

AFRL-RY-WP-TR-2020-0286

ADVANCING FULLY ADAPTIVE RADAR CONCEPTS FOR REAL-TIME PARAMETER ADAPTION AND DECISION MAKING

Peter John-Baptiste Jr.
The Ohio State University

SEPTEMBER 2020
Final Report

Approved for public release; distribution is unlimited.

See additional restrictions described on inside pages

© 2020 Peter John-Baptiste Jr.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS**.

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
September 2020	Thesis	16 September 2020 –16 September 2020

4. TITLE AND SUBTITLE
ADVANCING FULLY ADAPTIVE RADAR CONCEPTS FOR REAL-TIME PARAMETER ADAPTION AND DECISION MAKING

5a. CONTRACT NUMBER
FA8650-18-C-1632

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
65502F/62204F

6. AUTHOR(S)
Peter John-Baptiste Jr.

5d. PROJECT NUMBER
3005/6095

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
Y1RH

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Ohio State University
281 W Lane Ave.
Columbus, OH 43210

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command
United States Air Force

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYMF

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2020-0286

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
PAO case number 88ABW-2020-2842, Clearance Date 16 September 2020. © 2020 Peter John-Baptiste Jr.. Presented in partial fulfillment of the requirements for the degree Doctor of Philosophy in Electrical and Computer Engineering of the Graduate School of The Ohio State University. This work was funded in whole or in part by Department of the Air Force. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government. Report contains color.

14. ABSTRACT

Cognitive or Fully Adaptive Radar (FAR) is an area of research that is inspired by biological systems and focuses on developing a radar system capable of autonomously adapting its characteristics to achieve a variety of different tasks such as improved environment sensing and spectral agility. The FAR framework implements a dynamic feedback loop (sense, learn, adapt) within a software defined radar (SDR) system and the environment that emulates a Perception Action Cycle (PAC). The implementa-tion of the FAR framework on SD Rs relies on solver-based optimization techniques for their action selection. However, with the increase of optimization complexity, there becomes a heavy impact on time to solution convergence, which limits real-time ex-perimentation. Additionally, many "cognitive radars" lack a memory component resulting in repetitive optimization routines for similar /familiar perceptions. Using an existing model of the FAR framework, a neural network inspired refine-ment is made. Through the use of neural networks, a subset of machine learning, and other machine learning concepts, a substitution is made for the solver-based optimiza-tion component for the FAR framework applied to single target tracking. Static feed-forward neural networks and dynamic neural networks were trained and implemented in both a simulation and experimentation environment. Performance comparisons between the neural network and the solver-based optimization approaches show that the static neural network based approach had faster runtimes which lead to more per-ceptions and sometimes better performance through lower resource consumption. A comparison between the simulation results of the static feed-forward neural network, the dynamic recurrent neural network, and the solver is also made. These compar-isons further support the notion of neural networks being able to provide a memory component for cognitive radar through the incorporation of learning, moving toward truly cognitive radars. Additional research was also performed to further show the advantages of neural networks in radar applications of rapid waveform generation. The FAR framework is also extended from the single-target tracking FAR frame-work to a multiple target tracking implementation. The multi-target implementation of the FAR framework displays the benefits of adaptive radar techniques for multiple target environments where complexity is increased due to the increased number of targets present in the scene as well as the need to resolve all targets. Refinements and additions were made to the existing cost functions and detection/tracking frameworks due to the multiple target environment. Experimental and simulated results demon-strate the benefit of the FAR framework by enabling a robust adaptive algorithm that results in improved tracking and efficient resource management for a multiple target environment. In addition to this, the Hierarchical Fully Adaptive Radar (HFAR) framework was also applied to the problem of resource allocation for a system needing to per-form multiple tasks. The Hierarchical Fully Adaptive Radar for Task Flexibility (HFAR-TF)/Autonomous Decision Making (ADM) work applies the HFAR frame-work to a system needing to engage in balancing multiple tasks: target tracking, classification and target intent discernment ("friend", "possible foe", and "foe").

15. SUBJECT TERMS
cognitive radar, resource management, target tracking

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Muralidhar Rangaswamy
Unclassified	Unclassified	Unclassified	SAR	294	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Advancing Fully Adaptive Radar Concepts for Real-Time Parameter Adaptation and Decision Making

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Peter John-Baptiste Jr, B.S., M.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2020

Dissertation Committee:

Dr.Joel T. Johnson, Advisor

Dr.Graeme E. Smith,Co-Advisor

Dr. Lee Potter

Dr. Megan Roberts

Approved for public release; distribution is unlimited.

© Copyright by

Peter John-Baptiste Jr

2020

Approved for public release; distribution is unlimited.

Abstract

Cognitive or *Fully Adaptive Radar* (FAR) is an area of research that is inspired by biological systems and focuses on developing a radar system capable of autonomously adapting its characteristics to achieve a variety of different tasks such as improved environment sensing and spectral agility. The FAR framework implements a dynamic feedback loop (sense, learn, adapt) within a *software defined radar* (SDR) system and the environment that emulates a Perception Action Cycle (PAC). The implementation of the FAR framework on SDRs relies on solver-based optimization techniques for their action selection. However, with the increase of optimization complexity, there becomes a heavy impact on time to solution convergence, which limits real-time experimentation. Additionally, many *"cognitive radars"* lack a memory component resulting in repetitive optimization routines for similar/familiar perceptions.

Using an existing model of the FAR framework, a neural network inspired refinement is made. Through the use of neural networks, a subset of machine learning, and other machine learning concepts, a substitution is made for the solver-based optimization component for the FAR framework applied to single target tracking. Static feedforward neural networks and dynamic neural networks were trained and implemented in both a simulation and experimentation environment. Performance comparisons between the neural network and the solver-based optimization approaches show that

Approved for public release; distribution is unlimited.

the static neural network based approach had faster runtimes which lead to more perceptions and sometimes better performance through lower resource consumption. A comparison between the simulation results of the static feed-forward neural network, the dynamic recurrent neural network, and the solver is also made. These comparisons further support the notion of neural networks being able to provide a memory component for cognitive radar through the incorporation of learning, moving toward *truly cognitive* radars. Additional research was also performed to further show the advantages of neural networks in radar applications of rapid waveform generation.

The FAR framework is also extended from the single-target tracking FAR framework to a multiple target tracking implementation. The multi-target implementation of the FAR framework displays the benefits of adaptive radar techniques for multiple target environments where complexity is increased due to the increased number of targets present in the scene as well as the need to resolve all targets. Refinements and additions were made to the existing cost functions and detection/tracking frameworks due to the multiple target environment. Experimental and simulated results demonstrate the benefit of the FAR framework by enabling a robust adaptive algorithm that results in improved tracking and efficient resource management for a multiple target environment.

In addition to this, the *Hierarchical Fully Adaptive Radar* (HFAR) framework was also applied to the problem of resource allocation for a system needing to perform multiple tasks. The *Hierarchical Fully Adaptive Radar for Task Flexibility* (HFAR-TF)/*Autonomous Decision Making* (ADM) work applies the HFAR framework to a system needing to engage in balancing multiple tasks: target tracking, classification and target intent discernment (*"friend", "possible foe", and "foe"*).

Approved for public release; distribution is unlimited.

The goal of this Ph.D. is to combine these objectives to form a basis for establishing a method of improving current cognitive radar systems. This is done by fusing machine learning concepts and fully adaptive radar theory, to enable real-time operation of *truly cognitive* radars, while also advancing adaptive radar concepts to new applications.

Approved for public release; distribution is unlimited.

To my parents, Eddie & Sophia Parrilla and Peter Sr. for their constant support and for patiently listening when I talked about my work. To my brothers and sisters: Melissa, Sonja, Shidoya, Shideya, Sandro (Junie), Nile, and Josh for always keeping me sane and being there when I need them. As well as a special dedication to my little sister Nancy for her constant love. To my friends and roommates, particularly Erin, Eric, Origeinal, Teekay, and Michelangelo for being available for conversations to break up the work day. To my lab group: Moayad, Joe, DeGrafth, and Jakob for providing me an amazing work environment and always being willing to brainstorm with me.

Approved for public release; distribution is unlimited.

Acknowledgments

There are multiple people who have made this Ph.D. completion possible. I want to acknowledge Dr.Muralidhar Rangaswamy and Dr.Aaron Jones for their patronage. To Dr.Kristine Bell, I would like to give an abundance of gratitude for both lending her knowledge and advice, anytime it was needed. I would like to recognize Dr.Adam Mitchell as well, for lending his counsel during the beginning of my Ph.D. I would also like to extend my gratitude to my two advisors: Dr.Graeme Smith, for offering me the chance to work in cognitive radar and advancing/developing my expertise through the Ph.D. (as well as teaching me that most of the difficult tasks encountered in research are *part of the Ph.D.*); and Dr.Joel Johnson for advising, mentoring and further supporting me, allowing me to complete my research. To Dr.Lee Potter, I would like to extend my gratitude for his advice and notes that assisted in the development of this work. Different components of this work were supported by industrial and government membership fees to the Center for Surveillance Research (CSR), a National Science Foundation Industry and University Cooperative Research Program (I/UCRC); the United States Air Force Sensors Directorate; and the Air Force Research Lab Autonomy Technology Research Center (ATRC) Internship Program. However, the views and opinions expressed in this dissertation are those of the author and do not necessarily reflect the official policy or position of any agency of the U.S. government.

Approved for public release; distribution is unlimited.

Vita

2012 ..... HSD - St.Croix Education Complex
Christiansted, USVI

2016 ..... B.S in EE - Liberty University
Lynchburg, VA

2020 ..... M.S.in EE - The Ohio State University
Columbus, OH

2016-present ..... Graduate Research Associate
ElectroScience Laboratory

Publications

Journal Publications

P. John-Baptiste, G. E. Smith, K. L. Bell, and J. T. Johnson, "Fully Adaptive Radar for Multiple Target Tracking". *IEEE Aerospace and Electronic Systems Magazine.* (Manuscript in Preparation)

P. John-Baptiste, G. E. Smith, and J. T. Johnson, "Neural Network-Based Control of an Adaptive Radar". *IEEE Transactions on Aeorospace and Electronic Systems.* (In Review)

P. John-Baptiste, E. Zelnio, and G. E. Smith, "Parameter Selection using Neural Networks for SAR Image Regularization". *IEEE Access.* (Manuscript in Preparation)

P. John-Baptiste, A. Brandewie, J. Vinci, K. Bell , and J. T. Johnson, "Fully Adaptive Radar for Multi-Task Radar Functionality". *IET Radar, Sonar Navig.*, 2020 (Manuscript in Preparation).

Approved for public release; distribution is unlimited.

Conference Publications

P. John-Baptiste, E. Zelnio, and G. E. Smith, "Using deep learning for SAR image optimization". *Proc. SPIE 10647*, Algorithms for Synthetic Aperture Radar Imagery XXV, 106470D (27 April 2018).

A. M. Jones, C. P. Horne, H. D. Griffiths, G. Smith, A. Mitchell, and P. John-Baptiste, "Experimentation of an adaptive and autonomous RF signalling strategy for detection". *2018 IEEE Radar Conference (RadarConf18)*,Oklahoma City,OK,2018,pp.1213-1218.doi: 10.1109/RADAR.2018.8378735.

P. John-Baptiste and G. E. Smith, "Utilizing Neural Networks for Fully Adaptive Radar". *2019 IEEE Radar Conference (RadarConf19)*,Boston,MA,2019,pp. 1-6.

P. John-Baptiste, G. E. Smith, A. M. Jones, and T. Bihl, "Rapid Waveform Design Through Machine Learning". *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*,Le gosier, Guadeloupe, 2019, pp. 659-663.

S.Z. Gurboz, S.Brüggenwirth, P. John-Baptiste and G. E. Smith, "Neural Networks & Machine Learning in Cognitive Radar". *2020 IEEE Radar Conference (RadarConf20)*, Florence, Italy, 2020 (Accepted).

Fields of Study

Major Field: Electrical and Computer Engineering

Studies in:
- Radar Signal Processing
- Machine Learning

Approved for public release; distribution is unlimited.

Contents

Page

Abstract ii

Dedication v

Acknowledgments vi

Vita vii

List of Tables xiii

List of Figures xv

1. Introduction 1

    1.1 Overview 1
    1.2 Motivations and Contributions 3
    1.3 Outline 6

2. Background 8

    2.1 Radar Background 8
        2.1.1 Radar Basics 8
        2.1.2 State Estimation 24
        2.1.3 MAP-PF 27
    2.2 Neural Network Background 33
        2.2.1 Neural Network Basics 33
        2.2.2 Different Approaches to Learning in Neural Networks 38
        2.2.3 Supervised Learning Approaches 41
        2.2.4 Long Short Term Memory Recurrent Neural Network 43
    2.3 FAR and HFAR 47
    2.4 Optimization and Statistics Background 50

Approved for public release; distribution is unlimited.

2.4.1 FAR and HFAR Optimization 50
2.4.2 Hypothesis Testing and T-tests 53
2.5 Summary 54

3. Literature Review 56

3.1 Cognitive Science 56
3.2 Cognitive/Fully Adaptive Radar 57
3.3 Neural Networks for Radar Applications 60
3.4 Combining Neural Networks and Cognitive Radar 61

4. FARMS Algorithm and CREW Experimental Testbed 64

4.1 FARMS-Fully Adaptive Radar Modeling and Simulation 64
4.2 CREW Testbed 68
4.3 FARMS Data Flow 70

5. Neural Network Based Control of an Adaptive Radar 73

5.1 Motivation 74
5.2 Approach 82
5.2.1 Optimization-Based FAR 82
5.2.2 Optimization-Based HFAR 85
5.2.3 Neural Network Implementation Details 88
5.3 Results 93
5.3.1 Static Neural Network FAR Analysis 94
5.3.2 Static Neural Network HFAR Analysis 103
5.4 Conclusion 123

6. Rapid Waveform Generation using Neural Networks 126

6.1 Motivation 127
6.2 Approach 129
6.2.1 Neural Network Theory 129
6.2.2 Long Short Term Memory Neural Network Approach 133
6.3 Results 134
6.3.1 Preliminary Feasibility Results 134
6.3.2 ERA-NN Results 140
6.4 Conclusion 149

7. Fully Adaptive Radar for Multiple Target Tracking 150

7.1 Motivation 151

Approved for public release; distribution is unlimited.

7.2 Approach . . . 153
7.2.1 Multiple Target Tracking (MTT) Model . . . 154
7.2.2 Fisher Information Matrix . . . 161
7.2.3 Fully Adaptive Radar Multiple Target Tracking Optimization 165
7.2.4 *Fully Adaptive Radar Multiple Target Tracking* (FAR-MTT) Recursion . . . 173
7.3 Results . . . 174
7.3.1 Case 1 . . . 175
7.3.2 Case 2 . . . 182
7.3.3 Case 3 . . . 187
7.3.4 Case 4 . . . 193
7.4 Conclusion . . . 197

8. Fully Adaptive Radar for Multi-Task Radar Functionality . . . 199

8.1 Motivation . . . 200
8.2 Approach . . . 203
8.2.1 Framework Implementation . . . 204
8.2.2 FAR Multi-Task Tracking . . . 207
8.2.3 FAR Multi-Task Classification . . . 210
8.2.4 Optimization and Decision Making Discussion . . . 212
8.2.5 Simulation Setup . . . 219
8.2.6 Environment Modeling . . . 221
8.3 Results . . . 223
8.3.1 HFAR-TF Results . . . 225
8.3.2 HFAR-ADM Results . . . 228
8.4 Conclusion . . . 231

9. Conclusions and Future Work . . . 233

9.0.1 Future Work . . . 237

Appendices 240

A. FAR NN Appendix . . . 240

A.1 Optimization Reliability . . . 240

B. MTT Appendix . . . 245

B.1 Fisher Information Matrix Derivation . . . 245
B.2 Pre-Whitening Derivation . . . 252

Approved for public release; distribution is unlimited.

Bibliography 256

Approved for public release; distribution is unlimited.

List of Tables

Table		Page
5.1	*Fully Adaptive Radar with Neural Network Based Control* (FAR-NN) Operating Parameters	79
5.2	Measurement Cost Function Parameters	79
5.3	*Hierarchical Fully Adaptive Radar with Neural Network Based Control* (HFAR-NN) Operating Parameters	80
5.4	Measurement Cost Function Parameters	80
5.5	Cases Examined for FAR Static NN: 120s run	80
5.6	HFAR Cases Examined: 1200s run	80
5.7	FAR-NN Training Database	90
5.8	HFAR-NN Training Database	90
5.9	Dynamic NN Cases Examined: 1200s run	93
5.10	Network Structure	93
5.11	Hypothesis Test Result for Performance Comparison of FAR-NN with a Static NN ($\alpha = 5 \times 10^{-2}$)	99
5.12	Average Optimization Time of Neural Network & Solver (seconds)	102
5.13	Hypothesis Test Result for Simulation Performance Comparison of HFAR-NN with a Static NN ($\alpha = 5 \times 10^{-2}$)	110

Approved for public release; distribution is unlimited.

5.14 Average Optimization Time of Neural Network & Solver (seconds) . . 123

6.1 Waveform Parameters 131

6.2 *Long Short Term Memory* (LSTM) Network Architecture 134

6.3 AutoEncoder Results on Test Dataset 137

6.4 Notched Waveform Results on Test Dataset 139

6.5 Waveform Parameters 143

6.6 Notched Waveform Results on Test Dataset 148

7.1 Measurement Cost Function Parameters 172

7.2 Adaptive Radar Operating Parameters 175

7.3 Static Radar Operating Parameters 175

8.1 Target Priority Table 216

8.2 Radar Operating Parameters: Fixed Case 221

8.3 Radar Operating Parameters: Adapt Case 221

8.4 Target Scatterer Motion Parameters 223

A.1 Summary of Monte Carlo Trials for Optimization Reliability 241

Approved for public release; distribution is unlimited.

List of Figures

Figure **Page**

2.1 Basic radar representation 9

2.2 Basic radar representation 11

2.3 Pulsed Waveform Example 15

2.4 Pulse-Doppler Data Matrix 17

2.5 Time Domain of LFM Waveform 19

2.6 Phase vs. Time (top) and Frequency vs. Time (bottom) of the LFM . 20

2.7 Frequency Spectrum of LFM Waveform 20

2.8 Matched Filter Response of LFM Waveform 22

2.9 LFM Range Doppler Surface 23

2.10 Biological Neuron vs. Artificial Neuron 34

2.11 Three Layer Neural Network 37

2.12 Recurrent/Feedback Networks and Feed-forward Networks 37

2.13 Supervised Learning Diagram 39

2.14 Reinforcement Learning Diagram 40

2.15 Back-propagation Example 42

Approved for public release; distribution is unlimited.

2.16 Graphic Diagram of Unrolled LSTM in Time with hidden unit h and input x 44

2.17 Single LSTM Block in time with highlighted components for each composite layer within the LSTM: Forget Gate (red), Input Gate (green), Tanh layer (purple), Updated Cell State (brown), and filtered output (dark blue) 45

2.18 Fully Adaptive Radar Framework 47

2.19 Hierarchical Fully Adaptive Radar Framework 48

2.20 HFAR Descriptive Framework 50

4.1 Swerling Model I and II 66

4.2 Swerling Model III and IV 66

4.3 CREW RF Component Diagram 69

4.4 Data Flow Diagram for FARMS Single PAC 71

5.1 Fully Adaptive Radar Modeling and Simulation Framework 75

5.2 FAR Diagram with Neural Network 84

5.3 FAR Neural Network Training Diagram 91

5.4 HFAR Neural Network Training Diagram 92

5.5 4 Parameter Adaptation over Time: [Sim] 95

5.6 Tracks and Performance Results 96

5.7 Histograms of measurement and performance costs for both simulated and measured datasets 98

5.8 Empirical cdf of measurement and performance costs for both simulated and measured datasets 101

5.9 Empirical cdf of executive costs for both simulated and measured datasets 102

Approved for public release; distribution is unlimited.

5.10 4 Parameter Adaptation over Time: [Sim] 105

5.11 4 Parameter Adaptation Tracks and Performance [Sim] 107

5.12 4 Parameter Adaptation over Time: [Sim] 108

5.13 Overlaid histograms of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Sim] 109

5.14 Overlaid histograms of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Sim] 110

5.15 Overlaid empirical cdfs of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Sim-Tier 1, PAC 1] . . . 111

5.16 Overlaid empirical cdfs of the Executive Cost (top-a), Measurement Cost (middle-b), and Performance Cost (bottom-c): [Sim-Tier 1, PAC 2] 112

5.17 Single LSTM Block in time with highlighted components for each composite layer within the LSTM: Forget Gate (red), Input Gate (green), Tanh layer (purple), Updated Cell State (brown), and filtered output (dark blue) 113

5.18 4 Parameter Adaptation over Time: [Sim] 114

5.19 Tracks and Performance Results 115

5.20 Overlaid histograms of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Case 1- Sim] 119

5.21 Overlaid histograms of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Case 2-Sim] 120

5.22 Overlaid empirical cdfs of the Executive Cost (top-a), Measurement Cost (middle-b), and Performance Cost (bottom-c): [Case 1-Sim] . . 122

5.23 Overlaid empirical cdfs of the Executive Cost (top-a), Measurement Cost (middle-b), and Performance Cost (bottom-c): [Case 2-Sim] . . 123

6.1 Congested/Contested and Dynamic RF Environment 129

Approved for public release; distribution is unlimited.

6.2 Auto-encoder Block Diagram 131

6.3 Notched Waveform Block Diagram 131

6.4 Output of LSTM Time Domain Response Autoencoder Task 135

6.5 Output of LSTM Freq. Domain Response Autoencoder Task 136

6.6 Output of LSTM Time Domain Response Notched Waveform Task . 138

6.7 Output of LSTM Freq. Domain Response Notched Waveform Task . 138

6.8 Output of LSTM Freq. Domain Response Notched Waveform Task . 142

6.9 Output of LSTM Time Domain Response for Dynamic Interference at Location 1 ERA Comparison 144

6.10 Output of LSTM Freq. Domain Response for Dynamic Interference at Location 1 ERA Comparison 145

6.11 Output of LSTM Time Domain Response for Dynamic Interference at Location 2 ERA Comparison 146

6.12 Output of LSTM Freq. Domain Response for Dynamic Interference at Location 2 ERA Comparison 147

7.1 Target Model: Two Targets (for Case 1) 176

7.2 Target Model: Two Targets (for Case 2 & 3) 176

7.3 Target Model: Two Targets (for Case 4) 176

7.4 Parameters over Time: Adaptive Parameters **Case 1** [**Sim**] 177

7.5 Tracks over Time: **Case 1** [**Sim**] 178

7.6 Normalized Doppler and Performance Goals: **Case 1** [**Sim**] 180

7.7 Penalized RD Surface at $\sim$ 14.9 sec runtime: **Case 1** [**Sim**] 181

Approved for public release; distribution is unlimited.

7.8 Penalized RD Surface at $\sim$ 15.1 sec runtime: **Case 1** [**Sim**] 182

7.9 Parameters over Time: Adaptive Parameters **Case 2** [**Sim**] 185

7.10 Tracks over Time: Adaptive Parameters **Case 2** [**Sim**] 186

7.11 Normalized Doppler and Performance Goals: **Case 2** [**Sim**] 187

7.12 w_{TP} over Time: Adaptive Parameters 188

7.13 Parameters over Time: Adaptive Parameters **Case 3** [**Exp**] 189

7.14 Tracks over Time: **Case 3** [**Exp**] 191

7.15 Normalized Doppler and Performance Goals: **Case 3** [**Exp**] 192

7.16 Parameters over Time: Adaptive Parameters **Case 4** [**Sim**] 194

7.17 Tracks over Time: **Case 4** [**Sim**] 194

7.18 Normalized Doppler and Performance Goals: **Case 4** [**Sim**] 195

8.1 HFAR-TF Diagram 206

8.2 HFAR-ADM Diagram 207

8.3 Training Data Feature Space 211

8.4 HFAR-TF Decision Making Process 215

8.5 HFAR-ADM Decision Making Process 215

8.6 Target Motion 220

8.7 Target motion over time 222

8.8 Fixed Parameter: Track and Class Sensor Parameters [TaskFlex] . . . 226

8.9 4 Parameter Adaptation: Track and Class Sensor Parameters [TaskFlex] 227

8.10 4 Parameter Adaptation [ADM] 229

Approved for public release; distribution is unlimited.

A.1 Primary Solver: Local Solver 242

A.2 Primary Solver:Global Solver 242

A.3 Primary Solver: Local Solver [Overlaid histograms of Function Evaluation (top) , Function Difference between solvers (bottom)] 243

A.4 Primary Solver:Global Solver [Overlaid histograms of Function Evaluation (top) , Function Difference between solvers (bottom)] 244

Approved for public release; distribution is unlimited.

List of Acronyms

ADM *Autonomous Decision Making*

ADMM *alternating direction method of multipliers*

ANN *Artificial Neural Network*

ATR *Autonomous Target Recognition*

AWGN *Additive White Gaussian Noise*

ATR *Automatic Target Recognition*

BFGS *Broydon-Fletcher-Goldfarbo-Shanno*

BIM *Bayesian Information Matrix*

cdf *cumulative distribution function*

CNN *Convolutional Neural Network*

CPI *Coherent Processing Interval*

CREW *Cognitive Radar Engineering Workspace*

CRB *Cramer-Rao Bound*

CV-CNN *Complex-Valued Convolutional Neural Network*

CW *continuous wave*

DAWG *digital arbitrary waveform generators*

DoA *Direction of Arrival*

Approved for public release; distribution is unlimited.

DFT *discrete Fourier transform*

EM *electromagnetic*

ERA *Error Reduction Algorithm*

ESL ElectroScience Laboratory

FAR *Fully Adaptive Radar*

FARMS *Fully Adaptive Radar Modeling and Simulation*

FAR-MTT *Fully Adaptive Radar Multiple Target Tracking*

FAR-NN *Fully Adaptive Radar with Neural Network Based Control*

FAR-STT *Fully Adaptive Radar Single Target Tracking*

FIM *Fisher Information Matrix*

FFT *fast Fourier transform*

GPU *graphical processing unit*

GR-NN *Generalized Regression Neural Network*

HFAR *Hierarchical Fully Adaptive Radar*

HFARMS *Hierarchical Fully Adaptive Radar Modeling and Simulation*

HFAR-ADM *Hierarchical Fully Adaptive Radar for Autonomous Decision Making*

HFAR-NN *Hierarchical Fully Adaptive Radar with Neural Network Based Control*

HFAR-TF *Hierarchical Fully Adaptive Radar for Task Flexibility*

Approved for public release; distribution is unlimited.

IF *intermediate frequency*

IQ *In Phase and Quadrature*

LFM *linear frequency modulated*

LM *Levenberg-Marquardt*

LRDT *Likelihood Ratio Detector Tracker*

LSTM *Long Short Term Memory*

LSTM-RNN *long short term memory recurrent neural network*

MAP *Maximum A Priori*

MAP-PF *Maximum A Priori Penalty Function*

MDA *multiple discriminant analysis*

MLE *maximum likelihood estimation*

MMSE *Minimum Mean-Squared Error*

MSE *Mean-Squared Error*

MSTAR *Moving and Stationary Target Acquisition and Recognition*

MTI *moving target indication*

MTT *Multiple Target Tracking*

NaN *Not A Number*

NCV *Nearly Constant Velocity*

Approved for public release; distribution is unlimited.

NMSE *Normalized Mean Squared Error*

NN *Neural Network*

PAC Perception Action Cycle

PAPR *peak-to-average-power ratio*

PCRB *Posterior Cramer-Rao Bound*

pdf *probability density function*

pmf *probability mass function*

PIM *Predicted Information Matrix*

PRI *Pulse Repetition Interval*

PC-CRB *Predicted Conditional Cramer-Rao Bound*

PC-CRLB *Predicted Conditional Cramer-Rao Lower Bound*

P-CRB *Predicted Cramer-Rao Bound*

PC-BIM *Predicted Conditional Bayesian Information Matrix*

QoS *Quality of Service*

QP *quadratic programming*

RCS *radar cross section*

RD *Range/Doppler*

RF *radio frequency*

Approved for public release; distribution is unlimited.

RRE *radar range equation*

RNN *recurrent neural network*

RUWO *Reiterative Uniform Waveform Optimization*

ReLU *rectified linear unit*

SAR *Synthetic Aperture Radar*

SDR *software defined radar*

SINR *signal-to-interference-plus-noise ratio*

SIR *signal-to-interference ratio*

SNR *signal-to-noise-ratio*

sqp *sequential quadratic programming*

STT *Single Target Tracking*

TF *Task Flexibility*

B *Bandwidth*

F_{c} *Center Frequency*

f_{PRF} *Pulse Repetition Frequency*

P_{t} *Transmit Power*

N_{p} *Number of Pulses*

τ *Pulse Width*

Approved for public release; distribution is unlimited.

Chapter 1: Introduction

1.1 Overview

Developments in modern radar systems have led to *software defined radar* (SDR) systems capable of dynamic feedback loop behavior, differing from traditional radars. The feed-forward nature of traditional radars depend on the assumed characteristics of the perceived environment which produce fixed parameter settings to guarantee a predetermined *signal-to-interference-plus-noise ratio* (SINR) [1] or radar task performance. However, dynamic/changing environments can result in decreased task performance or mismanagement of system resources. Lack of adaptive control of radar front-end characteristics can result in either increased signal processing work at the radar back-end, heavy reliance on the radar operator or static radar system parameters set according to the worst-case scenario.

The *Fully Adaptive Radar* (FAR) framework aims at using the sensor parameter diversity enabled by modern SDR systems to allow autonomous adaption of the radar waveform characteristics in order to achieve a better environment perception and radar task performance [2]. The autonomous nature of the FAR framework also shifts towards moving the role of the radar operator to an advisory one as well as lessening the burden of additional signal processing for target information extraction.

Approved for public release; distribution is unlimited.

The FAR framework achieves autonomous adaption by attempting to emulate the neuroscience concept of cognition present in animals and humans. As discussed by [2, 3, 4, 5], cognitive processes must include five primary elements: perception, attention and analysis(intelligence), action and memory. In [6], Haykin discusses the difference between traditional active radar, FAR and cognitive radar. In particular, the difference between cognitive and FAR is highlighted. While FAR is able to achieve a better sense of its environment using a feedback link to connect the receiver perceived environment to the transmitter waveform probing, it is limited in its intelligence through its lack of long-term memory which is needed for *"true"* learning.

Focusing on the feedback nature of cognitive radar, it should be noted that optimization stands out as one of the most critical components as it enables the adaption of the radar front-end characteristics. In order to perform the optimization in cognitive radar processing, nonlinear functions are often used. These nonlinear functions are realized in optimization blocks that can perform radar parameter selection and updates through the minimization of nonlinear constrained objective functions. For the FAR framework, this *"executive optimization"* is realized in an *"executive processor block"* that attempts to balance cost functions capturing the task-based performance of the radar system (performance cost) and the consumption of sensor resources (measurement cost) in a *Quality of Service* (QoS) approach.

For its role in the FAR framework, the executive optimization is seen as the most crucial component [7]. Learning can be understood as combining attention, analysis and memory [8]. Using this view of learning, one can begin to see these components of learning being captured in the executive optimization. In the FAR framework, through combining attention and analysis achieved through tracking using the target

Approved for public release; distribution is unlimited.

state and the a priori knowledge (memory) of past observations to select an optimal parameter index choice, the executive processor implements *"limited learning"*. Due to the nature of the sensor parameter selection invoked in the executive processor, radar waveform parameters are mapped to radar tasks and goal performance, given a priori knowledge [7]. Additionally, due to this optimization based adaption, as the complexity of the optimization increases, so does the time to solution convergence and therefore limits to real-time capability.

In conceptual similarity to FAR, machine learning is a field of research under artificial intelligence which investigates how humans can gain knowledge, or learn, and represent these concepts in machines [9]. One subtopic of machine learning that has gained popularity is neural networks, through their ability to model and realize the learning process [10]. Association, pattern recognition and relationship modeling are all tasks that neural networks are efficient at, which can be used to provide a lower impact on system processing and begin learning through recognition/memory [11].

1.2 Motivations and Contributions

Learning proves to be a key component in cognitive systems, leading to the belief that learning is a principal component for cognitive radar. In [5], learning is defined as using past information to improve one's local measure of success. Using a collection of past observations and prior information to inform present and future actions, shows that learning is linked to memory. Nevertheless, in order to adequately learn from memory and actions, real-time capability and performance must be feasible. As discussed previously, due to the optimizations used for action selection, it can be seen

Approved for public release; distribution is unlimited.

that as the complexity of the problem rises so does the computational cost of the optimization. High computational cost and lack of memory challenge the realization of a *"truly"* cognitive system.

Many developments have been made in the areas of FAR and cognitive radar research; however, most center on adaptive systems that lack long-term memory and association. Similarly, many developments have been made in neural network and machine learning based radar research, but most focus on classification and image recognition based problems. This Ph.D. research will focus on showing how including regression-based neural networks can not only improve the current performance of FAR through lower impact to system processing but also lead to developing a *"truly"* cognitive system by including stronger notions of memory and extending this into demonstrating learning to aid in cognitive radar task performance. In addition to this, a demonstration of varying applications of cognitive radar approaches are also shown.

