Machine Learning for FMCW Radar Interference Mitigation

Ludvig Hassbring, Jessica Nilsson
tfy15lha@student.lu.se, tfy15jni@student.lu.se

6 January 2020 to 29 May 2020

LUNDS UNIVERSITET

Department Name: Division of Mathematical Statistics
University Name: Lund University
Country: Sweden
Supervisor: Andreas Jakobsson

Abstract

Frequency Modulated Continuous Wave radar is used for object detection and localization, e.g., for surveillance purposes or for automotive systems. The amount of radars is increasing, which leads to an increasing amount of radar interference. The radar signals from different devices are uncoordinated and therefore difficult to estimate beforehand. If several radars operate on the same frequency band, mutual disruptions will occur. Interference can lead to false detections, i.e., ghost objects, or missed detections. If identical radars are interfering with each other, a specific type of interference, so called coherent interference, is present. This type of interference together with a phenomena called clock drift is simulated in this project, as well as semi-coherent and non-coherent interference. Several mitigation algorithms are used to reduce/eliminate the different types of interference.

Convolutional Neural Networks (CNNs) are commonly used to find structure and patterns in data. By providing the network with clean data and data containing interference, the network can be trained to detect where interference is occurring and re-create the clean data set. Since the data containing interference has different attributes than clean data, interference can be detected and mitigation can be performed to reduce the interference.

Two CNN architectures, one shallow and one deep, are trained and evaluated using simulated data and the performance of the models are compared with conventional signal processing algorithms. The Signal-to-Interference-plus-Noise Ratio (SINR) and Error Vector Magnitude (EVM) of the different algorithms are compared. Training the CNNs to minimize SINR generates models useful for object detection, even when subject to a substantial amount of interference. By using Mean Square Error (MSE) as the objective function, the trained models are useful for interference mitigation, but ghost objects are occasionally classified as true objects. Generally, CNNs can be used as an alternative to common signal processing algorithms, especially when a majority of the data points are affected by interference. The complex networks are able to identify the data containing information about the objects, even when the data is hidden in a considerable amount of interference.

Keywords: FMCW radar, interference mitigation, convolutional neural network

Acknowledgements

We would like to extend our gratitude to our supervisors, Dr. Stefan Adalbjörnsson and Dr. Anders Mannesson for their guidance and support throughout this thesis. We would also like to thank our supervisor at the university, Prof. Andreas Jakobsson, at the Division of Mathematical Statistics, Lund University, for his much appreciated help.

Contents

Acronyms

AoA *Angle of Arrival*. 1

BS *Batch Size*. 28

CNN *Convolutional Neural Network*. 27

CR *Cross Range*. 13

DFT *Discrete Fourier Transform*. 7

FFT *Fast Fourier Transform*. 7

FMCW *Frequency Modulated Continuous Wave*. 2

IF *Intermediate Frequency*. 3

RA *Range-Angle*. 16

RD *Range-Doppler*. 15

RDA *Range-Doppler-Angle*. 17

ROC *Receiver Operating Characteristic*. 43

SINR *Signal-to-Interference-and-Noise Ratio*. 20

SNR *Signal-to-Noise Ratio*. 20

Glossary

B Bandwidth of a chirp. 4

D Distance between an object and the radar. 6

M Number of chirps in a frame. 10

T Duration of a chirp, also called sweep time. 5

α Slope of a chirp. 5

λ Wavelength of the signal. 7

ω_a Phase difference between consecutive antennas. 13

ω_v Phase difference between consecutive chirps. 12

τ Time delay between the transmitted and the received signal. 4

θ Angle of arrival. 7

c Speed of light. 6

d Distance between two neighbouring antennas. 7

f_B Intermediate frequency. 4

f_c Starting frequency of a chirp. 4

k Current antenna that is regarded. 7

m Current chirp that is regarded. 6

t_s Current time point within a chirp. 6

v Radial velocity of a detected object. 6

Chapter 1

Introduction

1.1 Research Problem

Radar is primarily used for object detection and localization. The radar system can be used for measuring distance, radial velocity, and *Angle of Arrival* (AoA) for one or several objects within a certain area. High-resolution detection can be obtained even in poor weather conditions such as fog or poor lightning, which is one reason for using radars [Aydogdu et al., 2019a]. Radar is used when some kind of object localization is needed and the applications include surveillance, weather forecasting and driver assistance systems [Chang et al., 2019], [Rock et al., 2019b], [Yeary et al., 2011].

The automotive industry is using radars in cars to localize objects such as humans, other cars or objects besides the road. The amount of radars within a car and the amount of cars with radars is increasing [Aydogdu et al., 2019b]. The demand for high performance surveillance and reconnaissance has also increased, leading to an increased amount of radars within this area as well [Chang et al., 2019]. To monitor a large area, one radar may not be sufficient. Instead, a network of several radars with overlapping surveillance areas can be employed, leading to more radars being in use. Even though the network increases the size of the monitored area, the radars might interfere with each other, causing disrupted signals [Nikolió et al., 2016]. To handle the increasing amount of radars, it is crucial to detect when disturbances are occurring and to know how they can be eliminated.

The disturbance includes interference by other radars or units which operate at the same frequency band as the transmitted radar signal. The different radar systems are not coordinated and, as they have similar properties, it is difficult to distinguish the desired signals from the disturbance [Aydogdu et al., 2019b][Rock et al., 2019b]. According to [Aydogdu et al., 2019a], interference can for example generate false detections or increase the noise floor which leads to an increase in missed detections. In this report, the goal is primarily to detect interference from other radars, with various characteristics, and remove it without disrupting the signals that carry correct information about the objects.

Disturbances must primarily be found in order to perform any type of mitigation. However, the type of interference is not known in advance, since it can originate

from different sources. Therefore, it can be difficult to find the correct algorithm to remove the different types of interference and at the same time maintain the original desired signal. *Machine Learning* (ML), which is an application of *Artificial Intelligence* (AI), is the scientific study of algorithms and statistical models that can be used to train a computer to perform desired tasks. In this report, it is desired to examine if an AI-based algorithm can be trained to solve the interference problem better than common signal processing algorithms.

1.2 Aim and Scope

The aim of this project is primarily to detect where in a radar signal interference is occurring, and to remove as much of the interference as possible without disrupting the desired signal. The interference can be divided into categories depending on how similar the interfering and the interfered signals are. Handling similar, also called coherent, interference together with a phenomena called *clock drift* is relevant when identical radars are interfering with each other. Simulating and mitigating this specific type of interference is a central part of this project.

To achieve this, a convolutional neural network is investigated and compared to several signal processing algorithms in order to determine if the network can outperform the other algorithms when mitigating interference. The type of radar investigated in this project is a *Frequency Modulated Continuous Wave* (FMCW) radar. It is assumed that the interference originates from other radars, which operates at the same frequency band as the interfered radar.

1.3 Outline of the Thesis

Initially, the thesis contains background information about FMCW radars and the signal model. Then, the theory behind object detection and interference is explained. A convolutional neural network is then presented along with previous research in this area. Other mitigation algorithms are also presented. Thereafter, a radar simulation is explained, which is used to generate training, validation and testing data. The simulated data is used to train several ML algorithms and their performances are compared to each other, to conventional signal processing algorithms and to non-interfered data. The results are then discussed to determine if any of the ML-methods is desirable to use.

Chapter 2

Theory

2.1 FMCW Radar Signal Model

A Frequency Modulated Continuous Wave can be used as a transmission signal in radars. The low weight and the low power consumption of this system is advantageous, as well as the low price compared to other radars [Kim et al., 2018]. Another advantage is the simplicity of the system which enables a smooth implementation in the hardware. Figure 2.1 shows the basic principles of an FMCW radar. Here, one

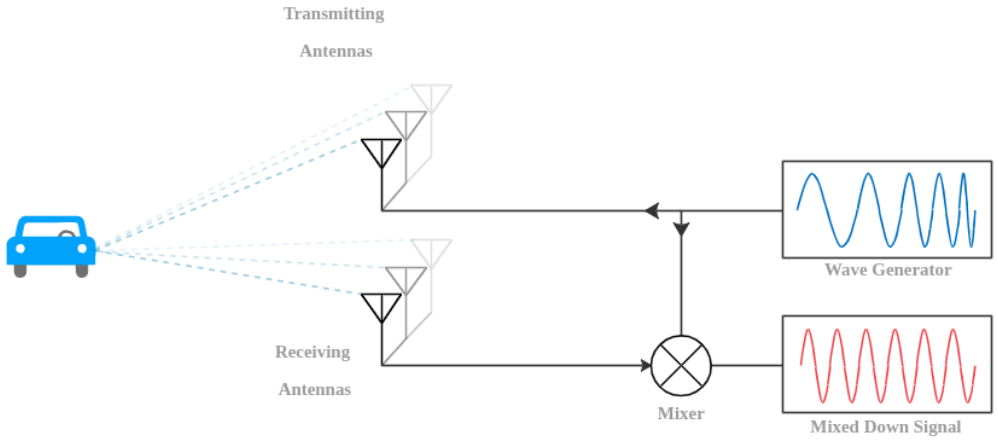


Figure 2.1: Schematic diagram of how an FMCW radar operates.

sinusoidal wave with a linearly increasing frequency is generated by a synthesizer and transmitted using one or several antennas. This kind of pulse is commonly denoted as a *chirp*. The same chirp that was transmitted is also sent to a mixer, an analog component which multiplies two inputs, for further processing. Afterwards, the mixed signal, also denoted *baseband signal* or *Intermediate Frequency* (IF) signal, is low-pass filtered and then digitized by an analog-to-digital converter. The filtering is performed to avoid aliasing, which could occur if the analog signal contains higher frequencies than the analog-to-digital converter can register due to its sampling frequency. Subsequently, the signal is further processed.

If an object is present within the detection range of the radar, the transmitted chirp is reflected by the object and registered by one or more receiver antennas on

the radar. The signal that each antenna is measuring is denoted a channel. The received signal, also called rx-signal, has the same appearance as the transmitted signal, also called tx-signal, except for a time delay τ which depends on how far from the radar the object is located [Sandeep Rao, 2016].

Figure 2.2 depicts the linearly increasing frequency over time of the transmitted and the received signal. In this figure, f_c denotes the starting frequency of the sweep, τ is the time delay between the signals, f_B is the absolute frequency difference between the signals and B is the bandwidth of the chirp. A chirp is usually followed by an idle time, when the radar is inactive before chirping up again. The figure illustrates three consecutive chirps. A series of multiple chirps forms a *frame* where the size of one frame is equal to the number of chirps times the number of samples in a chirp. By only using one chirp, the distance to an object can be measured. By extending the system to multiple chirps, the radial velocity of the object can also be measured. A further extension to multiple receiving antennas enables estimation of the AoA. By knowing the angle, the exact location of the object can be specified, instead of just the radial distance between the object and the radar. A summary of the different setups and what they can measure is presented below.

- One chirp and one antenna ⟶ Distance
- Multiple chirps and one antenna ⟶ Distance and radial velocity
- Multiple chirps and multiple antennas ⟶ Distance, radial velocity and AoA

The time delay τ, which depends on the distance to the object, gives rise to a constant frequency difference between the signals. Obtaining a constant frequency difference depending on the distance to the object is essential for object detection.

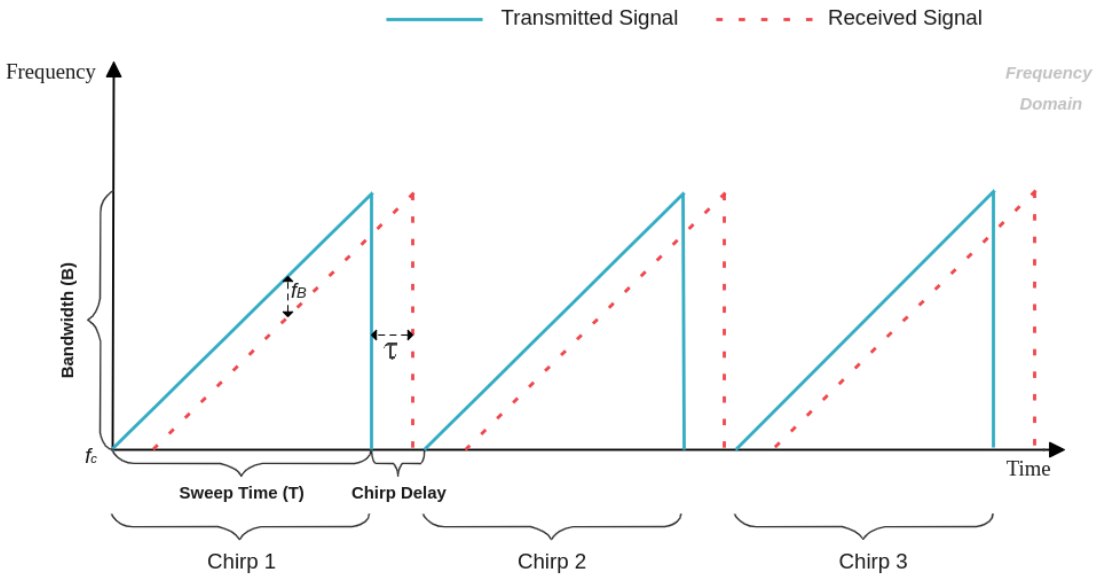


Figure 2.2: The linearly increasing frequency of the transmitted and the received waveform for a frame containing three chirps. A detected object gives rise to the received signal.

The calculation of the absolute frequency difference between the transmitted and the received signal is performed in the mixer. As stated previously, the function of the

mixer is to perform a multiplication of incoming signals. The mixing of a transmitted and a received signal can be visualized in Figure 2.3. The time difference τ in the received signal gives rise to an baseband signal, with a constant frequency. The

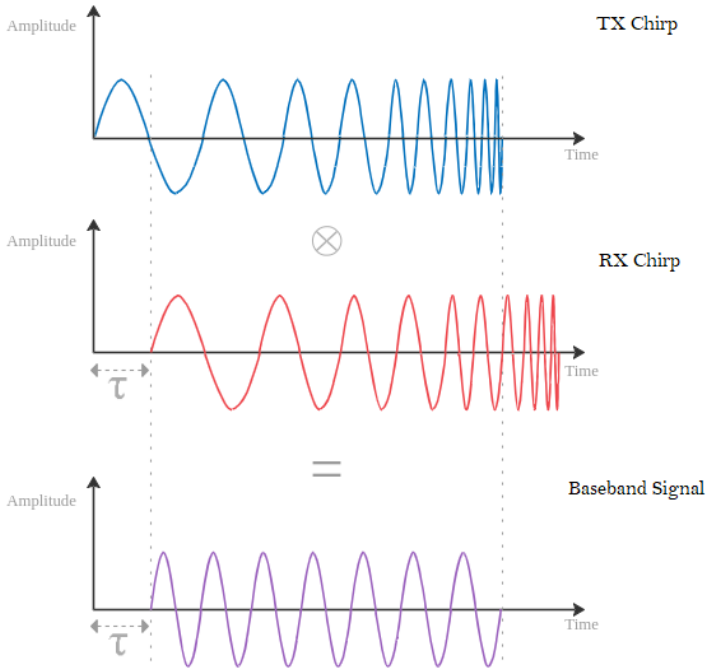


Figure 2.3: A visualization of how the transmitted and the received signal are mixed.

mixing can mathematically be expressed as

$$\cos(x)\cos(y) = (\cos(x+y) + \cos(x-y))/2, \tag{2.1}$$

which states that a multiplication of two sinusoids creates one term where the frequencies are added and one term where the frequencies are subtracted. In (2.1), $\cos(x+y)$ is filtered out by a low-pass filter connected to the analog-to-digital converter, due to its high frequency content. The frequency of the baseband signal, which is also called the beat frequency, will only consist of the absolute frequency difference between the transmitted and the received signal, $(x-y)$ in (2.1) or f_B in Figure 2.2. If multiple objects are present at different distances from the radar, the baseband signal will consist of several sinusoids, each with a unique, constant frequency [Sandeep Rao, 2016].

The derivation of the signal model, which is inspired by [Suleymanov, 2016], is presented below. A more in-depth derivation can be read in [Suleymanov, 2016]. According to Figure 2.2, the frequency of the transmitted signal can for one chirp be described by a linearly increasing function over time

$$f(t) = f_c + \frac{B}{T}t = f_c + \alpha t, \tag{2.2}$$

where T is the duration of one chirp, B is the bandwidth and $\alpha = \frac{B}{T}$ denotes the slope of the chirp. To extend this to multiple chirps, the time variable t can be substituted to

$$t = mT + t_s, \tag{2.3}$$

where $0 < t_s < T$. Here, m denotes the m:th chirp and t_s is the current time within a chirp. t is then the current time within a frame.

The phase of the transmitted signal is used to mathematically describe the waveform. The phase is inserted in the cos-expression to generate the wave. For one chirp, the phase can be expressed by regarding the frequency change over time according to

$$\phi(t_s) = 2\pi \int_0^{t_s} f(t)dt = 2\pi \int_0^{t_s} (f_c + \alpha t)dt = 2\pi \left(f_c t_s + \frac{\alpha t_s^2}{2} \right) + \phi_0, \tag{2.4}$$

where ϕ_0 is the initial phase of the wave. The normalized transmitted signal for one chirp can then be described by

$$x_t(t_s) = \cos(\phi(t_s)) = \cos\left(2\pi \left(f_c t_s + \frac{\alpha t_s^2}{2} \right) \right). \tag{2.5}$$

When one receiving antenna is used, the only difference between the transmitted and the received signal is a time delay τ, if only one object is present. The delay arises because it takes time for the signal to travel between the radar and the object. If the object is moving, the velocity will contribute to the delay. The total delay is

$$\tau(t_s) = \frac{2(D + vt)}{c} = \frac{2(D + v(mT + t_s))}{c}, \tag{2.6}$$

where D is the distance to the object, v is the radial velocity of the object and c is the speed of light. The factor 2 occurs due to the signal travelling back and forth from the radar. The received signal is therefore mathematically constructed according to

$$x_r(t_s) = \cos(\phi(t - \tau)) = \cos\left(2\pi \left(f_c(t_s - \tau) + \frac{\alpha(t_s - \tau)^2}{2} \right) \right). \tag{2.7}$$

As stated previously, the frequency of the baseband signal is solely the absolute difference between the transmitted and the received signal. The transmitted signal is mixed with the received signal in the mixer, which means that a multiplication of the two signals is performed and creates the baseband signal

$$\begin{aligned} x_m(t_s) &= x_t(t_s) x_r(t_s) \\ &\propto \cos\left[2\pi \left(f_c t_s + \frac{\alpha t_s^2}{2} \right) - 2\pi \left(f_c(t_s - \tau) + \frac{\alpha(t_s - \tau)^2}{2} \right) \right] \\ &\propto \cos\left[2\pi \left(f_c \tau + \alpha t_s \tau - \frac{\alpha \tau^2}{2} \right) \right]. \end{aligned} \tag{2.8}$$

By inserting the expression for τ from (2.6) into (2.8) and by making the simplifications performed in [Suleymanov, 2016] to remove negligible terms, an approximation of the baseband signal for one receiving antenna is

$$x_m(t_s, m) = \cos\left(2\pi \left(\frac{2\alpha D}{c} t_s + \frac{2 f_c v m}{c} T \right) + \frac{4\pi f_c D}{c} \right). \tag{2.9}$$

Equation (2.9) is valid if only one receiving antenna is present. For multiple antennas, a term depending on the antennas is included and the baseband signal is approximated as

$$x_m(t_s, m) = \cos\left(2\pi \left(\frac{2\alpha D}{c} t_s + \frac{2 f_c v m}{c} T + \frac{dk \sin(\theta)}{\lambda} \right) + \frac{4\pi f_c D}{c} \right) \tag{2.10}$$

according to [Suleymanov, 2016]. Here, θ is the AoA of the localized object, λ is the wavelength of the signal, d is the distance between two neighbouring receiving antennas and k is the antenna that is regarded. If only one antenna is used, k is set to 0 and (2.10) becomes (2.9). The AoA-estimation is explained in more detail in section 2.2.4.

2.2 Signal Processing for Object Detection

2.2.1 Basics About Fourier Transforms

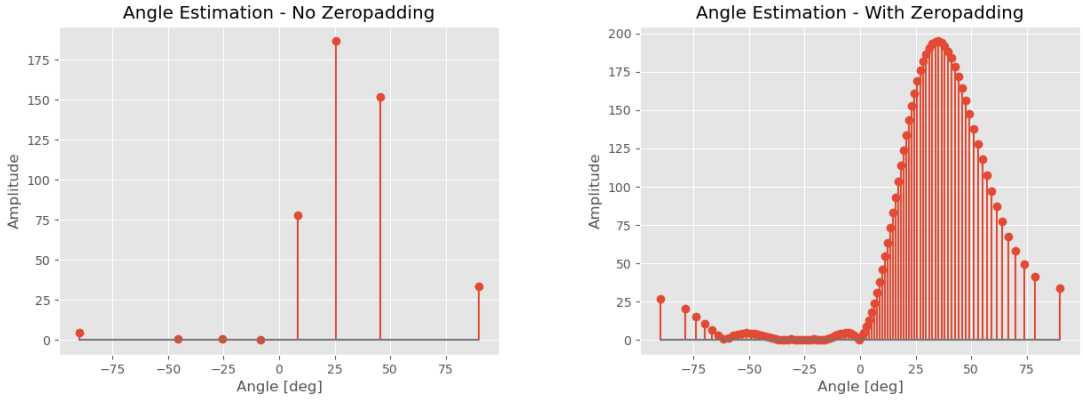
When processing the baseband signal, its frequency content is estimated using a *Discrete Fourier Transform* (DFT). DFTs have a limit to their resolution, determined by the amount of samples inserted into it. In the mathematical definition of the DFT,

$$X_j = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}jn} = \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N}jn\right) - i\sin\left(\frac{2\pi}{N}jn\right) \right], \tag{2.11}$$

N is the number of samples for the data inserted in the DFT. The output $\{X_j\} = X_0, ..., X_{N-1}$ has the same size as the input $\{x_n\} = x_0, ..., x_{N-1}$. When using the expression on radar signals, x_n consists of samples from the baseband signal seen in (2.10). All continuous frequencies lying in the interval $[X_j, X_{j-1}]$ will be represented by the same value, X_j, in the discrete domain (referred to as a *bin*). There is not an infinite amount of bins, so roundings to the closest value must occur. This puts constraints on the signal model, since distances, velocities and angles can no longer be represented perfectly. Objects too close to each other will have distances in the same bin, effectively merging them into one object. The result is that only one object will be detected. Increasing the number of samples, and thereby the resolution, comes with a performance trade-off, which in the case for the DFT is substantial with a computational complexity of $\mathcal{O}[N^2]$. For example, doubling the amount of samples will require four times as much computing power.

To make the calculations more computationally feasible, the number of samples can be chosen as a power of two (2^N), which enables the use of the *Fast Fourier Transform* (FFT) instead. With a computational complexity of $\mathcal{O}[N \log N]$, the FFT is more efficient than the DFT. Obtaining 2^N samples can either be done by changing the parameters of the model accordingly, or by adding zeroes to the signal before performing the FFT, referred to as *zero padding*. Zero padding the signal increases the number of bins, which makes it easier to distinguish nearby peaks in the spectrum. However, zero padding only enables interpolation between existing values, and not extrapolation in order to find values outside the current range. Therefore, it is essential to have a relatively large amount of samples before performing zero padding. Otherwise, information at the end points will be lost. Also, the actual resolution of the FFT has not increased, meaning that if the number of data samples in use are not enough to separate two distinct frequencies, zero padding will not solve the problem.

Another problem with FFTs is spectral leakage. The spectral estimate of a sinusoid computed using the DFT or the FFT does not only contain one spike and then zeros, instead it will have a dominant peak which is smudged out over several bins. Spectral leakage occurs due to finite windowing of the data, since the data set is never infinitely large. The FFT of a rectangular window, which means that the data points are not scaled when performing the FFT, is a sinc function. The sinc function has relatively high side lobes, which causes spectral leakage. A window, e.g., Hann or Hamming, can be used when performing an FFT to reduce the height of the sidelobes and hence reduce the spectral leakage. However, the resolution will be worse since any window has a wider main lobe than the rectangular window. It becomes a trade-off between having a good resolution and having less leakage. Figure 2.4 shows the effect of zero padding and the effect of the spectral leakage. A Hanning window is used to create the FFT-spectrum in the figure and spectral leakage is still occurring.



