Approximately Optimal Radar Resource Allocation

Joint Radar Communication for Automotive Applications

K. Jayachandra

Research under the i-CAVE project
in collaboration with NXP Semiconductors
by the MS3 Group, EEMCS, TU Delft

Approximately Optimal Radar Resource Allocation

Joint Radar Communication for Automotive Applications

by

K. Jayachandra

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday July 12, 2021 at 13:30.

Student number:	5166160	
Project duration:	November 23, 2020 – July 12, 2021	
Thesis committee:	prof. dr. A. Yarovoy,	(Supervisor) Chairman of MS3, TU Delft
	dr. ir. J. N. Driessen,	Associate Professor, TU Delft
	prof. dr. F. M. J. Willems,	Professor Emeritus, TU Eindhoven
	ir. M. Schöpe,	(Mentor) PhD Candidate, TU Delft
	dr. A. Filippi,	Technical Director, NXP Semiconductors

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Contact

The contact information of the people involved in the work done for this thesis is listed below:

Student

Name:	K. Jayachandra
Title:	MSc Student
Email:	k.jayachandra@student.tudelft.nl
Phone:	+31-6 47499001

TU Delft

Name:	prof. dr. A. Yarovoy
Title:	Professor, Chair, MS3 Group
Email:	a.yarovoy@tudelft.nl
Phone:	+31-15-27 82496

Name:	ir. M. Schöpe
Title:	PhD Candidate
Email:	m.i.schope@tudelft.nl
Phone:	+31-15-27 88227

NXP Semiconductors

Name:	dr. A. Filippi
Title:	Technical Director
Email:	alessio.filippi@nxp.com

Name:	dr. F. Laghezza
Title:	Principal Radar Engineer
Email:	francesco.laghezza@nxp.com

Name:	ir. J. Overdevest
Title:	Signal Processing Engineer
Email:	jeroen.overdevest@nxp.com

Abstract

Karan JAYACHANDRA

Approximately Optimal Radar Resource Allocation
Joint Radar Communication for Automotive Applications

Due to the rising number of wireless device users, it is expected that there will be a scarcity in the spectrum. The will especially true for the Automotive Spectrum between 77 and 81 GHz. In this thesis, we apply Sensor Management to the Joint Radar Communication scenario. We develop an algorithm that can allocate resources to both sensing and communication tasks. The resources are across both frequency bands and time division. The proposed solution minimizes the global uncertainty while avoiding interference. The solution is formulated via an optimization problem and can perform a comparison of communication and sensing costs side-by-side. The sensing resources are allocated using a cost minimization and the communication strategy is found by formulating a selection problem. The solution demonstrates that cooperation can lead to higher accuracy for all sensors involved across multiple targets while using resources more efficiently. The algorithm is able to allocate resources to multiple sensors in a non-myopic framework. A proof-of-concept for a distributed version of the algorithm is also shown.

Preface

After completing my Bachelor's degree in Technology from Amrita University in Electronics and Communication, I worked for three years to gain some work experience in SAP India Pvt. Ltd. However, my thirst for knowledge led me to TU Delft in pursuit of a Master's Degree. The thesis was part of the MSc Electrical Engineering program at EEMCS, TU Delft. This opportunity was created as part of an event for first-year master's students to interact with the work done at the MS3 (Microwave Sensing, Signals & Systems) research group. The MS3 Master's Market event led me to an interview with Dr. Filippi and Ir. Overdevest from NXP Semiconductors. They then credited me with an opportunity to work first as a student intern and then continue with a graduation project.

I would like to first thank my parents without whom none of this would have been possible. I have built my work while standing on their shoulders.

I would like to most importantly recognize the guidance provided by ir. Schöpe and thank him for his inputs throughout this period. His supervisorship and work have been instrumental to the successful completion of the thesis. The opportunities that the MS3 group lead by prof. dr. Yarovoy creates for students from other countries are invaluable as they don't have a professional network in The Netherlands. My utmost respect and gratitude to the group and Prof. Yarovoy. My experience working with MS3 has been exceptional. I have nothing but the utmost praise for the group, starting with the chair to my fellow graduate students.

I would also like to thank the team at NXP: dr. Filippi, dr. Laghezza, and ir. Overdevest, for providing me with this chance to learn how to work in a professional environment. I have grown both my technical and personal skills under their tutelage. Their inputs were invaluable to the work done in this thesis. I would finally like to thank Dr. Engin for making such a working relationship with NXP possible. The team at NXP has been very supportive and flexible in these trying times brought about by COVID-19. But their support has soothed the impact it has had on my experience in The Netherlands.

K. Jayachandra
Delft, July 2021

Contents

Contact iii

Abstract v

Preface vii

1 Introduction 1
- 1.1 Overview 1
- 1.2 Motivation 2
- 1.3 Assumptions 2
- 1.4 The Problem 3
- 1.5 Novelty 3
- 1.6 Nomenclature 4
- 1.7 Thesis Outline 4

2 State of the Art 5
- 2.1 Automotive Radar 5
  - 2.1.1 Modern Radar Hardware 5
  - 2.1.2 Modern Radar Signal Processing 7
  - 2.1.3 Radar-to-Radar Interference 8
- 2.2 Sensor Management 8
  - 2.2.1 Overview 9
  - 2.2.2 POMDP Solutions 10
  - 2.2.3 Policy Rollout 12
- 2.3 Joint Radar Communication 12

3 System Prototype 17
- 3.1 Target Model 17
  - 3.1.1 Target Unknowns 17
  - 3.1.2 State Dynamics 17
- 3.2 Sensor Model 18
- 3.3 Kalman Filter 19
- 3.4 Optimization Problem 20
  - 3.4.1 Cost Function 20
  - 3.4.2 Constraints 21
  - 3.4.3 General Optimization Problem 21
- 3.5 Policy Rollout 21
  - 3.5.1 Action Space 21
  - 3.5.2 Reduced Complexity 22
  - 3.5.3 Base Policy 22
- 3.6 Full Algorithm 22
- 3.7 Results 23
- 3.8 Conclusion 25

4 Sensing Time Optimization 29
- 4.1 Sensor Model 29
  - 4.1.1 Sensor Dynamics 29
  - 4.1.2 Polar Measurements 29
- 4.2 Extended Kalman Filter 30
- 4.3 Dynamic Optimization 31
- 4.4 Results 31
- 4.5 Conclusion 36

5 Communication Selection Optimization 37
5.1 Selective Communication . . . 37
5.1.1 Communication Variable . . . 37
5.1.2 Effect on Cost Function . . . 38
5.2 Results . . . 39
5.2.1 Static Results . . . 39
5.2.2 Dynamic Cases . . . 43
5.3 Conclusion . . . 46

6 Time and Frequency Division 47
6.1 Constraint Reformulation . . . 47
6.2 Results . . . 48
6.3 Distributed Solution . . . 49
6.4 Conclusion . . . 51

7 Conclusion 53
7.1 Conclusions . . . 53
7.1.1 Proposed System . . . 53
7.1.2 Results . . . 54
7.1.3 Implications . . . 54
7.2 Future Work . . . 55

A Simulation Parameters 57

B MATLAB Code / Tweaks 59
B.1 Radar Target . . . 59
B.2 Radar Sensor . . . 60
B.3 Extended Kalman Filter . . . 61
B.4 Best Sensor Action . . . 63
B.5 Best Policy . . . 65

C Dynamic Noise 71

Bibliography 73

1 Introduction

1.1. Overview

Among the top ten causes of human fatality, the only non-medical reason is road accidents. This macabre fact has triggered extensive interest in the research of automotive safety. Each accident could cost the persons involved dearly. Third parties around the scene may also suffer damages. Such events could also have a negative environmental impact due to congestion. The governments of the world invest heavily to reduce the effect of automotive mishaps. The Netherlands brought down the number of road fatalities from 1251 in 1996 to 610 in 2019 [6]. However, these numbers are still considered high, and there are plans to reduce accidents even more. Although the cause of most accidents is human error, it is up to technology to aid in a further reduction of fatalities.

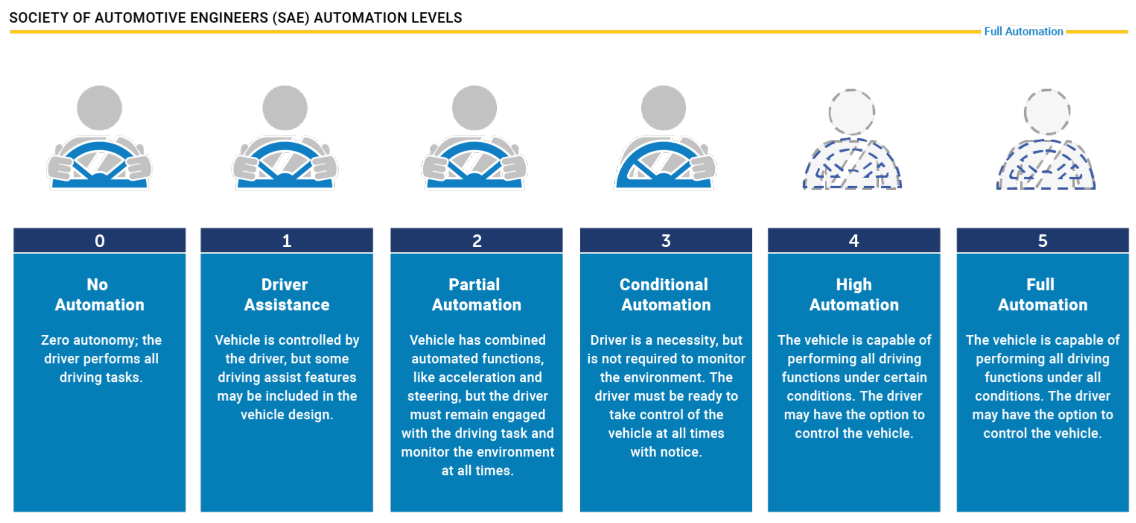

SOCIETY OF AUTOMOTIVE ENGINEERS (SAE) AUTOMATION LEVELS
Full Automation
0
No Automation
Zero autonomy; the driver performs all driving tasks.
1
Driver Assistance
Vehicle is controlled by the driver, but some driving assist features may be included in the vehicle design.
2
Partial Automation
Vehicle has combined automated functions, like acceleration and steering, but the driver must remain engaged with the driving task and monitor the environment at all times.
3
Conditional Automation
Driver is a necessity, but is not required to monitor the environment. The driver must be ready to take control of the vehicle at all times with notice.
4
High Automation
The vehicle is capable of performing all driving functions under certain conditions. The driver may have the option to control the vehicle.
5
Full Automation
The vehicle is capable of performing all driving functions under all conditions. The driver may have the option to control the vehicle.

Figure 1.1: The Road to Full Automation [25]

Removing the human element can significantly decrease the chances of accidents. However, to create fully automated driving solutions as described in the plan shown in figure 1.1, the current challenge is to have an accurate representation of the environment. Information about the surroundings aids the vehicle's systems to make better decisions on what actions to take. Without accurate sensors and an understanding of the environment, the actions taken by a fully automated vehicle could be much worse than a human driver. In recent decades, innovations in the automotive industry have been for the electrical and not the mechanical aspects of the vehicles. Currently, the most funded project in automotive research is radar.

Most commonly used automotive radar systems work using Frequency Modulated Continuous Waves (FMCW) and a Multiple Input Multiple Output (MIMO). Such systems can estimate target positions in range, elevation, azimuth, and velocity. Automotive radar systems are used to find the relative speed of targets and have good performance in poor visibility. These advantages over other sensors, such as LiDAR or Cameras, make automotive radar sensors vital.

NXP Semiconductors has been one of the leading manufacturers of integrated circuits for automotive, smart home and city, communication, mobile and industrial technologies. They have been building Radar Systems for the Automotive Sector such as the TEF82xx and the TEF810X [26] to help car manufacturers add more safety features for their customers. These Radars work in the 77 GHz band and have three transmitters and four receivers for MIMO capabilities. These systems work in three ranges, short, medium, and long, and help the vehicle build an accurate image of the targets in the scene. Automotive radar systems of the future will have even better resolution and imaging capabilities. The number of automotive radar systems is increasing dramatically and could require a large block of spectral resources.

Communication and sensing were predominantly separate fields and developed in parallel. But the large influx of wireless devices since the 1990s has put a strain on the scarce electromagnetic spectrum. Thus, solutions that use the frequency spectrum more judiciously are preferred. These systems are much better at utilizing resources, have lower costs and power consumption, and better performance [23]. Joint Radar Communication (JRC) systems use the same channel to estimate targets and communicate between nodes. There are three categories of JRC systems: Co-existence, Cooperation, and Co-design [7]. Radar nodes are said to co-exist when they try to use spectral resources efficiently. Such a solution would mitigate interference between nodes. Radar nodes that share information are said to co-operate. They perform better in such a system by considering other nodes while deciding their actions. Co-designed systems are designed from the ground up to sense and communicate. They have the best performance for all the nodes involved. This system usher in a new era of nodes that work together instead of considering other nodes as interferers.

The International Telecommunication Union (ITU) has allocated Automotive Radar a large block of the frequency spectrum between 76 and 81 GHz. This spectrum is going to be shared by a large number of users. Assigning these resources could be done in different ways across users. The best allocation can found using an optimization problem. Such an optimization assumes that some facets of the problem can be optimized, such as target uncertainty.

1.2. Motivation

Section 1.1 establishes the need for road safety. Here we consider the motivation to do this research. Although sensors have been rapidly improving in their detection capabilities and coverage, sensors have been working independently. Sensor performance could drop significantly due to radar-to-radar interference when a large number of sensors are operating simultaneously. Such a scenario could arise due to the increasing number of road vehicles and more sensors per vehicle. It could also be efficient to allow sensors to communicate with each other for mutual benefit. Detection algorithms have missed important targets due to various reasons. In such cases, other sensors could have a better understanding of the targets in the environment. Communicating this information to the original sensor could be vital to its decision-making. One proposed solution could be by encoding information about the target in the radar signal. These systems are known to have issues with reliability and design complexity.

This work focuses on resource allocation schemes to allow all sensors to sense and communicate independently in either time or frequency. Such a system should understand how to allocate resources to each sensor for sensing as well as communication.

Thus, this research aims to answer the question, **"In the context of automotive radar systems, can an algorithm be developed to allocate resources for both sensing and communication tasks"?**. The proposed system allocates resources, but the exact scheduling of the resources in time and frequency is not in the scope of this work. In other words, the resources are allocated for sensing and communication by each sensor. However, slotting these resources in time intervals and frequency bands requires future work.

1.3. Assumptions

It is vital to make certain assumptions to keep the scope of this research bounded. Real-world operations would challenge these assumptions. But the problem is framed and solved within their confines. The Mathematical Models and the MATLAB Simulations take these into account. The chapters and appendices of this thesis provide complete details of the proposed system.

All the assumptions are listed below and are fundamental to the proposed resource allocation system.

1. **Targets: Point Targets**
   The solution considers sensors tracking several point targets in a scene. This makes the proposed solution generic and it can be extended to different object types and different sensor types.

2. **Sensors: Homogeneous FMCW Radar Systems**
   The system considers all sensors to have identical specifications. These sensors are FMCW Automotive Radars that use frequency chirps to detect and track targets. However, the system considers the radiation pattern of these transceivers to be isotropic with 360° field of view (FOV).

3. **System: Centralized Approach**
   The research considers a central system performing the optimization and assigning the tasks to the sensors involved. The system assumes that each sensor in the scene can label the targets uniquely. The central system can therefore fuse data from different sensors to maintain a global track.

4. **Interference: Performance Impact**
   Radar-to-radar Interference causes loss in the performance of radar nodes. In some edge cases, it can also lead to missed detections. The proposed solution, therefore, doesn't allow sensors to operate simultaneously. The system considers the cost of interference to be infinite.

5. **Perfect Communication**
   The system does not need to define the model used for communication. The communication model is arbitrary. The sensor communication is assumed to happen reliably without the loss of data packets. The system requires only the unit resources needed to communicate.

1.4. The Problem

The proposed solution should consider three aspects from a technical and practical standpoint to address the research question defined in section 1.2. The objectives of the system to be developed are therefore listed below.

1. **Minimize Global Uncertainty**
   First and foremost, the solution should provide a better understanding of the environment to all radar nodes. The nodes can take the best course of action based on a better awareness of the environment. The system is not worthwhile if the radar nodes lose sensing performance.

2. **Fairness and Cooperation**
   All sensors should benefit from the cooperation that such a system proposes. It should not tolerate any biases towards or against any of the co-operating sensors. Thus, the system solution is to be entirely dependent on the environment and not the sensors themselves.

3. **Better Resource Use**
   The sensors should use the limited frequency and time resources efficiently for the mutual benefit of all. The system should (a) reduce electromagnetic pollution and (b) increase the longevity of the radar systems. The longevity of the sensors is increased by decreasing the effects of thermal degradation because not all sensors will operate all the time.

1.5. Novelty

The thesis proposes a novel solution to the problem discussed earlier. More complex use cases can use extensions of the proposed generic system. The system provides the following features which are unique and have not been considered in the literature so far:

1. The proposed system defines an adaptable cost function in a generic modular framework and operates in a multi-sensor multi-target scenario to allocate scarce spectral resources for both sensing and communication tasks. Prior literature has not developed such a solution to resource allocation for JRC.

2. The proposed approach evaluates the value of the information produced by every sensor. It decides if other radar nodes should receive this information. Other publications in the field have not shown such an approach.

1.6. Nomenclature

The thesis uses the following terms while discussing the problem:

1. Radar Node / Sensor refers to all antenna systems and the processing units in one vehicle. When a budget is allocated to a Sensor, it refers to all the transmitters.
2. Solution refers to the proposed algorithm and it results in the nearly optimal resource allocation strategy for sensing and communication.
3. Model refers to the model that are considered for the Sensor, Target and the Filter. In this thesis, we consider constant velocity models for the target movement for most simulations.

1.7. Thesis Outline

The organization of the rest of the thesis document is as follows:

State of the Art	Chapter 2 documents the current trends and developments in Automotive Radar, Sensor Management, and some aspects of Automotive Radar that help formulate the proposed solution.
System Prototype	Chapter 3 prototypes the new solution using a Kalman filter in a multi-sensor, multi-target environment with noisy cartesian readings of the target location.
Sensing Time Optimization	Chapter 4 extends the results from chapter 4 with polar readings of the target position. The system is now more realistic as it corresponds to an automotive setting.
Communication Selection Optimization	Chapter 5 considers a significant impact of the communication time between sensors and models a solution that optimizes both the sensing time and the communication time.
Time and Frequency Division	Chapter 6 extends the total available resources of the solution to include multiple frequency bands. The system can use both time and frequency division.
Conclusion	Chapter 7 summarizes the work done and possible future extensions to more realistic scenarios. To conclude, the thesis discusses the implications of the research.

2
State of the Art

To solve the problem discussed in chapter 1, three key technologies need to be considered. The latest developments in these areas have been discussed in the following chapter. We start with an overview of automotive radar systems to understand the sensors in use in this thesis, then continue with how these sensors are managed followed by how sensor management can be applied to both sensing and communication tasks.

2.1. Automotive Radar

The need for automotive safety has been discussed in section 1.1. Equipment used to sense the environment around an automotive vehicle can play a crucial role in the decision making process of the user or the processor in the case of autonomous vehicles. The use of such information is crucial for autonomous applications. For example, [32] shows how Cooperative Adaptive Cruise Control can help vehicles move autonomously in a straight line. Several sensor are currently in use to map the surroundings of a car such as Camera, LiDAR, Infrared or Radar. Each of these have their own advantages. This thesis however considers the case of Radar system tracking a target using polar measurements. Radar systems are used widely because of their capabilities to see through fog or mist. Radar system are also used as they have been widely studied in the recent decades. Radar systems can also be used to discern velocity and angular information about the target using MIMO systems. Current Radar Sensors operate in the $77GHz$ band as opposed to $24Ghz$ due to high spatial resolution and low antenna sizes.

2.1.1. Modern Radar Hardware

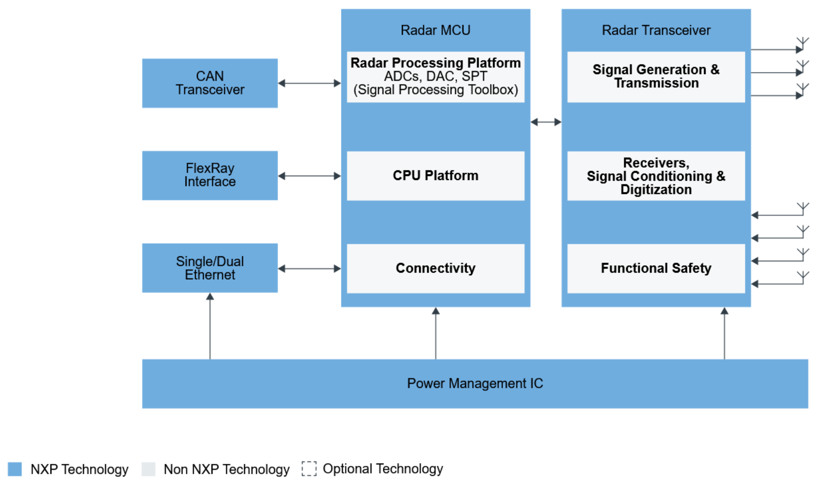


Figure 2.1: NXP Radar Solution [26]

Modern Radar Systems are varied in terms of Semiconductor Technology, Packaging and Antenna Systems. Several architectures have been proposed in the literature [14]. NXP Semiconductors has been providing a complete solution in the Automotive Radar sector and a block diagram of their offering is shown in figure 2.1. Modern Radar Systems require ingenious solutions for the antenna systems to provide high resolution while maintaining power efficiency. Current systems are broadly classified into three main categories as described in table 2.1.

Type	LRR	MRR	SRR
Maximum Transmit Power	55dBm	-9dBm/MHz	-9dBm/MHz
Bandwidth	600MHz	600MHz	4GHz
Distance Range	10-250m	1-100m	0.15-30m
Range Resolution (ΔR)	0.5m	0.5m	0.1m
Range Accuracy (δR)	0.1m	0.1m	0.02m
Velocity Resolution (Δv)	0.6m/s	0.6m/s	0.6m/s
Velocity Accuracy (δv)	0.1m/s	0.1m/s	0.1m/s
Angular Accuracy ($\delta\phi$)	0.1°	0.5°	1°
Azimuth ϕ Beamwidth (3dB)	±15°	±40°	±80°
Elevation θ Beamwidth (3dB)	±5°	±5°	±10°

Table 2.1: Radar Configurations [14]

These configurations are used for different use cases: Long Range Radar (LRR), Medium Range Radar (MRR) and Short Range Radar (SRR). Current antenna configurations allow for both Mechanical and Digital Beamforming. LRR is used in situations that require a long range with the need for resolution, such as Adaptive Cruise Control. MRR is used in cases where target are generally tracked as it offers better resolution. Cross Traffic Alert or Lane Change Assist employs this configuration. SRR is used when high resolution is required at low range such as Obstacle Detection and Parking Assist [14].

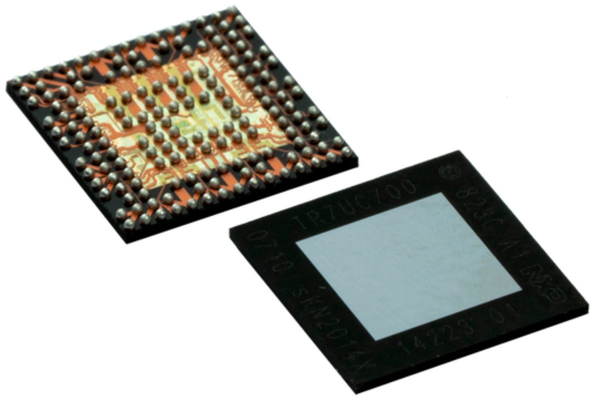


Figure 2.2: NXP Semiconductors TEF82xx: Fully Integrated 77 GHz RFCMOS Automotive Radar Transceiver [26]

Typically using the received power and processing the transmitted signal provides information about the target RCS, range, velocity and angular information. More details about the signal processing is discussed in section 2.1.2. The received power using a Radar system is well documented and described as shown in equation 2.1. When the range is know, the RCS of the target can be extracted.

$$P_R = \frac{P_T G_T G_R \lambda^2}{(4\pi)^3 R^4 L}\sigma \qquad N = kTBF \qquad \mathrm{SNR} = \frac{P_R}{N} \tag{2.1}$$

In equation 2.1, P_R and P_T refer to the received and transmitted power. G_T and G_R are the gains of the transmitted and the receiver antennas. λ refers wavelength corresponding to the center frequency of the transmitted signal, L refers to a loss fac tor and σ refers to the Radar Cross Section (RCS) of the target. The noise terms is defined by the Boltzmann Constant (k), the operating temparature (T), the bandwidth of the ADC (B) and the Noise Figure (F). These two terms therefore can be used to calculate the effective Signal-to-Noise Ratio (SNR).

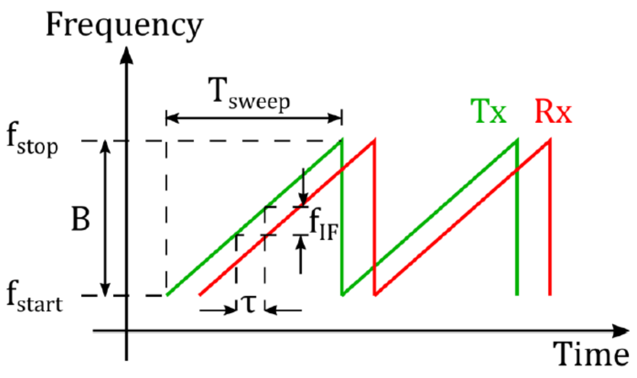


Figure 2.3: FMCW Up-Chirp [5]

2.1.2. Modern Radar Signal Processing

Automotive Radars transmit Frequency Modulated Continuous Wave (FMCW) signals. These signal are transmit waveforms with increasing or decreasing frequency, called up-chirp or down-chirp respectively. A depiction of the signal in the frequency domain is shown in figure 2.3. The received wave is generally an attenuated transmitted signal that has been delayed. Using the delay, the range information is extracted. Further, since several chirps are transmitted, the phase difference between chirps is used to extract doppler information from the signal.

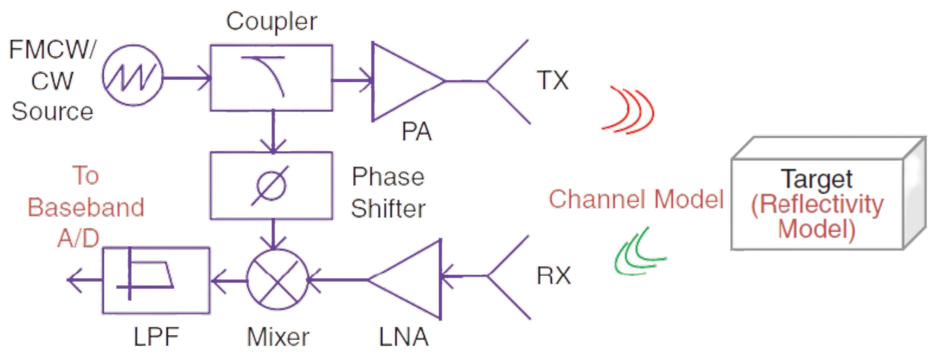


Figure 2.4: FMCW Signal Processing Chain [29]

An overview of the signal processing techniques for automotive radar has been shown in [29]. The block diagram of an FMCW Radar system is shown in figure 2.4. The received signal is mixed back with the transmitted signal to create a beat signal. This signal contains information about the difference in the two signals and brings it back to the complex base band representation which allows for easier processing. This signal is rearranged to separate individual chirps and using a two-dimensional Fast Fourier Transform (FFT), range and velocity information is extracted. A similar FFT can be applied across antennas in a MIMO system to extract angular information about the target. Several amplification steps can be included in the processing chain to help detect weaks targets. The use of FFT adds the received chirps coherently which helps detect signals that are below the noise floor.

