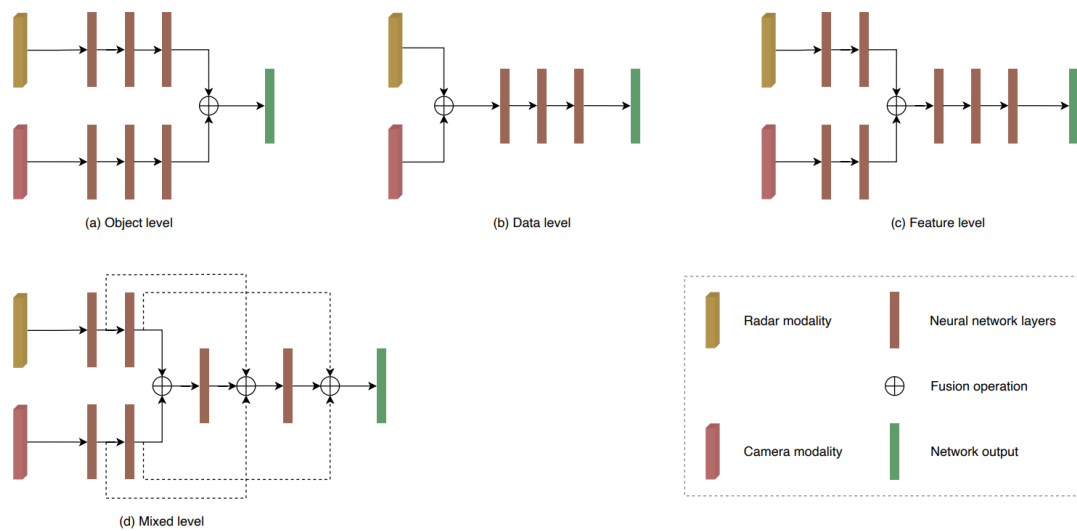


Figure 14 Levels of fusion indicated by the processing stage at which the fusion operation is applied [15].

The richest information is at the data level, but the computational load is also the greatest. It is common to extract features such as point clouds from the raw RF images at the edge node before communicating them to a fusion controller. This feature-level data is more compact but has lost some of its meaning. In object-level fusion, both sensor modalities work on their own until they have made sense of the features comprising entire objects. The objects are big real-world objects such as entire cars. Matching objects to one another is often easier due to there being only a handful of objects in view when compared to features of which there can be thousands in a picture. Mixed-level fusion combines data which are at different levels. This can mean for example combining an RGB image with a range-azimuth slice. Next, we look into what it takes to fuse data.

In order to make sense of the data, we must temporally align the data into a single dimension. The sensors may operate at different fps, have offsets in the sample time, or have a long latency from request to data readiness. The alignment can be done by calibration, buffering, latency prediction, or hardware synchronization. After temporal alignment, the data must be spatially aligned. This means searching for correspondences between the modalities. Spatial alignment can be done utilizing a target-based approach in which a known target such as a checkerboard pattern with a corner reflector for radar is used. Targetless approaches can also be used to calibrate the camera to radar point-to-point matching. Machine learning is often employed in a

targetless approach to re-calibrate the system on the go without having to bring out the specialized targets. From the point correspondences found with these approaches a transformation matrix can be established with the help of computer vision algorithms such as linear point-to-point, Perspective-n-Point, or Ransac.

With the calibrations now done, the data is mathematically put together into a single frame. The next stage in the network is often a machine learning stage so the frame must be in a form compatible with neural networks. The most basic fusion operation is concatenation. Concatenation entails making the N and M dimensional data into N+M dimensions by putting the data of radar after the data of the camera. This makes for a large input dimension and is therefore computationally hard. Addition, multiplication, average, and mean operations are also commonly used to put the data into the same frame. By performing these elementwise operations, the data from both modalities remains in the final frame but it is weighted differently based on the operation. In element-wise operations, the dimensions of the data are fixed which means the dimensions don't grow. This means however that the modalities must be brought to the same dimensions by interpolation before the element-wise operations can take place. The last operation of bringing data into the same frame of reference is transformation. This can mean taking 3D radar points in world coordinates and bringing them into a 2D image plane. It can also mean transforming image features into 3D world coordinates. As an end result, all the operations lead to a representation where data from both modalities is now in the same reference frame. [15]

2.6 Kalman filtering

Kalman filter is an algorithm that smooths out noise from sensors. It is a form of Bayesian filtering under the subcategory of g-h filters. It refines data by incorporating information from two sources: the input measurement and a prediction of the system's new state based on the previous state of the system. The Kalman filter then chooses its output value to be in between these two guesses. The Kalman filter chooses which source to trust based on the perceived error or the two sources. The data source with less error is given more trust. The result of filtering is that we get an estimate of the system state with less measurement noise. [26]

The Kalman filtering consists of two distinct operations: The prediction of the next state and the update of state information based on new measurements. These steps also consist of smaller sub-operations. The sub-operations of Kalman filtering are represented in the pseudocode presented in figure 15. The predict-update cycle is repeated for each incoming data point in a recursive fashion.

Initialization

1. Initialize the state of the filter
2. Initialize our belief in the state

Predict

1. Use system behavior to predict state at the next time step
2. Adjust belief to account for the uncertainty in prediction

Update

1. Get a measurement and associated belief about its accuracy
2. Compute residual between estimated state and measurement
3. Compute scaling factor based on whether the measurement or prediction is more accurate
4. set state between the prediction and measurement based on scaling factor
5. update belief in the state based on how certain we are in the measurement

Figure 15 Pseudo code for Kalman filter implementation [26].

The system state is represented as $\mathbf{x} = [x_1 \quad x_2]^T$, where the state variables can be for example place, speed, acceleration, temperature, color or any other unit of measurement. In addition to defining \mathbf{x} as the state we must represent whether this is the pre-measurement state or the state that has been mixed with the measurement. The state before incorporation of measurement data is referred to as the a priori state and is represented with line notation $\bar{\mathbf{x}} = [x_1 \quad x_2]^T$, whereas the state after measurement incorporation is referred to as the a posteriori state already presented as \mathbf{x} . The state variables can be split into observed variables that are measured directly or hidden variables if no measurement is made and the state is purely based on prediction. Hidden variables are determined by models such as Newtons laws for motion. Hidden variables are used to better represent system dynamics and to allow the Kalman filter to make predictions that are of the same order (number of derivatives) as the system under measurement. [26]

Kalman filtering uses Gaussian distributions as its internal data representation, which allows the use of only two variables: mean and variance, to describe a whole distribution. This is in stark contrast to discrete Bayes filters where each discrete value is assigned its own probability. In the state vector defined above, the variables represent the means of the distributions. The state variables also have corresponding variances expressed in the state covariance matrix \mathbf{P} .

For the Kalman filter to choose trust between a prediction and measurement, a model of the system must be formulated. This model is used to make the prediction. The Kalman filter uses a

state-based discrete-time model to make estimates of the new state. However, instead of defining a system dynamics matrix, \mathbf{A} often used in control theory, which works in the transfer function $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, the Kalman filter instead uses a state transition function \mathbf{F} , which defines how a new state forms from the old state after time Δt has passed. The evolving of the system state with time is expressed as

$$\bar{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad (33)$$

where \mathbf{B} is the control function and \mathbf{u} the control input. Small letters indicate vectors and capital letters indicate matrices. Now that the a priori state mean has been found, the algorithm calculates covariance for this state as

$$\bar{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q}, \quad (34)$$

where \mathbf{Q} is the covariance matrix of the process. With the mean and covariance of the state now predicted, the prediction step is now done. It is noted that the prediction step has the effect of making the covariances larger, which means that after a series of predictions without measurements in between we will be less certain of the actual system state. Incorporation of the measurement in the upcoming update step has the effect of making the estimate accurate again and therefore making the state variance \mathbf{P} smaller.

The update step begins after a new measurement, \mathbf{z} has been made with sensors. From this measurement, we calculate the residual as

$$\mathbf{y} = \mathbf{z} - \mathbf{H}\bar{\mathbf{x}}, \quad (35)$$

where \mathbf{H} is the measurement function. The residual tells how much the state estimate differs from the measurement. Next, the algorithm decides whether we should give more trust to the estimate or the measurement. This is done by calculating Kalman gain as

$$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^T(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^T + \mathbf{R})^{-1}, \quad (36)$$

where \mathbf{R} is the measurement covariance associated with mean found in \mathbf{z} . The Kalman gain is always in the range of 0 to 1 with 1 signifying all trust going to the measurement and 0 that the trust should be placed on the prediction instead. From the equations we observe that the Kalman gain is calculated by comparing measurement variance against prediction variance. With the amount of trust now decided we can produce the filtered a posteriori state as

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}\mathbf{y}, \quad (37)$$

after which we have to update also the a posteriori covariance for the state as

$$\mathbf{P} = (\mathbf{I} - \mathbf{KH})\bar{\mathbf{P}}, \quad (38)$$

where \mathbf{I} is the identity matrix. This concludes the update step. The results of filtering are found in the mean state vector \mathbf{x} and the state covariance matrix \mathbf{P} tells us the confidence ellipse. With the math now settled we move to discuss filter implementation.

In order to get the correct behavior from the filter the designer needs to know a model that can be used for prediction of the next state. This is represented by the \mathbf{F} matrix. For this process model the designer needs to design a covariance matrix \mathbf{Q} which represents information loss in the process due to time passing without measurement. The effect of \mathbf{Q} is evident when the predict function is used without the update function following it, which happens in cases where no measurement can be made of the system under occlusion or measurement faults. The \mathbf{F} matrix is often chosen as a motion model defined by Newton's equations for kinematics. Sometimes this model is simplified by the Euler method for Δt , after which only the first order of the terms of Δt exist in the model [26]. In the motion model this cuts off effects such as acceleration $x = \frac{1}{2}a \cdot \Delta t^2$. The \mathbf{Q} matrix is often chosen based on noise models such as the continuous white noise or piecewise white noise. These noise models take into account the Δt between measurements to determine the expected variance of the state. The \mathbf{Q} matrix can be approximated by including only the lowest Δt orders in the matrix, as the higher powers of Δt become increasingly smaller.

Next, the designer must implement the measurement matrix \mathbf{H} and associated measurement covariance \mathbf{R} . The \mathbf{H} matrix transforms data from state space into the measurement space and is therefore the primary place where sensor fusion is implemented. Oftentimes the way to represent data transformation between the spaces contains nonlinearities. This happens for example if a radar produces its output as a range-azimuth point. For example the equation to transform from x, y state space to a radar azimuth measurement is $\alpha = \tan^{-1} \frac{y_{target} - y_{radar}}{x_{target} - x_{radar}}$, which is a nonlinear operation. When we use sensor fusion to associate data from multiple modalities, we might have sensors with different accuracies. We can represent the knowledge of sensor accuracy in the measurement covariance matrix \mathbf{R} . In it, the covariance between sensors is often chosen as 0. The choice of values for \mathbf{R} must be relative in value to the process noise matrix \mathbf{Q} . Too big values for \mathbf{R} will cause the filter to trust only the predictions and too low values cause it to disregard the prediction in favor of noisy measurement. The fitting of the covariances is often done by hand by making simulated scenarios of known states and behavior with measurement noise or by fitting to a real-world dataset of measurement data. [26]

In the Kalman filter, we decide what we model based on real-world system dynamics. The best approach is to match the order of the dynamics. Some dynamics cannot be taken into account, and they are referred to as noise. Tire slip, nonlinear behavior, wind gusts, and cars suddenly breaking are all considered noise in the Kalman filter context, where every influence on the process not defined in the process model is considered noise. The design of the filter includes a step of defining what we expect the system to do and what we leave out and consider as noise. [26]

Because the internal data representation of Kalman is a Gaussian, the filter works optimally only for Gaussian-distributed input variables and unimodal distributions. The Kalman filter is a linear algorithm which means that nonlinearities in processes can cause the algorithm not to converge on the real system state. The use of Gaussians means that the memory and computation requirements of Kalman filtering are lesser as compared to other discrete Bayes filters. [26]

The bad response of the Kalman filter to nonlinear functions can be dealt with by using a variation of the Kalman filter referred to as the unscented Kalman filter (UKF). This version of the Kalman filter uses N number of samples instead of a single computation to take into account the effects of nonlinearities. Instead of a Gaussian the input to this algorithm is rather a number of points sampled from the input Gaussian. These points are chosen with specific algorithms such as Merwe scaled sigma points which aim to represent the whole input space with samples. The points have individual weights associated with them which are used in assembling the final output from the separate samples. The output of a nonlinear function is rarely a Gaussian. In the UKF approach, the non-Gaussian output distribution is calculated as a mean and a variance and then approximated as a Gaussian. In the next cycle the second UKF iteration samples its delta points from this Gaussian state the loop continues. In the UKF approach the measurement function, h , and the state transition function, f are turned into functions that can be given nonlinear implementations. The functions can no longer be interpreted as matrices because matrices only work with linear algebra. The UKF is considered by [26] to be superior to the Extended Kalman Filter (EKF) often used in modern systems. This is because the UKF deals with a Monte Carlo-based sampling approach to solving nonlinear equations whereas the EKF bases itself on linearization of the nonlinear function near its state, which is a laborious and often only a local solution to the problem.

A radar-specific problem associated with Kalman filters is that a Kalman filter is designed to track the state of a single object. If we want to track multiple objects, we need to initialize multiple Kalman filters. One for each object. This is because different objects have differing states. In radar applications multi-path propagation often exists which can make a single object seem like multiple objects at different depths. If the multi-path reflections are allowed to be fed into the Kalman filter as measurements, the filter may never converge on the real state of the object because of the false measurements.

2.7 Simultaneous Localization And Mapping

Simultaneous localization and mapping (SLAM) is a possible end-application for the system we are going to develop in the ensuing chapters. In this chapter, we first discuss what is the function of SLAM in the vehicle platform context. We then go on to study its challenges and approaches to solve them. Afterwards, some examples of SLAM systems are presented.

2.7.1 Function of SLAM in vehicle context

There are two things we want a vehicle to do in its environment. The first consists of the vehicle avoiding collisions by focusing on other vehicles, pedestrians, and their inherent motions. This is often referred to as an advanced driver-assistance system (ADAS) and includes systems such as collision avoidance, adaptive cruise control, lane change assist, and automatic emergency braking [15]. The second function we want autonomous vehicles to achieve is to be able to navigate the environment around them autonomously by figuring out the environment and how the vehicle is positioned relative to the mapped environment (SLAM) [27]. These two applications have distinct features that differentiate them from each other.

In an ADAS context, the system is interested in the dynamic objects in its field of view, mainly cars and pedestrians in order to avoid collisions with them. The static background holds interest only in cases of low-hanging bridges and road walls that the vehicle must avoid colliding with. In contrast to this, systems incorporating SLAM are interested in mapping, which needs key points called landmarks, which are points of interest in the environment that can be regarded as static. These static landmarks can be used to guide the vehicle in its quest to localize itself and find out its orientation. Where a radar signature from a car is desired in the ADAS context to avoid the dynamic object it is unwanted in the viewpoint of SLAM algorithms. This is because the car blocks the view of the sensor disabling its ability to locate the static landmarks it relies on to fulfill its localization task. The two subsystems of the autonomous vehicle algorithms can thus be divided

into ADAS, the function of which is to focus on dynamic objects, and SLAM which focuses on static objects. The meaning of static is used to refer to world coordinates instead of car coordinates, as a car traveling the same speed as our radar would appear to us as a stationary object instead of one in motion.

Having distinguished the two different functions of an autonomous vehicle we note that these categories are often strongly linked with each other. For example, we do not want an imaging radar used for mapping to exclude information about possible debris on the road surface just because its function is not ADAS-based. The two functions are thus often best implemented in parallel. When SLAM is implemented, the created maps also allow for path planning and obstacle avoidance to use this information to their advantage. The map allows the vehicle to understand the environment around it and to make informed decisions on how to act. [28]

Radar is impervious for the safety of the vehicle. In ADAS systems and in SLAM the other modalities of sensing e.g., the camera and Lidar may be occluded and even fail totally because of environmental hardship. Radar with its lower carrier frequency operation does not suffer from this possible blinding of sensors. It is noted in [29] that for SLAM in bad weather conditions, only the radar-based SLAM approaches kept track of the vehicle motion. Both the vision and Lidar-based systems failed to either keep track of the vehicle position during the whole dataset or failed to initialize completely due to the camera or lidar sensor being occluded with snow, fog, or water. [29] It is also noteworthy that the proper functioning of the camera and Lidar systems are strongly correlated. In an environment where a camera fails, a Lidar is also likely to fail [15]. These conditions include fog, snow, and raindrops on the sensors. These adverse conditions leave radar as the only robust sensor.

2.7.2 How SLAM systems work

SLAM systems take in sensory input and history values of known places, sometimes even predetermined maps. With the sensor data, the first task of SLAM is to localize the sensors into the world coordinate system. This can be thought of as finding the translation and rotation of the system with regard to an agreed-upon origin in a world coordinate system. Often the world coordinates are chosen to be cartesian as the curvature of the earth seldom comes into question in systems utilizing SLAM. The second task is the creation of the map itself. [27] The map can take on many different forms. The most common forms of maps are presented in figure 16.


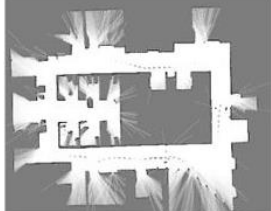
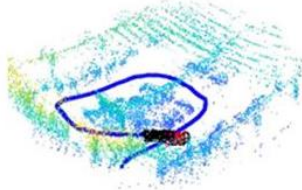
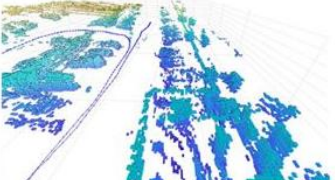
	Point Cloud Map (matching lidar outputs)	Grid Map (for obstacle avoidance)
2D Map		
3D Map		

Figure 16 Common map representations in SLAM systems [27].

A point cloud approach is a sparse map consisting of points of interest often referred to as landmarks. Landmarks in the visual domain are often found with feature detectors such as ORB, SURF, or SIFT -features. In the radar domain, the landmarks are a lot more sparse and are often found by peak detection algorithms from the reflection data with the CFAR algorithm. Lidar systems also inherently produce point clouds, but their point cloud information consists of more points with better accuracy. In contrast to the sparse nature of points, we can also approach the map in a grid-like fashion often referred to as pixels or voxels in 3D space. Every voxel in the map has a value so the information is dense. The representation is akin to the way the rodent brain is found to carry information about the world around it in grids [30]. This brain-inspired approach can be efficient if the grid size is appropriate so as not to waste space. We can also transition from a point cloud to a grid by sampling the number of points that appear inside a voxel [27].

With the map, the position of the vehicle can be estimated by comparing the present map to previous maps while keeping track of the path the vehicle itself has traveled (dead reckoning). In order to receive more localization information, the vehicle must move and discover more of the environment so that it can compare its history values with it. One way of localizing the vehicle from a known map is called the particle filter approach. In it, random samples are put into an already complete map. The observation from the sensors is then compared against the map. If what the sensor perceives is intelligible for the random spot, then that spot, also referred to as a particle survives. More iterations are performed as the vehicle moves. Every time the previously possible particles are filtered for possible locations and new randomized guesses are made around the already estimated place and orientation of the device. When the device has

traveled far enough, there exists only one particle cluster. This is the estimated device location. [31] The particle filter approach of randomly guessing the localization and checking whether this could be a valid guess is utilized in a popular algorithm called FastSLAM [32]. The filter attribute of the name refers to the fact that the algorithm is run on the go, without all the data being present [31].