(a) An AoA estimation without zero padding Estimated AoA: 25.4°

(b) An AoA estimation with zero padding Estimated AoA: 35.1°

Figure 2.4: AoA estimation with and without zero padding - Object AoA: 35.0°

Performing FFTs is the core to estimating the range, velocity and angle of arrival in an FMCW radar. To locate and track objects, three major steps are made.

1. Transmit a chirp and perform an FFT on the baseband signal to calculate the distance to the object. This is called a *Range-FFT*.

2. Transmit several chirps and perform a second FFT on the result from the first step to calculate the radial velocity. This is called a *Doppler-FFT*.

3. Use multiple receiving antennas and perform a third FFT on the result from the second step to calculate the AoA. This is called an *Angle-FFT*.

Below, a more detailed explanation and derivation of these steps can be read.

2.2.2 Range Estimation

The range to an object is measured by transmitting a chirp and then measuring the time it takes to detect a returning signal. This time can be expressed as

$$\tau = \frac{2D}{c}, \tag{2.12}$$

where the time τ depends on the distance the signal is travelling and on the speed of the signal, which is the speed of light. The expression is the same as the one in (2.6) for $v = 0$ since the velocity of the object is not taken into consideration when estimating the distance. If one object is present, the baseband signal will contain only one frequency, which is equal to the frequency difference between the transmitted and the received signal, f_B in Figure 2.2.

The frequency difference f_B can be calculated from the equation of a straight line according to

$$f_B = \alpha\tau, \tag{2.13}$$

where α is the slope of the chirp declared in (2.2) [Sandeep Rao, 2016]. Several objects will give rise to several constant frequencies which are proportional to the distance between the objects and the radar. The frequency spectrum of the baseband signal is estimated with an FFT. To calculate the distance to an object, the beat frequency, which is the frequency at the peak value of the FFT spectrum, is inserted in

$$f_B = \alpha\tau = \alpha\frac{2D}{c} \implies D = \frac{f_B c}{2\alpha}. \tag{2.14}$$

After the FFT, the x-axis consists of discrete bins which are normalized frequencies in the range $[-\pi, \pi]$. The number of bins is equal to the number of samples in the chirp if zero padding is not occurring, otherwise the number of bins can be altered by choosing the amount of zeros inserted at the end of the signal. To make the normalized frequencies depend on the sampling frequency f_s, the x-axis is scaled according to

$$f = \frac{n \cdot f_s}{N} \tag{2.15}$$

for each bin n. Here, N is the number of samples in the chirp, n is the current bin that is scaled and f_s is the sampling frequency, which is equal to

$$f_s = \frac{N}{T}, \tag{2.16}$$

since f_s denotes the amount of samples per second. Then, (2.14) can be seen as another re-scaling of the x-axis, so it displays the distance that each frequency corresponds to.

2.2.3 Radial Velocity Estimation

By sending out multiple consecutive chirps, the radial velocity of a detected object can be estimated. The range of the object will not change significantly between the chirps because of the small time difference. However, the velocity of the detected object will generate a phase change between the waves of the different chirps. The reflected waves will enter the radar with different phases, depending on the velocity of the object. Hence, the reflected wave for the second chirp will approximately contain the same frequency as the first chirp, but with another phase [Sandeep Rao, 2016]. The phase difference for two different waves is illustrated in Figure 2.5.

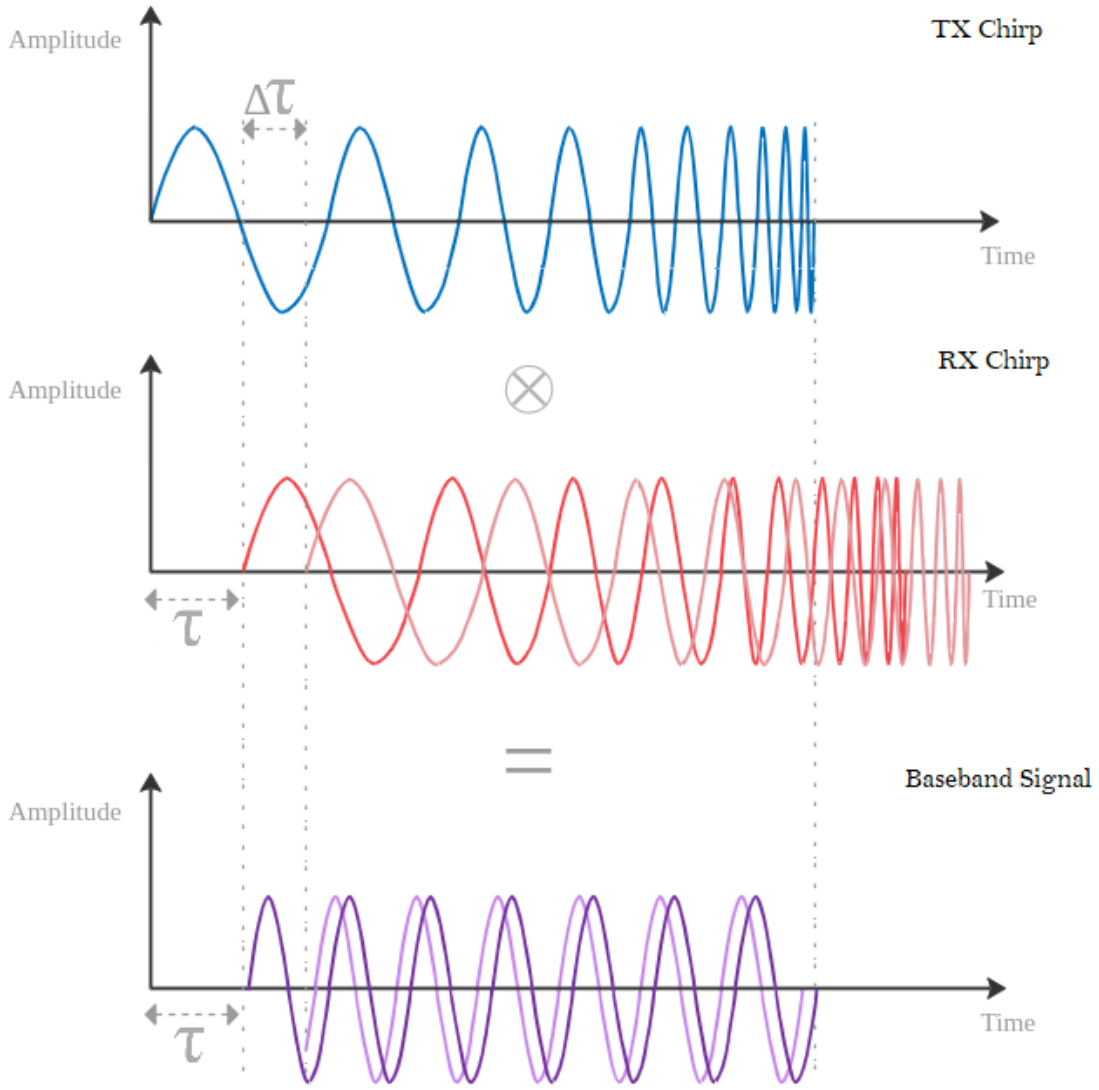


Figure 2.5: Two reflected waves with a phase difference resulting in baseband signals with a phase difference.

Figure 2.5 shows that the two baseband signals have the same frequency, but different phases. The brighter rx-wave in Figure 2.5 reaches the radar slightly later than the darker rx-wave. The phase of the tx-wave is different at time point $\tau + \Delta\tau$ compared to time point τ. The phase of the baseband signal depends on the phase of the tx-wave and on the phase of the rx-wave, which causes the phase difference of the two baseband signals.

The phase shift between consecutive chirps is the same for any two chirps and it can be calculated by performing an FFT with input from the different chirps composing the frame. The input is in the form of a Range-FFT from each chirp. The result of this process is a Range-Doppler spectra which is a matrix with the dimensions $N \times M$ where M is the number of chirps and N is the number of samples per chirp, if no zero padding is occurring. A visualization of the process can be seen in Figure 2.6 where each baseband signal has a certain phase. The beat frequency is constant when the frequency difference between the transmitted and the received

signal is constant. Each chirp gives rise to a baseband signal which is inserted column wise in a matrix. Column wise FFTs are performed for a range estimation. Then, row wise FFTs are performed for a radial velocity estimation. The result is one peak for each detected object where the row of the object corresponds to the range and the column corresponds to the velocity. The Doppler-FFT is shifted in frequency to change the range from $[0, 2\pi]$ to $[-\pi, \pi]$ in the normalized frequency domain. This enables detection of negative velocities.

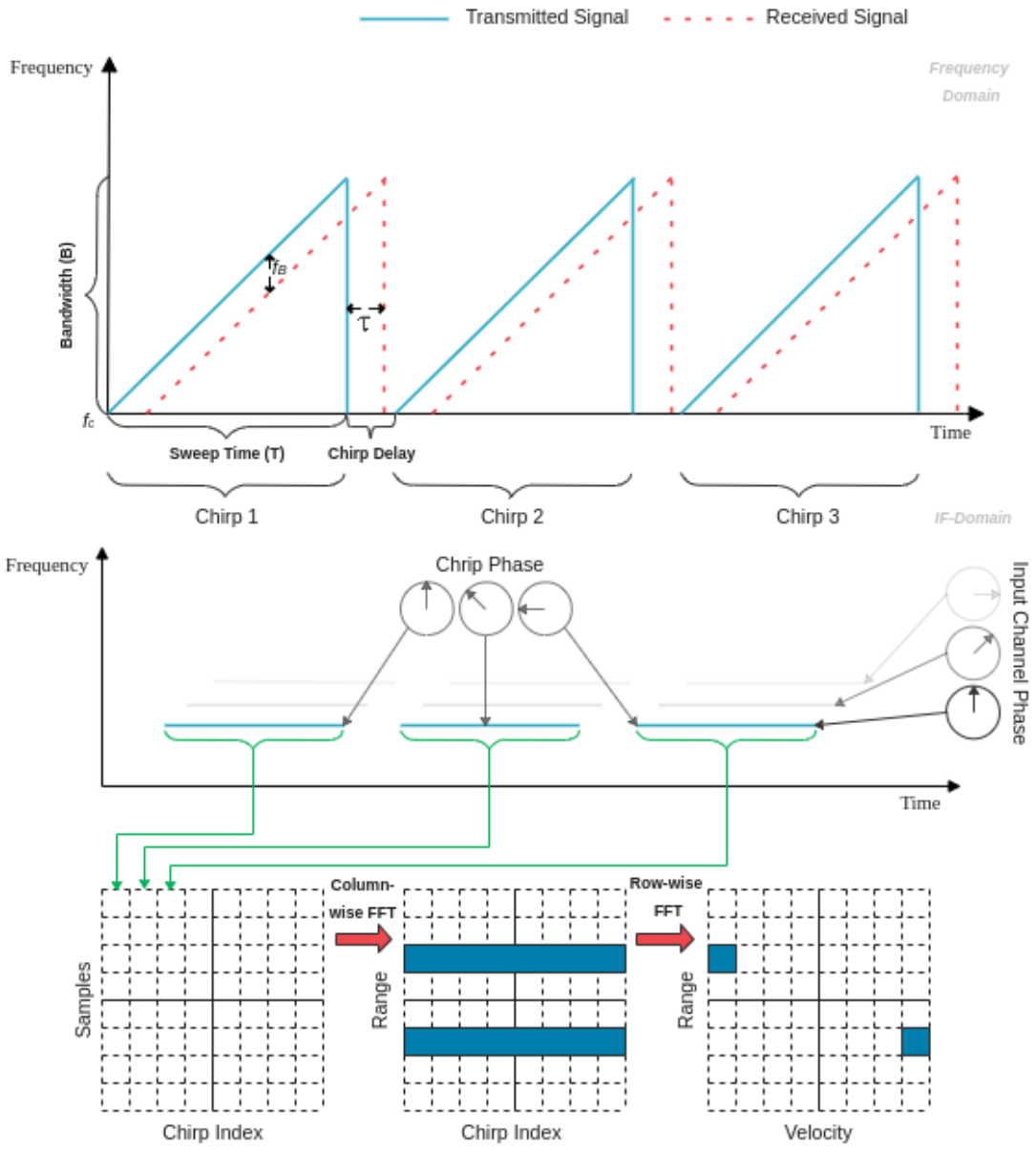


Figure 2.6: The basic principle of FMCW-radar. The transmitted and received waveform at each input channel are mixed. The data is then subject to a series of Fourier transforms, providing information of the location of potential objects.

As before, the x-axis of the FFT consists of normalized frequencies. The conversion from bin to non-normalized frequency is the same as before, where each bin is multiplied with the sampling frequency and divided with the total amount of samples.

However, the total amount of samples is now M since the FFT is performed across the chirps, and not for each chirp at a time. The sampling frequency is the amount of chirps during one second and can be expressed as

$$f_{chirps} = \frac{1}{T}, \tag{2.17}$$

where T is still the duration of one chirp. The bins are converted to frequencies according to

$$\omega_v = \frac{m \cdot f_{chirps}}{M} \tag{2.18}$$

for each bin m.

The phase difference between the chirps corresponds to a motion of the object of distance vT, which is how far the object has travelled during the chirp time [Sandeep Rao, 2016]. This means that the signal has travelled an extra distance of $2vT$. To convert distance to phase, it is essential to know that a shift of one wavelength λ corresponds a phase shift of 2π for a sinusoid. To convert the extra distance to a phase shift, it is scaled according to

$$\omega_v = \frac{2\pi}{\lambda} 2vT, \tag{2.19}$$

where λ is the wavelength of the carrier frequency. The radial velocity can then be calculated from the phase according to

$$v = \frac{\lambda \omega_v}{4\pi T}, \tag{2.20}$$

where ω_v is the frequency with the highest peak in the FFT.

2.2.4 AoA Estimation

To calculate the AoA, several receiving antennas are used and the phase difference between the antennas depends on the AoA and on the distance d between two neighboring antennas. A uniform linear array of antennas can be built by placing the antennas on a line according to Figure 2.7. Other constellations are also possible, especially if it is desired to measure the elevation of an object, compared to the radar, instead of only measuring the displacement sideways. Note that the distance d between the antennas is not the same as the distance D between the radar and an object. Detected objects are usually positioned in the far-field of the radar, meaning that the reflected waves can be approximated as being parallel. By assuming that the reflected signal consists of parallel waves, the calculations are simplified. The antennas will register the signal at different times due to a path length difference between the waves. The phase difference between three neighbouring antennas is presented in Figure 2.6 as the input channel phase. The incoming waves will have different phases when reaching the antennas due to the extra path length.

By using trigonometry, the path difference between adjacent antennas, x in Figure 2.7, is equal to $d\sin(\theta)$ where θ is the AoA [Richards et al., 2015]. The phase difference ω_a, due to the AoA, of two adjacent antennas is

$$\omega_a = \frac{2\pi d}{\lambda} \sin(\theta) \tag{2.21}$$

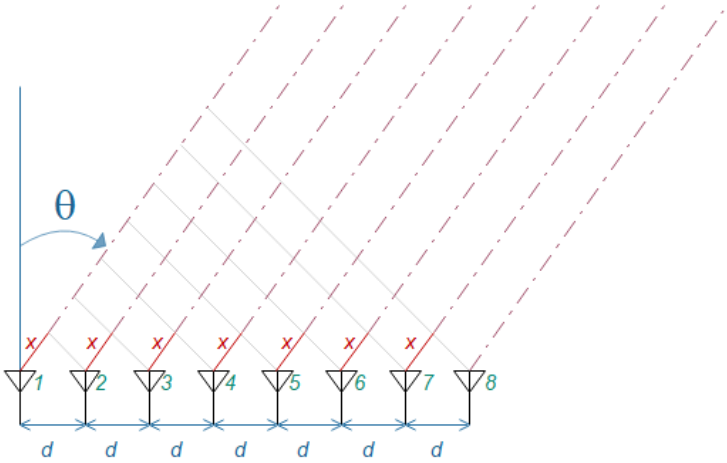


Figure 2.7: The angle of arrival generates path differences between the waves that the receiving antennas in the radar detect. The path difference between two neighboring antennas is denoted x.

according to the same re-scaling principle as in (2.19), because a phase shift of one wavelength corresponds to a phase shift of 2π.

It is assumed that the antennas are placed at a distance of $d = \frac{\lambda}{2}$ from each other. This placement gives the user the widest possible field of view, which is -90 to 90 degrees. The final expression for calculating the angle from the phase difference ω_a is

$$\theta = \sin^{-1}\left(\frac{\lambda\omega_a}{2\pi d}\right) \tag{2.22}$$

[Suleymanov, 2016]. This result can then be used to calculate the *Cross Range* (CR), which relates the angle to a distance. The CR, defined as

$$CR = D \cdot \sin(\theta), \tag{2.23}$$

describes how far the object is located sideways relative to the radar. This is simply another way of expressing the AoA depending on the distance between the radar and the object. The CR is visualized in Figure 2.8, where the blue point represents an object at a distance D from the radar.

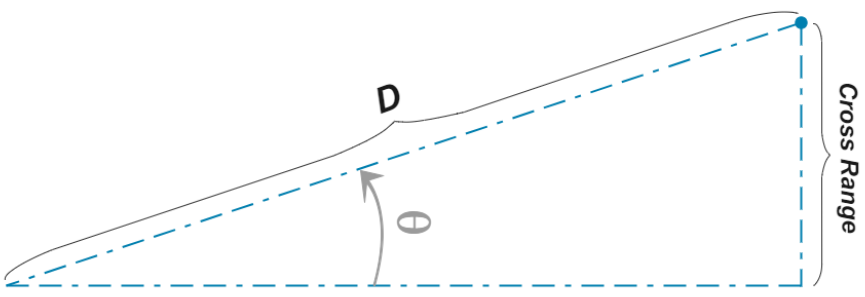


Figure 2.8: Definition of cross range.

2.2.5 Limitations to the Resolution

The resolution of the measurements depend on the chosen system parameters. The resolution for the range measurement is improved if the amount of samples per chirp (N) is large. The amount of chirps (M) determines the resolution for the velocity measurement, where many chirps produce data with good resolution. The amount of antennas determines the resolution in the angular direction. The maximum distance that can be detected depends on the sampling rate of the analog-to-digital converter and is expressed as

$$d_{max} = \frac{f_s c}{4\alpha}. \tag{2.24}$$

The maximum radial velocity that can be measured is

$$v_{max} = \frac{\lambda}{4T}. \tag{2.25}$$

The limitation of the velocity is due to an ambiguity that arises for phase differences greater than π. If it is allowed to measure a phase difference greater than π, it is impossible to know if the object is moving towards or away from the radar. The restriction on ω_v creates a restriction on the maximum measurable velocity, see Figure 2.9. In the case where the velocity of the object being measured is greater than v_{max}, information regarding both the velocity and the direction of movement will be lost [Sandeep Rao, 2016].

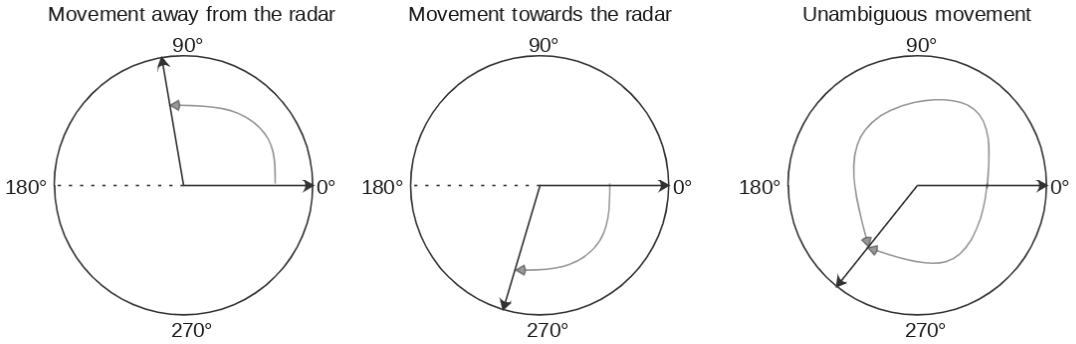


Figure 2.9: A phasor diagram depicting the restrictions to phase measurements. Without a limitation on the maximum measurable phase difference, it would not be possible to distinguish if an object is moving away from or towards the radar.

The measurement for the angle is slightly different from range and radial velocity estimation, due to (2.22) being non-linear. The non-linearity implicates that resolution is worse for objects far out to the sides and better for angles close to 0°. In Figure 2.4a, an example with eight antennas is shown, where every stem in the plot represents the location for an antenna bin. For objects right in front of the radar, the resolution is good, where it is possible to determine an object's angular position without generating large round-off errors. At the sides however, the resolution is poor.

Having this poor resolution in the angular domain is a fundamental limitation, but it should in theory be possible to tell apart an object between two bins. An object between two bins would have both of the bins indicating an object, but one bin would have a slightly higher power than the other, indicating that it is closer to that bin. Either one could try to figure out the relationship between relative bin power and the position of an object, or one could utilize the zero padding concept discussed at the start of Section 2.2.

2.2.6 Mirroring in FFTs

In a 1D-FFT, only the magnitude of the frequency is regarded, i.e., how much of each frequency a wave contains. This is because the data at the receiver is solely in the real domain. The magnitude is mirrored to the negative half plane because a pure sinusoid consists of a positive and a negative frequency according to Euler's formula. For each frequency, the spectrum will therefore show two peaks, one in the negative and one in the positive plane. In the Range-FFT, a peak corresponds to a distance and negative peak would correspond to a negative distance, which has no meaning. Therefore, half of the spectrum is discarded so only the positive half is used in further processing.

For the Doppler and Angle FFT, it would be unfortunate if the negative spectrum had to be omitted, since negative angles and velocities can occur. However, mirroring only happens when one takes an FFT of a purely real signal. Since the output of the first FFT is complex, there is no need to discard data when performing Doppler and Angle FFTs.

2.3 Visualization of Radar Data

2.3.1 The Range-Doppler Map

Perhaps the most common way to visualize a radar signal is to plot it after the second FFT, creating a so called *Range-Doppler* (RD)-map. Here, the velocity and the distance to the object can be observed. The angle remains unknown however, meaning that objects with the same velocity and distance will appear at the same location in this type of plot, see Figure 2.10. An RD-map has velocity on the x-axis and range on the y-axis. A bright spot, for example the one in Figure 2.11 implicates that an object is present and the distance and the velocity can be read by observing the coordinates of the spot. For example, the detected object in Figure 2.11 is 250 m away from the radar and has a velocity of -13 m/s. An RD-map is sometimes subject to smearing where objects are present. This is due to spectral leakage generated in the FFT, which was explained in Section 2.2.1. Smearing can also be a consequence of an object not being a point source. Then, the distance to the object will be spread out over several bins because different parts of the object falls in different bins after a range-FFT. The same applies for the velocity if an object contains moving parts with different velocities, e.g., a human walking and swinging the arms.