The contributions to the field of cognitive radar by the work discussed here, are as follows:

- A neural network inspired FAR framework, *Fully Adaptive Radar with Neural Network Based Control* (FAR-NN), was developed through replacing the optimization component in the *executive processor* with a feed-forward neural network in order to lower impact to system processing and integrate its inherent recognition/memory components.

- Simulated and real-time experimental results were collected for different parameter adaption cases and comparisons drawn between the local solver implementation and the neural network to show that the static feed-forward neural

Approved for public release; distribution is unlimited.

network was able to achieve lower measurement costs, faster optimization timing and similar executive cost performance.

- Implemented a static feed-forward neural network in the *Hierarchical Fully Adaptive Radar* (HFAR) framework through simulation for sensor parameter selection at each sensor Perception Action Cycle (PAC)'s *"executive processor"* in order to lower need impact to system processing due to multiple optimizations being performed.

- Implemented a dynamic *long short term memory recurrent neural network* (LSTM-RNN) in the FAR framework through simulation for sensor parameter selection to incorporate state-based adaptability to a changing environment and stronger notions of memory into the optimization component of the neural network inspired FAR framework, FAR-NN.

- Development of a LSTM-RNN for generating low latency, near-optimal radar frequency notched waveforms in dynamic spectrally congested environments.

- Simulation results of waveform generation comparing the LSTM-RNN to an existing specialized solver called the *Error Reduction Algorithm* (ERA) demonstrate similarity between the waveform design results of the network and the algorithm, with a reduction in time to generate waveforms for the LSTM-RNN.

- An extension of the existing *Fully Adaptive Radar Single Target Tracking* (FAR-STT) framework to the *Fully Adaptive Radar Multiple Target Tracking* (FAR-MTT) implementation with modified objective functions and expanded Fisher Information Matrix/Cramer Rao Bound metrics for the multiple target environment.

Approved for public release; distribution is unlimited.

- Simulated and experimental results were collected to demonstrate the benefit of applying a fully adaptive radar approach to multiple target tracking through being able to achieve target separation and maintain individual target tracks, while consuming less measurement resources.

- Development of a HFAR framework for a radar system needing to perform multiple tasks [ex:target tracking, classification and target intent discernment (friend, possible foe, and foe)] and assign radar resources autonomously.

- Simulated results were collected to demonstrate the benefit of applying a fully adaptive radar approach to a system engaged with multiple tasks through the use of adaptive waveform parameters versus a fixed parameter set.

- Highlighted use of both simulation and experimental demonstrations of fully adaptive radar concepts to demonstrate feasible realizations of cognitive radar concepts.

1.3 Outline

The rest of this thesis is organized as follows.

Chapter 2 discusses background on basic radar, fully adaptive radar, optimization, neural networks, and statistics as relevant to the work presented in the dissertation.

Chapter 3 provides a survey of similar work in the area of cognitive radar and neural networks.

Chapter 4 discusses the *Fully Adaptive Radar Modeling and Simulation* (FARMS) environment and algorithm used in this work as well as a brief overview on the experimental test-bed used to validate the simulated results with experimental collections.

Approved for public release; distribution is unlimited.

Chapter 5 discusses the implementations of a neural network inspired FAR framework along with comparative results against the local solver used in previous FAR and HFAR implementations.

Chapter 6 reviews a neural network approach for rapid notched waveform generation with comparisons against an existing specialized solver.

Chapter 7 provides a discussion on extending the FAR framework to a multiple target environment. Both simulation and experimental results were collected to demonstrate the advantage of a adaptive radar in a multiple target tracking environment.

Chapter 8 discusses the development of fully adaptive radar approach the problem of multi-functional radar systems in which a HFAR framework is applied to a radar system needing to engage in balancing multiple tasks: target tracking, classification and target intent discernment (friend, possible foe, or foe).

Chapter 9 summarizes the results of the dissertation, and gives areas of future research that can be based on this work.

Appendix A present further details on the reliability of the local solver used in the FAR framework versus a global solver routine.

Appendix B present further details on the Fisher Information Matrix derivation and the pre-whitening derivation used in the FAR-MTT work presented in Chapter 7.

Approved for public release; distribution is unlimited.

Chapter 2: Background

This section details a background on radar fundamentals, neural networks, the FAR and HFAR concepts, optimization and statistics to give sufficient context for the developments of the following sections.

2.1 Radar Background

2.1.1 Radar Basics

The radar theory provided draws on content from [12, 13]. Radar stands for **Ra**dio **D**etection **A**nd **R**anging, used for sensing an environment through transmission and reception of *electromagnetic* (EM) waves. A radar transmits an EM wave and can detect targets after reception of the reflections of this EM wave through the receiver. As represented in Figure 2.1, a radar can detect a target through its pulse echo, and the echo has a time delay associated with it. The time delay of the target's received pulse is :

$$t_d = \frac{2R}{c} \tag{2.1}$$

where c is the speed of light ($3 \cdot 10^8$ m/s) and R is the one-way range to the target.

In (2.1), the time delay associated with the target reveals the range to the target. The phase of a point target can be modeled as $\phi = \frac{4\pi}{\lambda}R$, where $\lambda = \frac{c}{f_c}$ and f_c is the

Approved for public release; distribution is unlimited.

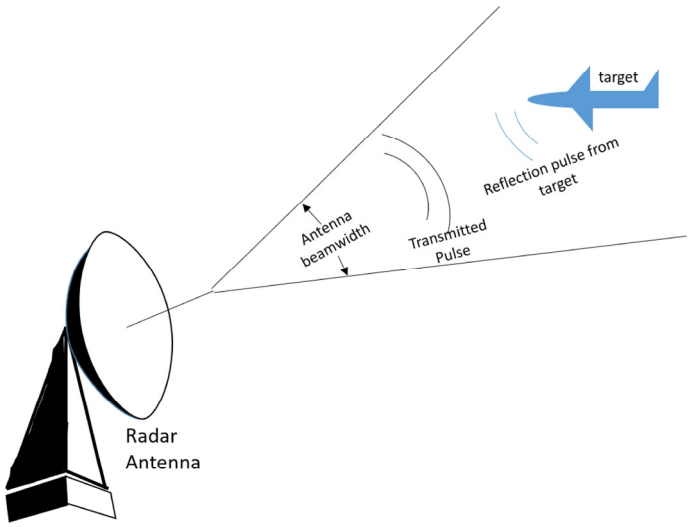


Figure 2.1: Basic radar representation

center frequency of the waveform in Hertz (Hz). In addition, there are propagation and spreading losses as the wave propagates the two-way path to the target and back. Therefore, the received waveform, $x_r(t)$, is a time shifted, amplitude scaled version of the transmit waveform, $x_t(t)$, as shown in (2.2).

$$x_r(t) = Ax_t(t - t_d)\exp(-j\phi) \tag{2.2}$$

and when substituting t_d with (2.1), the following can be obtained:

$$x_r(t) = Ax_t(t - \frac{2R}{c})\exp(-j\frac{4\pi}{\lambda}R) \tag{2.3}$$

where ϕ is the phase offset, t_d is the time delay associated with the target, R is the range associated with the target, and A is a factor accounting for the propagation terms and the target's *radar cross section* (RCS).

Approved for public release; distribution is unlimited.

For the case of a moving target, the effects of the Doppler shift must be considered. The velocity of the target results in a frequency shift of the reflected echoes, known as the Doppler frequency. This Doppler shift can be modeled by introducing a time variation in the range term, $R(t) = R_0 + vt$, of (2.3), for both the delay and phase terms. In this expression, R_0 is the initial range, and v the target's radial velocity. The Doppler frequency generated by a moving target for a mono-static, narrow-band radar can be represented as:

$$f_D = \frac{\partial}{\partial t}\frac{\phi_{app}(t)}{2\pi} = -\frac{2v}{\lambda} \tag{2.4}$$

where v is the target radial velocity and $\phi_{app}(t)$ is the apparent phase shift that varies with time t.

Now (2.2) can be rewritten to include the Doppler shift:

$$x_r(t) = Ax_t(t - t_d)exp(-j\phi)\exp(-j2\pi f_D t) \tag{2.5}$$

$$= Ax_t(t - t_d)\exp(-j\frac{4\pi}{\lambda}R)\exp(-j2\pi f_D t) \tag{2.6}$$

The fundamental building blocks of a mono-static radar as seen in Figure 2.2 are the antennas, transmit and receive elements, transmit/receive switch, low noise amplifier, mixer, detector, analog-digital converter and signal processor. The transmit and receive elements utilize the antenna horns and physical aperture to receive and transmit waveforms. The transmit/receive circulator is usually needed for a mono-static radar in order to achieve both transmit and receive without damaging the sensitive components in the receiver by providing isolation between the two. The low noise amplifier is used to boost the received signal while keeping the noise level relatively low before being fed to the mixer. The mixer is used to convert the signal

Approved for public release; distribution is unlimited.

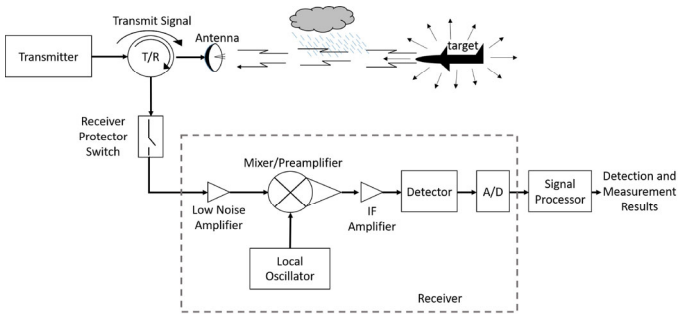


Figure 2.2: Basic radar representation

from *radio frequency* (RF) to an *intermediate frequency* (IF) using a local oscillator for easier processing. The IF signal is then amplified and sent to the analog to digital converter to convert into digital in-phase and quadrature elements. Finally, the digital elements are passed to the signal processor, where detection and other signal processing operations can take place. For this work, we focus solely on coherent systems. Coherence in this context means consistency or continuity in the radar system which translates to making use of both magnitude and phase information. Non coherency, in this context, means only using magnitude information which translates to a radar system performing some level of envelope detection leading to the phase information being discarded.

Once a wave is transmitted and received by a radar system, it is necessary to measure the signal strength or performance of the radar system in order to perform tasks such as detection. The *radar range equation* (RRE) is an important concept for evaluating the performance of a radar system. The RRE is often used to calculate the received power, or the *signal-to-noise-ratio* (SNR) for a particular system/target

Approved for public release; distribution is unlimited.

scenario. The RRE shows many relationships that are useful. The derivation for the radar range equation can be split into two parts, received power and noise power.

To begin modeling the received power we consider an isotropic radiator. Assuming an isotropic radiator, the power density measured at a target a distance R away is expressed in (2.7).

$$Q_i = \frac{P_\mathrm{t}}{4\pi R^2} \left(\frac{W}{m^2}\right) \tag{2.7}$$

where *Transmit Power* (P_t), R is the range, and P_i is incident power density.

The assumptions made in (2.7) are that the target is at a distance R away such that the waves can be approximated as plane waves which have constant phase on their surfaces.

However, recognizing that most radar's are directive in nature, the equation is rewritten to include the gain (G_t) attributed to the directivity of an antenna transmit element:

$$Q_i = \frac{P_\mathrm{t} G_\mathrm{t}}{4\pi R^2} \left(\frac{W}{m^2}\right) \tag{2.8}$$

Using the directional radiation modeled in (2.8), the re-radiation caused by the reflection of the transmitted EM wave from a target, such as a plane is expressed as:

$$Q_r = \frac{Q_i \sigma}{4\pi R^2} = \frac{P_\mathrm{t} G_\mathrm{t} \sigma}{(4\pi)^2 R^4} \left(\frac{W}{m^2}\right) \tag{2.9}$$

The reflected power density expressed in (2.9), assumes a target with a specific radar cross section σ, which defines the reflective properties of the target based on its size, shape and material.

Now, the expressions in (2.7)-(2.9)can be utilized to derive the power received by the radar, by considering that power density is expressed in $\frac{W}{m^2}$ and power is

Approved for public release; distribution is unlimited.

represented in W, so we need to multiply by the area of the aperture or the effective area, A_e:

$$A_e = \frac{G_\mathrm{r}\lambda^2}{4\pi} \tag{2.10}$$

where A_e is the effective aperture and G is the receiver gain.

Using 2.10,

$$P_\mathrm{r} = \frac{P_\mathrm{t}G_\mathrm{t}\sigma}{(4\pi)^2R^4}A_e = \frac{P_\mathrm{t}G_\mathrm{t}G_\mathrm{r}\lambda^2\sigma}{(4\pi)^3R^4} \tag{2.11}$$

In (2.11), the power received is a function of the directivity of the antenna (its effective aperture), the gain of the transmit antenna, the radar cross section of the target (which factors in the physical size, materials and shape of the target) and the range of the target. All of these factors contribute to how the target is perceived by the radar.

Secondly, the noise power of the environment which is considered to Gaussian white noise can be modeled as:

$$P_\mathrm{n} = kT_0B_\mathrm{n}F \tag{2.12}$$

where P_n is the noise power.

The relationship shown here models the perceived noise of the environment. The noise power is based on the Boltmann's constant ($k \approx 1.38 \cdot 10^{-23}$), reference temperature ($T_0 = 290K$), adjusted by a noise figure (F) and the equivalent noise bandwidth of the receiver is used for B_n. The power represented in (2.12) is modeled as thermal noise and helps measure the noise level of a radar systems perceived environment.

The RRE is a function of the different elements of a radar's parameters such as transmitted power, bandwidth and target's radar cross section. The general RRE is

Approved for public release; distribution is unlimited.

shown in 2.13:

$$\mathrm{SNR} = \frac{P_\mathrm{r}}{P_\mathrm{n}} = \frac{P_\mathrm{t} G_\mathrm{r} G_\mathrm{t} \lambda^2 \sigma}{(4\pi)^3 R^4 L_S k T_0 B_\mathrm{n} F} \tag{2.13}$$

where P_t is the transmit power, G_t and G_r are the transmit and receive antenna gain, λ is the wavelength, σ is the radar cross section, R is the radial range from the target to the radar, L_s is the system losses, k is the Boltzmann's constant, T_0 is the reference temperature, B_n is the instantaneous receiver bandwidth, and F is the receiver noise figure.

An important observation of the RRE is the fact that range has an inverse relation with power, at a rate of $\frac{1}{R^4}$. This concept is important as it puts a limit on how far a target can be detected. Rewriting (2.13), a new expression in terms of maximum detectable range of a mono-static radar system can be obtained:

$$R_{max} = \sqrt[4]{\frac{P_\mathrm{t} G_\mathrm{r} G_\mathrm{t} \lambda^2 \sigma}{(4\pi)^3 L_\mathrm{s} k T_0 B_\mathrm{n} F \, \mathrm{SNR}_\mathrm{min}}} \tag{2.14}$$

It can be seen that distance from the radar (among other variables) can reduce the SNR; however, there are methods for improving the SNR of received data, which leads to the idea of pulse compression. Before going into pulse compression, a review in radar waveforms is necessary. In radar there are two classes of waveforms: *continuous wave* (CW) and pulsed wave. Continuous waves can be seen as a transmitter continuously transmitting a signal without interruption. Pulsed radar systems, however, operate on the basis of transmitting (M) bursts of repetitive signals at a predefined interval. This means that there is a period of time in which the waveform is being transmitted and then another period of time where the transmitter is off and the receiver is on or *"listening"* as modeled in Figure 2.3.

Approved for public release; distribution is unlimited.

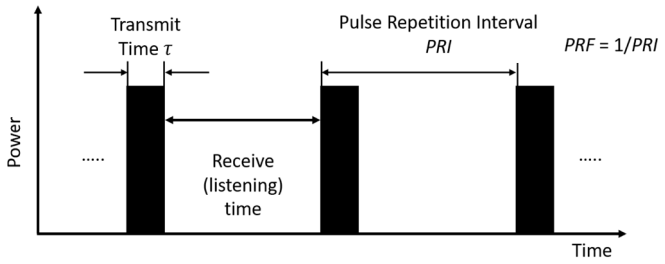


Figure 2.3: Pulsed Waveform Example

In Figure 2.3, we have the following parameters: *Pulse Width* (τ), *Pulse Repetition Interval* (PRI), *Pulse Repetition Frequency* (f_{PRF}), and *Number of Pulses* (N_{p}). For a mono-static pulsed radar system with a transmit/receive switch, there is a minimum detectable range for a target, since there is an on-period where the radar is not *"listening"*, we define this as:

$$R_{\min} = \frac{c\tau}{2} \tag{2.15}$$

For a radar system, the resolution of a range cell is determined by the minimum separation between targets. This principle can also be termed as the Rayleigh criterion, which is the separation between the peak and first null. Using the Rayleigh criterion, it can be stated that the range resolution (δR) for a mono-static pulsed radar system using a simple waveform is the minimum detectable range from (2.15). However, this resolution equation does not include the effects of SNR. When considering the precision of a range estimate, the SNR of the target has an impact and may be written as:

$$\delta R = \frac{c\tau}{2} \tag{2.16}$$

$$\sigma_R = \frac{\delta R}{\sqrt{\mathrm{SNR}}} \tag{2.17}$$

Approved for public release; distribution is unlimited.

In addition to the ability to resolve a target, it is also important to consider the ambiguity related to the range. The unambiguous range or range coverage of a pulsed radar system is a function of the PRI/f_{PRF}:

$$R_{\text{ua}} = \frac{c\text{PRI}}{2} = \frac{c}{2f_{\text{PRF}}}. \tag{2.18}$$

With a pulsed radar system, a series of N_{p} pulses are transmitted from the radar at a given PRI as displayed in Figure 2.3. A common representation for the received set of pulses is a two-dimensional matrix termed as the "fast time/slow time" matrix (range/pulse number), as shown in Figure 2.4.

For the fast time/slow time matrix, the number of pulses sent is represented as "slow time" and the number of samples collected in each pulse is represented as "fast time". One can expect the sampling rate in fast time to be at least equal to the signal *Bandwidth* (B) when upholding the Nyquist sampling theorem, $\frac{f_s}{2} \geq B$. The slow time sampling rate can be understood as the f_{PRF} of the radar. A collection interval, termed as the *Coherent Processing Interval* (CPI), can then be defined as the total number of pulses ($N_{\text{p}} = M$) transmitted multiplied by the PRI:

$$\text{CPI} = M \cdot \text{PRI}. \tag{2.19}$$

The Doppler response can be characterized for a waveform with a point target at an initial range of R_0 through introducing the time varying range $R(t) = R_0 + vt$ in (2.3). Additionally, this time varying range can be characterized in a pulsed system in terms of the m^{th} measured pulse as:

$$R[m] = R_0 + (m-1)v \cdot \text{PRI}, 1 \leq m \leq M \tag{2.20}$$

Approved for public release; distribution is unlimited.

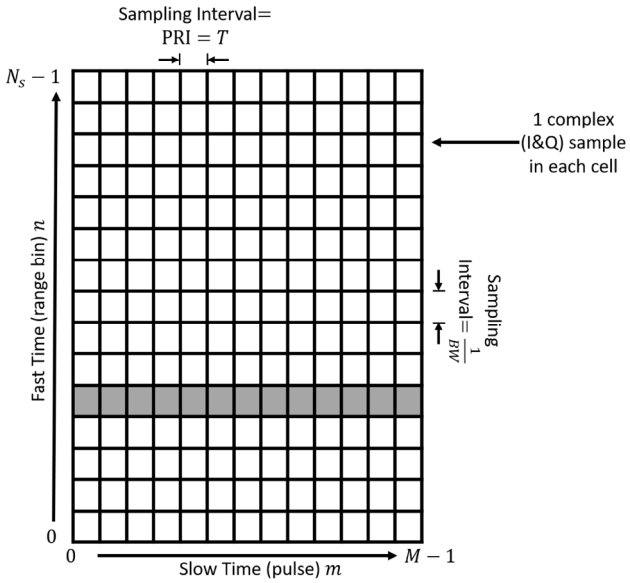


Figure 2.4: Pulse-Doppler Data Matrix

The Doppler or velocity response is of importance through its ability to separate moving targets and stationary clutter . Doppler can be used by radars to differentiate between clutter and moving targets when both the targets and clutter occupy the same range bins using *moving target indication* (MTI) or pulse-Doppler processing approaches. A basic MTI approach simply applies a high pass filter in time domain to the received signal across slow time for a single range bin in order to eliminate stationary clutter. A pulse-Doppler processing approach to this takes advantage of the frequency representation of the received signal across pulses (by taking the *discrete Fourier transform* (DFT)) and can provide separation in Doppler frequency between moving targets and stationary clutter to adequately remove clutter.

MTI approaches are more computationally efficient than pulse-Doppler processing but do not produce as much information as they are simply meant to indicate the

Approved for public release; distribution is unlimited.

presence of a moving target. Pulse-Doppler processing tends to have more computational cost (due to the need to take the DFT across all slow time for each range bin), but deliver more information as well as lead to greater SNR improvement. This processing improvement is due to the fact that the DFT acts as a form of coherent integration, such that the resulting SNR is increased by a factor of M. The remainder of this work will focus on pulse-Doppler processing and its output, defined as the *Range/Doppler* (RD) surface.

Now that an understanding about the CPI, the Rayleigh criterion for range, and the fast-time/slow-time matrix has been gained, one can define the Rayleigh criterion for Doppler resolution as:

$$\delta f_D = \frac{1}{\text{CPI}} = \frac{f_{\text{PRF}}}{M}. \tag{2.21}$$

where M is the number of pulses.

Similarly to the range precision, the precision of a Doppler estimate can be written by considering the influence of SNR as:

$$\sigma_{f_D} = \frac{\delta f_D}{\sqrt{\text{SNR}}}. \tag{2.22}$$

Utilizing these components, the unambiguous Doppler frequency can be derived as $\pm\frac{f_{\text{PRF}}}{2}$, keeping to the Nyquist criterion. One type of waveform that is commonly utilized with the pulsed configuration and is popular for both range and Doppler extraction is a *linear frequency modulated* (LFM) waveform. LFM waveforms are sinusoids whose frequency changes linearly with time and can be modeled as:

$$x_{\text{LFM}}(t) = A\exp(j2\pi f_0 t)\exp(j\pi\frac{B}{\tau}t^2)\ 0 \leq t \leq \tau \tag{2.23}$$

where A is the waveform amplitude, f_0 is the operating frequency, B is the waveform bandwidth, and $\frac{B}{\tau}$ is the chirp rate.

Approved for public release; distribution is unlimited.

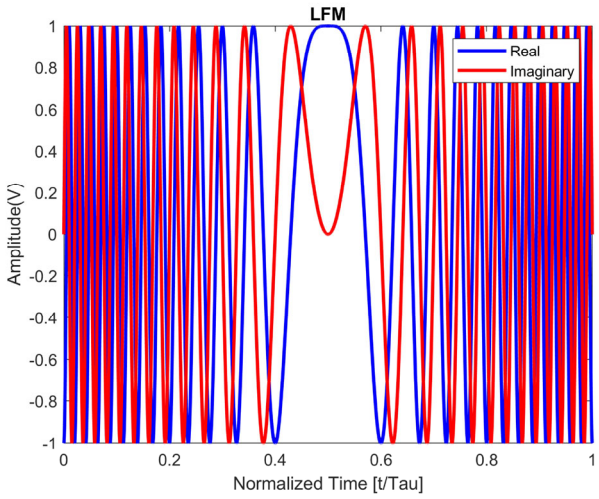


Figure 2.5: Time Domain of LFM Waveform

The time varying phase of an LFM waveform, $\phi(t)$, is quadratic and defined as follows:

$$\phi(t) = \frac{\pi B}{\tau} t^2 \tag{2.24}$$

The instantaneous frequency of the LFM can be defined as the derivative of the phase in (2.24), can be seen in Figure 2.6 as linear and can be defined as:

$$\begin{aligned} \frac{\partial \phi(t)}{\partial t} &= 2\pi \frac{B}{\tau} t \\ f(t) &= \frac{B}{\tau} t \end{aligned} \tag{2.25}$$

With an LFM waveform, the bandwidth (B) of the waveform determines the range resolution of the system. In looking at the resolution of a radar system when transmitting a LFM pulse, we use (2.15) to define the following:

$$\delta R = \frac{c}{2B} \tag{2.26}$$

Approved for public release; distribution is unlimited.

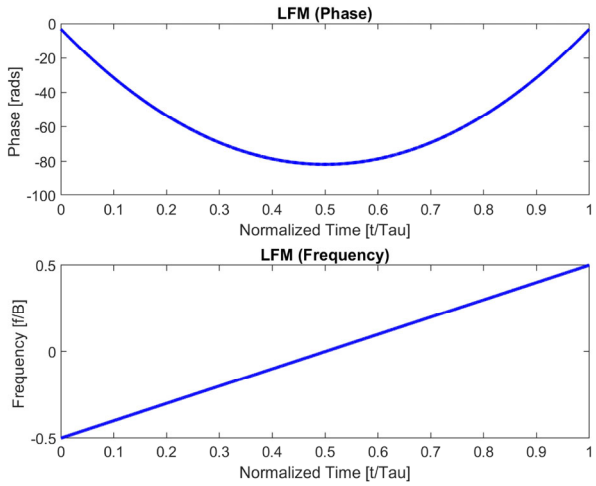


Figure 2.6: Phase vs. Time (top) and Frequency vs. Time (bottom) of the LFM

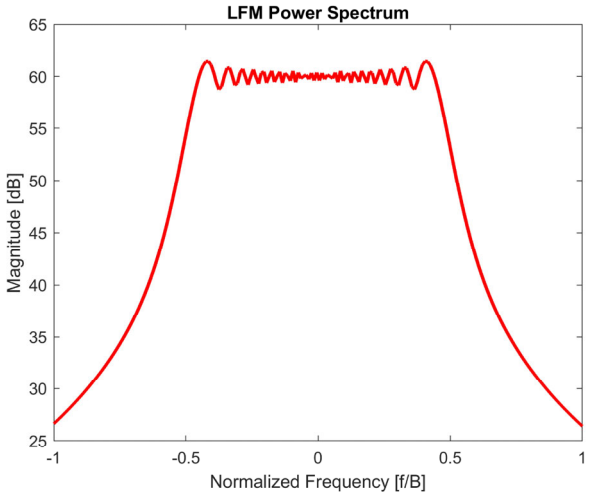


Figure 2.7: Frequency Spectrum of LFM Waveform

Approved for public release; distribution is unlimited.

Returning to the notion of pulse compression, it should be noted that an LFM waveform can be pulse compressed in order to increase the appeared transmit power by increasing the waveform bandwidth through frequency modulation. The matched filtered pulse compressed signal is then able to achieve a new SNR shown in (2.27).

$$SNR = \frac{P_{\mathrm{t}} G_{\mathrm{t}} G_{\mathrm{r}} \lambda^2 \sigma}{(4\pi)^3 R^4 k T B_{\mathrm{n}} F L_{\mathrm{s}}} \tau B \tag{2.27}$$

where τB is the time-bandwidth product.

The matched filter is a convolution of the time-reversed complex conjugated transmit signal with the received signal in the time domain or, equivalently, the cross-correlation of the received signal with the transmit waveform. The matched filter is also recognized as the SNR optimal detector, due to its ability to maximize the signal recognition under matched conditions. The form of the matched filter response is as follows:

$$y(t) = \int x_r(\alpha) h(t-\alpha) d\alpha = \int x_r(\alpha) x_t^*(\alpha - t) d\alpha \tag{2.28}$$

where x_r is the receive signal, h is the matched filter, y is the matched filter response, α is a dummy variable of integration, and x_t^* is the time-reversed conjugated transmit signal.

Once the matched filter is performed and the received signal's SNR is increased, the target range can be discerned. An example of a matched filtered signal is shown in Figure 2.8 in which the transmit signal is matched filtered with itself, resulting in what is known as the auto-correlation.

Due to the simplicity of generating LFM (pulsed) waveforms as well as their ease in processing for various application such as range/Doppler estimation, these waveforms

Approved for public release; distribution is unlimited.

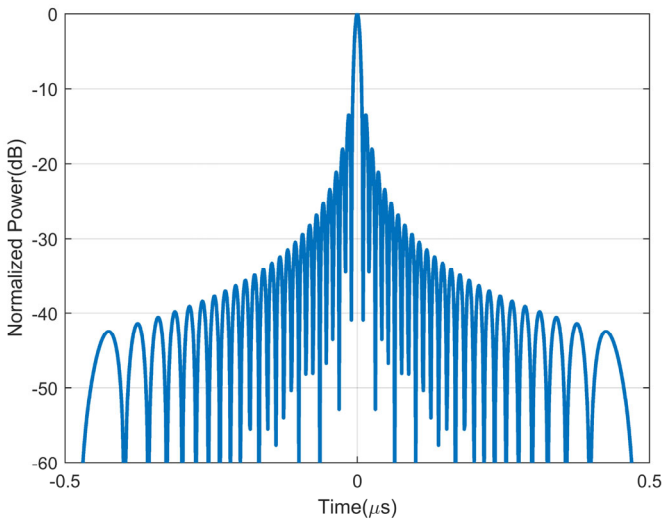


Figure 2.8: Matched Filter Response of LFM Waveform

will be the primary waveforms referenced throughout this dissertation and following *Cognitive Radar Engineering Workspace* (CREW) experimentation.

Range and Doppler detection, target tracking and classification are all tasks that radar systems can be used for. Through use of the reflections captured, detection of a target's range and speed is possible. The time delay and phase change between the reflection and transmission of the EM wave contributes strongly to these detections.

The SNR also plays an important role in establishing the ability for distinguishing a signal from an established noise level. In understanding, the components that contribute to SNR such as bandwidth (B), pulse width (τ), transmit power (P_{t}), *Center Frequency* (F_{c}) and range (R), methods can be used for improving SNR for better detection. Through utilization of pulsed compression and LFM waveforms, detection of targets can be improved.

Approved for public release; distribution is unlimited.

In this work, when performing detection and Doppler processing, the RD surface is used. By performing matched filtering and pulse-Doppler processing, a range-Doppler surface can be obtained where the processes for detecting and tracking targets can be instantiated. An example of a RD surface produced with an LFM waveform and a point target is shown in Figure 2.9.

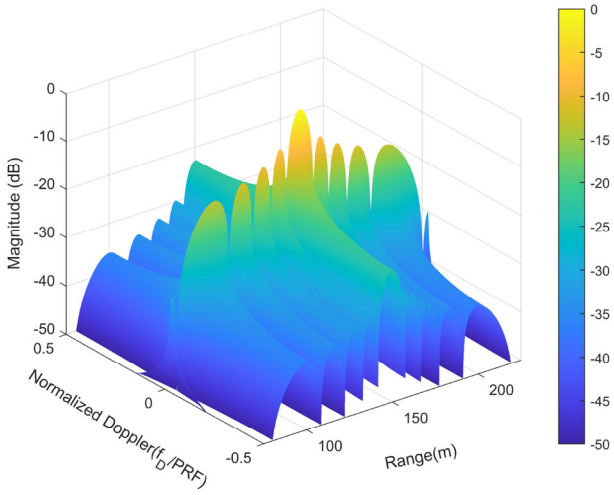


Figure 2.9: LFM Range Doppler Surface

Considering the effect of different parameters such as B, τ, N_{p} and f_{PRF} are important in distinguishing how best to improve the sensing ability of a radar. Recollecting some of the metrics mentioned above, it can be seen that the selection of these parameters can affect the resolution, precision and ambiguity of the radar's environmental perception. Parameters such as B, τ and N_{p} are seen to have effects in the range measurement space on the SNR response, the unambiguous range, the minimum and maximum detectable range, and the range resolution/precision. Similarly, f_{PRF} and

Approved for public release; distribution is unlimited.

N_{p} can impact qualities in the Doppler measurement space such as the Doppler resolution/precision and the unambiguous Doppler. When considering target tracking applications, qualities such as resolution, precision and ambiguity become imperative in performing target estimation.

2.1.2 State Estimation

For the work presented in this research and in different radar engineering applications, there is a need to monitor or *"track"* the states of an evolving system with noisy measurements through a process known as *"target tracking"*. This can be done by modeling the transitions of such a system over discrete time indices. In target tracking, the goal is to estimate the target state at a discrete index k, $\hat{\mathbf{x}}_k$, given a model of the target dynamics ($\boldsymbol{F}$), the measurement model ($\boldsymbol{H}$), and the measurement taken, $\mathbf{z}_k$.

In order to implement a tracking algorithm/process, an understanding of the target dynamics should be modeled which are termed here as the *"motion model"*. Considering a target state, $\mathbf{x} = [r_k \dot{r}_k]$, composed of range $[r_k]$ and velocity (range-rate) $[\dot{r}_k]$ states. The motion model captures the transition of the target states given the previous state, $\mathbf{x}_{k-1}$ and can be written as follows:

$$\mathbf{x}_k = \boldsymbol{F}(\mathbf{x}_{k-1}, \boldsymbol{\nu}_k) \tag{2.29}$$

where $\boldsymbol{\nu}_k$ is the process noise and $\boldsymbol{F}$ is the state transition/motion model.