A problem inherent in both the localization and mapping is error accumulation. Errors in measurements cause the maps not to look exactly like the ones in memory. This can lead to algorithms wrongly interpreting to be localized somewhere else than the real location. The bigger problem however is that the localization errors accumulate. Wrong localization causes the map to be written from a wrongly estimated position which causes the map itself to also be wrong. Thus, both the localization and mapping fail if error correction can't be applied. [31]

The most popular form of error correction is called pose graph optimization in which the device periodically looks into the historical data to see if the vehicle is now somewhere where it has been before. It looks to see if the vehicle has made a loop and the calculation of this occurrence is referred to as looking for loop closure. Pose graph optimization is a smoothing operation which means that at the time of running the algorithm, we have access to all the places the vehicle has been to. The vehicle translation and rotation state, otherwise known as pose, is interpreted as a graph with recorded localizations as nodes and traveled distances as edges. The pose graph optimization intends to make the beginning and the end nodes the same. To do this the edge distances and angles are given flexibility. The result of the calculation is that the accumulated historical errors can be smoothed out and the states that the vehicle was previously in are now better known. For a moment the vehicle is also in a well-localized state. There exists however a pitfall, that false loop closures will permanently bend the map into an erroneous form. The threshold for loop closures must be kept high in order to avoid false closures, as errors imprinted onto maps often can't be fixed and will cause trouble in the future. [31]

2.7.3 Challenges of SLAM

In addition to error accumulation, sensor failure and erroneous loop closure leading to map corruption SLAM also suffers from the same basic challenges discussed in the sensor fusion chapter. Namely the time and place synchronization of sensors in addition to sensor calibration. [27] As SLAM is run on a vehicle-wide level it will have many sources of data coming in. How this data is incorporated and what data is trusted more often has to be decided with knowledge of the external environment state. When camera and lidar fail in adverse weather conditions, trust must

be taken off them. However, when conditions are good, it would be foolish to trust radar alone, as it is a low-resolution form of sensing. Trust based on knowledge of the system state is of a higher order than what can be achieved with Kalman filter-based trust discussed in previous chapters and therefore a trust system must be implemented.

Computational cost is also a factor in SLAM. A central controller that receives raw data from all sensors will also have hard real-time constraints put on it, as the system is in control of the lives of its passengers. Together with the fast sensor communication channels it has to upkeep, a central controller will have to be extremely powerful. A distributed approach can help to relieve the computation and communication requirements. This means moving to higher levels of fusion like the feature- or object-level approaches discussed in previous chapters.

2.7.4 Examples of SLAM systems

SLAM algorithms are often divided into front- and back-end methods. Frontend deals with sensor data interpretation and making motion- and obstacle estimations for the perceived objects. This is also referred to as sensor-dependent processing whereas the backend deals independently from sensors. Backend algorithms often focus on map upkeep. The front- and backend processes are separable into their own threads of execution which again eases computational load by division into smaller parts using parallelism. [27]

As new SLAM algorithms are constantly made, I choose to direct attention to approaches with the most use in 2023. These approaches are listed in table 2 and are adapted from [33] by summarization. Online SLAM refers to a system that focuses calculation to its current state. Offline or full SLAM systems calculate the entire path the vehicle has traveled inside the map. This is often more computationally expensive. All of the approaches make compromises in some category. It is up to the system designer to decide which part of the system can be compromised.

Table 2 Most common SLAM approaches in 2023. Adapted from [33].

Name	Pro	Con
EKF SLAM	Mature	Memory
IF SLAM	Stable	Linearization errors
CEKF SLAM	Cost-effective	Map linkages
Fast SLAM	Robust to noise	Accuracy
Graph SLAM	Accurate	No online SLAM
iSAM2	Fast	Complexity as graph becomes dense
SLAM++	Fastest	Complexity as observations increase

3. PLATFORM

The AWR1843AOPEVM mmWave sensor from Texas Instruments (TI) was chosen for the framework of this master's thesis and for the radar demonstration platform used in Kovilta Oy based on multiple beneficial factors. The most time-saving and productivity-improving factor was the good support, documentation, and out-of-the-box demo software coupled with a software development kit, debugger, and large use of application programming interfaces in the codebase of the radar system on a chip (SOC). Inside the mmWave platform, multiple devices were also available to choose from. The mmWave category is split in half into industrial and automotive-focused products of which the automotive side was chosen. Then the selection came down to what subsystems were needed inside the radar SOC. As this work focuses on finding out the capabilities of the platform and the end product aims to be as fast and reliable as possible the device family with both an internal digital signal processor (DSP) and a hardware accelerator (HWA) was chosen. After these choices, the final choice was to decide between an antenna configuration drawn onto the PCB or an antenna on the package (AOP). As the PCB antenna would take up large space on the PCB and lead to additional calibration needs because of PCB parasitics it was chosen that the AOP configuration located atop the silicon in a ready-made array would be the best option. The inbuilt antennas also entail that no antenna designers need to be hired if Kovilta Oy were to append this device into their products.

3.1 Architecture of radar SoC

The functional block diagram of the AWR1843AOPEVM device is shown in figure 17. In it, we can discern 2 cores and a radar frontend. The radar front end is autonomous from the other cores and is addressed via an interface referred to as mmWaveLib. The front end is self-monitoring and -calibrating and its output, the ADC buffer, is connected to the DSP. The DSP unit also houses the hardware accelerator. The DSP-HWA block is the computation platform and the Cortex R4F processor acts as a controller of this computation. The Cortex microprocessor is also in control of the communications with the outside world. [34]

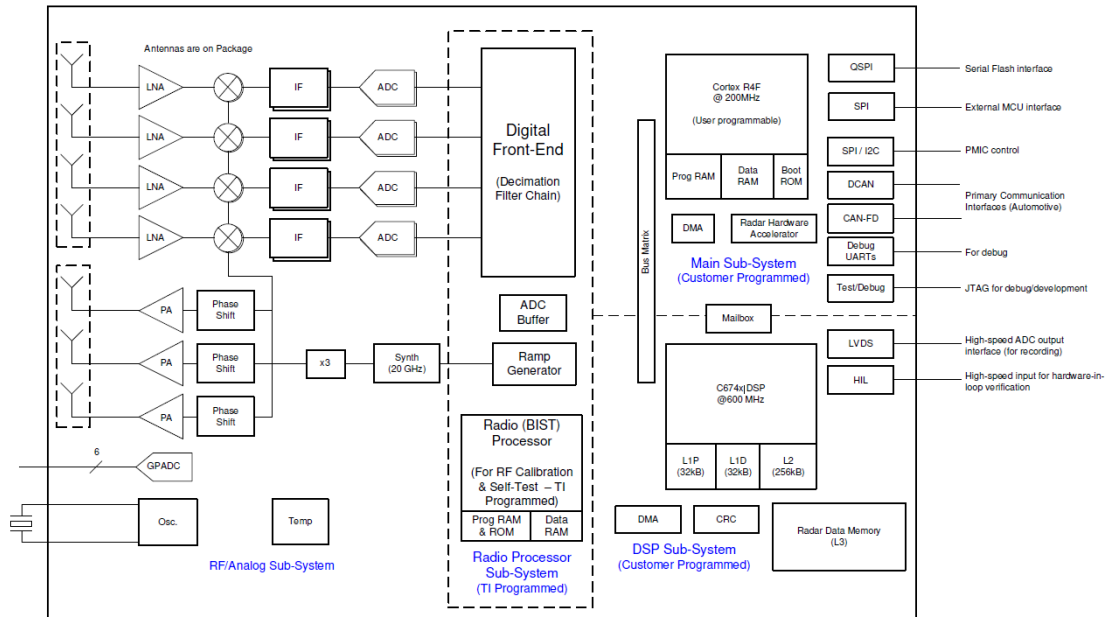


Figure 17 Functional block diagram of the AWR1843AOP system on a chip [35].

The external RF calibration mentioned in the angle of arrival chapter is named `compRangeBiasAndRxChanPhase`. It can be performed by placing a corner reflector at boresight at a known distance measured by tape measure in an otherwise anechoic space. As a result, we obtain gain and phase offsets for every single virtual antenna, which are used to correct deviations in angles of arrival. These values are stored in the DSP subsystem and are calibrated out after the radar ADC results are done. The phase shift circuits and power amplifiers (PA) in the front end have nothing to do with the calibration. [36]

3.2 Code resources

The system in this work is a mix of two different computation platforms: the embedded system integrating the radar SoC and the PC which interprets the data from the sensor and controls its operation via serial command line interface. As the codebases are separate, in different languages, and utilize different integrated development environments (IDE) we separate the inspection of these resources into their own chapters.

3.2.1 Embedded system code

Most of the embedded functionality is provided access to via the configuration file given to the SoC via the command line interface (CLI) serial port. In this work, the behavior of the SoC is configured via available switches and parameters rather than hardcoded into the SoC source

code. While it is important to understand the embedded side processing, this chapter focuses only on the top-level functionality of the SoC and its development environment.

The source code for the platform is given in MMWave SDK provided by Texas Instruments. It contains part-specific drivers, implementations of different algorithms on different hardware configurations, build tools, documentation, and demos. Most importantly the SDK is open source. Texas Instruments allows redistribution of both source codes and binaries with or without modification as long as the header disclaimers put in place by TI are not deleted and the name of TI is not used in the promotion of derived products without prior written permission. [34]

The bulk of the platform code review and writing is done in a TI-provided IDE called Code Composer Studio (CCS). This is due to software support for hardware-level debugging using the UART connection. CCS also gives apt developer help tools such as “Go to declaration”, “Go to reference” and “Explore macro expansion”. The TI Resource Explorer also provides the user with the option to make a modified copy of the outOfBoxDemo that works in a separate project from the original demo.

The outOfBoxDemo that our codebase is founded upon is split into two different languages for the two different domains of execution. While the embedded system containing the SoC runs on C, the TI-provided Demo Visualizer program that interprets the data packets streamed from the sensor is handled by JavaScript.

The C project constitutes approximately 50 000 lines of code (LoC), C libraries 340 000 LoC, while the JavaScript covers 85 000 LoC with all comment-like lines taken out. Texas Instruments provides documentation in the form of Doxygen websites for the C codes. The JavaScript is however documented only using comments. In C-code TI chooses to define some system variables in the compilation string given to gcc rather than in the source code itself. Therefore, debugging needs to take into account variables defined at compile time.

The embedded programs run on the 2 separate cores and have their own source code. Both the cores run TI RTOS, a real-time operating system capable of multithreading. The ARM Cortex R4F processor is referred to as the master subsystem (MSS) and the C674x is referred to as the DSP subsystem (DSS). Both systems run multithreaded and use inter-process communication to signal and synchronize messages between the cores. The threads and their processes are shown in table 3.

Table 3 Concurrently running threads of execution on the main processors.

Processor	Task name
MSS	MmwDemo_initTask
MSS	CLI_task
MSS	MmwDemo_mmWaveCtrlTask
MSS	MmwDemo_mssDPMTask
DSS	MmwDemo_initTask
DSS	MmwDemo_DPC_ObjectDetection_dpmTask

Let's first take a look at the MSS subsystem. The MmwDemo_initTask is the entry point of the program. It is run only once. Its function is to initialize component drivers, RF frontend communication channel, datapath communication, UART communication line, and ADC. After successful initialization, the thread will set up the other remaining threads in the MSS subsystem.

The first task to be created in the initialization is MmwDemo_mmWaveCtrlTask. This task is running an API for RF frontend communication. The task runs in a continuous loop with its scheduling based on priority. The priority of this thread is set to 5.

The second task is MmwDemo_mssDPMTask. The data path manager (DPM) is an API in charge of handling data transactions and inter-process communication (IPC) between cores. The system is run in *Remote Domains* -mode which sets the MSS subsystem to be in control of the transactions. The DSS subsystem in turn runs the DPC that this DPM task gets its data from. The task runs in a continuous loop with a thread priority of 4.

The final task running on the MSS subsystem is the CLI_task. The command line interface (CLI) provides a shell-like interface towards the UART line. This allows the MSS to communicate to an external computer via a USB. Using this CLI the user can configure the radar front-end parameters, communicate them via USB, and have the CLI take care of parsing the message and calling the appropriate API functions. The chirp parameters discussed in the FMCW chapter are set up via the CLI interface. The CLI task runs in a continuous loop with a thread priority of 3.

Next, we turn our attention to the DSS subsystem. Just like the MSS, the entry point of the DSS also lies in the initialization task MmwDemo_initTask. The task is run only once and is unique to this processor even though it shares the same name with its MSS counterpart. In the function, the system initializes component drivers including data path-related drivers for the hardware

accelerator (HWA) and the enhanced direct memory access controller (EDMA). The DPM module in control of running the aforementioned DPC is also initialized.

The last but also largest task is the `MmwDemo_DPC_ObjectDetection_dpmTask` run on the DSS. This task is in control of all computation. The object detection runs on this thread which implements a DPC that the master subsystem can share results with using the DPM API. The thread utilizes hardware accelerator resources and works on the DSP processor running a higher clock rate than the master subsystem. This thread handles the real-time calculation of range, Doppler, and angle of arrival from raw ADC results. The processing is done in a continuous loop with priority of 5.

3.2.2 PC side code

As the author is building a system meant for use in the company framework the JavaScript code given by TI in the `mmwaveDemo` was deemed too inefficient to interpret and to integrate into the environment. This was because of the sheer size of the project and its cleanliness. More work would have to be put into understanding the JavaScript than would be to produce our own implementation. Discussions with the code division at Kovilta oy led us to choose Python as the PC-side packet interpreter and system platform. Using Python enables us to use libraries vital for SLAM including OpenCV. PyCharm community editor was chosen for the IDE as it has seamless integration with virtual environments which we manage with Anaconda.

Next, we take a look at the code libraries utilized in this work and the resources they provided us with. NumPy is our first import and gives us access to modern datatypes such as `ndarray` and optimized arithmetic operations written with integrated c. Numpy functions as a base for nearly all modern computer vision algorithms.

Matplotlib is utilized for the visualization of the first milestone and the 3D plot of the fourth milestone. OpenCV is used in other milestones because of its camera integration, performance, and drawing capabilities. OpenCV also has feature detectors for camera applications that could be used to change the level of fusion of the system.

Pyserial was chosen for serial communication. For delays in serial communication and for framerate upkeep we used the library `Time`.

For the heavy lifting, we utilize You Only Look Once (YOLOv8), a state-of-the-art machine learning library by Ultralytics. This single-shot detector can identify objects by class, draw a bounding box over the object, and even segment the instance into either instance masks or polygons. On

top of the YOLOv8, an instance of BoT-SORT is seamlessly integrated by the Ultralytics team. BoT-SORT is a tracking library used to identify and keep track of object instances discovered by YOLOv8. The BoT-SORT algorithm is not fully integrated into the YOLOv8 framework as of yet. The re-identification task is still left undone by the authors at Ultralytics, the host company of the YOLOv8 library. The tracking algorithm is still found to work well.

Pyfilter is used to implement linear Kalman filtering and unscented Kalman filtering. Having done the introduction, we now move on to the implementation. The PC-side code is made open source by the author and is hosted in GitHub, where the reader can look at the code in more detail [37]. The code approaches used in this work are discussed on a high level, whereas the GitHub repository serves as the low-level information source.

4. IMPLEMENTATION

The work done in this thesis was done iteratively and is split into milestones with definite goals. With a new understanding of the sensor, the algorithms, the code resources, and the capabilities of data utilization new goals were set as the work continued and milestones were added to the pile. Next, we go through the phases of development in a time-ordered fashion.

4.1 Dense data

The first goal was to gather an RGBD image as many SLAM algorithms such as ORB-SLAM natively use an RGBD image as input. The D in the name refers to depth. RGBD images are normally obtained by using stereo vision cameras or time-of-flight based approaches [38]. In this chapter, we explore an approach using radar (which is a time-of-flight based sensor) to RGBD image creation. Together with the fused image, the data was to be examined and key observations made about its nature.

To get the radar data into the PC environment a device driver is needed. While the author did find implementations of TI mmWave device drivers such as [20] written in Python and [39] written in C, they both lacked the azimuthHeatmap decoding and visualization functionality that was deemed important for the map-making process in SLAM. By implementing a self-made driver, the data formats could also be explored more thoroughly. Therefore, a new device driver was written by the author from the basis of knowledge gathered from the JavaScript code on the PC side coupled together with the embedded system side C codes.

When all PC drivers are loaded and the EVM is setup in DCA1000 mode this exposes the PC to 2 separate UART channels coming from the device. These UART channels are driven by the MSS processor which controls the peripherals. The channels have their own COM ports, first of which implements the two-way command line interface and the second of which is used for one-way data streaming from the EVM to the PC. The first COM port is named the User/Application Port and implements an ASCII character-based command line interface much like a Linux command prompt. The device is configured via this port, and it can be queried for information during runtime if needed. The second port named Auxiliary Data port does not present the data in ASCII form but rather in bytes. The incoming data on this line is represented as type-length-value (TLV) frames.

The data channel has a little-endian byte ordering, and the order of items seems to follow the declaration order in c, although compiler optimizations can be seen to have an effect if optimization is allowed. The data transmission always starts with a magicWord = x"0102030405060708". This magic word can be listened to or searched from a buffer to find the beginning of a packet. The top-level packet then contains a header. In the header, information such as version number and frame number are represented. After the header comes the contents of the transmission. The list of packet types provided by the outOfBoxDemo is presented in table 4 together with its contents. The CLI port runs at a baud rate of 115200 baud/s while the data port uses a faster transmission speed of 921600 baud/s. In this milestone, we look specifically into the contents of AZIMUT_ELEVATION_STATIC_HEAT_MAP, which contains the zero-Doppler slice of the radar cube.

Table 4 Output data format types of the AWR18430AOP out-of-box demo.

Packet type as defined in C	Content of the packet
DETECTED_POINTS	DPIF_PointCloudCartesian * numObjOut
RANGE_PROFILE	uint16_t * numRangeBins
NOISE_PROFILE	uint16_t * numRangeBins
AZIMUT_STATIC_HEAT_MAP	azimuthStaticHeatMapSize * cmplx16ImRe_t
RANGE_DOPPLER_HEAT_MAP	numRangeBins * numDopplerBins * uint16_t
STATS	6 * uint32_t
DETECTED_POINTS_SIDE_INFO	DPIF_PointCloudSideInfo * numObjOut
AZIMUT_ELEVATION_STATIC_HEAT_MAP	azimuthStaticHeatMapSize * cmplx16ImRe_t
TEMPERATURE_STATS	10* rllnt16_t + rIUInt32_t + int32_t

The radar scan needs to be calibrated with the camera in order for the system to work as a whole. For this, the camera and radar need to be fixed with a known relation to one another. This was achieved by modeling and 3D-printing a case out of PLA plastic shown in figure 18. The modeling software employed was FreeCAD. The camera and radar are put atop one another. An additional access hole is put on the top in the location of the reset button of mmwaveicboost-board.