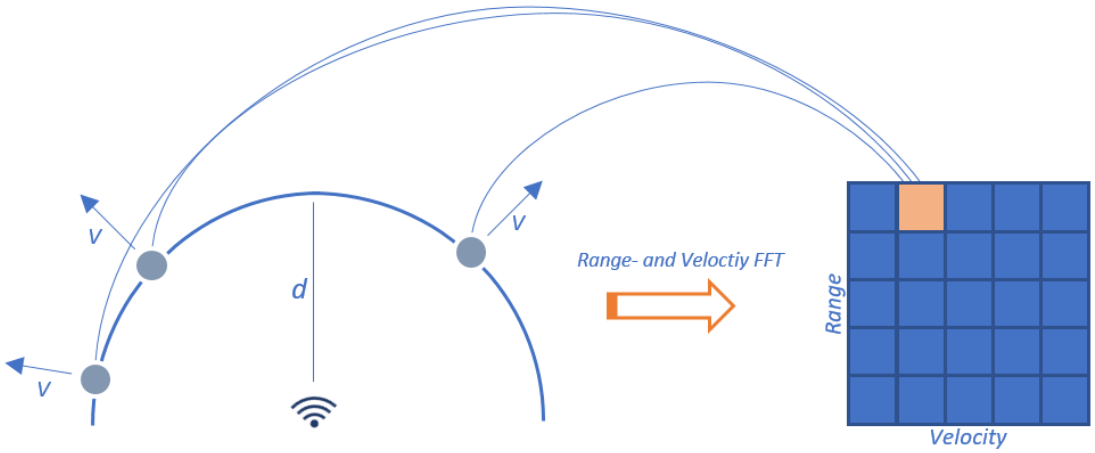
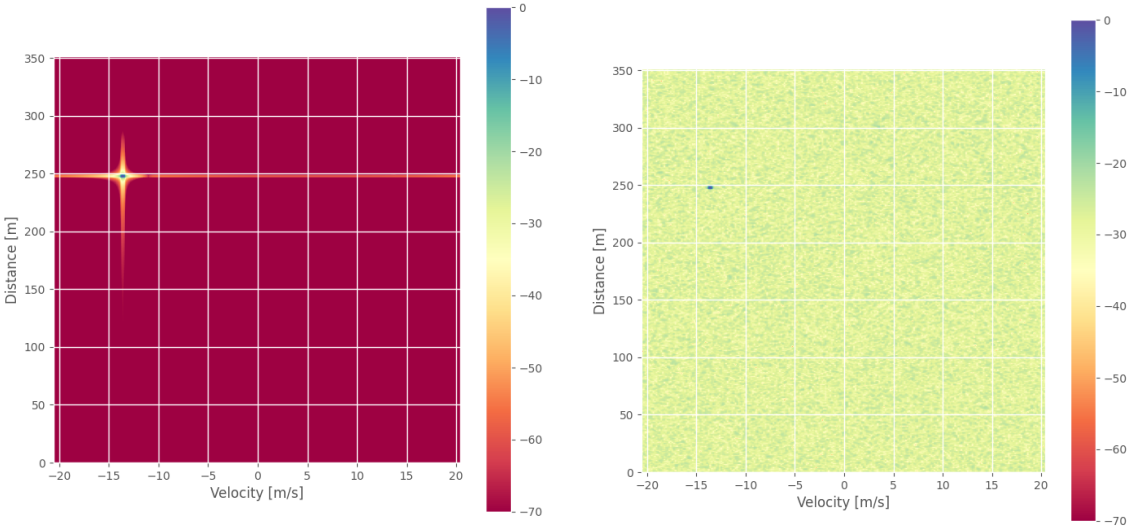


Figure 2.10: All objects at a distance, d, from the radar and with an equal velocity, v, will appear in the same bin in an RD-map.



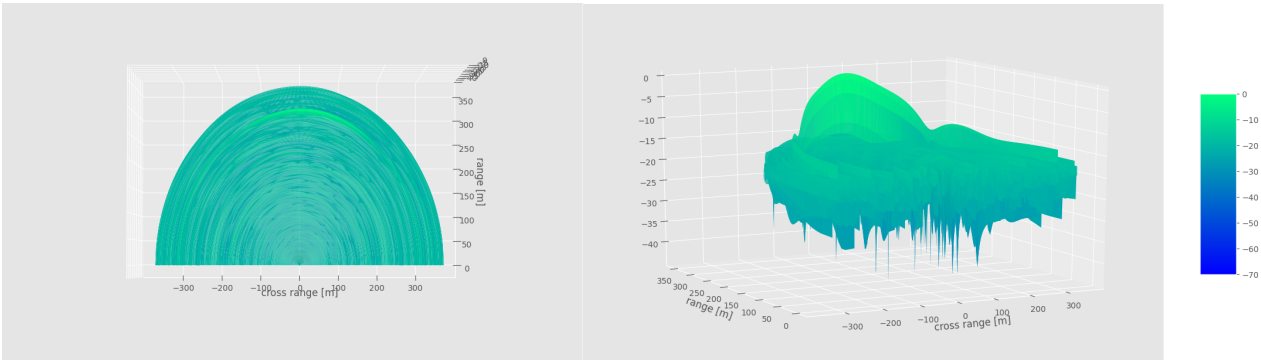
(a) An RD-map without noise. The object smearing is caused by leakage in the FFT.

(b) Same RD-map as in (a), but with white Gaussian noise. The object smearing is still present, but is below the noise floor.

Figure 2.11: RD-maps with and without white Gaussian noise.

2.3.2 The Range-Angle Map

The *Range-Angle* (RA)-map functions similarly to the RD-map, but with angles being displayed instead of velocities. Analogously to Figure 2.10, multiple objects with different velocities would not be distinguishable if they were at the same distance and angle from the radar. The RA-map is plotted with distance to the object on the y-axis, cross-range on the x-axis and peak intensity on the z-axis. An example of an RA-map can be seen in Figure 2.12. The bright spot corresponds to an object being present at a distance of 300 m from the radar and with a cross-range of approximately 50 m. The RA-map is also subject to smearing due to spectral leakage.



(a) An RA-map with white Gaussian noise, shown from above.

(b) Same RA-map as in (a), but shown from the side.

Figure 2.12: An RA-map shown from two different angles

2.3.3 The Range-Doppler-Angle Cube

The *Range-Doppler-Angle* (RDA)-cube displays the distance, velocity and AoA of an object. This requires all three dimensions to be used, which may make it a bit more cumbersome to analyze. This is further exacerbated by the angle estimation being non-linear, as described in Section 2.2.5. It can be hard to visualize all points, especially the ones in the middle of the cube. It is also necessary to turn the cube in order to observe the exact values of the object. Figure 2.13 shows an RDA-cube with sparse data points for the sake of visualization. One detected object is seen in the lowermost angle bin.

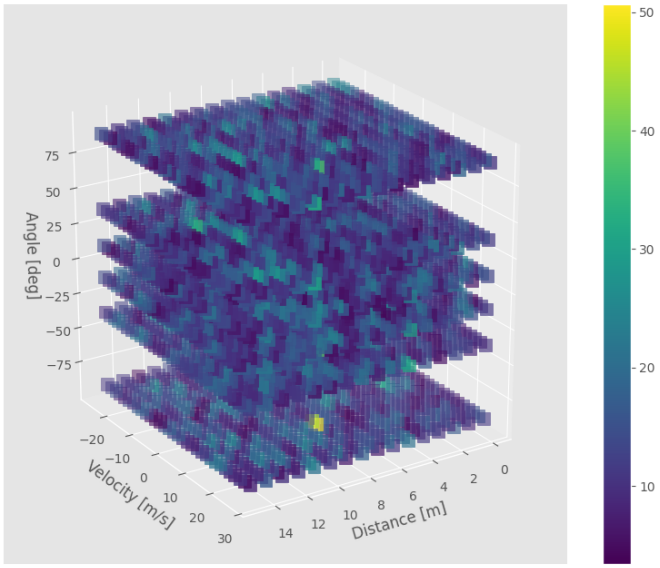


Figure 2.13: An RDA-cube with white Gaussian noise, where the point with the highest power corresponds to the position of the real object.

2.4 Disturbance from other radars

FMCW radars are becoming more popular, mainly due to the rapidly growing demand for smart and self-driving cars. Since many car manufacturers have developed their own technologies, the outgoing radar signals are different which makes interference even harder to combat since it can have different appearances and effects. For surveillance purposes, where the installed radars might be of very similar nature, other problems arise, where the interfering radar signal might sometimes be indistinguishable from a true object signal. Furthermore, the signal pulse from the interfered radar, from now on denoted the *victim* radar, has to be reflected from an object and return back, and it can be shown that the signal amplitude of the reflected wave is proportional to D^{-4}, where D is the distance between the object and the radar. The area, material and shape of the object also affects the signal amplitude. The signals from the interfering radars, from now on denoted *aggressor* radars, have not necessarily reflected off anything, and the signal power is therefore only proportional to D^{-2}. This can sometimes lead to cases where the interference is several orders of magnitude larger than the desired object signal. Figure 2.14 gives an idea of how the victim radar might be affected by interference.

Managing the interference is becoming more and more important - especially considering that the radar sector likely will continue to grow [Chang et al., 2019]. In the following parts of the report, several approaches to eliminating radar interference are modeled and discussed.

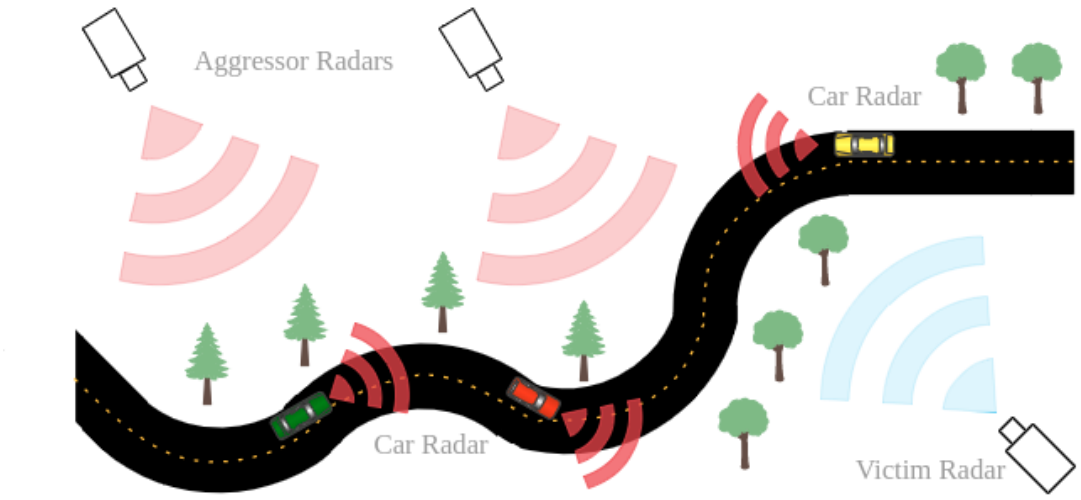


Figure 2.14: Schematic representation of interference from other radars.

2.4.1 Signal Model for Interference

It is always the baseband signal, which arises from mixing the aggressor and the victim signals, that is further processed, and not the two high-frequency signals. As mentioned previously, the frequency of the baseband signal only consists of the frequency difference between the victim wave and any incoming wave, in this case an aggressor wave. Because both the aggressor wave and the victim wave contain linearly increasing frequencies, the baseband signal will have a linear frequency. An example of how the baseband frequency is calculated be seen in Figure 2.15.

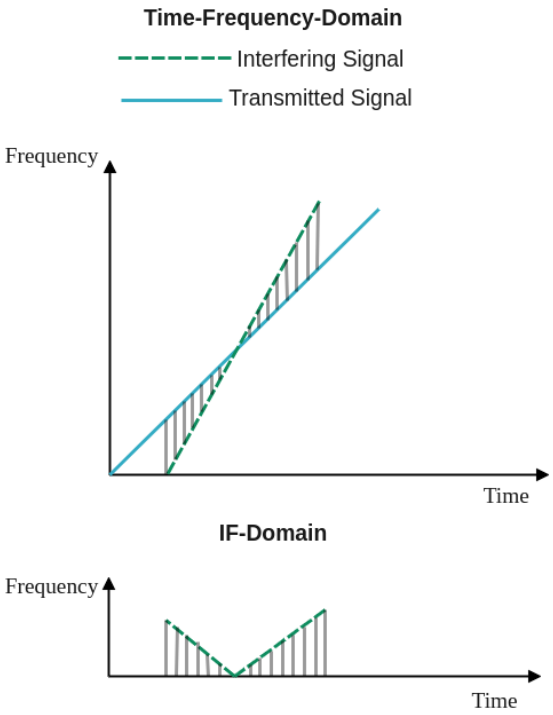


Figure 2.15: How the frequency of the baseband signal is calculated in the IF-domain, from mixing of the aggressor and the victim.

The aggressor contains a linearly increasing frequency according to

$$f(t) = \alpha_a t + f_0, \tag{2.26}$$

where f_0 is the starting frequency. It has the same appearance as the transmitted signal in Figure 2.2. The chirpyness constant, α_a, is defined by

$$\alpha_a = \frac{f_1 - f_0}{T_a}, \tag{2.27}$$

where f_1 is the highest frequency of the chirp and T_a is the duration of the aggressor wave. The chirpyness constant has the same meaning as the slope α for the victim.

The frequency of the aggressor's baseband signal is a linear function, seen in the lowermost graph in Figure 2.15. The linear function can be described by

$$f_{IF}(t) = k_{IF} t - k_{IF} \tau_a, \tag{2.28}$$

where τ_a is the point in time where the baseband signal is equal to 0, $f_{IF}(\tau) = 0$, and k_{IF} is the slope of the frequency of the baseband signal. Because the phase in the time domain is the integral of the frequency function ($\phi'(t) = 2\pi f(t)$), the phase of the disturbance is equal to

$$\phi(t) = \phi_a + 2\pi \int_0^t f(t')dt' = \phi_a + 2\pi \int_0^t (k_{IF}t' - k_{IF}\tau_a)dt' = \phi_a + 2\pi \left(\frac{k_{IF}}{2}t^2 + -k_{IF}\tau_a t \right), \tag{2.29}$$

where ϕ_a is the initial phase at the starting point of the wave. The final normalized expression of the aggressor wave is

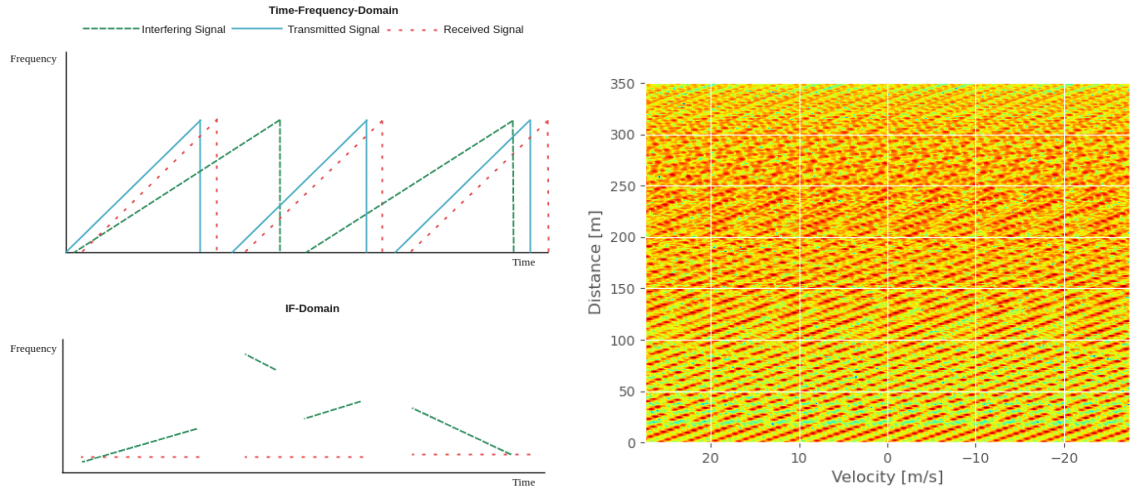
$$x(t) = \cos[\phi(t)] = \cos\left[\phi_a + 2\pi \left(\frac{k_{IF}}{2}t^2 + -k_{IF}\tau_a t \right) \right] \tag{2.30}$$

[Toth et al., 2018]. The baseband signal of the interference usually operates over a wide frequency band where different types of radars disturb each other, causing interference over several frequencies. Because many frequencies are affected, the interference will be spread out in the RD-map. The effects of interference are disruptions in the form of glitches and an elevation of the noise floor. The noise floor can be seen as a sum of all of the noise and interference within the system. If no object is present, the signal only consists of noise and interference, which forms the noise floor. Interference can in turn lead to false detections and to a decrease in sensitivity which makes it harder to detect objects that reflect weak signals [Yang and Mani, 2020]. A reduction of the interference can increase the *Signal-to-Noise Ratio* (SNR), which describes how strong the energy of signal is compared to the energy of the noise. The parameter *Signal-to-Interference-and-Noise Ratio* (SINR) can also be regarded. Here, the energy of the signal is compared to the energy of the noise and the interference. These metrics are explained more detailed in Section 3.3.1.

There are some generalizations to the different types of interference that the victim radar can expect to receive. These are non-coherent interference, semi-coherent interference and coherent interference. The properties of these types of disturbances differ and give rise to different effects.

2.4.2 Non-Coherent Interference

Non-coherent interference is the type of interference that occurs when the victim and the aggressor radar have **completely different** characteristics. This interference becomes very random in the eyes of the receiving radar, with few patterns and with different properties for all disturbances received. An illustration of this type of interference and its RD-map is shown in Figure 2.16.



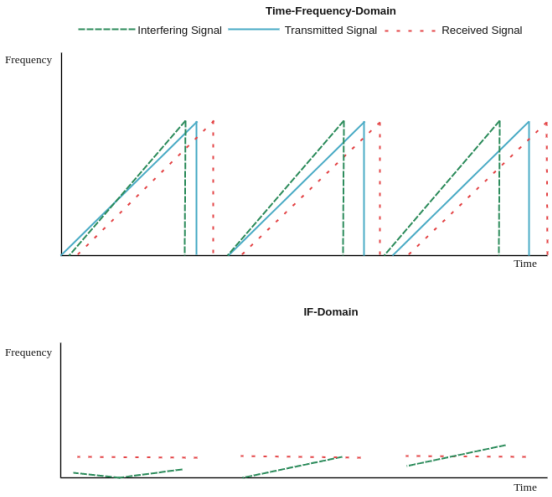
(a) A visualization of non-coherent interference in the frequency domain and in the IF-domain.

(b) An example of an RD-map with non-coherent interference.

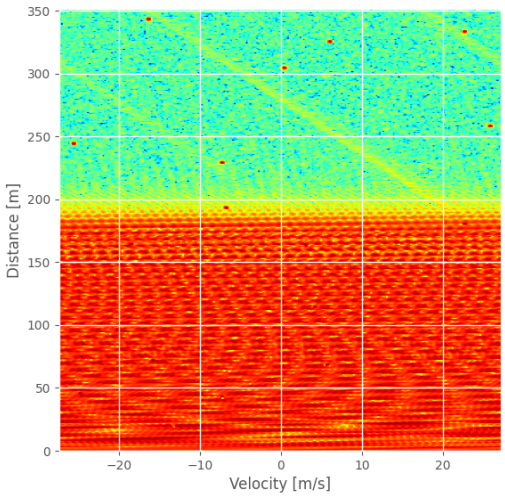
Figure 2.16: Non-coherent interference in the frequency domain and in an RD-map. An increased noise floor is present in the entire RD-map.

2.4.3 Semi-Coherent Interference

Semi-Coherent interference occurs when the victim and the aggressor radar have **slightly different** characteristics. This type of interference can be seen in Figure 2.17. Semi-coherent interference can be viewed as an intermediate between non-coherent and coherent interference. In Figure 2.17, the interference does not occur in the whole RD-map. This is due to the frequency difference between the aggressor and the victim not being as large as for non-coherent interference. A larger frequency spread of the baseband signal means that the interference covers more of the RD-map. It is possible that semi-coherent interference covers the whole RD-map, but it is more likely that this happens for non-coherent interference. Both semi-coherent and non-coherent interference will increase the noise floor and it will be harder to detect objects, where the interference is present. This can be seen by comparing the RD-maps without noise in Figure 2.11 with the interfered RD-maps in Figure 2.16 and Figure 2.17.



(a) A visualization of semi-coherent interference in the frequency domain and in the IF-domain.

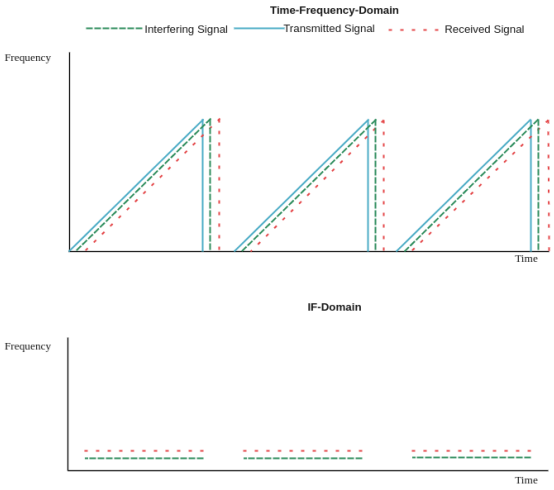


(b) An example of an RD-map with semi-coherent interference.

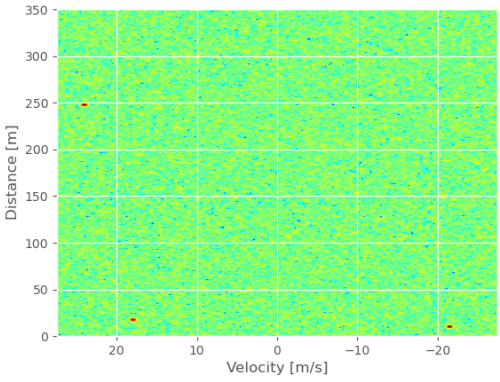
Figure 2.17: Semi-coherent interference in the frequency domain and in an RD-map. There is an increased noise floor in a part of the RD-map.

2.4.4 Coherent Interference

Coherent interference occurs when the victim and the aggressor radar have **identical** characteristics. When two identical radars are interfering with each other, coherent interference arises. This case is the most difficult one to deal with, since the interference becomes indistinguishable from a true object signal. Coherent interference is shown in Figure 2.18. If the interference is coherent, there will be no



(a) A visualization of coherent interference in the frequency domain and in the IF-domain.



(b) An example of an RD-map with coherent interference.

Figure 2.18: Coherent interference in the frequency domain and in an RD-map. There are only two objects in front of the radar, but the coherent interference makes it look like three.

difference between the transmitted, received and interfered wave form except for time delays. The baseband signal from the aggressor and the victim would therefore result in a wave with a constant frequency, f. Putting that into (2.29), the phase of

the baseband signal would become the difference between the phase of the aggressor and the phase of the victim according to

$$\phi_m(t) = \left(\phi_a + 2\pi \int_0^t f d\tau\right) - \phi_v = \phi_a - \phi_v + 2\pi f \cdot t, \tag{2.31}$$

where ϕ_v and ϕ_a is the initial phase for the victim and aggressor radar respectively. The resulting expression for the baseband signal then becomes

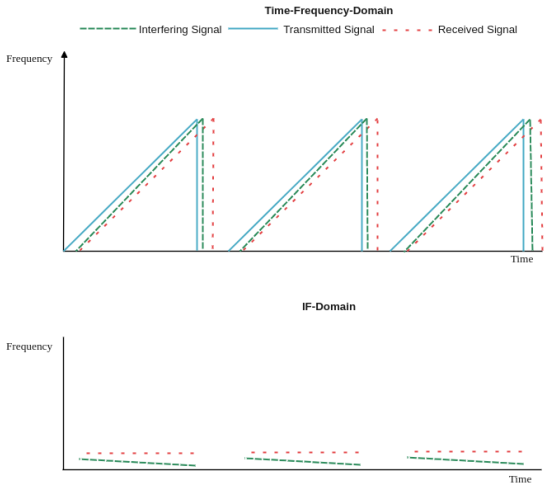
$$x_m(t) = \cos[\phi_m(t)] = \cos(2\pi t f + \phi_a - \phi_v). \tag{2.32}$$

The FFT of above expression would result in a single peak at frequency f, which is exactly what one would expect from a true object. A Doppler-FFT would find the phase difference between consecutive chirps, which is $\phi_a - \phi_v$. A constant range and a constant phase difference will look like an object in the RD-map. However, this is not a true object, but interference, hence creating a so called *ghost object* in the RD-map. A ghost object has the appearance of a true object, but it is generated by an aggressor. There is a possibility of separating a ghost object from a true object, by using a hardware phenomena called *clock drift*, see Section 2.4.5.

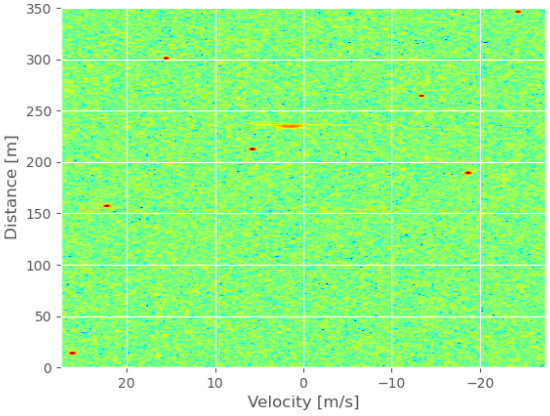
2.4.5 Clock drift

Clock drift is a phenomena that occurs due to different internal clocks within different radars. A radar has an internal clock which determines when the radar should chirp and when to be inactive, given a frequency. The internal clocks between radars are similar, but not exactly the same because it is not physically possible to create identical copies due to hardware limitations. Even if chirps of two different radars have the same properties on paper, they will be perceived differently by each of the radars. Coherent interference with clock drift can be seen in Figure 2.19 where the victim and the aggressor have the same properties, but are perceived differently. The figure depicts a coherent interference signal subject to clock drift, together with the victim radar's transmitted wave. The effect of the clock drift is that the aggressor chirp will be perceived like it lasts for a longer/shorter period of time than the victim chirp, according to the victim. This will primarily affect the slope of the aggressor, since the slope depends on how long the chirp is, given a constant bandwidth. The clock drift will also affect the subsequent aggressor chirps since they will start slightly sooner/later depending on when the previous chirp ended.