Similarly, the measurement model can be used to represent the measurements $\mathbf{z}_k$ using the state $\mathbf{x}_k$ as follows:

Approved for public release; distribution is unlimited.

$$\mathbf{z}_k = \boldsymbol{H}(\mathbf{x}_k, \boldsymbol{w}_k) \tag{2.30}$$

where $\boldsymbol{w}_k$ is the measurement noise and $\boldsymbol{H}$ is the measurement model.

Through balancing between the measurements obtained and the assumed target dynamics model, an estimate of the true state can be acquired.

Kalman Filter

For state estimation problems, a conventional and widely used approach is Kalman filtering. The Kalman filter is a recursive state estimation technique that operates based on the assumption that the target dynamics and measurement model are linear functions and the distributions of the errors are Gaussian, in order to obtain the *Minimum Mean-Squared Error* (MMSE) estimate.

Operating under the assumptions that $\boldsymbol{v}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_k)$, $\boldsymbol{w}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_k)$ and $\boldsymbol{F}$ & $\boldsymbol{H}$ are linear functions, where $\mathbf{0}$ represents a vector of zeros, one can rewrite the motion model and measurement model as:

$$\mathbf{x}_k = \boldsymbol{F}_k \mathbf{x}_{k-1} + \boldsymbol{v}_k \tag{2.31}$$

$$\mathbf{z}_k = \boldsymbol{H}_k \mathbf{x}_k + \boldsymbol{w}_k \tag{2.32}$$

Given this, the Kalman filter can be implemented as follows, beginning with initialization, where $\hat{\mathbf{x}}$ represents the state estimates and $\boldsymbol{P}$ represents the error covariance for the state estimates:

Approved for public release; distribution is unlimited.

Initialization

$$\text{Initial state estimate:} \qquad \hat{\mathbf{x}}_0 = \boldsymbol{\mu}_0 \tag{2.33}$$

$$\text{Initial state estimate error covariance:} \qquad \boldsymbol{P}_0 = \boldsymbol{\Sigma}_0 \tag{2.34}$$

where $\boldsymbol{\mu}_0$ is the initial state estimate and $\boldsymbol{\Sigma}_0$ is the initial estimate error covariance based on a priori knowledge of the system model.

Then for $k = 1, ..., K$, the following is computed:

Motion Prediction

$$\text{Predicted state estimate:} \qquad \hat{\mathbf{x}}_k^- = \boldsymbol{F}_k \hat{\mathbf{x}}_{k-1} \tag{2.35}$$

$$\text{Predicted error covariance:} \qquad \boldsymbol{P}_k^- = \boldsymbol{F}_k \boldsymbol{P}_{k-1} \boldsymbol{F}_k^T + \boldsymbol{Q} \tag{2.36}$$

where $\hat{\mathbf{x}}_{k-1}$ is the prior state estimate and $\boldsymbol{P}_{k-1}$ is the prior error covariance.

Measurement Update

$$\text{Measurement Pre-Fit Residual:} \qquad \hat{\boldsymbol{a}}_k = \hat{\boldsymbol{y}}_k - \boldsymbol{H}_k \hat{\mathbf{x}}_k^- \tag{2.37}$$

$$\text{Pre-Fit Residual Covariance:} \qquad \boldsymbol{L}_k = \boldsymbol{R}_k + \boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T \tag{2.38}$$

$$\text{Kalman Gain:} \qquad \boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T (\boldsymbol{L}_k)^{-1} \tag{2.39}$$

$$\text{Updated state estimate:} \qquad \hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \boldsymbol{K}_k \hat{\boldsymbol{a}}_k \tag{2.40}$$

$$\text{Updated estimate error covariance:} \qquad \boldsymbol{P}_k^+ = \boldsymbol{P}_k^- - \boldsymbol{K}_k \boldsymbol{H}_k \boldsymbol{P}_k^- \tag{2.41}$$

The motion prediction process performs a prediction update on the estimate of the state and the error covariance, given the prior. The measurement update process performs an update to the current state estimate through an obtained measurement and balances this measurement against the motion model to obtain a more accurate

Approved for public release; distribution is unlimited.

estimate. Through a recursive loop of the motion prediction and motion update components, the Kalman filter can be used for real-time state estimation through use of the current measurement, the prior state estimate and its uncertainty.

2.1.3 MAP-PF

The detection and association method used in this work implements the *Maximum A Priori Penalty Function* (MAP-PF) approach. The MAP-PF approach jointly performs target state estimation and advanced processing to obtain target measurements. This method approaches target tracking as a target state point estimation problem and its solution is a two-stage estimation process that involves extraction of one measurement for each target from sensor data followed by estimation of target state using Bayesian filtering.

Classical Bayesian tracking consists of three basic elements: 1. generation of contact from sensor data (detection), 2. association of contacts to targets (association), and 3. target state estimation. For the MAP-PF approach, the generation of measurements and state estimations are coupled by the penalty function and performed by the tracker while the association step is eliminated.

In order to use the MAP-PF method some assumptions must be made about the targets and the scene:

- Number of targets, N_{targs}, is known and does not change over scan times $t_1, t_2, \ldots, t_K$,
- The initial position of the target is known, $\mathbf{x}_0$
- Use of an auxiliary process such as the *Likelihood Ratio Detector Tracker* (LRDT) [14],

Approved for public release; distribution is unlimited.

- Definition of a *scan*, as a sensor response at a given time that contains contributions from all targets simultaneously, and
- Multiple target likelihood functions at distinct times are independent given system state.

A likelihood function is defined as the conditional *probability density function* (pdf) of the the observation, $\mathbf{z}_k$, given the target state, $\mathbf{x}_k$, and is written as $f(\mathbf{z}_k|\mathbf{x}_k)$. In the work presented in this dissertation, the conditional pdf of the observation depends on the natural parameters, $\boldsymbol{y}_k$, given the sensor parameters, $\boldsymbol{\theta}_k$, which results in the following form of the likelihood function, $f\left(\mathbf{z}_k|\boldsymbol{y}_k = \boldsymbol{h}(\mathbf{x}_k); \boldsymbol{\theta}_k\right)$, where $\boldsymbol{h}$ is the measurement model mapping states $\mathbf{x}_k$ to the natural/measured parameter space $\boldsymbol{y}_k$. However for the sake of simplicity and with slight notational abuse, the conditional pdf will be rewritten with the assumed parameterization by $\boldsymbol{\theta}_k$ as $f\left(\mathbf{z}_k|\boldsymbol{y}_k = \boldsymbol{h}(\mathbf{x}_k)\right)$

For a joint pdf, we define the following:

$$\mathbf{X} = [\mathbf{x}_1^T \cdots \mathbf{x}_K^T] \tag{2.42}$$

$$\mathbf{Z} = [\mathbf{z}_1^T \cdots \mathbf{z}_K^T] \tag{2.43}$$

$$\mathbf{Y} = [\boldsymbol{y}_1^T \cdots \boldsymbol{y}_K^T] \tag{2.44}$$

to be the state vectors ($\mathbf{X}$), observations ($\mathbf{Z}$) and natural parameters ($\mathbf{Y}$) across time, respectively. Given this, the joint pdf of the target state vectors and observations is of the following form:

Approved for public release; distribution is unlimited.

$$f(\mathbf{Z}, \mathbf{X}) = f(\mathbf{Z}|\mathbf{X})f(\mathbf{X}) \tag{2.45}$$

where

$$f(\mathbf{Z}|\mathbf{X}) = \prod_{k=1}^{K} f(\mathbf{z}_k|\mathbf{x}_k) \tag{2.46}$$

and

$$f(\mathbf{X}) = q_0(\mathbf{x}_0) \prod_{k=1}^{K} q(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{2.47}$$

where $q_0(\mathbf{x}_0)$ is the initial probability distribution and $q(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the transition probability capturing the probability that a target with state $\mathbf{x}_{k-1}$ will transition to state $\mathbf{x}_k$.

In *Maximum A Priori* (MAP) estimation, the goal is to obtain an estimate $\hat{\mathbf{X}}$ based on the observation $\mathbf{Z}$ in order to maximize the posterior density, $f(\mathbf{X}|\mathbf{Z})$ [15]. Using Bayes rule, it follows that the $\hat{\mathbf{X}}$ that maximizes $f(\mathbf{X}|\mathbf{Z})$ also maximizes the joint density $f(\mathbf{Z}, \mathbf{X})$ and its logarithm, $\ln f(\mathbf{Z}, \mathbf{X}) = \ln f(\mathbf{Z}|\mathbf{X}) + \ln f(\mathbf{X})$. Therefore, the optimization problem can be written as:

$$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} \ln f(\mathbf{Z}|\mathbf{X}) + \ln f(\mathbf{X}) \tag{2.48}$$

Following the formulations in [15], the natural parameters $\mathbf{Y}$ that are related to the state variables $\mathbf{X}$ are defined by a known, possibly nonlinear, mapping $\mathbf{Y} = \boldsymbol{h}(\mathbf{X}; \boldsymbol{\theta})$.The optimization in (2.48) can then be reformed as a constrained optimization:

Approved for public release; distribution is unlimited.

$$\hat{\mathbf{X}}, \hat{\mathbf{Y}} = \underset{\mathbf{X},\mathbf{Y}}{\arg\max} \ln f(\mathbf{Z}|\mathbf{Y}) + \ln f(\mathbf{X}) \quad s.t. \quad \mathbf{Y} = \boldsymbol{h}(\mathbf{X}; \boldsymbol{\theta}) \tag{2.49}$$

The constrained optimization problem in (2.49) can be solved by using the penalty function method [16]. For each collection interval, one can define a penalty function $\phi(\boldsymbol{y}_k, \mathbf{x}_k)$ that is equal to zero when $\boldsymbol{y}_k = \boldsymbol{h}(\mathbf{x}_k; \boldsymbol{\theta}_k)$ and becomes more negative as the distance between $\boldsymbol{y}_k$ and $\boldsymbol{h}(\mathbf{x}_k; \boldsymbol{\theta}_k)$ increases. For this work, the quadratic penalty function is used, given by:

$$\boldsymbol{\phi}(\boldsymbol{y}_k, \mathrm{x}_k) = -(\boldsymbol{y}_k - \boldsymbol{h}(\mathbf{x}_k; \boldsymbol{\theta}))^T \boldsymbol{\Sigma}_k (\boldsymbol{y}_k - \boldsymbol{h}(\mathbf{x}_k; \boldsymbol{\theta})) \tag{2.50}$$

where $\boldsymbol{\Sigma}_k$ is a matrix chosen to weigh the components of the constraint vector in a desirable manner. This matrix can be chosen to be the *Predicted Information Matrix* (PIM) after the transformation to the measurement/natural parameter space, $[\boldsymbol{H}\boldsymbol{P}_{\boldsymbol{k}}^{-}\boldsymbol{H}^T]^{-1}$, and will be further discussed in Chapter 7. One can then form a composite penalty function by summing over all collection intervals:

$$\boldsymbol{\Phi}(\mathbf{Y}, \mathbf{X}) = \sum_{k=1}^{K} \boldsymbol{\phi}(\boldsymbol{y}_k, \mathbf{x}_k) \tag{2.51}$$

Replacing the constraint in (2.49) with the penalty function, the MAP-PF optimization problem then becomes:

$$\hat{\mathbf{X}}, \hat{\mathbf{Y}} = \underset{\mathbf{X},\mathbf{Y}}{\arg\max} \ln f(\mathbf{Z}|\mathbf{Y}) + \boldsymbol{\Phi}(\mathbf{Y}, \mathbf{X}) + \ln f(\mathbf{X}) \tag{2.52}$$

The solution to this optimization problem is the same $\hat{\mathbf{X}}$ that is the solution to (2.48) and (2.49). The advantage of this approach is that it provides one the ability

Approved for public release; distribution is unlimited.

to break up the problem into separate, tractable problems and apply the alternating maximization method in which we alternately maximize over $\mathbf{Y}$ and $\mathbf{X}$ [15].

Alternating Maximization: Maximize over Natural Parameters

First, the optimization over $\mathbf{Y}$ for a fixed $\mathbf{X}$ is performed. In this step, only the first and second terms of (2.52) are involved in the maximization as the prior term does not depend on $\mathbf{Y}$, giving:

$$\hat{\mathbf{Y}} = \arg\max_{\mathbf{Y}} \ln f(\mathbf{Z}|\mathbf{Y}) + \mathbf{\Phi}(\mathbf{Y}, \mathbf{X})$$

$$\hat{\mathbf{Y}} = \arg\max_{\mathbf{Y}} \sum_{k=1}^{K} \ln f(\mathbf{z}_k|\boldsymbol{y}_k) + \boldsymbol{\phi}(\boldsymbol{y}_k, \mathbf{x}_k) \tag{2.53}$$

The terms decouple with respect to k, so the natural radar parameters can be estimated independently for each dwell,

$$\hat{\boldsymbol{y}}_k = \arg\max_{\boldsymbol{y}_k} \ln f(\mathbf{z}_k|\boldsymbol{y}_k) + \boldsymbol{\phi}(\boldsymbol{y}_k, \mathbf{x}_k). \tag{2.54}$$

The optimization problem in (2.54) represents a penalized maximum likelihood problem; however it can also be interpreted as a MAP estimation problem with the penalty function appearing where a prior term of the form $\ln f_0(\boldsymbol{y}_k, \mathbf{x}_k)$ would appear. Holding this view, the penalty function acts as an analytically simplified substitute for the prior term.

Traditionally, the natural radar parameter estimation would use a standard radar adaptive signal processing detection/estimation technique requiring a data association step to associate the detections to targets. In MAP-PF, the penalty function method automatically creates associated detections and restricts the estimates to be in the vicinity of the tracker predictions.

Approved for public release; distribution is unlimited.

The optimization problem in (2.54) is still a difficult multi-dimensional optimization problem because of an attempt to estimate the natural radar parameters for multiple targets simultaneously. However, one can employ the alternating maximization method, to sequentially maximize over the parameters for a single target while holding the parameters of the other targets fixed. To do this, we assume the penalty function can be written as a sum of the target penalty functions,

$$\mathbf{\Phi}(\mathbf{Y}, \mathbf{X}) = \sum_{m=1}^{M} \boldsymbol{\phi}_m(\boldsymbol{y}_{m,k}, \mathbf{x}_k), \tag{2.55}$$

where $\boldsymbol{y}_{m,k}$ is the m^{th} target's parameter vector

$$\boldsymbol{y}_{m,k} = \left[\boldsymbol{\eta}_{m,k}^T \ \gamma_{m,k}\right]^T \tag{2.56}$$

and $\boldsymbol{\phi}_m(\boldsymbol{y}_{m,k}, \mathbf{x}_k)$ is its associated penalty function. The single target optimization problem becomes:

$$\hat{\boldsymbol{y}}_{m,k} = \arg\max_{\boldsymbol{y}_k} \ln f(\mathbf{z}_k|\boldsymbol{y}_k) + \boldsymbol{\phi}(\boldsymbol{y}_{m,k}, \mathbf{x}_{m,k}) \tag{2.57}$$

For simplification, the RD surface is used to approximate the log-likelihood function where $10\log_{10}(\boldsymbol{RD}_{\mathbf{z}_k}) \approx \ln f(\mathbf{z}_k|\boldsymbol{y}_k; \boldsymbol{\theta}_k)$, in which case, the target natural parameter vector estimate is found from:

$$\hat{\boldsymbol{y}}_{m,k} = \arg\max_{\boldsymbol{y}_k} 10\log_{10}(\boldsymbol{RD}_{\mathbf{z}_k}) + \boldsymbol{\phi}(\boldsymbol{y}_{m,k}, \mathbf{x}_{m,k}) \tag{2.58}$$

Alternating Maximization: Target State Estimation

Once the natural parameter estimates, $\hat{\mathbf{Y}}$, have been obtained one can then fix $\mathbf{Y}$ and maximize over $\hat{\mathbf{X}}$ in (2.52). Due to their dependence on $\mathbf{X}$, only the second and third terms in (2.52) are used in this maximization, giving:

Approved for public release; distribution is unlimited.

$$\begin{aligned}\hat{\mathbf{X}} &= \arg\max_{\mathbf{X}} \mathbf{\Phi}(\mathbf{Y}, \mathbf{X}) + \ln f(\mathbf{X}) \\ &= \arg\max_{\mathbf{X}} \sum_{k=1}^{K} \{\boldsymbol{\phi}(\boldsymbol{y}_k, \mathbf{x}_k) + \ln q(\mathbf{x}_k|\mathbf{x}_{k-1})\} + \ln q_0(\mathbf{x}_0) \end{aligned} \tag{2.59}$$

Again, due to the fact that the terms decouple with respect to k in the second equation, we can estimate the natural radar parameters independently for each dwell,

$$\hat{\mathbf{x}}_k = \arg\max_{\mathbf{x}_k} \boldsymbol{\phi}(\boldsymbol{y}_k, \mathbf{x}_k) + \ln q(\mathbf{x}_k|\mathbf{x}_{k-1}) + \ln q_0(\mathbf{x}_0) \tag{2.60}$$

In this optimization problem, one can take advantage of the fact that the known mapping, $\boldsymbol{h}(\mathbf{X}; \boldsymbol{\theta})$ is linear; the distributions are Gaussian; and the system is linear and follows a Gauss-Markov motion model, to proceed with using the linear Kalman filter to obtain a solution to the optimization problem in 2.59.

2.2 Neural Network Background

In order to further understand the importance of combining the works of neural networks and cognitive radar, a discussion of neural networks is presented.

2.2.1 Neural Network Basics

The neural network theory provided draws on content from neural network tutorials and journal articles on the basics of an *Artificial Neural Network* (ANN) [17, 18, 19, 20]. Neural networks are a subcategory of machine learning inspired by the biological systems in the human brain. Neural networks, also referred to as artificial neural networks (ANN), aim to model the biological neurons in the brain through artificial neurons using mathematical functions. Figure 2.10 displays the labeling of the different parts of the neural network.

Approved for public release; distribution is unlimited.

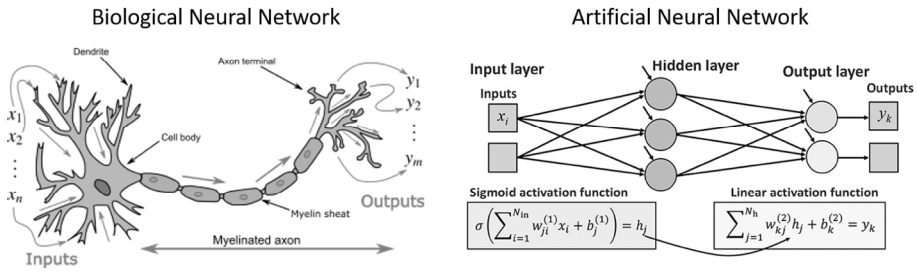


Figure 2.10: Biological Neuron vs. Artificial Neuron

[Images Licensed under CC BY and CC BY-SA [21, 22]]

A neuron is a special biological cell that processes information by receiving signals (impulses) from other neurons through its receivers (dendrites) and transmits signals generated by its cell body along the transmitter (axon). At the end of a dendrite is a synapse which is an important unit between neurons that has neurotransmitters released upon the arrival of a signal. The neurotransmitter spreads across the synaptic gap to enhance or prevent the receptor neuron's ability to emit electrical signals, which causes adjustments to the synapse to be made from which it can learn from the activities it participates in.

In simpler terms, a neuron is passed information in impulses from other neurons and the method in which the information is sent across neurons can be adapted such that the neuron learns. Figure 2.10 shows a side by side comparison of the biological neuron to the artificial neuron and their similar elements.

Artificial neurons are modeled in a similar way, with the neuron being equivalent to a mathematical weighted sum of the inputs set by an activation function, which can be on of many functions such as a *rectified linear unit* (ReLU). The rectified

60
Approved for public release; distribution is unlimited.

linear unit is a ramp function taking the max between its inputs and 0 resulting in only positive units for the output. The mathematical models for a neuron and the activation function are shown in (2.61) and (2.62), respectively.

$$y = \phi(\boldsymbol{w} \cdot \boldsymbol{x}) = \phi(\sum_{j=1}^{n} w_j x_j) \tag{2.61}$$

where w_j is the weight, x_j is the input, ϕ is the activation function, and y is the output.

Considering the prior notation for the inputs ($\boldsymbol{x}$) and weights ($\boldsymbol{w}$), the activation function for the ReLU can be defined as follows:

$$\phi = \max(0, \sum_{j=1}^{n} w_j x_j) \tag{2.62}$$

where ϕ is a ReLU activation function.

Following the same idea as biological neurons, the output of a neuron is then fed to other neurons with the activation function acting as the synapse's firing of the neurotransmitter. The weights shown in (2.62) are an important component of the neural network as they set the importance of the inputs. By applying a weight value to the inputs, a sense of connection between multiple neurons are established and propagated through the network. The establishment and development of these weights can be taken from concepts in Hebbian learning, which states that cells that fire together, wire together [23]. This concept means that if one neuron causes another neuron to fire their correlation should be strengthened resulting in an increase in weights for both, inversely so if the correlation between neurons are weak, the weights will be lowered. From this, it can be seen that weights establish correlation between neurons and allow for learning to take place.

Approved for public release; distribution is unlimited.

The aim of a neural network is to learn key features regarding their various inputs in order to encourage recognition in each layer. A layer can be viewed as a defined collection of neurons. Layers are important for feature extraction as each layer can be seen as extracting different features such as edges or image contrast in the case of image recognition. Layers vary depending on the type of neural network implemented, however the basic structure of a network can be broken into three layers: input, hidden, and output. The input and output layer are familiar concepts as they have been reviewed in prior paragraphs and in Figure 2.10, however, the hidden layers are new concepts.

A hidden layer can be perceived as the feature extraction layer which performs all the background tasks necessary for recognition and analysis. Referencing the example of image recognition, the eyes of humans can be seen as the input layer (collection of inputs) and the output layer can be seen as the recognition of a face or image. The hidden layer is the processing necessary to achieve the output from the input through extracting aforementioned features such as the boundaries of the image and the contrast [24]. These basic building blocks of neural networks are important and standard in most networks. An example of a three layer neural network is shown in Figure 2.11 where x_i-input neurons, w_{ij}-weights, a_i-activation function at neuron and y_i-output.

Neural networks can be split into two main architectures: feed-forward (multi-layer perceptron) networks and recurrent (feedback) networks. Feed-forward networks propagate their information forward through multiple layers and produce one set of output values. Also, the response of the network to an input is independent of the

Approved for public release; distribution is unlimited.

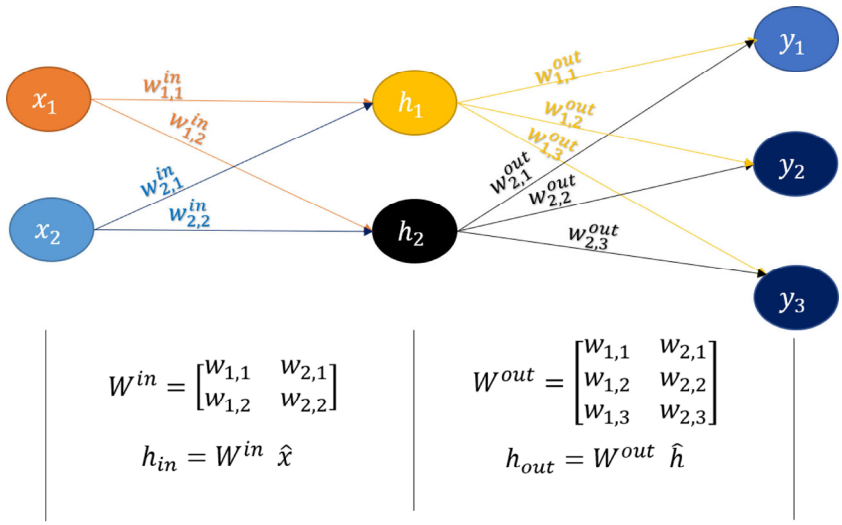


Figure 2.11: Three Layer Neural Network

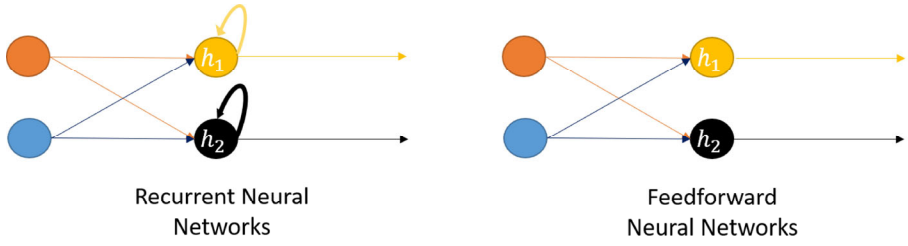


Figure 2.12: Recurrent/Feedback Networks and Feed-forward Networks

previous network state. Recurrent networks, however, depend on their previous network state for their responses. Recurrent networks generate an output from an input pattern and modify the inputs to the neuron in order to get a new network state. Figure 2.12 shows a graphical representation of feed forward and recurrent networks.

In addition to being composed of layers, activation functions and inputs, another important part of neural networks is the training employed.

Approved for public release; distribution is unlimited.

2.2.2 Different Approaches to Learning in Neural Networks

Just as there is variety to the architecture of a neural network, there are also different methods of learning. These methods are briefly summarized in the following sections, with further details on supervised learning being provided as it is the main type of learning used in this work.

Supervised Learning

One of the most utilized methods of learning in neural networks is supervised learning. Supervised learning is the use of a *"truth"* or *"teaching signal"* to adjust a machine learning model or system [25]. In supervised learning approaches, the teaching signal can either be a label or parameter value such that for some training dataset there are pairs of inputs, x,and corresponding desired targets or truth outputs, y^*, in which the goal is find a mapping function $f(x) \rightarrow y$ which maps the input x to an output $y \approx y^*$. The training performed for a supervised learning approach focuses on developing a system or model that learns the relationship between the inputs and targets leading to a data-driven learning process. Figure 2.13 displays an example of supervised learning with a *Mean-Squared Error* (MSE) function.

Approved for public release; distribution is unlimited.

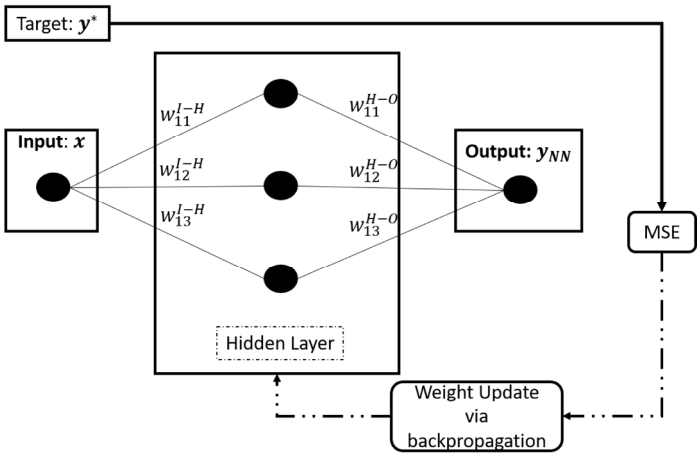


Figure 2.13: Supervised Learning Diagram

Unsupervised Learning

While supervised learning requires that there be a teaching signal or target output, unsupervised learning does not and shifts away from using labeled data. In unsupervised learning, the system or model changes its function using assumed patterns in the data according to rules/decision functions implicit in its design[25]. The assumed patterns are found by the model or system depending on the details of its design. Some implementations of unsupervised learning refer to using clustering algorithms or probability density approximations which capture the general assumed pattern of the data and match the outputs using extracted similarities.

Reinforcement Learning

Reinforcement learning is separate from supervised and unsupervised learning through capturing a realistic and experience based model of low-level learning in humans and animals [25]. Reinforcement learning is similar to supervised learning in the sense that learning is performed through trying to obtain a defined goal. A

Approved for public release; distribution is unlimited.

trial and error approach to learning is performed through the definition of reward functions and then exposing the network to an environment, in which it continuously performs actions that lead to a greater reward (performing autonomous learning). Reinforcement learning is best represented by the diagram shown in Figure 2.14.

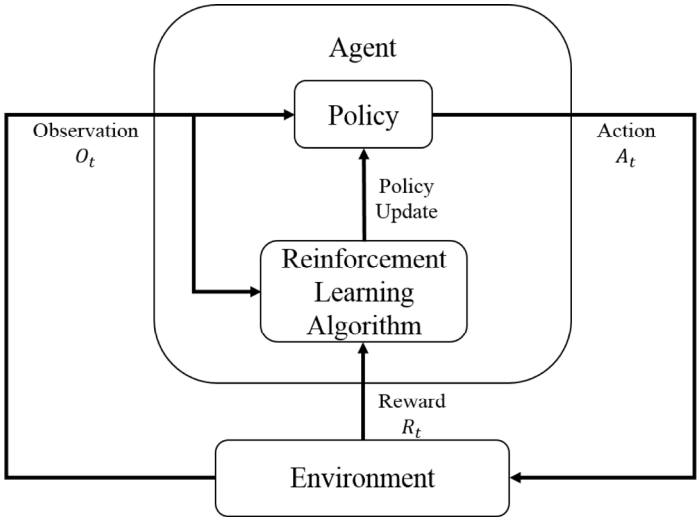


Figure 2.14: Reinforcement Learning Diagram

Reinforcement learning approaches focus on training an agent to select actions (A_t) that maximize the reward (R_t) based on an observation (O_t) from the environment. An agent is composed of two main components: a policy and a learning algorithm [26]. The policy is a way to map an observation of the environment (O_t) to an action (A_t) and the learning algorithm trains the policy based on the action (A_t), observation (O_t) and reward (R_t) to create an optimal policy that maximizes the reward received.

Approved for public release; distribution is unlimited.

2.2.3 Supervised Learning Approaches

In supervised learning, most learning problems associated with neural networks deal with the minimization of error. The back-propagation algorithm is the most commonly utilized for its ability to minimize error as well as ease of implementation. Back-propagation is a training method used for adapting the weights of a neural network using a defined error function. The weights being adapted for are those referenced in (2.61), which are used to weigh the input of the neuron.

The back-propagation algorithm involves taking the gradient of the error function at each iteration step, in order to correct the randomly initialized weights set in the network. The process of back-propagation can be defined through mathematical models. An error function, E, can be defined as any differentiable error function such as the MSE function, that takes in both the truth and neural network prediction as inputs. If E is held as an arbitrary, differentiable error function such as the the mean squared error, o_i is the i^{th} input of a neuron, δ_j is the error at the j^{th} neuron, Δ is the gradient operator, γ is the learning constant which defines the step length of the error correction and w_{ij} is the weight, the following can be written:

$$\frac{\partial E}{\partial w_{ij}} = o_i \frac{\partial E}{\partial o_i w_{ij}} = o_i \delta_j \tag{2.63}$$

$$\Delta w_{ij} = -\gamma o_i \delta_j \tag{2.64}$$

The equations in (2.63) and (2.64) can be seen as the back-propagated error and the weight gradient increment that is added to each weight w_{ij}, respectively. The weight gradient, Δw_{ij}, added to the weight, w_{ij}, is the weight update which is

Approved for public release; distribution is unlimited.

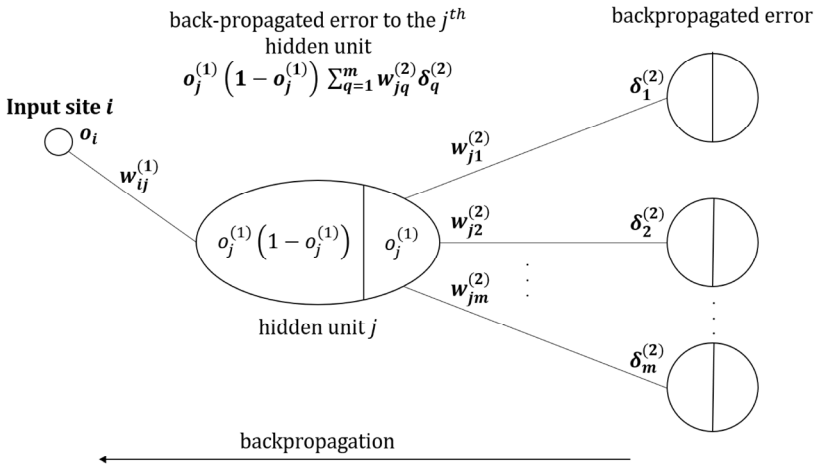


Figure 2.15: Back-propagation Example

correcting the weights according to the defined error function, E. In Figure 2.15, a graphical example of this process, without the weight update, is shown.

Given a brief review of the back-propagation process, it is imperative to point out the process in which supervised neural networks learn. The supervised learning process of neural networks can be likened to the human learning process.

Neural networks have three phases of learning: training, validation and testing. Correspondingly, the input data is separated into three portions: training data, validation data and testing data, similar to the spread of the phases.

The training phase is where the network does most of its learning or weight adjustment by presenting the training data set repeatedly to the network in a learning cycle called an *epoch* either as a whole or in separate amounts at a time called *batches* until the weight values are learned [27]. The training period can be likened to the period in which humans are introduced to new things for the first time or in childhood when children are learning shapes and colors.

Approved for public release; distribution is unlimited.

The validation phase is used to check the performance of the network in order to determine the point at which training should stop to avoid over-fitting [27]. Over-fitting is the point at which a network begins memorizing instead of learning which prevents it from being able to generalize its predictions when presented with unseen data. Over-fitting occurs when the error of the validation phase increases while the error from the training phase decreases. The optimum result is when both the training error and validation error converge to a minimal value. The validation phase can be compared to pop quizzes or homework on content learned in order to review concepts.