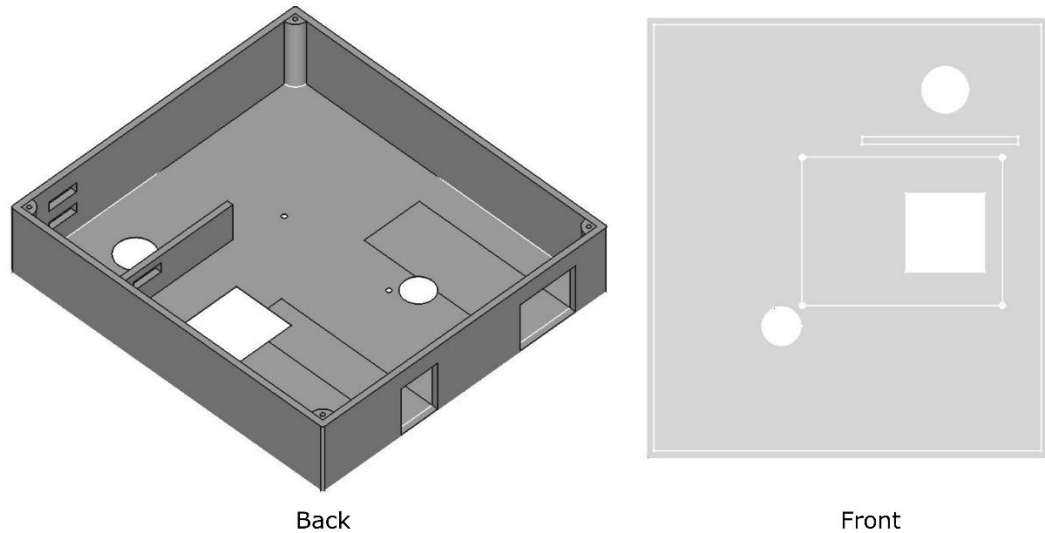


Figure 18 Housing for the radar and camera.

As we cannot put the sensors directly atop one another we have to define how the sensors relate to each other in 3D space. This is normally done by using a transformation matrix. A transformation consists of rotation and translation. The rotation is fixed to zero, so it is represented mathematically as an identity matrix. The translation matrix between the radar sensor and camera sensor middle points is defined as

$$\mathbf{t} = \begin{bmatrix} 0 \\ 0.005 \\ 0.055 \end{bmatrix}, \quad (39)$$

where the coordinates are presented in the form of x, y , and z in radar coordinates. The units represent the displacement of the image axis from the radar axis in meters. From the rotation and translation matrices, we can construct the whole transformation matrix. With this matrix we can translate 3D world coordinates perceived by the radar sensor into the coordinates of the camera. This is useful if we are to produce points of interest with the radar and try to relate them to the world map held by a SLAM system. For now, however, we choose to ignore the translation as its magnitude (5,5cm) is small in relation to the distances measured (10m). We discuss the validity of this approximation in the results section.

In this work, the radar system is used indoors. In the end application, the system will however be mounted on a vehicle. The end application mounting was first considered to be front-facing at bumper level. This would serve a good ADAS function as a collision avoidance radar. For SLAM, however, the sensor is better mounted in a place with high visibility to the static environment such as on the roof of the car. Many radars can also be used. A common approach is to put radars on all corners of a vehicle. In this approach, the sensor rotations will be nonzero. The

translation matrix method presented above will still hold but will have to be appended with the individual rotations of all the sensors.

To make the RGBD frame we need to translate the radar data from polar coordinates into the corresponding pixel locations in the image sensor. This is a common problem in computer vision and the geometry of the situation is shown in figure 19 where the depth D and angle ϕ are known and x is the coordinate on the x-axis that interest us. By using trigonometric definitions and by noting that the angle ϕ is constant between the two triangles, two congruent triangles are formed. The comparison between the sides of these triangles leads to an equation, from which the x coordinate is found.

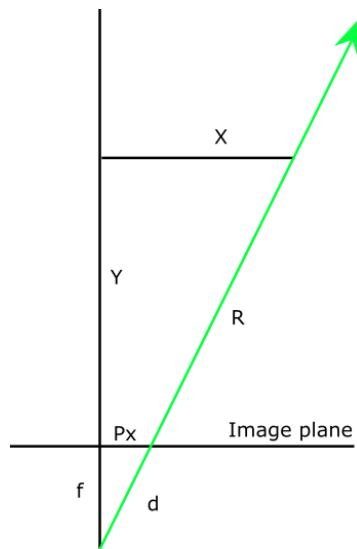


Figure 19 Definition of geometry in 2D. This figure extends the definitions of radar coordinates in figure 12 by adding the image plane and naming conventions.

The x dimension then needs to be fit into the pixel environment by discretizing it to pixel values. From this equation, singular pixel placements on the camera's image plane will be acquired. However, as the radar image is much smaller in size than the camera image in pixels, we need to use interpolation in order to cover the blank spaces left without a value. The camera image is of size 480x640 pixels whereas the radar azimuth slice is 12x256 pixels. The number 12 comes from the number of virtual antennas and 256 is the NUM_ANGLE_BINS. In the JavaScript approach, TI interpolates the result by padding the azimuth-FFT by zeros. Interpolation allows the system to represent values between the beam width sizes. TI presents its results in a 64x256 format which accounts to 433% more pixels.

The projection is done by finding the focal length equivalent, f described in figure 19. This parameter is found by performing an experiment. Radar points are hand matched to image coordinates by measuring a strong corner reflector outside in anechoic conditions. The 3D point and

the corresponding 2D image point corresponding to the center of the reflector are stored in Excel. We can then find the linear coefficient f from the geometry definition in figure 19 as

$$Px = f \cdot \frac{X}{Y}, \quad (40)$$

where Px corresponds to the x-coordinate of the measured 2D image centered at the origin and X and Y correspond to parts in the chosen radar point. This will produce a linear regression we can solve with the least squares approach. The results are presented in chapter 5.

Using this approach, some radar points were found to translate to pixels outside the desired camera frame. The camera thus had a lower FoV than the radar. The out-of-FoV data could be used to predict objects coming into the frame before the camera has a chance to capture the object. For now, however, we choose to ignore all detections outside the FoV of the camera.

In the data gathered, strange false reflections were found to be present. Their intensity was often higher than real objects and they were present even when aiming the radar sensor directly toward the sky where no reflection sources exist. The strangely large and static reflection values were found largely in the close-range bins. The observable falseness of the data was lesser in the bins far away from the sensor. It is probable that the false data is due to RF-power leakage from the transmit circuitry to the receiver. Leakage of the TX signal straight to the receiver without a delay would make for a close to zero mixer output and thus produce the perceived 0-depth clutter. This leakage is relatively static and can be calibrated out by taking a measurement of an empty scene with no reflections. The values in this calibration image result only from leakage. This calibration image can be subtracted from the incoming signal which results in cancellation of the digital value of the leakage. A concern of the author is that the assumption of static leakage might not hold under changing circumstances such as temperature changes, aging, or voltage fluctuations on the power lines. The calibration procedure for averaging out a number of the closest bins and down the line subtracting the previously measured bias from all further data is called `calibDcRangeSig`-calibration. The results of this calibration are explored in the results section.

The demonstration given by the author to the code division for the end of the first milestone contained live video and dense range-azimuth feed in polar coordinates of which a snapshot is given in figure 20. On top of the image frame, an interpolation of the projection of a polar-form radar frame is presented as 1D lines. The idea of the lines was to get a first verification of the projection used to fuse radar data to camera frames. In the image we can see the intensity value maximums shown in green align with the circular metallic reflectors, indicating that the value of f , the focal length equivalent previously calculated by the Excel experiment, is indeed correct. In

addition to reflection strength, the depth at which the reflection maximums were observed is plotted in red.

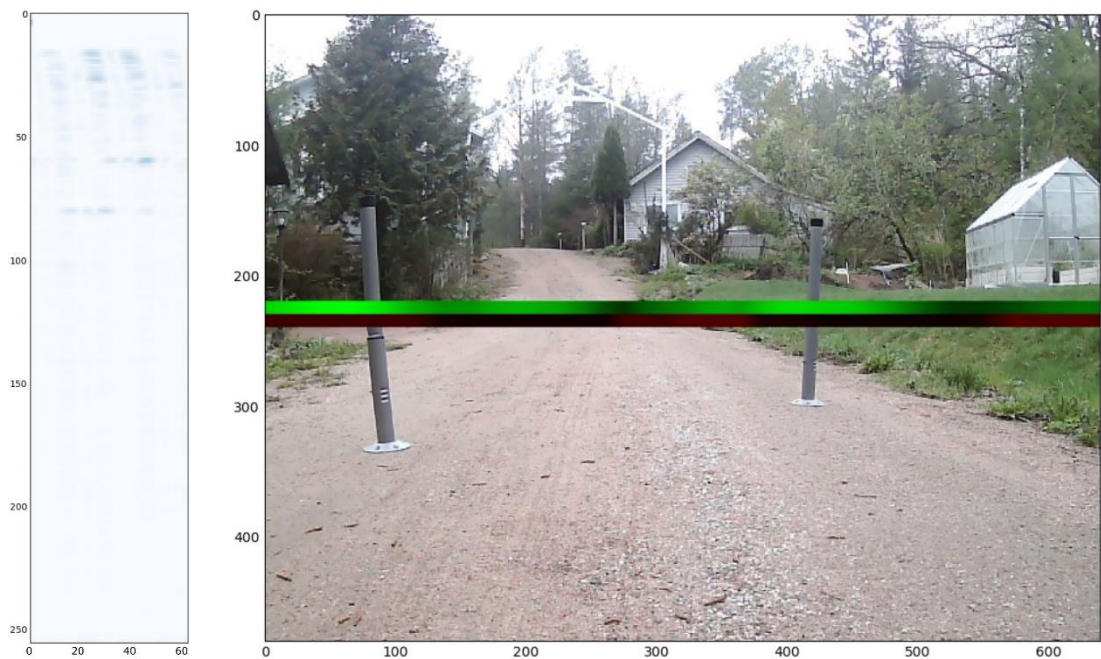


Figure 20 Range-azimuth radar frame (left) fused onto an image (right) after DC-calibration. The intensity of the green line indicates the strength of the reflected signal, and the intensity of the red line indicates the range of the maximum reflection. A brighter color indicates larger values. The x-axis of the radar frame corresponds to the angle, the y-axis to the range, and color to the received signal amplitude.

In this dense data representation, the temporal data alignment was handled by assuming radar to be the slower sensor. When a camera frame is taken directly after the radar frame the time instances are close enough to be regarded as the same. The spatial alignment was done via measurement where f was found. The focal length equivalent f , when used in projection, aligns the data of the sensor modalities into the same reference system, the system being pixels in the image plane.

The large communication bandwidth required by this dense range-azimuth slice limits the system fps to 1. The limit is due to EVM specific implementation which is itself limited by the USB speed. The slowness of the dense data format made the author question its effectiveness and explore the use of a CFAR feature detector to move into a sparser feature-representation-based data format instead. This was also due to the TI demo platform's promise of achieving up to 30 fps with this data format change. The full source code of the first milestone code can be found on GitHub as `get_input.py` [37].

4.2 Sparse data

A second milestone was set after additional design space exploration to get over the bottlenecks of the first milestone. The goal was to make the detections more accurate, reduce the need for further processing, reduce the amount of communication, and move to a 3D world coordinate system, thus moving to use full capabilities of the 4D radar instead of bare slices of the radar cube. This would entail putting to use all the virtual antennas. These design goals were approached by starting a move from the range-azimuth format into the point cloud data format.

In the exploration phase, an open-source library [20] was found that contained a different approach to device drivers. It also implemented an extraction model for point cloud data packets. The library functioned as the basis of the following code as it was found a better idea to use buffers and magic word searches rather than to listen for incoming data until the magic word appeared. The driver implementation was optimized by removing all expensive array copying. Temporal data alignment was also changed so that the driver returns all buffered data upon request instead of the last one in a FIFO manner. This allowed us not to lose any radar data points even if other system components were limiting performance. This meant that the temporal signature of a radar sweep was purposely lost and instead the system-level information gave us a system-fps limited delta-time inside of which we interpret all the detections as having occurred.

The point cloud is made by CFAR detection and AoA processing on the range-Doppler plane. Unlike the dense data, this sparse data now takes into account all the Doppler dimensions instead of just the 0th bin. An object recognition task is implemented on-sensor in the demo. It groups the detected points in range, Doppler, or both of these directions. The grouping means that fewer points need to be sent through the communication channel making the communication more efficient. We can compensate for the reduced number of points by thinking of the detected objects as larger objects than the singular point. This can be accomplished later in the radar to image data fusion. While the grouping algorithm lessens the number of points on the point cloud, it can't make a weak signal not appear at all. It only suppresses the number of points emanating from larger objects. Therefore, the grouping is not dangerous in terms of missing potential detections. TI refers to this process as peak grouping.

The detection accuracy was assessed by looking at the range spectrum together with the points output by the CFAR and peak grouping algorithm. A worry arose that the peaks that objects produce were hardly noticeable from the noise floor by human inspection and that the noise

floor itself was not flat. Test setup consisting of moving objects was then made by either a human moving in front of the sensor or by moving the sensor around in a room with 2 radar reflectors. From tests, it was confirmed that the CFAR algorithm also often failed to find real objects among all the clutter. To this end, an algorithm called static clutter removal was trialed, which made it so only objects in motion or motion of the camera itself produced peaks on which the CFAR algorithm was run. The static clutter removal is therefore a pre-processing step for the CFAR algorithm which helps it find peaks better. The results were a significant increase in true object detection from among the clutter and visibly better peaks for the CFAR algorithm to run on. The range spectrum in a test scenario of radar moving toward two radar reflectors is shown in figure 21 with no pre-processing and together with the static clutter removal algorithm.

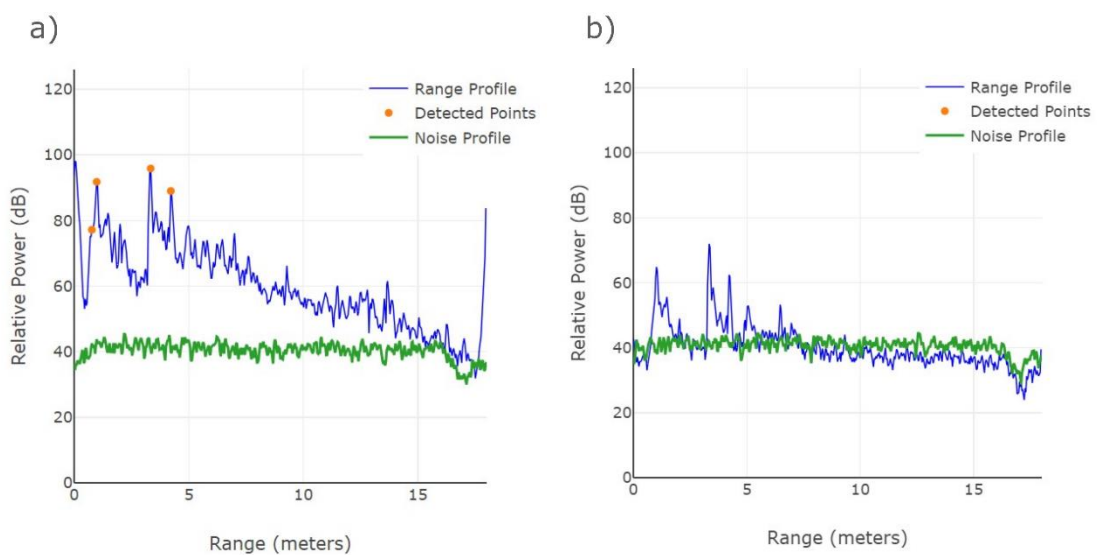


Figure 21 a) Range-power slice with CFAR results as orange points b) Range spectrum with static clutter removal pre-processing. Both measurements are made from the same environment. In b) the sensor had to be moved slightly and the measurement made during movement. This is because the static clutter algorithm only observes objects in motion and will detect a flat spectrum for a stationary scene.

With the static clutter removal algorithm in place, the radar started to pick up on smaller objects and reacted also to smaller motions. Where previously a moving metal corner reflector could be missed the radar could now detect even the slightest motion of a human without any reflectors. The radar cross-section difference of these objects is 4 orders of magnitude [40]. The addition of static clutter removal was the single most influential choice in this work. The point cloud was now also better in terms that it only contained points of interest. What interests us in radar are things that move. If a car doesn't move relative to you, it can't hit you. Therefore, no points are needed from those static objects. From a SLAM context, it is also true that a map maker cannot obtain any more information by standing still. Motion is inherent in the map-making process as

the observer has to move around the map in order to discover it. No crucial information is therefore lost by putting the radar in a mode where it sees nothing if there are no changes. Only changes have information. The static clutter removal algorithm also massively improves the effects of antenna crosstalk as it is also regarded as clutter by the algorithm and therefore removed. This allows the device to detect objects in the previously signal-congested space in the near vicinity of the radar in the 0 to 1 m range previously congested with noise.

The visualization library was also changed to OpenCV. This was done to accommodate the higher new system fps which was previously radar-limited but had now become software-limited. With Matplotlib the average was 2 fps, whereas OpenCV delivers 10 fps. The end deliverable contained a live feed demo of which figure 22 is a snapshot.

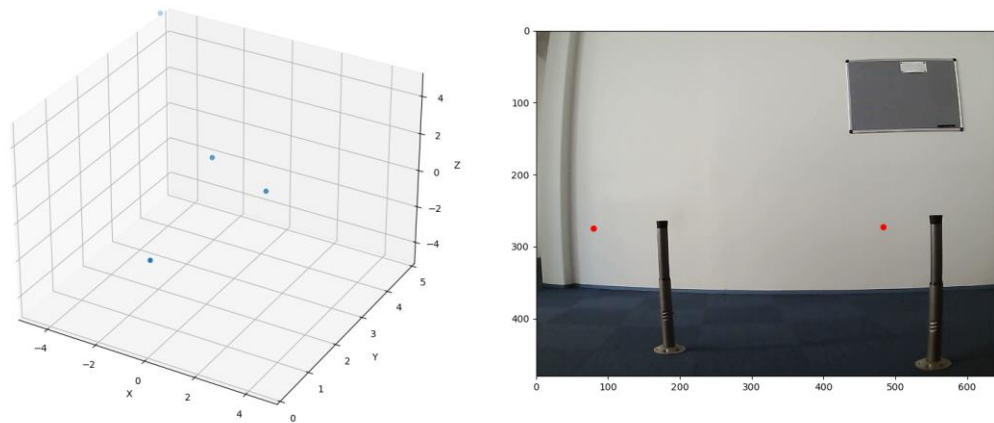


Figure 22 Point cloud objects and their projection onto the image frame. A sign of bad calibration is evident in reflections not perfectly aligning with the cylinders. This can be fixed with `compRangeBiasAndRxChanPhase` or by physically adjusting radar alignment, which may have slightly changed due to loose connectors.