The slope in the IF-domain will affect the beat frequency since this will be spread out over a few values instead of being constant as it would be without the clock drift. The frequency spread is seen by comparing Figure 2.18 with Figure 2.19. The phase of the baseband signal will also be altered since it depends on when the aggressor chirps start, relative to the victim chirps, and this will be affected by the clock drift. Instead of one constant phase shift between two consecutive chirps, the phase shift will be slightly changed depending on which chirp that is considered. The first aggressor chirp in one frame will start approximately when the victim chirp starts. The last aggressor chirp in the same frame will be more displaced as compared to the last victim chirp, and this will generate a larger phase shift. The result is a ghost object which is more spread out than a ghost object without clock drift. Figure 2.20 shows two examples with different amounts of clock drift.

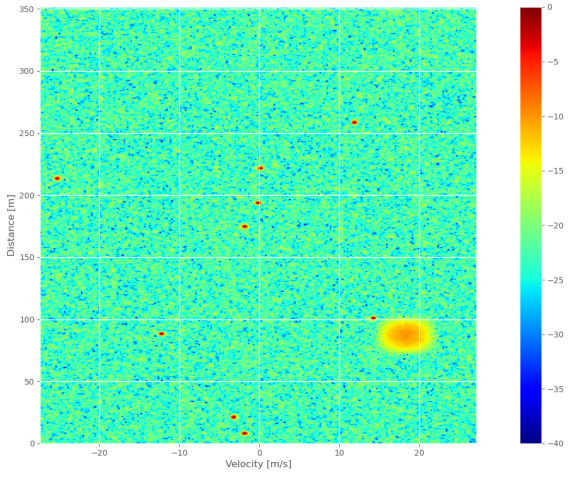


(a) A visualization of coherent interference with clock drift in the time-frequency domain and in the IF-domain.

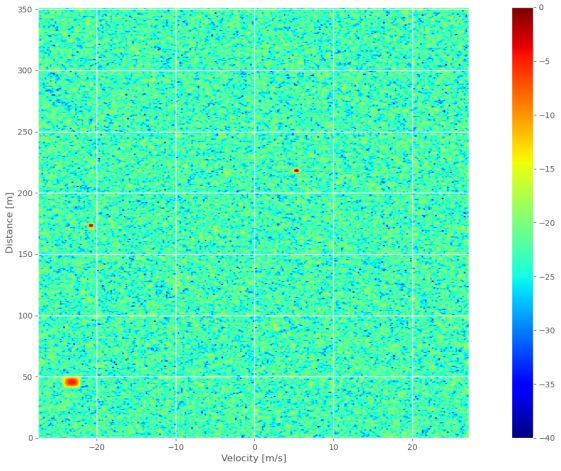


(b) An example of an RD-map containing coherent, interference with clock drift. The interference creates a smeared ghost object.

Figure 2.19: Coherent interference with clock drift in the frequency domain and in an RD-map.



(a) Example of an RD-map with a ghost object that has 20 PPM clock drift.



(b) Example of an RD-map with a ghost object that has 5 PPM clock drift.

Figure 2.20: Two RD-maps with objects where the data is subject to different amounts of clock drift.

Clock drift is a drawback in the radar system, since it is expected that products with the same specifications behave in the same way. The flaw can however be used when training a neural network. The slight variation of the chirps from different radars makes it possible to distinguish received radar signals, sent from the victim radar, from interfering signals. Received radar signals will have a constant frequency and a constant phase shift between the chirps. Interfering signals will have frequencies and phase shifts smeared over several values due to the clock drift. The difference can be used to train a neural network for interference detection and mitigation. Without the clock drift, it would be more difficult to determine if the receiving antennas are detecting true signals or interfering signals, since the signals would look the same.

2.4.6 IF-band

It can be desirable to only regard frequencies within a certain range from the transmitted frequencies, instead of regarding every possible frequency that can be detected by the radar. This range is termed the *IF-band* and it determines which frequencies should be included in the baseband signal. The width of the IF-band is the noted the IF-bandwidth.

Only certain parts of the aggressor is affecting the victim, depending on how close their frequencies are. The victim radar is only affected when the frequency of the aggressor falls into the IF-bandwidth of the victim. The IF-band determines how far a detection can occur. All frequencies that exceed the IF-band are filtered out and therefore not affecting the baseband signal.

The IF-band is usually set from 0 Hz to the frequency corresponding to the maximum detectable range, d_{max} in (2.24). By restricting the maximum intermediate frequency, aliasing is prevented because frequencies above the maximum detectable frequency are not regarded. The IF-band results in an efficient detection because of the frequency limitation [Murali et al., 2018]. The effect of the IF-band is shown in Figure 2.21. The IF-band can exclude different amounts of interference, depending on which type of interference that is regarded. When the frequency of coherent interference is present close to the victim frequency, the IF-band will have no impact at all, since all interference frequencies lies within the band. An example of this type of interference is seen in Figure 2.19. However, if the delay between the victim signal and the aggressor signal is large, there will not be any part of the aggressor within the IF-band and the whole aggressor signal will be excluded. Coherent interference usually either affects many samples in a frame or no samples at all, if the interference lasts for the entire frame.

The baseband signal of non-coherent interference has more high frequency content, which will be excluded since it is present outside the IF-band. The high frequency content can be seen in Figure 2.16. So for non-coherent interference, the IF-band causes a substantial amount of interference to be excluded since the frequency difference between the victim and the aggressor can be large. This can lead to a few samples being interfered at a time, instead of one entire chirp being subject to interference.

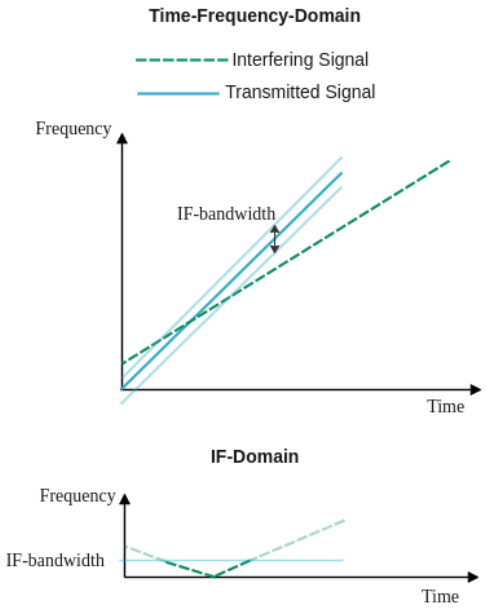


Figure 2.21: How the IF-bandwidth limits the amount of disturbance that is regarded in the baseband signal.

The effect of the IF-band on semi-coherent interference is an intermediate of the cases above.

2.5 Machine Learning - Artificial Neural Networks

A *Neural Network* (NN) is used to train a computer to solve various tasks, e.g., classification, language processing, and facial recognition. The network can be seen as a mathematical function with inputs and outputs. An NN consists of layers in the form of an input layer, an output layer and *hidden layers* between the input and the output. The inputs and outputs of the hidden layers are never examined, only what comes out of the output layer is examined. In that sense, the information to and from hidden layers are only used within the NN and never investigated, and therefore they are hidden to the observer. The hidden layers are compositions of simple functions and consist of hidden units which are activated if the input is strong enough. The input determines if an activation will occur and how strong it is. In each unit, the input is manipulated by a function, called an activation function, and the output is further sent to the following layer. The units in one layer are connected to send and receive signals from units in other layers.

In a fully connected NN, every node in one layer is connected to every node in the previous layer and to every node in the next layer. The input layer sends information to the first hidden layer, where the units can be activated and further send information to the next layer.The strength of the signal to one unit depends on the output from previous units, but also on the weights connected to the outputs. The previous units are more or less important to the next unit and the level of importance is quantified as a weight, where a greater weight leads to a larger impact [Lavrenko and Goddard, 2016a].

2.5.1 Convolutional Neural Networks

A *Convolutional Neural Network* (CNN) is a type of a deep neural network using the convolution operation with a certain filter and the input data. It is mostly used for pattern recognition because of the ability to find edges and patterns. Compared to other types of NNs, a CNN is less complex because it focuses on a specific type of detection. Instead of examining everything that is sent to the network and how it is structured, it scouts for patterns [O'Shea and Nash, 2015]. This yields a system with fewer parameters which does not require as much computational power as a fully connected NN. Another advantage with CNNs is that they are useful in the spectrogram domain, which tells how the frequencies in a signal vary over time [Rock et al., 2019b]. CNNs are beneficial because pattern recognition is the key concept in detecting radar interference since the waves subject to interference have a different pattern than clean waves. The network can find useful features from the pattern in the input data. An illustration of how a CNN can look is presented in Figure 2.22. The detected pattern for every unit depends on the structure of

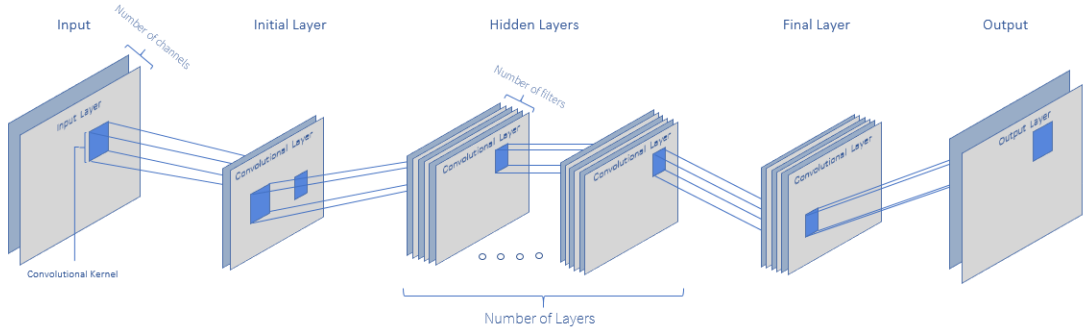


Figure 2.22: A typical CNN architecture

a sliding filter. The filter covers a few data points at a time and the convolution between the filter and the affected data points is calculated and sent to a unit in a consequent layer. Some filters are used to detect vertical lines, some detect curves and other characteristics can also be detected. Every unit is connected to a certain filter, which makes it specialized in detecting a certain pattern. There is a huge variety of filters and hence a huge variety of detectable characteristics. The features and characteristics of an object describes the object. So if the correct characteristics can be detected, the object can be classified [O'Shea and Nash, 2015].

The layers in a CNN are fairly basic in the beginning, but gradually gains complexity because the last layers gain information from more units than the first layers. The first layer may only find specific lines, which are sent to the next layer that can connect the lines to more meaningful patterns and in turn further send the information to a subsequent layer. The features become more abstract deeper into the network and and goes further away from the physically visible edges and lines, which the first layers handle.

2.5.1.1 Activation Function

An activation function takes an input to a unit and returns an output depending on the input and on the chosen activation function. The purpose of the activation

function is to introduce non-linearities to the NN. The non-linearities are important because otherwise the network would only be adding together a series of linear combinations using weights, which would result in a model that would probably not be powerful enough to learn the complex functional mappings required to reach the desired results. The result would be a linear combination of linear functions, which would generate the same result as only having one layer. There are several different activation functions, but the most common one is the *Rectified Linear Unit*, *ReLU*. It takes the output from the previous node and removes the negative values through the function

$$R(Z) = max(0, Z), \tag{2.33}$$

where Z is the output from the previous layer, and $R(Z)$ is the input to the new layer.

2.5.1.2 Batch Size

The *Batch Size* (BS) determines the number of samples sent to propagate through the network at once, during training. After each propagation, the weights of the NN are updated, and the next batch is sent through. A small BS usually leads to a network that trains faster, since the weights are updated more often. However, the network might miss out on some patterns that are only present over a large chunk of data. When all batches have been fed to the network, an *epoch* has passed.

An NN that is fed with all the data at once is fed with a *batch*. If the network is fed with anything less than the full data set (BS $\neq$ number of samples in training set), it is fed with a *mini-batch*.

2.5.1.3 Training the Network and Updating its Weights

Training a network is done by successively sending data through the network, calculating a model error, and then using that model error to update the network weights backwards, layer by layer. This process is called *back propagation*, and it is always performed when training deep neural networks. To calculate the model error, one uses an *objective function*, a function which determines how good a set of weights are based on some optimization criteria. The error is then used to calculate the new weights. The choice of objective function can determine if the training will take seconds, hours, or weeks, depending on the efficiency of the function. This choice is also highly dependent on the type problem the network should solve. The most common way of updating the weights is using an optimization function based on *gradient descent* [Zhang, 2019]. For a CNN, updating the weights corresponds to updating the values inside the convolutional kernels, as illustrated in Figure 2.23.

The amount of data used for the updates is adaptable and it can determine how accurate or fast each update is. Every update has a high accuracy if many data points are used, but a larger amount of data points makes the update slower. Fluctuations may occur when iterating though few data points, but the process is overall faster than iterating through all data points in a batch. When optimizing using gradient descent, it is not the absolute error that used to update the weights, but rather its derivative. The weights are updated in the direction of the gradient, to find the

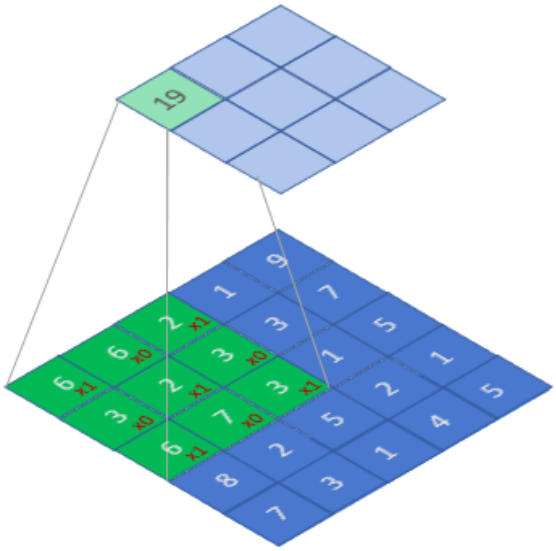


Figure 2.23: The figure illustrates how the convolutional kernel works. The darker numbers are the weights of the matrix, which are all multiplied with their corresponding number. The size of the green area is equal to the kernel size.

weights that give rise to the optimal value of the objective function. When updating using one data point,

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}} \tag{2.34}$$

is added to the weights, where $w_{i,j}$ is a weight in the network, E is the error and η is a constant which determines how much the weights should update on every iteration, denoted the *learning rate*. The learning rate influences how quickly a model adapts to the training set. Choose it too small and the model might get stuck at a local minimum since the steps taken to minimize the function are too small. But if the learning rate is chosen it too large the model might never converge since it passes the minimum with the large steps. There are several methods that use an adaptive learning rate to remedy the choice of the learning rate. The weights of the network are updated until the gradient is sufficiently small.

2.5.1.4 Over- and Underfitting

While it sounds like a good idea to increase the number of hidden layers and parameters to get a more accurate network, it can lead to overfitting, i.e., making the network fit the training data too well. This will lead to the network not being able to generalize to validation, test, or real data and only work well on training data. Overfitting can be detected if the network produces a model that handles training data well, but performs poorly on new data [O'Shea and Nash, 2015].

The problem of overfitting can be solved by reducing the complexity of the network, e.g., the number of layers and parameters. Then, the model will not be as specialized on the training set. A larger training set can also reduce the risk of overfitting since it is more likely that a large training set will not contain specific characteristics that the validation, test and real data do not have [Lavrenko and Goddard, 2016b].

2.5.1.5 Batch Normalization

Batch normalization is a technique for improving the speed, stability and performance of a neural network. By normalizing the mini-batches sent to each layer to have zero mean and unit variance, the training speed increased. Batch normalization enabled the choice of a larger learning rate, which made the training go faster. Another advantage is that the initialized parameters do not have to be chosen carefully due to batch normalization [Ioffe and Szegedy, 2015].

2.6 Previous Research

Noise reduction using CNNs on automotive radar signals has been made with good result, where object peaks have been preserved and the SNR has increased significantly, in [Rock et al., 2019b]. CNNs provided a larger increase in SNR compared to conventional signal processing methods. However, the CNN caused distortion of the peak values which obstructs further processing, e.g., angular estimation or object classification. Simulated data was used in this project with the maximum number of interfering radars being three.

In [Rock et al., 2019a] CNNs were used for automotive radar interference mitigation on real data, with added simulated interference. By comparing SINR-values, the CNN outperforms classical mitigation methods. The SINR-value of the predicted RD-map is even greater than the SINR-value of the non-interfered RD-map, which means that real world noise have been removed as well, and not only interference.

In [Ristea et al., 2020], interference mitigation was also performed on automotive radar signals with neural networks. Here, fully Convolutional Networks (FCNs) were used instead of CNNs. More information about FCNs can be found in, e.g., [Long et al., 2015]. In [Ristea et al., 2020], two types of FCNs were trained; one shallow network with fewer layers and convolution blocks, and one deeper network with more layers and convolution blocks. The input to the networks consisted of a spectrogram of the baseband signal, with the output being a range profile, which gives the same information as a Range-FFT, i.e., at which distance an object is present. The result from both of these networks were compared to a baseline zeroing algorithm. When evaluating the results, the deep FCN outperformed the other algorithms and almost performed as good as an oracle based on the true labels. The shallow network outperformed the baseline for most of the evaluation metrics. These tests were performed with simulated data with one interfering radar being present. The deep FCN also generated promising results when tested on real data.

In [Akeret et al., 2017], a special type of CNNs have also been used for classification purposes in the radio frequency domain. The algorithm achieved excellent result on simulated data where the true object locations were known. When training the model on real data, the algorithm performed slightly worse, but it can be due to the fact that the true object locations had to be estimated. Inaccurate estimation of true objects would make the network train on incorrect data and generate a worse model.

The phenomena clock drift has not been mentioned in any of the articles. Dividing the interference into categories to study the different types has not been done either. In this project, handling different types of interference, and especially handling coherent interference subject to clock drift contributes with novelty in this area. By both simulating data to make it resemble real-world data which especially contains interference from identical radars, and mitigating the interference, this project presents a more in-depth examination of interference mitigation.

Chapter 3

Methods

The following section will discuss the various approaches to the methods that were implemented for interference mitigation. Section 3.1 describes how the simulated data needed for the machine learning was created. Section 3.2 describes the different interference mitigation algorithms that were implemented, both the signal processing ones, as well as the CNNs. Finally, Section 3.3 describes the methods used to evaluate the results.

3.1 Generating Simulated Data

To properly evaluate different algorithms, information about object location and interference type needs to be known in advance. Then, it is possible to compare the prediction from each algorithm with the true values and determine which has the best result. Furthermore, the approach used to train the CNN required data marked either clean or interfered. Therefore, a simulation environment was created, where all the model parameters could be adjusted, and data could be generated. For every run, a few selected parameters were randomized from a uniform distribution. All of the randomly selected parameters used in the simulation environment can be seen in Table 3.1, along with the maximum and minimum values the parameters could have. The full bandwidth was always used when generating both the victim and the aggressor, hence all of the generated radar signals chirped up over the same bandwidth B. The assumption of all radars using the full bandwidth is commonly occurring for applications where radars are restricted to narrow frequency intervals due to regulatory requirements. The slope of the frequency over time was then only determined by how long the signal was. A shorter signal had a steeper slope as it had a shorter amount of time to go from the lowest to the highest frequency. This is visualized in Figure 2.16, where the victim signal is lasting over a shorter period of time than the aggressor and therefore has a steeper slope.

To make it as hard as possible for the CNNs to distinguish ghost objects from true objects, the coherent interference was generated to make ghost objects mimic true objects. The interference amplitude was chosen to always be equal to the victim amplitude, to assure that the energy of the ghost objects was the same as the energy of the true objects. When coherent interference was present it lasted for every chirp in a frame. This was also done in order to mimic true objects, since true objects are always present in all chirps, and not only in some of them.

To keep track of where the true objects were located, object masks were generated. These masks stored information about where in the different visualization maps the objects occurred. The strength of the signal was not shown in the masks, but only the object positions. An area of 3x3 data points marked the position of each object. Noise masks were also generated, which kept track of where objects were not present. The two masks can be seen as inverses of each other.

Table 3.1: Parameters randomized for every simulated frame.

Parameter Name	Minimum Value	Maximum Value
Object Parameters		
No. Interfering Radars	1	3
Number of Objects	1	20
Object Distance	0	d_{max}
Object Velocity	-$v_{max}/2$	$v_{max}/2$
Object AoA	-90°	90°
Non-Coherent Interference		
Interference Chirp Time	**1.2**·Victim Chirp Time	**1.8**·Victim Chirp Time
Interference Idle Time	**1.2**·Victim Idle Time	**1.8**·Victim Idle Time
Interference Amplitude	**0.1**·Amplitude	**199.9**·Victim Amplitude
Semi-Coherent Interference		
Interference Chirp Time	**0.98**·Victim Chirp Time	**1.02**·Victim Chirp Time
Interference Idle Time	**0.98**·Victim Idle Time	**1.02**·Victim Idle Time
Interference Amplitude	**0.1**·Victim Amplitude	**199.9**·Victim Amplitude
Coherent Interference		
Clock Drift	**-20** PPM	**20** PPM
Interference Amplitude	**1**·Victim Amplitude	**1**·Victim Amplitude

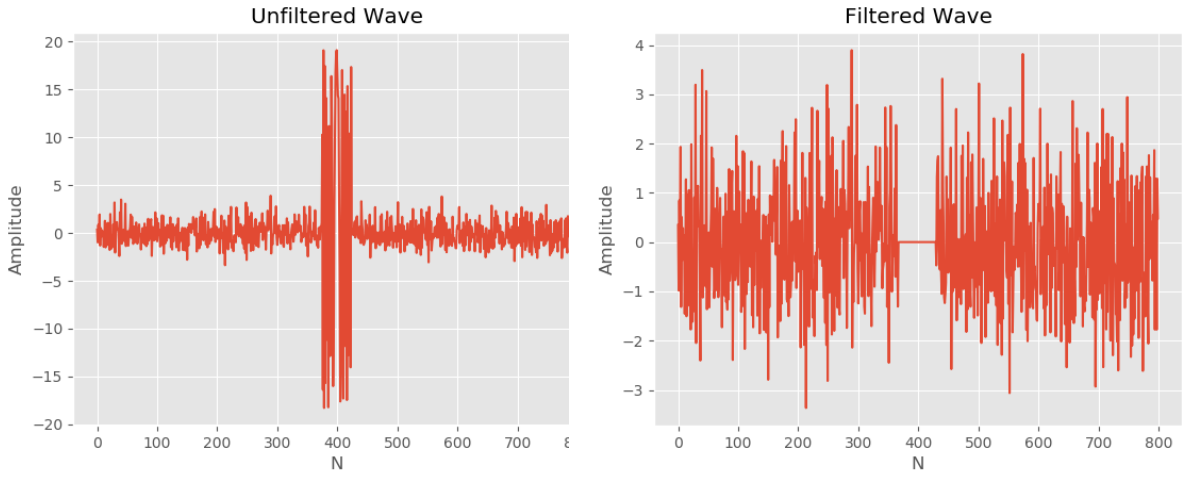
The data in the simulation was generated according to the signal model presented in Section 2.2. White Gaussian noise was also added, to resemble real-world data. Interference mitigation was performed for each of the three disturbance types. The training data fed to the CNN consisted of all three types of disturbances. For each frame, a disturbance type was randomly selected, as well as a starting time and an end time for the interference. There was an equal probability of generating non-coherent, semi-coherent and coherent interference for every aggressor radar. So one interfering radar generated one type of disturbance that lasted for a random amount of time within a frame, except for the coherent interference, which lasted for all chirps in a frame. Only the interference with frequencies present within the IF-band was taken into account and further processed, because frequencies outside the IF-band were larger than the maximum detectable frequency.