2.1.3. Radar-to-Radar Interference

Radar-to-Radar interference is caused with the transmitted signal from a different Radar Sensor either creates ghost targets when processing the received signal or adds interference that is noise-like to the signal processing chain. An overview of the impact of interference on signal processing has been shown in [1]. An example of an interfering signal in time and frequency domain representations is shown in figure 2.5. This cause a loss in overall performance that is sometimes as drastic as to cause missed detection despite performing coherent processing.

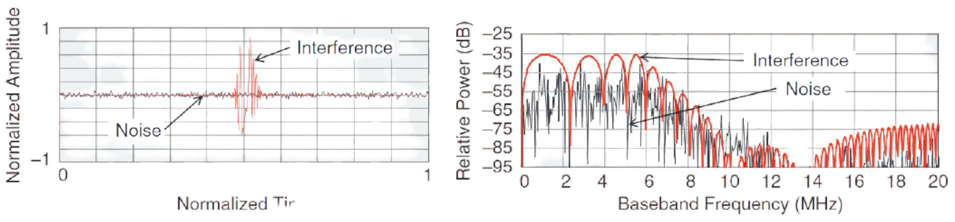


Figure 2.5: FMCW-FMCW Signal Inteference [1]

A comparison of different signal models in terms of interference impact has been done in [28] and shows that no signal is robust against interference. Therefore, to reduce the impact of interference several possible solutions have been proposed in [36], [38] and [17]. [38] and [17] propose solutions that are done to mitigate interference without any cooperation between vehicles as opposed to [36]. [38] proposed techniques to separate the interfering signal from the received signal while [17] proposes a solution that randomizes the Radar operation parameters such as Center Frequency and Bandwidth. The solution proposed in [36] considers active cooperation between radar nodes and points to solutions that involve managing a group of sensors. Such a solution involves Sensor Management and is discussed in section 2.2. [21] for example shows how Radar resources can be allocated to multiple users to allow for reduced mutual interference. The results of this has been shown in figure 2.6. To allow sensors to cooperate solutions might require synchronization between nodes. This has been demonstrated in [3] where sub-nano second level synchronization of radar nodes has been achieved to allow for cooperative radar nodes that act as passive and active radars and [10] shows that such synchronization can be done over-the-air as well. Further, [20] shows that a system level synchronization can be done to allow for communication between Radar Nodes as well. These systems use the same resource for both sensing and communication and are discussed in section 2.3. Some of these can be considered sensor management approaches.

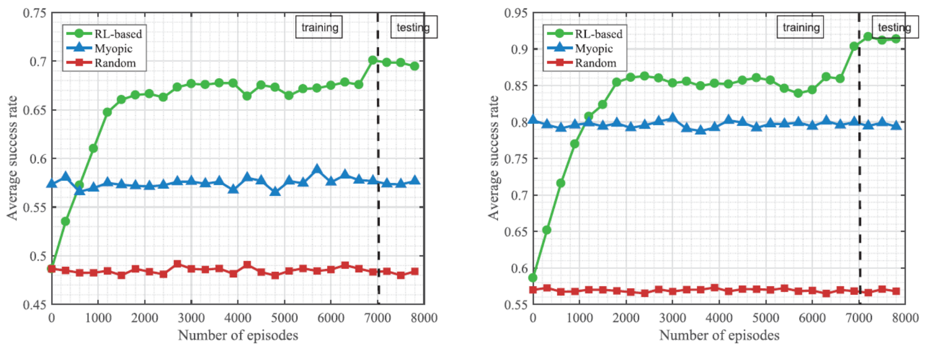


Figure 2.6: Interference Mitigation using Reinforcement Learning [21]

2.2. Sensor Management

Traditionally, Sensor Management referred to the problem of selecting the right sensor or set of sensors to use to generate a measurement. This was also referred to as 'Sensor Scheduling'. Both terms were used interchangeably when sensors were rudimentary and did not have many degrees of freedom. However, in modern applications, Sensor Management is defined as determining the optimal measurement configuration of a set of sensors. This configuration would affect the physical performance of the sensors and therefore, using Sensor Management, the performance of the system can be controlled.

2.2.1. Overview

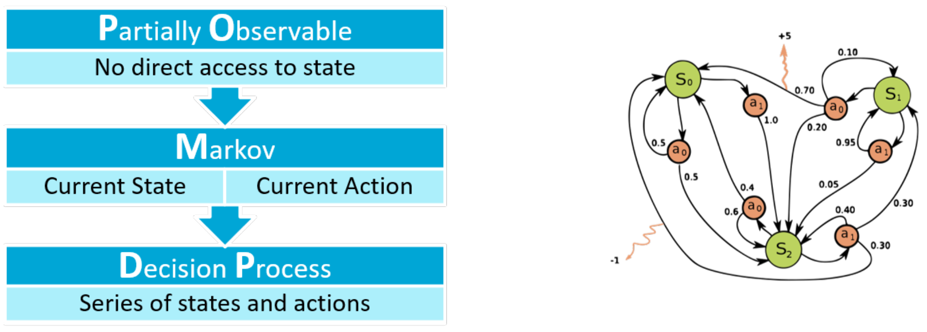


Figure 2.7: Partially Observable Markov Decision Process [40]

Historically, Sensor Management was developed because of the need for accurate measurements. Closed loop strategies were first steps developed to help take more accurate measurements. Initial applications were to decide what information would be relevant for the limited attentions pilots could spare to the cockpit [15]. With the advancement of sensor technology, new methods to manage sensors were developed. Most commonly the use of Multi-Armed Bandit (MAB) representations and Partially Observable Markov Decision Processes (POMDPs). POMDPs are described in figure 2.7 and formulate sensor management as a decision process. MAB approached calculated a reward associated with every action taken by Sensor Management. POMDP are a subset of MAB formulations. POMDPs are processes where an action decides the subsequent state of a system. The Markovian aspect refers to the system having no memory, that is, the next state is only dependent on the current action and not the previous ones. The partially observable property refers to the fact that direct access to the state might not be available to an observer.

The advantages of Sensor Management are well documented. [11] for example, shows that using sensor management can decrease power consumption of a distributed radar system by 50% when it is used to optimize power usage. When it is used to optimize power allocation, it can be used to minimize the mean-square-error (MSE) of the target track. This shows that Sensor Management can be used to optimize sensor use for different objective functions.

With Shannon's 'Mathematical Theory of Communication', POMDP were supplemented by evaluations of actions based on the information gain they provided. Current research methodologies have been focusing on such mathematical approaches. Several solution to the POMDP formulation have been developed over the course of the last few decades [8]. This led to optimization that favoured the minimization of the Cramer-Rao Lower Bound (CRLB) as shown in [4]. Here, Sensor Management is used to optimize budget allocation to a set of sensors tracking a target following a trajectory as shown in figure 2.8. The budget allocated per sensor have been documented in figure 2.9 and the results are as expected. The sensors closest to the target receive the most budget. Such approaches are common and are also documented in [41] and [27].

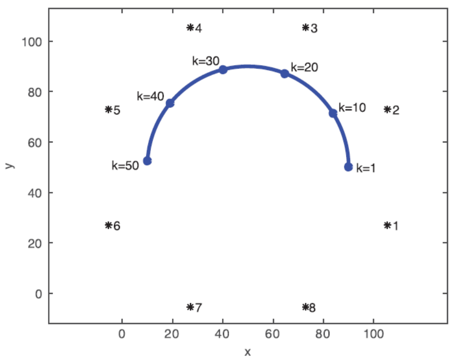


Figure 2.8: Target Trajectory (Figure from [4])

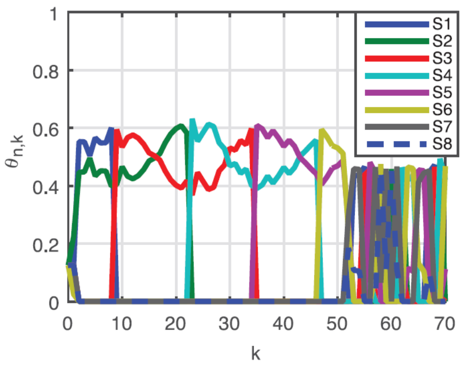


Figure 2.9: Dynamic Budget Allocation using Information Gain (Figure from [4])

2.2.2. POMDP Solutions

Solutions to POMDPs have generally been based on the approximation of Q-Values [8]. A Q-Value of an action can be defined as a reward function associated with taking that action. The Q-Value is defined per action and is dependent on the horizon (H). The horizon is defined as the number of time intervals in the future the system looks ahead to decide the reward of the action. To define the Q-Value, we look at Bellman's Principle at described in [8]. Bellman's Principle states that the optimal objective function $V_H^*(b_0)$ defined over a horizon H with an initial belief state b_0 to be:

$$V_H^*(b_0) = \max_a \left(r(b_0, a) + E\left[V_{H-1}^*(b_1) | b_0, a \right] \right) \tag{2.2}$$

An objective function defines a quantity that needs to be optimized by the process. Potential candidates could be accuracy or precision of the track or resource use. A belief state refers to the current best estimate of the state of the system or a probability distribution across possible states. Bellman's principle in essence states that the optimal objective function with a look ahead of H is decided by the expected value of reward obtained by taking an action a as the state transitions from b_0 to some future belief state b_1. This implies therefore that the best action to be taken can be defined as the action that maximizes the reward defined in 2.2.

$$\pi_0^*(b_0) = \arg\max_a \left(r(b_0, a) + E\left[V_{H-1}^*(b_1) | b_0, a \right] \right) \tag{2.3}$$

This principle can therefore be extended to a recursive process and an objective function can be defined for every action that can be taken within the POMDP framework. This function is referred to as the Q-Value of an action. There the best action that can be taken can be rewritten as the action that has the highest Q-Value as described in equation 2.5.

$$Q_{H-k}(b_k, a) = r(b_k, a) + E\left[V_{H-k-1}^*(b_{k+1}) | b_k, a \right] \tag{2.4}$$

$$\pi_k^*(b_k) = \arg\max_a Q_{H-k}(b_k, a) \tag{2.5}$$

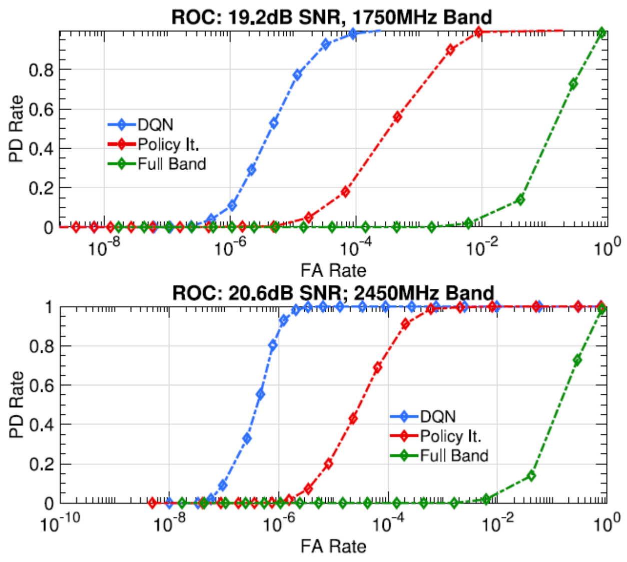


Figure 2.10: Deep Reinforcement Learning Control for Radar Detection and Tracking in Congested Spectral Environments [37]

$\pi_k^*(b_k)$ is referred to as the optimal policy as it describes the best action taken at belief state b_k. If the value of H is very large, the optimal policy can be considered to be stationary. This is because, if the function can decide the best action for a sufficiently large horizon, the best action would generally not change with time intervals. It is important to note that Q-Value is the sum of the term $r(b_k, a)$, immediate reward for choosing action a and $E\left[V_{H-1}^*(b_1)|b_0, a\right]$, the expected value of the reward until horizon H. The first term can be calculated easily but to calculate the second term, certain approximations are required as it is calculated for a large horizon. Several methods have been proposed to approximate Q-Values of the action. Some of them have been discussed below:

1. **Monte Carlo Sampling** methods use a computer to generate a large number of simulations using random variables and find the expectation using these simulations. This solution is generally avoided due to the computational complexity.

2. **Relaxation of the Optimization Problem** can be done in certain situations. This can result in formulations that are easily solved using known algorithms. The expected value term can be reformulated into such a problem leading to bounded problem.

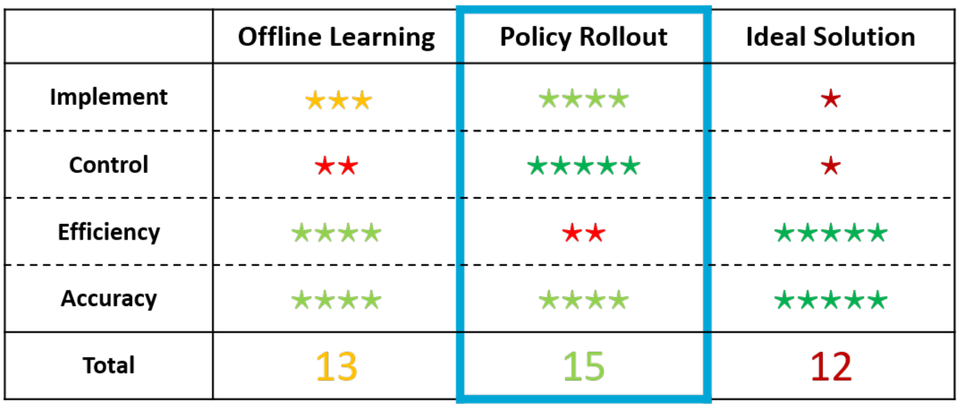
3. **Heuristic Approximations** can help simplify the calculation of the expected reward across the horizon using domain knowledge. Examples can include the use of terrain maps to assign rewards based on expected target positions. This can lead to easier calculation of the expected value for the future reward.

4. **Parametric Approximations** uses knowledge of how the Q-Value should behave to develop a numerical model to calculate the expected value of the reward. Popular solution for this approach include offline learning methods such as Reinforcement learning or Q-learning.

5. **Action-sequence Approximations** are used to describe solutions that fix the actions to be taken before they actually happen. This would lead to solutions that are easier to solve as compared to a Monte Carlo simulation.

6. **Rollout** is used when the best action is reached iteratively using a known value for a base policy. The base process can be calculated using a Monte Carlo simulations or Numerical Analysis.

Approximations are generally used in all solutions to the POMDP framework, however, the main considerations for the solution proposed in this thesis have summarized in figure 2.11. The two solutions based on Offline Learning and Policy Rollout were considered because other solutions are too simplistic and consider heuristic approximations. Although Offline learning scored better than the Policy Rollout in the operational efficiency, Policy Rollout had it's advantages in terms of the ease of implementation and degrees of control of the solution. The accuracy of the solutions have been shown to be the same. A third solution name 'Ideal Solution' has been added for comparison to describe the optimal best policy of the POMDP framework. This solution would be the most accurate and would not need to be calculated if already known. However, it cannot be implemented or controlled. Both implementations have been considered in the literature. The need to avoid interference has been shown in section 2.1. [37] uses Deep Reinforcement learning in this regard to alter the Bandwidth and Center Frequency of the transmitted LFM waveform to improve performance. A comparison with Policy Rollout is also done and it is shown to have better performance as shown in figure 2.10. [35] has also demonstrated how a Radar System can maximize its performance when it is required to operate in conjunction with a communication system using Reinforcement Learning.



	Offline Learning	Policy Rollout	Ideal Solution
Implement	★★★	★★★★	★
Control	★★	★★★★★	★
Efficiency	★★★★	★★	★★★★★
Accuracy	★★★★	★★★★	★★★★★
Total	13	15	12

Figure 2.11: POMDP Solution Comparison

The use of Policy Rollout has been described in section 2.2.3 and offers other advantages. Implementation of solution using Policy Rollout is faster as it does not require training data. In the context of this thesis, data sets for Joint Radar Communication are not easily available as is the case in new areas of research. The thesis aims to show that POMDPs can be used to answer the research question but other implementations could produce results more efficiently. As described in [31], both online and offline algorithms need to be developed in synergy with one another. They are not at odds with each other.

2.2.3. Policy Rollout

A Policy Rollout is used to describe solutions that define a base policy π_{base} and it is used as a reference to calculate a policy marginally better in the next iteration. This is repeated until the optimal policy π^* is found. Therefore the initial object function can be defined as as shown in equation 2.6. At every iteration, a new policy is found that marginally improves the Q-Value. Therefore a sequence of actions are calculated leading to the optimal action to be found at the end. Finding the optimal policy using Policy Rollout can also be done in parallel using multiple base policies. This would help in finding the optimal policy faster in instances where the base policy is quite different from the optimal policy.

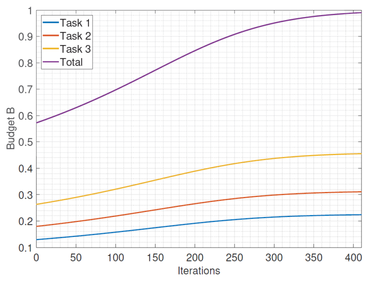


Figure 2.12: Budget Allocation using POMDP (Figure from [33])

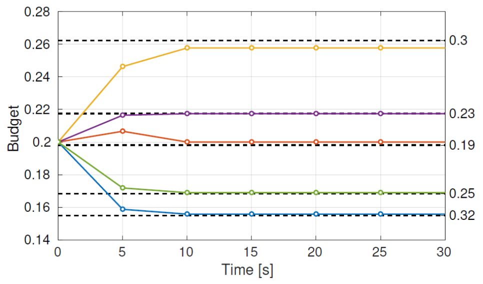


Figure 2.13: Dynamic Budget Allocation using POMDP (Figure from [34])

Research along these lines have been done in [33] and [34]. In [33] optimal budgets were allocated in a tracking scenario to different targets being observed using a Radar system. The budget is allocated based on the impact the action has on the track accuracy defined by the error covariance matrix of a Kalman Filter tracking the target. It uses a policy rollout with a Lagragian relaxation to solve the POMDP framework. The results are shown in figure 2.12. [34] extends these results dynamic scenario and the results are shown in figure 2.13. The results show that budget have been allocated for multiple 'tasks' denoted by the colored lines.

$$Q^{\pi_{\text{base}}}(b,a) = r(b,a) + E\left[V^{\pi_{\text{base}}}(b')|b,a\right] \tag{2.6}$$

These approaches however have proposed solutions that decide between similar tasks, i.e., sensing budgets. The latest trend wireless devices however has been in using the same resources for both sensing and communication. Therefore there is a need to develop a solution that can used to allocate resources for both tasks. In section 2.3 we motivate the need for such systems and current trends in their development and use.

2.3. Joint Radar Communication

As the number of users of wireless devices are growing, there is a need for more efficient use of spectral resources. This has forced the users of Radar bands to consider using the same resources as those used in communication standards. Such systems have shown to have benefits in [30]. Their use cases range from (a) Automotive and V2V Communications, (b) Commercial Flight Control, (c) Communications and Military Radar, (d) Medical Sensors and Monitoring and so on. Several topologies have also been considered to solve this problem and can be mainly categorized into the following types [23]:

1. **Non-integration** is the currently used scenario for wireless devices. This relies on users avoiding channels currently in use and relying on isolation from other users to operate. This however is not perfect and users are very susceptible to interference.

2. **Co-existence** scenario treat other users as interferers for the current use case. They use different mitigation strategies to avoid interference effects but with increasing number of users, this might not be feasible. This is the current level of sensing-communication RF convergence.

3. **Cooperation** between users can greatly benefit everyone involved and can create solutions that could not be possible earlier. Here information is exchanged willingly between users and they employ strategies to ensure minimal impact on others.

4. **Co-design** implies that systems are designed with other users in mind. For example, radar systems are designed with embedded communication waveforms to help other vehicles understand the surroundings better. This could also be used by communication users to use standards that could help Passive Radar operate.

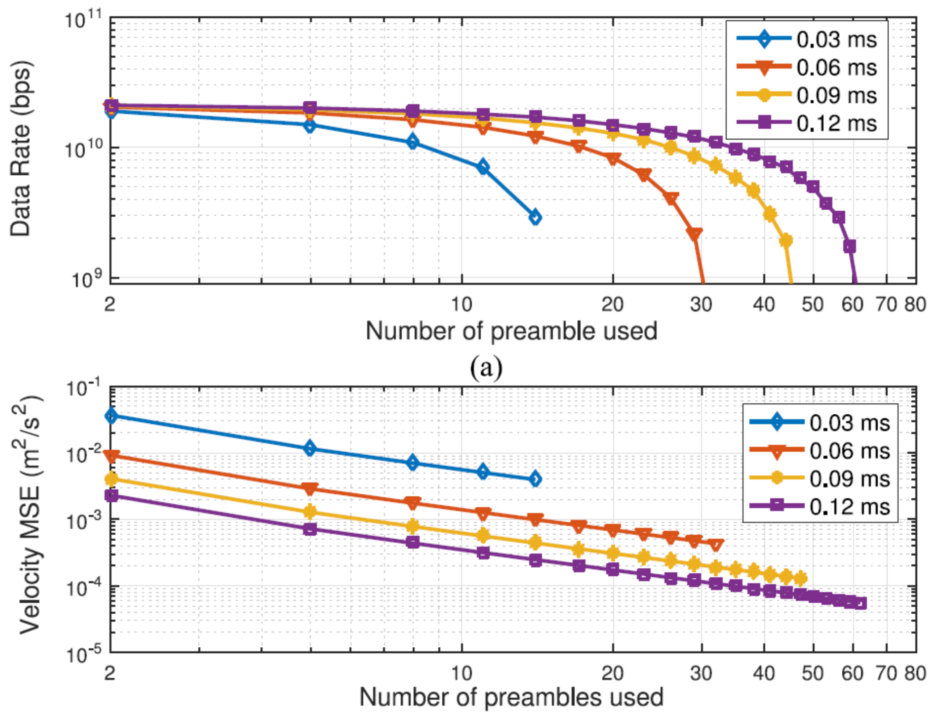


Figure 2.14: IEEE 802.11ad Radcom System Data Rate [18]

Employment of such systems has been proposed in the Automotive Radar spectrum as well in [23] and it has been shown that such applications require advanced signal processing techniques. Proposed solutions need to be robust because of high attenuation and short coherence times in this band. Such systems can also be used to remove Interference effects as described in section 2.1 in dense use cases. [18] proposes a solution based on IEEE 802.11ad standard and demonstrates that a GBps connection rate can be maintained while sensing targets as shown in figure 2.14. Experimental results have also been documented in [19] and have shown that using realizable hardware, phase coded FMCW waves are able to perform both sensing and communication operations.

Other solutions have employed the JRC framework to specifically mitigate the Interference cause of Radar Nodes as shown in [2]. Here a distributed network employing a novel solution called 'RadChat' was able to avoid interference using scheduling across time and frequency. Their solution is able to remove Radar-to-Radar interference in a timely manner with differing number of Radars as shown in figure 2.15. However this approach does not factor in the environment in their analysis. [22] however documents other approaches that can be considered while designing JRC systems. It is shown that time and frequency sharing as possible alternatives to signal sharing. As described earlier, the joint waveform design approaches require complex hardware and signal processing algorithms, while time or frequency duplexing can mitigate some of this complexity. The existing approaches for JRC have been also documented more thoroughly in [13] and have been summarized in table 2.2.

[12] have applied the time and frequency sharing approach by optimizing information gain as described in section 2.2 to decide both sensing and communication budget. However the solution proposed here is myopic and does not take into account information about the environment. Their results are shown in figure 2.16 and it is shown that an minimum CRLB can be meet with minimal violations while providing resources

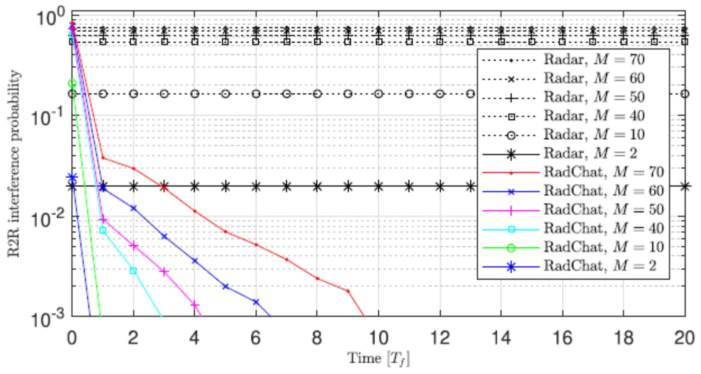
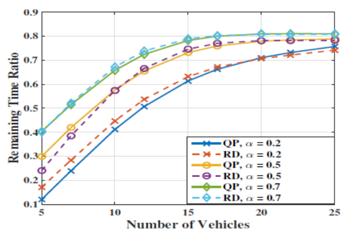


Figure 2.15: IEEE 802.11ad Radcom System Data Rate [2]

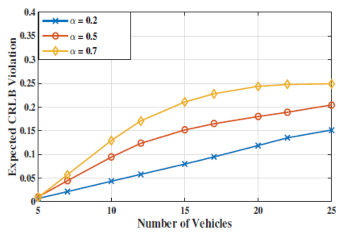
Methods	Coordination	Pros	Cons
Infrastructure-Based Methods	Coordinated	Easy to implement, efficient resource management, few changes on existing standards	Coordination overhead, construction and maintenance of control entities
Control channel-based methods	Coordinated	Low cost, efficient resource management, distributed implementation	Coordination overhead, reservation of a common control channel, synchronization
Distributed power control	Uncoordinated	Simultaneous spectrum access, no synchronization requirement, coordination-free	Spectrum sensing overhead, low efficiency due to lack of global information
Dynamic channel selection	Uncoordinated	Reliable protection of legacy users, local interference-free access, coordination-free	Spectrum sensing overhead, congestion on high-quality channels, power asymmetry
Adaptive frequency hopping	Uncoordinated	Fast discovery of free channels, flexible hopping set, protection of high-power devices	Spectrum discovery delay, unable to handle both internal and external interference
Listen-before-talk	Uncoordinated	Easy implementation, interference-free access, no synchronization, coordination-free	No fairness guarantee, low efficiency on handling inter-network coexistence
Spatial spectrum reuse	Uncoordinated	Simultaneous spectrum access, more practical modeling of interference	Potential hidden terminal problem, requires spatial information of coexisting devices
Fractional frequency reuse	Uncoordinated	Efficient inter-cell interference mitigation, coordination-free, no synchronization	Static spectrum allocation, only apply to dense homogeneous networks
Cognitive radio technology	Uncoordinated	Efficient reuse of licensed bands without affecting legacy users, coordination-free	Only apply to vertical spectrum sharing, spectrum sensing/discovery overhead
Sharing rule-based methods	Uncoordinated	Easy to implement, coordination-free, possible performance guarantees	Spectrum sensing overhead, require knowledge of neighbors before developing rules

Table 2.2: Spectrum Sharing Methods [13]

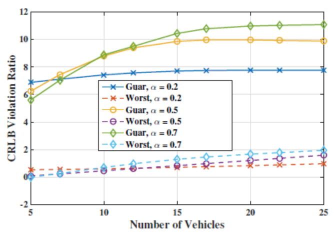
for communication. Here the optimization has been done to reduce the use of the resources. [42] provides a generic solution to the resource allocation problem by proposing sensor collaboration that considers the sensing environment. It optimizes the information gain based on sensor measurements while considering multiple sensors combined in a sensor network.



(a) Left Time Ratio for Other Communications



(b) Expected CRLB Violation Ratio



(c) Worst CRLB Violation and its Upper Bound

Figure 2.16: Optimal spectrum utilization in joint automotive radar and communication networks [12]

3
System Prototype

The following chapter develops the prototype used to build other models to solve the problem discussed in chapter 1. In an automotive scenario, when the vehicle has full awareness of the surroundings, it can take the best course of action in regards to human safety. To develop an awareness of the surroundings, maintaining a track of the different objects around the vehicle is generally used. Tracking can be done using several methods. In this chapter, we use a simple target, sensor, and tracking model. This tracking model is controlled using the prototype.