This demo contained the move to the new data representation together with projection, 3D plotting, and the effectiveness of the static clutter removal algorithm. The second milestone provided the basis for further implementation of data fusion to finally begin in an object-level context.

4.3 Object detection, tracking, and sensor fusion

The third milestone was to include target identification and tracking into the system. With the camera track and radar point cloud, a sensor fusion was to be made. This sensor fusion was deemed very lucrative as the two sensing modalities are complementary to each other. A camera sees light coming in as rays but does not know how far the object is. Only the relative offsets from the image center are observed. This results in perspective distortions as the camera cannot decipher a small object near it from a large object far away. Radar however is an absolute measurement device. By fusing the camera ray with a radar point we can in essence cut the ray of the camera and therefore lose the ambiguity of perspective. This allows the fused data to have better accuracy compared to the standalone sensors. This is done by using the radar which has better depth accuracy. Radar however lacks angular resolution, but thankfully the camera is complementary to it in this instance too. The sensors complement each other's weaknesses. The same pros and cons of sensor modalities also hold for the velocity measurement: Radar is good in radial measurement whereas the camera wins in axial measurements.

For object detection the latest version of a well-known single shot detector machine learning model YOLOv8 was chosen. On top of YOLOv8, an implementation of BotSort tracking algorithm utilizing Kalman filtering is used to keep track of the camera detections. The YOLO detector is used with pre-trained weights. The model was chosen as the smallest model with the least layers for efficiency reasons. It was found to produce very good results in human tracking and can simultaneously track multiple humans in an office environment. The detector is trained on the COCO dataset which allows us to generalize our algorithms to all of the object instances in it such as cars, bicycles and stop signs. The machine learning algorithms are run on the graphical processing unit (GPU) to achieve an inference time of 20ms for a single frame with tracking which corresponds to a theoretical maximum speed of 50 fps. The system is no longer bottlenecked by CPU processing speeds. The web camera fps is the smallest averaging around 10 fps whereas the radar can achieve an fps of 30.

The differing speeds of the sensors call for temporal alignment [27]. In this milestone, the camera is first read which calls the radar sensor to take a snapshot. Radar data is read from a buffer acquired during the delta time, which makes it fast because it does not have to wait for sensor execution. The machine learning is image-based and does not need radar data to function. It can be run directly after a camera frame is captured. The implementation however is polling, which makes us wait for the results instead of being able to do calculations on the CPU while the GPU calculates the outcome. In this milestone, we extend the delta time to stop only after the ML task has finished. Thus, radar detections during the computation of object detection are counted

to be inside this processing frame. This approach to temporal alignment is referred to as asynchronous. This is because we do not have a constant sampling period. This approach was taken due to the varying fps of the host systems. This system when run on a laptop achieves considerably lower performance than is required to read data in real time. Temporal alignment of a 30fps radar to a 10fps camera running on a 3fps laptop is thus reduced to 3fps time frames. The bottleneck is the laptop's lack of an Nvidia GPU for neural network processing.

For the sensor fusion, an intuitive approach was used. First, YOLO finds and segments all the objects in the frame. BotSort then assigns ID:s and ID-specific position and velocity tracks for the objects. BotSort uses re-identification which aims to keep the same ID for an object that was previously lost due to occlusion or going off the image area. The radar point cloud is then projected onto the image. If a radar point lies inside the object segmentation mask then it belongs to this ID. Most of the time not one, but many radar points hit the tracked object. It remains the job of the sensor fusion to decide how to pick the point, average of points, or any other combination that will produce the most robust, noise-free, and non-divergent output as the final measurement for the depth and speed of this object.

At first, the median of the incoming points was chosen as the final measurement. This was chosen because a median will pick one of the measurements instead of mixing together data from multiple different reflections. Later it was deemed that this might lead to excessive discarding of data and the inclusion of measurement noise into the output. This is because the delta time might be long and there might be 10s of measurements to choose from. Picking one will surely not leverage the full potential of all the 10 measurements because their information is not included in the output. A measurement also always contains noise and by picking a non-filtered measurement we include this noise. A change was then made to pick the point using the average. This resulted in less noise.

There was however a problem in both approaches. A radar echo does not always come from the object itself but may be bounced from objects around it on the forward or the return trip. This makes the path look longer. This may also give the illusion that an object has multiple objects lurking behind it. Thankfully, the fusion approach of fusing a camera to this radar measurement mitigates some of the multi-path propagation issues. This is because a multipath beam will be seen as emanating from the place of reflection and not the object sending it. For example, a person may have a radar “shadow” next to him when he goes near a wall. The camera segmentation mask is genius in that it will segment only the line-of-sight path to the object as seen by the camera. This lessens the inclusion of multi-path reflections, but sometimes the mask is not perfect and reflections that are not in the line of sight are counted in. An important observation was made that the line of sight is always the shortest path to the object. Therefore, we would want to try and choose our point based on the minimum observed distance. This is the third approach used in this work. Minimum filtering enforces physical propagation path limits and gives us more robust and accurate information.

In testing, it was observed that sometimes we will also have zero radar points or only multi-path reflections hitting the object. In these cases, the result given by any of the 3 algorithms will fail. Thinking that a multi-path reflection is the actual object may change the object’s distance by over 5 meters. This results in very noisy output data from the fusion stage. To this end, a time window was trialed. Previous fusion outputs were put into a window and another run with one of the 3 point-picking algorithms was run. Different window sizes were explored. Their effects were to lessen the radical errors created by a few erroneous measurements by incorporating previous temporal information. The effects of windowing and the use of the 3 different approaches to point picking are discussed in the results section.

With the system in this state, we can track multiple humans at once and know their radial distance and velocity. The third milestone was demonstrated working at 10 fps due to the slow capture rate of the web camera. A remake of the scene was run on a laptop with 2 fps and is shown in figure 23.

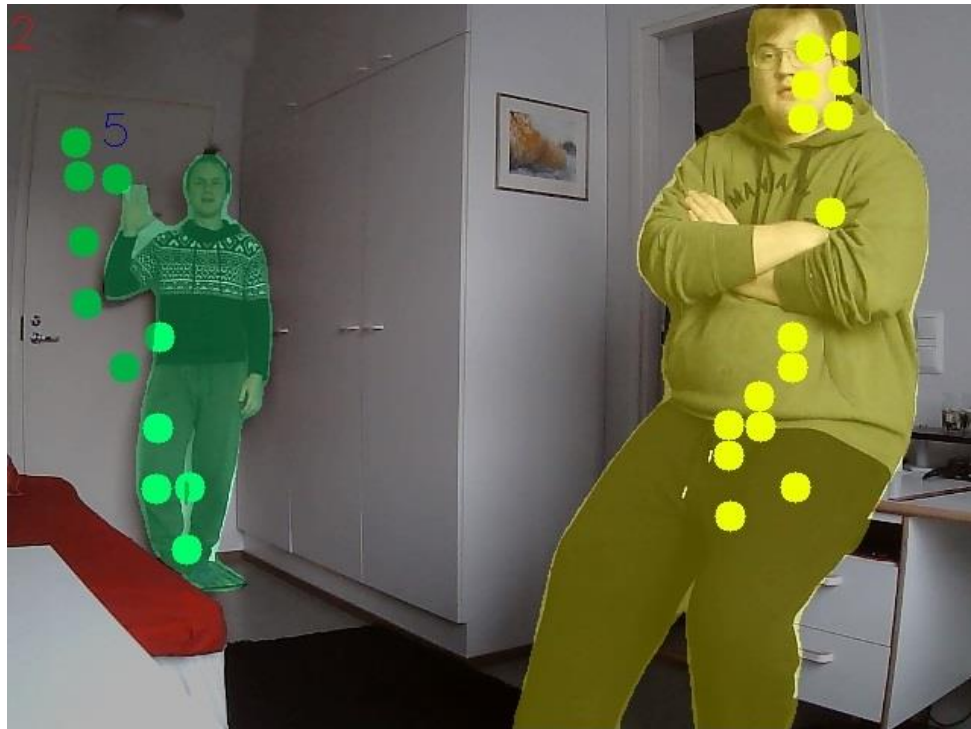


Figure 23 YOLOv8 combined with BotSort is able to track multiple objects at multiple distances. Object depth is illustrated with color. Object ID:s are printed on the top left corner of the mask.

In this demonstration, it is visible that a track can be kept from multiple people at multiple distances. The re-identification works for occlusion but going off-screen often tricks it into giving the same person a new ID. This is because the re-id feature of BotSort is not yet fully incorporated into YOLOv8. The system is later modified to track other objects from the COCO dataset including cars, bicycles, and motorcycles by changing the class-variable of the segmentation model for the YOLOv8 object detector.

4.4 3D representation using complementary sensor aspects

So far, the true speed vector has been unattainable as radar only measures radial velocity and disregards tangential velocity. The true speed is also not possible to be ascertained from a picture alone as the camera has a 2D viewing plane. The fourth milestone was set to fuse the two modalities into a true motion vector and to do so using the information from the more capable modality.

The implementation focused again on the idea of “cutting the light ray”. Because the depth is crucial to the idea of ray cutting resulting in a correct measurement, we want to get rid of meas-

urement noise and disregard spurious outlier points from measurements. To this end, we implemented both a linear Kalman filter and an unscented Kalman filter and compared them to one another. To get the Kalman filter working in this context many decisions were made, the first of which was time-related.

The Kalman filter uses a constant Δt as the time between the previous and current measurement-predict cycles and embeds it in the calculation of its matrices. The Kalman also assumes a single measurement per timestep. This is why our fusion approach in the previous chapter focused on delivering a single output per timestep. The fusion approach taken accommodates a Kalman filter seamlessly. There was however a different approach that was never implemented but remains a curious take on fusion and filtering. Where our approach feeds the Kalman data at a fixed rate, Δt in doing so it must use a point collapse function like mean, average, or minimum to take an n number of points down to a single measurement. This reduces the information content. We could however choose not to discard any data and feed all the data to the Kalman. In the current work, the Kalman update is run once in a measurement cycle. In the second approach, the Kalman update would be called as many times as we got data. Every single data point would be included instead of most of the data being thrown out. This entails splitting the Δt into n equisized time slices and then running the Kalman update multiple times. The problem with this approach is that a change in Kalman Δt entails we have to recalculate the F and Q matrices. This would have to be redone for every single measurement frame, which would raise the computational cost. This is also against standard Kalman filter mathematics and a varying Δt approach to Kalman filtering is thus seldom discussed in literature. The second approach would however allow us to incorporate not only the best but all the data. In the end, the standard approach of constant Δt was deemed safer and more computationally efficient.

Next, we make a world model that the Kalman filter uses to predict the future state of the object. In [26] it is argued that the best model has the same order as the object it is trying to measure. The equations chosen to represent our model were the Newtonian motion models of place, speed, and acceleration. Euler integration approximation of the acceleration was also explored.

An additional data association method was added in this stage to determine outliers from real measurements in the fusion step. This was able to be done because with the advent of Kalman filtering we now have memory and a trust estimate of where our object is located in space together with motion models expressing where it could be in the next time instant. With this information, we can deduce that measurements outside the range of possible places given by the motion model are false measurements. The false measurements can be discarded. This process is referred to as gating. In this work, 2 gating methods were explored. The square gate compares

the system state and the knowledge of state covariance to the measurement by using two if statements. If the measurement is within 3 standard deviations of the expected range and within 3 standard deviations of the expected speed, then the measurement is assumed correct. The second gate was implemented with Mahalanobis distance, which is a way to calculate the distance of a point from a distribution. Only a single if clause is needed for Mahalanobis gating, but the arithmetic for calculating the distance has more computational cost.

The process- and measurement noise were fit manually and tested on multiple sets of real-world measurement data. The convergence, variance, and residual were used visually in the determination of these parameters. For the Linear filter parameters, we chose the following matrices represented in table 5. We chose the continuous white noise representation for process noise as switching to discrete noise had little effect.

Table 5 Linear Kalman filter matrices.

\mathbf{x} , state vector	$\begin{bmatrix} x \\ v \end{bmatrix}$
\mathbf{F} , state transition matrix	$\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$
\mathbf{H} , measurement matrix	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
\mathbf{R} , measurement noise matrix	$\begin{bmatrix} 0.2^2 & 0 \\ 0 & 0.4^2 \end{bmatrix}$
\mathbf{Q} , process noise matrix	$2 \cdot \begin{bmatrix} \frac{1}{3}\Delta t^3 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 & \Delta t \end{bmatrix}$

In the above table, the effect of Δt on the matrix calculation can be seen. To tie this into system context this means that a platform with higher fps will have less noise in its process while the measurement noise holds constant. The fps of the system will thus affect the Kalman filter trust balance of measurement vs prediction, giving more trust to the prediction the higher the fps becomes. This is because the time over which we predict is shortened with higher fps.

The test data, visualization, gating, windowing, and point-picking algorithms are all available in the project GitHub repository under the file `fitKalmanFilter_radar.py` [37]. This file enables a quick verification platform for any future filtering approaches. The effects of these approaches on sensor fusion and filtering are discussed in the results section.

Now that the object-specific radar measurement has been filtered, we can move on to calculate the true position. This is done utilizing the same equations used inside the radar SoC, namely, equations (30)-(32), but this time the angles θ and ϕ are calculated from camera pixels using

the value of f attained in milestone 2. This way the better angular resolution of the camera is used for θ and ϕ and the range is left for the radar.

As calculations of angles using trigonometric functions such as \sin , \cos , and \tan are computationally expensive operations we want to avoid them. This can be done by defining them by using the congruency of triangles and the Pythagorean theorem as

$$\cos(\phi) = \frac{f}{\sqrt{f^2 + Px^2}} \quad (41)$$

$$\sin(\phi) = \frac{Px}{\sqrt{f^2 + Px^2}} \quad (42)$$

$$\cos(\theta) = \frac{f}{\sqrt{f^2 + Py^2}} \quad (43)$$

$$\sin(\theta) = \frac{Py}{\sqrt{f^2 + Py^2}} \quad (44)$$

where the notation follows the one presented in figure 19. Having now given the equations for true position we now move to the determination of true velocity.

The first approach to true velocity is to take the derivative of the true position. This works, but determination by position alone disregards information. This is because the inherent speed measurement given by radar is not used when we only calculate the derivative of position. This approach was implemented but later discarded as the idea of losing information by choice was not a good design approach.

Our approach to encompassing all speed information starts by noting the radar's inability to measure tangential speed and the camera's inability to measure depth. The vectors presented to the Kalman filter also have to align in direction in order for the Kalman filter to be able to sum them up. The speed vectors on the image frame and from radar naturally face in different directions for all but the boresight case. This calls for vector alignment as our Kalman filter uses only the magnitude portion of depth instead of directional vectors.

The first approach to alignment was to take only the y -axis component of radar speed and fill the x - z motion in from the camera. This approach however suffered a fatal flaw referred to as the cosine effect [41]. The cosine effect causes us to get a speed measure that gets smaller as the angle of the object gets further off-center. This is due to the components of the true speed aligning with the tangential part of the ray the radar cannot measure. The cosine effect is illustrated in figure 24.

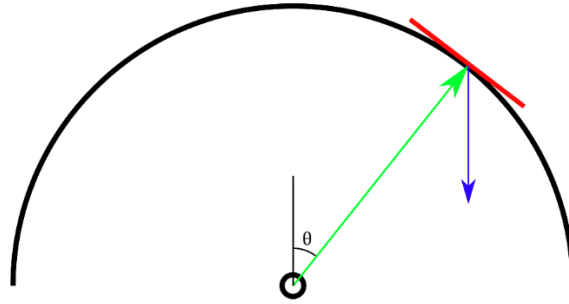


Figure 24 Illustration of the cosine effect. The larger the angle the beam (green) has to make to the object having a true speed (blue), the more of the components of speed align with the tangential direction of the beam (red).

As a result of the cosine effect, this approach to true speed was dropped. The other way to align speed is to use radial speed instead of the Euclidean y-component.

The radar sensor gives its results in Euclidean form as $(x, y, z, \text{radial speed})$ -tuples. In the Euclidean approach, we took the y-component of radial speed and gave it to the Kalman filter. The second option is to calculate the radial range from $x, y,$ and z and combine that with radial speed. This way the vectors given to the Kalman filter face the same direction and can be used in arithmetics.

In the radial approach, we have to modify the camera speed to be tangential to the radar beam and to the radial radar speed vector \bar{r} . This can be done by taking the beam direction, \bar{b} as the position vector of the detection. The radial component can then be made a unit vector, \hat{b} by dividing by its length. To take away the radial speed components from the camera speed vector, \bar{c} and turn it into the tangential speed, \bar{t} we can use the dot product between the camera speed and radial speed to get the length that needs to be removed, multiply that by the radial unit vector, and remove the component by subtraction. This can be expressed as

$$\bar{t} = \bar{c} - (\bar{c} \cdot \hat{b})\hat{b}, \quad (45)$$

which gives us the camera motion component in a 90-degree relation to the radar beam. This is equal to determining the red arrow in figure 24 using the camera to fill in the information the radar can't perceive. The tangential component formed by the camera and the radial component formed by the radar can now be added together to form the total true speed vector as

$$\bar{v} = \bar{r} + \bar{t}, \quad (46)$$

which now incorporates both the radar speed measurements and the velocity of object centers on camera. We know \bar{r} as the Kalman filtered radar result, but we now have to find a way to measure the speed of objects on the camera frame, \bar{c} .

A camera speed measurement natively exists inside the BotSort tracker utilizing Kalman filtering with speed in the x, and z-plane as hidden variables. The implementation is however terrible for speed estimation as the library implementation of the Kalman filter does not take into account the frame rate of the system and assumes $\text{fps} = 1$. This causes speed not to be sensible. Two approaches to camera speed estimation are explored: Kalman filtering of the center point of BotSort tracked objects having speed as a hidden variable and a direct approach of taking derivative of the object centers from frame to frame. The best approach for camera speed measurement is discussed in the results section.

We now have object-level position and speed estimates in 3D space. These estimates are drawn into a 3D coordinate system in order to visually verify their correctness. Exponential smoothing is also applied to the speed vector to slow down its jittery nature. The first deliverable of tracking 3D objects represented as dots with their corresponding velocity vectors represented as arrows pointing away from the points is represented in figure 25.

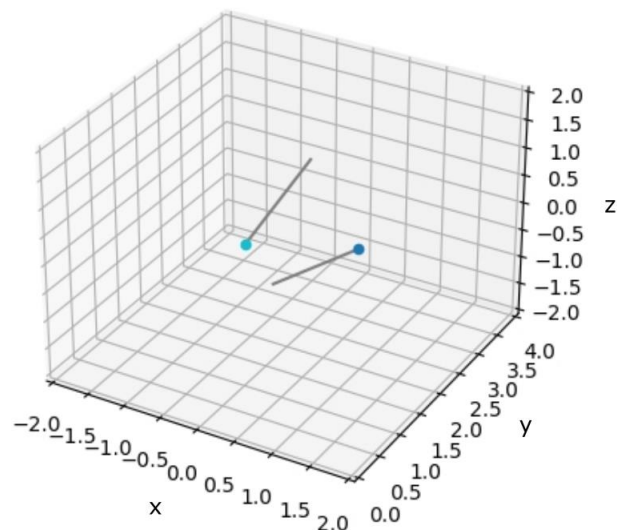


Figure 25 *Two people traveling in opposite directions. Lines indicate the true speed vectors of the objects.*

This object-level picture is a representation of the state of the fusion and tracking system at a certain time. A better demonstration was later made by integrating the object-level results into images. With this approach, we can demonstrate the motion vector of people. We can see the final deliverable of the fourth milestone in figure 26.