When performing the three FFTs for estimating the range, velocity and AoA of the objects, zero padding was only performed in the angle-direction. In real radars, the amount of chirps and the amount of samples per chirp is usually a lot greater than the number of receiving antennas, meaning that zero padding is most useful in this direction. A Hann window is used in all FFTs.

The simulated data was compared to real-world data collected with a radar, to determine if the simulation could be generalized to handle real problems. The comparison was only performed qualitatively, by regarding the RD- and RA-maps. The real-world data was raw data collected from a commercial radar system. The aim was to create the simulated data to resemble real data. Modelling how objects reflect radar signals was however very tricky, and depends on a lot of parameters, such as distance to the radar, object area, and if the object has both stationary and moving parts. From real-data observations, objects had a very similar signal strength and a decision was made to train networks primarily for the case where amplitudes remained constant. Networks were however trained for the case of varying amplitude as well, just to make sure that the amplitude being constant was not the sole reason that the network could learn.

3.2 Interference Mitigation

When performing interference mitigation, several different algorithms were used and compared. These included basic zeroing algorithms, both perfect and non-perfect, non-mutual data cancellation and CNNs. The non-perfect zeroing algorithm was constructed by us, from scratch. The other algorithms were implemented by us, but the idea behind the algorithms came from different articles. The aim was to first and foremost detect in which chirps and samples interference was occurring. The second step was to eliminate the interference from these samples without disrupting the rest of the signal and without causing new discrepancies. Figure 3.1 shows data before and after zeroing has been performed.



(a) Signal before the zeroing algorithm. *(b) Signal after the zeroing algorithm.*

Figure 3.1: The idea of the zeroing algorithm. The data subject to interference is zeroed out. Note the different scale in the figures.

3.2.1 Zeroing

Zeroing is the most straightforward interference mitigation technique, which simply sets all the interfered data points to zero. By doing this, only data that was not subject to interference was further processed with FFTs and the rest of the data processing chain. Due to the simple, yet intuitive nature of the algorithm, it is often used as a minimum requirement that any given implemented algorithm must outperform, i.e., a *baseline* algorithm. In this report both perfect zeroer, i.e., a zeroing algorithm that knows about the exact location of interference in advance, and a non-perfect one, i.e., an algorithm that tries to find interfered samples, have been implemented.

3.2.1.1 Non-Perfect Zeroing Algorithm

Although zeroing is rather straightforward in theory, in practice it might prove difficult to accurately pinpoint exactly where the data has been interfered, and where to cut the signal. A common method is to create a threshold line perpendicular to the y-axis, and remove all data above said line. This was done when creating the non-perfect zeroing algorithm. The problem then evolved to choosing the threshold. Some alternatives involved regarding a combination of the mean and the median of the signal. Then it was assumed that the interference was distinguishable from the true signal, e.g., with a higher amplitude of the signal wave. However, if the signal was not interfered, there would be be unwanted losses of real signal because a threshold would be set regardless of presence of interference. It was therefore essential to figure out if the signal contained interference, as well as figuring out where the threshold would be put to determine where in the signal the interference was present.

Classifying Interfered Data

A simple, yet effective way of checking for interference in data was to look at the maximum amplitude value of the data in a chirp. If the maximum value was greater than some value, ξ, then the chirp was marked as interfered, otherwise it was marked as non-interfered. This method was useful when the amplitude of the interference was substantially greater than the amplitude of the non-interfered signal. A good starting point for deciding the value of ξ would be to set it to the mean or the median of the data times some constant β. Since interference disturbed a minority of the samples in a data set, the median was more robust than the mean, because the amplitude of the interfering wave had a large impact on the mean. ξ was therefore chosen to be

$$\xi = \beta \cdot x_{med}, \tag{3.1}$$

where x_{med} was the median of the input data and β was empirically determined by studying which threshold level that detected most of the interference without classifying non-interfered data as interference.

Choosing the Threshold

If the data was classified as interfered, a threshold was designed, where the idea was that every data point above the threshold should be discarded. The absolute

value was taken on the data fed into the algorithm, and from here, it was possible to study the cumulative sum of data points. This was done by dividing the y-axis into discretized blocks (the red dashed lines in Figure 3.3a), and then adding up the cumulative number of data points (samples) present above each line. The cumulative sum was calculated by starting from the highest dashed line and going downwards. The first bin therefore contained the number of data points between the top-most red line and the second one from the top. The second bin contained the data points between the top-most red line, and the third line from the top, and so on. Figure 3.2 depicts the cumulative sum of points against bin number and the figure has a distinct, arched look for data with interference. It drastically increases at the end of the plot. This is due to the fact that true data points had an amplitude similar to each other while the interfered data points had a much greater amplitude in this data set.

By finding the maximum value of the second derivative of the cumulative sum, also known as the *knee point*, the optimal location for the threshold was found. This trend seemed to be general, and it was not unexpected. If interference was present, there would only be some data points in the low-numbered threshold bins, but the majority of the points would be at the bottom where the desired signal lied. This was true even for chirps with a great amount of interference, since the sample points of the interfered waves were spread out over a larger number of bins while the true sample points were concentrated to only a few bins. A larger spread meant that it was less likely that many data points ended up in the same bin. When one encountered the discretized region where the true signal lied, the cumulative number of points drastically increased, meaning that the threshold value should be set right before said threshold value, see Figure 3.2.

To avoid having non-interfered outlier points in the data zeroed out by the threshold, the algorithm only zeroed out the data if more than 3 points in an interval of 15 were above the threshold. The horizontal bins that the y-axis was divided into can be seen in Figure 3.3, along with a figure of the chosen threshold for this data set.

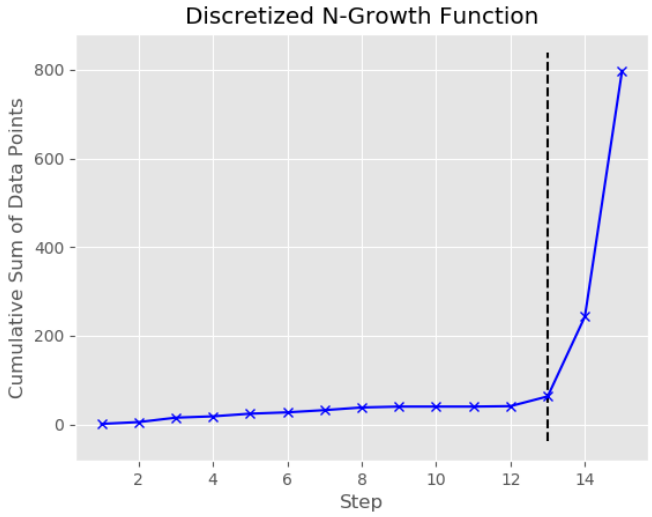
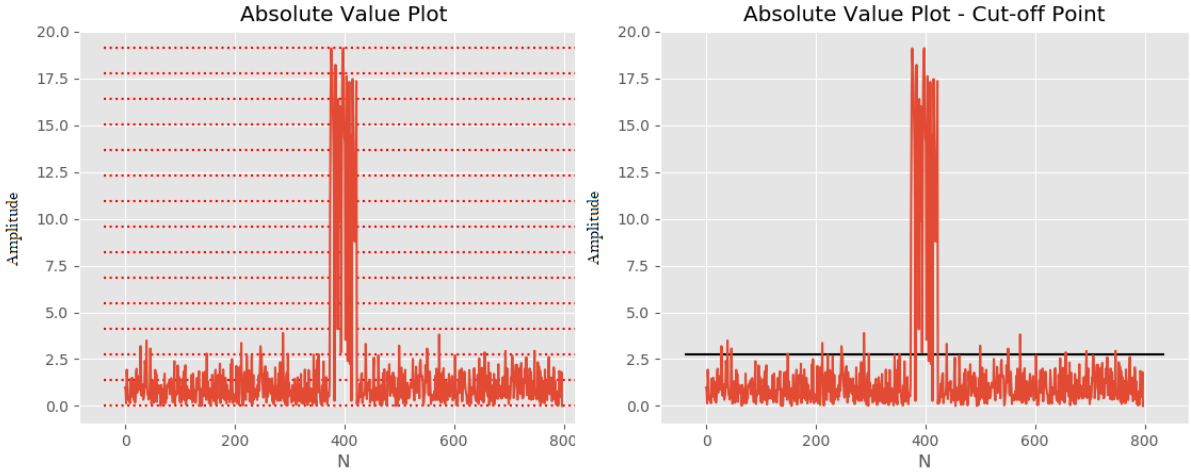


Figure 3.2: Knee point for a cumulative sum for an interfered waveform, where the bin number (13) indicates which bin in the discretized domain one should choose as threshold.



(a) Plot with all the bins (red-dashed lines).

(b) Choosing the bin number indicated in Figure 3.2 yields the threshold, marked with a black line.

Figure 3.3: Plots showing the discretized blocks and the chosen threshold for this data set.

3.2.2 Non-Mutual Data Cancellation

Non-Mutual Data Cancellation (NMDC) is a mitigation technique where one exploits the predictable magnitude response from objects in the frequency domain. The technique is thoroughly described in [Wagner et al., 2018]. In short, by assuming that an object was present in all chirps in a frame, a non-linear operator, e.g., the *min*-operator could be used over every sample to suppress data that was not consistently present in all chirps. The idea was that every chirp had information about an object, but only some chirps were affected by interference. By only extracting the information about the object and discarding everything else, the aim was to eliminate the interference while not disrupting data about the object. An illustration of the algorithm is presented in Figure 3.4.

The data inserted in the algorithm had been transformed with an FFT and was consequently in the frequency domain. Initially, the data was split into amplitude and phase, and only the amplitude was regarded. The *min*-operator operated on the first sample of every chirp and returned the smallest amplitude value among these. Then it operated on the second sample of every chirp, and this procedure continued until all samples in all chirps had been regarded. The data that was not one of the smallest values was discarded. The phase remained unchanged. Then, the smallest amplitude values were stacked into a matrix of the same size as the input data. The last step was to combine the new amplitude data with the original phase data. It was assumed that the amplitude of the interference was greater than the amplitude of the non-interfered data, and would therefore never be included in the result from the operator. So, only a small amount of data was used for further processing.

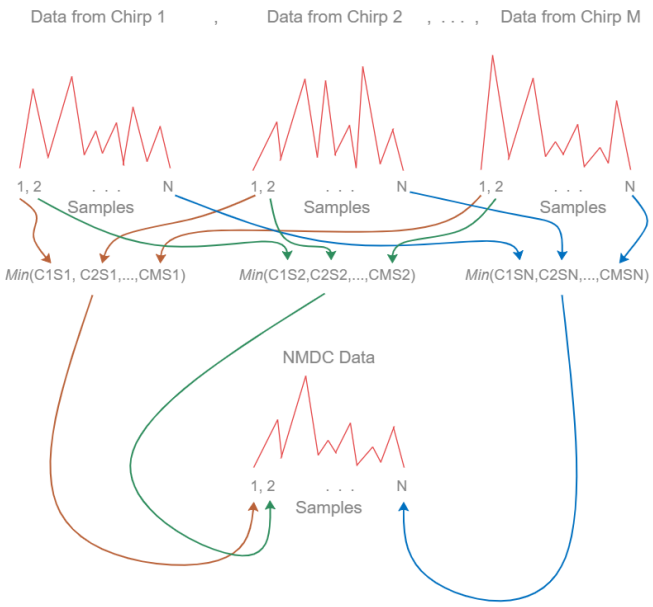


Figure 3.4: Illustration of how the NMDC algorithm works. Chirps are denoted C and samples are denoted S.

3.2.3 Convolutional Neural Networks

3.2.3.1 Inputs to the Neural Network

Generally, one wants to give the network data that is as relevant as possible for the task. Therefore, if the task is to denoise RD-images, the input layer of the network is fed with a training set consisting of RD-maps with interference. For the larger training runs, the input to the network consisted of a data set containing 500 independently created frames (meaning that a new type of interference was created for every frame), and was validated against 75 exclusively created frames for validation. Finally, the model was evaluated against a test set, also containing 75 exclusive frames.

The input to the network was specifically in the form of complex matrices with a certain size. More data was generated for the 2D-maps than for the 3D-maps due to the limited computational power. The parameters used for generating the 2D-maps are presented below. The amount of samples per chirp was set to 800. But due to mirroring in the FFT, half of these were discarded, so the number of samples per chirp used in the RD-map, N, was 400. The amount of chirps, M, was set to 256. The amount of antennas was set to 8, but zero padding was performed to increase the number of bins in the angular direction, so the final amount of angle bins, K, was set to 400.

When generating the RDA-cube that was used for training, 100 samples per chirps were used. Since half of the samples were discarded, N was equal to 50. M was set to 32 and 8 antennas were used, but since zero padding occurred here, the final amount of angle bins, K, was set to 50. Table 3.2 summarizes all the input sizes used.

Table 3.2: Input sizes for the different maps. The numbers in brackets are what remains after the first FFT, where half of the samples are discarded.

Sample Type	RD	RA	RDA
Samples per chirp	800 (400)	800 (400)	100 (50)
Number of chirps	256	0	32
Number of antenna bins	0	400	50

3.2.3.2 Preprocessing of the Data

Before the data was sent to the network, it was subject to standardization in order to increase the training speed. This was done using a standard zero-mean method which scale the mean of the data to zero, and its standard deviation to one, according to

$$\tilde{\mathbf{x}} = \frac{\mathbf{x} - \bar{x}}{\sigma}, \tag{3.2}$$

where σ is the standard deviation, and $\tilde{\mathbf{x}}$ denotes the scaled data.

3.2.3.3 Network Hyper-Parameters

Neural networks generally have a lot of tuneable parameters, such as kernel size, the number of hidden layers, learning rate etc. An idea would be to just try all the possible parameter configurations, called *grid searching*. However, grid searching as a way of finding suitable network structures might be flawed, since networks of higher complexity more often suffer from over-fitting, hence requiring extra precautions such as regularization to overcome the problem [Bishop, 2006]. However, since investigating every combination imaginable was, and usually is, unfeasible, grid search was used as an indicator of what might be a good starting point. An idea of what the most suitable model for the data would be was found in [Rock et al., 2019b], where a grid search had been performed for mitigation of automotive radar data. Some of the combinations they found most suitable was used in this thesis, see Table 3.3.

Table 3.3: CNN parameters for the different models.

Model Name	Kernel Size	Layers	Kernels	Parameters
Shallow 2D	(3x3)	4	2	160
Deep 2D	(3x3)	6	16	10002
Shallow 3D	(3x3x3)	4	2	452
Deep 3D	(3x3x3)	6	16	35680

It might seem like a good idea to have as many parameters as possible in the network, to train the model as accurately as possible. To do this, the number of layers and the number of units per layer could be increased. However, there are several drawbacks associated with a large amount of parameters. First of all, the computational complexity increases drastically as the number of parameters increases meaning that training the network would take time, a long time. The low amount

of parameters is one of the reasons for the choice of using a CNN instead of a different type of NN because the computational resources available could not handle a massive network. The sparsity of a CNN reduces the amount of parameters so less resources are needed.

Another reason for using fewer parameters is to avoid overfitting to the training data. A more complex system finds patterns in the training data that might not occur in the validation- or test data. It is not desirable to have a system that performs exceptionally well on data it has been trained on.

3.2.3.4 Training the Network

A schematic of the network structure used in this project is presented in Figure 3.5. The learning rate for the network, η, was set to 0.00005. The BS for the training data was two, which means that two frames were inserted in the network at a time. The chosen optimization algorithm used in the network was *Adaptive Moment Estimation* (Adam), which used adaptive learning rates, η. The learning rates were adapted by regarding the previous gradients in the optimization process. By using

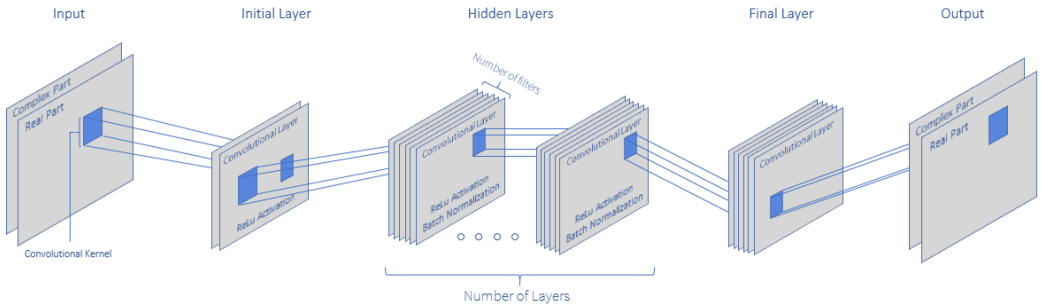


Figure 3.5: The architecture of the CNNs that were used.

previous gradients, faster convergence is enabled, compared to only using the gradient for the current data point [Zhang, 2019]. Also, the risk of getting stuck in a local minimum is reduced. The hyper-parameters β_1 and β_2, which determine how important the previously calculated gradients are when updating the weights, were set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. These specific values had shown good results in [Kingma and Ba, 2014], where more information about Adam can be found.

Each hidden layer consisted of performing the convolution operation, then performing batch normalization on the data and later passing the data to the activation function. The network was trained and evaluated against input data which only contained the signal from the objects, without noise or interference, see Figure 2.11a for an example of this type of data. The result of this was that the network was rewarded for suppressing noise, since noise was not present in the target data. However, this means that the network trained against objects that were subject to leakage, and as a result were smeared. For every batch, the loss was printed, and if the loss did not improve for 50 epochs the program stopped. At the end of each run, the losses were plotted in order to see if the CNN had converged.

3.2.3.5 Objective Functions

SINR

The SINR function calculated the SINR using the predicted map from the model, together with the noise- and object masks from the training set, i.e, the true positions of the objects. By using a noise- and object mask where the location for the object and noise were stored, a SINR estimation was calculated using

$$\text{SINR} = 10 \log\left(\frac{\frac{1}{N_O}\sum_{\{n,m\}\in O}|\tilde{S}_{RD}[n,m]|^2}{\frac{1}{N_N}\sum_{\{n,m\}\in N}|\tilde{S}_{RD}[n,m]|^2}\right), \tag{3.3}$$

where N_O is the number of objects in the object mask, N_N is the number of data points in the noise mask, O denotes where objects are present, N denotes where noise is present and $\tilde{S}_{RD}$ is the predicted RD-map. Since the SINR only increases significantly when the predicted object lies within the true object mask, location precision was enforced. If an algorithm would decrease the SINR, it would indicate that the algorithm added noise to the data.

Mean Squared Error

The *Mean Squared Error (MSE)* loss measures the mean square error between each element in the input and the target according to

$$\text{MSE} = \frac{1}{n}\sum_{i}^{n}(x_n - y_n)^2. \tag{3.4}$$

In the equation, x_n is the prediction and y_n is the target. This objective function aims to rebuild the target map, which includes the leakage from the FFT:s. So instead of only regarding the object positions, which the SINR objective function did, MSE regarded the entire map. The aim with this objective function was to make the whole predicted map similar to the whole clean map, rather than just regarding the data points containing objects.

3.3 Evaluating the Result

The CNN-models were evaluated using a simulated test set, containing a mix of all types of interference. Each 2D-model was evaluated for both 2D-maps: the RD-map and the RA-map, to find out if training on one type of map could produce a network that could handle another type of map. Three evaluation metrics were chosen: SINR, EVM and ROC-curve, and all of them are explained below.

3.3.1 Signal-to-Interference-plus-Noise Ratio

SINR was evaluated using (3.3), and the metric is suitable for determining how well an algorithm has managed to enhance the object peaks relative noise and interference. Since the metric only increased when the algorithm predicted a peak at the correct location, it also worked as an objective function. A large SINR-value implicated that the signal strength was greater than the strength of the noise and the interference.

3.3.2 Error Vector Magnitude

The Error Vector Magnitude (EVM) is a metric defined as

$$\text{EVM} = \frac{1}{N_O} \sum_{\{n,m\} \in O} \frac{|S_{RD}[n,m] - \tilde{S}_{RD}[n,m]|}{|S_{RD}[n,m]|}, \tag{3.5}$$

where S_{RD} is the clean Range-Doppler map and $\tilde{S}_{RD}$ is the predicted Range-Doppler map and N_O is once again the number of objects. The result was summed over all object peaks, so during evaluation the predicted object peaks were extracted from the Range-Doppler map, and compared to the object peaks in the clean map. The object masks determined which data points were counted as objects. Only these data points were used in the EVM calculation. The input and the output of the network consisted of complex-valued data points and the normalized sum of the difference between the points is the EVM. EVM gave information about how close the magnitude of the predicted objects and the magnitude of the true objects were, i.e., how well the predicted object signal strength captured the signal strength of the true objects. EVM also gave information about the phase error of the prediction. The phase error determines how suited the data is for further processing, where a low phase error means that the prediction has not distorted the data. A low phase error in the RD-domain meant that the data could be used for AoA-estimation; and vise versa for the RA-domain, where the data could be used for velocity estimation. A small EVM meant that the predictions matched the target better, see Figure 3.6 for a visualization of the metric. In the figure, it can be seen that the metric can be divided into a magnitude error and a phase error.

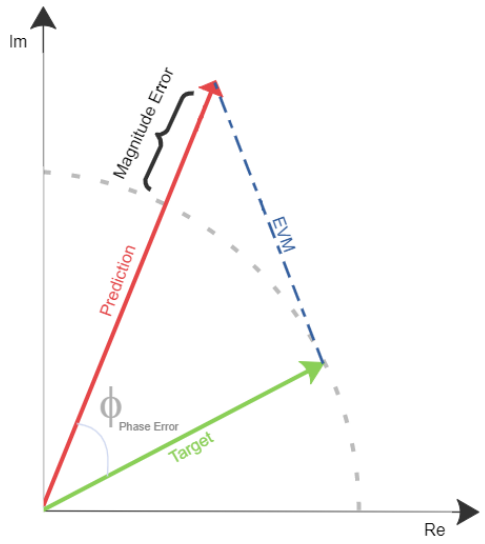


Figure 3.6: Definition of EVM, non-normalized.

3.3.3 Receiver Operating Characteristic Curve

The *Receiver Operating Characteristic* (ROC) curve evaluates a measurement based on its *True Positive Rate (TPR)*, plotted against its *False Positive Rate (FPR)* at various threshold settings. It evaluates how well a classifier labels the true objects as true ones and the interference as false objects. The threshold is initially set high, and then the classifier does not label anything as positive. Then, the threshold is gradually lowered until all of the data points are labeled as positive. If the predictions are perfect, the ROC will not have any false positives, represented by the green line in Figure 3.7. If the predictions are good, it is expected that the likelihood of finding true positives is higher than the likelihood of finding false positives, which gives the curve a concave appearance, see the blue line in the figure. If the predictions are completely random, the result is a straight line, represented by the red line in the plot. The ROC-curve for the predictions was evaluated as the mean for all frames' TPR and FPR in the test set. The ROC-curve is generated by iteratively building

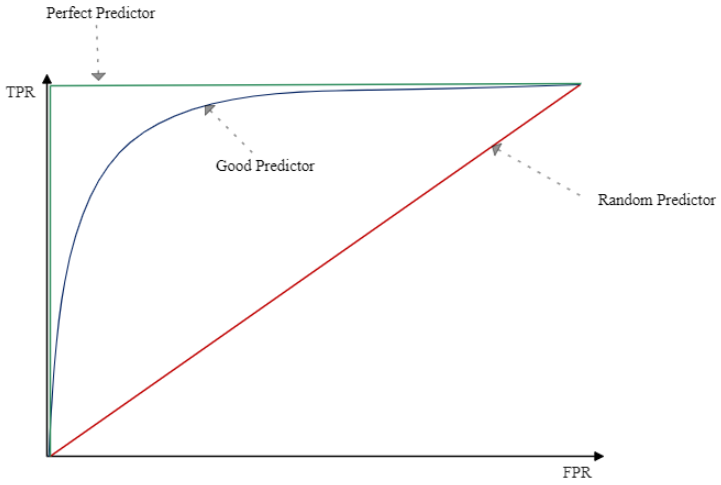


Figure 3.7: A ROC-curve, indicating how a perfect, good and random predictor is defined.

an object mask from the various algorithms, where all the objects above a certain threshold is included. When the true object mask is built, a 3x3 grid is built around the object in order to combat the smearing that is present, and the same is done when iteratively constructing the object mask from the algorithms.