3.1. Target Model

A target model uses Newtonian mechanics to understand its movement. Complicated models of movement exist that consider higher-order derivatives of the laws of motion. However, the proposed system can be applied to any model that can be codified using a state dynamics equation which is discussed in the following sections. In this case, we assume a simple model for target movement. The target follows a constant velocity in two dimensions with noise.

3.1.1. Target Unknowns

Consider the case of tracking a target in two dimensions. The unknowns about the target are the positions and velocities in two dimensions. Let the dimensions be labeled x and y. The unknowns can be then characterized by the vector **s**. The vector **s** encapsulates all the required unknowns about the target as shown below. p_x and p_y refers to the position in the x and y dimensions and v_x and v_y refers to velocity in the x and y dimensions.

$$\mathbf{s} = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{3.1}$$

3.1.2. State Dynamics

As mentioned earlier, to demonstrate the proposed solution, the target is assumed to follow a constant velocity model. With this in mind, the state dynamics of the target are defined as shown below. The position of the target in a dimension changes based on the velocity in the corresponding dimension and the time interval. The velocities of the target are constant and hence do not change with time.

$$p_x^{'} = p_x + v_x \, \Delta t \tag{3.2}$$

$$p_y^{'} = p_y + v_y \, \Delta t \tag{3.3}$$

$$v_x^{'} = v_x \tag{3.4}$$

$$v_y^{'} = v_y \tag{3.5}$$

This can be described in matrix form as shown below and describes the dynamics of the target.

$$\begin{bmatrix} p_x^{'} \\ p_y^{'} \\ v_x^{'} \\ v_y^{'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{3.6}$$

Henceforth, the we define a matrix **F** as which relates the new state of the target $\mathbf{s}'$ to the old state **s** as shown in equation 3.8 (equation 3.6 in matrix form) :

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$\mathbf{s}' = \mathbf{F}\,\mathbf{s} \tag{3.8}$$

However, the dynamics of the system are known to have noise in the dynamics that can be characterized by a random process. This adds a noise term **w** to the system dynamics equation shown in equation 3.8, thereby making the system dynamics equation in noise:

$$\mathbf{s}' = \mathbf{F}\,\mathbf{s} + \mathbf{w} \tag{3.9}$$

The noise term **w** can be quantified in terms of a random process that is dependent on the time interval Δt. The covariance matrix of the random process is defined as shown in equation 3.10 with the term σ_w dependent on the target. For more information about how this is defined please refer to appendix C.

$$\mathbf{Q_w} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \frac{\Delta t^2}{2} & 0 & \Delta t & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 & \Delta t \end{bmatrix} \sigma_w^2 = \begin{bmatrix} \frac{\Delta t4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix} \sigma_w^2 \tag{3.10}$$

3.2. Sensor Model

With the dynamics of the system defined, we now address how the state of the target is measured. Assuming that a measurement of the position is provided by a sensor, we can define the measurement as shown below in relation to the target state. The measurement in the two dimensions are defined as m_x and m_y. This is an extremely unrealistic measurement model where independent measurements of the position in x and y dimensions is provided by the same sensor.

$$\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{3.11}$$

In matrix form we define the measurement vector as **y** to be related to the target state **x** by the matrix **H** defined in equation 3.12 and the measurement equation itself as shown in equation 3.13.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.12}$$

$$\mathbf{y} = \mathbf{H}\,\mathbf{x} \tag{3.13}$$

However, sensors cannot measure the target with perfect accuracy. The measurements themselves have noise. This results in a noise term **v** added to the measurement equation and results in a noisy measurement equation as shown in equation 3.14.

$$\mathbf{y} = \mathbf{H}\,\mathbf{s} + \mathbf{v} \tag{3.14}$$

The statistics of the noise are dependent on the sensor and the target properties. In the case of automotive radar, this would depend on the radar transceiver properties such as radiation pattern, transmit power, and noise figure and the target properties such as radar cross-section and distance from the radar. We, therefore, define the measurement noise to be independent in the two dimensions and as shown in equation 3.15

$$\mathbf{Q_v} = \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix} \frac{1}{\tau} \tag{3.15}$$

The term τ is the measurement time and is vital to control the sensor. If the sensor is given a longer time to measure the target, it can negate the effects of the noise using various techniques. Most commonly the Fast Fourier Transform can be used across multiple chirps of an FMCW radar can provide better results for measurements in noise if we consider the τ to be the number of chirps that the automotive radar transceives. It is important to note that using the variable τ we can control the accuracy of the sensor measurements.

3.3. Kalman Filter

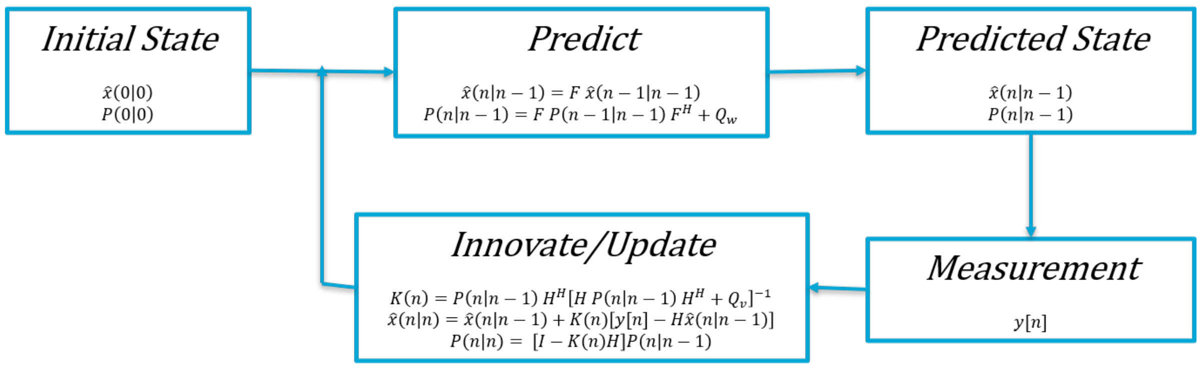

Initial State
$\hat{x}(0|0)$
$P(0|0)$
Predict
$\hat{x}(n|n-1) = F\,\hat{x}(n-1|n-1)$
$P(n|n-1) = F\,P(n-1|n-1)\,F^H + Q_w$
Predicted State
$\hat{x}(n|n-1)$
$P(n|n-1)$
Innovate/Update
$K(n) = P(n|n-1)\,H^H[H\,P(n|n-1)\,H^H + Q_v]^{-1}$
$\hat{x}(n|n) = \hat{x}(n|n-1) + K(n)[y[n] - H\hat{x}(n|n-1)]$
$P(n|n) = [I - K(n)H]P(n|n-1)$
Measurement
$y[n]$

Figure 3.1: Kalman Filter

In 1960, R. E. Kalman published his landmark paper, "A New Approach to Linear Filtering and Prediction Problems" [16]. It has had a huge impact on various applications. A Kalman Filter complements measurements of the target with known dynamics that the target is expected to follow. Thus for a system about which complete information is known, in terms of (a) the noise, (b) the system dynamics, and (c) the measurement model, the Kalman Filter can produce results quite close to the actual trajectory with very high accuracy. It is optimal for linear systems. However, it should be noted that this method is known to suffer from biases for non-linear systems. In this chapter, we demonstrate that a solution can be provided to the original research question using a simple model. The Kalman Filter provides a solution to estimate the unknown target state **x** when the state dynamics is described as shown in equation 3.9 and the measurement is described as shown in equation 3.14. These equations are rewritten in recursive form in equations 3.16 and 3.17. Here we extend the previous equations to a case with multiple time intervals. This implies that the state of the target depends on the state of the target in the previous time interval and the measurement is dependent on the current state of the target. This results in the introduction of the term n that denotes the current state and $n+1$ that denotes the next state or the predicted state.

$$\mathbf{s}[n+1] = \mathbf{F}\,\mathbf{s}[n] + \mathbf{w}[n] \tag{3.16}$$

$$\mathbf{y}[n] = \mathbf{H}\,\mathbf{s}[n] + \mathbf{v}[n] \tag{3.17}$$

The operation of the Kalman Filter can then be shown to be recursive and described as shown in figure 3.1. The filter starts with a initial estimate of the target state $\hat{\mathbf{x}}(\mathbf{0}|\mathbf{0})$ and error in the state $\mathbf{P}(\mathbf{0}|\mathbf{0})$. When this is completely unknown, the target state can be taken based on random values with large error covariance. The filter then predicts the next state of the target based on this initial estimate using the prediction equations for the state and the error as described in equations 3.18 and 3.19. This is referred to as the 'Predict' step of the process.

$$\hat{\mathbf{s}}(n|n-1) = \mathbf{F}\,\hat{\mathbf{s}}(n-1|n-1) \tag{3.18}$$

$$\mathbf{P}(n|n-1) = \mathbf{F}\,\mathbf{P}(n-1|n-1)\,\mathbf{F}^T + \mathbf{Q_w} \tag{3.19}$$

The filter then receives a measurement of the target which is used to update the prediction based on the new information. This is sometimes referred to as the 'Update' or the 'Innovate' step of the process. The step is then used to update the target state and the error as described in the equations. This new state is then used to predict the state for the next time interval.

$$\mathbf{K}(n) = \mathbf{P}(n|n-1)\,\mathbf{H}^T\left[\mathbf{H}\,\mathbf{P}(n|n-1)\,\mathbf{H}^T + \mathbf{Q_v}\right]^{-1} \tag{3.20}$$

$$\hat{\mathbf{s}}(n|n) = \hat{\mathbf{s}}(n|n-1) + \mathbf{K}(n)\left[\mathbf{y}(n) - \mathbf{H}\,\hat{\mathbf{s}}(n|n-1)\right] \tag{3.21}$$

$$\mathbf{P}(n|n) = [\mathbf{I} - \mathbf{K}(n)\,\mathbf{H}]\,\mathbf{P}(n|n-1) \tag{3.22}$$

From equation 3.20 we can see that the $\mathbf{Q_v}$ can be used to affect the accuracy of the trajectory. We also know that we can control the matrix $\mathbf{Q_v}$ using the parameter τ as described in equation 3.15. This parameter can therefore be used to optimize the error in the target trajectory as described in the following sections. It should be noted that the update step of the Kalman Filter can be done using multiple measurements within the same interval. Using measurements from different sources, the update step can be repeated multiple times, provided the measurement noise matrix $\mathbf{Q_v}$ is also adjusted. If we consider a set of M sensors updating the Kalman Filter each with a corresponding value for τ characterized as τ_i. Then the vector τ can be defined as shown in equation 3.23.

$$\boldsymbol{\tau} = \begin{bmatrix}\tau_1 & \tau_2 & \dots & \tau_M\end{bmatrix}^T \tag{3.23}$$

This update and predict process is done recursively for every time interval and the estimated state of the Kalman Filter converges to the actual trajectory of the target after multiple recursions even for bad initial estimates provided that the measurements are accurately described.

3.4. Optimization Problem

The algorithm developed is required to provide the best selection of the element of the vector τ that is defined in equation 3.23. In essence, the algorithm decides the sensing time for the M sensors considered in the optimization. It is therefore clear that the variable used in the optimization problem is the vector τ but the cost function and constraints are yet to be defined. The cost function helps evaluate the performance of choosing a particular set of sensing times and the constraint defines the conditions for the optimization. These are described in the following sections.

3.4.1. Cost Function

To optimize the accuracy of the track we would need to define a cost function that captures the performance of the tracking filter. This cost function would therefore vary based on the optimization variable τ. Since the target state itself is not of concern (with the simplistic measurement model) to as much as its accuracy the error in the state of the Kalman Filter is the first and most obvious choice for the cost function. Thus, to formulate this optimization problem, we consider the sum of the first two diagonal terms of the Kalman Filter Error **P**, as the cost function as described in equation 3.24. Other Cost functions can be formulated for different use cases. The dimensions of the Kalman Filter Error is a 4 by 4 matrix with the first two diagonal elements describing the error in the estimate of the position in the two dimensions. A good selection of the element of the vector τ will provide a low value of the cost and a bad selection of the elements of the vector will increase the overall cost function implying a worse tracking accuracy.

$$C(\boldsymbol{\tau}) = P_i^{11} + P_i^{22} \tag{3.24}$$

3.4.2. Constraints

The system cannot operate with unlimited resources, thus the operation time of the sensors needs to be within a pre-allocated budget, B. Further, it is assumed in the thesis as described in chapter 1 that no two sensors can operate simultaneously. Therefore the system needs to operate under the assumption that the summation of the sensing time should be equal to the total available budget, B. If the total time assigned is lower than the budget, the sensing performance would be worse. Therefore the constraint for the optimization problem is shown in equation 3.25 where the vector **1** is a vector with the same length as $\boldsymbol{\tau}$.

$$\mathbf{1}^T \boldsymbol{\tau} = B \tag{3.25}$$

3.4.3. General Optimization Problem

From equations 3.24 and 3.25 we can write the full optimization problem as shown in equation 3.26. We see that if the optimization minimizes the cost function as described 3.24 when the constraint defined in 3.25 is met, the overall set of M sensors would track the N target with the least error in the Kalman Filters.

$$\begin{aligned} \underset{\boldsymbol{\tau}}{\text{minimize}} \quad & C(\boldsymbol{\tau}) \\ \text{subject to} \quad & \mathbf{1}^T \boldsymbol{\tau} = B \end{aligned} \tag{3.26}$$

The optimization is solved using the Lagrangian relaxation as shown in equation 3.27. The maximization of the Lagrangian multiplier or the dual variable ensures that the constraint is satisfied while the Policy Rollout is used to find the optimal values for the elements of the vector $\boldsymbol{\tau}$ [33]. Refer to chapter 2 on the choice of the Policy Rollout to solve the optimization problem.

$$Z_D = \underset{\lambda}{max}\left(\underset{\boldsymbol{\tau}}{min}\left(C(\boldsymbol{\tau}) + \lambda\left(\mathbf{1}^T \boldsymbol{\tau} - B\right)\right)\right) \tag{3.27}$$

To perform the gradient descent, the first derivative of equation 3.27 with respect to the dual variable λ is done and the resulting update equation for next recursion of the descent is shown in equation 3.28 where $\lambda^{'}$ is the update dual variable and λ is the current dual variable. Δ is the step size for the recursion and is a design parameter. The recursion ends when the end condition defined as the gradient is close to zero within a degree of error ϵ.

$$\lambda^{'} = \lambda + \Delta\left(\mathbf{1}^T \boldsymbol{\tau} - B\right) \tag{3.28}$$

3.5. Policy Rollout

The Policy Rollout is used to search for the best possible sensing times for the M sensors in the optimization. It provides a great deal of control over the parameters and the use of the Kalman Filter helps make the solution non-myopic. The Policy Rollout defines a set action space to find the best action that minimized the cost function. The action space and the process to find the best action is detailed in the following sections.

3.5.1. Action Space

The Action Space is defined as all the possible actions that the optimization problem can take to minimize the cost function. In this case, the action space would be all combinations of sensing times that can be allocated to each of the sensors. For example, if the total sensing budget can be quantified using 2 chirps and there are 2 sensors involved. There are 8 possible actions.

Sensors	Actions							
Sensor 1	0	0	1	1	2	2	0	2
Sensor 2	0	1	0	2	1	0	2	2

Table 3.1: Simple Action Space

Although some of the actions defined in table 3.1 do not satisfy the constraints, it is still possible to choose one of these actions in the context of the sensing times. The Policy Rollout evaluates the cost associated with each of these actions by plugging in the values of the sensing times in the Kalman Filter measurement noise as described in 3.15 using the variable τ and calculating the cost using equation 3.24. The action space defined in table 3.1 is quite small and it is easy to calculate the cost of the actions. However, it is easy to see that the number of possible actions and the sensors decide the size of the action space. If the total number of actions is K in number, the actions and the action space are defined as follows.

$$A = \{a_1, a_2, a_3, ..., a_K\} \tag{3.29}$$

$$\left|A^M\right| = K^M \tag{3.30}$$

Consider all set of possible actions for each sensor as shown in equation 3.29, all the possible actions that a set of M similar sensors can take is the M-ary Cartesian power of the set A which can be defined as A^M where M denotes the total number of sensors in the optimization. The set A^M therefore defines the action space for the Policy Rollout. The size of this action space can be easily written as shown in equation 3.30. For 4 sensors and 128 possible actions, the action space size, therefore, becomes 268435456. This implies an algorithmic complexity of $O(n^2)$ using the Policy Rollout.

3.5.2. Reduced Complexity

To simplify the process in the Policy Rollout and bring down the complexity, we employ a tweak where we consider one sensor at a time and find the best action for that sensor only. This is done by calculating the cost using the best actions for the sensors that are not optimized from the previous recursion and then applying all possible actions for the optimization sensor to choose the best action for it. This is repeated for every sensor and this brings the algorithmic complexity to $O(n)$. This is a significant difference because each sensor has to evaluate K actions only. This has also been shown and implemented in [39].

Sensors	Actions		
Sensor	0	1	2

Table 3.2: Simple Action Space

The new action space for the example discussed earlier, therefore, reduces to the one shown in table 3.2. This is however repeated for every sensor in the optimization process.

3.5.3. Base Policy

The Policy Rollout starts with assumptions for the elements of the optimization variable vector τ. When this assumption is close to the solution of the optimization the number of recursions needed before the algorithm reaches the end condition also drops. This is a design choice but can also be complemented based on information from other sources such as machine learning models. For this thesis, we assume a base policy of equal resource allocation to all sensors. This implies that when all sensors provide information of equal importance the algorithm converges faster as compared to imbalances in the information quality of the sensors involved. This is because the solution is already closer to the expected result.

3.6. Full Algorithm

The overall process of the optimization is shown in figure 3.2. The process starts with an initial estimate of λ and calculates the best actions for each sensor using this value. Then the new set of best actions are used to calculate the gradient. If this gradient value is within ϵ the recursion ends, else the λ is updated based on equation 3.28. This process repeats until the end condition is reached. It should be noted that the tracking of the target is not implemented in the system prototype. The results of this chapter are proof-of-concept that such a system can be implemented where the cost defined by the error in the track can be optimized using the sensing times. The **Input** parameters of the algorithm are listed below. The algorithm uses this to provide the best allocation of sensing times (τ) across all sensors that would be used in the next time interval as **Output**.

1. **Action Space** : The action space needs to be passed to the algorithm for the Policy Rollout to evaluate the best action for each sensor. This is done by passing three parameters in the MATLAB implementation: (a) Sensing Space Size: Decides how the total Budget should be divided and (b) Total Budget: Total Sensing Time available for optimization.

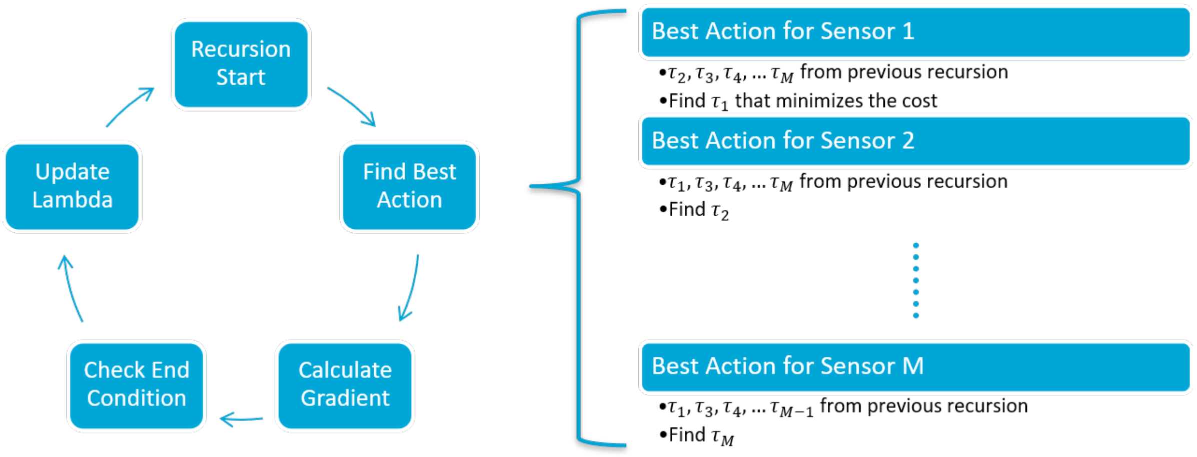

Recursion Start
Find Best Action
Calculate Gradient
Check End Condition
Update Lambda
Best Action for Sensor 1
•$\tau_2, \tau_3, \tau_4, \dots \tau_M$ from previous recursion
•Find τ_1 that minimizes the cost
Best Action for Sensor 2
•$\tau_1, \tau_3, \tau_4, \dots \tau_M$ from previous recursion
•Find τ_2
Best Action for Sensor M
•$\tau_1, \tau_3, \tau_4, \dots \tau_{M-1}$ from previous recursion
•Find τ_M

Figure 3.2: Algorithm

2. **Optimization Parameters** : The optimization parameters contain information required for the gradient descent. Parameters such: (a) Initial Lambda (λ), (b) Step Size (Δ), (c) Horizon Length and (d) Max Recursions are passed to the algorithm.

3. **Environmental Input** : The algorithm also requires information about the accuracy of the sensors and information about the target. Thus we pass (a) Measurement Variances (σ_x and σ_y), (b) Maneuverability Variance (σ_x), and the (c) Prior Error Covariance which captures the prior knowledge about the situation.

3.7. Results

This thesis doesn't consider single target or signal sensor scenarios as they are trivial and don't require any optimization. This is because: (a) In cases where there is one target, in a simple case, the target with the best accuracy can be given the full sensing time (b) For single sensor scenarios, no optimization is needed as all the sensing time is allocated to the single sensor. In this section, we look at the sensing times allocated by sensors based on the accuracy of the sensors involved. The prototype considers a single target being tracked by two sensors in a two-dimensional case. The two sensors have different accuracy in the two dimensions and the system allocates a budget based on the impact the information has on the prior information about the target. The scenario is described in figure 3.3.

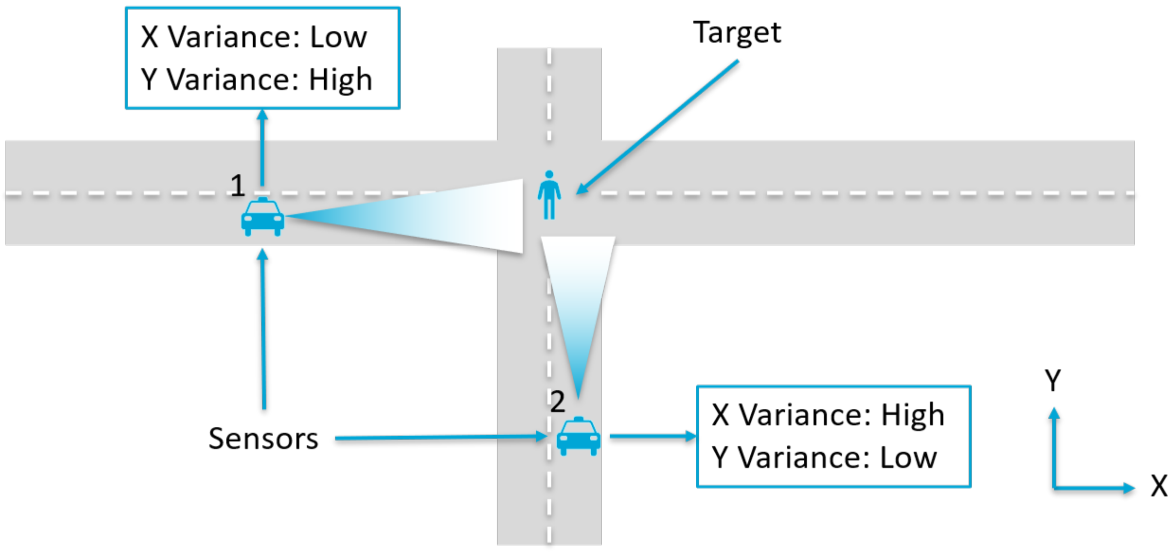


Figure 3.3: Scenario for Prototype

The prior information is described using the initial Error Covariance matrix $\mathbf{P_{init}}$. This matrix is randomized for the different simulation runs and the sensors are described using the measurement error matrix $\mathbf{Q_v}$. Both sensors has different measurement error matrices $\mathbf{Q_{v1}}$ and $\mathbf{Q_{v2}}$. For the first test, we keep $\mathbf{P_{init}}$ and $\mathbf{Q_{v2}}$ constant while increasing decreasing the accuracy of the sensor 1 using $\mathbf{Q_{v1}}$. The accuracy of the sensor is inversely related to the measurement noise. In this scenario, we run the simulation 20 times with increasing measurement noise. The simulation values are tabulated in table 3.3 and the results are shown in figure 3.5.

Variable	Value
P_{init}	$\begin{bmatrix} 84.48 & 0 & 0 & 0 \\ 0 & 26.80 & 0 & 0 \\ 0 & 0 & 22.80 & 0 \\ 0 & 0 & 0 & 29.30 \end{bmatrix}$
Q_{v_1} , Q_{v_2}	$\begin{bmatrix} var & 0 \\ 0 & 35 \end{bmatrix}, \begin{bmatrix} 20 & 0 \\ 0 & 5 \end{bmatrix}$

Table 3.3: Effect of Sensor Accuracy - Simulation Parameters

The values of var referred to in table 3.3 are show in figure 3.4 as $\diamond$. It is seen that with an increasing value of var, the weight allocated to sensor 1 is decreasing until it reaches a set minimum value. This is the smallest possible action that a sensor can take. Since there are only two sensors involved, the rest of the budget is automatically allocated to sensor 2 and this value is not shown to prevent redundancy. From this figure, we can attest that the algorithm correctly judges the value of the information provided by the sensors and only considers the impact of the information of the sensor on the error covariance for the optimization.

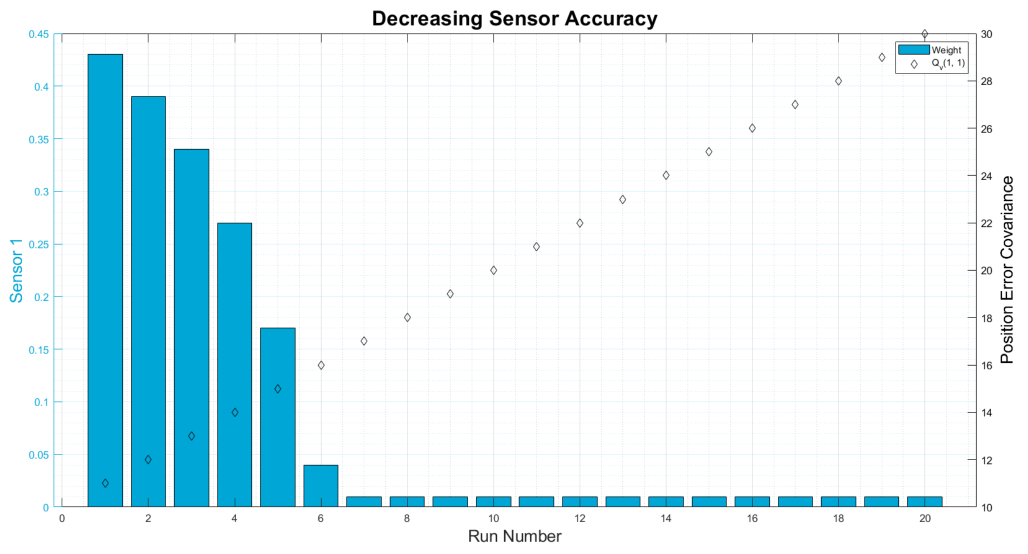


Figure 3.4: Effect of Sensor Accuracy - Results

For the second test, the sensor accuracy is made constant using $\mathbf{Q_{v1}}$ and $\mathbf{Q_{v2}}$ while the values of the initial error covariance matrix changes. The values used in the simulation are shown in table 3.4. It is important to note that Sensor 1 has better accuracy in the first dimension and Sensor 2 has better accuracy in the other dimension based on the measurement noise. Two simulation parameters are changed over the simulation runs that define the initial error in position in the two dimensions for the track. The values used in the simulations for the two variables in each dimension are shown in $\diamond$ and $*$ in figure 3.5. We can observe that as the initial error increases in the first dimension, the weight given to sensor 1 increases. This is to ensure a good overall track of the target. The opposite can be observed as well. As the tracking accuracy in the second dimension gets better, the weight given to the sensor 2 drops. The optimization algorithm, therefore, checks the impact of the information from each sensor in both dimensions on the initial error covariance and chooses weights that minimize the total error as defined in equation 3.24.