Figure 26 Demonstration of the true motion vector stemming from the true position of the calculated object center projected onto the image plane. To illustrate a 3D vector projection in the 2D image we encode motion towards the sensor with red and motion away from the sensor with blue.

In addition to the function of the system, focus was later placed on performance. The demo platform is limited by its communication bandwidth via USB 2.0 and is intended for demonstrations only. As described by TI, true applications should instead use LVDS or CAN-bus interfaces for communication. We can optimize the most out of the present channel by limiting the amount of redundant data. We do this by filtering out radar detections that are outside the view of the camera and thus can't have an effect on the fusion. This is done experimentally by guessing the maximum view angle of the camera, drawing the resulting box onscreen, and iterating the guessed value until the box barely fills the full image. This gives us the camera FoV. The radar also allows us to select an FoV outside of which points are not communicated. This lessens the need for communication bandwidth. The radar also sends additional point information by default, which contains point-specific signal-to-noise ratio and noise level information from the CFAR processing stage. This function can be disabled to save more bandwidth.

The motion model is also trialed in vehicular tracking. To this end, the radar parameters have to be adapted to allow for further range, sensitivity, and speed. This is done by sacrificing resolution. The results of vehicle detection expressed in milestone 3 combined with this motion estimation are presented in the ensuing chapters.

5. RESULTS

In this chapter, we look into the achieved system performance, effectiveness of the fusion approach, filtering results, bottlenecks, and observed secondary effects of radar sensing. In the final portion, we examine the silent knowledge -type of information that was learned by using the AWR1843AOPEVM sensor.

5.1 System performance

The system in its present state after milestone 4 can achieve a performance of 30 fps running on a PC with Intel Core i9 10900k and an NVIDIA GeForce RTX 2080 Ti GPU. This is bottlenecked by web camera fps and communication bandwidth of radar SoC. The radar point cloud was read from the device with a speed of 25 fps. The SoC could achieve the promised communication speed of 30 fps, but this often crashed the demo because of the limited bandwidth of the communication channel. By moving away from USB to LVDS, this bottleneck can be fixed, and a higher radar measurement interval can be achieved. The implemented code is system agnostic and can scale its performance with both added computational power from stronger GPU:s and by using other radar sensors with more capable sensors. There is no limiting factor for the number of radar points, resolution, or number of detected objects.

The radars part in the system context can best be explored by looking at the limits of its capabilities. The radar sensor maximum capabilities are represented in table 6. These maximum numbers are specific to the AWR1843AOP device and are not achievable at the same time. Rather, these numbers represent the possibilities of the sensor if all other aspects are compromised to achieve them.

Table 6 Maximum performance of the AWR1843AOP device [36].

Frame rate	30 fps
Range resolution	3.9 cm
Speed resolution	0.144 kmh
Max range	50 m
Max speed	56 kmh

Compromises are made between these parameters when setting up modulation schemes for the radar. This affects system performance and therefore two independent radar configurations are explored later on. The first configuration addresses short range radar in an office environment and the second pertains to longer-range vehicular tracking.

5.2 Sensor fusion

The goal of this work was to explore the use of radar sensing. As a part of this goal, we introduced a feature projection-based sensor fusion approach in which we project radar reflections to the camera domain. This allows us to associate neural network segmentation masks with the radar detections in order to fuse radar features onto image objects. This combines the two modalities allowing us to re-define the object speed and position by combining accurate depth information from the radar together with the good angular resolution of the camera.

In projection-based fusion, it was found that calibration is of much importance. Points from the camera must align with the radar in order to be fused. If there remains rotation or scaling error the points may not align with the masks resulting in wrong or no points being included in the object. This is sometimes addressed by enlarging the masks or by using more elaborate nearest neighbor or fuzzy logic matching. In a demonstration, a mask enlarging was trialed by making far away objects have an increased area. This was not found helpful in an office environment however as increasing the mask size might make two masks overlap. This will result in data association giving the data to the wrong object which produces entirely false measurements.

The area from which points are associated as belonging to this object was first chosen as the bounding box provided by Yolov8. In the latter work, a move was made to the segmentation masks. The effect this had was that fewer outliers made it past fusion. This is to be expected. Radar detection gives a valid range only for objects in the line of sight. The bounding box allowed non-line-of-sight reflections to also be associated with the object. The segmentation mask was also smaller in pixel count and a tighter representation of the line-of-sight path to the object. Some multipath reflections still made it beyond even the segmentation mask. This is evident in figure 27 where the movement of a person is followed by three “ghosts” at distances 10cm, 50cm, and 150cm caused by multipath propagation echoes of radar coinciding with the segmentation mask. Purple points represent all the points hitting the segmentation mask of this object.

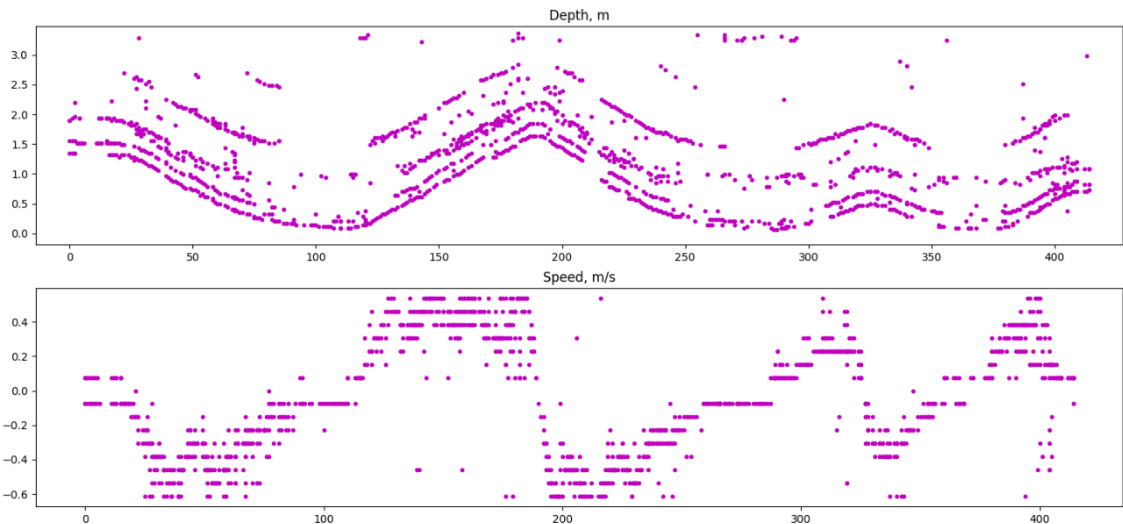


Figure 27 Effect of multipath propagation on sensor fusion. Only one subject is observed, but several echoes are present behind them.

The masking is regarded as the first stage of fusion. The above picture represents the input stage of the point-picking part of sensor fusion. Many avenues were explored in deciding a single measurement given multiple radar points hitting an object. The approaches were first explored inside a single frame. Approaches of median, average, and minimum-based dimension reduction were explored. The output of different reduction techniques is shown in figure 28, where blue points correspond to the points chosen by the given algorithms. From the figures we observe the advantage of minimum filtering over the other methods in that it retains the closest track of the object and does so with the least noise.

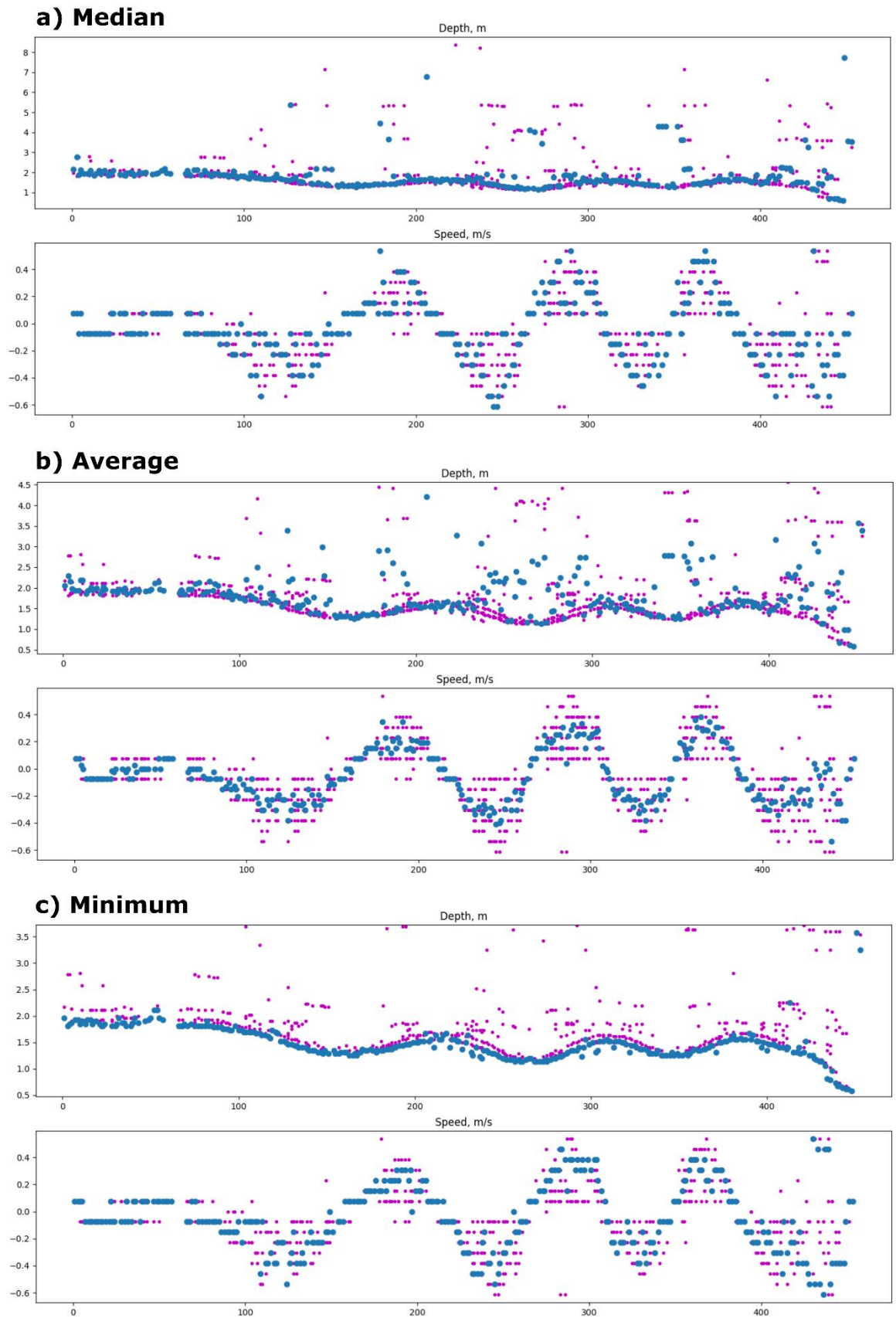


Figure 28 Filtering of points using median, average, and minimum filtering approaches.

Gating was also employed in the form of square and Mahalanobis-based distance metrics. The use of gating first requires the system to understand its state. This is given by the Kalman filter. In figure 29 we illustrate the Kalman state as an orange line with yellow bands illustrating the state covariance, P calculated into 1 standard deviation long bands. The gating throws out points outside these bands. Referred to as the gate size, a good length after which points are thrown out is argued in [26] to be approximately 5 standard deviations. In figure 29 minimum filtering is used with a linear Kalman filter. The gated points are shown in red. The minimum filtering allows us to use a tighter gate which is illustrated in the figure by dropping the gate size to 3 standard deviations.

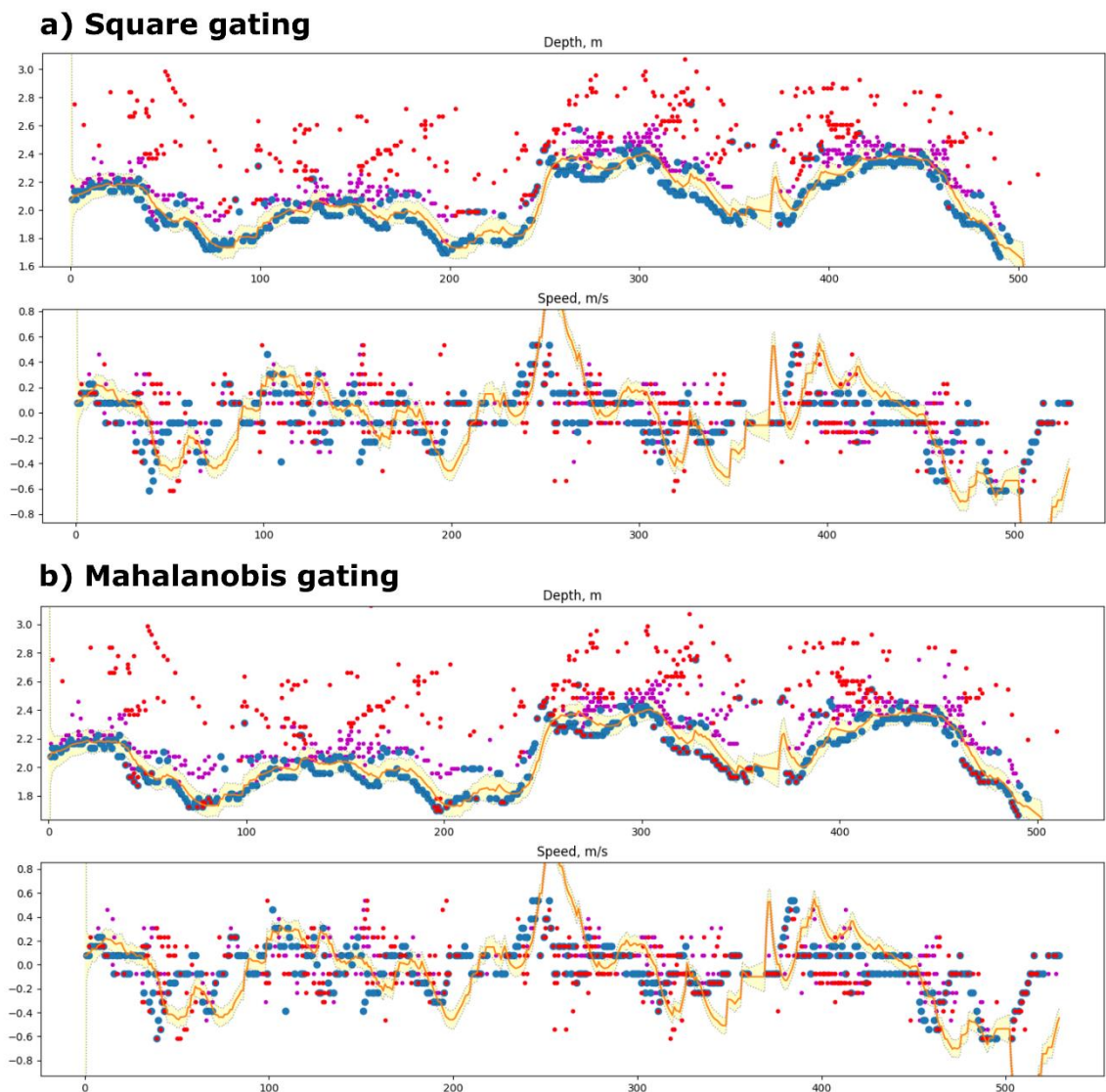


Figure 29 Comparison of gating techniques. Both gates have a size of 3 standard deviations.

From the figure, we can see that with the same gate size the square gate lets in more points. This is because its rectangular surface area is larger than the ellipse produced by Mahalanobis. A problem with gating is that in some cases the results may not converge on the correct track. This is very evident when multipath detections are present. The Kalman may converge on the multipath values instead of the true object path and because of gating it will regard the true measurement as false. The author advises that the minimum point-picking algorithm be used alone and that gating not be used at all.

Minimum filtering shows the most promise as it incorporates information about the propagation path length into the picking algorithm, whereas the mean and average are swayed by multipath propagation. As minimum filtering already disregards multipath propagation a further need for gating does not exist.

The addition of time into the fusion was performed by using a window of previously chosen points. This required a second use of the point-picking algorithm, this time in the time- rather than the space domain. Minimum filtering was chosen also for the base algorithm. In figure 30 we can see the effect of expanding the time view to windows instead of delta times. In it, different window sizes are compared to each other.

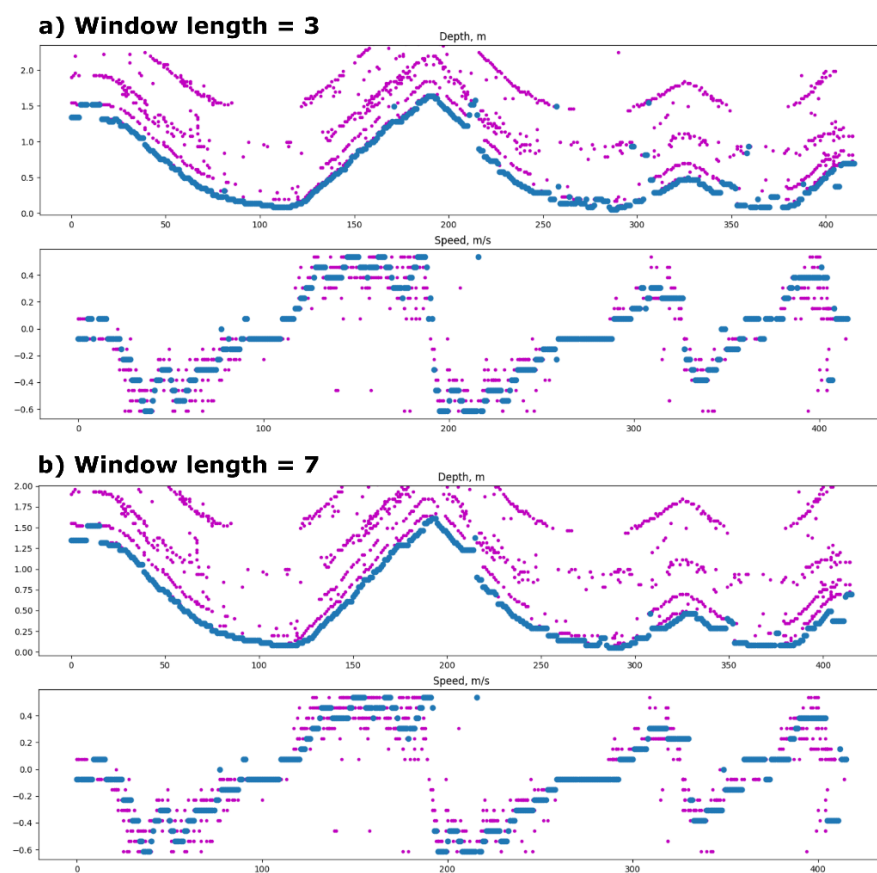


Figure 30 Window size comparison on data fusion output.