Chapter 4

Results

Section 4.1 presents the quantitative results, i.e., the numerical values and a ROC-curve produced by the evaluation metrics presented in Section 3.3. Section 4.2 contains qualitative assessments of the results, which includes various visualization maps from the different algorithms. Here, maps of both real and simulated data are presented. Only RD- and RA-maps are displayed since these are the maps that contribute with most information about the performance of the algorithms.

4.1 Quantitative Evaluation Metrics

Table 4.1 shows the EVM and SINR metrics where the input to the algorithms consists of RD-maps. In Table 4.2 the inputs are instead RA-maps. Finally, in Table 4.3 the inputs to the algorithms are RDA-maps. The numbers in bold indicate where the highest SINR-values and the lowest EVM-values are produced.

In Table 4.1, it can be seen that when evaluating SINR on RD-maps (see the leftmost columns), the CNNs generate higher SINR-values than the other algorithms. The deep CNN trained on SINR generates a substantially greater SINR than the other algorithms, both for evaluation on RD-maps and on RA-maps. The EVM for the deep CNN on the RD-map is also very large. Overall, it looks like it is difficult to obtain both a high SINR and a low EVM. The deep CNN trained on MSE seems to be the model that generates a high SINR and a low EVM, when evaluated on the RD-maps. All CNNs generate an EVM greater than or equal to one when they are evaluated on RA-maps (see the rightmost columns). The only algorithm that increases the SINR without inflating the EVM is NMDC, when evaluated on RA-maps.

One observation that can be made by investigating Table 4.1 and Table 4.2 is that the performance of the CNNs is better when the model is evaluated with the same type of map as it has been trained on.

Table 4.1: Results from training on RD-inputs.

Algorithm	SINR RD	EVM RD	SINR RA	EVM RA
Non-interfered	39.8 dB	0.0180	17.9 dB	0.101
Interfered	25.9 dB	0.215	17.7 dB	0.101
True zeroing	32.5 dB	0.317	14.2 dB	**0.336**
Zeroing	21.3 dB	0.857	8.34 dB	0.922
NMDC	46.9 dB	0.521	18.7 dB	0.498
Deep CNN Models				
CNN SINR	**135 dB**	74.9	**62.2 dB**	1.01
CNN MSE	67.7 dB	**0.155**	18.9 dB	1.00
Shallow CNN Models				
CNN SINR	91.5 dB	6.56	0.599 dB	1.00
CNN MSE	56.4 dB	0.61	12.0 dB	1.00

By regarding Table 4.2, it can be seen that NMDC increases the SINR the most and the EVM is one of the lowest among the algorithms, when performing the evaluation on RD-maps. For RD-maps, no CNN has managed to outperform the interfered spectrum. When RA-maps are used for evaluation, the deep CNN trained on MSE generates the lowest EVM and the SINR is slightly increased.

Table 4.2: Results from training on RA-inputs.

Algorithm	SINR RD	EVM RD	SINR RA	EVM RA
Non-interfered	39.8 dB	0.0180	17.9 dB	0.101
Interfered	25.9 dB	0.215	17.7 dB	0.101
True zeroing	32.5 dB	**0.317**	14.2 dB	0.336
Zeroing	21.3dB	0.857	8.34 dB	0.922
NMDC	**46.9 dB**	0.521	18.7 dB	0.498
Deep CNN Model				
CNN SINR	18.1 dB	1.06	**28.5 dB**	0.993
CNN MSE	16.0 dB	0.997	18.7 dB	**0.0953**
Shallow CNN Model				
CNN SINR	24.1 dB	1.75	26.0 dB	1.02
CNN MSE	24.6 dB	2.08	18.6 dB	0.317

When analyzing Table 4.3, it is once again seen that the deep CNN trained on SINR generates the highest SINR, but also the highest EVM. NMDC, the shallow CNN trained on MSE and the deep CNN trained on MSE are the algorithms that increase the SINR without having an EVM greater than one.

Table 4.3: Results from training on RDA-inputs.

Algorithm	SINR RDA	EVM RDA
Non-interfered	23.2 dB	0.0680
Interfered	19.5 dB	0.258
True zeroing	19.3 dB	**0.358**
Zeroing	13.1 dB	0.868
NMDC	23.2 dB	0.674
Deep CNN Models		
CNN SINR	**47.8 dB**	8.92
CNN MSE	25.2 dB	0.742
Shallow CNN Models		
CNN SINR	39.3 dB	1.30
CNN MSE	24.4 dB	0.774

The ROC-curve for the CNNs and the signal processing algorithms is presented in Figure 4.1. A high TPR while maintaining a low FPR is desired. The non-perfect zeroing algorithm stands out from the rest as it starts to deviate early from the y-axis and from the other algorithms, meaning that it generated more false positives than the other ones. The deep CNN trained on SINR follows the y-axis the longest, so the amount of false positives is low. For this algorithm and for the shallow CNN trained on MSE, it takes a while to reach a TPR of one, i.e., classifying all true objects as true. The shallow CNN trained on SINR and the deep CNN trained on MSE follow each other and these are the models that classify all true objects as true first, since they reach a TPR of one before the other models. The perfect zeroing algorithm performs better than the non-perfect zeroing algorithm but worse than the other ones. NMDC has one of the steepest curves until a TPR of 0.9 is reached; afterwards the FPR increases more than before.

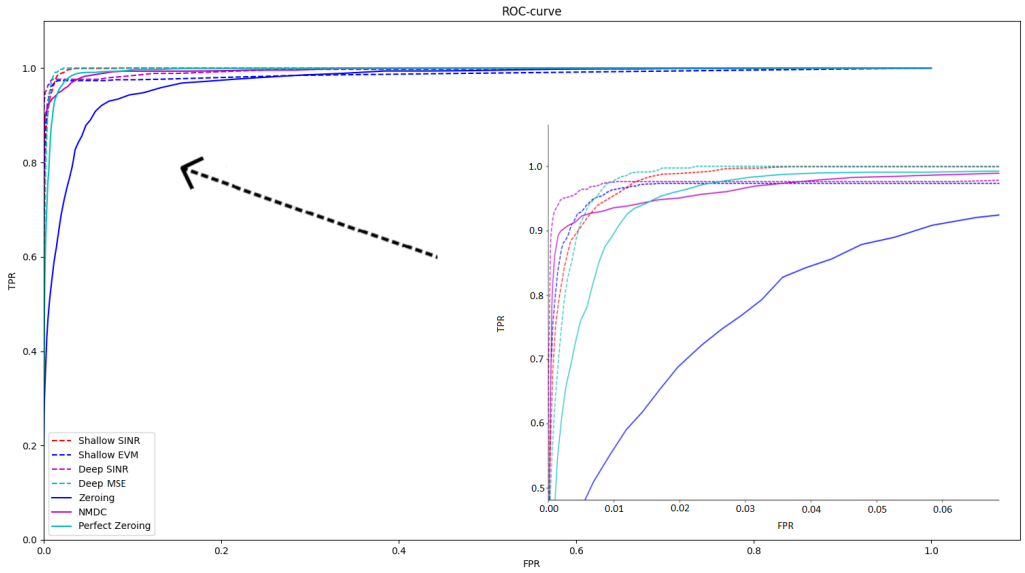
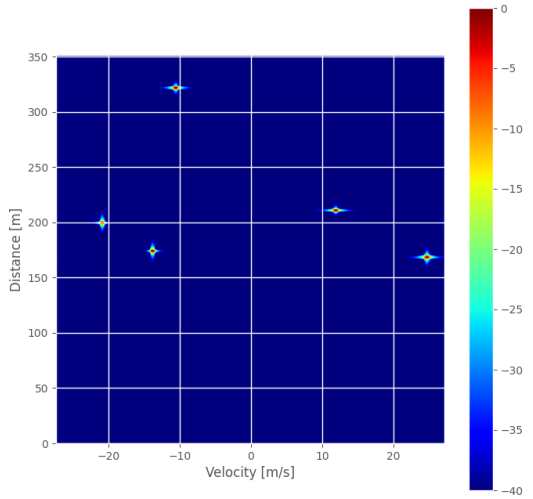


Figure 4.1: ROC-curve from the different algorithms with RD-inputs. The right box shows a zoomed in version of the curve.

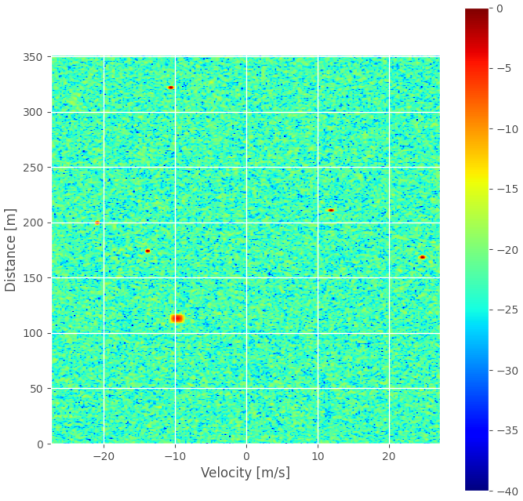
4.2 Qualitative Comparison of the Maps

RD-maps showing data subject to coherent interference are seen in Figure 4.2. Figure Figure 4.2a shows the clean RD-map. The interference is present in Figure 4.2b as a ghost object at (115, -10) in. It is a ghost object since it is not present in the clean RD-map in Figure 4.2a. NMDC manages to partially remove the interference while preserving the other objects and decreasing the noise floor, but some artefacts are present at (115, -10) in Figure 4.2c. Figure 4.2d shows the result after the perfect zeroing algorithm, which has not been able to make any predictions at all. The non-perfect zeroing in Figure 4.2e performs better, but the noise floor is higher than in the interfered RD-map and artefacts are present. The deep CNN trained on SINR manages to preserve all of the objects while removing the ghost object and decreasing the noise floor.

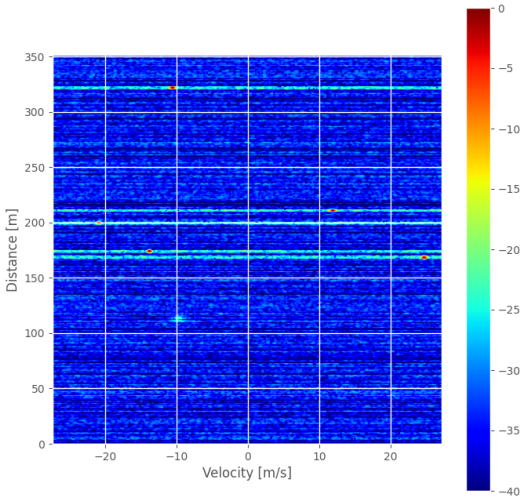
Figure 4.3 depicts the output from the algorithms with data subject to non-coherent or semi-coherent interference that lasts over the entire RD-map. Figure 4.3a shows data without interference and noise, and Figure 4.3b shows the data with both interference and noise. Both NMDC, Figure 4.3c, and the perfect zeroing algorithm, Figure 4.3d, manages to suppress the interference and reduce the noise floor while preserving the objects. However smearing in the velocity direction occurs in both figures. The non-perfect zeroing algorithm finds some objects but a lot is buried under the noise floor in Figure 4.3e. The deep CNN trained on MSE finds all true objects and reduces the noise floor more than the other algorithms, see Figure 4.3f.



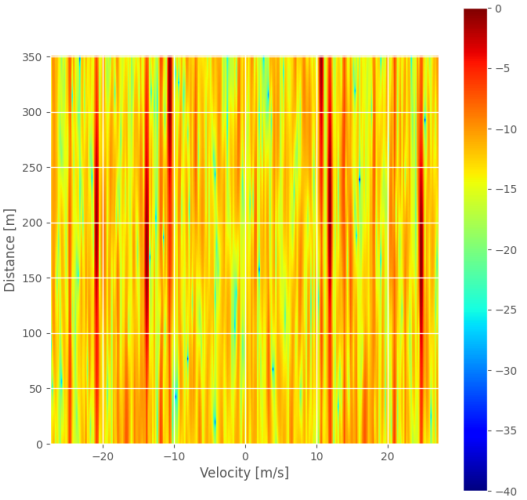
(a) The clean RD-map, without noise and interference.



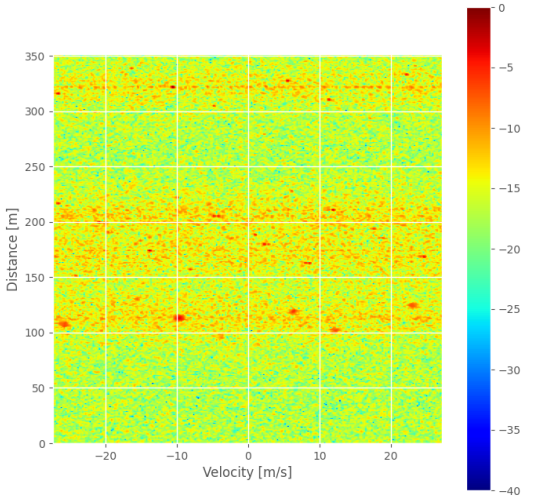
(b) RD-map with noise and interference.



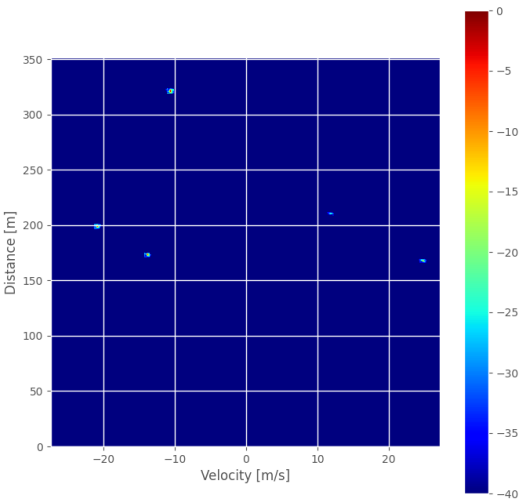
(c) RD-map after the NMDC algorithm.



(d) RD-map after perfect zeroing.

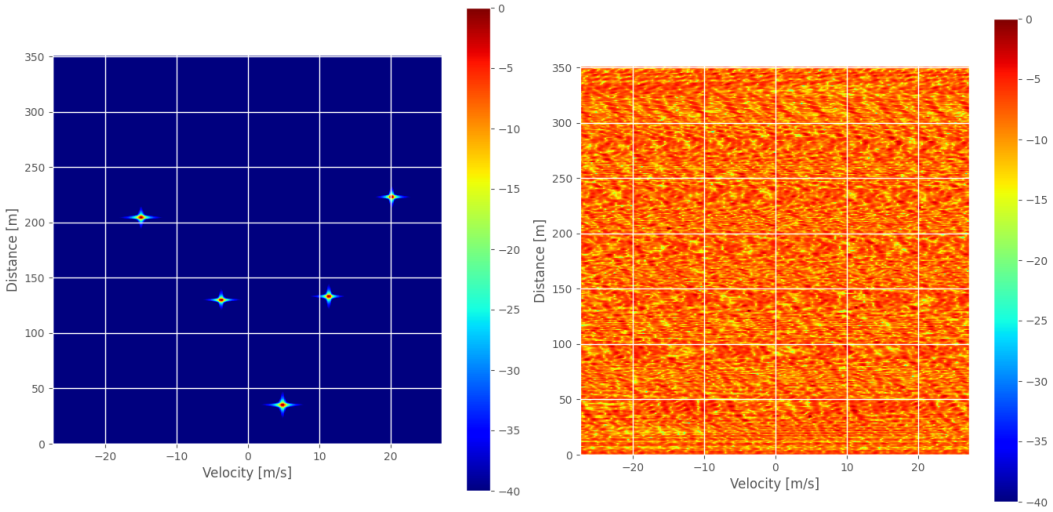


(e) RD-map after non-perfect zeroing.

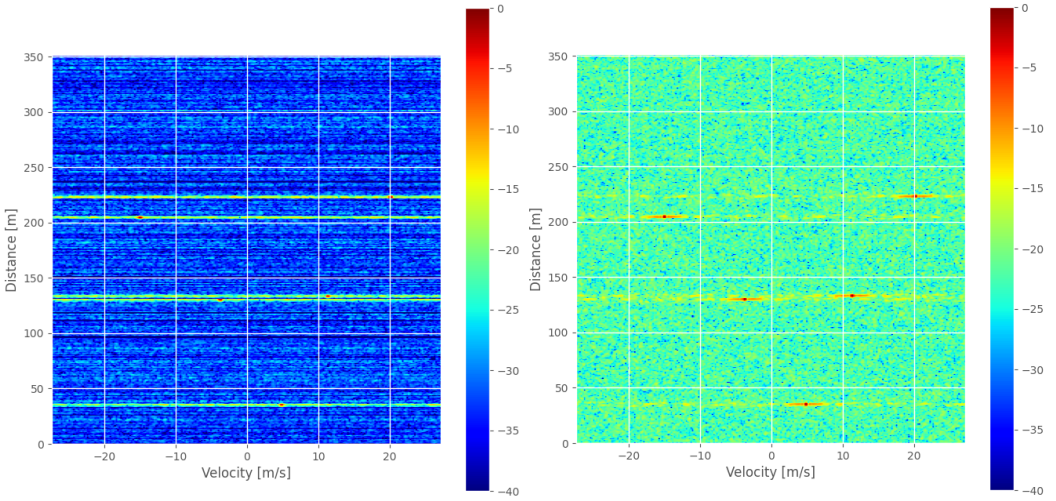


(f) RD-map after the deep SINR-based CNN.

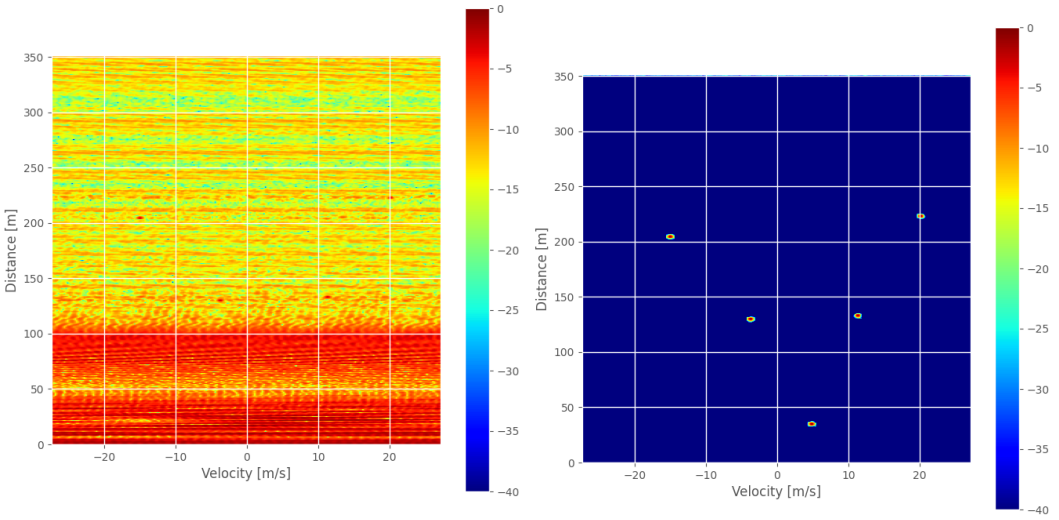
Figure 4.2: Performance comparison of the algorithms for a coherent interference.



(a) The clean RD-map, without noise and interference. *(b) RD-map with interference and noise.*



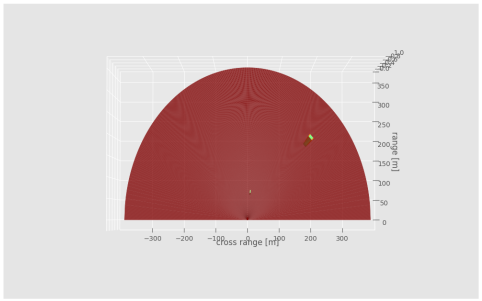
(c) RD-map after NMDC. *(d) RD-map after perfect zeroing.*



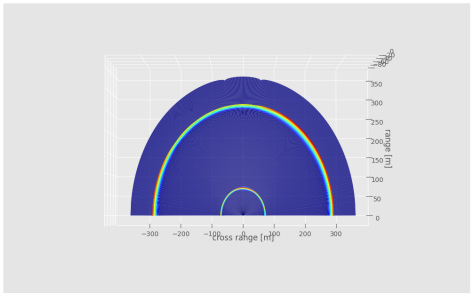
(e) RD-map after non-perfect zeroing. *(f) RD-map after deep CNN trained on MSE.*

Figure 4.3: Performance comparison of the algorithms for non/semi-coherent interference.

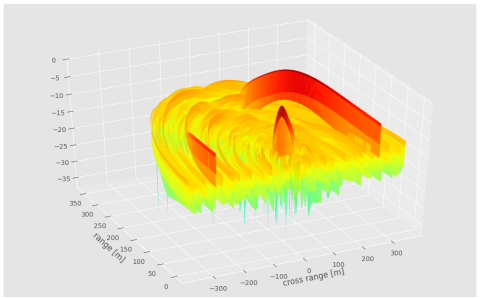
Figure 4.4 shows RA-maps from the different algorithms, where the data has been subject to non-coherent or semi-coherent interference over the entire map. The two bright points in Figure 4.4a shows where two objects are located and Figure 4.4b shows the RA-map without noise or interference. Here, object smearing in the angular direction is present since the objects are not present at only one or a few data points. Figure 4.4c shows the same data as the previous figures, but with noise and interference. The deep CNN trained on MSE manages to reduce the noise floor the most but the object smearing is still present, see Figure 4.4e. NMDC, in Figure 4.4d, and the deep CNN trained on SINR, in Figure 4.4f has approximately the same noise floor level but the CNN generates more distinct object peaks.



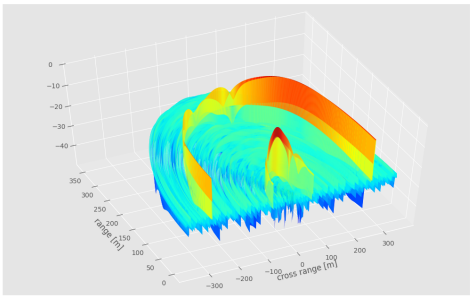
(a) The RA-map noise mask.



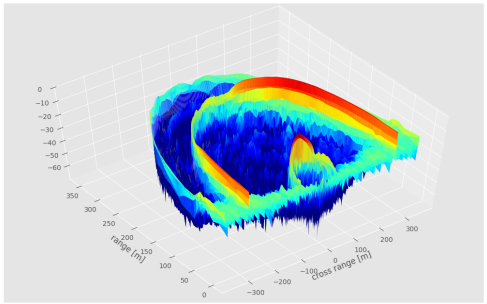
(b) The clean RA-map, without noise and interference.



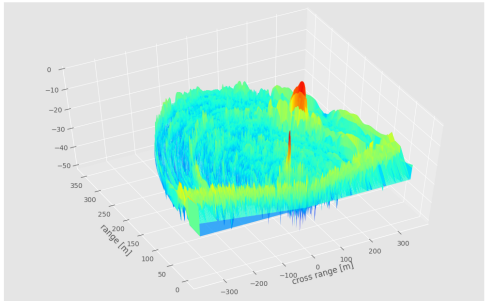
(c) RA-map with noise and interference.



(d) RA-map after NMDC.



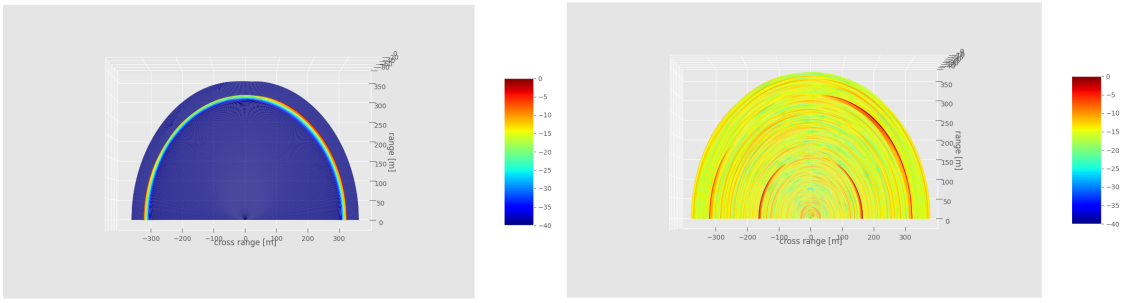
(e) RA-map after the deep MSE-CNN.



(f) RA-map after the deep SINR-CNN.