With the effect of the sensor accuracy and the initial error covariance of the filter documented, the final important result to discuss is shown in figure 3.6. As discussed in section 3.5.2, the need to optimize the solution for situations with a large number of sensors is necessary. To verify this approach, two simulation runs were performed using the same parameters for both algorithms: (a) Using all of the action combinations and (b) optimizing one sensor at a time. For the full code of all these simulations please refer to appendix B. The first simulation uses the action space as defined in table 3.1 while the second simulation use the actions as defined in table 3.2. Both simulations extend the action space size to 100.

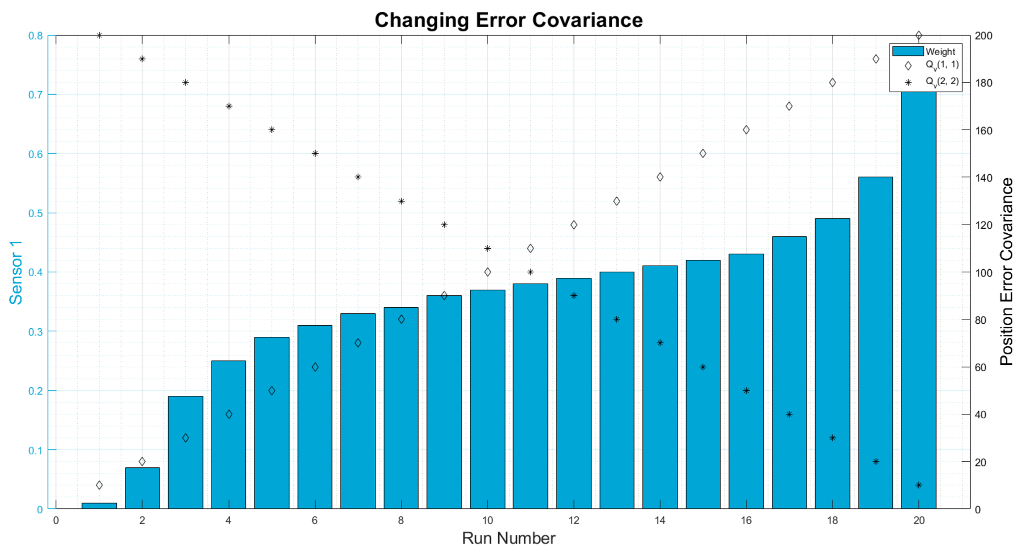


Figure 3.5: Effect of Initial Error Covariance - Results

The results of this simulation are shown in figure 3.6. It is clearly seen both algorithms converge to the same result within a degree of precision. However, the algorithm time is significantly reduced. Using all combinations of actions for two sensors using an action space of size 100, the algorithm time was: **45.93 seconds** for 20 simulation runs. However using the proposed approach of optimizing one sensor at a time, the algorithm time was: **1.99 seconds** for 20 simulation runs. This can be attributed to the fact that using the first approach, each recursion within a simulation run had to evaluate the cost 10000 times whereas using the second approach, each recursion evaluated the cost 200 times. This effect increases with the number of sensors and for a simulation with larger than five sensors, it would not be practical to use the first approach. The computation time would be too high. An action space size of 100 would require evaluating the cost 1e10 times for every recursion.

3.8. Conclusion

The results from chapter 3 considered a system running Kalman Filters to track targets using noisy Cartesian measurements from sensors and allocating sensing budgets to the sensors involved. The model assumed that: (a) The communication time for the measurement information was considered negligible. This meant that all information would be sent to the central system and the system would share this information with all sensors, both accurate and inaccurate, operating within it. (b) The system did not allow interference and allocated resources such that only one sensor could be operational at a time. Interference might cause sensors

Variable	Value
P_{init}	$\begin{bmatrix} var_1 & 0 & 0 & 0 \\ 0 & var_2 & 0 & 0 \\ 0 & 0 & 102.64 & 0 \\ 0 & 0 & 0 & 117.53 \end{bmatrix}$
Q_{v_1} , Q_{v_2}	$\begin{bmatrix} 10 & 0 \\ 0 & 35 \end{bmatrix}, \begin{bmatrix} 20 & 0 \\ 0 & 5 \end{bmatrix}$

Table 3.4: Effect of Initial Error Covariance - Simulation Parameters

to observe higher noise floors and thereby lose information about the target. This impact was considered to be too high. (c) The scenarios consider here were static. This system was able to allocate resources to sensors more efficiently based on the impact the information they provided had on its track accuracy. Thus this system was able to optimize the sensing times of the sensors involved. The system could improve the longevity of the sensors due to reduced usage. As sensors didn't always stay 'on', the thermal degradation of the sensors reduced. The key results of the simulations are listed below:

1. The system could optimally assign resources to sensors based on the impact they had on its track accuracy.
2. When sensors didn't have worthwhile information to provide, the system automatically assigned them the lowest possible resources. More accurate sensors were assigned more resources to observe the target and this led to better accuracy of the system.
3. The prior information about the targets played a role in the budget allocation for the sensors involved. In a scenario where the system had a good understanding of one target but had vague information about another target, the system assigned high resources to a sensor that could provide information about the second target despite other sensors providing better information overall.

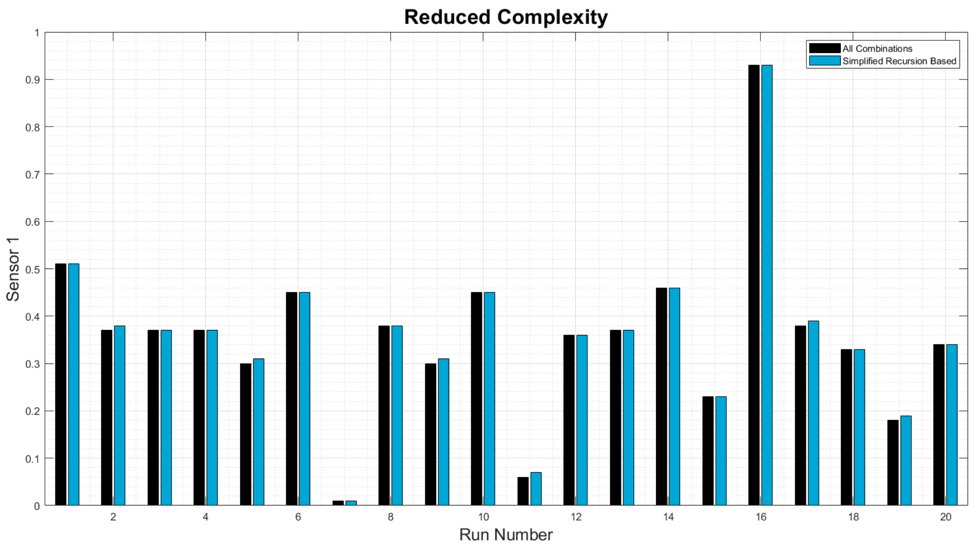


Figure 3.6: Reduced Complexity - Results

4
Sensing Time Optimization

Chapter 3 demonstrated that using the tracking accuracy as a cost function can help algorithms choose actions that sensors can take to track targets. This solution provided did not however actually track targets but worked with only the initially Error Covariance Matrix. The following chapter deals with the implementation of a more realistic tracking scenario using an Extended Kalman Filter to consider measurements taken in polar coordinates by a set of sensors. The target model remains the same as the one discussed in section 3.1. However, this chapter extends it to N targets.

4.1. Sensor Model

As discussed in section 3.2, the prototype assumed that the sensors were static. Chapter 3 developed a system that did not estimate the target state. Its results were based on the Error Covariance Matrix. This negligence of state estimation meant that the position of the sensor was not of concern, only the error in its measurements was important. In this section, the state estimation is performed. The position of the sensors is important for the central system to fuse the data from different radar nodes. This requires that the central system know the positions of the sensor. This is now assumed.

4.1.1. Sensor Dynamics

The sensors are not treated as targets and are assumed to move using the same constant velocity model as defined for the targets. Assuming the range m_r and azimuth m_θ of the target is estimated by the sensors, the data is fused into the filter using the relations shown in equation 4.1 where g_x and g_y are the two-dimensional positions of the radar node. The central system receives the measurements done by the radar nodes for each target, offsets these measurements based on the sensor position to obtain measurements in the global coordinates. This way information from different sensors can be fused considering the context in which the measurements were made. The sensors label the measurements that they provide uniquely such that the central system can combine the data using the tracking algorithm.

$$\mathbf{y} = \begin{bmatrix} m_r \cos\theta - g_x \\ m_r \sin\theta - g_y \end{bmatrix} \tag{4.1}$$

4.1.2. Polar Measurements

Although the target model does not change, the sensor is now assumed to take measurements of the target position in polar coordinates. The measurements taken by the sensors are related to the target state based on equations 4.2 and 4.3. Since this is not a linear relation, the tracking algorithm needs to be updated.

$$m_r = \sqrt{p_x^2 + p_y^2} \tag{4.2}$$

$$m_\theta = \tan^{-1}\left(\frac{p_y}{p_x}\right) \tag{4.3}$$

The noise in the polar measurements is now estimated based on the signal-to-noise ratio of the target. The signal strength is defined using the radar relationship as defined in equation 2.1 where P_t refers the to transmitted power, G_t and G_r refer to the gain in the transmitter and receiver, λ refers to the central wavelength of the transmitted signal, σ is the Radar Cross Section (RCS) of the target and R refers to the range of the target. The noise strength is defined based on the equation 2.1 where k refers to the Boltzmann constant, T and B refer to the operating temperature and the Bandwidth and F refers to the Noise Figure. The signal to noise (SNR) ratio can then be defined as shown in equation 2.1.

$$P_R = \frac{P_T G_T G_R \lambda^2}{(4\pi)^3 R^4 L}\sigma \qquad N = kTBF \qquad \text{SNR} = \frac{P_R}{N} \tag{2.1 revisited}$$

Two methods can be then be used to define the accuracy of the measurement based on the signal-to-noise ratio. The first method calculate the accuracy in range based on the fact that, for FMCW Radars, accuracy is defined as shown in equations 4.4 and 4.5 where δR and $\delta\theta$ are the resolution in range and azimuth. In these equations c refers to the speed of light, B refers to the bandwidth, N refer to the number of virtual element and s refers to the spacing between elements. Since we assume an isotropic antenna, θ is considered to be zero. The second method assumes a reference accuracy (δ_0) at a particular range and calculates the accuracy based on different RCS and the Range of the target based on equation 4.6.

$$\delta R = \frac{\Delta R}{\sqrt{SNR}}, \ \Delta R = \frac{c}{2B} \tag{4.4}$$

$$\delta\theta = \frac{\Delta\theta}{\sqrt{SNR}}, \ \Delta\theta = \frac{0.886}{(N-1)s\cos\theta} \tag{4.5}$$

$$\delta = \delta_0 \frac{R^2}{\sqrt{RCS}} \tag{4.6}$$

4.2. Extended Kalman Filter

Extended Kalman Filters are used in scenarios where there are non-linearities in the state dynamics or the measurement equations defined by equations 3.16 and 3.17. Matrices **H** and **F** cannot be defined using linear relationships in many cases and an Extended Kalman Filter helps build a solution in these scenarios.

$$\mathbf{x}[n+1] = \mathbf{F}\,\mathbf{x}[n] + \mathbf{w}[n] \tag{3.16 revisited}$$

$$\mathbf{y}[n] = \mathbf{H}\,\mathbf{x}[n] + \mathbf{v}[n] \tag{3.17 revisited}$$

One of the simplest solutions to solving the non-linearity problem is to linearize the equation. The Extended Kalman Filter does this using by calculating the Jacobian of the non-linear relationship. Since the targets are assumed to move using the same model, the state dynamics equation behaves in the same way as discussed in chapter 3. Hence, the matrix **F** does not change. However, we now assume that the sensor takes polar measurements as discussed in section 4.1, of the target reflective of automotive radar sensors. In this case, the measurements taken by the sensor are related to the target using the relations shown in equations 4.2 and 4.3. Applying the Jacobian operator to this relationship provides a variable **H** matrix. The Jacobian relationship is shown in equation 4.7. Evaluating the partial derivatives results in the terms of the matrix that is dependant on the target state as shown in equation 4.8.

$$\begin{bmatrix} m_r \\ m_\theta \end{bmatrix} = \begin{bmatrix} \frac{\partial m_r}{\partial p_x} & \frac{\partial m_r}{\partial p_y} & \frac{\partial m_r}{\partial v_x} & \frac{\partial m_r}{\partial v_y} \\ \frac{\partial m_\theta}{\partial p_x} & \frac{\partial m_\theta}{\partial p_y} & \frac{\partial m_\theta}{\partial v_x} & \frac{\partial m_\theta}{\partial v_y} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix} m_r \\ m_\theta \end{bmatrix} = \begin{bmatrix} \frac{p_x}{p_x^2+p_y^2} & \frac{p_y}{p_x^2+p_y^2} & 0 & 0 \\ \frac{-p_y}{\sqrt{p_x^2+p_y^2}} & \frac{p_x}{\sqrt{p_x^2+p_y^2}} & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{4.8}$$

$$\mathbf{H} = \begin{bmatrix} \frac{k_{px}}{k_{px}^2 + k_{py}^2} & \frac{k_{py}}{k_{px}^2 + k_{py}^2} & 0 & 0 \\ \frac{k_{py}}{\sqrt{k_{px}^2 + k_{py}^2}} & \frac{k_{px}}{\sqrt{k_{px}^2 + k_{py}^2}} & 0 & 0 \end{bmatrix} \tag{4.9}$$

Since the target position is not directly available, the **H** is created using the last best known state of the target. This matrix is calculated after every time interval to update the Extended Kalman Filter. Therefore the measurement matrix **H** is now defined based on equation 4.9 where the values of k_{p_x}, k_{p_y}, k_{v_x} and k_{v_y} are the estimates of the target position and velocity based on the current information.

$$\hat{\mathbf{x}}(n|n-1) = \mathbf{F}\,\hat{\mathbf{x}}(n-1|n-1) \tag{3.18 revisited}$$

$$\mathbf{P}(n|n-1) = \mathbf{F}\,\mathbf{P}(n-1|n-1)\,\mathbf{F}^T + \mathbf{Q_w} \tag{3.19 revisited}$$

The update and the predict steps of the Extended Kalman Filter remain the same as discussed in section 3.3. The prediction of the state and the error is defined in equations 3.18 and 3.19. The definition of matrix **H** in equation 4.9 and the measurement y defined in the equation 4.1 are used to update/correct the filter.

$$\mathbf{K}(n) = \mathbf{P}(n|n-1)\,\mathbf{H}^T\left[\mathbf{H}\,\mathbf{P}(n|n-1)\,\mathbf{H}^T + \mathbf{Q_v}\right]^{-1} \tag{3.20 revisited}$$

$$\hat{\mathbf{x}}(n|n) = \hat{\mathbf{x}}(n|n-1) + \mathbf{K}(n)\left[\mathbf{y}(n) - \mathbf{H}\,\hat{x}(n|n-1)\right] \tag{3.21 revisited}$$

$$\mathbf{P}(n|n) = [\mathbf{I} - \mathbf{K}(n)\,\mathbf{H}]\,\mathbf{P}(n|n-1) \tag{3.22 revisited}$$

4.3. Dynamic Optimization

To consider multiple targets, the cost function of the optimization assumes that all targets are equally important. It sums the cost for the error in target trajectory across multiple targets to define the cost of the optimization as defined in equation 3.24. Here equation 4.10 refers to the cost of the tracking the i^{th} target. Equation 4.11 therefore defines the cost for tracking N targets. Minimizing this cost will result in a total decrease in the overall tracking error of the system.

$$C_i(\boldsymbol{\tau}) = P_i^{11} + P_i^{22} \tag{4.10}$$

$$C(\boldsymbol{\tau}) = \sum_{i=1}^{N} C_i(\boldsymbol{\tau}) \tag{4.11}$$

With the new definition for the cost function, the optimization problem can now be formulated as shown in equation 4.12. It is important to note that the sensing time is shared across targets within a sensor. Since the sensor has a full Field-Of-View, when the sensor is on, measurements of the positions of all targets are taken simultaneously. More sensing time implies more accurate measurements across all targets that a radar node can observe.

$$\begin{aligned} \underset{\boldsymbol{\tau}}{\text{minimize}} \quad & \sum_{i=1}^{N} C_i(\boldsymbol{\tau}) \\ \text{subject to} \quad & \mathbf{1}^T\,\tau = B \end{aligned} \tag{4.12}$$

The optimization problem is solved using the Lagrangian relaxation to satisfy the constraints and the policy rollout to decide the sensing times as discussed in chapter 3. The solution produced by the system now optimizes the tracking accuracy of N targets by observed by M sensors. Henceforth, the reduced complexity algorithm is used as defined in section 3.5.2 since its efficacy has been proven in chapter 3.

4.4. Results

Complete details of the simulation parameters are provided in appendix A. The first simulation assumes two sensors observing two targets, both targets and sensors are static. The simulation is for 20 time intervals. For each time interval, the number of total FMCW chirps is 128. The sensors each are allotted a certain number of chirps which are processed coherently before being sent to the central system. The system uses this information to update the filters that are tracking the targets. Figure 4.1 shows the locations of the sensors and targets. The sensors are marked using Diamonds and the target measurements are marked using black points. The estimation of the target position by the filter is marked using colored circles. It is clear from the diagram that sensor 1 is much further from the targets compared to sensor 2. Since the target SNR is based on the range, we can say that sensor 2 has a better understanding of the target positions. The results of the optimization are shown in figure 4.2. In this chapter the targets have the same RCS value of $150m^2$.

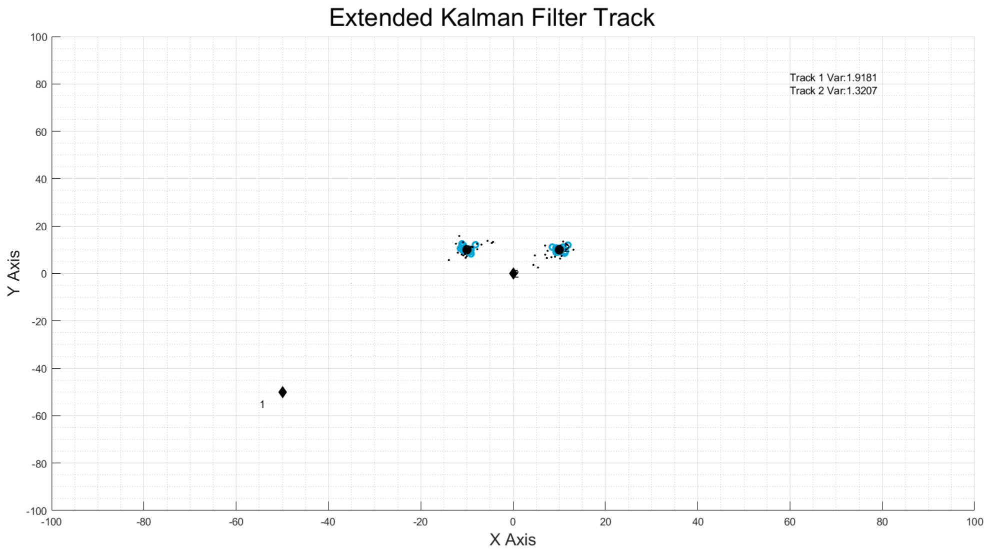


Figure 4.1: Scenario 1 for Sensing Model

The first 10 time intervals are not optimized. The Kalman Filters are run for this duration to let them achieve a steady state. Running the optimization without a steady-state might result in incorrect results for the weights. The results show that the optimization program starts allocating more budget to sensor 2 as it realizes that the accuracy of measurements from it is lower. The more the optimization is run, the more confidence the optimization has that sensor 2 will provide more information. This is reflected in the result. In this scenarios a basic case if considered to demonstrate how the optimization might work and the simulation results after 20 seconds are not considered. The simulations will reach a steady value after several time steps further into the simulation.

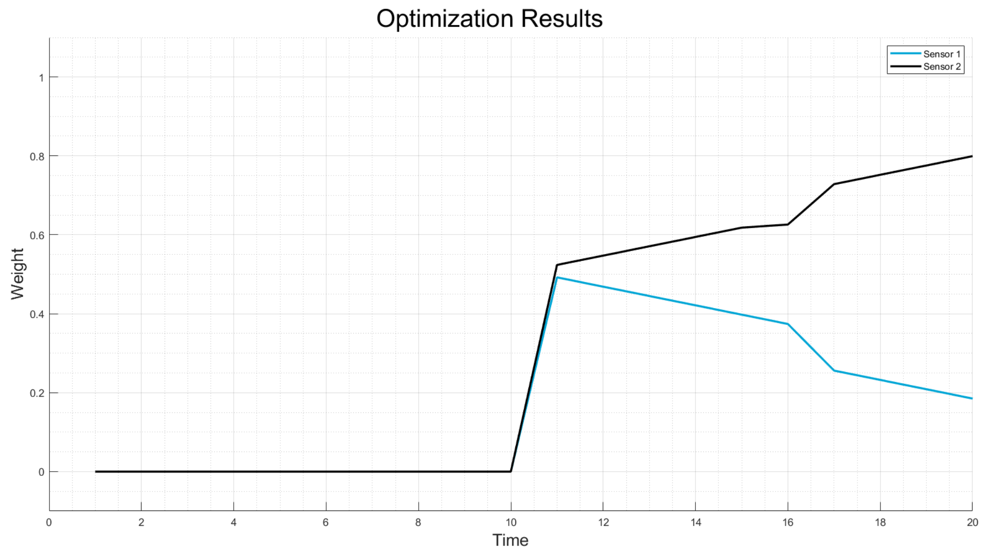


Figure 4.2: Results of Scenario 1

Henceforth, the result of the optimization is not shown when the steady-state of the Kalman Filter is not reached. In the second scenario, we assume that the targets are moving, while the sensors are stationary.

This actual trajectory of the target is shown in figure 4.3 as the black line. The measurement of the sensors are shown using black dots and the trajectory the Filter take are shown using the colored circles. One target is moving away from sensor 1 diagonally while the second target is moving vertically upwards. The implies that initially, both sensors have accurate information about one target. But as the target move, sensor 2 starts providing more accurate measurements as the targets are closer to it. The results of this simulation are shown in figure 4.4.

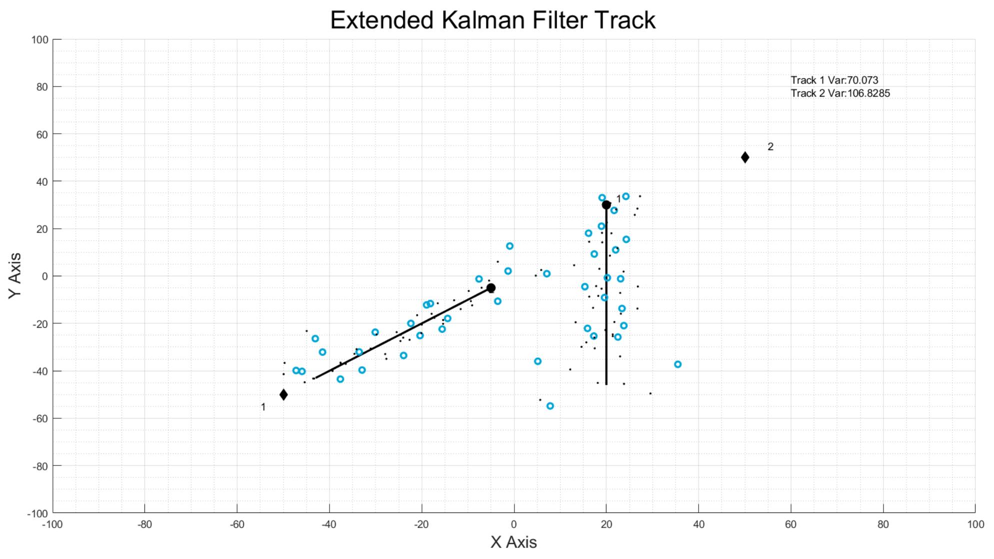


Figure 4.3: Scenario 2 for Sensing Model

The results are as expected. Initially, both sensors are given approximately the same weight. But as the simulation continues, sensor 2 starts taking up more sensing time. As the targets are moving towards sensor 2, the measurements it makes are more accurate because of lower SNR. Therefore if sensor 2 is allowed to transmit and coherently process more chirps, the more accurate the measurement is and this results in an overall better track.

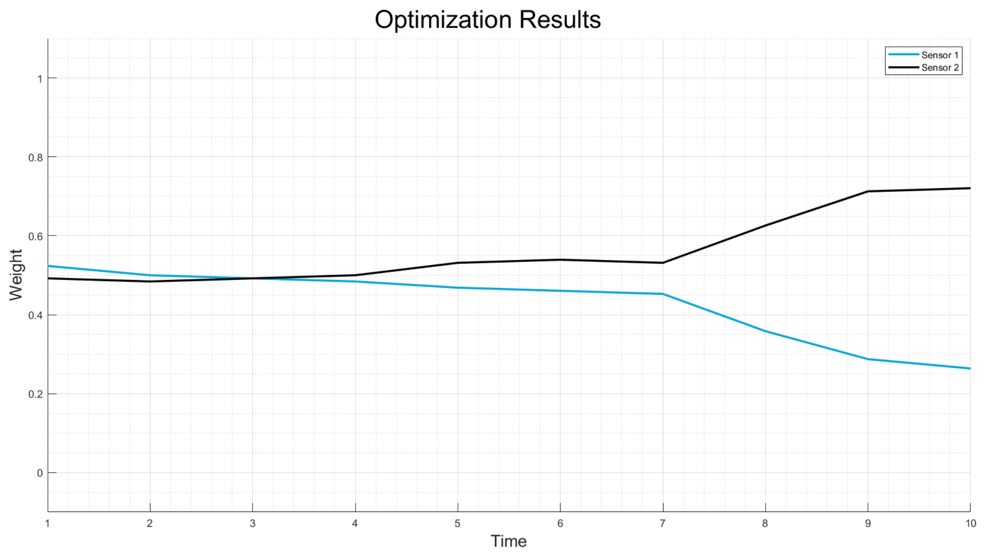


Figure 4.4: Results for Scenario 2

Scenario 3 assumes that the sensors are moving while the targets are stationary as shown in figure 4.5. The sensors are moving diagonally, sensor 2 starts near the origin as have better measurements but starts to move away from the targets, however, sensor 1 is closer to the target at the end of the simulation. This results in a slow shifting of weight from sensor 2 to sensor 1 as shown in figure 4.6. As sensor 1 moves towards the targets, we see more sensing time being allocated to it. The final scenario simulates both sensors and targets moving as shown in figure 4.7. For comparison, the simulation is repeated with both sensors receiving equal sensing times in figure 4.8.