From the figure, we observe that the longer the window the fewer outliers make it past the fusion stage. This comes however at the price of a delay in objects moving away from us. The minimum filtering inside a window keeps the minimum value for the length of the window, so the length of the window affects the response time of the filter when values are rising. On a laptop with a frame rate of 30 fps a window of size 3 will result in a delay of 1 second. On a desktop PC with 200 fps, the same window will result in just 0.15 seconds of delay. This delay is however of no interest to ADAS applications. This is because rising values with added delay correspond to objects moving away from us. An object moving away from us can't collide with us. The objects of interest which are the ones moving toward us have no associated delay resulting from minimum filtering in the time domain. Minimum filtering is thus deemed appropriate for the time domain too.

In other works based on projection, it is sometimes a custom that the projection be made to a third independent domain instead of on the top of one of the sensor domains. This allows for sensor autonomy which is deemed the key selling point of radar in ADAS. Works such as these use feature level fusion instead of the mixed level feature-to-object level approach used in this work. In those works, the data association is made by point grouping in 3D space instead of on top of the 2D image. Sensor autonomy is desirable because radar is often used as a redundancy sensor for harsh environmental conditions. If the sensors cannot autonomously produce object level detections, then occlusion or failure of even one of the sensors might lead to loss of function of the whole system. Because our system associates points from the radar to the camera instead of associating any point from any sensor to an object, it does not have sensor autonomy. This is why the object detection and tracking system in this work best serves as a standalone detector that assists rather than fully defines the sensing for an ADAS application.

5.3 Kalman filtering

The most time-intensive part of fitting a Kalman filter is finding out the user parameters R , and Q which dictate how the filter gives out trust to measurements and predictions. We explored the fitting of the measurement noise matrix, R by performing experiments.

The first experiment was to look at the raw measurements coming into the sensor fusion stage from the projection. As no ground truth for human motion exists and conditions change a lot depending on the room and size of the target, the standard deviation of the measurement was

approximated by hand by drawing lines inside of which one standard deviation (63%) of points would fit. The first experiment looks into radar error.

The next experiment was performed for the output data of the sensor fusion with a minimum filtered window of 3. This was to reflect the effect that minimum filtering has on reducing the radar measurement error and to look at what the Kalman filter input will likely see as its variance. From both the experiments the worst-case scenario for measurement error was found for the `python/scatter_20fps.cfg` sensor configuration. The errors before and after the point collapse part of sensor fusion are shown in table 7. The measurements are done on sensor data and thus lack information about biases compared to real-world distances and velocities. The tests were run on the GPU-equipped machine as the long frame times of a laptop would drastically affect the results before filtering.

Table 7 Worst-case standard deviation of radar measurement before and after filtering.

	Before filtering	After filtering
STD_{depth}	1 m	0.1 m
$STD_{velocity}$	0.3 m/s	0.2 m/s

From these tests, we deduce that radar error can be large if the results are not filtered properly. The worst-case errors were found when multi-path propagation occurred. The minimum filtering acts to fix the multi-path problem. An interesting observation can be made about the errors of depth vs velocity. When unfiltered, the velocity appears to have less error. From the shortest propagation path of waves we can however deduce a minimum filtering logic which when applied will make the depth estimate more accurate than the velocity. It must also be noted that while the velocity error seems good in this format, when compared to the maximum velocity its problem becomes evident. With a maximum velocity of 0.63 m/s, the standard deviation even after filtering fills more than 30% of the entire range with ambiguity. In comparison, only 1.7% of the 12 m maximum range is filled with ambiguity in the filtered range measurement. The range measurement is thus a lot more trustworthy than the velocity measurement, and this should be taken into account in the measurement noise matrix R . Compared to the RF parameters for range and velocity resolution of this radar configuration, it is safe to say the true error is likely 2-3 times the theoretical resolution.

The linear filter approach was compared to nonlinear Monte Carlo sampling Unscented Kalman Filter by output comparison of filter state. No significant improvement was however noticed in moving to this approach. This is likely because the system that we are modeling, eg. an object in motion under Newton's laws, is inherently linear. It is stated in [26] that the linear filter is indeed

a perfect fit for a linear system and that the nonlinear approach of the UKF has nothing to add to the accuracy or convergence of the system.

Noise matrix was chosen as continuous white noise because no observable difference was made between it and the piecewise white noise in terms of filter convergence. The world model for a human in motion was best fit with a first-order Newtonian which assumes constant speed between sampling times. A second-order acceleration model was tested with both a full motion model and an Euler integration-based acceleration model. None of the acceleration models were found to accurately capture the acceleration of objects even under moments of speed direction reversal. What was worse however was that the addition of second-order terms into the model changed the model assumptions to assume a constant acceleration during breaks between samples. This led to large convergence issues. The issues were exacerbated on the laptop platform due to longer time intervals of assumed constant acceleration. This led us to conclude that a human is best modeled as a first-order motion model and to adjust our Kalman state accordingly.

A Kalman filter was also proposed by the BOTSORT tracking algorithm for filtering the camera-produced motion vector. For this, BOTSORT proposed a world model consisting of x, y, w, h , and their derivatives, where the state consists of the coordinates on the image plane together with the width and height of the bounding box of the object. In total 8 states were introduced half of which were hidden states. The model worked on an assumption of a constant sampling rate of 1 Hz. After testing, the proposed model was rejected because velocity measurements did not correspond to real-world speeds likely due to the sampling rate assumption which did not hold.

A Kalman filter of our own was then fit for the camera. The world model was chosen to be a 2-dimensional first-degree Newtonian model. From the camera domain, we want to obtain a smooth position with a responsive speed estimate. This means we need to give more trust to the measured position instead of the calculated position. Using these parameters, the Kalman was then compared to a naïve approach of simply determining speed as the derivative of position that we obtained from the BOTSORT algorithm. The derivative was all but equivalent to the Kalman filter output. Thus, a derivative of the BOTSORT position result suffices on its own. No extra Kalman is needed for camera motion estimation.

5.4 Calibration

In the first milestone, we find that system calibration can remove interference in the first 32 range bins as shown in figure 31. In addition to this `calibDcRangeSig`-calibration option, the `compRangeBiasAndRxChanPhase`-calibration was also run. Although hard to notice from the figure below, the `compRangeBiasAndRxChanPhase`-calibration will shift the azimuth bins so that the center bin aligns with the beam emanating from the sensor center. The DC-removal calibration was found useful only for dense data, as the use of a clutter removal algorithm has the same effect of mitigating crosstalk between Tx and Rx. The effect of phase compensation was small, but the alignment function it fulfills is critical to the fusion, where both camera and radar must face the exact same way.

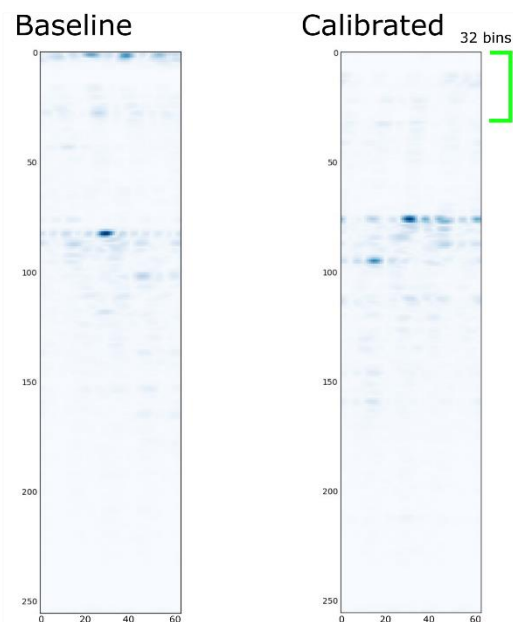


Figure 31 Effect of `calibDcRangeSig`-calibration on dense data.

The point-to-point matching sequence to find f , the focal length equivalent transform from meters to pixels was performed by using a radar reflector outside in an empty field, where no other echoes were present. The results of the point-to-point matching run through the projection function (40) are represented in figure 32, where the radar z-axis (image y-axis) is negated in an effort to align the coordinate directions of the camera and radar.

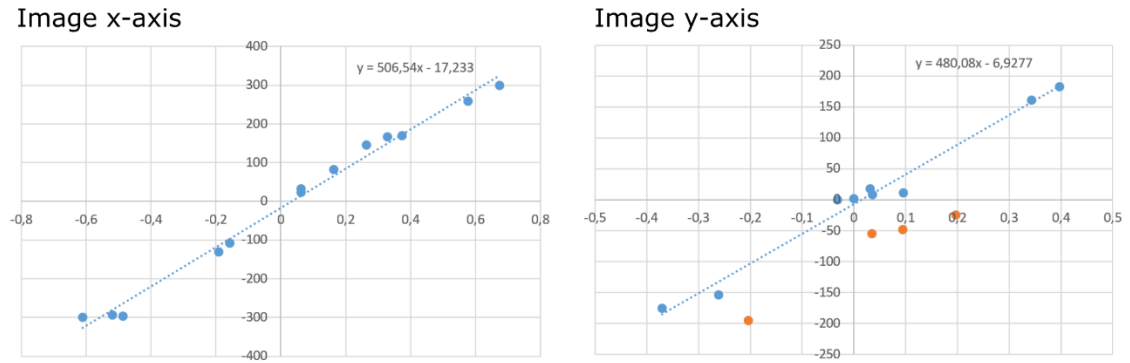


Figure 32 Point-to-point matching results for finding the slope of the line, which corresponds to f . Orange points have been regarded as outliers and excluded from linear regression.

The average of the azimuth and elevation measurements was calculated to finally arrive at a value of $f = 493 \text{ pix/m}$. In the projection-based approach of comparing the center point of an object to its calculated 3D-to-2D projection calculated by equations (30) to (32), we found that f is actually closer to 900 pix/m . This number given to f will project a point in 3-dimensional space to its image plane starting point without error. The projection approach is correct, and the spreadsheet approach is found to be lacking. The reason for the different outcomes is found from the range of the sampled angles. The point-to-point measurements taken were not far enough from the centerline to find that f is actually larger. The effects of a larger f become evident only on the very edges of the image. The error made using f from point-to-point measurements was so small in implementation that nearly all points fit correctly inside the bounding boxes given by YOLOv8. The effects of bad calibration for f are thus deemed not that important. The FoV of the camera was also found by projection from 3D-to-2D space. An azimuth view angle of $\theta = \pm 20^\circ$ and an elevation view angle of $\phi = \pm 15^\circ$ were found to fit the whole image frame. These results are useful in relieving the demo sensors' largest bottleneck, the communication bandwidth, by filtering away points not seen by the camera.

The assumption that the radar and camera sensor origins align was determined to have very little effect and therefore no additional computation was needed. They can be compensated by adding the translation vector between the sensor origins to the coordinates. The error made is only the size of the translation vector defined previously as \mathbf{t} in milestone 1. The 5.5cm vertical offset and 0.5cm depth offset are thwarted by the radar sensor resolution.

5.5 Vehicular tracking

We can equip the system with a radar configuration file presented in table 8 and set the camera object detection to detect cars instead of humans. With this configuration, we take a look into porting the implementation to vehicle motion and position tracking.

Table 8 Radar parameters optimized for detecting small radar cross-sections at long ranges at the cost of resolution.

Range resolution	0.434 m
Maximum unambiguous Range	50 m
Maximum radial velocity	11.73 m/s
Radial velocity resolution	0.37 m/s

The approach of asynchronous sensor readout and delta times that were found to work well for human tracking did not work well for vehicular tracking. In speed measurements of vehicles, the varying fps of the system coupled with an assumption of constant fps for Kalman matrices sometimes caused the radar speed measurement to not agree with the velocity calculated from position and delta time length. This is illustrated in figure 34 where the Kalman filter calculates a different speed using range information than what the radar is claiming the speed to be by using doppler shift.

A laptop was chosen as the PC platform for mobility purposes. The laptop limited the system fps to 3 and might have contributed to the accuracy of detections because of the longer frame times associated with slower processing speeds. In the measurements, the maximum number of cars tracked at the same time was limited to 3. The maximum range from which a car was detected was 50m which accounts for the maximum range of the radar subsystem.

The limiting factor of detection was found to be camera-to-radar calibration and radar cross-section of targets. Calibration errors resulted in the projected radar points being slightly to the side of objects instead of on top of them as shown in figure 33. This leads to the masking stage of sensor fusion to discard the radar measurement. The problem is exacerbated because of the large distance between the camera and the object which means that the object mask occupies only a small number of pixels. This problem is sometimes avoided by using a vertical column instead of a mask in the data association step [42]. This however allows multipath propagation to corrupt results. The effects of bad calibration could also be mitigated by using fuzzy logic, nearest neighbor, or other more robust data-association algorithm to fuse the points to the objects.

The radar cross-section problem was addressed by changing the RF modulation to prefer sensitivity to resolution. The relatively small sensitivity means that a small car at 50 meters produces only 1-to-3 point cloud dots.



Figure 33 The Distant car is masked correctly but the radar data is not accurately calibrated to hit the car at a distance.

Because of these limitations, fewer points make it past data association, and therefore the output of sensor fusion can be sparse when compared to previous human tracking results. An example dataset of a car going past the sensor in a 40 km/h zone is shown in figure 34. In total 20 cars were measured with an average of 10.1 associations per vehicle during the 50m range of the radar. The speed limit of this road is 40 km/h and the radar measured a max speed for this vehicle of 9.9 m/s = 35 km/h. This vehicle had 11 associations which should be enough for the calculation of speed through the range. It is however evident that the direct measurement and filtered results are in opposition to each other. This could be due to the speed measurement being at the limits of its workable range. In this situation, the Kalman-filtered result is more trustworthy.

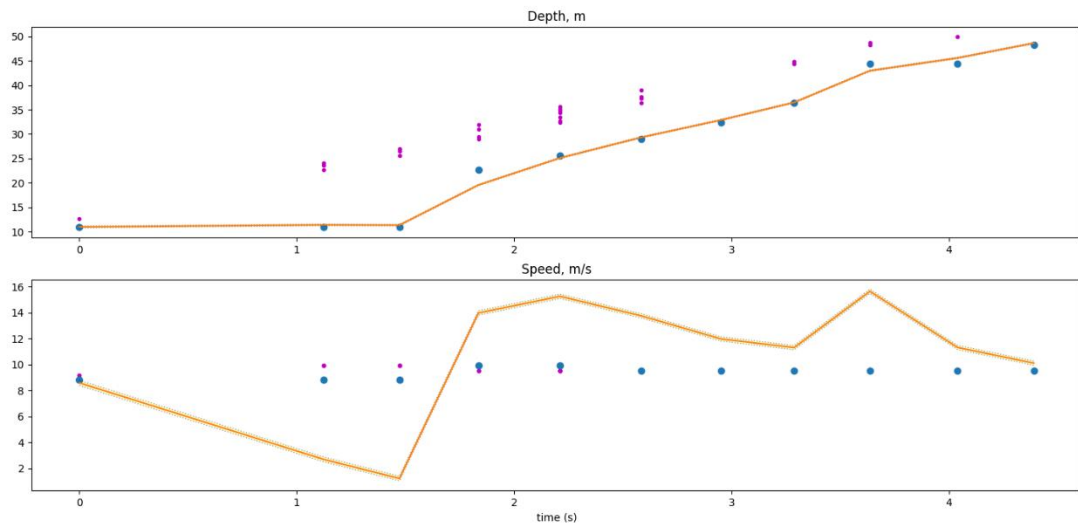


Figure 34 Example reflections from a car. The raw speed measurement differs from calculation through range leading to Kalman not agreeing with radar measurement.

In the mobile setup, the system performance is limited by PC processing power. With increased power, the system could run a more sophisticated neural network with more accuracy together with a faster frame rate which helps in tracking. Currently, the system uses a pre-trained YOLOv8 model called YOLOv8n-seg which is the lowest accuracy but highest performance network of all the available ones. For example, by moving to the high-accuracy network of YOLOv8x-seg the provider of the library estimates a 40% increase in the mean average precision of mask coverage compared to the current network. A network architecture change may also help in raising the total number of objects perceived at one time.

In conclusion, the system is shown to be adaptable to the vehicular context. In particular, cars were used for the measurement targets, which required changing the RF parameters to accommodate higher top speed and higher sensitivity. Bicycles and stop signs can also be selected as targets if desired. A longer detection range brought up the need for tighter calibration between the radar and the camera. A greater points/second speed from the radar would also be beneficial as that would make detection more probable. By moving to a larger neural network, the camera object detection could also be improved.

5.6 Inherent features of radar sensing

A goal of this work is to explore for Kovilta Oy, what are the ins and outs of radar sensing. This chapter addresses the practical knowledge obtained by working with the MIMO radar sensor from TI. This chapter addresses data. Comments are made on the form of the observed data, how the data can be manipulated, and what it takes to make this data.

The first observation of interest was made in the first milestone regarding the dense radar data. The range azimuth slices of the radar cube are cluttered with continuously present signals. These signals persist even if the sensor is pointed to the sky where no signal can reflect back to the receiver. The clutter is found to be the strongest in the first few range bins. The 0-range clutter was found to be due to antenna leakage from the transmit to the receive path. The 0-delay signal when mixed with the ramp signal produces a low frequency IF that is perceived in our sensing as a very nearby echo. Clutter signals also exist in higher frequencies as was evident in figure 20. A possible cause for the high-frequency clutter could be nonlinearities in the RF subsystem. The clutter makes detection of objects hard. The only way to decipher true objects from clutter is to look at changes instead of absolutes. The clutter signals remain stationary whereas signals coming from objects change when the sensor or object is moving. The CFAR algorithm will often fail to find actual objects from the scene if clutter removal is not used.

A problem inherent to radar, lidar, and self-illuminated camera systems is the doubled path length the light has to take to get to the sensor. In these systems, the photons need to get from the transmitter to the object and then back. This will cause a drop in power of $\frac{1}{r^4}$ relative to the distance of the object [21]. This limits the effective range of a radar system because of the diminishing returns that can be obtained solely by raising the transmit power of the system. In the AVR1843 the power is also transmitted to the whole view angle. The system has no beam. It also does not scan. It receives echoes from all directions at the same time and deciphers the angle of objects based on MIMO principles rather than from the direction of its antennas. This allows the system to observe its whole environment but reduces the power available for any single angle.

In comparison to our radar with an angular resolution of 30° , Lidar systems in 2020 already boasted angular resolutions of 0.1° or less [43]. We also get better resolutions on the camera side of sensing. An intel Realsense depth camera can produce a depth image of resolution 1280×720 at an FOV of $87^\circ \times 58^\circ$ [24]. This accounts to an angular resolution of $0.068^\circ \times 0.08^\circ$. Our AVR1830 system is not however a top-of-the-line product and by employing more antennas

by coupling together 4 AWR1243 sensors the newer mmWave radars from TI achieve a resolution of 1° [40]. In order to achieve the 0.1° resolution of common lidar systems a total of 1146 virtual antennas are needed in plane as calculated with equation (29). Even at the top of the mmWave band, at antenna spacing of $l = \frac{\lambda}{2} = 3.75mm$ this accounts to a receiver length of 4.3 meters. This is not suitable for a car and demonstrates that radar sensors at mmWave band utilizing only basic MIMO can never have as good an angular resolution using this technology as lidar and camera. The problem of poor angular resolution in radar is still present in 2023 and was mentioned even in some of the publications of the IEEE Radar Conference 2023 [44].