Lastly, the testing phase is the final phase at which the ultimate evaluation of the networks performance is done, in which it is presented with unseen data, and the output of the data is solely generated by the learned input-output mapping [27]. The testing phase is like a midterm or cumulative exam in which all ideas taught are tested on new problems with similar concepts.

2.2.4 Long Short Term Memory Recurrent Neural Network

The *Long Short Term Memory* (LSTM) neural network is a variant of *recurrent neural network* (RNN) meant to solve the vanishing gradient problem (error contributions leading to the incapability of learning) with the short term portion realized through a forget layer that extracts necessary information. This recurrent neural network is a form of dynamic network that has the capability to adapt its response based off current and predicted values. An unrolled LSTM is shown in Figure 2.16, where h_t is a hidden unit at time step t, c_t is the cell state at time step t, G is the LSTM layer, and x_t is the input at time step t.

Approved for public release; distribution is unlimited.

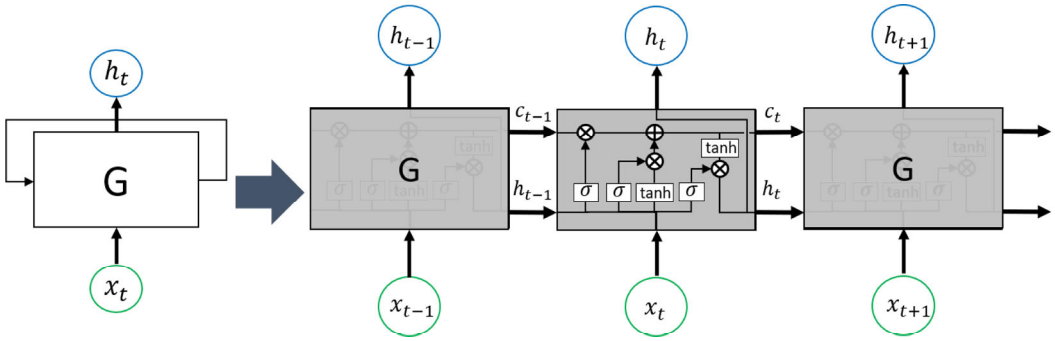


Figure 2.16: Graphic Diagram of Unrolled LSTM in Time with hidden unit h and input x

Using the LSTM-RNN unrolled in time, we describe its process as follows. The LSTM block uses the initial state (c_0, h_0) of the network and the first time-step of the input sequence to compute the first output and the updated cell state. At time step, t, the LSTM block uses the current state of the network (c_{t-1}, h_{t-1}) and the next time step of the input sequence (x_t) to calculate the output (c_t, h_t). The hidden state at the time step t, h_t, contains the output of the LSTM layer. The cell state at the time step t, c_t, contains information learned from the previous time steps. At each time step, the LSTM layer adds to or removes information from the cell state.

Approved for public release; distribution is unlimited.

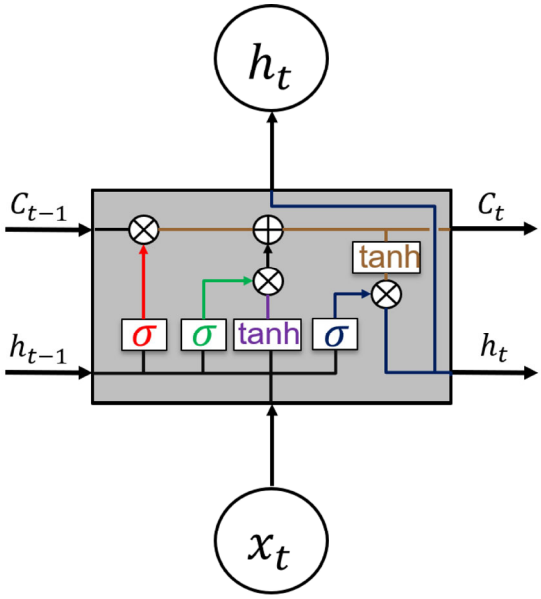


Figure 2.17: Single LSTM Block in time with highlighted components for each composite layer within the LSTM: Forget Gate (red), Input Gate (green), Tanh layer (purple), Updated Cell State (brown), and filtered output (dark blue)

Defining the specific components of the LSTM shown in Figure 2.17, we have the following:

$$\textbf{Forget Gate Layer:} \quad f_t = \boldsymbol{\sigma}\left(\boldsymbol{W}_f x_t + \boldsymbol{R}_f h_{t-1} + b_f\right) \tag{2.65}$$

in which the forget gate layer (components in red) decides the information to dispose of from cell state.

$$\textbf{Input Gate Layer:} \quad i_t = \boldsymbol{\sigma}\left(\boldsymbol{W}_i x_t + \boldsymbol{R}_i h_{t-1} + b_i\right) \tag{2.66}$$

in which the input gate layer (components in green) decides what values to update.

$$\textbf{Tanh Layer:} \quad \tilde{c}_t = \textbf{tanh}\left(\boldsymbol{W}_{\tilde{c}} x_t + \boldsymbol{R}_{\tilde{c}} h_{t-1} + b_{\tilde{c}}\right) \tag{2.67}$$

Approved for public release; distribution is unlimited.

in which the hyperbolic tan (**tanh**) layer (components in purple) choose candidate values for cell state, $\tilde{c}_t$.

$$\textbf{Update:} \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.68}$$

in which c_t, is the updated cell state (components in brown) and $\odot$ is the Hadamard product (element-wise product).

$$\textbf{Filter Output:} \quad (o_t, h_t) \tag{2.69}$$

in which the filtered output (components in dark blue) is one in which the inputs fed through the sigmoid ($\boldsymbol{\sigma}$) decide which parts of the cell state to output, $o_t = \boldsymbol{\sigma}_{\tilde{c}}(\boldsymbol{W}_o x_t + \boldsymbol{R}_o h_{t-1} + b_o)$; the cell state, c_t, is put through the **tanh** layer; and then multiplied by the output sigmoid gate so the hidden state retained is composed of the relevant portions of the cell state giving $h_t = o_t \odot \textbf{tanh}(c_t)$. In the components mentioned above, $\boldsymbol{R}$ represents the recurrent weights, $\boldsymbol{W}$ represents the input weights and b represents the bias which are all learnable weights within the LSTM.

The dynamic/changing portion of the LSTM-RNN is the updated cell state. All weights (input weights-$\boldsymbol{W}$, recurrent wights-$\boldsymbol{R}$ and bias b) stay fixed once trained, however the updated cell state of the LSTM allows the Hadamard product between the output value and the **tanh** activated updated cell state to change the inner state of the network. As the LSTM-RNN was implemented using MATLAB, there are two functions which can be used to produce the output of a trained LSTM-RNN. These two functions are: `predict()` and `predictandUpdateState()`. The `predict()` function uses the trained network to predict a sequence of responses for a given input sequence using the trained network and its current state (state remains fixed), whereas

Approved for public release; distribution is unlimited.

the `predictandUpdateState()` function predicts the response for the input data and updates the network state at each time step for a dynamic network implementation.

2.3 FAR and HFAR

In this section, a brief overview of the FAR and HFAR frameworks implemented in the FARMS and *Hierarchical Fully Adaptive Radar Modeling and Simulation* (HFARMS) environments are presented as these frameworks are used and improved upon in sections of the work discussed in this dissertation. Within the FARMS framework there exists two distinct processors: the perceptual and the executive. Interaction between them forms a PAC in the FARMS framework. The framework takes on a hierarchical form when multiple PACs are stacked in tiers within the same system.

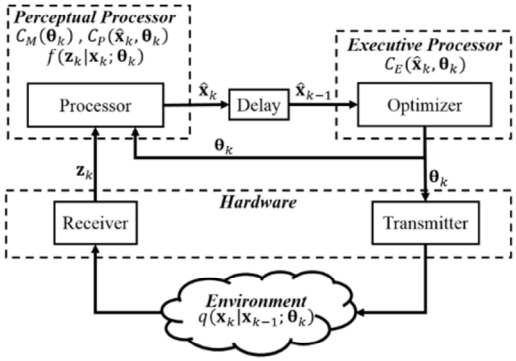


Figure 2.18: Fully Adaptive Radar Framework

Bell, Mitchell and Smith's work in [28] gives detailed insight on the FAR and HFAR frameworks. The receiver and transmitter blocks in the hardware layer are only responsible for the transmission and collection of data with specific parameters.

Approved for public release; distribution is unlimited.

The perceptual and executive layers however form the major cognitive nature of the framework.

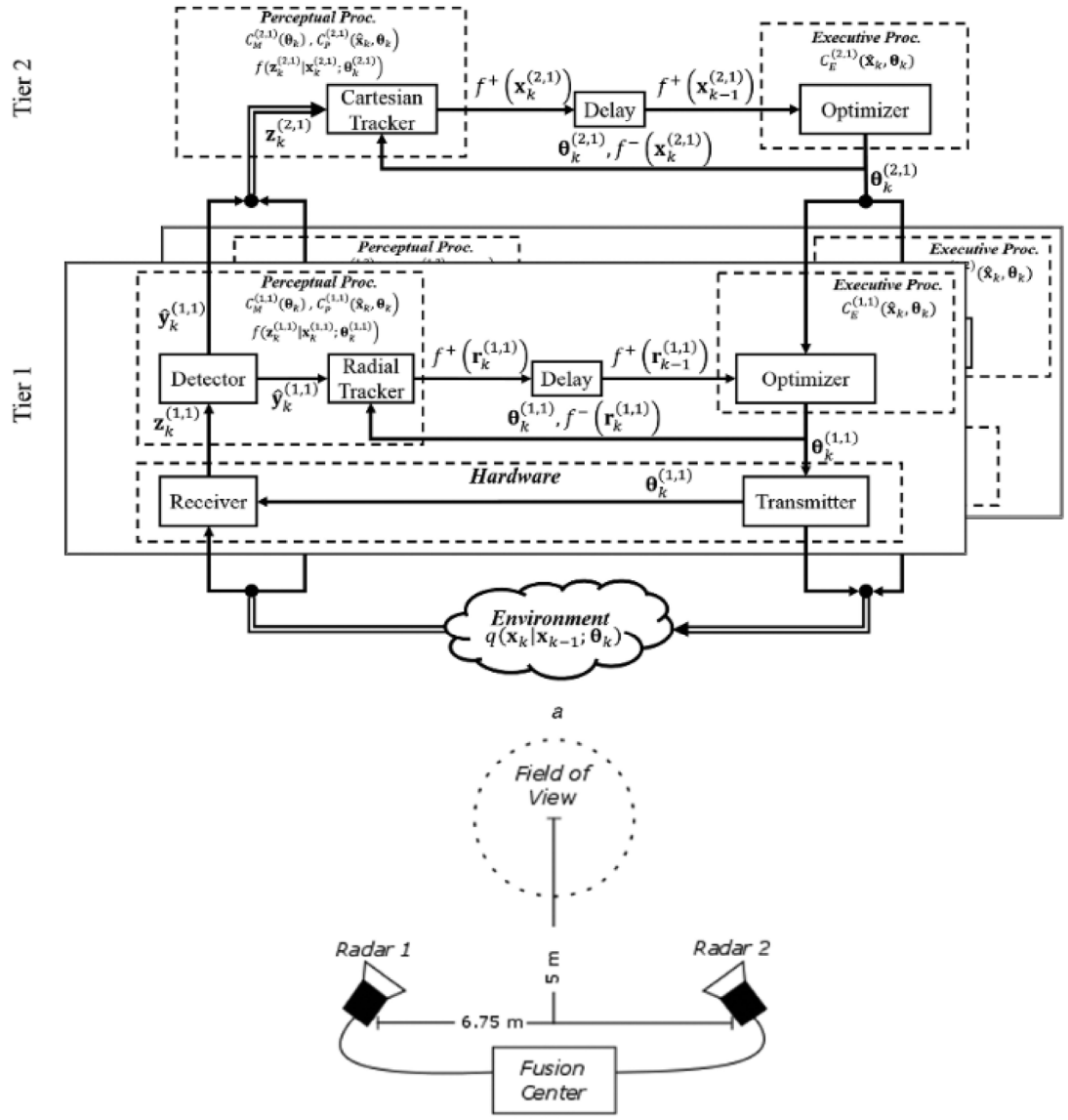


Figure 2.19: Hierarchical Fully Adaptive Radar Framework

Approved for public release; distribution is unlimited.

For non-hierarcical FAR with a single PAC, the perceptual processor is defined by its measurement likelihood function $f(\mathbf{z}_k|\mathbf{x}_k;\boldsymbol{\theta}_k)$, measurement cost function $C_M(\boldsymbol{\theta}_k)$, and performance $C_P(\boldsymbol{\theta}_k;\hat{\mathbf{x}}_k)$ cost function. The executive processor is defined by the executive optimization cost function $C_E(\boldsymbol{\theta}_k;\hat{\mathbf{x}}_k)$. HFAR on the other hand, requires several PACs which can be arranged by Q hierarchical layers and A parallel layers. The equations defining the perceptual and executive processors change to accomodate these configurations. The perceptual processors are defined by their measurement likelihood functions $f(\mathbf{z}_k^{(Q,A)}|\mathbf{x}_k^{(Q,A)};\boldsymbol{\theta}_k^{(Q,A)})$, measurement $C_M^{(Q,A)}(\boldsymbol{\theta}_k)$ cost functions, and performance $C_P^{(Q,A)}(\boldsymbol{\theta}_k;\hat{\mathbf{x}}_k)$ cost functions. The executive processors are defined by the executive optimization cost functions $C_E(\boldsymbol{\theta}_k;\hat{\mathbf{x}}_k)$. Figures outlining the existing FAR and HFAR framework are shown in Figure 2.18 and 2.19, respectively. Figure 2.20 is a combination of Figures 2.18 & 2.19 using descriptive convention for simplicity instead of the equations described above.

Approved for public release; distribution is unlimited.

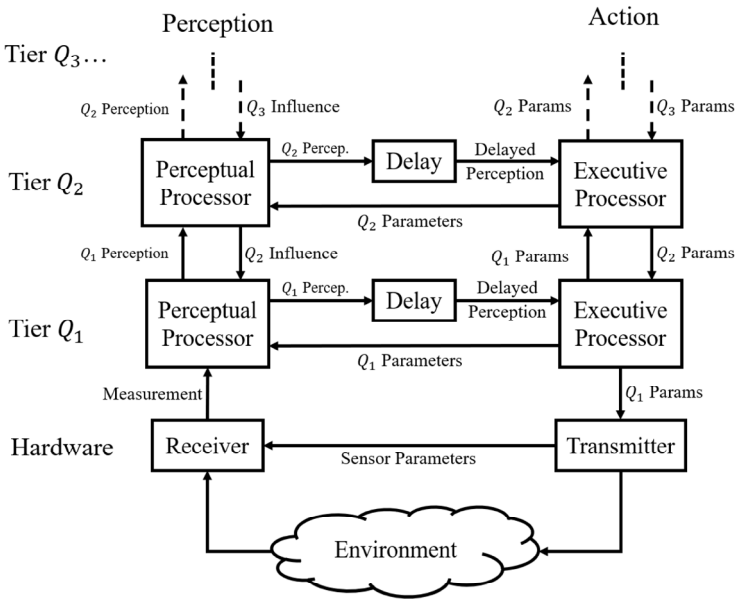


Figure 2.20: HFAR Descriptive Framework

2.4 Optimization and Statistics Background

2.4.1 FAR and HFAR Optimization

In cognitive radar work, much of the action selection boils down to solving an optimization problem. In order to solve these optimization problems, many different approaches can be taken such as *sequential quadratic programming* (sqp) [29], simplex [30], interior point [31], gradient projection [32], penalty function [33], branch and bound [34], or heuristic [35] methods to name a few. However, this dissertation will focus on reviewing the sequential quadratic programming approach to nonlinear constrained optimization problems as well as the penalty method as these are implemented for the discussed works in Chapters 5 and 7.

The sqp algorithm is an iterative process in which a *quadratic programming* (QP) sub-problem is formed from the general problem (2.70) and minimized.

Approved for public release; distribution is unlimited.

The general problem can be stated as:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{2.70}$$

$$s.t. \begin{cases} G_i(\boldsymbol{x}) = 0, \ i = 1, ..., n_e \\ G_i(\boldsymbol{x}) \leq 0, \ i = n_e + 1, ..., n \end{cases}$$

with the Lagrangian function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{n_e} \lambda_i G_i(\boldsymbol{x}) + + \sum_{i=n_e+1}^{n} \lambda_i G_i(\boldsymbol{x}). \tag{2.71}$$

where $\mathcal{L}$ is the Lagrangian, $f(\boldsymbol{x})$ is the objective function, $G(\boldsymbol{x})$ are the constraints, n_e is the number of equality constraints, λ are the Lagrange multipliers, and n is the total number of constraints.

Given the general problem and the Lagrangian, the sqp approach solves a QP sub-problem. This QP sub-problem is solved at each iteration k to determine the search direction, $\boldsymbol{q}_k$, and find the next trial solution $\boldsymbol{x}_{k+1}$ [36]. The sub-problem can be written as follows:

$$\min_{\boldsymbol{q}} \frac{1}{2} \boldsymbol{q}^T \boldsymbol{H} \boldsymbol{q}^T + \nabla f(\boldsymbol{x}_k)^T \boldsymbol{q} + f(\boldsymbol{x}_k) \tag{2.72}$$

$$s.t. \begin{cases} \nabla G_i(\boldsymbol{x}_k)^T \boldsymbol{q} + G_i(\boldsymbol{x}_k) = 0, \ i = 1, ..., n_e \\ \nabla G_i(\boldsymbol{x}_k)^T \boldsymbol{q} + G_i(\boldsymbol{x}_k) \leq 0, \ i = n_e + 1, ..., n \end{cases}$$

where ∇ represents the derivative operator and $\boldsymbol{H}$ is the Hessian matrix.

The Hessian matrix, $\boldsymbol{H}$, is a positive-definite approximation that can be calculated at each iteration k using the *Broydon-Fletcher-Goldfarbo-Shanno* (BFGS) method:

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{\boldsymbol{b}_k \boldsymbol{b}_k^T}{\boldsymbol{b}_k^T \boldsymbol{a}_k} - \frac{\boldsymbol{H}_k \boldsymbol{a}_k \boldsymbol{a}_k^T \boldsymbol{H}_k^T}{\boldsymbol{a}_k^T \boldsymbol{H}_k \boldsymbol{a}_k} \tag{2.73}$$

where $\boldsymbol{a}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{b}_k = \nabla \mathcal{L}(\boldsymbol{x}_{k+1}) - \nabla \mathcal{L}(\boldsymbol{x}_k)$.

Approved for public release; distribution is unlimited.

The next trial solution $\boldsymbol{x}_{k+1}$ can then be written as follows:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{\alpha}_k \boldsymbol{q}_k \tag{2.74}$$

where $\boldsymbol{\alpha}_k$ determines the step length parameter.

The setting of $\boldsymbol{\alpha}_k$ and the resulting new trial solution $\boldsymbol{x}_{k+1}$ obtained rely on the use of a line search method to minimize the penalty function, $\Phi(\boldsymbol{x})$. The penalty function is used here to evaluate whether one point is better than another such that there is a sufficient decrease in the penalty function.

The penalty function that is evaluated in MATLAB implementations of the sqp method can be written as follows [37]:

$$\Phi(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{n_e} r_i G_i(\boldsymbol{x}) + \sum_{i=n_e+1}^{n} r_i \max[0, G_i(\boldsymbol{x})] \tag{2.75}$$

where r_i is the penalty parameter and $\max[0, G_i(\boldsymbol{x})]$ is the max operator used to enforce penalties only for positive values of $G_i(\boldsymbol{x})$.

While the penalty function form is mentioned above for use in the sqp method, a more general overview is stated here. The penalty function method is an unconstrained approach to solving a constrained optimization problem through which the constraints are added to the objective function and penalized according to the constraint evaluation [38]. Beginning with the general problem shown in (2.70), one can form the penalty function based unconstrained approximation:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + P(\boldsymbol{x}). \tag{2.76}$$

where $P(\boldsymbol{x})$ is a continuous function that maps to the penalty space, which is a measure of constraint compliance.

Approved for public release; distribution is unlimited.

One possible implementation for $P(\boldsymbol{x})$ is:

$$P(x) = \sum_{i=1}^{n_e} c_i \left|G_i(\boldsymbol{x})\right|^2 + \sum_{i=n_e+1}^{n} c_i \left|G_j(\boldsymbol{x})\right|^2 \tag{2.77}$$

where $c_i > 0$.

In this implementation, it can be seen that deviations from the constraint goal leads to a penalty equal to the square of that violation for equality and inequality constraints.

2.4.2 Hypothesis Testing and T-tests

Statistical tools such as the Welch's t-test are used in Chapter 5, and a brief summary is provided here.

Welch's t-test, also known as the 2 sample t-test, is a 2 sample hypothesis test that can test whether two populations have equal means [39]. Welch's t-test can be used to compare two mutually exclusive groups and is more reliable than the Student's t-test when the samples have unequal variances and/or unequal sample sizes.

Welch's t-test defines the test statistic t as:

$$t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_x^2}{n} + \frac{s_y^2}{m}}} \tag{2.78}$$

where $\bar{x}$ and $\bar{y}$ are the sample means, s_x and s_y are the sample standard deviations, and n and m are the sample sizes.

When the two data samples, x and y, are from populations with unequal variances, then the test statistic under the null hypothesis has an approximate Student's

Approved for public release; distribution is unlimited.

t-distribution with a number of degrees of freedom provided by Satterthwaite's approximation, ν.

$$\nu = \frac{(\frac{s_x^2}{n} + \frac{s_y^2}{m})^2}{\frac{s_x^4}{n^2\nu_x} + \frac{s_y^4}{m^2\nu_y}} \tag{2.79}$$

where $\nu_x = n - 1$, the degrees of freedom associated with the first variance estimate of x; $\nu_y = m - 1$, the degrees of freedom associated with the second variance estimate of y.

For t-test's, an important parameter is the significance level, α, which represents the percentage of risk in concluding that a difference between the population means exist when in fact there is no actual difference (false-alarm). In similar importance as α, another parameter known as the p-value supplies additional understanding of the statistics.

The p-value, p , is a measure of statistical significance of the observed hypothesis. In combination with the significance level, the p-value indicates whether we should reject our null hypothesis or fail to reject our null hypothesis. For the work to be discussed in Chapter 5, the null hypothesis used is that the two population means are equal, also known as the two-tailed test. Using the p-value and the significance level, one can conclude one of the following: when $p \leq \alpha$ one should reject the null hypothesis as the difference between the means is statistically significant; or when $p > \alpha$ one should fail to reject the null hypothesis as the difference between the means is not statistically significant.

2.5 Summary

This section is meant to give a brief overview on pertinent radar, neural network, optimization and statistic theory for better interpretation of the results discussed in

Approved for public release; distribution is unlimited.

the chapters to follow. Through the radar theory, track acquisition & track estimation discussed, analysis of the cognitive radar systems presented in Chapter 5, 6, 7 and 8 can take place. Additionally, the statistical tools discussed in this section are used to provide a better performance comparison between the neural network and solver implementations discussed in Chapter 5. Complementary to this background section, a discussion of cognitive science and relevant work within the area of cognitive radar & neural networks is also necessary and will be presented in the next section.

Approved for public release; distribution is unlimited.

Chapter 3: Literature Review

3.1 Cognitive Science

Cognitive science is an interdisciplinary field that focuses on the study of the mind/brain or, phrased in another way, the idea of cognition [40]. The interdisciplinary nature of cognitive science combines the different disciplines of artificial intelligence, education, linguistics, neuroscience, philosophy, psychology and anthropology. These disciplines are used to begin to understand cognitive processes such as perception, memory, language, critical thinking, understanding, learning and action [40]. Two fields in the interdisciplinary makeup of cognitive science stand out: neuroscience and artificial intelligence (subfield of computer science). Neuroscience studies the various levels and functions of the nervous system, whereas artificial intelligence is the study of the computations that make it possible to perceive, reason and act [40, 41].

Neuroscientist Joaqin Fuster is well-known for his fundamental contributions to the understanding of the neural structures underlying cognition and behavior. In [42], Fuster discusses concepts of cognition in which he mentions the plausibility of a connectionist model of knowledge organization in the cerebral cortex. The connectionist network model is an artificial intelligence concept for cognition in which knowledge

Approved for public release; distribution is unlimited.

is established by a distributed interactive network made up of multiple connections between nodes (equivalent to brain cells) [43]. In the field of artificial intelligence, the neural network is seen to capture basic concepts of memory in the weights such that even though the weights change there are still memory traces, due to the distributive nature of the connectionist model, allowing for memory retrieval [44]. Fuster defines a similar concept to neural networks that he refers to as cognits, which are interactive, overlapping networks that represent units of memory/knowledge. The cognits are assumed to be arranged in a hierarchical structure, separated by a perception-action divide bringing about the concept of the PAC. This PAC is aimed at capturing the main elements of cognition defined by Fuster in [42], which are perception, memory, attention, language and intelligence.

3.2 Cognitive/Fully Adaptive Radar

Building off ideas from neuroscience and cognitive science research, Haykin developed an idea known as cognitive radar. In [6], Haykin mentions three ingredients considered to be the basic portions of cognitive radar: intelligent signal processing (learning through environment interaction), feedback from the receiver to the transmitter (facilitating intelligence through perception) and preservation of information content of radar returns (memory-usually realized by the Bayesian approach to target detection through tracking). Inspired by the physical realization of these three ingredients in the echo-location systems of bats, Haykin proposes a cognitive radar architecture. Haykin distinguishes a cognitive radar from an adaptive radar, through mentioning three important differences: a cognitive radar continuously learns and updates its receiver through knowledge gained about the environment from experience,

Approved for public release; distribution is unlimited.

the transmitter of a cognitive radar adjusts its illumination of the environment intelligently conditioned by its task, and the whole radar system constitutes a dynamic closed feedback loop encompassing the transmitter, environment and receiver.

In addition to the idea of learning from experience proposed by Haykin's idea of cognitive radar, Guerci promoted the idea of using a knowledge database for a knowledge-aided approach of cognitive radar where memory is from an exogenous source and does not need to be learned [45]. The idea of such knowledge-aided approaches is beneficial as the question becomes less about how to learn (acquire memory from experience) and more about how to adequately use what you have learned (apply memory).

Brüggenwirth et al., also presents a cognitive radar framework that uses a skill-based, rule-based and knowledge-based layer to enable cognitive functionality [46]. The skill-based layer represents the basic signal-generation and processing capabilities of the radar system; the rule-based layer represents the radar systems actions to the perceived scene; and the knowledge-based layer represents the planning and prediction component of the cognitive radar system using a-priori knowledge.

In line with some form of Haykin, Guerci and/or Brüggenwirth's proposals of a cognitive radar framework, different takes on cognitive radar have been developed. Two popular views of cognitive radar are its applications to spectral agility and target perception. Due to the crowded EM spectrum as well as intentional adversarial interference, spectral agility techniques have been researched by many [47, 48, 49, 50, 51, 52, 53, 54, 55].These different approaches take many forms such as frequency shifting in order to avoid operating on the same frequency [51, 52]; calculation of "safe regions" for co-existence of radar & communication systems and

Approved for public release; distribution is unlimited.

scheduled suppression of communication signals (interpreted as interference) when necessary [49]; and bandwidth change either through spectral notching or band shifting [47, 50, 53, 54]. One of the dominant views of this type of cognitive radar work is shown by [47] and [54] who focus on generating specialized waveforms using an optimization scheme for notched waveform design to avoid in-band interference. The central theme of the cognition derived from these works takes the form of forming a spectral perception of the environment and then using the captured perception to take action through adapting future transmit waveforms and in some cases tracking the dynamic interference to better adapt future waveforms.

Besides adapting to the spectral environment, another view of cognitive radar is adapting to the target. Adapting the radar system to the target takes the form of attempting to achieve a higher order aim such as improved detection, localization, tracking performance and classification [56, 57, 58]. These objectives are achieved by adapting the radar waveform and system parameters using feedback from the output of the radar back-end. The work of Bell and Smith is one example of this adaptation and takes advantage of the PAC defined by Fuster to form the FAR framework [4]. The focus of the FAR framework is on adapting the radar waveform parameters according to a set of costs defined in the processor blocks conditioned on the receiver's perception, see Figure 2.18.

Bell, Mitchell, and Smith extended this concept to a hierarchical view of cognitive radar founded in the ideas presented by Fuster to allow for multiple PAC's, the capability of sensor fusion, and the development of a fully adaptive hierarchical muti-task sensor [28, 59], see Figure 2.19.

Approved for public release; distribution is unlimited.

3.3 Neural Networks for Radar Applications

Similar to the notion of applying intelligence to radar research, neural networks have been a frequently used tool in radar applications. Most radar based neural network applications, however, have been classification based work. Travoss shows the use of neural networks for ground penetrating radar by using an ANN to classify areas potentially containing object reflections [60]. Briones applies neural networks to target detection in which ANNs are used for automatically detecting and classifying different targets by feeding one-dimensional representations of two-dimensional images of different targets and classifying the target types [61].

The largest use of classification based neural networks in radar research is in *Synthetic Aperture Radar* (SAR) imagery. Del Frate applies ANNs to detect oil spills by feeding a set of features providing information about an oil spill candidate collected from a SAR image and estimating the probability for the candidate to be a real oil spill [62]. The most recent work in neural networks applied to SAR imagery classification has been in target recognition using state-of-the-art *Convolutional Neural Network* (CNN). CNNs take advantage of their convolutional feature to get local correlations and are inspired by the connectivity pattern between neurons seen in the organization of the visual cortex of animals [63]. Xie applies CNNs to target recognition in SAR images and demonstrates its ability to classify different targets within the *Moving and Stationary Target Acquisition and Recognition* (MSTAR) dataset [64].

While most radar applications for neural networks are for classification, there have been regression based neural networks used for radar applications. Parameter estimation is one form of the regression based neural networks used in radar work, through which the neural network is used to estimate parameters to be used either by a signal

Approved for public release; distribution is unlimited.

processing algorithm, high order task or information extractor. Xiao uses an ANN to perform rainfall estimation from radar measurements through feeding the network reflectivity, differential reflectivity and distance information (height/elevation of radar) to estimate the amount of rainfall [65]. Timofey applies an ANN to *maximum likelihood estimation* (MLE), in which the ANN is used to perform a fast computation of the bearing that shows the direction to the source of the EM wave received by a passive radar system with an antenna array [66]. Noone uses a recurrent backpropagation neural network to perform parameter estimation of the pulse gaps (PRI) for a radar pulse train in an electronic warfare environment after deinterleaving the multiple pulse trains received and predicting the next pulse event by tracking the pulse train [67].

Aside from parameter estimation, neural networks have also been used for image enhancement tasks. Gao applies CNN's to radar imaging by feeding the radar echo signal to a complex valued CNN, termed CV-CNN, and generating an image to replace computationally intensive iterative algorithms [68].

3.4 Combining Neural Networks and Cognitive Radar

Research done in both neural networks and cognitive radar independently have resulted in great progress in both areas from image recognition tasks to improved environment sensing. However, little research has been done in the combination of both areas. Recently, some researchers have been looking at how to combine the benefits of a cognitive radar outlook with the reduced complexity of neural network approaches. Shaghaghi, in particular, presents a new approach to cognitive radar task scheduling using an approximate algorithm based on the Monte Carlo tree search

Approved for public release; distribution is unlimited.

method, which takes advantage of bound and dominance rules to eliminate nodes from the search tree as well as a policy neural network to reduce the width of the search [69].

Implementations of neural networks are also feasible for the work discussed by Brüggenwirth et al., in which the author mentions the use of convolutional neural networks for classification tasks at the skill-based layer [46]. Additionally, in viewing the framework discussed by Brüggenwirth et al., other implementations of neural networks can be seen such as: at the rule-based layer in order to assist with tracking tasks & radar resource management through the use of function approximation networks like radial basis neural networks or policy networks; and at the knowledge-based layer in order to assist with planning and optimization tasks using the inherent knowledge incorporated in neural networks.

Elbir at al., presents a method for cognitive radar antenna selection using CNNs to perform classification (selecting optimal subarrays) based on covariance samples of received array signal. The goal of the author's work is to find an optimal antenna array for the next scan in which only K antennas from an M-element antenna array will be used in order to perform *Direction of Arrival* (DoA) estimation of targets [70].

Aside from these works though, very few researchers have looked at combining neural networks with cognitive radar concepts past those focused on classification uses. In viewing the components of cognitive radar implementations and the benefits of neural networks, there is an apparent need to incorporate memory based learning and a reduction of computational cost for real-time cognitive radar systems. Neural networks have the capability to reduce computational costs (through offline learning implementations) as well as incorporate memory-based learning (through dynamic

Approved for public release; distribution is unlimited.

neural networks, online learning, and reinforcement learning). Thus, this dissertation will focus on discussing how neural networks can be incorporated into a cognitive radar architecture to demonstrate memory-based learning; reduce computational costs to allow the realization of a *"truly"* cognitive radar;and enable autonomous behavior shifting the role of the radar operator to a advisory one. This dissertation will also look at furthering the development of cognitive radar concepts through extending the current FAR single target framework to a multiple target environment and applying it to multiple task functionality.