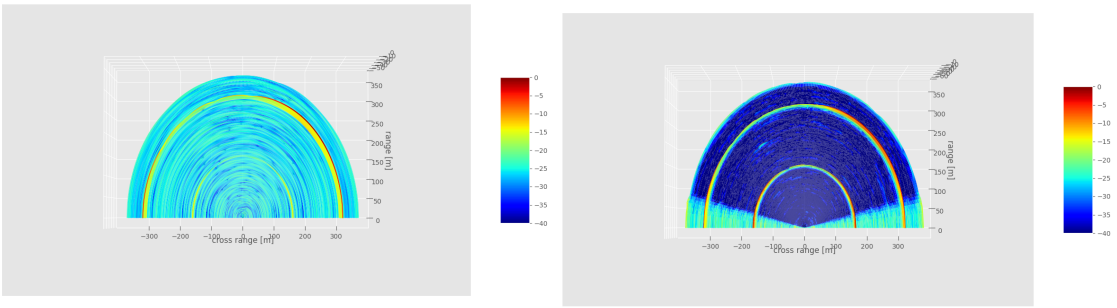
Figure 4.4: Performance comparison of the algorithms in the RA-domain. Here, non-coherent interference is present in the form of an elevated noise floor.

Figure 4.5 shows RA-outputs where the data has been subject to coherent interference. The clean RA-map in Figure 4.5a only shows one object but the interfered RA-map in Figure 4.5b contains two objects. The object that is not present in the clean RA-map is a ghost object due to the coherent interference. Neither the NMDC-model, in Figure 4.5c nor the CNN-model, in Figure 4.5d, manages to mitigate the ghost object present approximately 150 m from the radar.



(a) The clean RA-map, without noise and interference.

(b) RA-map with interference and noise.

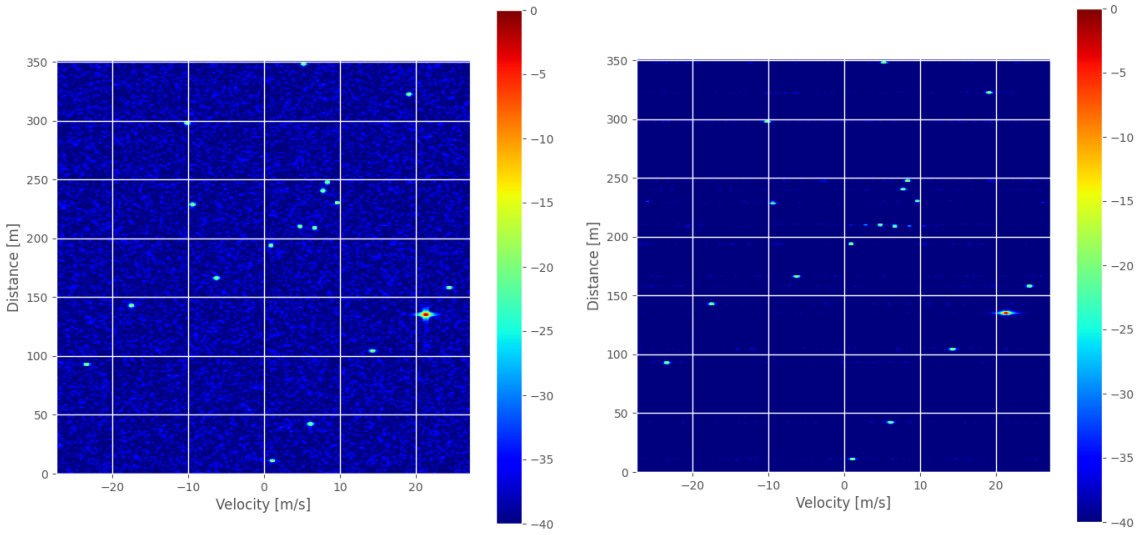


(c) RA-map after NMDC.

(d) RA-map after the deep CNN trained on MSE.

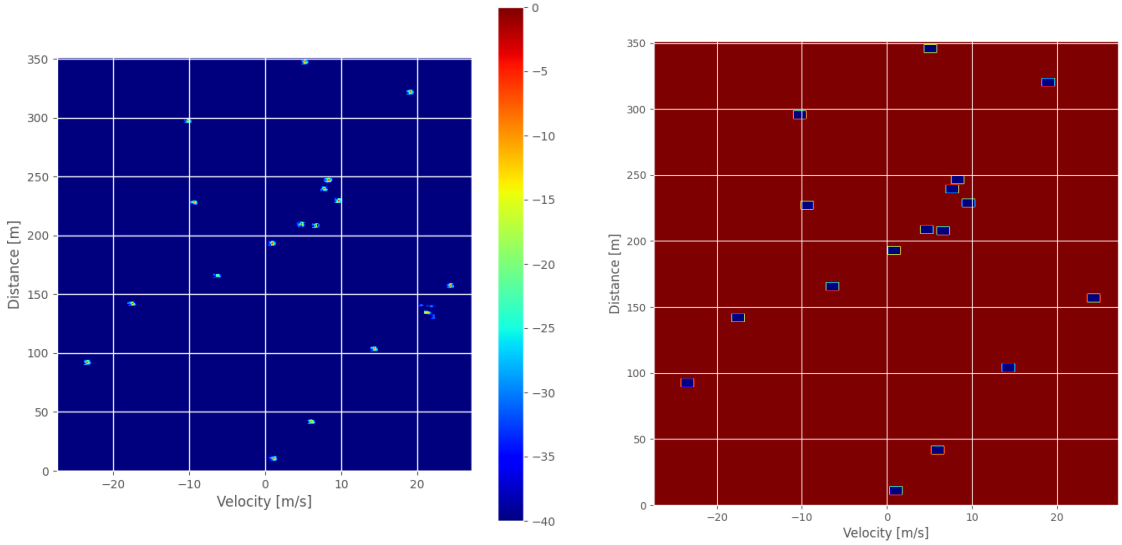
Figure 4.5: A typical result in the RA-domain for the different algorithms. Coherent interference is present as a ghost object at a range of approximately 150 m from the radar.

Figure 4.6 shows the result after evaluating test data containing coherent interference with an amplitude greater than the victim signal. Interference mitigation has been performed on coherent interference with a small amount of clock drift. By comparing Figure 4.6d with Figure 4.6a, the interfered map contains an object at (135, 22), which is not occurring in the noise mask, hence it is a ghost object generated by the interference. The ghost object has not been successfully removed by any of the algorithms. The deep CNN trained on SINR in Figure 4.6c has some artifacts at that coordinate, but it performs slightly better than NMDC in Figure 4.6b, which contains a brighter ghost object and more artifacts in the background.



(a) An RD-map containing noise and coherent interference.

(b) RD-map after NMDC.

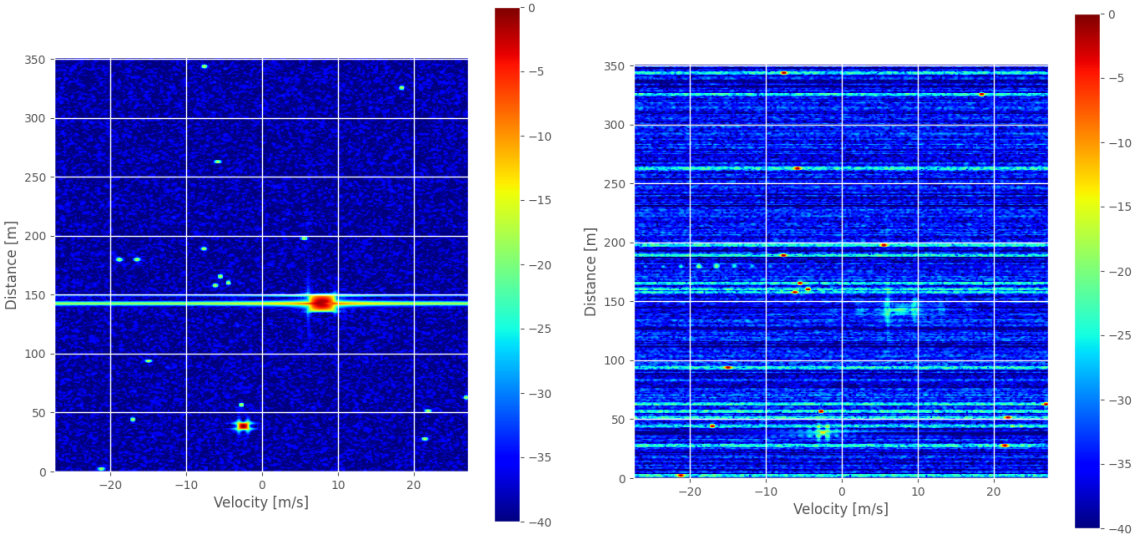


(c) RD-map after the deep CNN trained on SINR.

(d) Noise mask for the data.

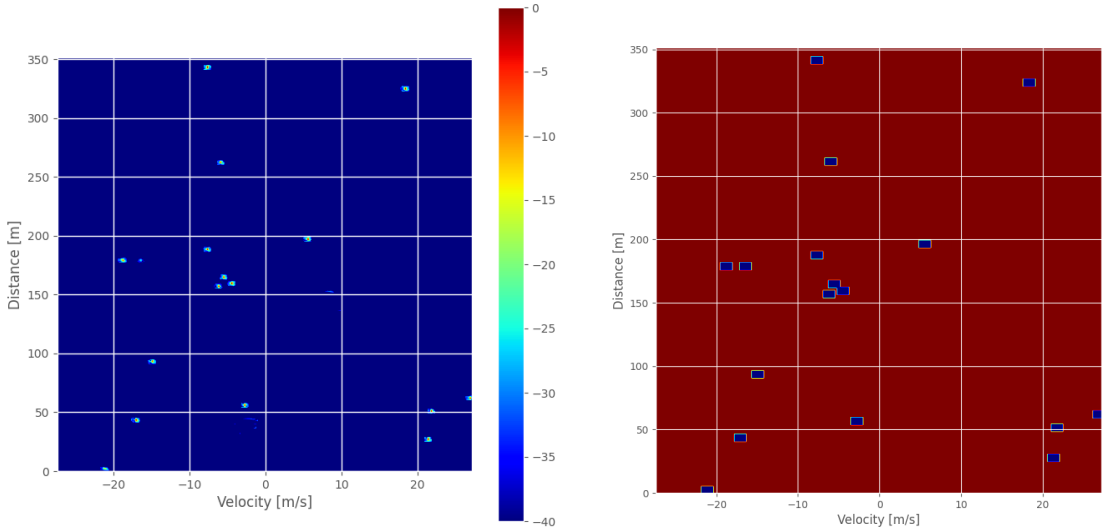
Figure 4.6: RD-maps after different mitigation algorithms for data containing coherent interference with a small clock drift. This is presented as a ghost object at (21, 146).

Figure 4.7 shows an example of when a CNN has been able to completely mitigate coherent interference while NMDC has not. The evaluation data in the figure contains coherent interference with a higher amplitude than the victim signal. By comparing the noise mask in Figure 4.7d with the interfered signal in Figure 4.7a, it can be seen that the two brightest objects in the interfered RD-map are not true objects, but ghost objects with different amounts of clock drift. Here, the deep CNN trained on SINR in Figure 4.7c performs better than NMDC in Figure 4.7b since more artifacts are present in the map from NMDC.



(a) An RD-map containing noise and coherent interference.

(b) RD-map after NMDC.



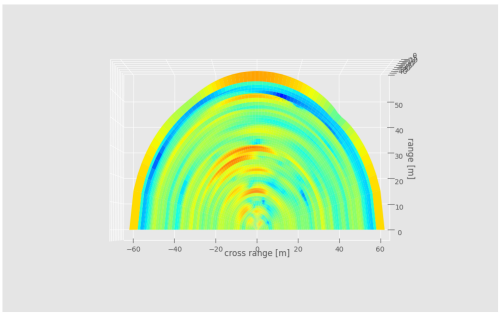
(c) RD-map after the deep CNN trained on SINR.

(d) Noise mask for the data.

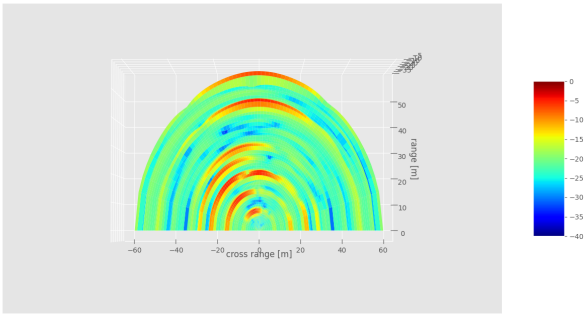
Figure 4.7: RD-maps after different mitigation algorithms for data containing two types of coherent interference with different amounts of clock drift.

To show that the simulation environment is able to produce data similar to real data, Figure 4.8 presents one RA-map with real-world data, one RA-map with simulated data where the aim is to make the simulated data resemble real data, and one RA-map consisting of a prediction from the deep CNN trained on SINR. There is one ghost object 26 m from the radar due to coherent interference. When evaluating the CNN by comparing Figure 4.8a and Figure 4.8c, the noise floor has decreased substantially and the coherent interference present at a range of 26 m is eliminated. Most of the true objects have however also been eliminated by the CNN.

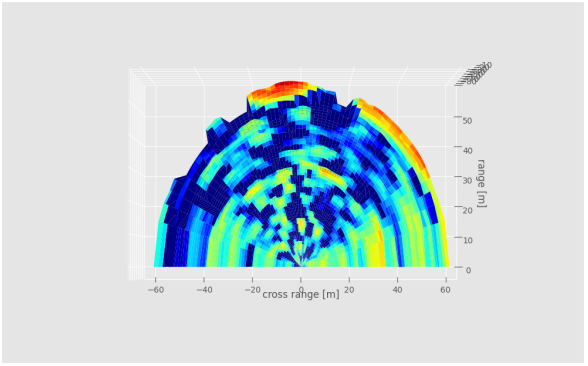
Figure 4.9 shows one RD-map with the same real-world data as in Figure 4.8, one RD-map with simulated data and one RD-map with a prediction from the deep CNN trained on SINR. It is seen that the simulated maps resemble the real ones, indicating that the simulation environment can be used to produce realistic data. By comparing Figure 4.9c with Figure 4.9a, it can be seen that the CNN has once again been able to decrease the noise floor substantially and the coherent interference is completely removed. However, the algorithm has also removed a few true objects since the number of stationary objects are fewer in Figure 4.9c than in Figure 4.9a.



(a) An RA-map containing real-world data with stationary objects and coherent interference present as a ghost object 26 m away from the radar.

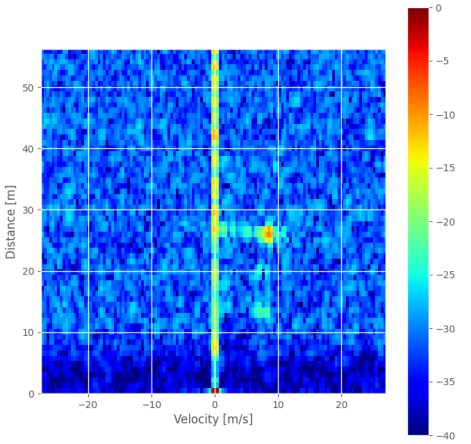


(b) A simulated RA-map, which is supposed to mimic the real-world RA-map in (a).

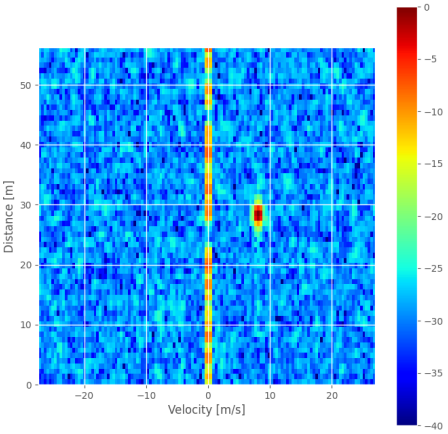


(c) A prediction made on the data in (a) from the deep CNN when it has been trained on SINR.

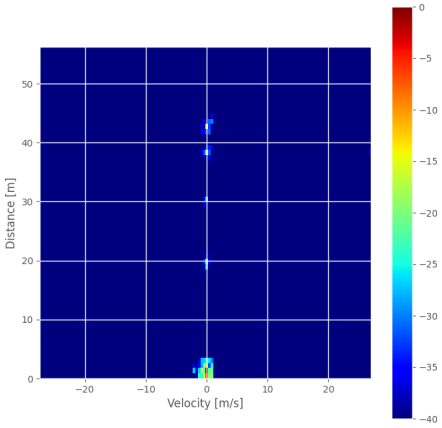
Figure 4.8: RA-maps of real-world data, simulated data and a prediction of the objects.



(a) An RD-map containing real-world data with stationary objects and a ghost object at (8, 26), generated by coherent interference.



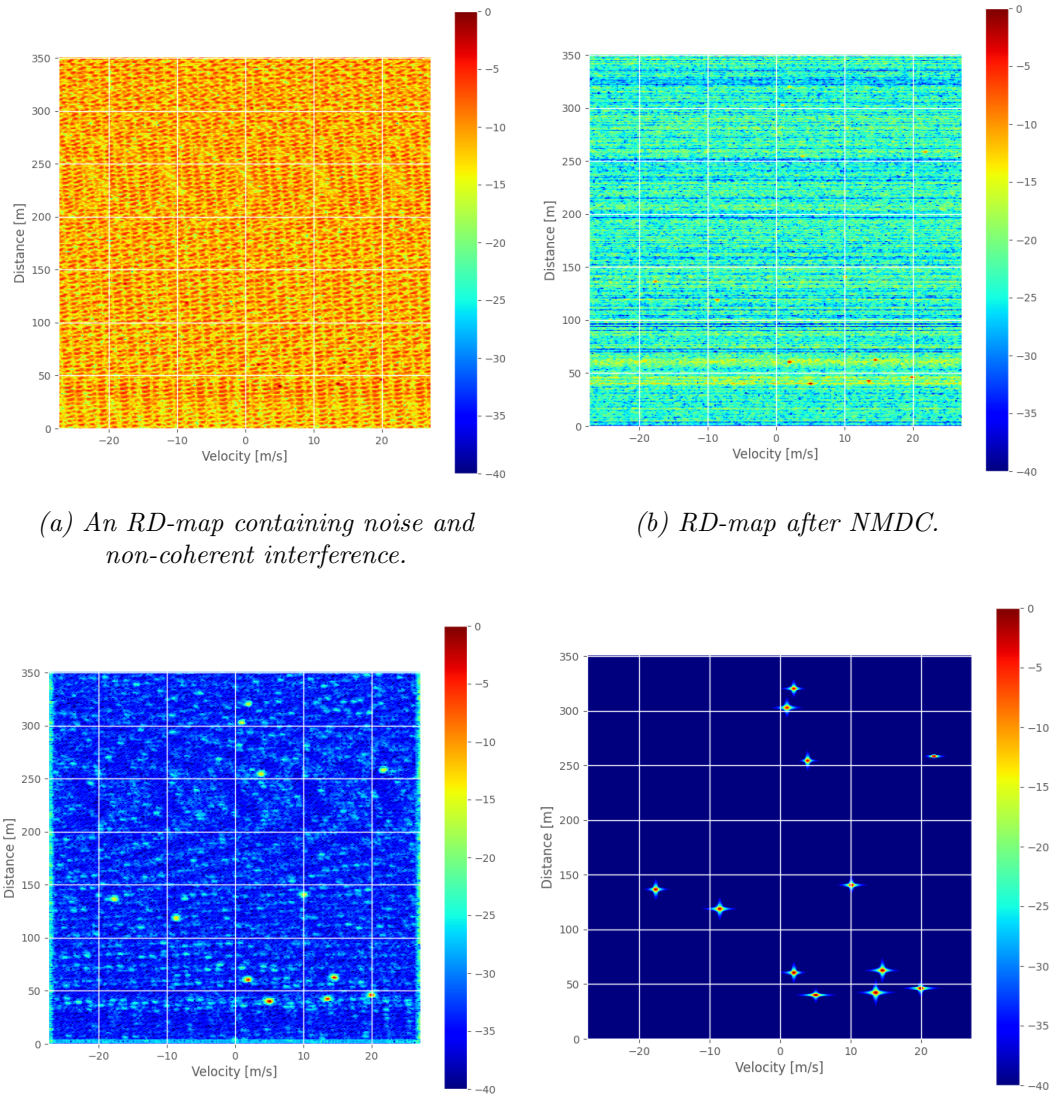
(b) A simulated RD-map, which is supposed to mimic the real-world RD-map in (a).



(c) A prediction of the data in (a) from the deep CNN when it has been trained on SINR.

Figure 4.9: RD-maps of real-world data, simulated data and a prediction of the objects. The bright red spot at (0,0) in (a) is due to reflections from the inside of the radar dome.

The final test was to vary the signal strength of the objects, by decreasing the power for objects far away from the radar. The result of this is presented in Figure 4.10. By comparing Figure 4.10a with Figure 4.10d, it can be seen that many objects are buried under the noise floor in the interfered map and therefore hard to see. The CNN trained on MSE, in Figure 4.10c, decreases the noise floor and finds all of the objects, but the map also contains bright spots where objects are not present. NMDC, in Figure 4.10b, finds the objects without generating any other bright spots, but the noise floor is higher than for the CNN.



(a) An RD-map containing noise and non-coherent interference.

(b) RD-map after NMDC.

(c) RD-map after the deep CNN trained on EVM.

(d) Clean RD-map.

Figure 4.10: RD-maps after different mitigation algorithms for data containing non-coherent interference. The object amplitudes depend on the distance to the radar.

Chapter 5

Discussion

This section will discuss the results seen in the previous section. Section 5.1 discusses the signal processing algorithms and the CNNs ability to deal with non-coherent, semi-coherent and coherent interference, as well as what happens when all are present at the same time. Section 5.2 discusses the CNNs, how they responded to different inputs and the hyper-parameters that were chosen. Section 5.3 discusses the evaluation metrics used for this project. Section 5.4 discusses what happened when the models were used on real radar data.

5.1 Evaluating on Different Disturbance Types

5.1.1 Non-Coherent Interference

Non-coherent interference is characterized by randomness and spontaneousness. This often results in fewer samples being disturbed in a frame, since the interference appears in small bursts, rather than acting consistently over the entire frame. Due to the IF-band, high frequency content in the baseband signal is excluded and therefore fewer samples are considered interfered. Furthermore, the disturbed samples often appear with a larger amplitude than the non-interfered samples, making the interference easier to locate. Multiple aggressors do seemingly not make it harder for the algorithms, compared to only one aggressor. The zeroing algorithms are therefore useful here, since it is easier to identify interference, and removing it comes with few drawbacks.

NMDC works very well, since non-coherent interference is not present at the same samples for all chirps in a frame. Hence, there are usually some samples in the chirps that are not interfered, and these are used to reconstruct the zeroed-out signal. NMDC also manages to remarkably reduce the amount of white noise in the data, which is expected since it is only the noise with the lowest amplitude over all chirps that is included in the reconstructed signal.

The CNNs outperform the tested signal processing algorithms, both by strictly assessing the metrics and by observing the figures. The CNNs only really struggles when there are objects present below the noise floor. However, this is the case for the other algorithms as well.

5.1.2 Semi-Coherent Interference

Semi-coherent interference disturbs more samples where interference is present, and it is more common that the interference lasts over a more extensive part of a chirp. This is due to the fact that the aggressor and the victim have similar properties. Many aggressors increase the probability of many samples in a frame being disturbed, which the NMDC algorithm struggles with. Classifying data as interfered also becomes more difficult, especially if there are many interfering radars present because then the interference will not appear in small bursts. The perfect zeroing algorithm removes more data than in the non-coherent case since more samples are interfered, which causes the algorithm to perform worse here.

5.1.3 Coherent Interference

Adding coherent interference to an RD map does not significantly decrease the SINR like the other interference types. This is because the signal strength of the interfering signal has been adjusted to be equal to the strength of an object in order to make it harder to distinguish the two. A successful removal of a ghost object does not increase the SINR significantly. The SINR as an evaluation metric should therefore not be as highly regarded, since noise reduction plays a bigger role for the metric in this case.