Even with the 30° beam width, the TI system still allows very fine angular movements even at distances up to 50 meters to be distinguished. This is seen by the author to be due to the interpolation done by 0-padding of angle FFT:s. The resulting FFT has more bins for the angle and can thus provide the following DSP stages with enhanced precision. The interpolation is deemed a very successful approach by results obtained in the vehicular tracking chapter. Objects within the 30° beam can be followed throughout their motion in the beam with more than a singular angle detection. This wouldn't be possible without the interpolation.

Having made notes about the resolution of the radar it must be noted that while the resolution is important in order to distinguish two objects close to each other, this task is often fulfilled with the range-Doppler view instead. Thus, ADAS systems do not inherently need a high-resolution radar. Where resolution is important is in mapping and localization. In this context, we want to find features at high resolution and use them in order to better localize the vehicle onto a road surface. As other sensors such as GPS can achieve a localization within 1 meter accuracy the task we set out to achieve with SLAM is to further reduce the ambiguity. This can best be performed with sensors with higher accuracy. The radar resolution is therefore more important for SLAM than ADAS.

The main selling point of radar is its consistency in adverse weather conditions where the other modalities of sensing cannot be utilized reliably [17] [29] [45] [46]. Radar sensing is often employed as a reliability and robustness enhancer due to the lesser correlation of sensor failures occurring in this modality at the same time as other sensors. In comparison to camera and lidar which both fail at similar environmental conditions such as occlusion, rain, fog, and snow the radar fails at different environmental conditions. This gives the system redundancy. In order to make use of this redundancy however an additive approach to sensor fusion must be made instead of a multiplicative one. In additive fusion approaches the sensors independently produce data into an observation dimension. The robustness of a radar can however be compromised

under certain conditions. These conditions are mostly interference related. The problem arises when up to 6 radars are fit on a single vehicle and hundreds of vehicles are placed on a road in close proximity to each other. From the FMCW modulation scheme, we can deduce that an interferer will produce false detections at arbitrary range bins. This can be hard for ADAS systems because a sudden false detection of an incoming collision at 10cm from the vehicle's bumper may prompt the system to employ emergency braking leading to safety concerns. The interference susceptibility of FMCW radar was not tested in this work but its existence leads one to desire other modulation schemes for safety-critical applications working under severe interference conditions.

The radar point cloud generated from the radar cube by CFAR detection and AoA calculation is sparse in comparison to lidar or camera features [17]. Where a lidar will receive a point in the point cloud for each beam sent, radar won't receive detection from every place its beam hits. Instead, the radar point cloud emanates from positions where strong reflections come from. This means that instead of grid-like scan patterns of lidar or camera a radar will produce data in a less dense format. If a radar frame was thought of as a camera frame, most of the pixels would be empty and have no value. This is a good feature to have for communication bandwidth but bad for mapping. Sometimes point grouping algorithms are also run to further lessen the amount of data in the point cloud. This aims at object-level data analysis but loses out on features that could be used in classification tasks. If more points per point cloud are desired one can lower the threshold of the CFAR detector which brings in more points. This will however increase the number of false positives.

As a part of the first milestone, a dense depth map was made out of the range-azimuth heatmap. This was done to demonstrate how the radar cube data format functions. It was noted that it is inaccurate to try and measure depth at every beam angle. In principle, the maximum reflectance value in a beam angle should correspond to an object in the direct line of sight of the radar. When experimented on, the maximum values inside an angle bin almost always corresponded instead to clutter. This led us to the conclusion that looking at the range-azimuth dimension alone is useless. Instead, we need to know what is clutter and remove that before the data has real-world meaning.

Radar detections come with an associated measurement of radial speed. Every point on the point cloud has an individual speed associated with it which is calculated by sending many chirps inside a frame and calculating FFT over the chirps. The speed resolution can be tuned by changing the number of chirps transmitted before the FFT is made over all of them. In contrast to the

camera and lidar, the velocity is therefore already calculated inside the sensor. This is done because the velocity is needed in order to correct a skew in the depth measurement which comes from the Doppler shift. It is also possible to send just a single chirp instead of a frame of chirps. This way the measurement rate of depth is increased. This however means that speed is no longer measured. Depth corrections can also no longer be made which leads to unreliable sensing. Therefore, it is advisable that even in the pursuit of fast measurement rates, enough chirps are put into a frame so that Doppler correction to depth can be made accurately. These are features of the FMCW modulation employed in radar. FMCW modulation can also be employed in Lidar sensing where the modulation correspondingly gives lidar scans an associated speed estimate of the object it is hitting [47].

Radar can see multiple objects within the same beam if they are at different distances. This means that even a large beam with a bad angle resolution keeps its accuracy in the range direction. Our MIMO radar in the dense data format has 256 range bins which makes depth the most sampled dimension of the system. In comparison to the camera and Lidar the width of the radar beam is what allows the radar to see multiple objects inside one beam. This is in effect caused by the larger sampling area of the beam compared to small laser dots of Lidar systems.

The frame rate of a sensor determines its effectiveness in reacting to fast threats on the road. The radar in this work produces data at a maximum fps of 30, which is slow compared to RGBD cameras such as the Intel Realsense depth camera which can offer a performance of 90 fps [24]. For comparison the fastest Lidar systems from PreAct technologies achieve an fps of 125 [48]. When it comes to radar sensing our sensor with an fps of 30 represents the high end of the fps spectrum for automotive radars. The highest fps of a commercial automotive radar is produced by Uhnder and is claimed to be at least 50 fps [49]. From these numbers, we conclude that radar data is produced in a slow fashion. Compared to multi-thousand fps cameras with fast reaction times the radar is better suited to the task of adding robustness.

6. WHAT COULD BE IMPROVED

This section introduces the reader to better alternatives, future developments, and propositions for radar platform development that are however out of the scope of a master's thesis. It shortly explores the proceedings of radar conferences and new commercial products in the field of radar sensing.

6.1 Optimization of chirp parameters

The system was evaluated and fit on the chirp configuration found in `python/radar_configurations/scatter_20fps.cfg` which targeted human perception in an office environment. Later `python/radar_configurations/scatter_20fps_max_radar_crosssection_sensitivity.cfg` was used for automotive measurements which focused on increasing the observation distance. Some exploration of chirp parameters was performed but more complex chirp frames were left out from the analysis. The complex options for chirp frame design allowed for by the SDK could entail programming different phase shifts of chirps inside the frame. With this, an implementation of beamforming or other more complex transmission types could be achieved. Especially beamforming could add region of interest type sensing into the radar system by directing the transmitted energy into the direction of interest.

6.2 Alternative fusion approach

In order to make the system more robust to sensor failure and occlusion, we need to perform the data association task in another way. In the current approach, the radar point cloud is associated with an image object using masks given by a deep learning segmentation network. As the radar data is projected onto the image directly both sensors need a good view of the object in order to make associations. In [15] we find examples of projection that take features from both camera and radar and instead of projecting them on top of each other, the results are projected into a third coordinate space, into which all sensors can independently produce features. In that new measurement space point grouping algorithms can then be run to extract object-level information which is sensor independent. Other approaches to sensor independent fusion than the projection approach used in this work include mapping raw RF images to RGB values and imposing them on top of the camera image, pseudo image generators that generate images

Next, the 3D spaces are to be combined in an independent, third, space. This third space is the fusion space. The best operation for combination is via the addition operand. Addition allows both sensors to produce results independently even if the other sensor produces a 0-output. The fusion space can be implemented as a grid map or a point cloud-based representation such as described previously in the chapter regarding SLAM. In the fusion space, point grouping and object segmentation tasks can be run. These algorithms can produce 3D bounding boxes, the size of which can help in the object classification task.

In this way a more robust detection platform can be formed. In essence, the sensor autonomy this approach provides is due to the additive nature of data combination. Additivity allows sensors to independently produce their output. This way no one sensor can hinder the performance of other sensors. This added robustness would allow the system to be used also in safety critical applications.

6.3 Digital Code Modulation

In its 2022 press release, a company under the name of Uhnder brought to the market a new digital approach to radar modulation. Compared with the analog FMCW modulation used in this work the new PMCW radar shifts most of the complexity of modulation and RF performance into the domain of high-speed data converters and DSP [50]. The press release boasts 16x resolution and 30x better contrast compared to the analog FMCW radar counterparts [51]. The system claims an angular resolution of 1.5° and a range resolution down to 30cm. The implementation is single chip based, contains 192 virtual antennas, and claims to require less space than standard FMCW radar implementations. With the new system, vehicles can be detected over 300 meters away with pedestrian tracking available up to 150m away. [49]

In its whitepaper about the new modulation technique, Uhnder refers to the technique as Digital Code Modulation (DCM). To highlight more the achievements of this new type of modulation Uhnder introduces 2 new figures of merit: High Contrast Resolution (HCR) measures the ability of the radar to distinguish small objects next to larger ones. Need for this merit is found in traffic situations where pedestrians are in traffic next to large cars or cyclists need to be found even when a truck is masking them. Next, the Interference Susceptibility Factor (ISF) characterizes the resilience of the radar to cross- and self-interference. ISF factor is very important as the number of radar vehicles on the road rises [12] [21]. The radars must not cause other radars to lose their function. The Uhnder radar is less susceptible to interference than the previous technologies and has also a better HCR.

PMCW modulation scheme sends out a pseudorandom string of bits that is unique to the transmitter. The information is encoded as a phase shift of either 0 for a 0-bit or 180° for a 1-bit. The receiver calculates the correlation between the sent and the received string to decipher the time-of-flight of the signal. The correlation operation produces very sharp peaks in its response which are easier to interpret than the FFT peaks produced by FMCW. This gives the DCM radar better high contrast resolution compared to FMCW. The PMCW modulation scheme is shown in figure 36. Speed can also be measured by sending out multiple strings one after another. The idea of this is akin to a chirp frame. Doppler shift effect on this modulation is to rotate the phase of the return signal around the I-Q origin. This forces the DCM to have a DAC with higher bit count to correctly perceive the Doppler shift information. [52]

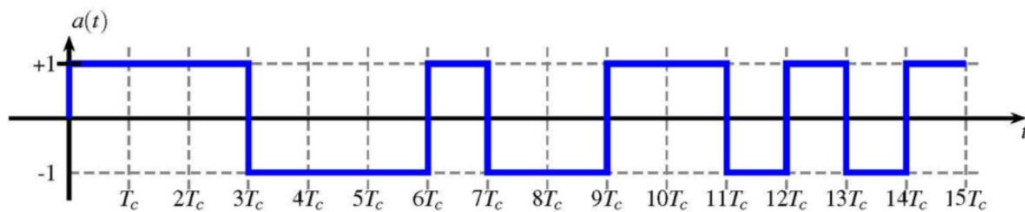


Figure 36 PMCW modulation scheme of Uhnder DCM [52]

The ISF for DCM is higher than its FMCW contenders because the information is represented in a code division multiplex fashion, where each receiver is only listening for a known code and won't be as interfered by other radars not using this particular code. The problem with DCM is that it requires ADC and DSP operations in the order of Ghz as compared to the 10Mhz range in FMCW radars. This digital modulation approach has only been used in military applications until 2020 when Uhnder first published a whitepaper on the subject. [52]

The Uhnder 4D imaging radar was deemed more suitable for the work due to its technical capabilities. It was however determined that it is more important for Kovilta to first get to know radar in its current development phase before moving to these forefront of technology approaches. The novelty of this platform would have also required establishing a communication between Kovilta and Uhnder to get the needed access into resources and documentation as this information was not public knowledge but rather industry secrets. These reasons lead to the FMCW radar being chosen over the superior PMCW radar in this work with a limited scope. For people looking to expand the scope of this work, the Uhnder radar could be of high interest. This is because it has superior interference mitigation, resolution capability, and contains a simpler RF chain. The simpler RF frontend leads to better yield and fewer non-idealities in circuit performance.

6.4 Radar based classification

In this work, the radar data is preprocessed into a point cloud. By moving to this feature-level information, we have discarded rich semantic information present in the raw RF symbols. This information does not relate directly to the speed or depth measure of an object but rather to the qualities of the reflective surface. The common way to mine this deeper semantical information is with the help of machine learning [15]. Many networks have been implemented that can discern extra details about the object such as the type of reflective surface (metal or flesh), size of the object, micro-Doppler signature, and road surface type [53]. The state of the object can also be understood. The output of the system can then be a classification such as “This is a person in fast running motion” or “This is a cyclist at rest”. These labels and the size information can be fed to further tracking and path planning stages which use them in object motion predictions.

An interesting point about semantic information mining from radar data is to look at the ADC bit depth. For example, Uhnder’s PMCW modulation chooses to use 8-bit I and Q channels, which may be more than what is required by the Doppler shift observation task. Overfitting of ADCs would entail that the added data is used to make a richer RF image, which in turn can be fed to a neural network in order to find previously hidden state information about the environment and the objects within. Additional ADC bits could therefore aid in the classification task.

The classification task is implemented widely in present-day radar applications. This means that machine learning has become in large part entangled with the radar environment. Rarely any networks accept point clouds as input and the preferred modalities for radar data representation are different slices of the radar cube. In addition, much semantic information is found by making a short-time Fourier transformation (STFT) on the Doppler data to acquire the micro-Doppler signatures of objects.

The training of the machine learning models is a problem that needs lots of labeled data. To this end [15] proposes that transfer learning, domain adaptation, semi-supervised learning, and life-long learning be applied in the training process of the networks. Transfer learning can be used to transfer the information learned by other networks into a new network. This is often done by reusing the feature extractor parts of previously trained networks that have already found the best features to extract from this sensing modality. [54] Datasets and pre-trained networks for radar-camera fusion are readily available online. The most common dataset is the nuScenes dataset. In April of 2023, the best-performing radar-camera perception system for the nuScenes dataset is CRN [55], which outperforms lidar and camera-based approaches [15].

6.5 Attention

In sensor fusion, the different sensor modalities cannot always be expected to produce the best information content possible. Weather conditions can create data that is unreliable and should not be trusted. The idea of attention is to focus the trust we put in sensors to the modality that produces information in the most reliable way. This is often done in the algorithmic context by implementing an attention matrix that implements weights for trusting the different sensing modalities. [15] Attention is a key driver in the safety of automotive travel as trusting faulty data leads to a garbage in garbage out situation.

The attention approach can also be used to direct sensing into regions of interest in order to reduce the computation of unimportant sections of the environment. In a vehicle context, this is best achieved by radar detecting far-away objects and passing clues to the camera to look into a specific portion of the frame to visually classify that object. In this way, the camera doesn't need to run expensive neural computations on full frames all the time but can instead use its computation resources only on regions of interest. Radar detection is well suited for the region proposal task as the depth resolution and range of the sensor are better than the ones provided by stereo cameras. In the case a sensor becomes occluded, the attention can also be shifted to favor sensors which are not compromised [17].

Attention has become a widespread approach to machine learning since the advent of large language models (LLM) such as ChatGPT. These systems use attention in their building blocks, the transformers. [56] As LLM adoption increases, new hardware accelerators for this task become available. The attention task is therefore of growing interest and its implementation is growing to be cheaper and faster. Applying attention to radar sensing would allow the sensor to increase its effectiveness compared to cameras. This is because radar provides its own illumination. Compared to a camera the radar can direct illumination into parts of the world it deems more important whereas a camera relies on passive illumination. This can be achieved by using beamforming to direct transmission power to the attention hotspots.

6.6 Adaptive waveform

A company called Oculii has developed a hand-in-hand AI-to-adaptive waveform radar that brings the perception capabilities of radar further than all competitors. The company changes the traditional edge computation approach to radar sensing and instead transmits all radar data to a central domain controller. The domain controller acts as the brains of the system. The controller runs an AI algorithm that keeps track of a map. The radar hardware can change its waveform on the fly by command of the controller which is aware of its surroundings. An intelligent design arises from an observation that not all information has to be measured directly from a received radar echo. Instead, by having a map of what we think is where we can derive information also from the part of the radar echo that did not return. The missing data can thus act as data. By adapting the waveform of the radar, we can control the part of signal that is expected to go missing. [57] More information can thus be extracted from the adaptive waveform transmission, while utilizing large parts of the mature FMCW radars hardware implementation. The adaptation of the waveform sets the radar system into feedback mode. Where a normal radar only receives, the Oculii system closes the loop by controlling the transmission too.

The end result of this intelligent waveform control is huge. Oculii claims an enhancement of angular resolution up to 100x. The current flagship model EAGLE boasts an azimuth resolution of 0.5° , 5W power consumption, 350+ m range, 0.16m range resolution. It also completely turns on its head the notion of radar data being sparse by producing 50 000 points per second. These industry-leading specs are however not the most notable feature of the system. It is instead the future scalability.

By using the adaptive waveform together with AI, Oculii achieves orders of magnitude better resolution with fewer antennas than predicted by MIMO antenna configuration equations like (29). The resolution performance also scales with the number of antennas in a more than linear fashion as depicted in figure 37.

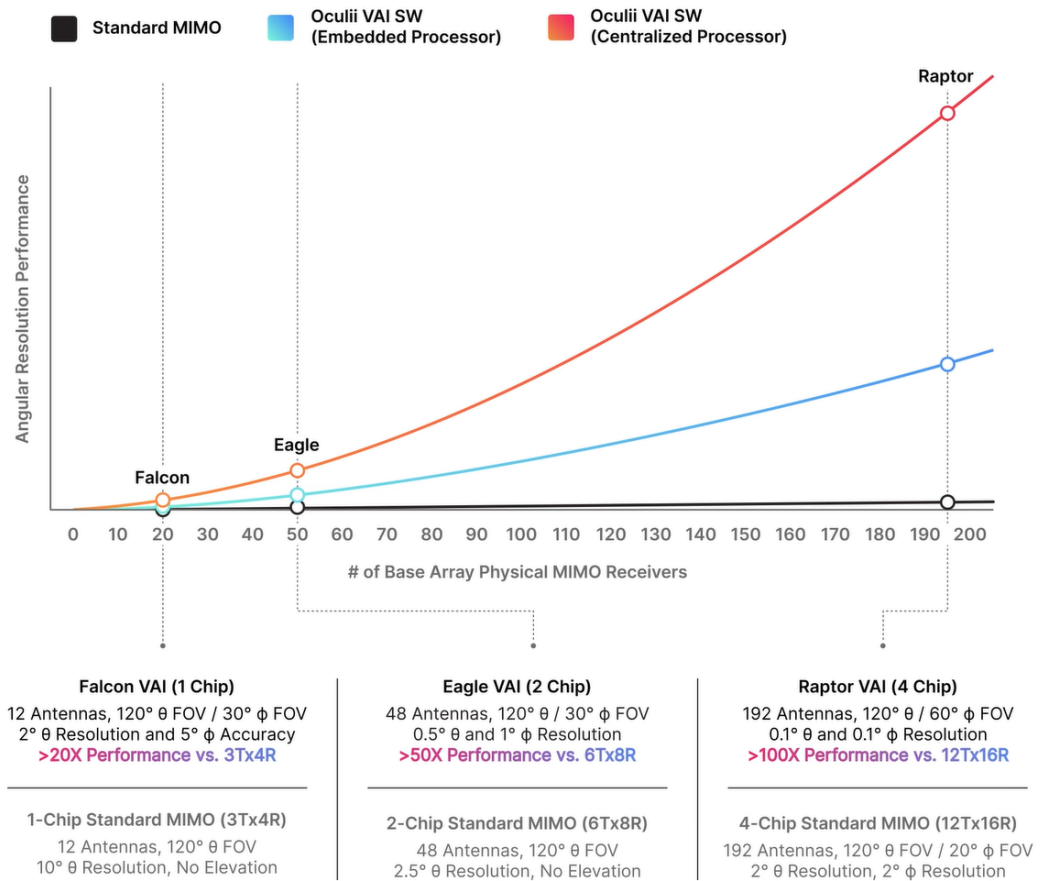


Figure 37 Performance of Oculii systems vs traditional MIMO radar [18]

With the scalability in mind, the Oculii system while being at the top of the resolution performance now will be a clear leader if the predicted performance scaling comes to fruition. Where the Oculii and Uhnder radars have resorted to changing the RF modulation itself, some super-resolution techniques using the old FMCW modulation also exist for reaching similar performance to these systems but without the scalability or interference mitigation provided by the new approaches [12]. The most exciting super-resolution technique for FMCW MIMO was proposed in [58] achieving a resolution of 0.5° using only 12 antennas.