Approved for public release; distribution is unlimited.

Chapter 4: FARMS Algorithm and CREW Experimental Testbed

4.1 FARMS-Fully Adaptive Radar Modeling and Simulation

The Fully Adaptive Radar Modeling and Simulation (FARMS) environment is the software set used to implement simulation and experimentation of cognitive radar concepts for this dissertation work. The FARMS environment contains the flexibility to switch between two components: a simulation environment that could be used to compare against the Cognitive Sensing Lab (CSL) environment at the Ohio State University ElectroScience Laboratory and real time experimentation using an experimental test-bed called the Cognitive Radar Engineering Workspace (CREW). Both simulation and experiment were used in the work presented in this dissertation; therefore, a brief overview of the software architecture in addition to the CREW test bed is provided in this chapter.

FARMS Architecture

The FARMS codebase is an object-oriented architecture that provides an interface for the various components (optimizer, sensor, processor, etc.) that are defined by base classes in order to implement the cognitive PAC structure. The general

Approved for public release; distribution is unlimited.

form of the architecture contains eight objects defined as follows: *Scene* [Simulation Only], *Sensor*, *Processor*, *Optimizer*, *StorageManager*, *DisplayManager*, *TimingManager*, and *PerceptionActionCycle*. Specific implementations of these objects work in conjunction to implement the cognitive radar concepts used in this work.

The *Scene* object is defined and used only in simulation as it defines the target, clutter and noise characteristics of the environment observed. The environment model for the scene assumes *Additive White Gaussian Noise* (AWGN) with stationary clutter approximated with a Gaussian distribution. An arbitrary number of targets and scatterers can be present within the scene. Each scatterer is defined by its Cartesian position as a function of time, RCS, and the bi-static reflection angle. The *Scene* object returns the position and RCS of the targets over time, using interpolation when necessary.

The *Scene* object is capable of assuming different distributions for the target's RCS. The common model assumed for the target is the Gaussian/Normal distribution, in which the target is modeled as a collection of three single point scatters with the RCS fluctuating from scan-to-scan. Another RCS modeling of the target is through the use of Swerling models, in which a single point target is used with any of the four Swerling cases. The Swerling I and II cases model the target as one consisting of a collection of equally large isotropic reflectors which are distributed on a surface (as shown in Figure 4.1); whereas the Swerling III and IV cases model the target as one consisting of a dominant isotropic reflector superimposed by many small reflectors (as shown in Figure 4.2). The Swerling models are based on the chi-square distribution with specified degrees of freedom.

Approved for public release; distribution is unlimited.

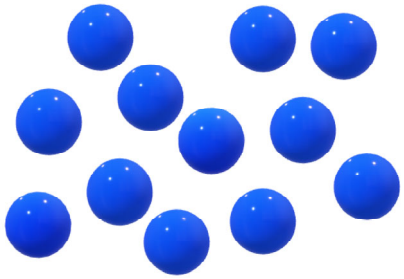


Figure 4.1: Swerling Model I and II

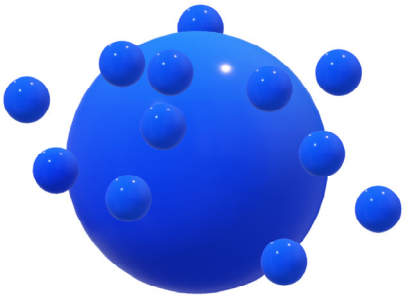


Figure 4.2: Swerling Model III and IV

Swerling I and II use the chi-square distribution, with two degrees of freedom defined as:

$$p(\sigma) = \frac{1}{\bar{\sigma}} \exp(\frac{-\sigma}{\bar{\sigma}}) \tag{4.1}$$

where $\bar{\sigma}$ is the average of all values of RCS for the reflector.

Swerling III and IV use the chi-square distribution with four degrees of freedom defined as:

$$p(\sigma) = \frac{4\sigma}{\bar{\sigma}^2} \exp(\frac{-2\sigma}{\bar{\sigma}}) \tag{4.2}$$

Depending on the Swerling model chosen, the target fluctuation will vary from scan-to-scan or from pulse-to-pulse. For Swerling I and III the target's RCS is assumed

Approved for public release; distribution is unlimited.

constant pulse-to-pulse but varies from scan-to-scan, conversely Swerling II and IV assume a pulse-to-pulse variation rather than a scan-to-scan. However, for the work presented in this dissertation, the target is assumed to be modeled using a Gaussian distribution and varies from scan-to-scan.

The *Sensor* object controls the capture and transmission of signals through obtaining raw data from simulation, pre-recorded data or experimentation. In addition to the *Sensor* object, there is a connected component, called the *Hardware* object which is the interface to the physical AWG's and digitizer for the CREW. This *Hardware* object is responsible for collecting the radar measurements in experimentation and artificially generating returns based in target/scatter information received from the *Scene* object when in simulation mode.

The *Processor* object converts the raw data capture from the sensor into a refined perception though data processing steps such as matched filtering, pulse-Doppler processing, detection, and Bayesian filtering. One of the Processor object's main goals is to model how the environment evolves over time and balance this with the system's measurements.

The *Optimizer* object implements the action selection based on the environment perception. This action selection is done by solving an executive optimization in order to obtain the next set of sensor parameters, $\boldsymbol{\theta}$, with which to observe the environment. For our implementation, we focus on balancing the cost of using measurement resources against the need to meet user-defined performance goals. In addition to this any form of optimization approaches/algorithms can be implemented in this component for action selection. Due to the MATLAB environment used, there are a wide range of solvers available when using the MATLAB Optimization toolbox.

Approved for public release; distribution is unlimited.

The *StorageManager* object stores and maintains a history of variables of interest from the *Scene*, *Sensor*, *Processor* and *Optimizer* in order to introduce a component of long term memory into the fully adaptive radar framework.

The *DisplayManager* object is used to provide visualization of the system performance through displaying values and figures of interest each cycle while the algorithm runs and providing a final summary display of important metrics.

The *TimingManager* object records the processing times required by each step of the perception-action cycle. In addition to this, it also keeps track of when each measurement is taken and can coordinate timing between multiple perception-action cycles.

The *PerceptionActionCycle* object is meant to be a realization of a single PAC used to coordinate the perception forming and coordinated action selection defined for a FAR system through a sequential calling of methods within each of the above objects. The properties within the *PerceptionActionCycle* object also act as sort of short term memory by passing necessary data between the other objects.

4.2 CREW Testbed

The CREW consists of four independent transmit chains and four receive chains, whose signal flows are illustrated in Figure 4.3. The CREW system is made up of the following: four transmit heads, four receive heads, two arbitrary waveform generators, a Balun box, a base-band converter, four digitizers that can sample at 3 GSa/s, two local oscillators, and the control PC.

For the transmit chain, there are independent arbitrary waveform generators, or AWGs which generate differential signals at the first IF, or IF1 as 0.3-1.3 GHz.

Approved for public release; distribution is unlimited.

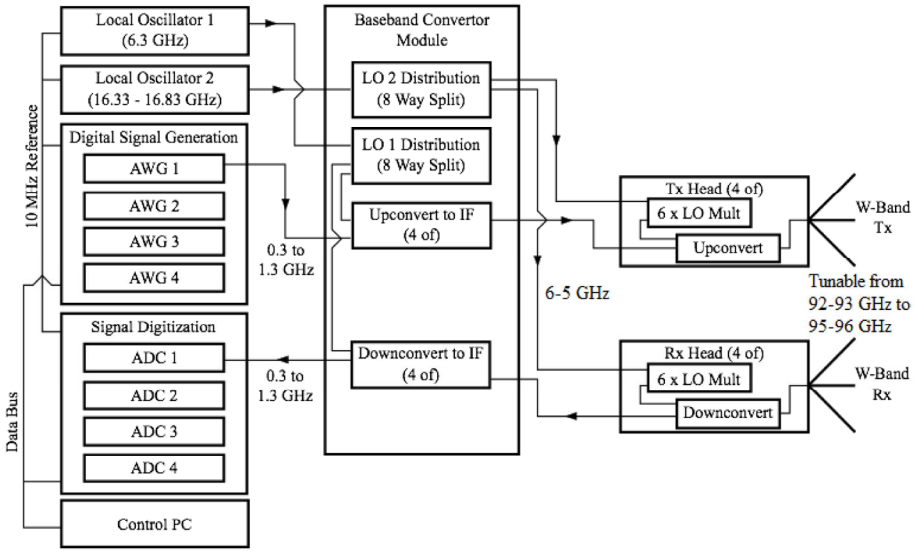


Figure 4.3: CREW RF Component Diagram

Following this are two Keysight digitizers, a master and slave, which hold channels 1&3 and 2&4, respectively. The four channel outputs are then fed into a base-band converter module, which uses a high side conversion architecture with LO1 to convert the signals to IF2 at 5-6 GHz. IF2 exits the base-band converter along with a ∼16 GHz LO signal from LO2, to the four transmit heads which multiply the LO ×6 and mix with the IF signal to W-band (92-96 GHz) for transmission. The LO frequency can be changed to provide a 4 GHz operational range of transmitted frequency, with a center frequency between 92.5-95.5 GHz.

For the receive chain, the received signals are down-converted inside the four receive heads (using separate LO distribution and mixing) and the resulting 5-6 GHz IF2 signal is then shifted to base-band in the base-band converter module's receive channels. The base-band signals are then digitized using four ADCs residing on a

Approved for public release; distribution is unlimited.

single Keysight digitizer, which is controlled via the host PC allowing for real time signal analysis and adaptation.

The CREW is a unique test-bed allowing research into fully adaptive & cognitive algorithms as well as distributed/multi-static operation all within the laboratory. The extremely high bandwidths result in a range resolution of a few centimeters making it possible to observe multiple scattering centers in smaller targets and easy to range-gate out the walls of the laboratory. The W-band center frequency allows the narrow-band assumption to be made even if the full 1 GHz instantaneous bandwidth is used. As such, it is easy to set-up sophisticated experiments using the CREW within the CSL. The system can be programmed using MATLAB which allows for rapid prototyping of fully adaptive and cognitive algorithms.

4.3 FARMS Data Flow

Given the summary provided for the FARMS algorithm and the CREW test-bed, we now look at the diagram in Figure 4.4 displaying the data-flow of a single PAC. Through this diagram, it can be seen that through use of the *FAR_Engine* we begin to instantiate the FAR process. To begin, data is either captured from the environment in experimentation or artificially generated from the *Scene* object using the *Hardware* object. Using an event/listener to indicate the completion of data capture by the *Hardware*, the raw data can be sent in order to begin forming the perception, using the *Sensor* and *Processor*. Again, through the use of an event/listener as a trigger to indicate when the perception is formed, pertinent data is extracted and sent to the *Optimizer* for action selection. Once an action is selected and indicated using the event/listener, the *DisplayManager* and *StorageManager* are used to store

Approved for public release; distribution is unlimited.

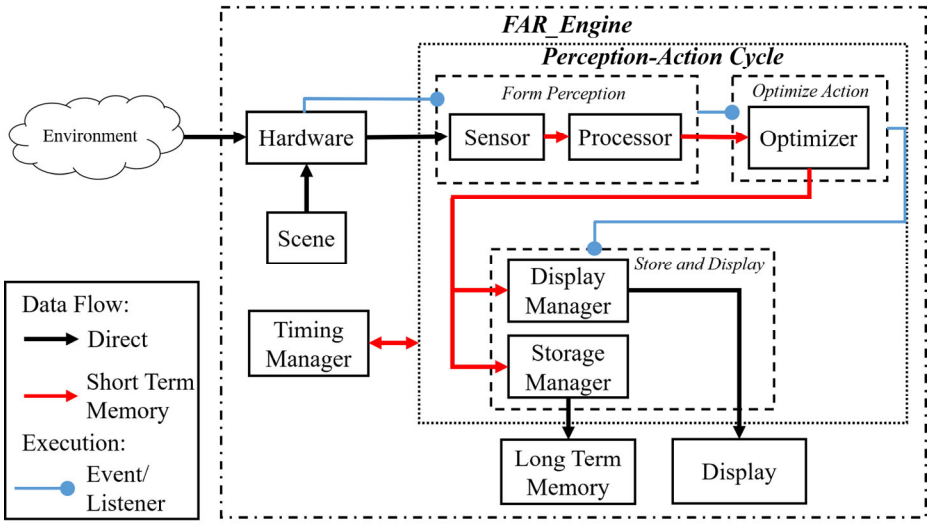


Figure 4.4: Data Flow Diagram for FARMS Single PAC

and display relevant information. The *PerceptionActionCycle* is used to connect the *Sensor*, *Processor*, *Optimizer*, *DisplayManager* and *StorageManager* components for an adaptive radar implementation. The *TimingManager* keeps track of the timings for measurements, processing and optimization and shares this information with the *PerceptionActionCycle* where it is needed.

The process described reflects a sequential and iterative process that is performed over a total observation time/defined number of CPI's. A PAC implementation can be representative of a single mono-static radar system and when considering a multi-static or multi-node radar system, each PAC can become representative of each individual mono-static radar system. However, a larger abstraction is the view that each PAC implementation is composed of a sense of an environment (sensed using hardware or passed from a lower tier PAC), a processed perception of that environment and an action execution.

Approved for public release; distribution is unlimited.

The FARMS algorithm provides one the ability to conduct adaptive radar research through advanced radar simulations that can be supported by laboratory experiments. By being able to simulate adaptive radar concepts and perform collections using the CREW test-bed, investigations in new and vast adaptive radar concepts can be investigated. Investigations performed using this algorithm will be presented in the chapters to follow.

Approved for public release; distribution is unlimited.

Chapter 5: Neural Network Based Control of an Adaptive Radar

[1] This chapter discusses the integration of a neural network in the FAR framework to replace the non-linear optimization at the core of a FAR system. Use of the neural network is shown to reduce computational complexity, leading to faster updates of the radar parameters. The feed-forward neural networks used were trained using the *Levenberg-Marquardt* (LM) algorithm along with a *Generalized Regression Neural Network* (GR-NN) architecture. Training data was obtained from an existing FAR optimization system using the system's parameter selections and target states for single-target tracking in range and velocity. The trained networks were then used to perform radar parameter selection based on the predicted (forward propagation of prior probabilities) probabilities represented by the target state (target range, velocity & SNR) and error variance (target range and velocity variance). Results are compared between the existing optimization-based and neural-network-based control of the system. Both simulation and experiments involving single-target tracking are used. The remainder of this chapter is organized as follows. Section 5.1 introduces the problem considered and the motivation behind using neural networks. Section 5.2 reviews the approach for neural network implementation and analysis through

[1]Manuscript in Review

Approved for public release; distribution is unlimited.

discussion of the optimization at the center of the fully adaptive radar framework that provides the baseline that the neural network is designed to emulate. Section 5.3 presents and discusses the simulation and experimental results. Concluding remarks on the work are then provided in Section 5.4.

5.1 Motivation

Cognitive or FAR is an area of growing interest in the radar community due to the allure of having a radar system capable of autonomous adaptation of its properties in order to achieve a number of different tasks such as improved tracking, better detection, and improved environmental sensing. Implementing cognition into radar tasks can improve performance by autonomously optimizing radar parameters for a specified task against an observed scene. This approach has been successfully demonstrated for target tracking, detection, and imaging [4, 71, 72]. Often, this autonomous adaptation is based on a non-linear constrained optimization that can be computationally costly. For the FAR framework of [7, 71], this optimization is realized in an "executive processor" that attempts to balance the cost functions of both the performance and measurement according to radar tasks and goals.

Conceptually similar to FAR, machine learning is a field of artificial intelligence that investigates how humans are able to learn and acquire knowledge and the representation of these concepts within machines [9]. Neural network systems typically use offline learning methods to eliminate the need for heavy processing loads at system runtime. Neural networks are particularly known for emulating the recognition ability of humans [73], and continue to gain popularity for their ability to model and realize the learning process [10]. Association, pattern recognition, and learning relationships

Approved for public release; distribution is unlimited.

are all tasks for which neural networks are efficient. Their capability of being universal approximators [74] also makes them suitable for function approximation and parameter estimation.

For function approximation tasks, arguments can be made for use of alternative methods such as a lookup table in which a specified set of inputs produces a deterministic output [75]. While a lookup table approach could also be applied to replace the optimization of a fully adaptive radar, a large number of inputs can rapidly make the table dimensions so large that all possible parameter cases cannot be *a priori* considered in producing the table. The approximations inherent in a trained neural network, however, can provide potential improvements in interpolating or extrapolating from the values on which the network was trained to other conditions. Hash tables could also be an option for this process [75], but suffer similar issues as those identified for lookup tables. The static nature of both lookup and hash tables also leave no flexibility for adjusting the mapping of input to output [76]. Recurrent/Feedback neural networks are capable of providing the benefit of being able to learn on the fly, allowing the networks to become more dynamic and less static in response to a changing environment.

FAR Single Target Tracking Problem

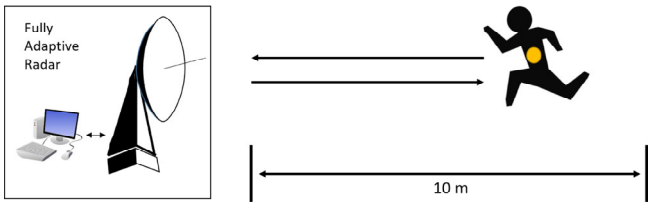


Figure 5.1: Fully Adaptive Radar Modeling and Simulation Framework

Approved for public release; distribution is unlimited.

Although the results to be presented can be generalized to many other configurations, the specific scenario examined in detail is defined here to clarify subsequent discussions. Both simulated and measured datasets were used for a scene in which a single target is moving towards and away from (sinusoidal motion) a radar within a range of 10 meters as seen in Figure 5.1. An existing fully adaptive radar simulator was used to generate simulated datasets [4, 71]. The simulated radar return waveforms to generate the simulated datasets for the scene include a stochastic model for the target waveforms (which varies from dwell-to-dwell according to a Gaussian distribution), changes in the received power with range according to the radar range equation, thermal noise contributions, and simulated stationary clutter.

The RCS scattering from the target was modeled as a distributed target (collection of point targets) with a Gaussian distribution (in which the standard deviation is 10% of the mean value) in order to attempt to capture the distributed nature of a time varying target (for example a walking human as used in the laboratory experiments performed). The target moved within a 10 meter range, with varying absolute velocities between 0 - 2.7 m/s and SNR fluctuations from 20 to 50 dB.

The existing framework's simulator also accounts for the effects of varying radar waveform parameters such as the f_{PRF}, the N_{p} in a CPI, the waveform bandwidth B, transmit power P_{t}, and the τ of the simulated LFM waveform depending on the target range and the radar waveform parameters used.

It should be expected that longer pulse lengths τ and higher transmit power P_{t} should be desirable when the target is at increased range due to the improvements in SNR that would be achieved. Changes in waveform bandwidth impact the nominal system range resolution $\delta R = \frac{c}{2B}$ and the associated tracking performance; it is noted

Approved for public release; distribution is unlimited.

that this performance is also dependent on the target SNR [7]. Changes in the f_{PRF} and the number of pulses in a CPI affect the velocity precision $\delta V = \frac{\lambda}{2}\frac{1}{CPI} = \frac{\lambda f_{\text{PRF}}}{2N_{\text{p}}}$, while the f_{PRF} alone affects the maximum Doppler frequency amplitude to avoid Doppler aliasing $f_{D_{max}} = \frac{f_{\text{PRF}}}{2}$. The number of pulses in a CPI further impacts the SNR of the target due to the coherent processing performed within a CPI.

In terms of radar resource usage, increased B, P_{t}, τ, N_{p}, and f_{PRF} values all represent increased resource use in terms of either energy consumed or the update rate of radar measurements. For this reason, we define a radar "measurement cost" function as:

$$C_M(\boldsymbol{\theta}_k) = \sum_{i=1}^{5} w_i \left| \frac{\theta_{k,i} - \theta_i^{\dagger}}{\theta_i^{\dagger}} \right| \tag{5.1}$$

where $\theta_{k,i}$ represents the ith sensor parameter at time step k, while $\theta_i^{\dagger}$ is a value for this parameter deemed desirable and w_i is the i^{th} weight selection from a set of weights $\boldsymbol{w}$ used to assign importance to distinct parameters. The measurement cost for the example considered focuses on radar timeline minimization [7]. Note, for this work, we will only consider up to four waveform parameter adaptations (PRF, B, N_{p}, and τ) and hold the transmit power (P_{t}) as a static/non-changing parameter.

In what follows, the four waveform parameters considered will be grouped into a sensor parameter vector $\boldsymbol{\theta}_k$. The goal of the radar adaptation is to select "optimum" parameters $\boldsymbol{\theta}_k$ for a given measurement to minimize the measurement cost while meeting performance requirements for the currently observed scene. The radar parameter ranges available are provided in Table 5.1, and are commensurate with the scenario investigated. Note parameter values are selected from discrete values within the ranges indicated. The goal values and weights for the measurement cost function are provided in Table 5.2 as defined in [7]. Note that, we require a high f_{PRF}, low N_{p},

Approved for public release; distribution is unlimited.

low B and low τ in order to ensure that we have a fast update rate and to minimize the time required for performing a radar task.

The weights, w_i, attributed to each parameter indicates the importance of meeting the goal values through which we observe that it is highly important to use a high f_{PRF} when possible, followed by the importance of using a low N_{p}. These two parameters affect the CPI of the collections which directly affects the track update time. f_{PRF} is given more importance than the other parameters due to the fact that lower f_{PRF} values can lead to Doppler aliasing, which is to be avoided when possible. Reduction in the amount of data the radar has to process (number of fast time samples) can also affect our track update time leading to the lower importance of adapting the B and τ parameters as they affect the number of fast time samples in a measurement.

Examples will be shown in what follows that involve the adaptation of one, two, three, or four radar parameters for the FAR-NN case. The parameters adapted were the f_{PRF} for a single parameter adaptation; f_{PRF} and B for two parameter adaptation; f_{PRF}, B and N_{p} for three parameter adaptation; and f_{PRF}, B, N_{p} and τ for four parameter adaptation. Table 5.5 describes these four cases, and includes the duration of the simulation/experiment as well as the resulting number of CPI's within the measurement period used for the statistical analysis. Similar to the FAR-NN work, examples for the adaptation of four radar parameters will be shown for the *Hierarchical Fully Adaptive Radar with Neural Network Based Control* (HFAR-NN) work, where the four parameter adaptation corresponds to the following four parameters: P_{t}, B, N_{p} and τ. Table 5.6 describes the single four parameter adaptation case to be used for the statistical analysis of HFAR-NN. Note that there are different

Approved for public release; distribution is unlimited.

Table 5.1: FAR-NN Operating Parameters

Parameter	**Number of Steps**	Value Range
f_{PRF}	10	12kHz - 2kHz
N_{p}	10	64 - 1024 pulses
P_{t} [RF]	1	7.5 dBm
B	10	100-1000 MHz
τ	10	0.15- 1.2 μs

Table 5.2: Measurement Cost Function Parameters

Parameter	**Measurement Cost Weight (w_i))**	$\theta_i^{\dagger}$
f_{PRF}	20	12kHz
N_{p}	10	64 pulses
B	1	100 MHz
τ	1	0.15 μs

numbers of samples for each parameter adaptation case even though all of the simulation runs are for the same simulation time. This is a direct result of the adaptation of the radar parameters which causes the CPI to vary, so that a varying number of samples is acquired for a fixed measurement time.

Due to the fact that changing the parameters leads to different CPI lengths, the CPI length varied as a function of f_{PRF} and N_{p}. Given the parameter selections for f_{PRF} and N_{p} in the FAR-NN work, the CPI length could vary from 5.3 ms (f_{PRF}=12kHz, N_{p}=64 pulses) - 512 ms (f_{PRF}=2kHz, N_{p}=1024 pulses); whereas for the HFAR-NN work, the CPI length could vary from 16 ms (f_{PRF}=4kHz, N_{p}=64 pulses) - 128 ms (f_{PRF}=4kHz, N_{p}=512 pulses).

Approved for public release; distribution is unlimited.

Table 5.3: HFAR-NN Operating Parameters

Parameter	**Number of Steps**	Value Range
f_{PRF}	1	4kHz
N_{p}	10	64 - 512 pulses
P_{t} [RF]	10	-1.5 - 10 dBm
B	10	100-1000 MHz
τ	10	0.3- 1.2 μs

Table 5.4: Measurement Cost Function Parameters

Parameter	**Measurement Cost Weight (w_i))**	$\theta_i^{\dagger}$
P_{t} [RF]	1	-1.5dBm
N_{p}	10	64 pulses
B	1	100 MHz
τ	10	0.15 μs

Table 5.5: Cases Examined for FAR Static NN: 120s run

Parameter Adapt Case	**# of Samples (L) [Sim]**	**# of Samples (L) [Exp]**
1 Par (f_{PRF}) [Solver]	484	622
1 Par (f_{PRF}) [NN]	484	629
2 Par (f_{PRF} & B) [Solver]	485	613
2 Par (f_{PRF} & B) [NN]	484	636
3 Par (f_{PRF}, N_{p} & B) [Solver]	457	437
3 Par (f_{PRF}, N_{p} & B) [NN]	460	573
4 Par (f_{PRF}, N_{p}, B & τ) [Solver]	482	541
4 Par (f_{PRF}, N_{p}, B & τ) [NN]	482	693

Table 5.6: HFAR Cases Examined: 1200s run

Parameter Adapt Case	**# of Samples (L) [Sim]**
4 Par (f_{PRF}, N_{p}, B & τ) [Spiral Motion-Solver]	4000
4 Par (f_{PRF}, N_{p}, B & τ) [Spiral Motion-LMGR-NN NN]	4001

Approved for public release; distribution is unlimited.

The simulation model was developed in part to represent the observations of a 94 GHz multi-static software defined radar system at the Ohio State University ElectroScience Laboratory (ESL) known as the CREW [77], for which test results are also reported in Section 5.3. In the experiments performed with CREW, the sinusoidal motions and positions of a human target were captured within the 10 meter space of a controlled lab setting and the radar waveform parameters used were selected over allowed values identical to those used in the simulations.

One difference between simulations and experimental tests is that there are no limits on processing time in the simulations. Radar parameters in simulations can therefore be updated for each CPI regardless of the computational time required to perform any optimization. This is not possible in CREW measurements, so that delays between CPI measurements can potentially occur due to the time required to perform an optimization. It is also important to recognize that adaptive radar measurements vary in response to the scene observed. This implies that repeated trials for configurations that appear similar can result in differing parameter selections over time because of the stochastic properties used for target returns and thermal noise. Comparisons between neural network-based and optimization-based parameter selections and outcomes will therefore be examined in terms of their statistics over the duration of a measurement.

Approved for public release; distribution is unlimited.

5.2 Approach

5.2.1 Optimization-Based FAR

The FAR framework of [7, 71] can be broken down into three major components as shown in Figure 5.2: the sensor hardware, a perceptual processor, and an executive processor. The hardware performs measurements using sensor parameters $\boldsymbol{\theta}_k$ to probe the environment to obtain sensor data $\mathbf{z}_k$, here taken as the range/Doppler map obtained from a radar CPI measurement. This sensor data is processed into a perception of the scene in the perceptual processor through detection, association, and tracking functions performed using the MAP-PF [4, 7, 28] in combination with a tracking algorithm. Costs are attributed to the sensor parameter selections through $C_M(\boldsymbol{\theta}_k)$ as described previously.

The optimization performed by the executive processor is

$$\boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_k} C_M(\boldsymbol{\theta}_k)\ s.t.\ \begin{cases} \sigma_R(\boldsymbol{\theta};\hat{\mathbf{x}}_k) \leq \sigma_R^\dagger \\ \sigma_V(\boldsymbol{\theta};\hat{\mathbf{x}}_k) \leq \sigma_V^\dagger \end{cases}$$

where

$$\boldsymbol{\Theta}_k = \boldsymbol{\theta}_k \in \boldsymbol{Z}^N : \begin{cases} \theta_{k,1}^{lb}(\hat{\mathbf{x}}_k) \leq \theta_{k,1} \leq \theta_{k,1}^{ub} \\ \theta_2^{lb} \leq \theta_{k,2} \leq \theta_2^{ub} \\ \theta_3^{lb} \leq \theta_{k,3} \leq \theta_3^{ub} \\ \theta_4^{lb} \leq \theta_{k,4} \leq \theta_4^{ub} \end{cases} \tag{5.2}$$

Here $\sigma_R^\dagger$ and velocity $\sigma_V^\dagger$ are goals for the track errors in range and velocity, respectively; $\sigma_R(\boldsymbol{\theta};\hat{\mathbf{x}}_k)$ and $\sigma_V(\boldsymbol{\theta};\hat{\mathbf{x}}_k)$ represent predicted values for these quantities as a function of $\boldsymbol{\theta}_k$, given the currently estimated target state $\hat{\mathbf{x}}_k$; $\boldsymbol{\Theta}_k$ is the set of available sensor parameter selections; $\boldsymbol{Z}^N$ is the set of integer values, $\boldsymbol{\theta}^{lb}$ & $\boldsymbol{\theta}^{ub}$ are the lower and upper bounds for the sensor parameter selections, respectively; and $\boldsymbol{\theta}_{k,1}^{lb}(\hat{\mathbf{x}}_k)$ is the parameterized lower bound for the f_{PRF} sensor parameter selection for which the available sensor parameter selections for f_{PRF} may change at a given time

Approved for public release; distribution is unlimited.

t_k. This changing lower bound was included in order to prevent Doppler-aliasing by limiting the set of available sensor parameter selections for f_{PRF} at a given time t_k based on the maximum predicted target Doppler response. The executive processor seeks to minimize measurement cost subject to these performance constraints. It is this optimization process that is to be replaced by the neural network. **Note**: In implementation, the parameter selections for the sensor parameters are optimized based on a set of bounded continuous values but the selections are rounded to correspond to discrete indices.

The expected sensing performance can also be characterized using a "performance function"

$$\begin{aligned} C_P(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k) =& \frac{(\sigma_R(\boldsymbol{\theta}; \hat{\mathbf{x}}_k) - \sigma_R^\dagger)}{\sigma_R^\dagger} u(\sigma_R - \sigma_R^\dagger) \\ &+ \frac{(\sigma_V(\boldsymbol{\theta}; \hat{\mathbf{x}}_k) - \sigma_V^\dagger)}{\sigma_V^\dagger} u(\sigma_V - \sigma_V^\dagger) \end{aligned} \tag{5.3}$$

in which $u(\cdot)$ represents the unit step function, so that costs are assigned only if predicted track errors exceed the goal values. Note that $C_P(\cdot)$ is not explicitly used in the optimization performed in the executive processor; it is rather computed separately as an assessment of radar constraints for the radar measurement performance as compared to the assigned goals.

The FAR framework uses the `fmincon()` MATLAB local solver that implements a sqp algorithm to solve the constrained minimization problem. The reader is referred to [29] and [71] for more information on these routines. Although the local solver cannot guarantee a global minimum, tests were performed in which the local optimizer determined final cost values were compared to those of a global solution. Because only modest differences were found between the global solution and the local solver,

Approved for public release; distribution is unlimited.

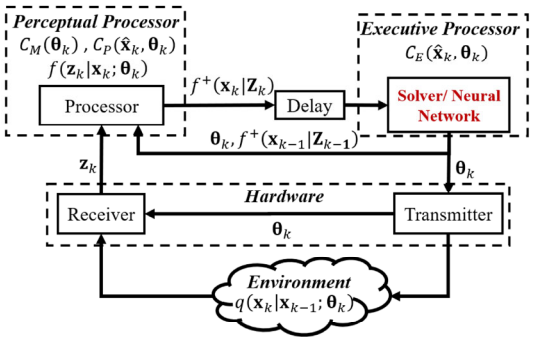


Figure 5.2: FAR Diagram with Neural Network

the local solver was used as it produced solutions faster than the global solver to allow for real-time operation comparison.

For comparisons between the local solver and the neural network, the "executive cost function" $C_E(\cdot)$ is used in some cases:

$$C_E(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k) = \frac{C_M(\boldsymbol{\theta}_k)}{||C'_M(\boldsymbol{\theta})||} + \frac{C_P(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k)}{||C'_P(\boldsymbol{\theta}; \hat{\mathbf{x}}_k)||} \tag{5.4}$$

where $||\cdot||$ is the Euclidean norm, $C_M(\boldsymbol{\theta}_k)$ is the measurement cost value for the k^{th} selection of θ_k, $C_P(\boldsymbol{\theta}_k)$ is the performance cost value for the k^{th} selection of θ_k , $C'_M(\boldsymbol{\theta})$ is the vector of measurement cost values over a collection of L samples and $C'_P(\boldsymbol{\theta}; \hat{\mathbf{x}}_k)$ is the vector of performance cost values over a collection of L samples. The sample count, L, follows from Table 5.5.

This function captures the trade-off between the measurement cost (consuming the least amount of resources) and the performance cost (obtaining low range and track errors).

Approved for public release; distribution is unlimited.