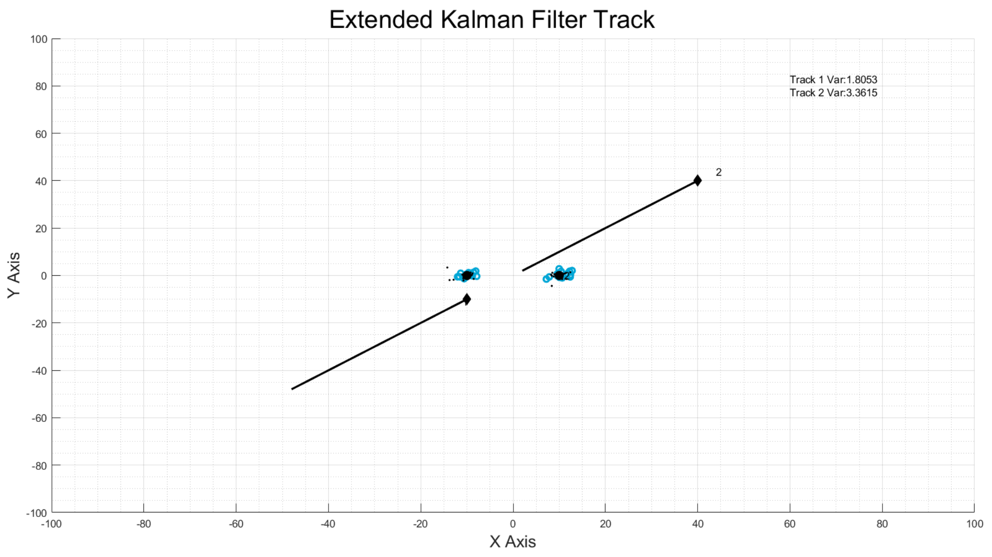


Figure 4.5: Scenario 3 for Sensing Model

Sensor 1 is moving diagonally while sensor 2 is moving vertically upwards. The two targets are moving away from the origin horizontally in opposite directions. It can be seen that initially sensor 2 has a better understanding of the target positions and is given higher sensing time. However, as the simulation progresses, the targets are closer to sensor 1 and it starts receiving more weight. The tracking error is also mentioned for each target on the top right side of the graph.

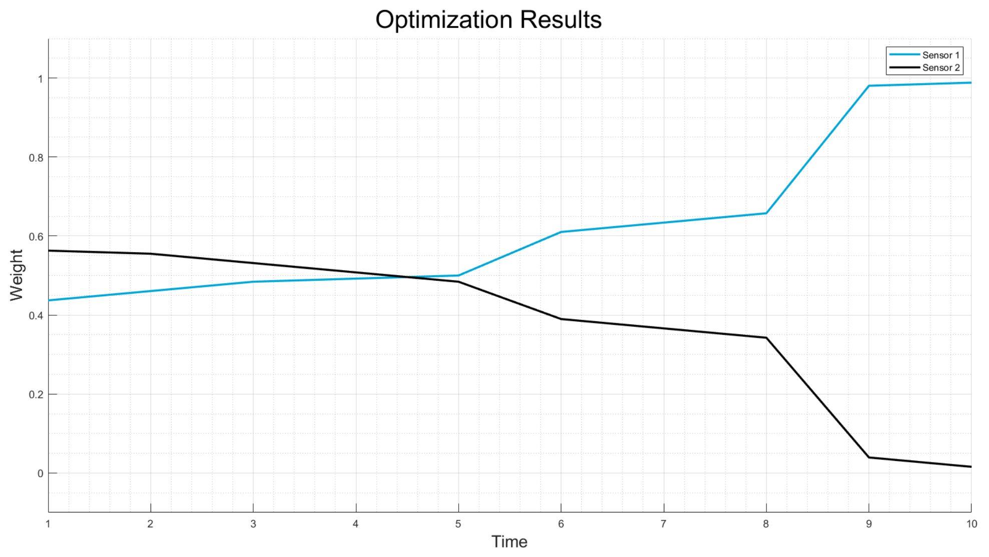


Figure 4.6: Results for Scenario 3

The scenario discussed in this chapter demonstrated that sensors can be optimized to sense targets. It is important to note the results of scenario 4 where a comparison was done with a system that always assigned equal budgets to the targets. They have been shown in table 4.1. With equal budgets, the track errors for the

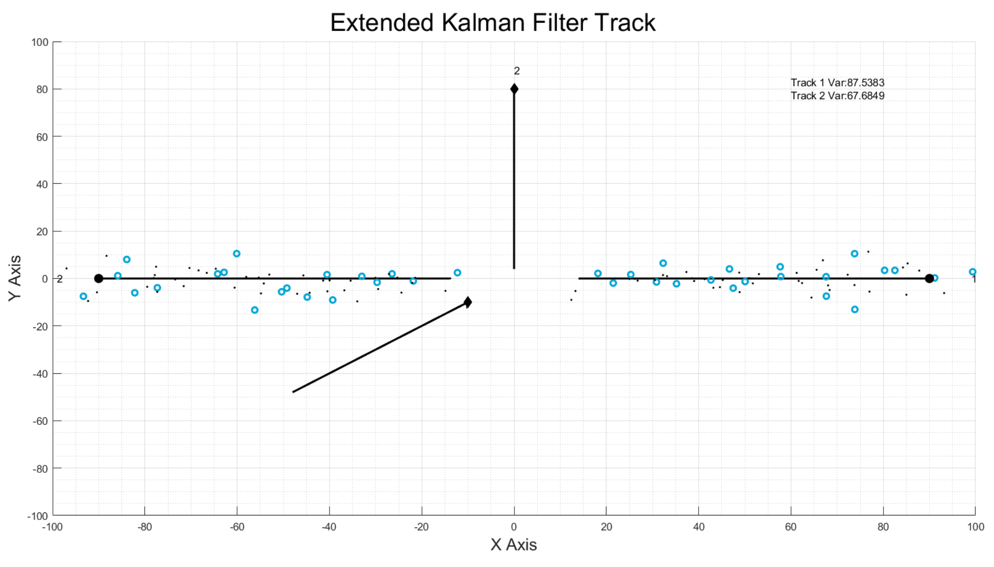


Figure 4.7: Scenario 4(a) for Sensing Model

Allocation	Track 1	Track 2
Equal Budgets	$72.14m^2$	$144.21m^2$
Optimized Budgets	$87.53m^2$	$67.68m^2$

Table 4.1: Tracking Accuracy of Scenario 4

two targets were $72.14m^2$ and $144.21m^2$. With the optimized approach, the errors were $87.53m^2$ and $67.68m^2$. This amounts to a 30% reduction in the tracking error.

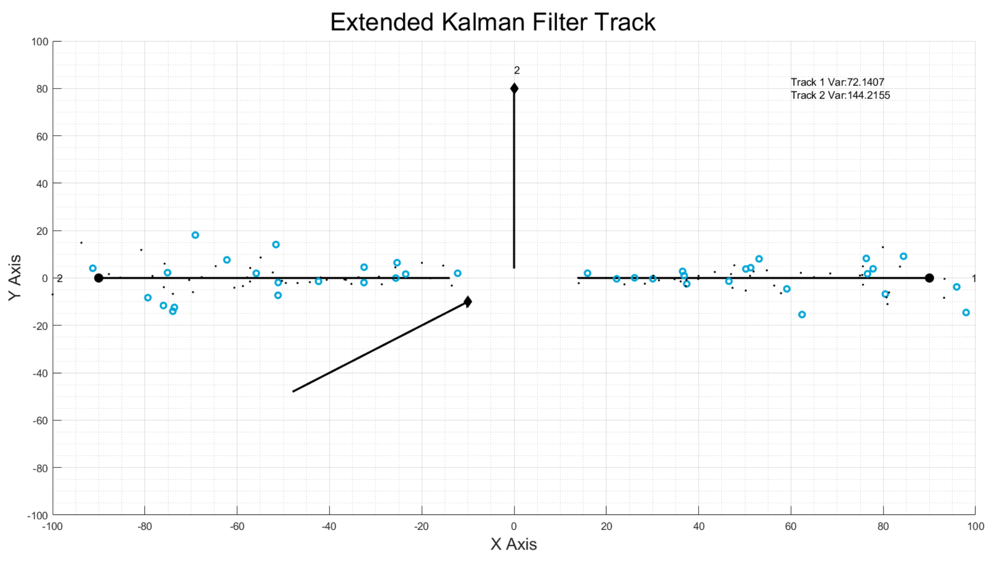


Figure 4.8: Scenario 4(b) for Sensing Model

4.5. Conclusion

Chapter 4 extended the results of the Kalman Filter to a more realistic measurement model in polar coordinates. This solution has the same assumptions as of the prototype but could (a) automatically calculate the accuracy of measurement based on the signal-to-noise ratio and provided a more realistic use case in an automotive environment. (b) The model also extended the simulation to dynamic cases where the optimization might be run repeatedly based on moving sensors and targets within a constant velocity model. The model could accurately track the targets and adapt to changing environments. While the previous chapter's simplistic model helped prototype the solution, this model could be considered the first real use case for the system. Some interesting results worth noting are:

1. The system automatically assigned resources to the sensors based on the predicted information they might provide. This prediction horizon was configurable.
2. The system was able to accurately track objects in the Cartesian plane using range and angle information from the sensors and combining the information to have a better understanding of the environment.
3. Despite certain sensors not being allocated resources, they had a better understanding of the environment based on the information provided to them by the system.
4. Compared to a system where there was no cooperation between sensors, the system was able to provide information about targets that might not be observed by certain sensors using the information provided by other sensors.

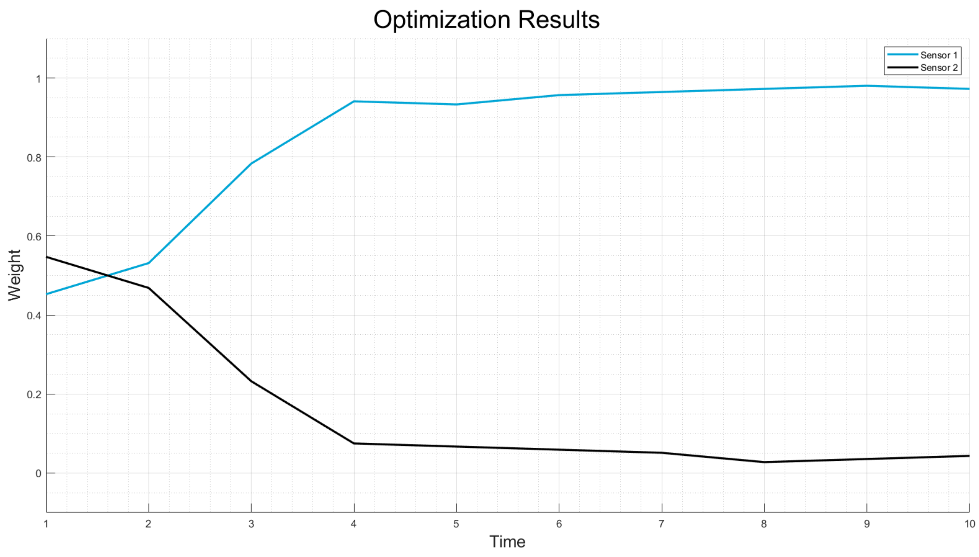


Figure 4.9: Results for Scenario 4

5
Communication Selection Optimization

In the previous chapters, the communication was assumed to happen using a second channel with unlimited bandwidth. No constraints were placed on the amount of communication done by the radar nodes and the system. In this chapter, we introduce this constraint and look at results in a scenario with multiple targets and sensors. The target, sensor, and filter operation remains the same. However, the optimization algorithm is updated.

5.1. Selective Communication

Chapter 2 provided evidence for the advantage of using the same channel for sensing and communication. To accommodate for such an operation, the optimization problem requires to be changed to consider the new constraint. The need for selective communication can be derived from the fact that not all information a sensor provides is accurate. Inaccurate information uses scarce communication bandwidth with little impact on the track performance. This chapter formulates the use of the same bandwidth for sensing and communication. The constraint in the optimization problem takes this into account. If the effect the information has on the filter is considered negligible, the system can automatically decide what information is not to be communicated across radar nodes and to the system.

5.1.1. Communication Variable

To consider the communication time used by the sensor, we assume that the time taken for communication about one target by one sensor is f. This would be a constant defined by the communication process, predominantly the data load. To consider the possibility of communication, we define a binary variable T. T can only be 0 which corresponds to no communication or 1 which means that the data is communicated. It is assumed that every sensor has information about every target in the simulations. In cases where a sensor cannot detect the target, a solution could be formulated where the sensor assumes the target position with extremely high error and this cause the filter to neglect information it provides. In a scenario with N targets, every sensor has N selection variables one for each target. If there are M sensors in the scene, the total number of selection variables are considered to be $M \times N$. Thus a matrix $\mathbf{T}$ can be defined as shown in equation 5.1 that describes all the variables related to the communication process.

$$\mathbf{T} = \begin{bmatrix} T_{1,1} & T_{1,2} & T_{1,3} & \dots & T_{1,M} \\ T_{2,1} & T_{2,2} & T_{2,3} & \dots & T_{2,M} \\ T_{3,1} & T_{3,2} & T_{3,3} & \dots & T_{3,M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T_{N,1} & T_{N,2} & T_{N,3} & \dots & T_{N,M} \end{bmatrix} \tag{5.1}$$

The total time need for communication can therefore be considered in the constraint using this variable as shown in equation 5.2. The term $\mathbf{1}^T\mathbf{T}\mathbf{1}f$ refers to the total time needed for communication and as discussed previously $\mathbf{1}^T\boldsymbol{\tau}$ refers to the total sensing time. Therefore the optimization problem can now consider both sensing and communication time using the same constraint. This implies that the system has an overall budget for communication and sensing. This budget should now be assigned to both sensing and communication tasks at each radar node. It is important to note that the communication variable is defined per target per sensor whereas the sensing variable is defined per sensor and is shared across targets.

$$\begin{aligned} \underset{\boldsymbol{\tau}}{\text{minimize}} \quad & \sum_{i=1}^{N} C_i(\boldsymbol{\tau},\mathbf{T}) \\ \text{subject to} \quad & \mathbf{1}^T\boldsymbol{\tau} + \mathbf{1}^T\mathbf{T}\mathbf{1}f = B \end{aligned} \tag{5.2}$$

5.1.2. Effect on Cost Function

The variable **T** is used to decide if the filter is updated with the information from the radar node. The effect on the cost function is described using the flowchart in figure 5.1. This describes the process of updating the filter which is repeated for every target in the scene.

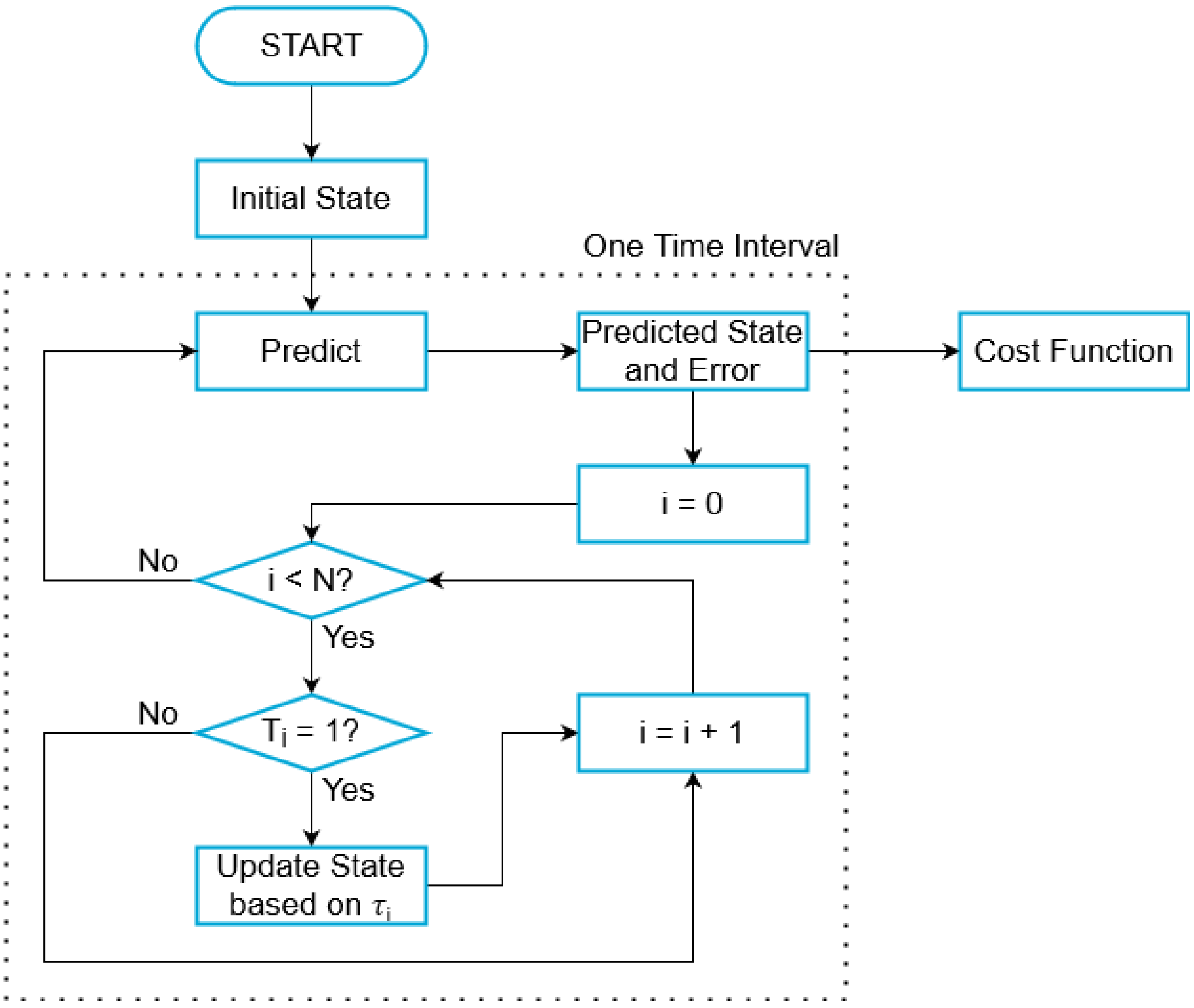


Figure 5.1: Communication Selection

Therefore the cost function is dependent on the values of the matrix **T**. Using this effect, we can allow the matrix containing binary elements to decide what information is useful to the filter tracking the targets. If the cost for communication f (communication budget) is high, the algorithm is more selective with regard to the communication. If this value is low, all information is communicated despite its minimal effect on the cost. Applying this additional variable does not change the target, sensor, or filter operation. They remain the same as discussed in chapter 4. As described in figure 5.1, the update of the filter is just more selective to accommodate for the cost of communication within the given budget. This budget allocation for the communication follows the same principles of prediction as discussed in the sensing time optimization. The algorithm considers the effect of a communication action multiple steps into the future.

5.2. Results

The results of the algorithm are discussed in this section in two sections. First, we look at static problems where the targets and the sensors are not moving. We observe the behavior of the algorithm using these simplified cases as simple inferences cannot be made in more complex dynamic scenarios.

5.2.1. Static Results

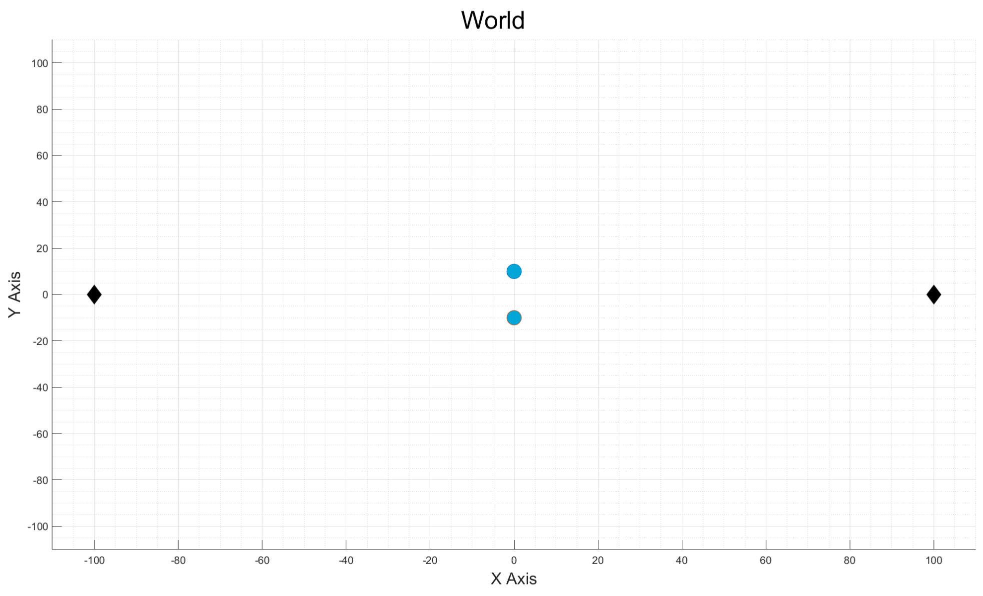


Figure 5.2: Communication Selection - Scenario 1

The first scenario we consider is shown in figure 5.2. Here we have two Radar Nodes denoted by black filled circles and two blue circles that show the location of the targets. The targets are located at (0, 10) and (0, -10) while the sensors are located at (-100, 0) and (100, 0). In this scenario, both the targets and the Radar Nodes are quite close to each other. Each target is at a distance of approx. 100 m from both Radar Nodes. Both Radar Nodes are symmetrically placed and therefore neither sensor provides any information that the other cannot provide. Further, since the Radar Nodes are equidistant from the sensor, information from both nodes is approximately equally accurate. The algorithm can therefore choose either node and use information from both to track the targets. The results generated by the proposed solution are shown in figure 5.3. The bar chart on the left shows the relative sensing budgets for each Radar Node. The right graph is split into four squares, this defines the communication strategy. If the square is shaded blue, it denotes that information about the target is requested from the Sensor. The values inside each box represent the accuracy of the measurements of position in the two dimensions. Low values represent accurate measurements and high values imply vague measurements. In this case, all information is requested by the central system and it is used to update the Extended Kalman Filter track. This can be easily explained because all information is accurate and this information can significantly reduce the previously defined cost function. In this case, the cost of communication is considered to be worth it, because of how the update step of the filter can decrease the tracking error. This can be considered the simplest relevant case for the thesis as the single target or single sensor systems are considered trivial.

For the second scenario, we consider two sensors and two targets placed as shown in figure 5.4. One sensor is quite close to both targets as compared to the other. In this case, the accuracy of the measurements for the first sensor is therefore low. The second sensor however due to its proximity to the targets has a much lower measurement noise. The accuracy of position measurements in the two axes is shown on the graph on the right in figure 5.5. It should be noted that the sensing allocation and the individual track accuracy are

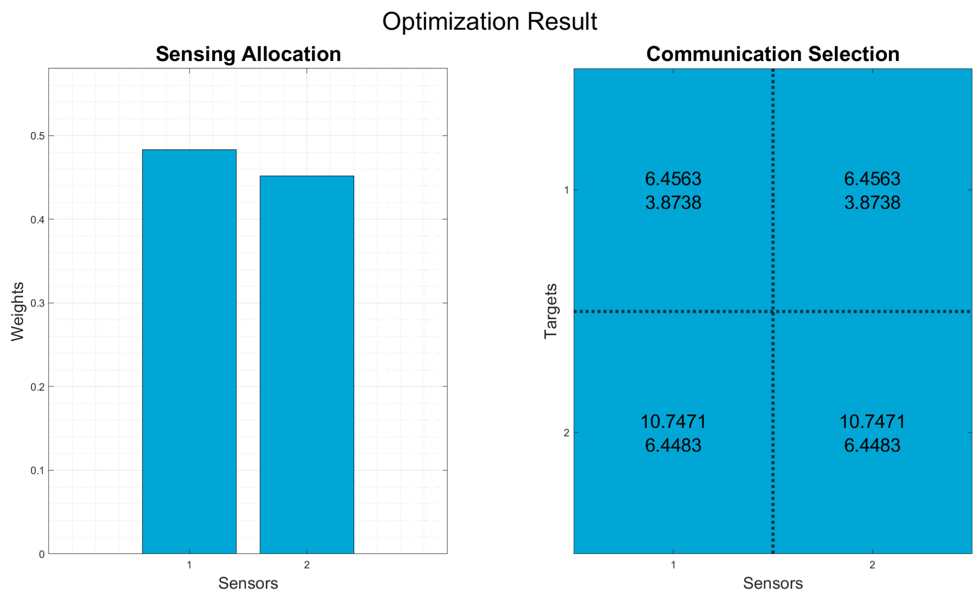


Figure 5.3: Communication Selection - Scenario 1 - Result

affected by the RCS of the targets. In this case however, the target have the same RCS value.

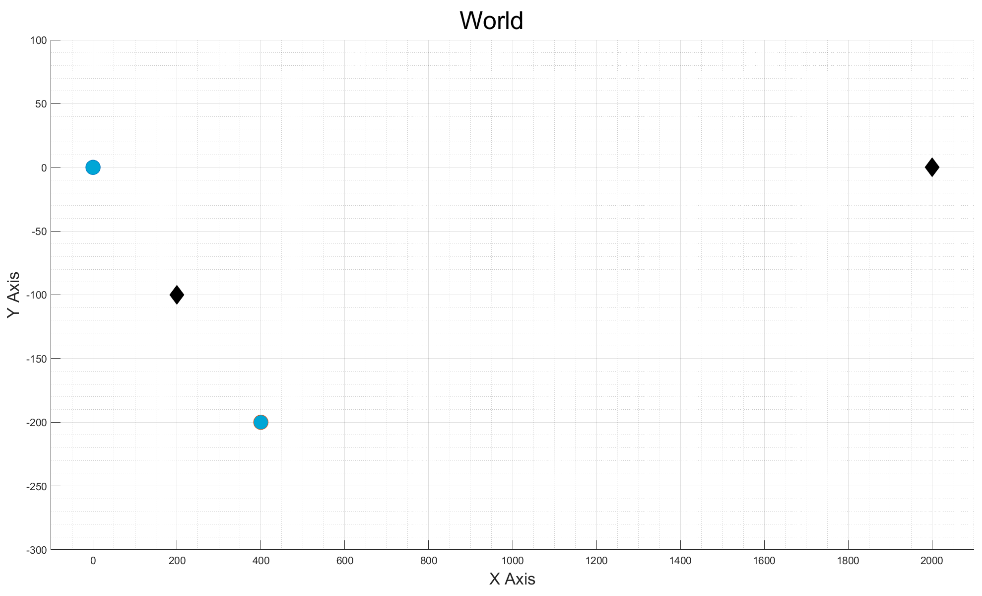


Figure 5.4: Communication Selection - Scenario 2

Here, the algorithm correctly chooses the sensor that has a lower error in its measurements and allocates it most of the sensing time. The first sensor is still allocated some sensing time based on the lowest value in the action space. The lowest value can be decided based on some safety settings in case communication to the central system fails. This would allow the sensor to still have some local information about the targets. The algorithm considers the cost of the communication by the sensor against the value of the information it can provide. It decides that the potential gain in information from Sensor 1 does not impact the cost function as much as providing the same resources to Sensor 2 for sensing operations.