7. CONCLUSIONS

This work explored the basics of radar sensing, sensor fusion, filtering, ADAS, and SLAM. A demo platform was constructed which can track an unlimited number of cars, pedestrians, cyclists, stop signs, and other instances of the COCO dataset. The radar sensing extends to 50 meters and to speeds of up to 56 kmh. The tracking and sensor fusion algorithms used in the platform are sensor agnostic and can scale their performance with better cameras and radars. An innovative way to combine the most accurate parts of a sensing modality into a true velocity estimate was developed. It uses the radial motion components from radar and combines them with axial motion captured from the camera. This combines angular resolution of the camera with the depth resolution of the radar achieving not only 3D perception with classification, but also estimates that are more accurate than those predictable by the sensors separately.

Real world measurements of the system with tuned parameters for camera to radar calibration were performed for both an indoor office environment involving human tracking and for outdoor tracking of vehicles. The outdoor application was more demanding due to the longer propagation length which makes detection harder. The resolution also plays a role in longer range detection as objects further away occupy a smaller incoming angle. From literature, a future challenge for radar was also found. Although not evident in any of the measurements of this work, the interference of multiple radars between each other is set to increase as the number of radar-equipped vehicles on the road keeps increasing. Answers to all of these problems were discussed in the section labelled "What could be improved". In it, new modulation schemes with increased resolution, beam forming capability, code division multiplexing, interference mitigation, and better scalability with antenna area are discussed.

The use of radar in SLAM and ADAS systems is often as a redundancy and path planning aid. The redundancy arises from radar seeing through occlusion such as rain, snow, and fog. Other sensing modalities, such as camera and lidar, are very susceptible to these environmental conditions and tend to both fail at the same time. Radar avoids this by using light with a lower frequency which does not scatter as easily from occlusion. This allows the radar to see through the snow cover. The path planning and safety applications arise from the radars ability to detect targets from further ranges than the camera. At 350+ meters of range capability, the newest radars can give the system time to react to changes in the environment not afforded by other modes of sensing. More reaction time translates to better chance of accident avoidance.

The information presented in this master's thesis is made fully available to the public by releasing the project as open source on GitHub under the MIT license. The files can be found at [37]. Kovilta Oy and the author have agreed to this under the pretense that no official support is to be offered and that the author is representative of himself only and not to be affiliated with the company he is in relations with.

8. REFERENCES

- [1] C. Iovescu, S. Rao, "The fundamentals of millimeter wave radar sensors (Rev. A)", Texas instruments, July 2020. [Online]. Available: https://www.ti.com/lit/wp/spyy005a/spyy005a.pdf?ts=1657531632660&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FAWR2243 [Accessed: Dec. 30, 2023].
- [2] M. Chowdhury, L. Deka, "Transportation Cyber-Physical Systems", Elsevier, 2018.
- [3] C. Wolff, "Radar basics - Radial Speed", radartutorial.eu. [Online]. Available: <https://www.radartutorial.eu/11.coherent/Radial%20Speed.en.html> [Accessed: Dec. 30, 2023].
- [4] CAS dataloggers, "THE BASICS OF SIGNAL ATTENUATION". [Online]. Available: https://www.dataloggerinc.com/wp-content/uploads/2016/11/16_Basics_of_signal_attenuation.pdf [Accessed: Dec. 30, 2023].
- [5] M. Parker, "Digital Signal Processing 101: Everything You Need to Know to Get Started. 2nd edition", Oxford: Elsevier Science, 2017.
- [6] C. Powell, "Technical Analysis: Beamforming vs. MIMO Antennas", Radio Frequency systems, March 2014. [Online]. Available: https://www.rfsworld.com/userfiles/white_papers/2014/Ooredoo_White%20Paper_Mar14.pdf [Accessed: Dec. 30, 2023].
- [7] Ó. García, "Signal Processing for mmWave MIMO Radar", Master's thesis, University of Gävle, June 2015. [Online]. Available: <http://www.diva-portal.se/smash/get/diva2:826028/FULLTEXT01.pdf> [Accessed: Jan. 3, 2023].
- [8] NXP Semiconductors, "Automotive 4D Imaging Radar Demo from NXP Semiconductors". [Online]. Available: https://www.youtube.com/watch?v=Al-OGgV4mV0&ab_channel=NXPSemiconductors [Accessed: Dec. 30, 2023].
- [9] D. Joyce, "Summary of trigonometric identities", Clark university. [Online]. Available: <https://www2.clarku.edu/faculty/djoyce/trig/identities.html> [Accessed: Dec. 30, 2023].
- [10] M. DeCross et al., "Small-Angle Approximation", Brilliant. [Online]. Available: <https://brilliant.org/wiki/small-angle-approximation/> [Accessed: Dec. 30, 2023].
- [11] M. Vazquez, "Radar Resolution: How Accurate Can a Radar Be?", Renesas, March 2022. [Online]. Available: <https://www.renesas.com/us/en/blogs/radar-resolution-how-accurate-can-radar-be> [Accessed: Dec. 30, 2023].
- [12] J. Fuchs et al., "A Machine Learning Perspective on Automotive Radar Direction of Arrival Estimation", IEEE Access, vol. 10, pp. 6775-6797, Jan 2022, doi: 10.1109/ACCESS.2022.3141587.
- [13] C. Katzlberger, "Object Detection with Automotive Radar Sensors using CFAR-algorithms", Johannes Kepler University Linz, Sep 2018. [Online]. Available:

- https://www.jku.at/fileadmin/gruppen/183/Docs/Finished_Theses/Bachelor_Thesis_Katzlberger_final.pdf [Accessed: Dec. 30, 2023].
- [14] Z. Han et al., “4D Millimeter-Wave Radar in Autonomous Driving: A Survey”, arXiv, Jun 2023. [Online]. Available: <https://arxiv.org/abs/2306.04242> [Accessed: Dec. 30, 2023].
- [15] S. Yao et al., “Radar-Camera Fusion for Object Detection and Semantic Segmentation in Autonomous Driving: A Comprehensive Review”, IEEE Transactions on Intelligent Vehicles, pp. 1-40, Jan 2023, doi: 10.1109/tiv.2023.3307157.
- [16] Y. Wang et al., “RODNet: A Real-Time Radar Object Detection Network Cross-Supervised by Camera-Radar Fused Object 3D Localization”, IEEE Journal of Selected Topics in Signal Processing, vol. 15, no. 4, pp. 954-967, June 2021, doi: 10.1109/JSTSP.2021.3058895.
- [17] K. Youngseok, C. Jun Won, K. Dongsuk, “GRIF Net: Gated Region of Interest Fusion Network for Robust 3D Object Detection from Radar Point Cloud and Monocular Image”, 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), doi:10.1109/IROS45743.2020.9341177.
- [18] Oculii / Ambarella, [Online]. Available: <https://www.oculii.com> [Accessed: Dec. 30, 2023].
- [19] J. Huang, “The Next AI Moment is Here”, NVIDIA, presented at COMPUTEX 2023, Taipei, Taiwan, May 30 - June 2, 2023. Available: <https://www.nvidia.com/en-us/events/computex/> [Accessed: Dec. 30, 2023].
- [20] ibaiGorordo, “AWR1843-Read-Data-Python-MMWAVE-SDK-3-“, GitHub. [Online]. Available: <https://github.com/ibaiGorordo/AWR1843-Read-Data-Python-MMWAVE-SDK-3-> [Accessed: Dec. 30, 2023].
- [21] Texas Instruments, “mmWave radar sensors”, video lectures, 2017. [Online]. Available: <https://www.ti.com/video/series/mmwave-training-series.html> [Accessed: Jan. 3, 2023].
- [22] B. Vogginger et al., “Automotive Radar Processing With Spiking Neural Networks: Concepts and Challenges”, Frontiers in Neuroscience, Vol. 16, April 2022. Available: <https://doi.org/10.3389/fnins.2022.851774> [Accessed: Jan. 3, 2023].
- [23] A. Shashua, “Mobileye Under The hood”, Mobileye, Presented at CES 2022, Las Vegas, Nevada, Jan 5 – 7, 2022. Available: <https://www.youtube.com/watch?v=QV7PGBfI49k> [Accessed: Dec. 30, 2023].
- [24] Intel, “Intel Realsense Depth Camera D457”. [Online]. Available: <https://www.intelrealsense.com/depth-camera-d457/> [Accessed: Dec. 30, 2023].
- [25] Kovilta, “Technology. Sensor-level Processing”. [Online]. Available: <https://kovilta.fi/technology/> [Accessed: Dec. 30, 2023].
- [26] R. Labbe, “Kalman and Bayesian Filters in Python”, GitHub, Oct 2020. [Online]. Available: <https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python> [Accessed: Dec. 30, 2023].

- [27] MathWorks, “Developing Autonomous Mobile Robots Using MATLAB and Simulink”. [Online]. Available: <https://se.mathworks.com/campaigns/offers/next/autonomous-mobile-robots.html> [Accessed: Dec. 30, 2023].
- [28] MatWorks, “What Is SLAM? 3 things you need to know”. [Online]. Available: <https://se.mathworks.com/discovery/slam.html> [Accessed: Dec. 30, 2023].
- [29] Z. Hong, Y. Petillot, A. Wallace, S. Wang, “Radar SLAM: A Robust SLAM System for All Weather Conditions”, Edinburgh Centre for Robotics, arXiv, 2021. [Online]. Available: <https://arxiv.org/pdf/2104.05347.pdf> [Accessed: Dec. 30, 2023].
- [30] D. Ball et al., “OpenRatSLAM: an open source brain-based SLAM system”, *Auton Robot* vol. 34, pp. 149–176, 2013. Available <https://doi.org/10.1007/s10514-012-9317-9> [Accessed: Dec. 30, 2023].
- [31] Matlab, “Autonomous Navigation”, Aug 2020. [Online]. Available: <https://www.youtube.com/playlist?list=PLn8PRpmsu08rLRGrnF-S6TyGrmcA2X7kg> [Accessed: Dec. 30, 2023].
- [32] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, “FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem”, *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002. Available: <http://robots.stanford.edu/papers/montemerlo.fastslam-tr.pdf> [Accessed: Dec. 30, 2023].
- [33] Z. Shuran et al., “Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis”, *Remote Sensing*, Vol. 15, no. 4, pp. 1156. Available: <https://doi.org/10.3390/rs15041156> [Accessed: Dec. 30, 2023].
- [34] Texas Instruments, “mmWave software development kit (SDK)”. [Online]. Available: <https://www.ti.com/tool/MMWAVE-SDK> [Accessed: Dec. 30, 2023].
- [35] Texas Instruments, “AWR1843AOP Single-chip 77- and 79-GHz FMCW mmWave Sensor Antennas-On-Package (AOP)”. [Online]. Available: <https://www.ti.com/lit/ds/sym-link/awr1843aop.pdf> [Accessed: Dec. 30, 2023].
- [36] Texas Instruments, “Millimeter Wave (mmw) Demo for XWR18XX”. [Online]. Available: https://dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/3.6.0/ [Accessed: Dec. 30, 2023].
- [37] K. Paasio, “AWR1830AOPEVM_camera_sensor_fusion”, GitHub, Jan 2024. [Online]. Available: https://github.com/kulle777/AWR1830AOPEVM_camera_sensor_fusion [Accessed: Jan. 2, 2023].
- [38] P. Kumar, “What are RGBD cameras? Why RGBD cameras are preferred in some embedded vision applications?”, *Technology deep dive*, May 2022. [Online]. Available: <https://www.e-consystems.com/blog/camera/technology/what-are-rgbd-cameras-why-rgbd-cameras-are-preferred-in-some-embedded-vision-applications/> [Accessed: Dec. 30, 2023].
- [39] A. Astro and C. Avizzano, “fmcw-RADAR”, Sant'Anna Scuola Universitaria Superiore Pisa, GitHub, 2021. [Online]. Available: <https://github.com/Oxastro/fmcw-RADAR> [Accessed: Dec. 30, 2023].

- [40] C. Händel, H. Konttaniemi, M. Autioniemi, "State-of-the-Art Review on Automotive Radars and Passive Radar Reflectors", Lapin Ammattikorkeakoulu, May 2018. Available: <https://urn.fi/URN:ISBN:978-952-316-223-5> [Accessed: Dec. 30, 2023].
- [41] D. Sawicki, "Cosine Effect Error on Radar and Lidar", Police Radar Information Center, 2016. [Online]. Available: <https://copradar.com/chaps/chapt2/ch2d1.html> [Accessed: Dec. 30, 2023].
- [42] Y. Zhou, Y. Dong, F. Hou, J. Wu, "Review on Millimeter-Wave Radar and Camera Fusion Technology", Sustainability, vol. 14, no. 9, pp. 5114, April 2022. Available: <https://doi.org/10.3390/su14095114> [Accessed: Dec. 30, 2023].
- [43] S. Ipek, R. Kapusta, "LIDAR for Autonomous System Design: Object Classification or Object Detection?", Analog Devices, Oct 2020. [Online]. Available: <https://www.analog.com/en/analog-dialogue/articles/lidar-for-autonomous-system-design-object-classification-or-object-detection.html> [Accessed: Dec. 30, 2023].
- [44] V. Janoudi et al., "Antenna Array Design for Coherent MIMO Radar Networks", presented at 2023 IEEE Radar Conference, San Antonio, TX, USA, May 1 - 5, 2023. Available: <https://ieeexplore.ieee.org/document/10149789> [Accessed: Dec. 30, 2023].
- [45] K. Marenko, "Why Hi-Resolution Radar is a Game Changer", Fierce Electronics, Aug 2018. [Online]. Available: <https://www.fierceelectronics.com/components/why-hi-resolution-radar-a-game-changer> [Accessed: Dec. 30, 2023].
- [46] M. Vazquez, "Radar For Automotive: Why Do We Need Radar?", Semiengineering, March 2022. [Online]. Available: <https://semiengineering.com/radar-for-automotive-why-do-we-need-radar/> [Accessed: Dec. 30, 2023].
- [47] D. Coldewey, "Aeva and NASA want to map the moon with lidar-powered KNaCK pack", Techcrunch, April 2022. [Online]. Available: <https://techcrunch.com/2022/04/21/aeva-and-nasa-want-to-map-the-moon-with-lidar-powered-knack-pack/?guccounter=2> [Accessed: Dec. 30, 2023].
- [48] PreAct technologies, "Mojave™ Sensor". [Online]. Available: <https://www.preact-tech.com/mojave-sensor> [Accessed: Dec. 30, 2023].
- [49] Magna International, "Magna ICON Digital Radar", Aug 2021. [Online]. Available: <https://www.youtube.com/watch?v=aFkZefmtruE&t=228s> [Accessed: Dec. 30, 2023].
- [50] X. Gao et al., "Experiments with mmWave Automotive Radar Test-bed", 53rd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 2019, pp. 1-6, doi: 10.1109/IEEECONF44664.2019.9048939.
- [51] E. Walz, "The First 4D Imaging Radar-on-a-Chip Developed by Texas-based Startup 'Uhnder' Will Debut on Vehicles in 2022", Future Car. [Online]. Available: <https://www.futurecar.com/5253/The-First-4D-Imaging-Radar-on-a-Chip-Developed-by-Texas-based-Startup-Uhnder-Will-Debut-on-Vehicles-in-2022> [Accessed: Aug 7, 2023].
- [52] W. Stark, M. Ali, M. Maher, "DIGITAL CODE MODULATION (DCM) RADAR FOR AUTOMOTIVE APPLICATION", Uhnder, Jan 2020. [Online]. Available:

- https://www.uhnder.com/images/data/DCM_Radar_for_Automotive_Application_Final.pdf [Accessed: Dec. 30, 2023].
- [53] S. Sabery et al., “Road Surface Classification Based on Radar Imaging Using Convolutional Neural Network”, *IEEE Sensors Journal*, vol. 21, no. 17, pp. 18725 - 18732, June 2021, doi: 10.1109/JSEN.2021.3087336.
- [54] P. Baheti, “A Newbie-Friendly Guide to Transfer Learning”, Microsoft, Oct 2021. [Online]. Available: <https://www.v7labs.com/blog/transfer-learning-guide> [Accessed: Dec. 30, 2023].
- [55] Y. Kim et al., “CRN: Camera Radar Net for Accurate, Robust, Efficient 3D Perception”, presented at ICCV 2023, Paris, France, Oct 2 – 6, 2023. Available: <https://arxiv.org/abs/2304.00670> [Accessed: Dec. 30, 2023].
- [56] Kjung, “Replacing Recurrence with Attention: Improving Large Language Models with Self-Attention”, *Edge Analytics*, Oct 2021. [Online]. Available: <https://medium.com/edge-analytics/replacing-recurrence-with-attention-improving-large-language-models-with-self-attention-6baa95b5e649> [Accessed: Dec. 30, 2023].
- [57] S. Ward-Foxton, “AV Radar Moves to Domain Controller for First Time”, *EE-times*, Jan 2023. [Online]. Available: <https://www.eetimes.com/av-radar-moves-to-domain-controller-for-first-time/> [Accessed: Dec. 30, 2023].
- [58] R. Sun et al., “A millimeter-wave automotive radar with high angular resolution for identification of closely spaced on-road obstacles”, *Nature, Scientific Reports*, vol. 13, no. 3233, Feb 2023. Available: <https://www.nature.com/articles/s41598-023-30406-4> [Accessed: Dec. 30, 2023].
- [59] C. Wolff, “Frequency-Modulated Continuous-Wave Radar (FMCW Radar)”, *radartutorial.eu*, [Online]. Available: <https://www.radartutorial.eu/02.basics/Frequency%20Modulated%20Continuous%20Wave%20Radar.en.html> [Accessed: Dec. 30, 2023].
- [60] C. Liechti, “pySerial”, 2020. [Online]. Available: <https://pyserial.readthedocs.io/en/latest/pyserial.html>. [Accessed: Dec. 30, 2023].