5.2.2 Optimization-Based HFAR

Similar to the FAR framework, the HFAR framework is a multi-static,multi-PAC, multi-tiered adaptive framework which has been applied to Cartesian tracking of a single target through adaptive control of multiple sensors in [28]. As discussed in Section 2.3, the HFAR framework can be understood as a tiered version of the FAR framework that can have multiple PACs, in which each PAC has the same three major components mentioned for FAR as shown in Figure 2.19: the sensor, perceptual processor, and executive processor. A summary of the HFAR framework was given in Section 2.3 while the details can be viewed in [28], therefore the focus of this section is on describing the optimization to be replaced by the neural network.

In a high-level description of the the HFAR framework for Cartesian tracking in [28], there are two tiers: the first tier is structurally similar to the FAR framework discussed in Section 5.2.1 in which the radial tracks $\boldsymbol{r}_k = [\rho\ \dot{\rho}\ \chi]^T$ are obtained by individual sensor PACs observing the environment with their respective measurement parameters $\boldsymbol{\theta}_k^{(1,i)}$, where ρ is the radial range, $\dot{\rho}$ is the radial range rate, χ is the target SNR and $\boldsymbol{\theta}_k^{(1,i)}$ corresponds to the sensor parameters for the i^{th} radar at time k in Tier 1; and the second tier is responsible for sensor fusion from sensor 1 and 2 to produce the Cartesian track estimates $\mathbf{x}_k$ with its "measurement" parameter as $\boldsymbol{\theta}_k^{2,1} = \left[\boldsymbol{R}_k^{\dagger(1,1)}, \cdots, \boldsymbol{R}_k^{\dagger(1,N)}\right]^T$, where $\boldsymbol{R}_k^{\dagger(1,N)}$ is the upper limit on the measurement covariance of the N^{th} radar at time k in Tier 2. The measurement error covariance for the individual radars span the measurement space of Tier 2 of range (ρ) and range-rate ($\dot{\rho}$).

Through this framework, the needs for maintaining an acceptable Cartesian track are optimized for in Tier 2 and restrictions for the radial track measurement error are

Approved for public release; distribution is unlimited.

passed down to Tier 1. The PACs in Tier 1 use these restrictions along with their own needs for radial tracking in order to optimize its selection of measurement parameters for each radar. In viewing this framework, one can see that multiple instances of optimizations are being performed in three places: Tier 2 (measurement covariance restriction selection), PAC 1 of Tier 1 (sensor 1 measurement parameter selection), and PAC 2 of Tier 1 (sensor 2 measurement parameter selection). With these multiple optimizations being performed as well as the possible repetition of optimizations for similar/previously perceived environments, there can be a noticeable increase on the impact to system processing and a limit on real-time performance.

Considering that neural networks can provide a lower impact to system processing, a neural network is proposed for sensor parameter selection at the independent sensor level (at Tier 1). Through this, the sensor parameter selections at Tier 1 will be replaced with a neural network, whereas the optimization taking place at Tier 2 will use the same `fmincon()` MATLAB local solver mentioned in 5.2.1. For the sake of clarity, the optimization taking place for Tier 1, will be mentioned below.

The optimization performed by the executive processor at Tier 1 is

$$\boldsymbol{\theta}_k^{(1,A)} = \underset{\boldsymbol{\theta^{(1,A)}} \in \boldsymbol{\Theta}_k^{(1,A)}}{\arg\min} \; C_M^{HFAR}(\boldsymbol{\theta}^{(1,A)}) \; s.t. \begin{cases} \sigma_\rho(\boldsymbol{\theta}^{(1,A)}) \leq \sigma_\rho^\dagger \\ \sigma_{\dot{\rho}}(\boldsymbol{\theta}^{(1,A)}) \leq \sigma_{\dot{\rho}}^\dagger \\ \boldsymbol{R}_{ii}(\boldsymbol{\theta}^{(1,A)}) \leq \boldsymbol{R}_{ii,k}^{\dagger,(1,A)}, i = 1,2 \end{cases}$$

where

$$\boldsymbol{\Theta}_k^{(1,A)} = \boldsymbol{\theta}^{(1,A)} \in \boldsymbol{Z}^N : \begin{cases} \theta_1^{lb,(1,A)} \leq \theta_{k,1} \leq \theta_1^{ub,(1,A)} \\ \theta_2^{lb,(1,A)} \leq \theta_{k,2} \leq \theta_2^{ub,(1,A)} \\ \theta_3^{lb,(1,A)} \leq \theta_{k,3} \leq \theta_3^{ub,(1,A)} \\ \theta_4^{lb,(1,A)} \leq \theta_{k,4} \leq \theta_4^{ub,(1,A)} \end{cases} \tag{5.5}$$

Here $\sigma_\rho^\dagger$ and velocity $\sigma_{\dot{\rho}}^\dagger$ are goals for the track errors in range and velocity, respectively; $\sigma_\rho(\boldsymbol{\theta}^{(1,A)})$ and $\sigma_{\dot{\rho}}(\boldsymbol{\theta}^{(1,A)})$ represents the predicted values for these quantities as a function of $\boldsymbol{\theta}^{(1,A)}$ for the A^{th} sensor;$\boldsymbol{R}_{ii,k}^{\dagger,(1,A)}$ is the upper bound for the the A^{th} sensor's

Approved for public release; distribution is unlimited.

measurement covariance matrix; $\boldsymbol{Z}^N$ is the set of integer values; $\boldsymbol{\theta}^{lb,(1,A)}$ & $\boldsymbol{\theta}^{ub,(1,A)}$ are the lower and upper bounds for the A^{th} sensor parameter selections, respectively. **Note**: In implementation, the parameter selections for the sensor parameters are optimized based on a set of bounded continuous values but the selections are rounded to correspond to discrete indices.

For HFAR-NN, the radar "measurement cost" function is the same as in (5.1) for each sensor PAC, however only B, P_{t}, τ, and N_{p} are adapted and considered. In terms of radar resource usage, increased B, P_{t}, τ, and N_{p} values all represent increased resource use in terms of either energy consumed or the update rate of radar measurements.

The expected sensing performance can also be characterized using a similar performance function as in (5.3), however we replace the state $\hat{\mathbf{x}}_k$ with the individual states for each sensor PAC, $\hat{\boldsymbol{r}}_k$ and its associated range and range-rate errors $\sigma_\rho(\boldsymbol{\theta}_k^{(1,A)}; \hat{\boldsymbol{r}}_k)$ & $\sigma_{\dot{\rho}}(\boldsymbol{\theta}_k^{(1,A)}; \hat{\boldsymbol{r}}_k)$ as well as add an additional performance objective for the measurement covariance,$\boldsymbol{R}_{ii,k}^{\dagger,(1,A)}$.

As in Section 5.2.1, the executive processors used in the two PACs at Tier 1 seek to minimize measurement cost subject to their respective performance constraints and it is this optimization process that is to be replaced by the neural network. The optimization taking place at Tier 1 performs two optimizations (one for each sensor PAC) and the result is two executive costs (one from each sensor PAC).

For comparisons between the local solvers and the neural network, the "executive cost function" $C_E^{(1,A)}(\cdot)$ is used:

$$C_E^{(1,A)}(\boldsymbol{\theta}_k^{(1,A)}; \hat{\boldsymbol{r}}_k) = \frac{C_M^{(1,A)}(\boldsymbol{\theta}_k^{(1,A)})}{||C_M^{(1,A)'}(\boldsymbol{\theta}_k^{(1,A)})||} + \frac{C_P^{(1,A)}(\boldsymbol{\theta}_k^{(1,A)}; \hat{\boldsymbol{r}}_k)}{||C_P^{(1,A)'}(\boldsymbol{\theta}_k^{(1,A)}; \hat{\boldsymbol{r}}_k)||} \tag{5.6}$$

Approved for public release; distribution is unlimited.

where $||\cdot||$ is the Euclidean norm, $C_M^{(1,A)}(\boldsymbol{\theta}_k^{(1,A)})$ is the measurement cost value for the k^{th} selection of $\boldsymbol{\theta}_k^{(1,A)}$, $C_P^{(1,A)}(\boldsymbol{\theta}_k^{(1,A)};\hat{\boldsymbol{r}}_k)$ is the performance cost value for the k^{th} selection of $\boldsymbol{\theta}_k^{(1,A)}$, $C_M^{(1,A)'}(\boldsymbol{\theta}_k^{(1,A)})$ is the vector of measurement cost values over a collection of L samples and $C_P^{(1,A)'}(\boldsymbol{\theta}_k^{(1,A)};\hat{\boldsymbol{r}}_k)$ is the vector of performance cost values over a collection of L samples. The sample count, L, follows from Table 5.5.

This function captures the trade-off between the measurement cost (consuming the least amount of resources) and the performance cost (obtaining low range and track errors) for each individual sensor PAC.

5.2.3 Neural Network Implementation Details

Static Neural Network

The neural network used for part of this work was a feed-forward neural network trained using the LM training algorithm paired with a generalized regression neural network structure. A basic feed-forward neural network uses input, hidden, and output layers to predict the output after being trained. In order to train the neural network, the LM training algorithm was used. The LM training algorithm uses an approximation of a Hessian matrix for a "Newton-like" weight update when performing backpropagation [78]. The neural network structure as well as initial training of the network followed a generalized regression network setup. GR-NNs are a type of radial basis network composed of three layers (input, hidden & output). GR-NNs use single pass training with a radial basis function, in which the network is only trained once, has biases only in the hidden layer, and the cumulative weighted inputs are normalized [79].

Approved for public release; distribution is unlimited.

The neural networks were trained according to the functional block diagram shown in Figures 5.3 & 5.3 for the FAR-NN and HFAR-NN, respectively to perform parameter adaption based on the target state (target range, velocity & SNR), error variances (in range and velocity), and the output local solver optimization $\boldsymbol{\theta}^*$ values selected from a tabulated set of radar parameters. Training datasets were obtained from simulations of the optimizer based FAR & HFAR framework for single-target tracking in range and Doppler.

It should be noted that the network structure shown in Tables 5.7 & 5.8 and captured in Figures 5.3 & 5.4 are of the Input Size (U) x Number of Neurons (Q_1) x Output Size (Q_2).

For the FAR-NN implementation, the five inputs are: range state, velocity state, SNR state, range error (σ_R^2), and velocity error (σ_V^2). The five outputs are the waveform parameters f_{PRF}, B, N_{p}, τ, and P_{t}. However the radar transmit power was held as a constant in the simulation and was not varied, so that a maximum of four adapted parameters result.

For the HFAR-NN implementation, the seven inputs are: range state, velocity state, SNR state, range error (σ_R^2), and velocity error (σ_V^2), upper bound for measurement error in range ($R_\rho^\dagger$), and upper bound for measurement error in range-rate ($R_{\dot{\rho}}^\dagger$). The four outputs are the waveform parameters P_{t}, B, N_{p}, and τ with a fixed f_{PRF} value.

The structure of the network being trained is shown in Figure 5.3 where we see the inputs to the network as well as a breakdown of the layers inside of the network. In Figure 5.3, U, Q_1 and Q_2 correspond to values for the dimensions as captured in

Approved for public release; distribution is unlimited.

Table 5.7: FAR-NN Training Database

Parameter Adapt Case	**Database**	**Samples**	**Neural Network Structure** $(U \times Q_1 \times Q_2)$
1 Par (f_{PRF})	60s sim run	242	$5 \times 242 \times 5$
2 Par (f_{PRF} & B)	60s sim run	243	$5 \times 243 \times 5$
3 Par (f_{PRF}, N_{p} & B)	60s sim run	231	$5 \times 231 \times 5$
4 Par (f_{PRF}, N_{p}, B & τ	60s sim run	243	$5 \times 243 \times 5$

Table 5.8: HFAR-NN Training Database

Parameter Adapt Case	**Database**	**Samples**	**Neural Network Structure** $(U \times Q_1 \times Q_2)$
4 Par (P_{t},N_{p}, B & τ)	120s sim run	2395	$7 \times 395 \times 4$

Table 5.7. The input defined as the collection of the target states and error variances representing the target motion statistics are captured in the $U \times 1$ vector, $\boldsymbol{y}$. This input vector is fed to the radial basis layer, where we apply two functions: the Euclidean distance (dist) and the radial basis activation functions. The Euclidean distance is computed between the $Q_1 \times U$ layer weight matrix ($\boldsymbol{LW_{1,1}}$) and the input vector $\boldsymbol{y}$. The output is then point-wise scaled by the $Q_1 \times 1$ bias ($\boldsymbol{b^1}$) vector resulting in the $Q_1 \times 1$ vector, $\boldsymbol{n^1}$. The radial basis activation function is then applied to this $Q_1 \times 1$ vector ($\boldsymbol{n^1}$), resulting in the $Q_1 \times 1$ radial basis layer output, $\boldsymbol{a^1}$. The second layer represents the output layer in which we receive the input $\boldsymbol{a^1}$ and apply two functions: a normalized dot product weight function (nprod) and a

Approved for public release; distribution is unlimited.

linear activation function (purelin). Given the input, $\boldsymbol{a^1}$ we compute the normalized dot product weight function with the $Q_2 \times Q_1$ layer weight matrix ($\boldsymbol{LW_{2,1}}$) which produces the $Q_2 \times 1$ vector, $\boldsymbol{n^2}$. After this, the linear activation function (purelin) is applied resulting in the neural network's $Q_2 \times 1$ vector output, $\boldsymbol{\theta_{NN}}$. The MSE is then evaluated between the externally provided target values (truth values produced by the solver), $\boldsymbol{\theta^*}$, and the neural network's solution output, $\boldsymbol{\theta_{NN}}$. This error is used to perform the weight update using backpropagation according to the LM training algorithm, until we arrive at user-defined acceptable threshold. Note that the superscripts for $\boldsymbol{n}$, $\boldsymbol{b}$, and $\boldsymbol{a}$ shown here are solely for variable identification purposes and not meant to indicate exponents.

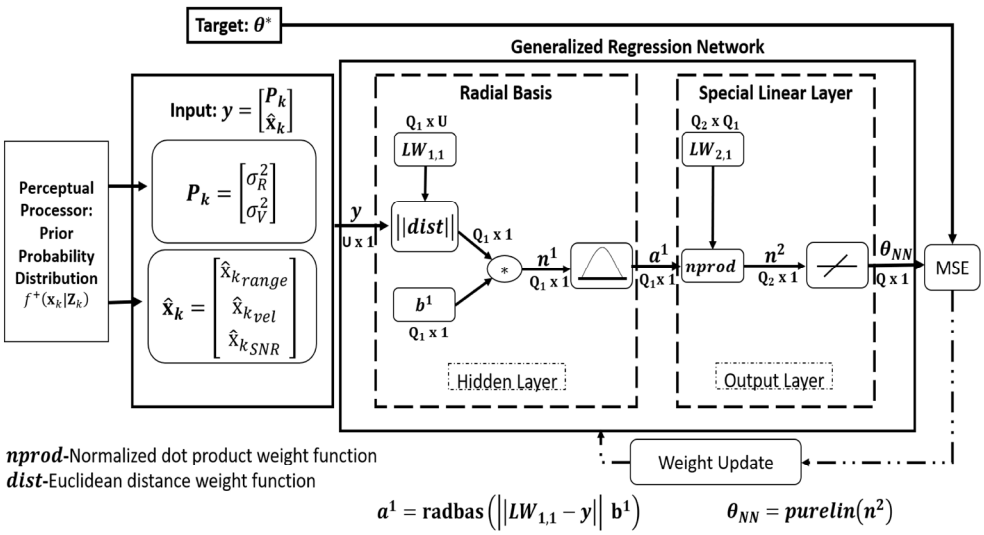


Figure 5.3: FAR Neural Network Training Diagram

Approved for public release; distribution is unlimited.

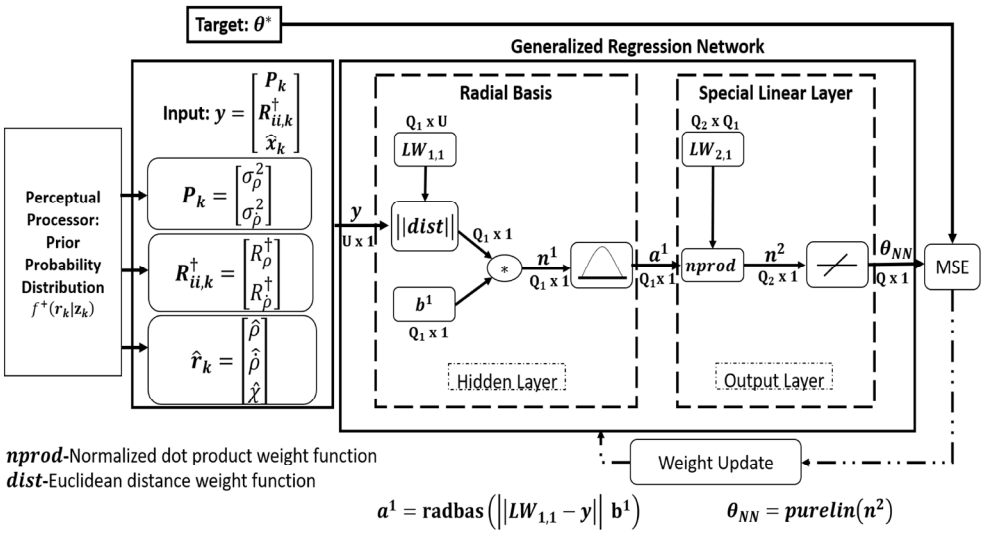


Figure 5.4: HFAR Neural Network Training Diagram

Dynamic Neural Network

In hopes of capturing the benefits dynamic networks offer to an adaptive framework, a LSTM-RNN was implemented using MATLAB's Deep Learning toolbox, where the inputs and outputs of the network remain the same as in the FAR-NN static network but the architecture is shifted from that of a feed-forward network to a feedback/recurrent network. Chapter 2.2.4 reviews LSTM-RNNs, so this section will be used to present the architecture of the LSTM-RNN to be compared to the trained static network (LMGR-NN) and the existing local solver. The comparison between the three optimizers will only be shown for the four parameter adaptation case in which f_{PRF}, N_{p}, B and τ are being adapted. Table 5.10 displays the details of the LSTM-RNN trained on the same database as the static network for the four parameter adaptation case mentioned in Table 5.7.Results based on statistical analysis for

Approved for public release; distribution is unlimited.

Table 5.9: Dynamic NN Cases Examined: 1200s run

Parameter Adapt Case	**# of Samples (L) [Sim]**
4 Par (f_{PRF}, N_{p}, B & τ) [Solver]	4830
4 Par (f_{PRF}, N_{p}, B & τ) [LMGR-NN NN]	4839
4 Par (f_{PRF}, N_{p}, B & τ) [LSTM NN]	4865

this are presented in Section 5.3.2 using a 1200 second simulation run. Details of the simulation run are shown in Table 5.9.

Table 5.10: Network Structure

Block	**Layers**
Block 1	LSTM: Input-240 $\times$ 5, Recurrent-60 Hidden Units
Block 2	FC:60N
Loss Layer	MSE Loss

In Table 5.10, Input refers to the input layer weights of the LSTM-RNN, Recurrent refers to the recurrent hidden units (hidden states) in the LSTM-RNN, FC:xN refers to a fully connected layer with x neurons, and Loss Layer refers to a layer which uses the indicated loss function for error back-propagation.

5.3 Results

The results for this work are presented in order to help promote the benefit of neural networks in an adaptive radar framework, leading to three separate result

119
Approved for public release; distribution is unlimited.

sections. Section 5.3.1 presents the results of implementing a static neural network based on the LMGR-NN in the FARMS framework. Section 5.3.2 presents the results of implementing a similar static neural network based on the LMGR-NN in the HFARMS framework. Finally, Section 5.3.2 discuses the results of implementing a dynamic neural network based on an LSTM-RNN in the FARMS framework.

5.3.1 Static Neural Network FAR Analysis

Previous works in FAR [4] have demonstrated the advantages of the optimization-based approach for radar parameter adaptation in single target tracking as formulated in Section 5.2.1. An initial example of the results obtained from a four parameter adaptation simulation case is shown in order to begin to draw comparisons between the solver-based and the neural network based methods. In Figures 5.5-5.6, we see an example of the four parameter adaptation simulation case where the solver results are held as a benchmark to compare against the neural network results. Figure 5.5 displays the parameter vs. time plot, in which the blue solid line represents the baseline local solver implementation and the orange dashed line represents the neural network implementation for f_{PRF}, N_{p}, B and τ. The parameter adaptations over time fluctuate with the need to meet the range and velocity standard deviation performance metrics shown in Figure 5.6. Figure 5.6 shows the tracks: range, velocity and SNR as well as the performance in terms of the normalized Doppler frequency, velocity standard deviation, and range standard deviation. Figure 5.6 (a) displays the tracks and performance for the solver implementation and Figure 5.6 (b) shows the tracks and performance for the neural network implementation.

Approved for public release; distribution is unlimited.

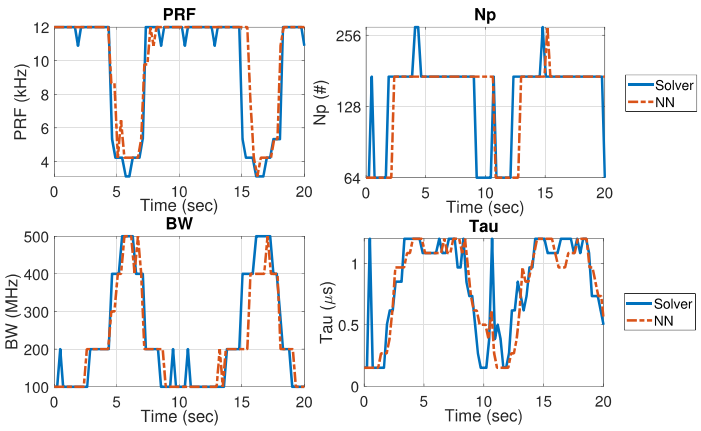
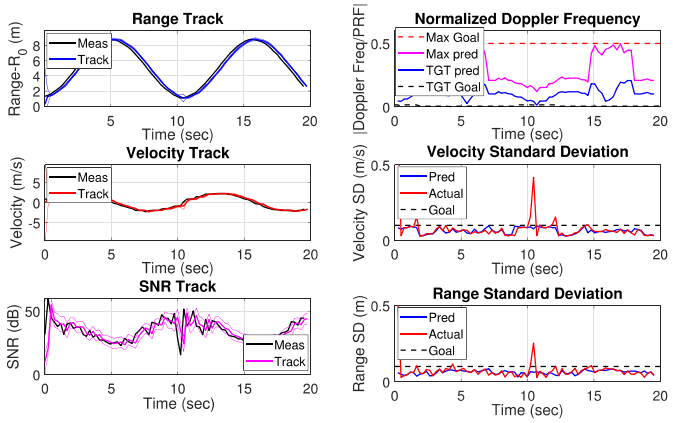


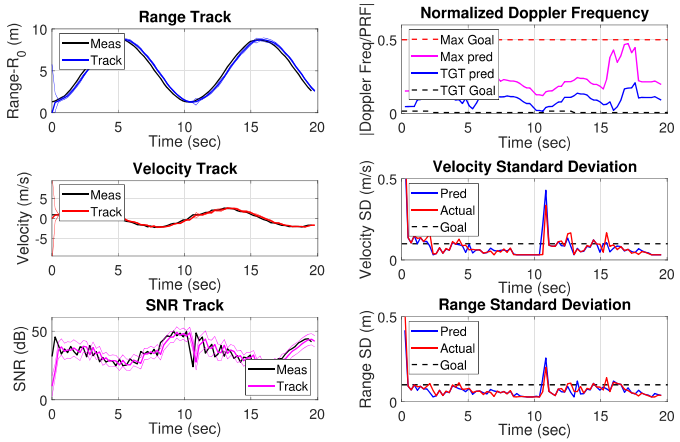
Figure 5.5: 4 Parameter Adaptation over Time: [Sim]

In Figure 5.5, we observe that both the neural network and solver have similar parameter selections. Also the neural network tends to select parameter values that result in lower measurement resources being consumed (ex: higher f_{PRF}, lower N_{p}, lower B, and lower τ). When we pair the results shown in Figure 5.5 with our track and performance over time shown in Figure 5.6, we can begin to attribute meaning to the parameter selections. For example, when the target is further away, our parameter adaptations for B and τ in both the neural network and the solver increase, as described in Section 5.1. Conversely these parameters reduce when the target is nearer, saving radar resources as defined in the measurement cost function C_M. Similar adaptations also occur for the number of pulses and the f_{PRF} in order to maintain acceptable velocity sensing performance.

Approved for public release; distribution is unlimited.



(a) 4 Parameter (Sim-Solver)



(b) 4 Parameter (Sim-Neural Network)

Figure 5.6: Tracks and Performance Results

Note that in simulation, time is static so comparison of the neural network performance to the solver is more of a performance comparison on the neural network's prediction. Since the neural network is a function approximator, it makes sense that its results are not as accurate as the solver. However, when viewing the experimental comparison, we see that due to the speed of the neural network, the approximate

Approved for public release; distribution is unlimited.

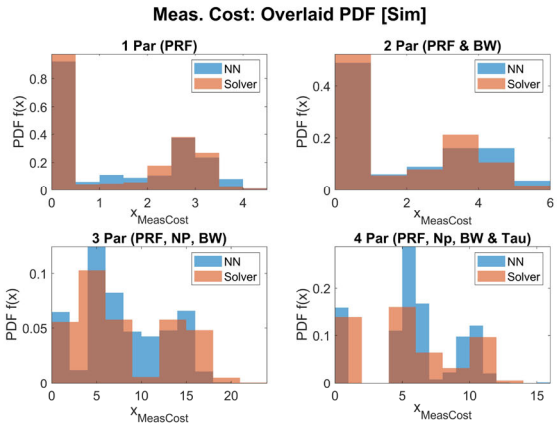
solutions lead to a better overall performance due to having more responses during runtime when needed.

The statistical analysis (using Welch's t-test and the pdf) presented in this section is evaluated on longer runs of both the simulation and experimental setup with the number of samples stated in Table 5.5. Histograms of the measurement and performance costs are shown in Figure 5.7 for the solver-based and neural-network based systems in both simulation and experiment. The results are generally similar, indicating the success of the neural network in reproducing the solver-based parameter selections. However, some moderate differences become apparent as the number of parameters optimized is increased from one to four. The results in these cases show that higher average performance costs (but lower measurement costs) are obtained from the neural network as compared to the local solver in both simulation and experiment. This result suggests that the neural network may require additional training data or a revised structure in order to fully capture the locally-determined solution for these higher dimensional cases. It is noted however that the results for the executive cost, in Table 5.11 remain comparable between the two solutions even in these cases.

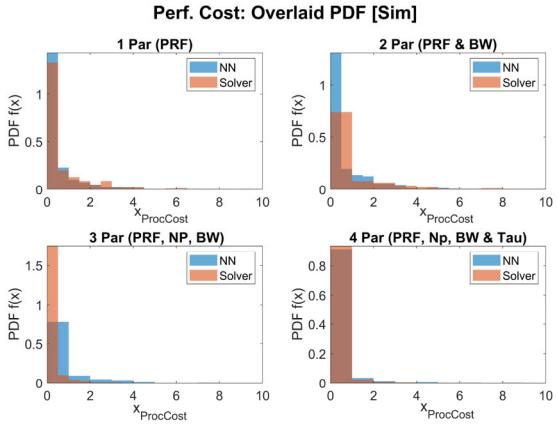
Statistical methods for assessing the "equality" of two samples (i.e. the neural-network selected and solver selected) were also applied. Welch's t-test in particular is used for assessing whether two samples have similar means by evaluating the group mean difference [39].

For this work, our null hypothesis is that the two population means are equal, also known as the two-tailed test. The MATLAB `ttest2()` function was used for this purpose at a significance level $\alpha = 0.05$. Situations in which the p value of the test satisfy $p \leq \alpha$ then indicate a statistically significant difference in the sample

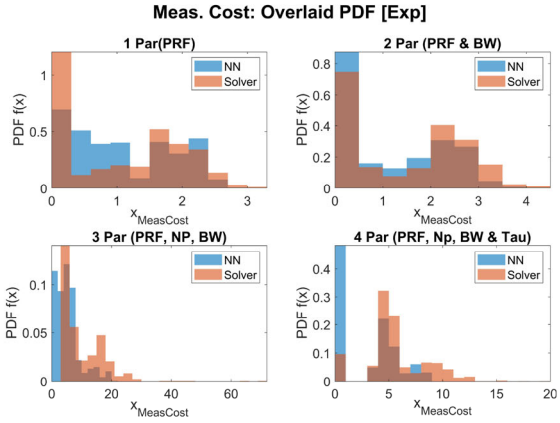
Approved for public release; distribution is unlimited.



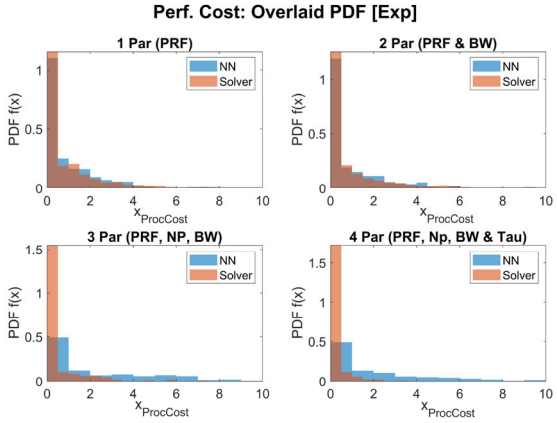
(a) Measurement cost histograms for the solver-based neural network based approaches using simulated data for one to four parameter adaptation.



(b) Performance cost histograms for the solver-based neural network based approaches using simulated data for one to four parameter adaptation.



(c) Measurement cost histograms for the solver-based neural network based approaches using CREW Laboratory measurements of a human target for for one to four parameter adaptation.



(c) Performance cost histograms for the solver-based neural network based approaches using CREW Laboratory measurements of a human target for for one to four parameter adaptation.

Figure 5.7: Histograms of measurement and performance costs for both simulated and measured datasets

Approved for public release; distribution is unlimited.

means (different average performance)) resulting in $H = 1$, while $p > \alpha$ indicates that the difference between the sample means is not statistically significant (similar means) resulting in $H = 0$.

The resulting hypothesis test result for the measurement, performance, and executive cost function samples are provided in Table 5.11. The p values were evaluated against α, such that when $p \leq \alpha$ the resulting H value is 1, while when $p > \alpha$ the resulting H value is 0. The results show the corresponding H values for the executive cost tend to be 0 across parameter adaptation cases and simulation/experiment, such that the samples are considered to have similar mean costs. It is also worth noting, that although in the higher parameter adaptation cases, our H values for the measurement and processor costs are indicated as statistically significant (indicated by $H = 1$), meaning that there is a difference in the sample means , we still achieve similar performance in our executive cost (indicated by $H = 0$).

Table 5.11: Hypothesis Test Result for Performance Comparison of FAR-NN with a Static NN ($\alpha = 5 \times 10^{-2}$)

FAR Optimizer	**Meas. Cost Diff Hypothesis Test Result** H	**Proc. Cost Diff Hypothesis Test Result** H	**Exec. Cost Diff Hypothesis Test Result** H
1 Par (f_{PRF}) [Sim]	0	1	0
1 Par (f_{PRF}) [Exp]	0	0	0
2 Par (f_{PRF} & B) [Sim]	0	1	0
2 Par (f_{PRF} & B) [Exp]	1	0	0
3 Par(f_{PRF}, N_{p} & B) [Sim]	0	1	1
3 Par (f_{PRF}, N_{p} & B) [Exp]	1	1	0
4 Par (f_{PRF}, N_{p}, B & τ) [Sim]	1	0	0
4 Par (f_{PRF}, N_{p}, B & τ) [Exp]	1	1	0

As a means of further analysis on the results of the neural network performance versus the solver, we can look at the empirical *cumulative distribution function* (cdf)s

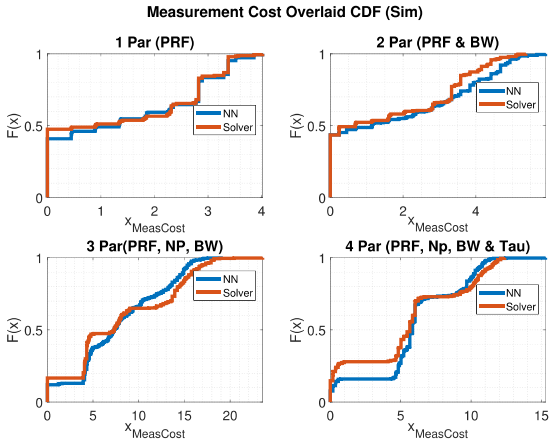
Approved for public release; distribution is unlimited.

of both the solver and the neural network overlaid on each other. The cdf acts as a simpler and clearer analysis of performance comparison by providing the the percentage of the data that falls within a specified range of values. For example, it can be seen that in Figure 5.8 (a) & (b) that the empirical cdf's for the solver and static neural network (LMGR-NN) in simulation are well aligned. Specifically, we see that up until the mean (0.5 on the y-axis), the values obtained by both the solver and network are relatively well aligned and lend to lower values in both measurement and executive costs being obtained; however past this point we begin to see deviation between the network and the solver for the two, three and four parameter adaptation cases. For the simulation, the performance viewed in the cdf between the neural network and the solver are comparable which agrees with the Welch's t-test and the pdf for the simulation case.