The final static scenario that is considered is shown in figure 5.6. In this scenario, each target is close to one sensor and form a pair close to each other. The pairs are however are distant from each other. This implies

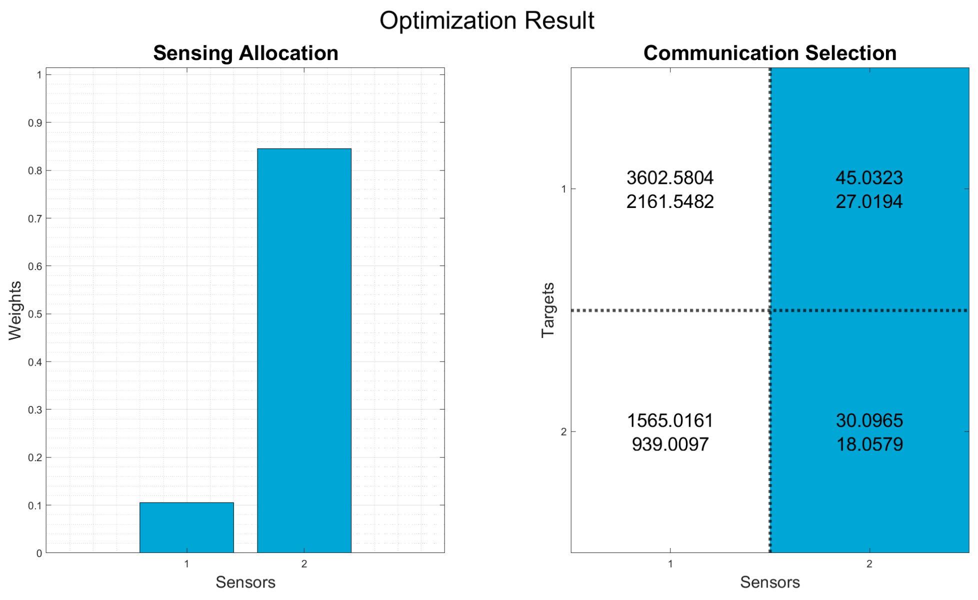


Figure 5.5: Communication Selection - Scenario 2 - Result

that each sensor has accurate information about one target and provides inaccurate information about the second target. In these simulations, these accuracy values are based on the range and the target's unique RCS value. In this scenario the RCS values of both targets are the same. Therefore it is quite obvious that each sensor can only sense the target close to it accurately.

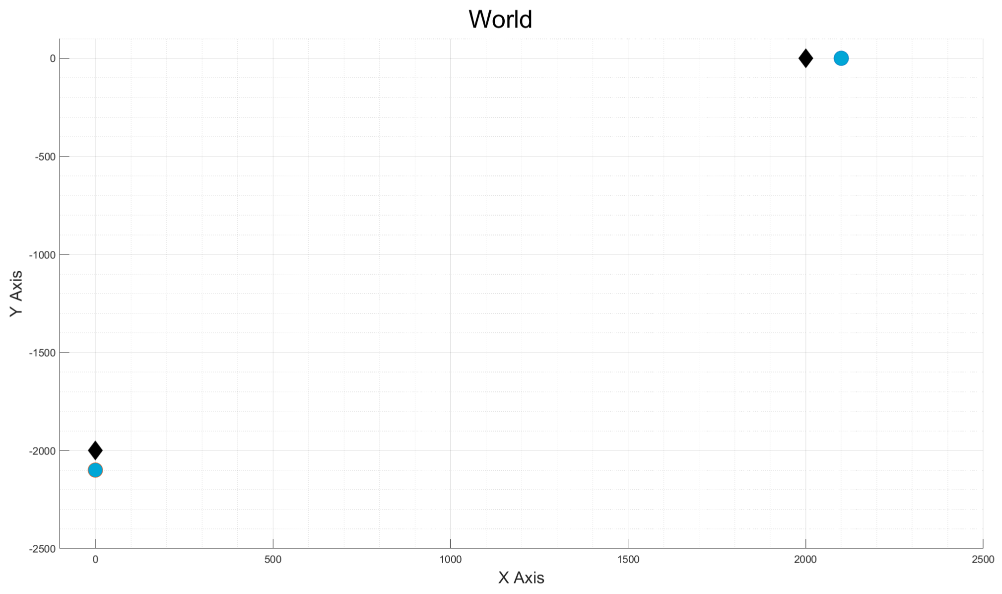


Figure 5.6: Communication Selection - Scenario 3

In this case, the algorithm selects information from each sensor that is accurate and does not use information that can reduce the tracking error. From these three simple scenarios, we can show the proposed algorithm can select accurate information from sensors while optimizing sensing time across multiple sensors. The communication cost is weighed against the decrease in the tracking error and based on the effect it has, only specific information is requested from the targets. Here, Sensor 1 provides information about Target 1 while Sensor 2 provides information about Target 2. This is used to update the filters while the unused resources for communication are used to increase the sensing budget for the overall set of sensors.

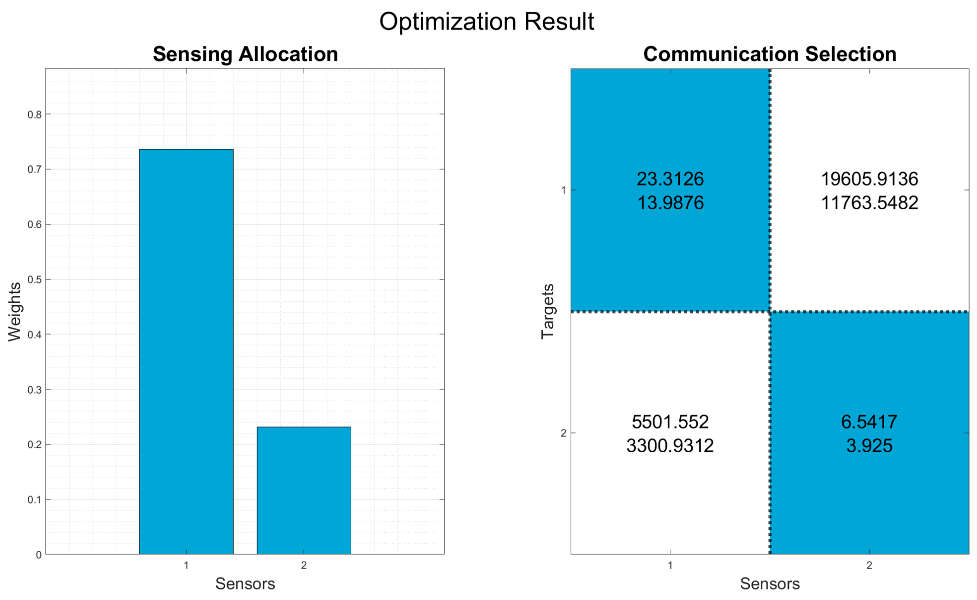


Figure 5.7: Communication Selection - Scenario 3 - Result

Please refer to figure 5.8 for results of a Scenario with 6 sensors. The results here show that the algorithm sometimes prefers sensors with an average accuracy of position measurements of all targets as compared to a combination of sensors each with the best accuracy of position measurements of one target. This result implies that when the algorithm dedicates a large amount of the sensing resources to a particular sensor, it attempts to use information from these sensors as best as it can to decrease the tracking error. When another communication strategy is chosen by the algorithm, the sensing strategy also needs to be adapted. The communication strategy denotes the sensors that the algorithm would like to invest sensing time to. Therefore, a new communication strategy requires a new sensing strategy and vice versa. This describes the jointness of the operation of the optimization algorithm for both sensing and communication tasks. The sensing and communication budgets selected by the algorithm are intertwined.

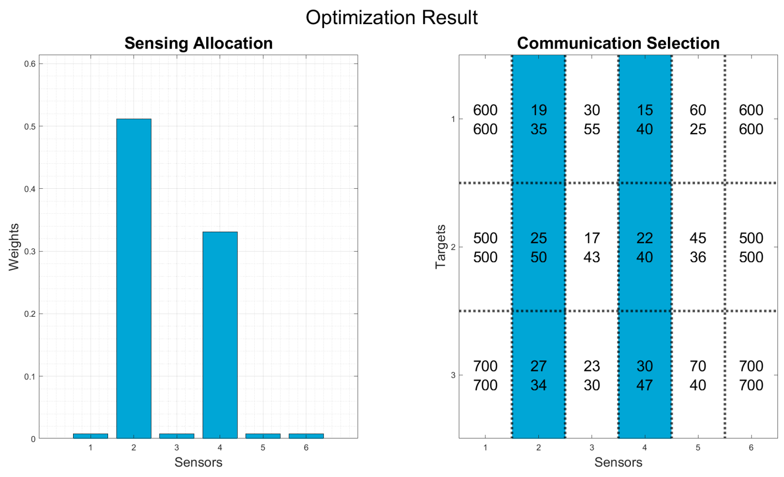


Figure 5.8: Communication Selection - Scenario 4 - Result

5.2.2. Dynamic Cases

The effectiveness of the algorithm lies in the fact that it is non-myopic. It predicts the worth of information in the future and can therefore anticipate the value of information multiple steps into the future to decide the sensing and communication strategy. This therefore cannot be observed in a static scenario. Therefore, we consider dynamic cases in this section to observe the behavior of the algorithm in more complicated scenarios. First, a simple scenario is considered to demonstrate an anomaly in the algorithm. In this case, both targets and sensors are static and are shown to be positioned as considered in Scenario 3 in the previous section in figure 5.9. In addition to the positions of the sensor (diamonds) and the targets (black circles), the measurements made by the sensors are shown in dots and the track of the Kalman Filter is shown in blue circles. The results of the simulation are shown in figure 5.10. The graph on the left is shows the sensing budget allocated to each Radar Node

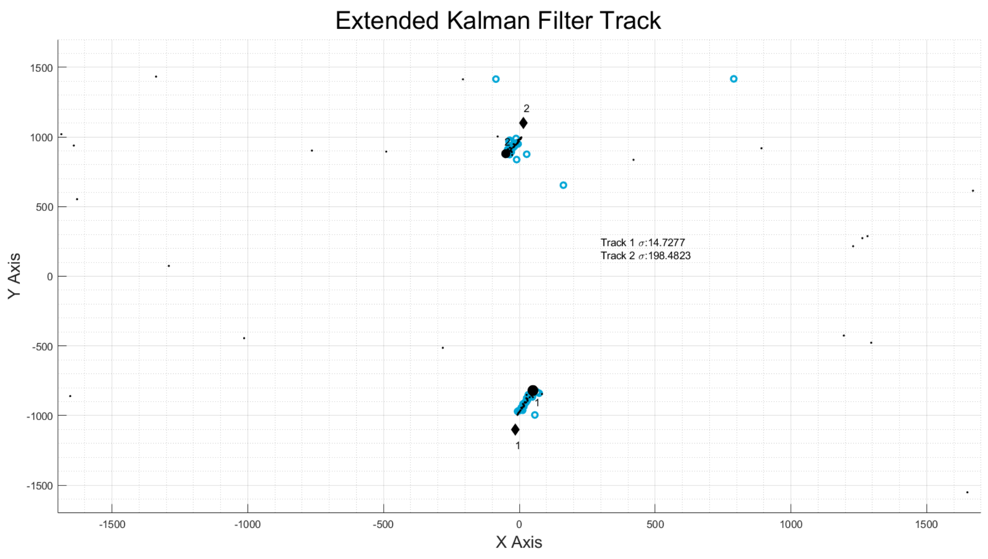


Figure 5.9: Communication Selection - Dynamic Scenario 1

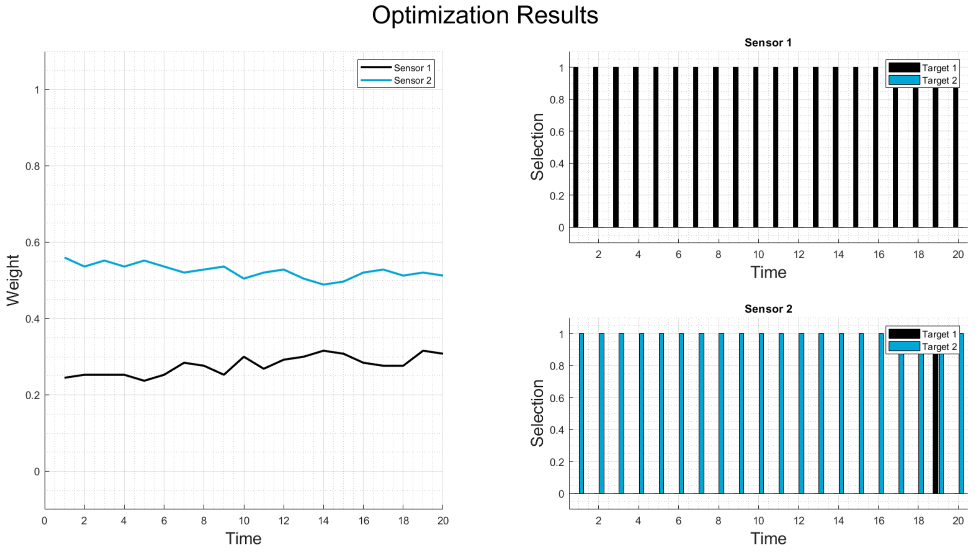


Figure 5.10: Communication Selection - Dynamic Scenario 1 - Result

The targets are moving as shown in figure 5.9 based on the trail left behind by the circles and the circle size denotes the RCS of the target. The circles denote the end point positions of the target. Larger circle sizes denote higher RCS values of the target. The simulation is run for a total of 20 seconds. As in the case of Chapter 4, the simulation is run for the first 10 seconds without the optimization algorithm to allow for the Kalman Filters to stabilize in terms of the tracking error. The Filters are initialized with a large error covariance with an initial estimate at the origin. The estimates do not indicate the actual positions of the targets in the simulation scenarios. The left graph in figure 5.10 describes the relative sensing budget allocated to each sensor and can take values between 0 and 1 based on the action space. The graph on the right denotes the communication selection for each time step. If the value is 1, it means the information is transmitted and vice versa. The anomaly occurs at simulation time 19 seconds, where despite having incorrect information about target 1, the solution request information from sensor 2. This occurs rarely where the algorithm gets stuck oscillating across two values across the minima. In this situation, the algorithm does not converge and an incorrect strategy might be selected. A solution to this problem might be to consider an adaptive step size for the recursion instead of a fixed step size. This ensures that initially, the convergence is fast while at the tail end, it is accurate.

The next dynamic scenario under consideration moves each of the targets initially placed close to one sensor towards the other sensor as the simulation progresses. This is shown in figure 5.11. Target 1 starts near sensor 1 but at the end of the simulation, it is closer to sensor 2. The black circles denote the end positions of the targets. The average errors of each track are also shown in the graph along with the Kalman Filter track using blue circles. The measurements are shown as black dots. Despite the measurements being quite varied, the filter can start an accurate track early in the target trajectory. Here it is expected that initially each sensor communicates information about the target closest to it and this switched towards the end of the simulation. It should also be noted that Target 1 has a larger RCS by approximately 1.5 times.

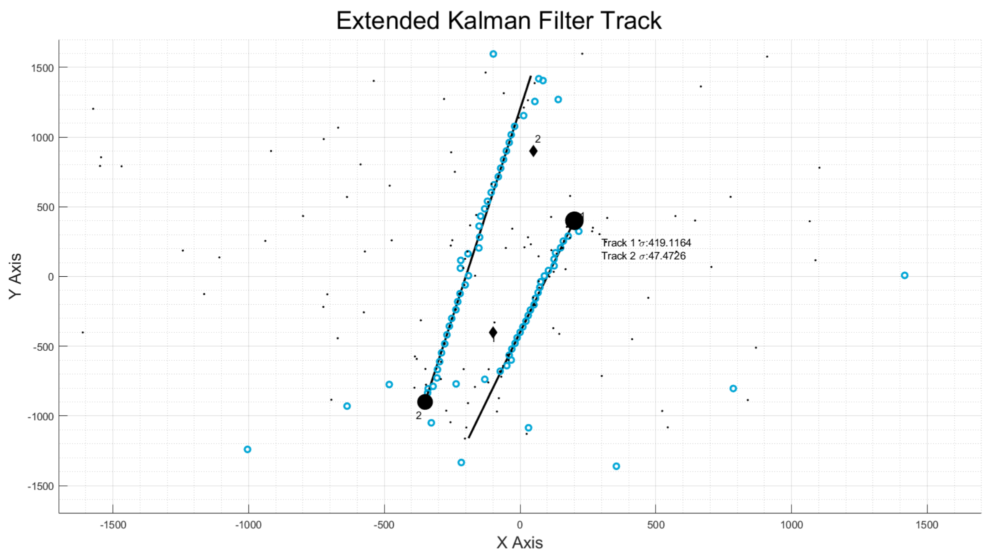


Figure 5.11: Communication Selection - Dynamic Scenario 2

The results of the simulation are shown in figure 5.12. The behavior of the communication selection is as expected. Until 13 seconds, Sensor 1 is used to track Target 1 and Sensor 2 is used to track Target 2 but information about both targets is requested. However, as the simulation progresses after 20 seconds, this is reversed. In the middle of the simulation between 14 and 20 seconds, Sensor 1 gets most of the sensing resources and this could be attributed to it being closer to the target with the lower RCS. Both sensors might provide similar accuracy in the position measurements of the other target. In this case, the algorithm allocates most of the sensing resources to Sensor 1 and requests information from Sensor 2 rarely. When information is only used from Sensor 1, its sensing budget also increases to almost the entire budget. This ensures the information requested by the algorithm is as accurate as it can be. However, it should be noted that in the tail end, the budgets equalize when the targets are close to the sensors and then start diverging away from each other to their own stable value corresponding to the Target RCS values. These results show that the algorithm can adapt in dynamic cases but the results are not as obvious as they seem in more complex scenarios. This is seen in the results of the following chapter.

Finally, we look at a scenario that could be slightly more realistic in an automotive case as shown in figure

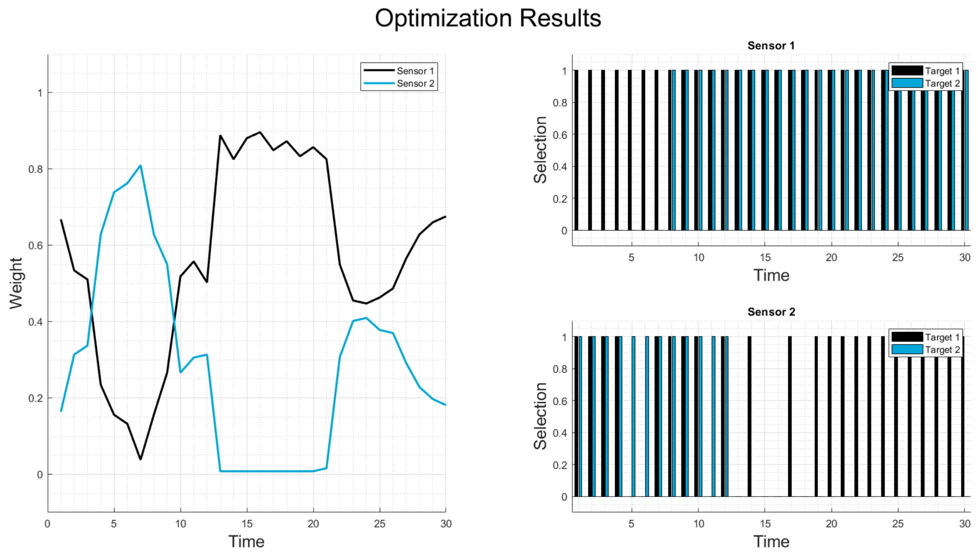


Figure 5.12: Communication Selection - Dynamic Scenario 2 - Result

5.13. One target is moving with a negative velocity in the y axis while the other target is initially moving with a positive velocity along with y axis and takes a turn and starts moving along the x axis with constant velocity. One of the sensors is moving as well while the other is static. This simulation could be visualized as a road junction and the static sensor could be infrastructure place to track targets while the moving sensor could be a vehicle. A pedestrian could be moving across the field of view of the vehicle from the top of the graph while another vehicle arrives at the junction and takes a turn.

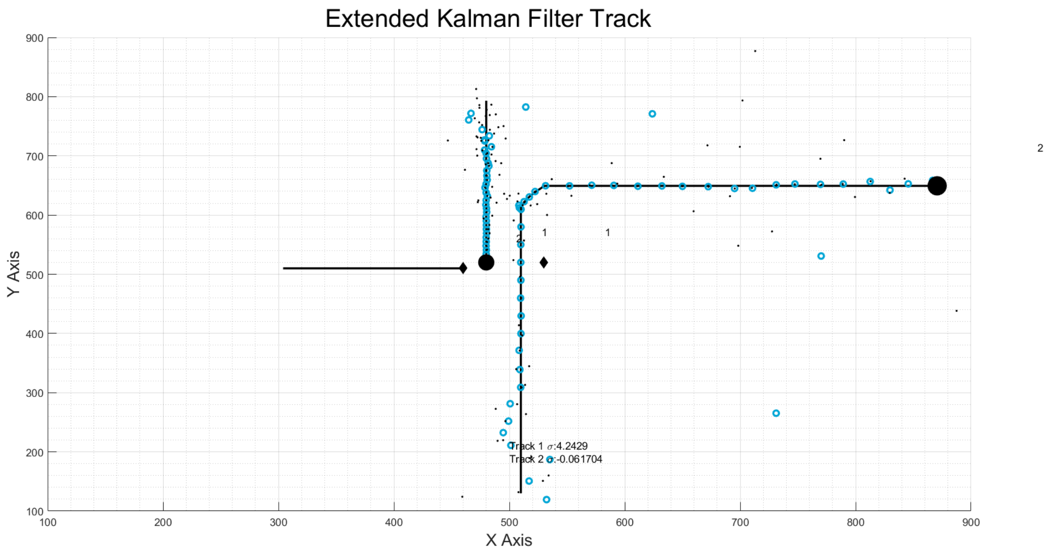


Figure 5.13: Communication Selection - Dynamic Scenario 3

The results are shown in figure 5.14 and as discussed earlier, for the first 10 seconds, the optimization is not run. In this scenario, the RCS of Target 1 is approximately 1.5 times the RCS of Target 2. Once the steady state of the Kalman Filter is reached, the simulation starts and runs for 20 seconds as shown on the graphs. Sensor 1 has better measurements accuracy of both target from throughout the simulation as it is closer to the targets and is therefore assigned most of the sensing budget and it is stable throughout the simulation. The algorithm also uses information about both target from this sensor while Sensor 2 is considered only to track target 1 as it has a much lower accuracy when measuring the position of target 2. The communication

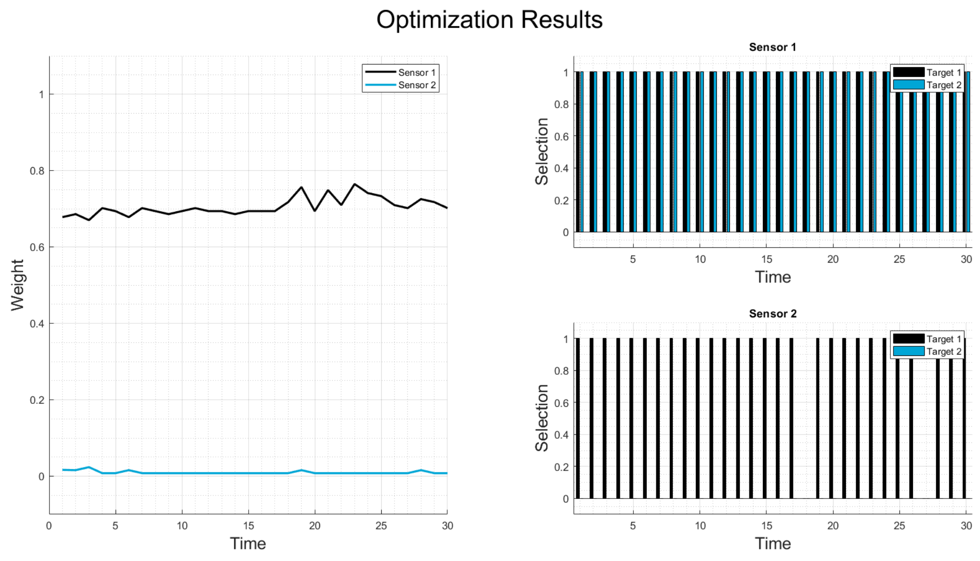


Figure 5.14: Communication Selection - Dynamic Scenario 3 - Result

from Sensor 2 also reflects this. This could reflect a real-world scenario well. As a vehicle approaches a road junction, it uses it's sensors less and relies on more accurate sensors attached to the infrastructure at the junction.

5.3. Conclusion

Chapter 5 introduced the aspect of selective communication in a scenario where the communication time could not be considered negligible. Although it built upon the model discussed in chapter 4, it challenged the assumptions from chapter 3 and provided a solution that could optimize both the sensing and communication time using the same resources. The model could automatically decide what information would be communicated by each sensor given the size of the communication packet. In contrast to all the information being communicated, this decreased the communication load to only the information that improved the global situational awareness of the system. This removed the ambiguity in the system about its efficiency in terms of communication performance. The results of this chapter were:

1. The system automatically selected the information that the sensor could provide that was useful in updating the track.
2. The solution decided the important information that should be communicated and passed this to the sensors. The sensors then sensed the targets using the allocated sensing time but only communicated information that had been requested by the solution.
3. This system dynamically predicted and requested only information about certain targets from certain sensors. This meant that certain sensors that had no information to provide were not allocated any sensing or communication budget but received information from the system and would operate for a minimum amount of time for safety precautions.

6
Time and Frequency Division

In this chapter, we consider a case where multiple frequency bands can be used by the sensors. The automotive radar spectrum is between 77 GHz to 81GHz. Most radars work within a fraction of this bandwidth. However, they are capable of shifting this bandwidth across the entire frequency band. This implies that several non-overlapping frequency bands are available for usage within the automotive spectrum. Chapters 4 and 5 did not allow sensors to operate simultaneously. However, when sensors operate in different frequency bands, the interference is minimal to none.

6.1. Constraint Reformulation

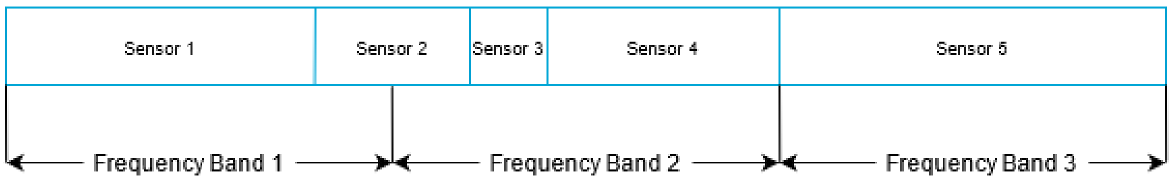


Figure 6.1: Frequency Blocks

To consider multiple frequency bands, we assume that the number of sensors is greater than the number of available frequency bands. If the number of sensors is lower, each radar node can occupy one band and the optimization of resources is not required. However, when the number of sensors (M) is greater than the number of frequency bands (F), we can allocate resources in multiple configurations. To accommodate this, we also assume that the radar node can use only one frequency band at a time. When the number of frequency bands increases, we can increase the budget available using a simple product between the bands and the sensing time interval within each band. Therefore we can define a virtual budget (B_v) as shown in equation 6.1 where the budget consider in the precious optimization problems (B) is multiplied by the frequency bands.

$$B_v = B\,F \tag{6.1}$$

Thus the optimization problem can now be reformulated to consider the new budget as shown in equation 6.2. The constraint now assumes that the budget is increased based on the number of frequency bands available. Once the optimization result is calculated, the allocated weights can be stacked into different frequency bands by drawing virtual boundaries as shown in figure 6.1.

$$\begin{aligned} \underset{\boldsymbol{\tau}}{\text{minimize}} \quad & \sum_{i=1}^{N} C_i(\boldsymbol{\tau}, \mathbf{T}) \\ \text{subject to} \quad & \mathbf{1}^T\boldsymbol{\tau} + \mathbf{1}^T\mathbf{T}\mathbf{1} f = B_v \end{aligned} \tag{6.2}$$

However, the optimization needs to consider the fact that radar nodes cannot operate simultaneously in multiple frequency bands. In a scenario where one sensor has the most accurate information about all the targets, the optimization problem described in equation 6.2 would assign all the budget across time and frequency to this sensor. This is not practical and to consider the limitations of the sensor, an additional constraint needs to be placed on the optimization problem. The maximum budget that can be allocated to each sensor cannot exceed the budget available in a single frequency band. This will now ensure that no radar node is allocated more resources what it can consume. Thus the optimization problem can now be expressed as shown in equation 6.3.

$$\begin{aligned} \underset{\boldsymbol{\tau}}{\text{minimize}} \quad & \sum_{i=1}^{N} C_i(\boldsymbol{\tau}, \mathbf{T}) \\ \text{subject to} \quad & \mathbf{1}^T \boldsymbol{\tau} + \mathbf{1}^T \mathbf{T} \mathbf{1} f = B_v \\ & \tau_i \leq B \; \forall \; i \in \{1, 2, \dots, M\} \end{aligned} \tag{6.3}$$

Without the second constraint, the problem remains the same as discussed in the previous chapters. Therefore, to simplify the problem, we allow actions in the action space that meet the constraint related to the budget allocated to individual sensors. This consideration in the policy rollout, effectively negates the need for the constraint and the optimization problem previously discussed can be used within a new calculation for the virtual budget available. This in effect reverts the optimization to the one described in equation 6.2.