Interference subject to clock drift is smeared in the spectra, see Figure 4.2a. This in conjunction with the signal strength of the ghost objects being the same as for true objects results in a lower energy density for the smeared objects. The true objects are therefore slightly more intense in the RD-map than their interfered counterparts, when clock drift is present. To make sure that the CNNs simply did not learn by just checking the amplitude of the signal, evaluations with higher powered coherent disturbances were performed, for instance in Figure 4.9. The network managed to remove those disturbances as well, indicating that the networks can handle different interference amplitudes.

When all samples in a frame are interfered, the perfect zeroing algorithm virtually removes all data samples, which makes the RD-map very difficult to interpret, see Figure 4.2d. When coherent interference is present within the IF-band, a huge amount of samples are interfered and hence discarded by the zeroing algorithm. The NMDC algorithm manages to lower the interference amplitude slightly, but the ghost object is still clearly visible in the RD map, see Figure 4.2e. The non-perfect zeroing algorithm is struggling with distinguishing ghost objects and real objects, and as a result the algorithm does basically nothing to remove interference.

The CNNs manage to successfully remove almost all ghost objects, where the only remains are fragments, making them seemingly the best for denoising coherent interference out of the tested alternatives, see Figure 4.2f. However, when the clock drift is too small, the networks, as well as the signal processing algorithms occasionally fail to remove the interference. In Figure 4.6, a ghost object with very little clock drift is present. When the clock drift is too small, the disturbance does not visibly drift in frequency, meaning that it will have no vertical smearing, i.e., no smearing in the range direction. There will still be drifts in phase, which results in horizontal

smearing, but it will be very small, and sometimes too small to distinguish from a real object. When there is no frequency drift, the ghost object remains in the same samples over all chirps, exactly like a real object, which results in difficulties for both the NMDC algorithm and the CNNs, see Figure 4.6. In cases where there is enough clock drift for the signal to change its primary frequency bin, the network, as well as NMDC, performs a lot better, suppressing the interference almost completely, see Figure 4.7. This indicates that the network have learned to mitigate interference in a similar manner as the NMDC algorithm. However, due to leakage in the FFTs, nearby frequency bins also register values. When the NMDC algorithm takes the minimum value over all samples, the amplitude of the remaining signal will be proportional to the amount of leakage in the FFT, since the minimum value will be what leaks out from the primary frequency bin. This is why the NMDC does not manage to remove the coherent interference entirely. However, the CNNs do, showing that it has learned to identify coherent interference. Since the coherent interference used in this project has been chosen to affect all chirps in a frame, the main advantage of NMDC disappears. This algorithm is useful when not all chirps contain interference, since it uses the non-interfered samples to reconstruct the true signal. If all chirps contain interference, so will the reconstructed signal.

5.1.4 A Combination of the Interference Types

The algorithms and the networks were evaluated against a test set where the interference types were randomized. In one frame, due to the possibility of several aggressor radars, several different types of interference could occur. It was noted that even if a frame had a lot of non-coherent interference, the kind of interference that signal processing algorithms are good at mitigating, a single source of coherent interference severely limited the signal processing algorithms ability to mitigate any type of interference. This is mainly due to the zeroing algorithms removing all contaminated samples, which means that if coherent interference is detected, a lot of data is removed. The NMDC algorithm is not as affected, since the likelihood of interference occurring over every chirp is still unlikely. However, the NMDC algorithm does still not manage to remove coherent interference to the extent that the network could.

In the RD-domain, a CNN is seemingly the best option for the combined test set, outperforming all the signal processing algorithms in the quantitative measurements seen in Table 4.1. The deep CNN generates an increase of SINR of 109.1 dB compared to the interfered signal when the network is trained on SINR and fed with RD-maps according to the table. When the network is trained on MSE, the EVM is decreased by 0.06. These are the best results when evaluating on RD-maps, if only one metric is regarded at a time. A qualitative assessment of Figure 4.2 and Figure 4.3 further supports that CNNs are preferable over the other algorithms. Furthermore, a CNN is also the best performing mitigation technique in the ROC-analysis, giving more true positives for any given threshold level than the other algorithms, see Figure 4.1. We believe that the great performance of the CNNs is mainly attributable to the ability to find and eliminate the coherent interference and reduce the noise floor. In terms of mitigation of the non-coherent and semi-coherent interference, the result of the signal processing algorithms are similar to

the result of the network. A qualitative assessment of Figure 4.2 and Figure 4.3 also shows that it is a lot easier to see the peaks in the maps from the CNN rather than from its signal processing counterparts. The CNNs are however a lot worse in the RA-domain, often failing at removing the interference, see Table 4.2.

5.2 Evaluating the CNNs

5.2.1 Input to the Network

5.2.1.1 Analyzing the Different Input Maps

The input to the network could be one of three things; the RD-map, the RA-map or the RDA-map. Due to computational limitations, the number of data points used in the RDA-map had to be scaled down substantially.

From the beginning, the aim was to train and validate data from the data cube and handle the three object metrics (range, velocity and AoA) simultaneously. However, it was observed that the computational power needed to process the huge amount of data was not available. In order to train on the cube at all, the amount of data points had to be reduced substantially. The RD-map consisted of $N \times M$ data points, where N is the number of samples per chirp and M is the number of chirps in a frame. The RA-map consisted of $N \times K$ data points, where K denotes the amount of angular bins. K was greater than the amount of antennas due to zero padding. By generating the cube according to the original idea, with the same parameters as for the 2D-maps, the cube would consist of $N \times M \times K$ data points. By converting this to numbers, the RD-map consisted of $400 \times 256 = 51\,200$ data points. The RA-map consisted of $400 \times 400 = 160\,000$ data points. The RDA-map would consist of $400 \times 256 \times 400 = 40\,960\,000$ data points. Training on 2D-maps required a lot of computational power and each training lasted for one to two days using a RTX 2070 Super GPU, depending on the amount of training data and the number of epochs. The original size of the 3D-maps would be 256 or 400 times the size of the 2D-maps, so the training time would increase substantially. It was desired to perform several different trainings and to include the massive cube would not be possible. To be able to train on the cube within a reasonable amount of time, the number of data points had to be reduced significantly. With a reduced amount of data points, the final RDA-map consisted of $50 \times 32 \times 50 = 80\,000$ data points, which is more similar to the amount of data in the 2D-maps.

It was decided that training the model on the two 2D-maps would give more information than training it on one 3D-map with a significantly reduced amount of data points. The 2D-maps would in total give information about the range, velocity and AoA, which was the aim of the model. The 3D-map had fewer parameters in all three directions which impaired the resolution in every direction. Zero padding could increase the resolution, but this would require more computational power, so this was limited to the angle-direction. Plots of the cube did not contribute to any gain in information since it was difficult to visualize anything but the outer edges. Visualizing RD-maps and RA-maps was more useful. The cube does however seem promising, so for future work, with more computational power, training on a full-

sized cube might be a better option than training on the 2D-maps separately.

Looking at Table 4.2, it is clear that a model trained on RA inputs does not yield a good result in the RD-domain. Conversely, the network trained on RD-maps managed to mitigate interference in the RA-domain, see Table 4.1, once again indicating that RD-data is the best input to the network. This could perhaps be due to the networks training on object masks, which do not consist of any object smearing. The objects in the RA-maps are more smeared than the ones in the RD-maps, so the object masks are worse at capturing the objects in the RA-maps. This is further discussed in Section 5.2.3.

5.2.1.2 Varying the Amplitude of the True Objects

Keeping the object amplitude constant in the training data made it easier for the networks to find the objects, indicating that they learned how to find values with a similar signal strength. However, the networks did learn how to combat coherent interference with equal or higher signal strength, indicating that they also had deeper understanding of interference mitigation. When the object signal strength varied, the CNNs performed worse. The networks still managed to learn the task, as visible in Figure 4.10, but they struggled with finding objects below the noise floor, which the NMDC-algorithm was more successful at. The networks were however very difficult to train for these scenarios, often failing to improve after just 50 epochs into the run. To improve the chances for the networks to succeed, a much larger network is proposed, but currently lies outside the scope for what is possible with the available hardware. Furthermore, trying to make the object peaks more uniform in amplitude before feeding it to the network by pre-processing could perhaps also make it better. The variable amplitude case reassembled reality better, but it was harder for both the signal processing algorithms, as well as the network, to deal with.

5.2.2 Deep vs Shallow Model

The deep model performed better across all evaluations, but the shallow model did not fall too far behind considering the size of the model, see Table 4.1. For an embedded system with limited computational power, it is much more feasible to go with the smaller variant, even if it performs slightly worse. It does not manage at all to denoise data in the domain for which it was not been trained on however, so one would have to choose which domain to use in order to make it work. In the case of the cube, the network also performs impressively well, coming in just short of the deeper model, see Table 4.3. When regarding Table 4.2, the shallow model has higher SINR when evaluated on RD-inputs compared to the deep one. However, both of the models fail to increase the SINR as compared to the interfered signal, so none of these are in fact useful in this case. When the models are evaluated on RA-inputs, the deep model performs slightly better than the shallow one.

5.2.3 Objective Functions

5.2.3.1 SINR

The SINR objective function is, as visible in (3.3), only evaluating the results against an object- and noise mask. Since the object mask only tracks objects in a 3x3 grid, the object smearing in the clean input maps is not included in the evaluation. An example of a noise mask, which is the inverse of an object mask, can be seen in Figure 4.7d. Because of the 3x3 grid, the model trains to disregard the smearing.

Rather than recreating the clean input data (which consists of smearing), the CNNs are trained to implicitly find object locations in the data. Since the clean input maps are free of noise, the CNN also learns to denoise the image. From qualitative assessment of the figures, it seems that the SINR-based networks have learned to adapt a threshold to the interfered image, and only include the data above the line. This works when all the objects are above the noise floor, which might not be the case for very interfered environments. It seems that this threshold is very accurate, based on its success in finding ghost objects with very little clock drift, something that the other algorithms or the other objective function did not manage to the same extent. A reminder that ghost objects with clock drift have a lower energy density, which makes the object peak intensity slightly lower.

However, the MSE objective function does not use the object mask to find objects, but rather just subtracts the output of the network from the clean data. The best result from this metric is therefore obtained when the output from the network is identical to the input, i.e., smearing is still present in the peaks. Since a well-trained SINR-based CNN will have learned to disregard the smearing in the peaks, the EVM might become exceptionally large for a CNN trained against the SINR, even if all interference and noise have been mitigated, see Table 4.1.

The SINR-based networks also have a lot of variance in the way the object is represented in the prediction. The predicted objects are all of difference size and intensity, a pattern that seems random, but might be due to the relative noise floor around each object in the maps with noise and interference.

In Figure 4.4, it is made clear that an object is heavily smeared in the spectrum, which has consequences for the signal processing algorithms. The NMDC for instance manages well with removing most noise, but does not manage to narrow down the object location. The SINR-based CNN does a much better job at this, mostly due to it training against the true positions of the objects, rather than the actual, smeared out peaks. The smearing is however also present in the interference, which means that learning how the smearing behaves is important in order to mitigate it.

In the RA-domain, it was discovered that the network was most of the time not capable of getting rid of coherent interference, and instead classified it as objects, showing that this network structure is not suitable for interference mitigation in this domain. This is likely due to the network not being trained against the non-linear nature of the smearing.

5.2.3.2 MSE

Training the networks on MSE resulted in a better EVM and slightly worse SINR, as visible in Tables 4.1 and 4.2. This is likely due to the object smearing that the network has learned to recreate, see Figure 4.2f. Overall, the MSE-based network generates the best result in Figure 4.3, where the detected objects are all similar looking in the plots and have a unison intensity. MSE-based networks are also a lot better at finding objects in heavily disturbed frames than its SINR counterpart, which indicates that MSE-based networks have managed to learn how to mitigate interference, rather than just detecting objects. It is however not as good at mitigating coherent interference with low clock drift, sometimes missing ghost objects that look to similar to true ones.

Figure 4.4 and Figure 4.5 show mitigation in the RA-domain when non-coherent and coherent interference is present respectively. By comparing Figure 4.4e with Figure 4.5d, it is clear that the deep CNN trained on MSE performs well when it comes to mitigating non-coherent interference but it is not able to remove the coherent interference.

Since the MSE-networks have been trained with smeared plots, these networks are in theory a much better fit in the RA-domain, seeing as combating smearing is crucial for learning interference mitigation in this domain. The non-linear nature of the smearing however makes it difficult for the network to fully learn. In qualitative assessments, it seems like the network has not managed to learn how to remove interference, but rather just how to reduce noise in the spectrum. It might be that the non-linearity in the smearing is difficult for the network to learn, and that making changes to the network structure could yield better results.

5.3 Comparison of the Evaluation Metrics

The quantitative measurements, EVM and SINR, are not enough to determine if a model is good or bad. For instance, a lower EVM is desirable, but the worse case does not seem to be when the EVM is large, but rather when it is around 1.0. When the metric starts to reach higher values than one, it is an indication that the model has learned to identify the objects, and also how to remove object smearing from them, yielding a much higher EVM. An EVM around one happens either by pure chance, or when all of the prediction values are equal to zero, see (3.5), where the latter seems to be the most common case. An EVM equal to one is therefore an indication that the model has not managed to learn how to mitigate the interference, instead it does not do any predictions at all. A large SINR in combination with this often means that the model has learned noise reduction, i.e., decreased the noise floor, but not how to mitigate the interference. This can be observed in Figure 4.5d, where the noise floor has been reduced, but the coherent interference is still present. Table 4.2 quantitatively presents these results, where the EVM is close to one when the SINR is high.

Looking at Table 4.2, the EVM from the clean and the interfered domains are seemingly the same. They do differ, but only moderately, indicating that the inter-

ference somehow is suppressed in the RA-domain in comparison to the RD-domain. The data set used for evaluation is the same for both domains, it is only the FFTs performed on the data that differ. A possible explanation is that the object mask was set to be the same in both the RD- and RA-domain, i.e., a 3x3 grid around every object. In the RA-domain however, the energy is spread out over a much larger amount of samples, see Figure 4.4c for an example. During the SINR-calculation, a lot of the energy was therefore not taken into account since only a 3x3 grid was regarded. It is possible that the models would learn better if interference was adjusted for the RA-domain, since right now the non-coherent and semi-coherent interference are low in amplitude for this domain. Coherent interference seems to be at the correct amplitude though, as visible in Figure 4.5. This has the upside of enforcing object precision, which can be seen in Figure 4.4f.

SINR, EVM and the ROC-curve are using object masks and the metrics are only regarding the data points that the object masks label as objects. Hence, the masks play an important role in the evaluation metrics. The size of the object grid in the mask, (3x3) seems reasonable when regarding the clean RD-map in Figure 4.3a and the noise mask in Figure 4.7d because one object in the mask covers the size of a true object. The size of the grid could however further be investigated to more optimally capture the shape of the true objects. The grid could for instance be made smaller to include less smearing. Generally, the chosen grid should reflect what properties that are desired to be investigated.

The same characteristics of the object/noise mask, with a 3x3 grid, are used for the RA-maps, shown in Figure 4.4a. However, this noise mask does not match the shape of the objects in the clean RA-map due to the substantial angular object smearing. This is probably the explanation of why both the SINR and EVM in Table 4.2 are lower than for the RD-map. An object mask that would capture the object smearing, i.e., a banana shaped grid, would perhaps increase the value of both metrics. An idea could be to use an object mask where data points with a power higher than a certain threshold would be classified as objects. This would generate an arbitrary grid that could perhaps suit more maps than just the RD-map.

The reason to evaluate with ROC-measurements mainly stem from wanting to see how many false alarms every algorithm generated. The curve shows how accurately a true object is classified as true and a false object classified as false. This is an important metric for radar systems, since radars are common within the automotive and military sector, where a false alarm can have steep consequences. The ROC-curve also shows how accurate the different algorithms can be, since a true positive can become a false positive if it predicts the wrong location for an object.

Deciding which model that is the best from strictly assessing the ROC-curve depends on the use case. The graphs in Figure 4.1 show that different models are preferable for different use cases. If it is more important to have few of false positives, but as a trade-off miss some true positives, the deep SINR-model is preferable since it follows the y-axis the most. If finding all the objects is more important, but as a trade-off have more false positives, the deep-EVM is better.

5.4 Validation on Real Data

Neural networks can be very sensitive to the input it is fed with. Even if the simulated inputs are just slightly off from what the real world looks like, the network might not function at all. In Figure 4.9, an example of what real world data looks like is shown, side by side with what the simulator is able to reproduce. In Figure 4.9c, the result of what happens when real data is fed into the deep SINR-based network is shown. The CNN has successfully managed to detect and remove the coherent interference present at around (8, 26), which is a great feat for the network. It has however managed to remove some of the true objects as well, for example one at (0, 8). For the plot shown, the number of data points are very few in comparison to what the network has trained on, which might limit the performance. Furthermore, the network has been trained on objects being randomly distributed in distance, velocity and AoA, which is not true in the real world, since many objects are likely to be stationary in front of the radar. Also, for real data, the signal power decreases as the distance to the radar increases. This has not been modelled in this project and that could make a difference for the predictions.

Figure 4.8 shows real data and how it is handled in the RA-domain. The deep CNN has not been able to make an accurate prediction at all, see Figure 4.8c. The coherent interference present at the range of 26 m is removed, but so are many true objects as well. The network has lowered the noise floor, but it has not been able to recreate the true objects. Overall the networks have struggled with RA-inputs on simulated data, so it is not a big surprise that they fail on real data.

The fact that the CNN has been able to remove the coherent interference, without being trained on real data, is promising. Using labelled real data as training data would possibly improve the performance of the model. This could perhaps also improve the RA-prediction.

Chapter 6

Conclusion

Two different CNN architectures, a deep and a shallow one, have been trained for a variety of hyper-parameters and evaluated against some conventional signal processing algorithms, with the aim to detect and mitigate FMCW radar interference. To create the necessary amount of data required to enable machine learning, a simulation environment was built according to the signal model. It has been shown that is is possible to build a simulation of coherent interference with clock drift, as well as simulating semi-coherent and non-coherent interference. The novelty that this project contributes with is modelling especially coherent interference with clock drift and training CNNs to mitigate it.

Besides altering the size of the CNN-models, two different objective functions were used: SINR and MSE. The input to the network was also altered between RD-maps, RA-maps and RDA-maps and the aim was to estimate range, velocity and AoA to several objects in heavily interfered environments.

Different varieties of interference were evaluated against the network and the signal processing algorithms, and it was concluded that the networks outperformed its signal processing counterparts both in quantitative metrics, and in qualitative assessment. This was mainly due to the problems caused by the coherent interference type, which inhibited the signal processing algorithm far more than the network, who successfully managed to learn how to remove them.

By comparing the two objective functions SINR and MSE, it can be concluded that training to minimize SINR produced a model that is useful for object detection, since it was great at distinguishing true objects from ghost objects, but it struggled with identifying objects below the noise floor. By minimizing MSE instead, the generated model became better at mitigating interference, which enabled the detection of objects below the noise floor. However, it became less robust at detecting and eliminating ghost objects. In summary, training on SINR reduced the amount of false positives and training on MSE increased the probability of detection.

Future research within this area can consist of collecting more real data to train and evaluate models outside a simulation environment. Also, using labelled real data as training data would perhaps make the CNNs perform better on real data. Continued investigation into modelling the amplitude of objects in front of the radar

could also contribute to making simulated data more realistic. An increased amount of computational power would make it possible to investigate a network with full-scaled RDA-cubes as input to the network, instead of using down-scaled maps. It could also enable the training of more massive networks that might manage to find the non-linear patterns present in the RA-domain. These networks may handle variable amplitudes better than the ones tested in this project. Testing different types of object- and noise masks might improve the results, especially in the RA-domain. A more detailed investigation of the three interference types (non-coherent, semi-coherent and coherent) and their effects could also be performed to perhaps have different mitigation techniques for different types of interference. Otherwise, it may be relevant to focus on mitigating the most common interference type to limit the complexity of the system.

Bibliography

[Akeret et al., 2017] Akeret, J., Chang, C., Lucchi, A., and Refregier, A. (2017). Radio frequency interference mitigation using deep convolutional neural networks. *Astronomy and computing*, 18:35–39.

[Aydogdu et al., 2019a] Aydogdu, C., Carvajal, G. K., Eriksson, O., Hellsten, H., Herbertsson, H., Keskin, M. F., Nilsson, E., Rydström, M., Vanäs, K., and Wymeersch, H. (2019a). Radar interference mitigation for automated driving. *arXiv:1909.09441*.

[Aydogdu et al., 2019b] Aydogdu, C., Garcia, N., Hammarstrand, L., and Wymeersch, H. (2019b). Radar communications for combating mutual interference of FMCW radars. *arXiv:1807.01497*, pages 1–6.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer, New York.

[Chang et al., 2019] Chang, C., Woo, A., Forry, H., Sherman, J., Recht, M., Clark, R., and Levin, R. (2019). Hisar-300: An advanced airborne multi-mission surveillance radar. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–6.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*.

[Kim et al., 2018] Kim, J., Chun, J., and Song, S. (2018). Joint range and angle estimation for FMCW MIMO radar and its application. *arXiv:1811.06715*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

[Lavrenko and Goddard, 2016a] Lavrenko, V. and Goddard, N. (2016a). *Generalisation and Evaluation*, Lecture, Introductory Applied Machine Learning INFR10069, delivered in 3 october 2019, University of Edinburgh.

[Lavrenko and Goddard, 2016b] Lavrenko, V. and Goddard, N. (2016b). *Neural Networks*, Lecture, Introductory Applied Machine Learning INFR10069, delivered in 18 november 2019, University of Edinburgh.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440.

[Murali et al., 2018] Murali, S., Subburaj, K., Ginsburg, B., and Ramasubramanian, K. (2018). Interference detection in FMCW radar using a complex baseband oversampled receiver. In *2018 IEEE Radar Conference (RadarConf18)*, pages 1567–1572, Oklahoma City.

[Nikolió et al., 2016] Nikolió, D., Popovic, Z., Borenović, M., Stojković, N., Orlić, V., Dzvonkovskaya, A., and Todorovic, B. M. (2016). Multi-radar multi-target tracking algorithm for maritime surveillance at oth distances. In *2016 17th International Radar Symposium (IRS)*, pages 1–6.

[O'Shea and Nash, 2015] O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv:1511.08458*.

[Richards et al., 2015] Richards, M. A., Scheer, J. A., and Holm, W. A. (2015). *Principles of Modern Radar*. Scitech Publishing, Raleigh.

[Ristea et al., 2020] Ristea, N.-C., Anghel, A., and Ionescu, R. T. (2020). Fully convolutional neural networks for automotive radar interference mitigation. *TechRxiv:11919102*.

[Rock et al., 2019a] Rock, J., Toth, M., Meissner, P., and Pernkopf, F. (2019a). Cnns for interference mitigaton and denoising in automotive radar using real world data. In *2019 NeurIPS Workshop on Machine Learning for Autonomous Driving*, Vancouver.

[Rock et al., 2019b] Rock, J., Toth, M., Messner, E., Meissner, P., and Pernkopf, F. (2019b). Complex signal denoising and interference mitigation for automotive radar using convolutional neural networks. *arXiv:1906.10044*.

[Sandeep Rao, 2016] Sandeep Rao, T. I. (2016). Introduction to mmwave sensing: FMCW radars.

[Suleymanov, 2016] Suleymanov, S. (2016). Design and implementation of an fmcw radar signal processing module for automotive applications. Master's thesis, University of Twente.

[Toth et al., 2018] Toth, M., Meissner, P., Melzer, A., and Witrisal, K. (2018). Analytical investigation of non-coherent mutual FMCW radar interference. In *2018 15th European Radar Conference (EuRAD)*, pages 71–74, Madrid. IEEE.

[Wagner et al., 2018] Wagner, M., Sulejmani, F., Melzer, A., Meissner, P., and Huemer, M. (2018). Threshold-free interference cancellation method for automotive FMCW radar systems. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, Florence.

[Yang and Mani, 2020] Yang, Z. and Mani, A. (2020). Interference mitigation for AWR/IWR devices. pages 1–6.

[Yeary et al., 2011] Yeary, M., Crain, G., Zahrai, A., Kelley, R., Meier, J., Zhang, Y., Ivic, I., Curtis, C., Palmer, R., Yu, T. ., and Doviak, R. (2011). An update on the multi-channel phased array weather radar at the national weather radar testbed. In *2011 IEEE RadarCon (RADAR)*, pages 971–973, Kansas City.

[Zhang, 2019] Zhang, J. (2019). Gradient descent based optimization algorithms for deep learning models training. *arXiv:1903.03614*.