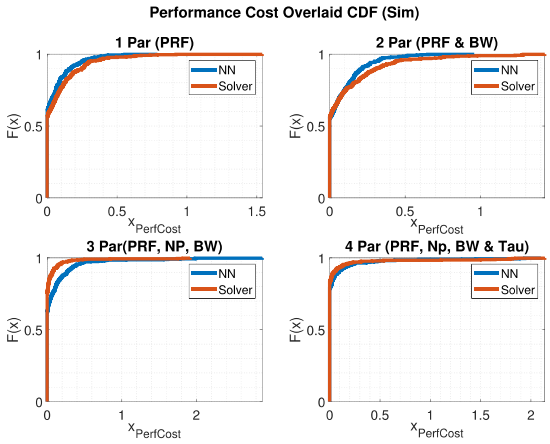
Now, when viewing Figure 5.8 (c) & (d), we see that while the mean of the solver and network are comparable in the different adaptation cases, there are visibly larger deviations between the cdfs of the neural network and the solver, especially for the higher parameter adaptations. These deviations begin to show that similar measurement and performance costs are obtained for the the LMGR-NN versus the solver for the lower parameter adaptations, whereas lower measurement costs and higher performance costs are obtained for the higher parameter adaptation cases.

However, when viewing the costs in terms of overall performance, the executive costs provides the best comparison and when viewing Figure 5.9, we see that in simulation and experiment both the neural network and the solver performances are similar, with the benefit of faster optimizations (due to real-time experimentation) from the neural network leading to slightly lower executive costs.

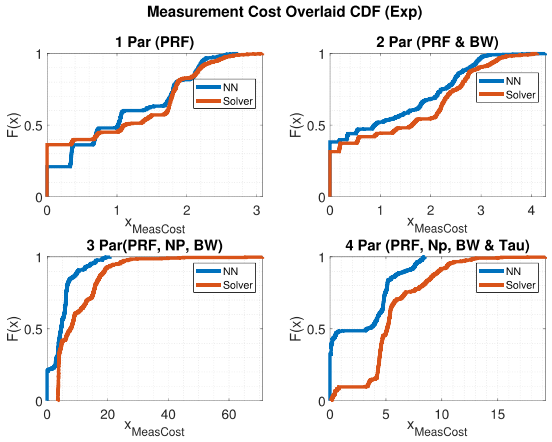
Approved for public release; distribution is unlimited.



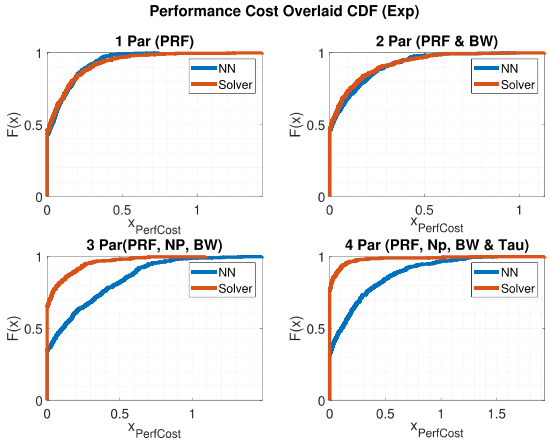
(a) Measurement cost empirical cdf for the solver-based neural network based approaches using simulated data for one to four parameter adaptation.



(b) Performance cost empirical cdf for the solver-based neural network based approaches using simulated data one to four parameter adaptation.



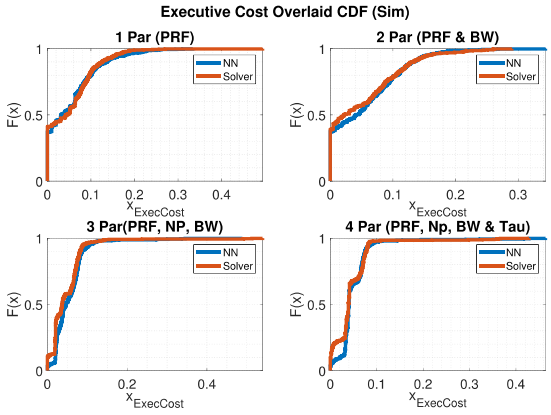
(c) Measurement cost empirical cdf for the solver-based neural network based approaches using CREW Lab measurements of a human target for one to four parameter adaptation.



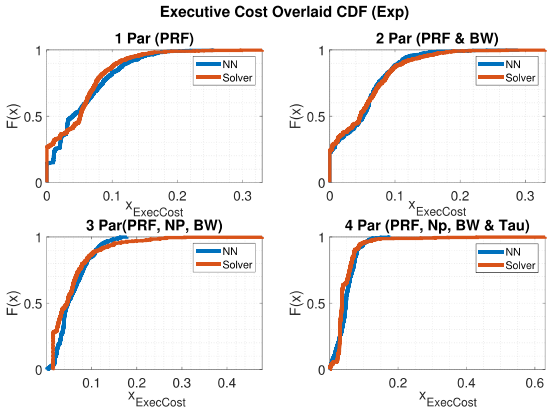
(d) Performance cost empirical cdf for the solver-based neural network based approaches using CREW Lab measurements of a human target for one to four parameter adaptation.

Figure 5.8: Empirical cdf of measurement and performance costs for both simulated and measured datasets

Approved for public release; distribution is unlimited.



(a) Executive cost empirical cdfs for the solver-based neural network based approaches using simulated data for one to four parameter adaptation.



(b) Executive cost empirical cdfs for the solver-based neural network based approaches using simulated data one to four parameter adaptation.

Figure 5.9: Empirical cdf of executive costs for both simulated and measured datasets

Table 5.12: Average Optimization Time of Neural Network & Solver (seconds)

Adapt Case	**Solver** $\times 10^{-2}$	**Neural Network** $\times 10^{-2}$
(1 Par- f_{PRF}) [Sim]	1.04	.34
(1 Par- f_{PRF}) [Exp]	1.03	.39
(2 Par- f_{PRF} & B) [Sim]	1.19	.35
(2 Par- f_{PRF} & B) [Exp]	1.18	.42
(3 Par- f_{PRF}, N_{p} & B) [Sim]	1.39	.35
(3 Par- f_{PRF}, N_{p} & B) [Exp]	1.26	.43
(4 Par- f_{PRF}, N_{p}, B & τ) [Sim]	1.42	.34
(4 Par- f_{PRF}, N_{p}, B & τ) [Exp]	1.52	.39

Additional improvements for the neural network system in the experimental measurements occur due to the faster response time of the neural network, leading to

Approved for public release; distribution is unlimited.

more optimized parameter selections. Insight into this distinction is provided in Table 5.12, which compares average parameter selection times for the local solver versus the neural network and shows selections times faster by an order of magnitude for the neural network. A more detailed examination of solution times showed that the neural network processing time is basically constant throughout the simulation (as expected) while the local solver processing time varied depending upon the scene observed.

5.3.2 Static Neural Network HFAR Analysis

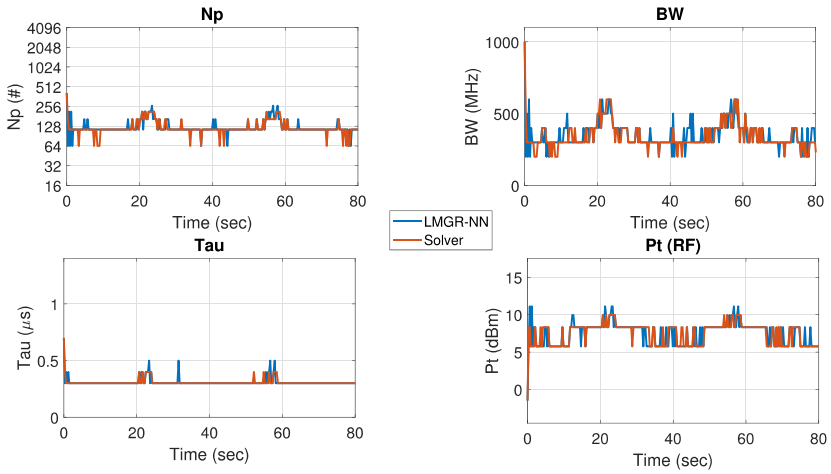
In viewing the results of the previous section, it can be seen that neural networks can begin to provide benefit to an adaptive framework by lowering the impact to system processing and still being able to obtain similar performance to that of a local solver/optimization routine. In this analysis section, we further promote the notion of using a neural network in an adaptive framework in the more complex environment of the HFAR framework. Previous works in HFAR [28] have demonstrated the benefits of the optimization-based approach for radar parameter adaptation in a multi-tiered, multi-PAC, and multi-static single target Cartesian tracking example as mentioned in Section 5.2.2 and Chapter 2.3.

As in the previous section, an example of the results obtained from a four parameter adaptation simulation case is shown in order to begin to draw comparisons between the solver-based and the neural network based methods for HFAR framework, where two radars observe a target moving in an inward spiral starting from the edge towards the center of the field of view. Figures 5.5-5.6, show an example of the four parameter adaptation simulation case where the solver results are held

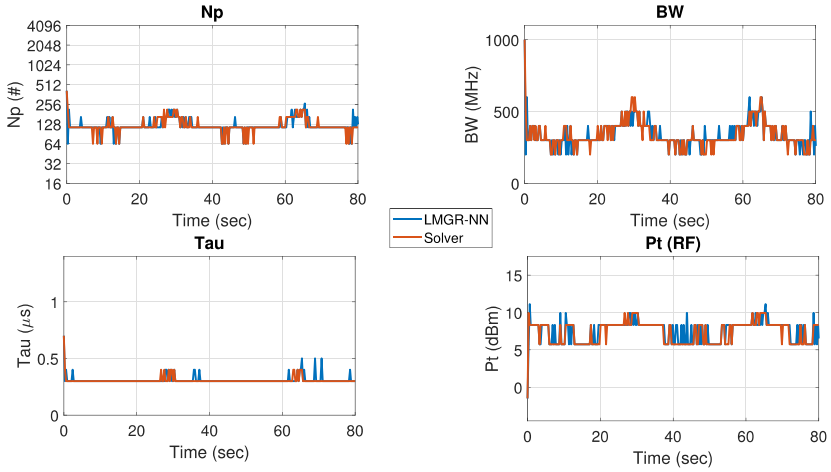
Approved for public release; distribution is unlimited.

as a benchmark to compare against the neural network results. Figure 5.10 (a-b) displays the parameter vs. time plot, in which the orange solid line represents the baseline local solver implementation and the blue solid line represents the neural network implementation, for P_{t}, N_{p}, B and τ for the first and second PACs respectively. The parameter adaptations over time fluctuate with the need to meet the range and velocity standard deviation performance metrics shown in Figure 5.11 and remain below the measurement covariance upper bound in Figure 5.12. Figure 5.11 shows the tracks: range, velocity and SNR as well as the performance in terms of the normalized Doppler frequency, velocity standard deviation, and range standard deviation. Figure 5.12 displays the Cartesian tracks (in x and y coordinates) for position, velocity, position error, velocity error, as well as the measurement error for each sensor represented in the measurement space of Tier 2 by range R_{11} and range-rate R_{22}. Figures 5.11 & 5.12-(a) & (c) displays the tracks and performance for the solver implementation and Figures 5.11 & 5.12-(b) & (d) show the tracks and performance for the neural network implementation.

Approved for public release; distribution is unlimited.



(a) Tier 1, PAC 1



(b) Tier 1, PAC 2

Figure 5.10: 4 Parameter Adaptation over Time: [Sim]

Similar to the adaptations seen in the previous section, it can be observed that both the neural network and solver have similar parameter selections for both sensors (PAC 1 and PAC 2) when viewing Figure 5.10, with the neural networks selections "shadowing" the solver selections. Separate from the adaptations seen in the previous

Approved for public release; distribution is unlimited.

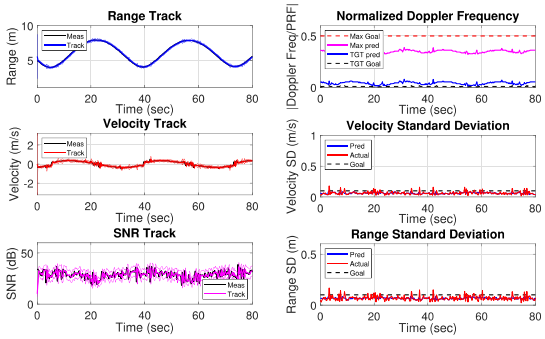
section, the four parameters adapted here are P_{t}, N_{p}, B and τ, where N_{p} is primarily responsible for Doppler resolution (due to the fixed f_{PRF} value), B is primarily responsible for range resolution and P_{t} & τ have direct impact to the SNR. For the HFAR Cartesian tracking framework, the measurement cost was designed to reflect favoring small collection sizes due to the single computer used for processing and to keep the revisit time low (similar to the radar timeline minimization). This translates into meaning that higher importance was placed on avoiding high values for N_{p} & τ, than B & P_{t} which we can see by the more restrictive variation in N_{p} and τ in Figure 5.10 for both sensors (PACs).

In pairing, the results shown in Figure 5.10 with the track and performance over time shown in Figure 5.11, we see that the higher parameter selections for B, P_{t} and N_{p} correspond to the target being further away from the radar, whereas the higher selections for τ seem to correspond to the target being further away from the radar and slowing down. The higher variation in N_{p} than τ seen in Figure 5.10 results from the impact N_{p} has in increasing the SNR response from a target as well as the Doppler resolution. Due to this dual need, we see that while the values for N_{p} are relatively low their high variance results from the need to both achieve better Doppler resolution when the target is near the zero-Doppler and increase the SNR response from a far target.

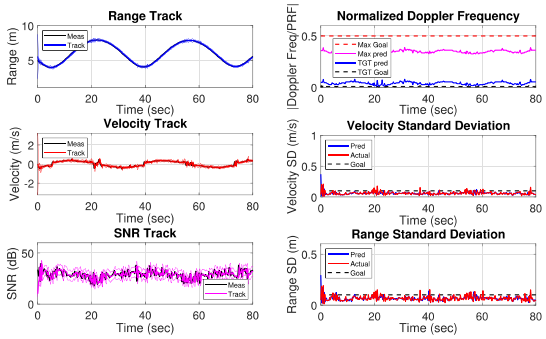
From Figures 5.10 & 5.11, we see that the neural network based control obtains similar results to the solver in both PACs through maintaining the radial tracks at each of the radars and obtaining an acceptable error in both range anf velocity standard deviation by remaining below the goal line. Additionally, when viewing the Cartesian track of Tier 2, PAC 1 in Figure 5.12 we see that the Cartesian track

Approved for public release; distribution is unlimited.

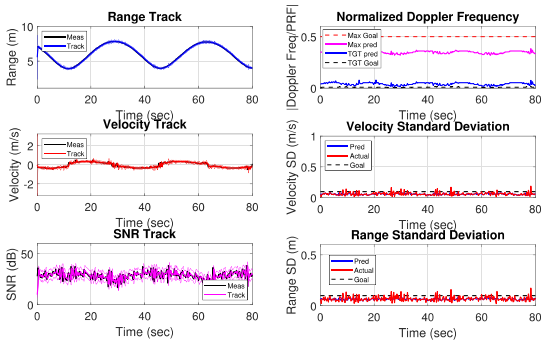
estimates, obtained by fusing the radial tracks obtained at Tier 1 using the neural network parameter selections, are comparable to the estimates and tracks obtained when using the local solver. It should also be noted that we remain below our goal line for the measurement error covariance shown in Figure 5.12 for the neural network, with comparable performance in measurement error to that of the solver.



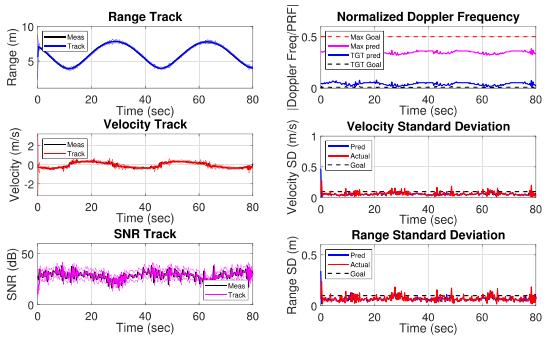
(a) Tier 1, PAC 1 (Solver)



(b) Tier 1, PAC 1 (Neural Network)



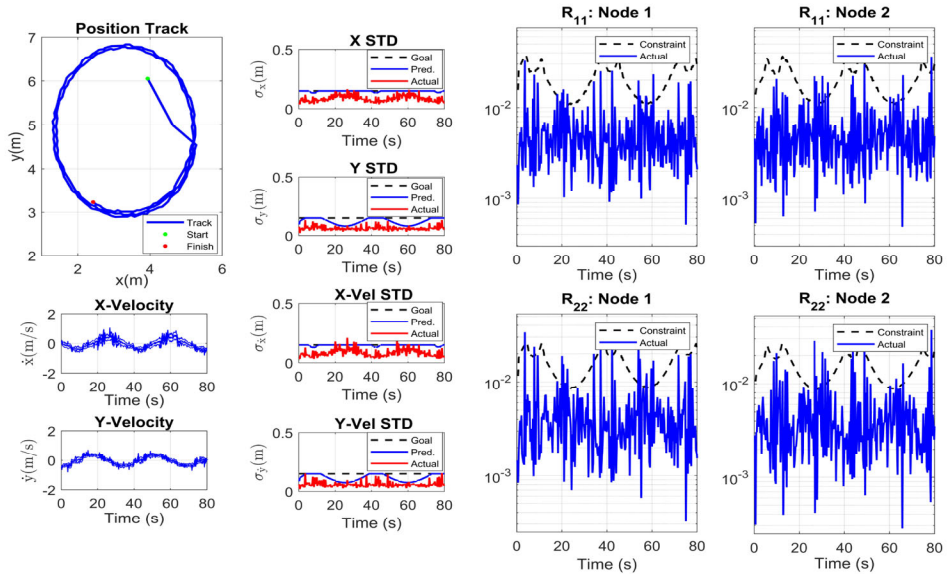
(c) Tier 1, PAC 2 (Solver)



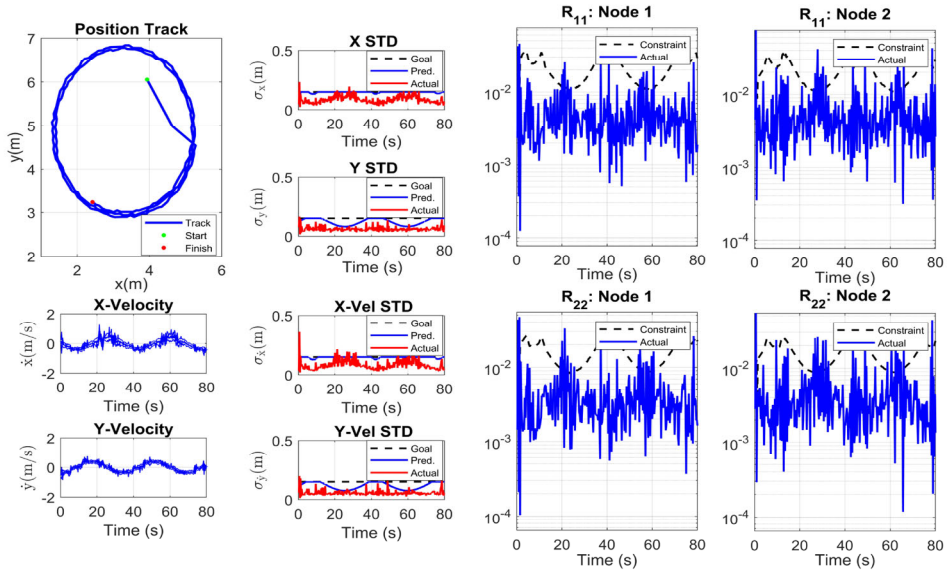
(d) Tier 1, PAC 2 (Neural Network)

Figure 5.11: 4 Parameter Adaptation Tracks and Performance [Sim]

Approved for public release; distribution is unlimited.



(a) Tier 2, PAC 1 (Solver)



(b) Tier 2, PAC 1 (Neural Network)

Figure 5.12: 4 Parameter Adaptation over Time: [Sim]

As in the previous section, the null hypothesis evaluated for the Welch's t-test is that the two population means are equal. The resulting hypothesis test result for

Approved for public release; distribution is unlimited.

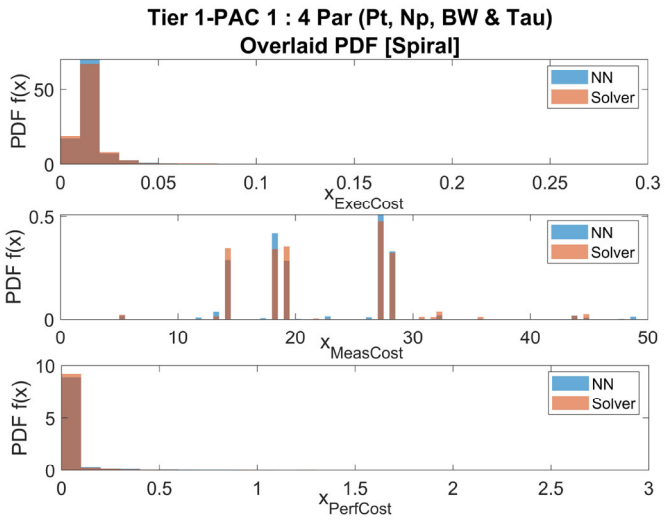


Figure 5.13: Overlaid histograms of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Sim]

the measurement, performance, and executive cost function samples are provided in Table 5.13. The results show the corresponding H values for the executive cost tend to be 0 for both PAC 1 and PAC 2 in simulation, such that the samples are considered to have similar mean costs. Furthermore, we see that the values for the performance costs between the solver and static network are statistically significant (indicated by $H = 1$), meaning that there is a difference in the sample means, whereas there remains similar performance in both the measurement and performance costs (indicated by $H = 0$) for both PACs. This can also be seen from the similar histograms for the neural network and the solver shown in Figures 5.13 & 5.14.

The cdf of the costs is again used here to gain further insight on the comparative performance between the LMGR-NN and the solver for both PAC 1 & PAC 2. In viewing Figure 5.15, we see the cdf for the executive, measurement and performance costs, through which we see that the distributions of both the network and the solver

Approved for public release; distribution is unlimited.

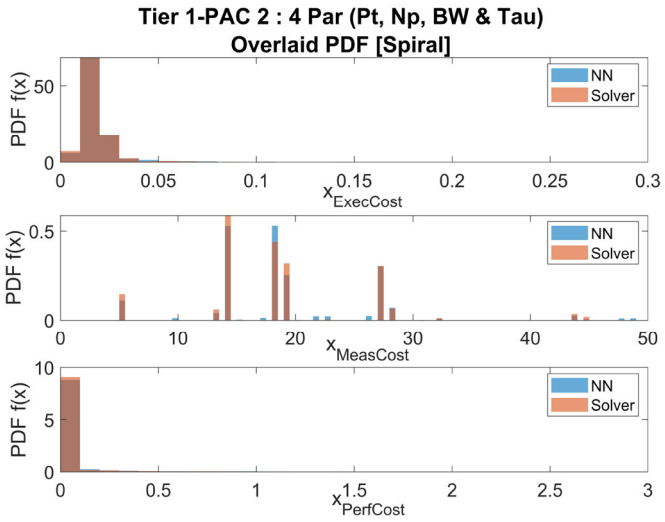


Figure 5.14: Overlaid histograms of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Sim]

Table 5.13: Hypothesis Test Result for Simulation Performance Comparison of HFAR-NN with a Static NN ($\alpha = 5 \times 10^{-2}$)

FAR Optimizer	Meas. Cost Diff Hypothesis Test Result H	Perf. Cost Diff Hypothesis Test Result H	Exec. Cost Diff Hypothesis Test Result H
4 Par (P_t, N_p, B & τ) [Tier 1-PAC 1]	0	1	0
4 Par (P_t, N_p, B & τ) [Tier 1-PAC 2]	0	1	0

Approved for public release; distribution is unlimited.

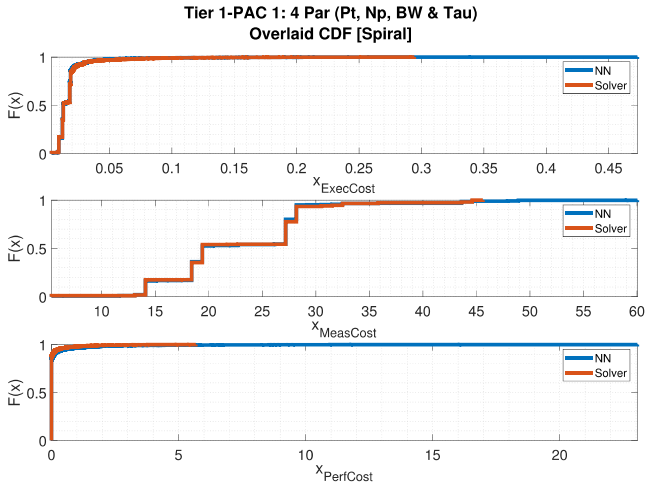


Figure 5.15: Overlaid empirical cdfs of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Sim-Tier 1, PAC 1]

are well aligned, with the network obtaining slightly higher measurement and performance costs at times. Additionally, when viewing Figure 5.16 we see very similar alignment of the distributions of the solver and the LMGR-NN where we again see higher performance and measurement cost values being obtained by the neural network at times. Overall, the comparable performance between the network and the solver shown in the cdfs as well as the higher percentage of low measurement, performance and executive cost obtained by both agree with the performance observed in Figures 5.11 and the results of the t-test. Through this, it can be seen that a neural network can be used in a more complex adaptive framework and obtain similar results to that of a solver/optimization routine while having a lower impact to system resources through its faster optimization timing as noted in Table 5.12.

Approved for public release; distribution is unlimited.

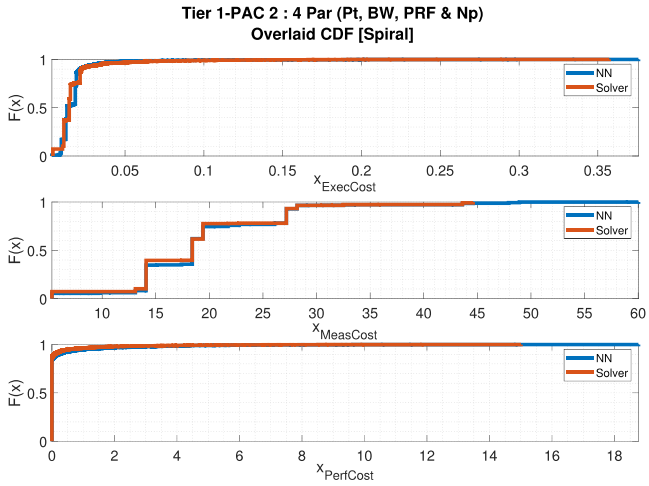


Figure 5.16: Overlaid empirical cdfs of the Executive Cost (top-a), Measurement Cost (middle-b), and Performance Cost (bottom-c): [Sim-Tier 1, PAC 2]

Dynamic Neural Network FAR Analysis

The previous analysis sections have shown the benefit of a static neural network in an adaptive framework and have proven the feasibility of using a neural network for sensor parameter selection in both a FAR and HFAR system. In line with the motivation of using neural networks for control of an adaptive framework, this section is used to present the benefit of an instance of a dynamic network in a FAR system. In this section, a LSTM-RNN is specifically used due to its ability to learn "on-the-fly" through adapting its hidden states in accordance with filtering its input and using relevant data to update its response. The recurrent nature of the LSTM allows it to infer time series progression due to prior history [80]. The selection of the number of hidden units for the LSTM was tuned in order to obtain acceptable performance. The number of hidden units in an LSTM can be understood as concrete storage within the

Approved for public release; distribution is unlimited.

network or the number of learnable parameters which affects the learning capacity of the network.

Through use of the forget gate layer, input gate layer, tanh layer, updated cell state, and the filtered output discussed in Chapter 2.2.4, the LSTM can use the updated cell state, c_t and the filtered output, o_t to change the hidden state, h_t over time. Figure 2.17 which displays a LSTM block for a set point in time, t, is shown again in Figure 5.17 for reference. Through adapting the hidden cell state over time, the LSTM network can adapt its generated output according to the cell state and its filtered output. Further details of the LSTM-RNN are discussed in Chapter 2.2.4.

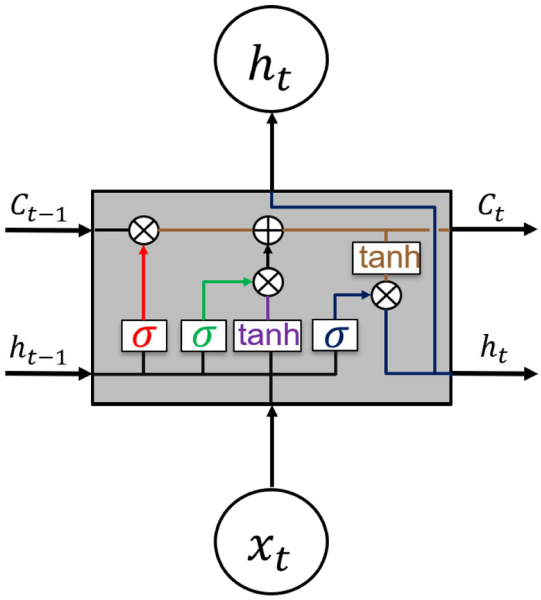


Figure 5.17: Single LSTM Block in time with highlighted components for each composite layer within the LSTM: Forget Gate (red), Input Gate (green), Tanh layer (purple), Updated Cell State (brown), and filtered output (dark blue)

Approved for public release; distribution is unlimited.

An initial example of the results obtained from a four parameter adaptation simulation case is shown in order to begin to draw comparisons between the solver-based, static neural network based and recurrent neural network based methods. In Figures 5.18-5.19, an example of the 4 parameter adaptation simulation case is presented where the solver results are held as a benchmark to compare against both neural network results. Figure 5.18 displays the parameter vs. time plot, in which the blue solid line represents the baseline local solver implementation and the orange dashed line represents the neural network implementation for f_{PRF}, N_{p}, B and τ. The parameter adaptations over time fluctuate with the need to meet the range and velocity standard deviation performance metrics shown in Figure 5.19. Figure 5.19 shows the tracks: range, velocity and SNR as well as the performance in terms of the normalized Doppler frequency, velocity standard deviation, and range standard deviation. Figure 5.19 (a) displays the tracks and performance for the solver implementation, Figure 5.19 (b) & (c) shows the tracks and performance for the static neural network (LMGR-NN) and dynamic neural network (LSTM-RNN), respectively.

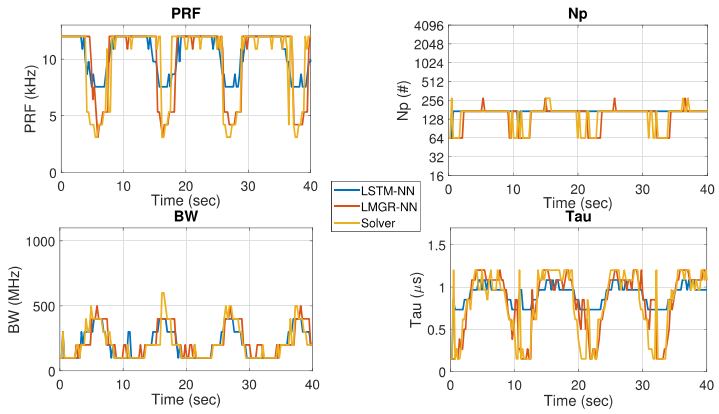
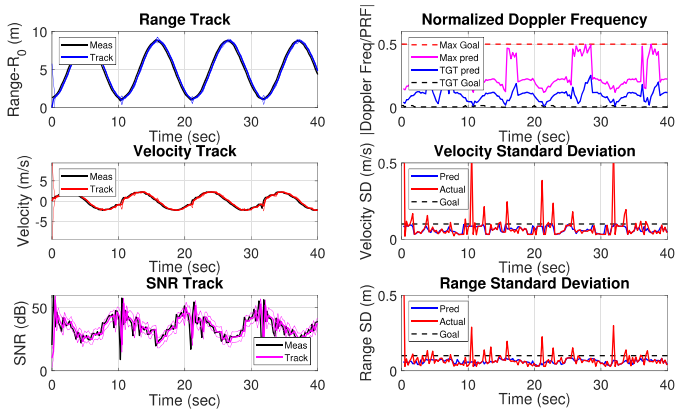
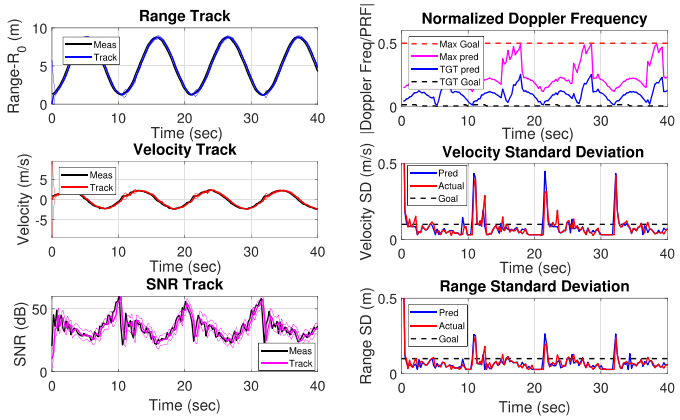


Figure 5.18: 4 Parameter Adaptation over Time: [Sim]

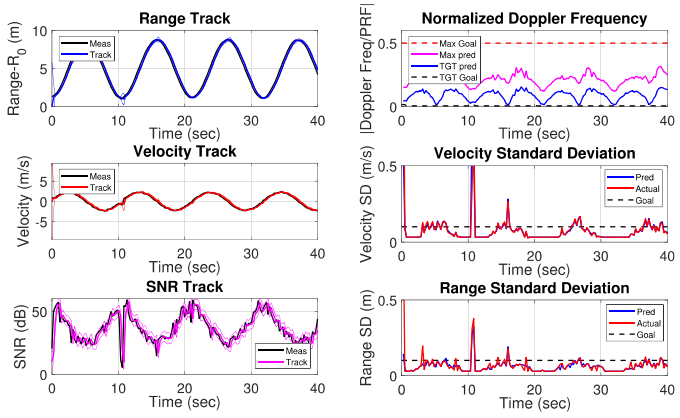
Approved for public release; distribution is unlimited.