6.2. Results

To test this approach, we consider the case of three sensors tracking three targets in a scenario with 2 frequency bands. Each of the three targets is moving with constant velocity with random start positions and velocities as shown in figure 6.2. The actual target trajectories are shown as lines while the target themselves are black circles with their size dependent on their RCS values. In this case, the RCS values are approximately the same. The black circles are the end positions of the targets. The black dots represent the position measurements of the targets by all the sensors, and the blue circles represent the estimation of the target track by the Kalman Filter.

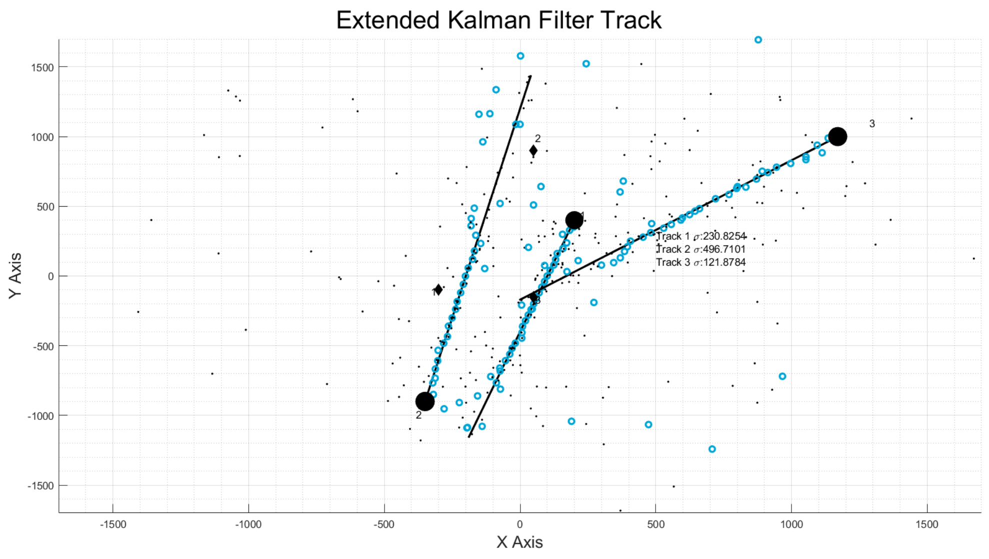


Figure 6.2: Frequency Division - Dynamic Scenario 1

Each of the targets initially starts close to one of the sensor and at the end, the closest sensors to each target are different. Target 2 for example starts as the target closest to Sensor 2 but at the end of the simulation is closest to Sensor 1. At the start of the simulation, until 4 seconds, Sensor 2 is given low weightage, and therefore, the track corresponding to Target 2 suffers in terms of converging to the actual track. However, Sensors 1 and 3 are given high weights and the tracks converge much faster. The communication strategy also reflects this, initially until time interval 4, all information from Sensor 1 and 3 is requested while lesser information is used from Sensor 2. In the middle of the simulation, between the time interval 5 - 20 seconds, the algorithm is spoiled for choice of the source of information for the Extended Kalman Filter update. In this case, the algorithm can be more selective about the communication strategy as accurate information is being provided by all sensors. Therefore, in the middle of the simulation at around 5 seconds, Sensor 1 starts losing sensing budget as it is allocated to Sensor 3. This is because Sensor 3 can provide the same information that Sensor 1 can. Similarly, Sensor 2 loses sensing budget, and information about fewer targets is requested from it. Sensor 3 however is placed centrally when considering all targets and is therefore used to more the majority of the accurate information to the algorithm. At the end of the simulation, after 25 seconds, the algorithm reverts to using all information from all sensors as accurate information is not available. Therefore all information is requested from all sensors to update the Kalman Filter track. These interpretations however are subjective and are based on previous results. In complex scenarios, it is more difficult to assign direct causes for the algorithm's results.

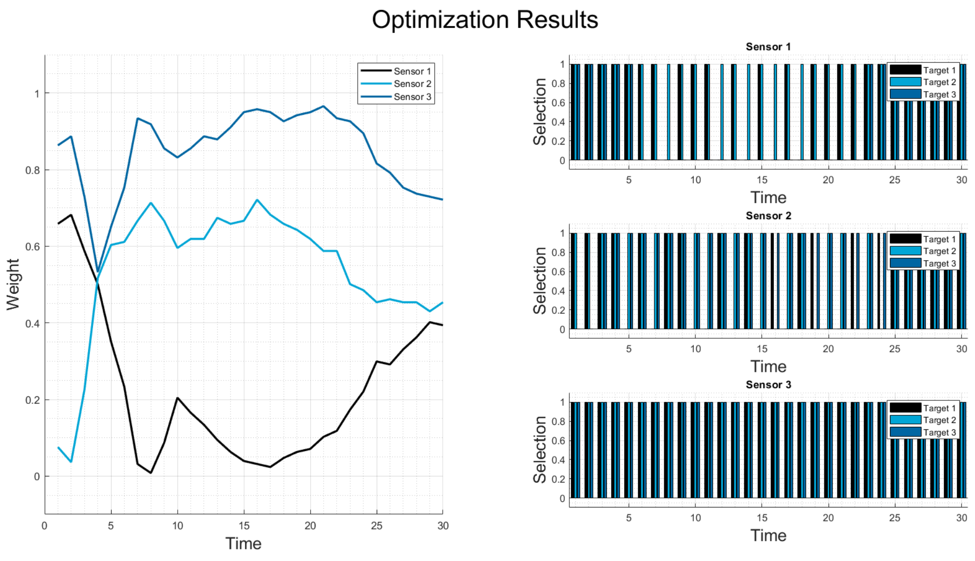


Figure 6.3: Frequency Division - Dynamic Scenario 1 - Result

6.3. Distributed Solution

$$\begin{aligned} \underset{\boldsymbol{\tau}}{\text{minimize}} \quad & \sum_{i=1}^{N} C_i(\boldsymbol{\tau}, \mathbf{T}) \\ \text{subject to} \quad & \mathbf{1}^T \tau + \mathbf{1}^T T \mathbf{1} f = B_v \\ & \tau_i \leq B \; \forall \; i \in \{1, 2, \ldots, M\} \end{aligned} \tag{6.3 revisited}$$

With the full algorithm discussed in the previous sections, we finally show the possibility for a distributed version of the algorithm in this section. We start with the equations for the optimization problem as described for frequency division. This is shown in equation 6.3. To distribute the algorithm we discuss the concepts of free and complicating variables. In a distributed solution, variables that can be decided by each node of the distributed network on its own are called free variables, and variables that need to be decided by the entire network are called complicating variables. It is important to note that without complicating variables, the solution is much simpler. This refers to optimization problems that can be broken into sub-problems that can be evaluated in parallel. When there are complicating variables involved, this is more difficult. Because these variables need to be shared across the nodes to agree upon a solution. In the case of this optimization, we consider each sensor to be a node in the distributed network. The free variables and the complicating variables for Sensor 1 are shown in the below equation, with a color code.

$$\tau = \begin{bmatrix} \tau_1 & \tau_2 & \ldots & \tau_M \end{bmatrix}^T \tag{6.4}$$

$$\mathbf{T} = \begin{bmatrix} T_{1,1} & T_{1,2} & T_{1,3} & \dots & T_{1,M} \\ T_{2,1} & T_{2,2} & T_{2,3} & \dots & T_{2,M} \\ T_{3,1} & T_{3,2} & T_{3,3} & \dots & T_{3,M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T_{N,1} & T_{N,2} & T_{N,3} & \dots & T_{N,M} \end{bmatrix} \tag{6.5}$$

The variables marked in blue are the free variables of the optimization for Sensor 1 and the variables in black are dependent on the network. To solve this problem in a distributed sense, we would have to formulate this problem so that the recursion steps can be performed by the nodes individually. After the recursion steps are completed, the nodes must then come to a consensus about the complicating variables of the optimization. To accomplish this, we first formulate the Lagrangian of the problem as shown in equation 6.6. The second constraint relates to the maximum sensing time is skipped as it is taken care of by the action space. In chapter 3, to simplify the algorithm, we allowed the calculation of the sensing and communication strategy for each sensor in parallel by using the values of the sensing and communication strategy from the previous recursion of the other sensors. We follow a similar procedure here where each sensor finds the best communication strategy and sensing time for itself. However, the value of λ needs to be constrained. To do this, we split the Lagrangian into multiple parts each with its own λ value as shown in equation 6.7.

$$Z_D = \underset{\lambda}{max}\left(\underset{\boldsymbol{\tau}}{min}\left(C(\boldsymbol{\tau},\mathbf{T}) + \lambda\left(\mathbf{1}^T\boldsymbol{\tau} + \mathbf{1}^T\mathbf{T}\mathbf{1}f - B\right)\right)\right) \tag{6.6}$$

Each of the individual parts of the equation relates to the policy of one of the sensors in the network. This implies that if the values of λ are consistent across the sensors in the network, the solution will converge to the centralized approach. This means that at the end of each recursion the values of lambda across different sensors need to be aggregated into a common value that can be used in the next recursion, that is, $\lambda_1 = \lambda_2 = \ldots = \lambda_M$. This can be done using a weighted sum as shown in [24] called the Projected Consensus Algorithm. A simple version of this algorithm would be to average the values of λ after every recursion.

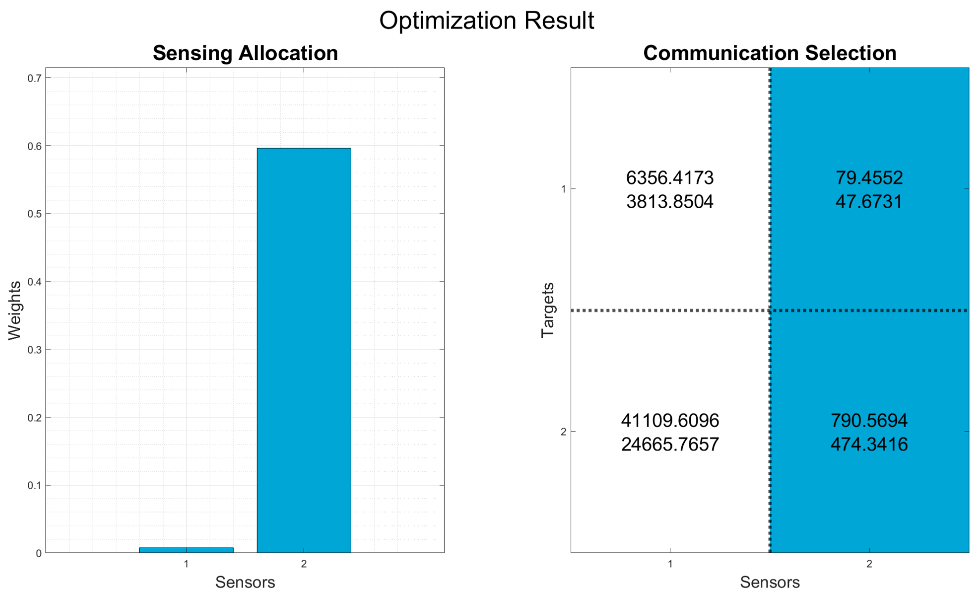


Figure 6.4: Distributed Solution - Scenario 1 - Result

$$\begin{aligned}
\lambda\left(\mathbf{1}^T\boldsymbol{\tau}+\mathbf{1}^T\mathbf{T}\mathbf{1}f-B\right)=&\lambda_1(\tau_1+f\sum_{j=1}^{N}T_{1,j}-\frac{B}{M})+\\
&\lambda_2(\tau_2+f\sum_{j=1}^{N}T_{2,j}-\frac{B}{M})+\\
&\lambda_3(\tau_3+f\sum_{j=1}^{N}T_{2,j}-\frac{B}{M})+\\
&\dots\\
&\lambda_M(\tau_M+f\sum_{j=1}^{N}T_{M,j}-\frac{B}{M})
\end{aligned} \tag{6.7}$$

Therefore, to use this solution, during the optimization process, constant communication is done by the sensor network. The optimization algorithm starts with an initially agreed-upon lambda, base policy, and step size by all the nodes. The sensors then perform one recursion and transmit the updated policy and the individual λ values to the other nodes. Once all the nodes receive information from all other nodes, they consider the mean of the Lagrangian and perform the next recursion using the updated policies from the other nodes. This averaging and recursion process is repeated until the same end condition as previously defined is reached. Using this solution, the static results of the previous cases as shown in chapter 5 have been replicated as shown in figures 6.4 and 6.5. However more testing needs to be done to shown that the solution is a viable alternative to the centralized approach.

6.4. Conclusion

Chapter 6 introduced the addition of the frequency resources along with the time division discussed in earlier chapters. This allowed for a realistic resource use case where the sensor could operate in different frequency bands within the available spectrum. The solution to this model could allocate resources to the sensors assuming a scarcity of the spectrum and that sensors could not operate simultaneously in two different bands. This model however assumed that the performance across the frequency bands is identical. That is, a sensor using different frequency bands to observe the same target has no performance difference. This solution also provided some interesting results as listed below:

1. The system was able to consider resources in both time and frequency and allocate resources across both.
2. The system performance increased as the total resources available increase based on the number of frequency bands as compared to the previous models.
3. The system performance devolved exponentially with an increasing number of sensors, targets, and prediction horizon.

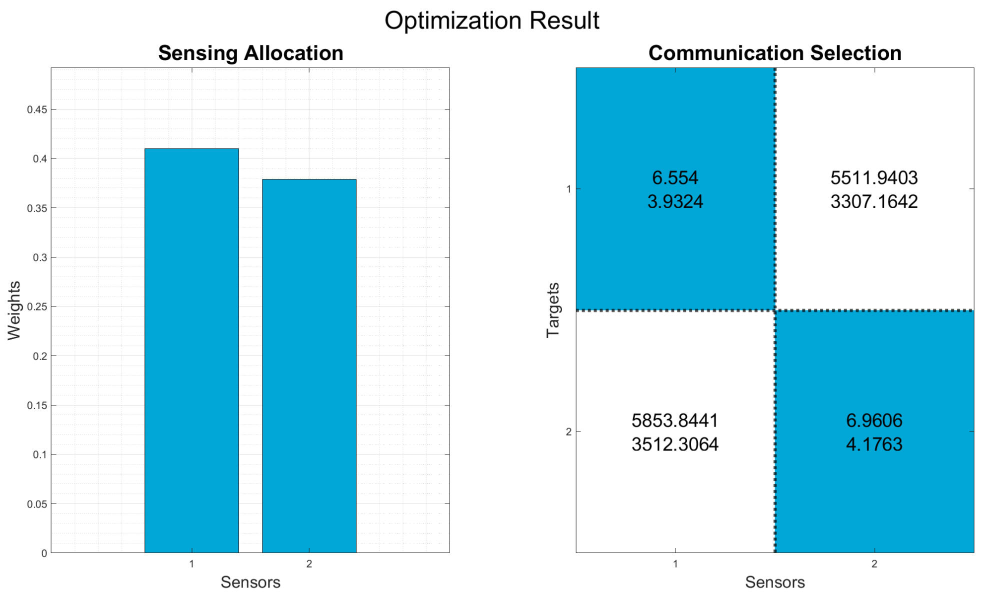


Figure 6.5: Distributed Solution - Scenario 2 - Result

7 Conclusion

After discussing the background needed to formulate the problem and its solution in chapters 1, 2, and 3, the thesis considered multiple scenarios of tracking targets and their solutions in chapters 4, 5, and 6. To conclude, this chapter summarizes the results and reflects on their implications. Finally, it considers the possible directions of future work.

7.1. Conclusions

The problem under consideration dealt with creating a system that would allocate time and frequency resources to both sensing and communication tasks for a set of Automotive Radar sensors. From the results of the simulations discussed previously, resources have been assigned to sensors in multiple use cases with different assumptions. However, there were some overarching assumptions. The proposed system was centralized. Its task was to manage homogeneous FMCW Radar Systems to track multiple point targets. The system did not allow any radar-to-radar interference. Perfect communication between sensors was assumed. Although the system could allocate resources, the scheduling of tasks in the sensor timeline was outside the scope of this thesis. The goal of this work was to create a system that would minimize the uncertainty in the environment in a tracking scenario. The system should have to maintain fairness across sensors and foster cooperation for mutual benefit. The effect of the system would be better resource use, reduce EM pollution, and potentially increase the longevity of the sensors.

7.1.1. Proposed System

The proposed system allocates time and frequency resources to both sensing and communication tasks in a multi-sensor and multi-target environment. It was configurable in terms of the (a) Cost Functions, (b) Prediction Horizon, (c) Number of Policy Rollouts, and (d) Unit time for sensing and communication. The prediction horizon defined how far the system would look into the future. The Number of Policy Rollouts defined how many scenarios the system would simulate. The unit time for sensing and communication specified the smallest resource block needed for each task. The following list documents the capabilities of the system:

1. The proposed system managed a set of sensors that provides information about multiple targets in an environment.

2. It evaluates the information that a sensor provides based on its impact on a cost function such as tracking error in the context of some prior knowledge about the targets.

3. The system allocates resources to those sensors that best improves the accuracy of the target track.

4. The system can run recursively in a dynamic scenario. It is non-myopic by looking into the future to see what information would have the best impact on the tracking error.

5. Compared to a scenario with no transfer of information between sensors, the proposed solution provided sensors a better understanding of the environment.

6. The system is also able to selectively request only certain information from a sensor if communication was associated with a cost.

7.1.2. Results

Some of the interesting results from the simulation of the proposed system are discussed in this section.

1. Certain situations require sensors to switch off. In this state, the sensors transmit using minimal resources for safety but receive all information from other sensors. These sensors observe the environment vicariously through other sensors. The information other sensors provide helps them have a good understanding of the environment.

2. Not all information that a sensor can provide is useful in the overall situation. When target measurements from a sensors are not precise, this information is not deemed worthy to use communication resources. This ensures that sensors that provided more precise results are provides more resources.

3. It is important to consider the information that a system already contains. New information should be judged based on the effect it has on the overall information gain. In situations were the information gain from a measurement is the same, it needs to be considered within the backdrop of the existing information. If existing information about a target is vague, new information about this target should be prioritized.

4. The effect an action has on the future should be considered. Targets generally move within the confines of some models. This information can be used to decide the best actions based on predictions. This makes the solution non-myopic as it predicts the quality of information in the future when allocating resources to each sensor.

5. Multiple frequency bands increase the available budget for the optimization. By using the proposed framework, this extension can be easily achieved because of it's modular and generic nature. The framework however needs to consider that no transmitter can operate at two different bandwidths simultaneously.

The novelty of the thesis is in the modular non-mypoic framework developed specifically for the automotive spectrum. Such work has not be published before. The modularity in the framework is it's strength and each individual piece can be replaced to create different implementations with different solutions. The cost function, tracking algorithm and the policy rollout can be replaced with different implementations. The thesis is also the first to implement a solution that compares the cost of communication and sensing side-by-side to decide the optimal strategy.

7.1.3. Implications

In the future, the development and usage of multitudes of wireless devices will likely place an enormous strain on spectral resources. New devices used for sensing use large bandwidths to provide higher resolution. The ubiquity of such devices seems inevitable in the area of human safety. Automotive Radar is one such field. Despite it currently being allocated a huge spectrum, the expected number of vehicles and the number of radar nodes per vehicle will quickly use up most of the available resources. The resources in the spectral domain are finite and cannot accommodate an exponential increase in the number of users. Responsible use of spectral resources is the expected challenge. Newly developed sensors can either collaborate or work independently from each other. Working alone would require them to perform well despite other sensors. Whereas working in collaboration could produce results that are of a higher standard because of auxiliary sensors. This thesis considers the latter and places it to the test.

The research provides a novel method to the resource allocation problem. The novelty lies in the fact that it is non-myopic, a generic and modular framework and has shown how both sensing and communication tasks can be evaluated in the same context. It considers the effect of information that radar node provides multiple steps into the future to decide the resource allocation scheme. The horizon for the non-myopic solution is limited by the available hardware due to computational complexity. The proposed solution is modular. Each part in the proposed system can be swapped out to create different implementations. For example, a Particle Filter can replace the Kalman Filter or the Policy Rollout can be replaced with a machine learning model. An ML model could also help add more information to the proposed system. Such a generic framework for the resource allocation scheme has not been proposed earlier. The solution can also consider other sensors embedded in the infrastructure such as cameras at traffic signals provided they have an estimate of the error in the provided information. This allows for data fusion between different sensor types.

When the number of users of the spectrum is much larger than what it can accommodate, the proposed solution can be vital to help everyone sense and communicate. The results of the thesis show simulations of the use of sensors in such a scenario. The frequency spectrum should not devolve into a shouting match between different users, that is, the most powerful transmitters end up using all the resources. It should let every user have their share of the available resource allowing everyone to enjoy its benefits. A system that promotes cooperation between sensors would be more than the sum of its parts.

7.2. Future Work

In the future the assumptions of the thesis detailed in chapter 1 could be challenged and a solution could be considered using the following ideas:

1. Although theoretically, the solution supports N sensors, the simulation time increases exponentially with sensors, targets, horizon length and number of policy rollouts. More efficient implementations need to be created to use this framework in real-time. One such approach has already been proposed in [9].

2. Although the system allocates resources to the sensors, the actual scheduling of the sensing and communication on a timeline is not considered. This could be done in future work.

3. The impact of interference could be taken into consideration to decide if multiple sensors could operate simultaneously. Currently this impact is considered to be too high and thus the system doesn't allow for simultaneous operation.

4. The system could be extended to consider a distributed system. Currently a central system tracks the targets and provides this information to the sensors involved. We have proposed an approach in chapter 6 but it requires more testing.

5. Network Analysis principles can be applied to the solution. Currently it is assumed that all sensors can communicate with each other and all information is pertinent to each sensor.

6. The system also assumes that each target can be uniquely labelled by the sensors. Two targets close to each other could be ambiguous and this could be taken into account in a more realistic setting.

A

Simulation Parameters

The following table documents all the parameters used for the simulation. However, the target RCS, maneuverability and the measurements are randomly generated. The base policy considers all sensors to be allocated equal budget and all information is communicated by all sensors.

Parameter	Value
Time Interval	2 sec
Total Budget	1
Action Space Size	128
Communication Time	0.05
Steady Time	10 sec
Maximum Recursions	1000
Initial Lambda	1500
Step Size	100
Horizon	2
Rollouts	4
End Condition Precision	0.01

Table A.1: Simulation Parameters

B
MATLAB Code / Tweaks

B.1. Radar Target

```
classdef radarTarget
    % radarTarget Creates a radar target that moves with constant velocity
    %   This class simulates a target moving with a constant velocity in
    %   two dimensions
    properties
        startPosition(2, 1) {mustBeReal} % Define the starting position
        startVelocity(2, 1) {mustBeReal} % Define the constant velocity
        maneuverability {mustBeReal} % Maneuverability noise of target
        radarCrossSection {mustBeReal} % RCS of the target
        currentPosition(2, 1)  {mustBeReal} % The current position of the target
    end
    methods
        function obj = radarTarget(initPosition, initVelocity)
            % radarTarget Initialized the target to start at a particular
            % position with a particular velocity
            if nargin == 0
                obj.startPosition = [0; 0];
                obj.startVelocity = [0; 0];
            elseif nargin == 1
                obj.startPosition = initPosition;
                obj.startVelocity = [0; 0];
            elseif nargin == 2
                obj.startPosition = initPosition;
                obj.startVelocity = initVelocity;
            else
                error('Class not initialized, provide correct arguments');
            end
            obj.maneuverability = round(10 * rand()) + 1;
            obj.radarCrossSection = round(400 * rand()) + 1;
            obj.currentPosition = obj.startPosition;
        end
        function obj = move(obj, T)
            % moveTarget Moves the target based on the given time interval
            obj.currentPosition = obj.currentPosition + T * obj.startVelocity;
        end
    end
end
```

B.2. Radar Sensor

```
classdef radarSensor
    % radarSensor Simulates a sensor observing a target
    %   This object will observe a target with a certain noise level that
    %   is predefined in polar coordinates
    properties
        startPosition(2, 1) {mustBeReal} % Define the starting position
        startVelocity(2, 1) {mustBeReal} % Define the constant velocity
        currentPosition(2, 1) {mustBeReal} % Location of the sensor
        measurement(2, :) {mustBeReal} % Current Polar Measurement
        noise(2, :) {mustBeReal} % Sensor Noise for target
        coherence {mustBeReal}  = 1 % Coherence Gain for Sensor
        trackedTargets {mustBeInteger} = 0 % Currently tracked targets
    end
    methods
        function obj = radarSensor(location, velocity)
            % Initializes the sensor to a particular position
            if nargin == 0
                obj.startPosition = [0; 0];
            elseif nargin == 1
                obj.startPosition = location;
            elseif nargin == 2
                obj.startPosition = location;
                obj.startVelocity = velocity;
            else
                disp('Class not initialized, provide correct arguments');
            end
            obj.currentPosition = obj.startPosition;
        end
        function obj = move(obj, T)
            % moveTarget Moves the target based on the given time interval
            obj.currentPosition = obj.currentPosition + T * obj.startVelocity;
            obj.trackedTargets = 0;
            obj.measurement = [];
        end
        function obj = measure(obj, radarTarget, coherence)
            % Generates noisy measurements based on locations
            obj.trackedTargets = obj.trackedTargets + 1;
            offsetLocation = radarTarget.currentPosition - obj.currentPosition;
            [theta, range] = cart2pol(offsetLocation(1), offsetLocation(2));
            obj.noise(:, obj.trackedTargets) = calcVariance(range, theta,
                radarTarget.radarCrossSection, 'S');
            obj.noise(:, obj.trackedTargets) = obj.noise(:, obj.trackedTargets) ./
                coherence;
            obj.measurement(1, obj.trackedTargets) = range + abs(sqrt(obj.noise(1,
                obj.trackedTargets))) * randn();
            obj.measurement(2, obj.trackedTargets) = theta + abs(sqrt(obj.noise(2,
                obj.trackedTargets))) * randn();
            obj.coherence = coherence;
        end
    end
end
```

B.3. Extended Kalman Filter

```matlab
classdef extendedKalman
    % extendedKalman Creates an extended Kalman Filter
    %   This class takes polar measurements of targets from multiple
    %   sensors at different locations and provides the best estimate of a
    %   target location using a constant velocity model in two dimensions
    properties
        F(4, 4) {mustBeReal} % State Transition Matrix
        H(2, 4) {mustBeReal} % Measurement Matrix
        Qw(4, 4) {mustBeReal} % Maneuverability Noise Matrix
        Qv(2, 2) {mustBeReal} % Measurement Noise Matrix
        kState(4, 1) {mustBeReal} = [1; 1; 1; 1;] % Best Known Current Location
        kError(4, 4) {mustBeReal} = diag([1e6 1e6 1e4 1e4]); % Accuracy of Current
            Location
        kGain(4, 2) {mustBeReal} % Kalman Gain
    end
    methods
        function obj = extendedKalman(radarTarget, T)
            % Initalizes the EKF based on the time interval
            if nargin == 2
                obj.F = [1 0 T 0;
                         0 1 0 T;
                         0 0 1 0;
                         0 0 0 1];
                w = [T^2/2     0;
                         0 T^2/2;
                         T     0;
                         0     T];
                obj.Qw = w * w' * radarTarget.maneuverability;
            else
                disp('Class not initialized, provide correct arguments');
            end
        end
        function obj = update(obj, radarSensor, targetID)
            % update Updates the Kalman Filter based on the
            % measurement of target position by a sensor at a location
            % given it's accuracy
            x = obj.kState(1) - radarSensor.currentPosition(1);
            y = obj.kState(2) - radarSensor.currentPosition(2);
            [theta, range] = cart2pol(x, y);
            obj.H = [   x/range    y/range 0 0;
                     -y/range^2 x/range^2 0 0];
            obj.Qv = [radarSensor.noise(1, targetID) 0;
                      0 radarSensor.noise(2, targetID)];
            obj.kGain = obj.kError * obj.H' / (obj.H * obj.kError * obj.H' + obj.Qv)
                ;
            obj.kState = obj.kState + obj.kGain * (radarSensor.measurement(:,
                targetID) - [range; theta]);
            obj.kError = (eye(4) - obj.kGain * obj.H) * obj.kError;
        end
        function obj = predict(obj)
            % predictState Predicts the state of the target in the next
            % time interval given the maneuverability of the target
            obj.kState = obj.F * obj.kState;
            obj.kError = obj.F * obj.kError * obj.F' + obj.Qw;
        end
        function errorCost = cost(obj)
            % getAccuracy Calculates to total error of the EKF based on
            % Error Covariance Matrix
            errorCost = obj.kError(1, 1) + obj.kError(2, 2);
        end
    end
end
```

B.4. Best Sensor Action

```
function bestSensorAction = bestAction(optimizationSensor, currentPolicy, targets,
    sensors, filters, system, simulation)
% BESTACTION Finds the best action for a given sensor for a given lambda

    % Creating a table of all possible actions
    actionSpace = makeActions(system.totalBudget, system.actionSpaceSize, system.
        totalTargets);
    % Counting the total number of actions
    totalActions = size(actionSpace, 1);

    % Output variable for cost of each action
    actionCosts = zeros(1, totalActions);
    % Looping through every action and calculating total cost
    parfor actionIndex = 1:totalActions
        % Choosing the action under consideration
        action = actionSpace(actionIndex, :);
        % Output variable for cost of each rollout
        rolloutCosts = zeros(1, simulation.rollouts);
        % Looping through rollouts to be averaged
        for rolloutIndex = 1:simulation.rollouts
            % Output variable for cost of tracking each target
            targetCosts = zeros(1, system.totalTargets);
            % Looping through each target to find cost of each
            for targetIndex = 1:system.totalTargets
                % Creating a virtual target based on original target
                virtualTarget = targets(1, targetIndex);
                % Using the filter for the the same state and error data
                virtualFilter = filters(1, targetIndex);
                % Walk through the horizon to find the cost in the future
                for horizonIndex = 1:simulation.horizon
                    % Calculate total used budget for current action
                    usedBudget = 0;
                    % Loop through the sensors to update the filter
                    for sensorIndex = 1:system.totalSensors
                        % Create a virtual sensor to measure the target
                        virtualSensor = radarSensor(sensors(sensorIndex).
                            currentPosition);
                        % Update the filter if the sensor is not to be optimized
                        if sensorIndex ~= optimizationSensor
                            % Choose the action to be taken based on the history
                            sensingAction = currentPolicy.sPolicy(sensorIndex);
                            communicationAction = currentPolicy.cPolicy(sensorIndex,
                                targetIndex);
                        else
                            % Select the actions to be taken on the filter based
                            sensingAction = action(1);
                            communicationAction = action(targetIndex + 1);
                        end
                        % If communication happens, update the sensor
                        if communicationAction == 1
                            % Create a measurement based on the sensing action
                            virtualSensor = virtualSensor.measure(virtualTarget,
                                sensingAction);
                            % Update the filter using the measurement
                            virtualFilter = virtualFilter.update(virtualSensor, 1);
                        end
                        % Saving budgets for later calculation
                        sensingBudget = sensingAction;
                        communicationBudget = sum(communicationAction) * system.
                            communicationTime;
                        usedBudget = usedBudget + sensingBudget +
                            communicationBudget;
                    end
                    % The filter then predicts the location of the target and
                    % updates the error
                    virtualFilter = virtualFilter.predict();
                    % The cost for the target is then calculated based on the
                    % accuracy of this track given a particular horizon
                    targetCosts(targetIndex) = targetCosts(targetIndex) +
                        virtualFilter.cost();
                    % Move the target for the next horizon
                    virtualTarget = virtualTarget.move(system.timeInterval);
                end
            end
            % Calculating the Total Used Budget
            totalBudget = usedBudget / simulation.horizon;
            % Store the cost of the current action for further processing
            rolloutCosts(rolloutIndex) = sum(targetCosts) + currentPolicy.lambda *
                totalBudget;
        end
        actionCosts(actionIndex) = mean(rolloutCosts);
    end
    % The best action for the given sensor is returnes based on the least
    % cost from the action costs table
    [~, bestActionIndex] = min(actionCosts);
    % Choose the best action from the action space
    bestAction = actionSpace(bestActionIndex, :);
    % Save the return values
    bestSensorAction.sPolicy = bestAction(1);
    bestSensorAction.cPolicy = bestAction(2:end);
end
```

B.5. Best Policy

```
function new = bestPolicy(history, targets, sensors, filters, system, simulation)
% BESTPOLICY finds the best policy for the given set of sensors and targets
%   history = bestPolicy(policy, targets, sensors, filters, system, simulation)
%   Takes the initial policy and checks of the policy is the best for the
%   give scenario using information about targets, sensors, the system and
%   the simulation parameters
%   Input Parameters:
%       - history: Variables from previous run
%       - targets: radarTarget object array
%       - sensors: radarSensor object array
%       - filters: extendedKalman object array
%       - system: structure with system parameters
%       - simulation: structure with simulation parameters
%   Refer to README.md for more information.

    % Initializing the recursion variables
    rPrimalCost = zeros(1, simulation.maxRecursions);
    rDualCost = zeros(1, simulation.maxRecursions);
    rGradient = zeros(1, simulation.maxRecursions);
    rLambda = zeros(1, simulation.maxRecursions);
    rLambda(1) = history.lambda;

    % Setting the initial policy
    sPolicy = zeros(system.totalSensors, simulation.maxRecursions);
    sPolicy(:, 1) = history.sPolicy;
    cPolicy = zeros(system.totalSensors, system.totalTargets, simulation.
        maxRecursions);
    cPolicy(:, :, 1) = history.cPolicy;

    % Staring the recursion process
    for recursionIndex = 2:simulation.maxRecursions
        % Save the current policy for easy access
        currentPolicy.sPolicy = sPolicy(:, recursionIndex - 1);
        currentPolicy.cPolicy = squeeze(cPolicy(:, :, recursionIndex - 1));
        currentPolicy.lambda = rLambda(recursionIndex - 1);

        if simulation.diagnosticsFlag == 'Y'
            disp(currentPolicy.sPolicy);
            disp(currentPolicy.cPolicy);
        end

        % Find best action for each sensor given lambda
        parfor sensorIndex = 1:system.totalSensors
            % Find the best action for given sensor
            bestSensorAction = bestAction(sensorIndex, currentPolicy, targets,
                sensors, filters, system, simulation);
            % Change the best action for the next recursion
            sPolicy(sensorIndex, recursionIndex) = bestSensorAction.sPolicy;
            cPolicy(sensorIndex, :, recursionIndex) = bestSensorAction.cPolicy;
        end

        % Slow down the recursion to make sure it converges
        sPolicy(:, recursionIndex) = slowDown(sPolicy(:, recursionIndex), sPolicy(:,
            recursionIndex - 1), system);

        % Calculate the cost and the budgets for above actions
        % Looping through each target and build a cost
        targetCosts = zeros(1, system.totalTargets);
        for targetIndex = 1:system.totalTargets
            % Creating a virtual target using the original target for
            % simulations
            virtualTarget = targets(1, targetIndex);
            % Using the filter from the original run to use the same state
            % and error covariance
            virtualFilter = filters(1, targetIndex);
            % Looping through the the sensors to update the filter with the
            % appropriate sensing action if communication happens
            for sensorIndex = 1:system.totalSensors
                % Creating a sensor at the same location as the original
                % sensors for simulations with different measurements
                virtualSensor = radarSensor(sensors(sensorIndex).currentPosition);
                % Choosing the actions to be applied
                sensingAction = sPolicy(sensorIndex, recursionIndex);
                communicationAction = cPolicy(sensorIndex, targetIndex,
                    recursionIndex);
                % If communication happens about the target by the sensor,
                % update the global filter with the simulated measurement
                if communicationAction == 1
                    % Creating a new measurement based on sensing action
                    virtualSensor = virtualSensor.measure(virtualTarget,
                        sensingAction);
                    virtualFilter = virtualFilter.update(virtualSensor, 1);
                end
            end
            % The filter then predicts the location of the target and
            % updates the error
            virtualFilter = virtualFilter.predict();
            % The cost for the target is then calculated based on the
            % accuracy of this track given a particular horizon
            targetCosts(targetIndex) = virtualFilter.cost();
        end

        % Consider the budgets for the new action to calculate budgets
        sensingBudget = sum(sPolicy(:, recursionIndex));
        communicationBudget = sum(squeeze(cPolicy(:, :, recursionIndex)), 'all') *
            system.communicationTime;
        % Calculate total used budget using sensing and communication
        usedBudget = sensingBudget + communicationBudget;

        % Calculate gradient based on the budget
        rGradient(recursionIndex) = usedBudget - system.virtualBudget;

        % Update lambda based on the gradient for next recursion
        rLambda(recursionIndex) = currentPolicy.lambda + (simulation.stepSize *
            rGradient(recursionIndex));

        % Store the cost of the new action for further processing
        rPrimalCost(recursionIndex) = sum(targetCosts);
        rDualCost(recursionIndex) = rPrimalCost(recursionIndex) + rLambda(
            recursionIndex) * rGradient(recursionIndex);
```

```matlab

        % Printing the state of the optimization
        if simulation.diagnosticsFlag == 'Y'
            fprintf('recursion: %d, lambda: %0.3f, gradient: %0.3f, budget: %0.3f\n', ...
                recursionIndex, rLambda(recursionIndex), rGradient(recursionIndex), usedBudget);
        end

        % Checking exit conditions by building a window
        windowStart = recursionIndex - simulation.windowLength + 1;
        windowEnd = recursionIndex;
        if windowStart < 1
            windowStart = 1;
        end
        windowValues = abs(rGradient(windowStart:windowEnd));
        clear windowStart windowEnd;

        % Based on the type of check break the loop
        if simulation.endCondition == 'W'
            windowCheck = windowValues < simulation.endThreshold;
            if simulation.diagnosticsFlag == 'Y'
                disp(windowValues);
                disp(windowCheck);
            end
            % Break loop if all values lie within the prescribed window
            if sum(windowCheck) == simulation.windowLength
                disp('End Condition Reached');
                break;
            end
        elseif simulation.endCondition == 'D'
            firstDerivative = gradient(windowValues);
            secondDerivative = del2(windowValues);
            if simulation.diagnosticsFlag == 'Y'
                disp(firstDerivative);
                disp(secondDerivative);
            end
            % Break loop if the first and second derivative go to zero
            if firstDerivative(end) < simulation.endThreshold
                if secondDerivative(end) < simulation.endThreshold
                    break;
                end
            end
        end
    end

    % Action of the last recursion as result
    new.sPolicy = sPolicy(:, recursionIndex);
    new.cPolicy = squeeze(cPolicy(:, :, recursionIndex));
    new.lambda = rLambda(recursionIndex);

    % Create plots if the make plot flag is set
    if simulation.makePlotFlag == 'Y'
        % Variables for plotting
        timeStamp = fix(clock);
        timeStamp = strrep(num2str(timeStamp(2:5)), ' ', '');
        customColorMap = [0    0    0;
                          0  166  214];
        customColorMap = customColorMap ./ 255;

        % Create diagnostics plots
        if simulation.diagnosticsFlag == 'Y'
            gcf = figure('visible', 'off');

            set(gcf, 'Position', get(0, 'Screensize'));
            sgtitle('Diagnostics', 'FontSize', 24);
            subplot(2, 2, [1 2]);
            hold on;
            grid on;
            grid minor;
            plot(2:recursionIndex, rPrimalCost(2:recursionIndex), 'LineWidth', 2);
            plot(2:recursionIndex, rDualCost(2:recursionIndex), 'LineWidth', 2);
            hold off;
            colormap(customColorMap);
            legend('Primal', 'Dual', 'Location', 'best', 'FontSize', 14);
            xlabel('Recursion Step', 'FontSize', 16);
            ylabel('Cost', 'FontSize', 16);
            title('Cost and Langrangian Functions', 'FontSize', 20);
            subplot(2, 2, 3);
            plot(2:recursionIndex, rGradient(2:recursionIndex) + system.
                virtualBudget, 'LineWidth', 2);
            grid on;
            grid minor;
            xlabel('Recursion Step', 'FontSize', 16);
            ylabel('Total Used Budget', 'FontSize', 16);
            title('Budget Utilization', 'FontSize', 20);
            subplot(2, 2, 4);
            plot(2:recursionIndex, rLambda(2:recursionIndex), 'LineWidth', 2);
            grid on;
            grid minor;
            xlabel('Recursion Step', 'FontSize', 16);
            ylabel('Lambda', 'FontSize', 16);
            title('Gradient Descent', 'FontSize', 20);
            saveas(gcf, append(timeStamp, '_optdiag.png'));
            close(gcf);
        end

        % Create results plot
        gcf = figure('visible', 'off');
        set(gcf, 'Position', get(0, 'Screensize'));
        sgtitle('Optimization Result', 'FontSize', 24);
        subplot(121);
        bar(new.sPolicy, 'FaceColor', customColorMap(2, :));
        grid on;
        grid minor;
        ylim([0 max(new.sPolicy * 1.2)]);
        xlabel('Sensors', 'FontSize', 16);
        ylabel('Weights', 'FontSize', 16);
        title('Sensing Allocation', 'FontSize', 20);
        subplot(122);
        imagesc(new.cPolicy');
        customColorMap(1, :) = [1 1 1];
        colormap(customColorMap);
        customColorMap(1, :) = [0 0 0];
        yticks(1:system.totalTargets);
        xticks(1:system.totalSensors);
        ylabel('Targets', 'FontSize', 16);
        xlabel('Sensors', 'FontSize', 16);
        title('Communication Selection', 'FontSize', 20)
        for targetIndex = 1:system.totalTargets - 1
            yline(targetIndex+0.5, 'k:', 'LineWidth', 3);
        end
        for sensorIndex = 1:system.totalSensors - 1
            xline(sensorIndex+0.5, 'k:', 'LineWidth', 3);
        end
        for targetIndex = 1:system.totalTargets
            for sensorIndex = 1:system.totalSensors
                rangeAccuracy = sqrt(sensors(sensorIndex).noise(1, targetIndex));
                thetaAccuracy = rad2deg(sqrt(sensors(sensorIndex).noise(2,
                    targetIndex)));
                caption = {num2str(rangeAccuracy), num2str(thetaAccuracy)};
                t = text(sensorIndex, targetIndex, caption);
                t.HorizontalAlignment = 'center';
                t.FontSize = 18;
            end
        end
        saveas(gcf, append(timeStamp, '_solution.png'));
        close(gcf);

        % Create world map
        gcf = figure('visible', 'off');
        set(gcf, 'Position', get(0, 'Screensize'));
        sgtitle('World', 'FontSize', 24);
        hold on;
        grid on;
        grid minor;
        for targetIndex = 1:system.totalTargets
            scatter(targets(targetIndex).currentPosition(1), targets(targetIndex).
                currentPosition(2), 80, 'MarkerEdgeColor', customColorMap(2, :));
        end
        for sensorIndex = 1:system.totalSensors
            scatter(sensors(sensorIndex).currentPosition(1), sensors(sensorIndex).
                currentPosition(2), 80, 'filled', 'kD');
        end
        hold off;
        xlabel('X Axis', 'FontSize', 16);
        ylabel('Y Axis', 'FontSize', 16);
        saveas(gcf, append(timeStamp, '_worldmap.png'));
        close(gcf);
    end
end
```

NOTE: For test scripts and run-time code, please reach out to me or one of the contact persons via email.

C
Dynamic Noise

The noise defined in the target dynamics is based on acceleration. We assume that the within a time step, the object undergoes a constant acceleration a_k what is normally distributed with mean 0. Therefore the motion equation can be written as:

$$\mathbf{s_{k+1}} = \mathbf{F}\mathbf{s_k} + \mathbf{W}a_k \tag{C.1}$$

The matrix **F** is defined as described in chapter 3 however the matrix **W** is dependent on the acceleration. The terms related to acceleration therefore need to be considered with respect to time. In this case The matrix W would therefore be as follows:

$$\mathbf{W} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \tag{C.2}$$

This results in the dynamics of the target state dependent on the acceleration which is treated as a random process. Thus, the Covariance Matrix can be calculated using **W**.

$$\begin{bmatrix} p_x^{'} \\ p_y^{'} \\ v_x^{'} \\ v_y^{'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} a_k \tag{C.3}$$

Bibliography

[1] Stephen Alland, Wayne Stark, Murtaza Ali, and Manju Hegde. Interference in automotive radar systems: Characteristics, mitigation techniques, and current and future research. *IEEE Signal Processing Magazine*, 36(5):45–59, 2019. doi: 10.1109/MSP.2019.2908214.

[2] Canan Aydogdu, Musa Furkan Keskin, Nil Garcia, Henk Wymeersch, and Daniel W. Bliss. Radchat: Spectrum sharing for automotive radar interference mitigation. *IEEE Transactions on Intelligent Transportation Systems*, 22(1):416–429, 2021. doi: 10.1109/TITS.2019.2959881.

[3] Ofer Bar-Shalom, Nir Dvorecki, Leor Banin, and Yuval Amizur. Accurate time synchronization for automotive cooperative radar (cord) applications. In *2020 IEEE International Radar Conference (RADAR)*, pages 500–505, 2020. doi: 10.1109/RADAR42522.2020.9114861.

[4] Kristine L. Bell, Christopher J. Baker, Graeme E. Smith, Joel T. Johnson, and Muralidhar Rangaswamy. Cognitive radar framework for target detection and tracking. *IEEE Journal of Selected Topics in Signal Processing*, 9(8):1427–1439, 2015. doi: 10.1109/JSTSP.2015.2465304.

[5] Akanksha Bhutani, Sören Marahrens, Michael Gehringer, Benjamin Göttel, Mario Pauli, and Thomas Zwick. The role of millimeter-waves in the distance measurement accuracy of an fmcw radar sensor. *Sensors*, 19(18), 2019. ISSN 1424-8220. doi: 10.3390/s19183938. URL `https://www.mdpi.com/1424-8220/19/18/3938`.

[6] CBS. Centraal bureau voor de statistiek, 2020. URL `https://opendata.cbs.nl/statline/#/CBS/nl/dataset/71936ned/table?ts=1539365088669`.

[7] A. R. Chiriyath, B. Paul, and D. W. Bliss. Radar-communications convergence: Coexistence, cooperation, and co-design. *IEEE Transactions on Cognitive Communications and Networking*, 3(1):1–12, 2017. doi: 10.1109/TCCN.2017.2666266.

[8] Edwin K. P. Chong, Christopher M. Kreucher, and Alfred O. Hero. Partially observable markov decision process approximations for adaptive sensing. *Discrete Event Dynamic Systems*, 19(3):377–422, Sep 2009. ISSN 1573-7594. doi: 10.1007/s10626-009-0071-x. URL `https://doi.org/10.1007/s10626-009-0071-x`.

[9] Thies de Boer. Radar Resource Management for Multi-Target Tracking Using Model Predictive Control. Master's thesis, TU Delft, Delft, The Netherlands, 2021.

[10] Markus Gardill, Johannes Schwendner, and Jonas Fuchs. An approach to over-the-air synchronization of commercial chirp-sequence automotive radar sensors. In *2020 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNeT)*, pages 46–49, 2020. doi: 10.1109/WiSNeT46826.2020.9037593.

[11] Hana Godrich, Athina P. Petropulu, and H. Vincent Poor. Power allocation strategies for target localization in distributed multiple-radar architectures. *IEEE Transactions on Signal Processing*, 59(7):3226–3240, 2011. doi: 10.1109/TSP.2011.2144976.

[12] You Han, Eylem Ekici, Haris Kremo, and Onur Altintas. Optimal spectrum utilization in joint automotive radar and communication networks. In *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–8, 2016. doi: 10.1109/WIOPT.2016.7492941.

[13] You Han, Eylem Ekici, Haris Kremo, and Onur Altintas. Spectrum sharing methods for the coexistence of multiple rf systems: A survey. *Ad Hoc Networks*, 53:53–78, 2016. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2016.09.009. URL `https://www.sciencedirect.com/science/article/pii/S1570870516302153`.

[14] Jürgen Hasch, Eray Topak, Raik Schnabel, Thomas Zwick, Robert Weigel, and Christian Waldschmidt. Millimeter-wave technology for automotive radar sensors in the 77 ghz frequency band. *IEEE Transactions on Microwave Theory and Techniques*, 60(3):845–860, 2012. doi: 10.1109/TMTT.2011.2178427.

[15] Alfred O. Hero and Douglas Cochran. Sensor management: Past, present, and future. *IEEE Sensors Journal*, 11(12):3064–3075, 2011. doi: 10.1109/JSEN.2011.2167964.

[16] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[17] Joud Khoury, Ram Ramanathan, Dan McCloskey, Russell Smith, and Timothy Campbell. Radarmac: Mitigating radar interference in self-driving cars. In *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9, 2016. doi: 10.1109/SAHCN.2016.7733011.

[18] Preeti Kumari, Junil Choi, Nuria González-Prelcic, and Robert W. Heath. Ieee 802.11ad-based radar: An approach to joint vehicular communication-radar system. *IEEE Transactions on Vehicular Technology*, 67(4):3012–3027, 2018. doi: 10.1109/TVT.2017.2774762.

[19] Utku Kumbul, Nikita Petrov, Fred van der Zwan, Cicero S. Vaucher, and Alexander Yarovoy. Experimental investigation of phase coded fmcw for sensing and communications. In *2021 15th European Conference on Antennas and Propagation (EuCAP)*, pages 1–5, 2021. doi: 10.23919/EuCAP51087.2021.9411464.

[20] Franz Lampel, Faruk Uysal, Firat Tigrek, Simone Orru, Alex Alvarado, Frans Willems, and Alexander Yarovoy. System level synchronization of phase-coded fmcw automotive radars for radcom. In *2020 14th European Conference on Antennas and Propagation (EuCAP)*, pages 1–5, 2020. doi: 10.23919/EuCAP48036.2020.9135417.

[21] Pengfei Liu, Yimin Liu, Tianyao Huang, Yuxiang Lu, and Xiqin Wang. Decentralized automotive radar spectrum allocation to avoid mutual interference using reinforcement learning. *IEEE Transactions on Aerospace and Electronic Systems*, 57(1):190–205, 2021. doi: 10.1109/TAES.2020.3011869.

[22] Nguyen Cong Luong, Xiao Lu, Dinh Thai Hoang, Dusit Niyato, and Dong In Kim. Radio resource management in joint radar and communication: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 23(2):780–814, 2021. doi: 10.1109/COMST.2021.3070399.

[23] K. V. Mishra, M. R. Bhavani Shankar, V. Koivunen, B. Ottersten, and S. A. Vorobyov. Toward millimeter-wave joint radar communications: A signal processing perspective. *IEEE Signal Processing Magazine*, 36(5):100–114, 2019. doi: 10.1109/MSP.2019.2913173.

[24] Angelia Nedic, Asuman Ozdaglar, and Pablo A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010. doi: 10.1109/TAC.2010.2041686.

[25] NHTSA. National highway traffic safety association, 2019. URL `https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety`.

[26] NXP. Nxp semiconductors, 2021. URL `https://www.nxp.com/applications/automotive/adas-and-highly-automated-driving/automotive-radar-systems:RADAR-SYSTEMS`.

[27] Roland Oechslin, Sebastian Wieland, Andreas Zutter, Uwe Aulenbacher, and Peter Wellig. Fully adaptive resource management in radar networks. In *2020 IEEE Radar Conference (RadarConf20)*, pages 1–6, 2020. doi: 10.1109/RadarConf2043947.2020.9266440.

[28] J. Overdevest, F. Laghezza, F. Jansen, and A. Filippi. Radar waveform coexistence: Interference comparison on multiple-frame basis. In *2020 17th European Radar Conference (EuRAD)*, pages 168–171, 2021. doi: 10.1109/EuRAD48048.2021.00052.

[29] Sujeet Milind Patole, Murat Torlak, Dan Wang, and Murtaza Ali. Automotive radars: A review of signal processing techniques. *IEEE Signal Processing Magazine*, 34(2):22–35, 2017. doi: 10.1109/MSP.2016.2628914.

[30] Bryan Paul, Alex R. Chiriyath, and Daniel W. Bliss. Survey of rf communications and sensing convergence research. *IEEE Access*, 5:252–270, 2017. doi: 10.1109/ACCESS.2016.2639038.

[31] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for pomdps. *J. Artif. Int. Res.*, 32(1):663–704, July 2008. ISSN 1076-9757.

[32] Alexander Schwab, Lisa-Marie Reichelt, Philipp Welz, and Jan Lunze. Experimental evaluation of an adaptive cruise control and cooperative merging concept. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 318–325, 2020. doi: 10.1109/CCTA41146.2020.9206156.

[33] Max Ian Schöpe, Hans Driessen, and Alexander Yarovoy. Optimal balancing of multi-function radar budget for multi-target tracking using lagrangian relaxation. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8, 2019.

[34] Max Ian Schöpe, Hans Driessen, and Alexander Yarovoy. Multi-task sensor resource balancing using lagrangian relaxation and policy rollout. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–8, 2020. doi: 10.23919/FUSION45008.2020.9190546.

[35] Ersin Selvi, R. Michael Buehrer, Anthony Martone, and Kelly Sherbondy. Reinforcement learning for adaptable bandwidth tracking radars. *IEEE Transactions on Aerospace and Electronic Systems*, 56(5): 3904–3921, 2020. doi: 10.1109/TAES.2020.2987443.

[36] Sida Song, Sha Ma, Lutao Gao, Lei Wan, and Yong Wu. Cooperative automotive radar system for inter-radar interference avoidance. In *2019 International Radar Conference (RADAR)*, pages 1–6, 2019. doi: 10.1109/RADAR41533.2019.171413.

[37] Charles E. Thornton, Mark A. Kozy, R. Michael Buehrer, Anthony F. Martone, and Kelly D. Sherbondy. Deep reinforcement learning control for radar detection and tracking in congested spectral environments. *IEEE Transactions on Cognitive Communications and Networking*, 6(4):1335–1349, 2020. doi: 10.1109/TCCN.2020.3019605.

[38] Faruk Uysal and Sasanka Sanka. Mitigation of automotive radar interference. In *2018 IEEE Radar Conference (RadarConf18)*, pages 0405–0410, 2018. doi: 10.1109/RADAR.2018.8378593.

[39] Bas van der Werk. Approximately Optimal Radar Resource Management for Multi-Sensor Multi-Target Tracking. Master's thesis, TU Delft, Delft, The Netherlands, 2021.

[40] WIkipedia. Markov decision process, 2021. URL `https://upload.wikimedia.org/wikipedia/commons/2/21/Markov_Decision_Process_example.png`.

[41] Haowei Zhang, Weijian Liu, Binfeng Zong, Junpeng Shi, and Junwei Xie. An efficient power allocation strategy for maneuvering target tracking in cognitive mimo radar. *IEEE Transactions on Signal Processing*, 69:1591–1602, 2021. doi: 10.1109/TSP.2020.3047227.

[42] Feng Zhao, Jaewon Shin, and J. Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, 19(2):61–72, 2002. doi: 10.1109/79.985685.