(a) 4 Parameter (Sim-Solver)



(b) 4 Parameter (Sim-LMGRNN [Static NN])



(b) 4 Parameter (Sim-LSTMRNN [Dynamic NN])

Figure 5.19: Tracks and Performance Results

Approved for public release; distribution is unlimited.

Similar to the static network results presented earlier, it can be observed that both networks and the solver have fairly similar parameter selections, with the exception of the N_{p} parameter for the LSTM-RNN. In comparing the LSTM-RNNs selection to the solver and the static network, we can see that the recurrent network seems to stay within a median range of selected values choosing not to select parameter values that are too low or too high. This parameter selection from the LSTM-RNN leads to an adequate consumption of resources by choosing not to over-spend nor under-spend sensor resources in order to achieve performance goals. An over-spending of resources can be considered higher selections for B,N_{p},τ and lower selection for f_{PRF}; whereas under-spending resources can be considered to be lower selections for B,N_{p},τ and higher selection for f_{PRF}. When we pair the results shown in Figure 5.18 with the track and performance over time shown in Figure 5.19, this feature can be highlighted. Specifically, in the case where the target is further away, the values for f_{PRF} and B shift to higher resource consumption values (ex:low f_{PRF} and high B) for the solver & the static network, but keep a lower consumption for the dynamic network.

One possible explanation for this selection is due to f_{PRF} being weighted the highest in the measurement cost, resulting in the LSTM-RNN selecting values that are not too low and compensating for this by setting N_{p} to higher values over the duration of the simulation as it has a lower cost than f_{PRF} in the measurement cost. Of particular note is the consistently higher selection of τ and N_{p} in comparison to the solver and static network by the LSTM-RNN in Figure 5.18, which leads to a better track performance when the target gets near the zero-Doppler and is closer to the radar. This higher selection for τ may be due to the importance of its impact

Approved for public release; distribution is unlimited.

on the measurement cost (captured through the weight) being lower than the other parameters resulting in more flexibility for its adaptation as mentioned in [7].

Another highlight to make is that there appears to be a learned perception with the LSTM-RNN, where we see when viewing Figure 5.19 (a-c) that there is a point at $\sim$ 10 seconds where the sensor is unable to prevent the target from entering the zero-Doppler region (through increasing Doppler resolvability) which results in a decreased SNR from the nulled zero-Doppler region and an increase in error for range and velocity for the solver, the static network and the dynamic network. However, we notice that while this trend of the target entering the zero-Doppler continues for both the solver and the static neural network, it is prevented for the LSTM-RNN over time.

This adapted performance begins to highlight the benefit of a dynamic memory-based network in an adaptive framework through its ability to adapt its behavior according to previous perceptions in order to improve performance. The avoidance of losing the target in the zero Doppler when the target is close obtained by the LSTM-RNN is due to its higher τ and N_{p} selections at these points. In particular, we can see that at first the LSTM-RNN around $\sim$ 10 seconds chooses to lower τ and lower N_{p} as the target gets closer and begins slowing down, then due to the high error that results it increases the values of τ and N_{p} soon after around $\sim$ 10.8 seconds. "Learning" from its past decision of lowering the value of τ and N_{p} as the target gets closer and slows down, the LSTM-RNN chooses to leave N_{p} at a higher value and decrease τ. While this is still an initial analysis of the use of dynamic networks with an adaptive radar framework, its performance is promising and offers

Approved for public release; distribution is unlimited.

additional benefit over static and solver-based approaches through its ability to recall previous perceptions.

Similar to the statistical analysis presented in the previous section, the results of the LSTM-RNN, LMGR-NN and the solver were compared quantitatively using the pdf and cdf evaluated on longer runs of the simulation with the number of samples used for the analysis stated in Table 5.9. Histograms of the measurement, performance and executive costs are shown in Figure 5.20 for the solver-based and neural-network based systems in simulation for two cases. The first case (Case 1) is for a simulation where the target moves within a 10 meter range with a sinusoidal motion at absolute velocities between 0 - 2.7 m/s, whereas the second case (Case 2) is also a simulation but for a different sinusoidal motion in which the target is moving within an 8 meter range at absolute velocities between 0 and 1.1 m/s. Both cases use the 1200 second simulation run with a different random seed for repeatability.

In viewing the histograms between the LSTM-RNN, the LMGR-NN and the local solver in Figures 5.20 & 5.21, we see for both motion cases the LSTM-RNN obtains a low performance cost due the maintenance of a median measurement cost value, resulting in a lower executive cost as compared to the solver and the static network in the pdf. Furthermore, in Figures 5.20 and 5.21, we see that the frequency of obtaining a lower performance cost is very similar between the LSTM-RNN and the solver, meaning that similar performance is obtained between the LSTM-RNN and solver for both Case 1 and Case 2. Additionally it should be noted that the color change seen in the plots aside from the orange, blue and yellow are due to the overlay of the pdfs.

Approved for public release; distribution is unlimited.

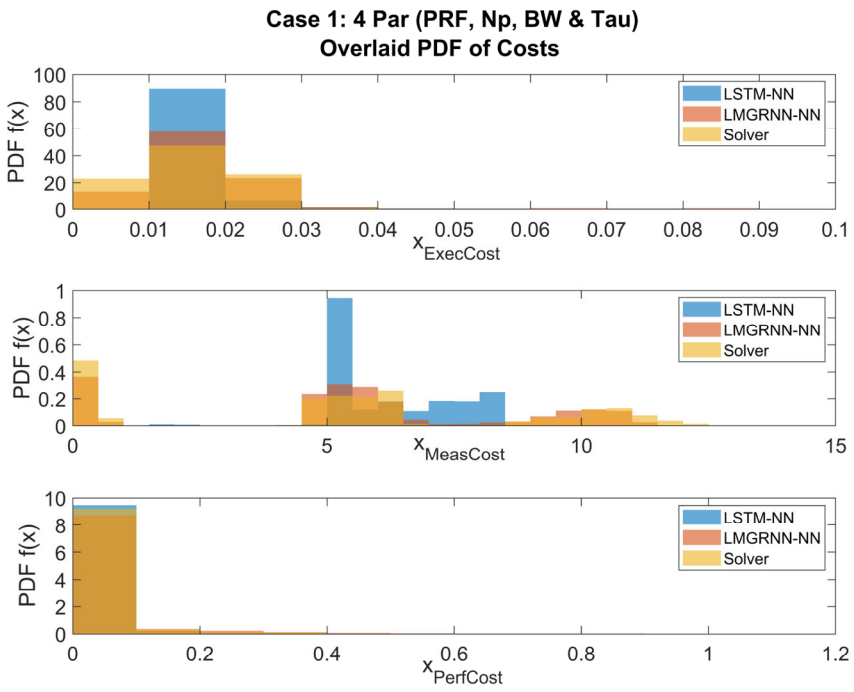


Figure 5.20: Overlaid histograms of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Case 1- Sim]

Approved for public release; distribution is unlimited.

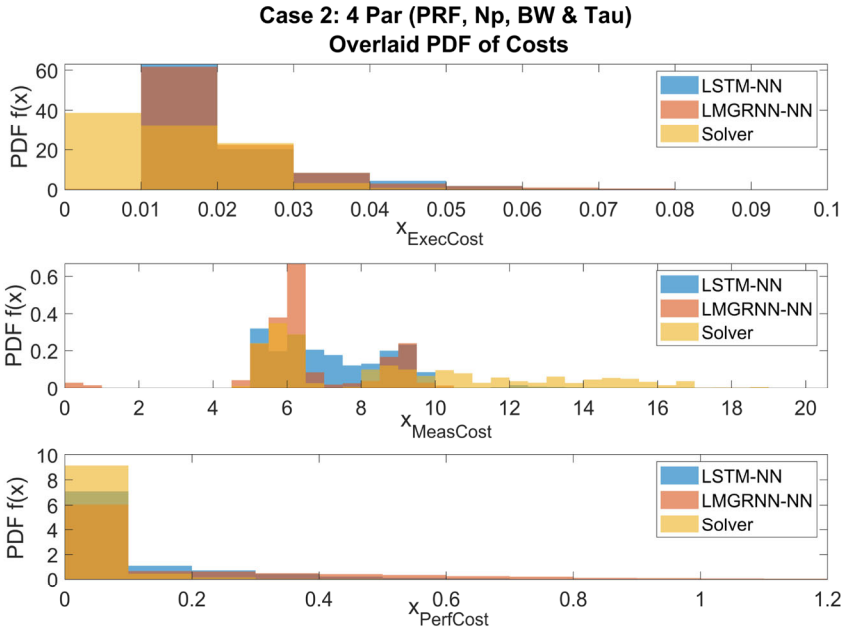


Figure 5.21: Overlaid histograms of the Executive Cost (top), Measurement Cost (middle), and Performance Cost (bottom): [Case 2-Sim]

For additional insight into this, we view the cdf plots in Figures 5.22 and 5.23 where we see that the LSTM-RNNs distribution in measurement and executive cost is visibly deviated from that of the static neural network (LMGR-NN) and the solver. Particularly, we see that in Case 1, unlike the static (LMGR-NN) and the solver, the LSTM-RNN does not have executive cost values below 0.01, however it is more likely to have executive cost values lower than 0.02 than both the static neural network and the solver. This is again viewed for Case 2 in Figure 5.23, but we see that the solver and the static network also do not have many values in executive cost below 0.02. This resulting lower executive cost for the other two optimizers, is due to the fact that the LSTM-RNN sacrifices measurement resources more often in order to obtain a lower performance cost. For Case 1, this supports the observations from the

Approved for public release; distribution is unlimited.

parameter selections over time in Figure 5.18 and the resulting track performance for the LSTM-RNN in comparison with the static network and the solver from Figure 5.19. When looking at the cdf for Case 2 in Figure 5.23, it can be hypothesized that the dynamic nature of the LSTM allows it to still perform well given a change in the observed scene's motions, whereas the LMGR-NN, while still performing adequately, suffers in its performance and executive costs due to the static nature of the network. It should be noted that further generalization of the static network can occur when given more varied training data.

While the LSTM-RNN has the benefit of being dynamic in its response and the ability to obtain lower performance costs in comparison to the solver and the static neural network, there is a a cost of sacrificing the optimization timing. The optimization timing of the LSTM-RNN implementation is on the same scale as the solver, with the static neural network operating an order of magnitude faster than both the dynamic network and the solver. This is due to the `predictandUpdateState()` function used in MATLAB which is used to both produce the network output and update the network state for the next set of inputs, using the network's current states and inputs. While this may shift from the notion of lowering the impact to system processing, it should be noted that there are methods in which LSTM-RNN may increase its processing speed such as the use of a *graphical processing unit* (GPU) or creation of a *.mex* file for the processing intensive portions of the LSTM. While, methods of speeding up the LSTM-RNN were not performed here, a view of the benefit of a dynamic network in an adaptive radar framework was investigated and provides promising initial improvements to the FAR framework worthy of further investigation.

Approved for public release; distribution is unlimited.

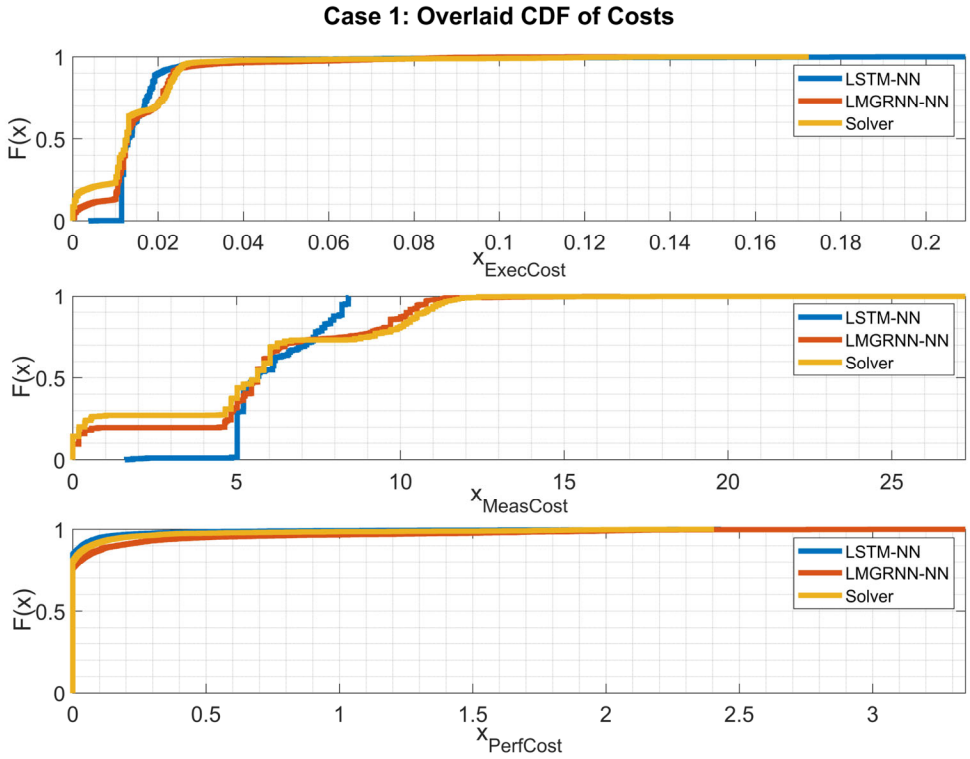


Figure 5.22: Overlaid empirical cdfs of the Executive Cost (top-a), Measurement Cost (middle-b), and Performance Cost (bottom-c): [Case 1-Sim]

Approved for public release; distribution is unlimited.

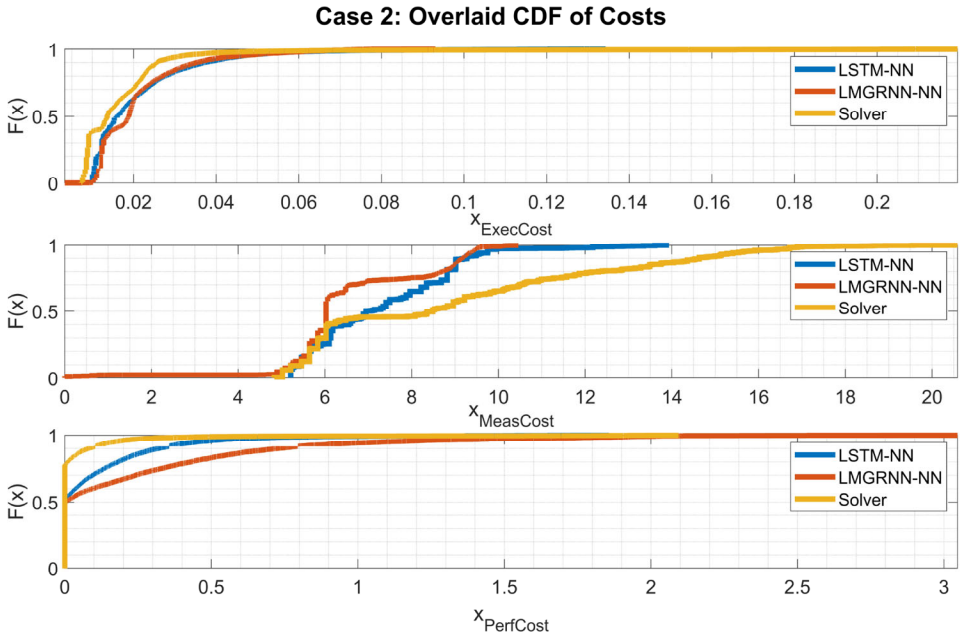


Figure 5.23: Overlaid empirical cdfs of the Executive Cost (top-a), Measurement Cost (middle-b), and Performance Cost (bottom-c): [Case 2-Sim]

Table 5.14: Average Optimization Time of Neural Network & Solver (seconds)

Adapt Case	**Solver** $\times 10^{-2}$	**LMGRNN** $\times 10^{-2}$	**LSTM-RNN** $\times 10^{-2}$
(4 Par- f_{PRF}, N_{p}, B & τ) [Sim]	1.42	.34	5

5.4 Conclusion

The results of the study show that static neural networks can approximate the optimization solutions of a fully adaptive radar system to achieve a greatly reduced

Approved for public release; distribution is unlimited.

processing time for parameter selection. For the single and two parameter adaption cases, similar or improved measurement and performance costs were obtained in the simulated and experimental cases, respectively. It was also observed that similar executive cost performance for all parameter adaptation cases were obtained between the static neural network and the local solver in both simulation and experimental cases for the FAR & HFAR framework implementation. These results were supported in the average costs summary in Tables 5.11 - 5.13 and in comparisons of the statistics of the selected parameters.

With the feasibility of using a neural network for dynamic control of a single PAC adaptive radar framework, it was also shown that neural networks applied to a more advanced adaptive model such as the HFAR framework, can still achieve similar performance to that of a solver. Through the use of a single trained network to implement sensor parameter selection for dual sensors tracking a target in the Cartesian space, the neural network approach provides a lower impact to system processing by limiting the number of solver-based optimizations being performed to a single local solver implemented at the higher tier (Tier 2). The implementation of a neural network based approach to sensor parameter selection for both the FAR & HFAR framework help promote the benefit of a static neural network for parameter selection.

In addition to this, a preliminary implementation of a dynamic network based on a LSTM-RNN was used to show the benefit of a network capable of learning *"on-the-fly"* and its ability to incorporate memory of its past experiences to its decisions through the adaptation of its internal states, resulting in lower performance costs through an adequate consumption of measurement resources.

Approved for public release; distribution is unlimited.

Through the results shown here, it can be concluded that neural networks are promising as a replacement for local solvers to reduce processing times & further enable real-time radar adaptation (through static neural network implementations) and begin incorporating stronger notions of memory into learning to promote a "truly" cognitive radar.

Approved for public release; distribution is unlimited.

Chapter 6: Rapid Waveform Generation using Neural Networks

In this chapter, we discuss the feasibility of the novel application of RNNs in designing low-latency, near-optimal radar waveforms in a dynamic interference environment. Traditional approaches to adaptive radar waveform design typically require cumbersome optimization routines and highly specialized solvers that can be slow to converge. In an effort to decrease the time of convergence, while still being robust to dynamic environments and practical implementation concerns, we provide results using RNN tools. In these initial trials, we achieve waveform design results with comparable characteristics comparable to those of the ERA. The remainder of this chapter is organized as follows. Section 6.1 introduces the problem considered and provides motivation for using neural networks. Section 6.2 reviews the approach for neural network implementation. Section 6.3 presents and discusses the simulation results of the autoencoding and notched waveform tasks performed. Concluding remarks on the work are then provided in Section 6.4.

Approved for public release; distribution is unlimited.

6.1 Motivation

This chapter focuses on demonstrating the feasibility of using a RNN to generate waveforms, in order to progress towards real-time spectral notching capability. Real-time operation can be possible through using the speed of neural networks for adaptive radar waveform design for complex spatial and spectral environments. An example is shown in Figure 6.1.

Considerable work and success exists in cognitive radar waveform design and optimization in improvement of the SINR in congested environments [81, 82, 83, 84, 85, 86]. The ability to shape the spectrum of the transmit waveform due to hardware advances in *digital arbitrary waveform generators* (DAWG) provides flexibility to transmit nearly endless configurations. Currently, a multitude of waveform design objective functions, exist encompassing a wide variety of RF sensing situations and mode needs. However, solutions are often computationally complex [87]. This reality motivates the acceptance of computationally manageable, but perhaps sub-optimal, waveform solutions.

With the addition of better communication technologies, the growing Internet of Things technologies, autonomous cars and many other radio frequency devices there is very little available spectrum [88]. Due to the lack of availability in the spectral environment, there have been proposed techniques to allow technologies to coexist. One such technique is spectral notching or interference avoidance in which the spectral location of an interferer is sensed and avoided in order to allow coexistence of primary and secondary users [49, 89]. Spectral notching techniques attempt to balance the appeal of having contained, low notches where interfering signals are present in the frequency domain and a constant envelope in the time domain for

Approved for public release; distribution is unlimited.

power considerations. In order to achieve this balance, optimization methods are used to create optimal waveforms for spectral coexistence [55, 90].

Neural networks are a subset of machine learning that has been gaining popularity in different communities for their ability to provide fast and consistently timed solutions with reduced computational complexity in areas of both classification/recognition and regression [91, 92, 93, 94]. With the help of GPU's, the processing time for neural networks can be further reduced [95]. Static neural networks are particularly useful for this task as their weights are pre-established through off-line training and do not change when implemented. Currently, different notched waveform design algorithms exist such as the ERA [96] and the *Reiterative Uniform Waveform Optimization* (RUWO) [97], however most of these algorithms suffer from prolonged and variable time to solution convergence for a given notched waveform objective. Neural network approaches allow development of a system that can produce consistent results with minimal timing. In addition to this, given a fixed set of possible solutions, neural networks can be used to perform function approximation to complex optimization functions and map a given set of inputs to the desired output.

The overall aim of this research is to investigate reducing the time to converge (i.e., latency) of a transmit waveform solution, improve flexibility of the possible waveform solutions and improve convergence from data to decision in what to transmit using modern neural network principles.

Approved for public release; distribution is unlimited.

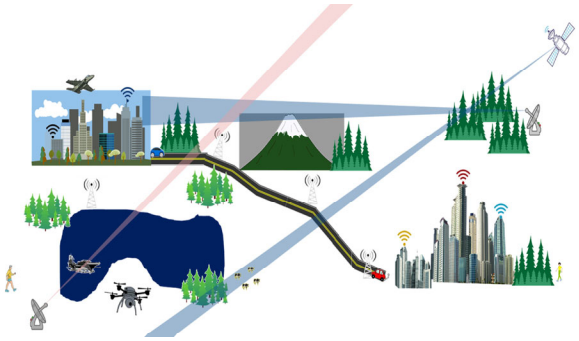


Figure 6.1: Congested/Contested and Dynamic RF Environment

6.2 Approach

The driving force behind the waveform generation is the optimization utilized to create optimal transmit waveforms. The optimality of the waveform is conditioned on two constraints: the spectral content and power constraints. These constraints are represented through the ability to achieve an optimal waveform such that the spectral notch is low and there is a constant envelope/near unity *peak-to-average-power ratio* (PAPR) in the time domain. For a more detailed review of this waveform design concept, the reader is referred to [98].

The work in this chapter uses an LFM chirp for both the autoencoder and notched waveform neural network tasks, in which the waveform parameters used are detailed in Table 6.1. The details of the tasks and their use of the chirp waveform are mentioned in Section 6.2.1.

6.2.1 Neural Network Theory

The network this chapter will focus on is a LSTM-RNN. LSTM-RNN are state-of-the-art [99] for time series prediction, sequence forecasting and even audio generation

Approved for public release; distribution is unlimited.

[100], [101]. Due to the recurrent nature of these networks, they are able to use prior history of the time series in order to perform effectively. LSTM's are also known to be capable of learning a mapping function for inputs over time to an output as well as being mostly noise resilient and capable of modeling nonlinear relationships [102], [103]. Due to its error resilient qualities, its ability to capture patterns through storage of prior history and the sequence forecasting nature of LSTM's, they prove to be a useful network option for waveform design.

In order to establish the feasibility of using this network, the network was first trained for an auto-encoder task in which the network was fed a LFM waveform at the input layer and needed to output the same LFM waveform for transmit, see Figure 6.2. Next, after completion of the autoencoder task, the networks were tasked with producing a notched LFM waveform given an LFM waveform with an interference signal present. Both networks were trained on 10 different self-generated complex time domain LFM waveform signals with all parameters fixed for the unperturbed transmit (TX) except τ at a sampling rate of 1MHz. Similarly to the unperturbed TX waveforms, the interferer waveform parameters were all fixed except B and τ for the notched waveform case. The waveform variety was realized in τ of the transmit signal and interference signal ranging from 0.1ms to 1ms, in which the bandwidth (B) of the unperturbed transmit was held fixed at 200 kHz and the interference bandwidth varied from 1 kHz to 10 kHz (varies with τ for a time-bandwidth product of 1). The waveform parameters used for these two tasks are shown in Table 6.2. The signals of the unperturbed transmit and interference were summed together in order to produce an interference perturbed signal for the notched waveform task. For the autoencoder

Approved for public release; distribution is unlimited.

Table 6.1: Waveform Parameters

Parameters	TX	Interferer
τ	0.1-1 ms	0.1-1ms
B	200 kHz	1-10 kHz
f_0	Baseband (0Hz)	50 kHz

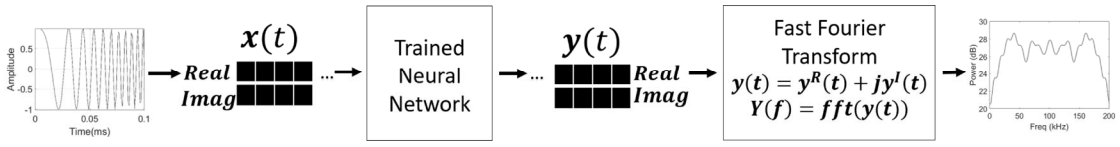


Figure 6.2: Auto-encoder Block Diagram

task, only the unperturbed transmit signal was used (which uses the parameters under TX). Flowcharts of both of these processes can be seen in Figures 6.2 & 6.3.

For this work, instead of predicting the next elements of a time series prediction for a given sequence, the LSTM will be used to generate a suitable time-domain signal given a separate time-domain *In Phase and Quadrature* (IQ) input and task. At this level, the LSTM will be used as a function approximator for two different tasks. The first task is to act as an identity function and map a signal back to itself in an autoencoding task. The second task will be to map a signal with the presence of

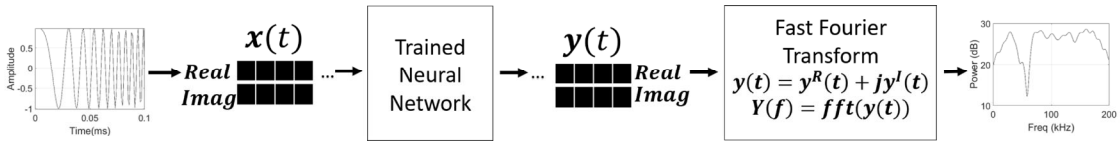


Figure 6.3: Notched Waveform Block Diagram

Approved for public release; distribution is unlimited.

an interferer to a suitable signal with a notch placed at the interfering signal location. The autoencoding task is a preliminary check in order to verify the capabilities of using a neural network for waveform generation. Once accomplished, the task of producing a notched waveform will be the focus.

Once the signals to use were established, consideration for how to feed the data to the input of the network is needed. The method for feeding data into the network for the tests were to use the sampled complex time domain signal as inputs into the network. Based on the task, the network will either produce the suitable notched waveform at the output to be transmitted or produce the same signal at the output. The network approach discussed in this paper uses MATLAB 2019a and its Deep Learning Toolbox to train and test the networks.

The input to the network for the autoencoder task can be defined as the following base-band LFM signal:

$$\boldsymbol{x}_{in}^{AEW}(\boldsymbol{t}) = A\exp(j\pi(\frac{B}{\tau}\boldsymbol{t}^2)). \tag{6.1}$$

where $\boldsymbol{x}_{in}^{AEW}(\boldsymbol{t})$ is the complex time-domain base-band LFM signal input to the neural network.

The input for the notched waveform task is defined as follows:

$$\boldsymbol{x}_{Intf}(\boldsymbol{t}) = A\exp(j2\pi(f_0\boldsymbol{t} + \frac{B}{2\tau}\boldsymbol{t}^2)) \tag{6.2}$$

$$\boldsymbol{x}_{TX}(\boldsymbol{t}) = A\exp(j\pi(\frac{B}{\tau}\boldsymbol{t}^2)) \tag{6.3}$$

$$\boldsymbol{x}_{in}^{NW}(\boldsymbol{t}) = \boldsymbol{x}_{TX}(\boldsymbol{t}) + \boldsymbol{x}_{Intf}(\boldsymbol{t}) \tag{6.4}$$

Approved for public release; distribution is unlimited.

where $\boldsymbol{x}_{in}^{NW}(\boldsymbol{t})$ is the complex time-domain base-band LFM signal input to the neural network made up of two LFM waveforms- the ideal transmit waveform $\boldsymbol{x}_{TX}(\boldsymbol{t})$ and the interference waveform $\boldsymbol{x}_{Intf}(\boldsymbol{t})$.

6.2.2 Long Short Term Memory Neural Network Approach

For the LSTM implementation, a single LSTM network was trained on IQ time domain data samples. The inherent ability of LSTM networks in dealing with time series data allowed for the use of a single network. In addition to this, the complexity of the network was minimal as a shallower (small amount of layers) network gave acceptable results and can be better for runtime by reducing the size of the network. The recurrent nature of the LSTM allows it to infer time series progression due to prior history [80]. The selection of the number of hidden units was tuned in order to obtain acceptable performance. The number of hidden units in an LSTM can be understood as concrete storage within the network or the number of learnable parameters which affects the learning capacity of the network. The network architecture contains a single LSTM layer and a fully connected layer as illustrated in Figure 2.16. It should be noted that performance of the network can be improved through further tuning and different network architectures, however the focus of this chapter was establishing the feasibility of using neural networks for waveform design.

In Table 6.2, Input refers to the input layer weights of the LSTM-RNN, Recurrent refers to the recurrent hidden units (hidden states) in the LSTM-RNN, fc:xN refers to a fully connected layer with x neurons, Dropout Rate refers to a dropout layer of the same size as network block with given rate and Loss Layer refers to a layer which uses the indicated loss function for error back-propagation.

[2]Loss Layer is only used during training for error back-propagation

Approved for public release; distribution is unlimited.

Table 6.2: LSTM Network Architecture

Block	Layers
Block 1	LSTM: Input: 1600×2 , Recurrent: 400 Hidden Units
Block 2	fc:$400N$, ReLU, Dropout Rate: 0.2
Loss Layer	MSE Loss [2]

Once the proposed architecture was trained for the two tasks using the data described in Section 6.2.1, the network was then tested by feeding similar time-domain inputs as the training database examples and observing the outputs in both the time and frequency domain. For the autoencoder task, a qualitative visual metric was used to evaluate its performance by viewing how well the network's output resembled the truth signal. For the spectral notching task, the notch level of the neural network output will be used to discern the performance of the network. The last metric used is the timing of the neural network performance in terms of its execution timing statistics in order to verify that the network is producing results consistently and quickly.

6.3 Results

6.3.1 Preliminary Feasibility Results

The two preliminary tests that were conducted were an auto encoder and notching task. For both of these tasks, the results for both network types are shown. Three plots are shown for each task-network pairing in which we show the time domain, envelope and frequency domain plots. The results are shown visually in Figures 6.4-6.7.

160
Approved for public release; distribution is unlimited.

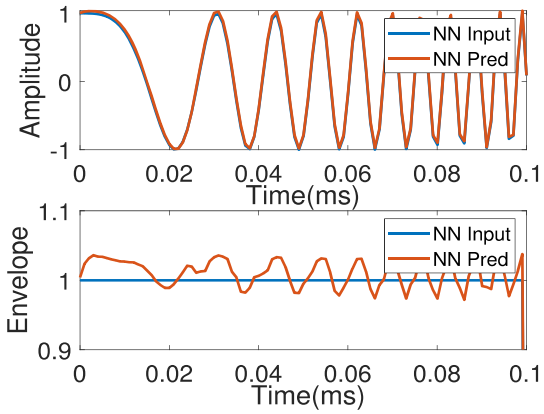


Figure 6.4: Output of LSTM Time Domain Response
Autoencoder Task

Auto-Encoder Task

For the first task, we have the network performing autoencoding. Figures 6.4 - 6.5 show the results for the LSTM in autoencoding. In viewing the LSTM, we see that it obtains reasonable performance in the time domain by aligning well with the truth waveform. In addition to this, it also obtains a comparable envelope to the truth waveform with slight oscillations. When looking at the frequency domain plot in Figure 6.5, we see that again the network aligns well with the truth. The LSTM frequency domain spectra also remains at relatively the same power level and frequency alignment as the truth spectra.

The network was evaluated over a test dataset of 5000 LFM waveforms made up of a 50 uniformly distributed transmit center frequencies with 100 uniformly distributed τ values. These waveforms were a set of different transmit waveforms in which the transmit center frequency varied uniformly between 0-50 kHz and the τ values were varied from 0.1 - 1 ms. This resulted in each selected transmit center frequency having

Approved for public release; distribution is unlimited.

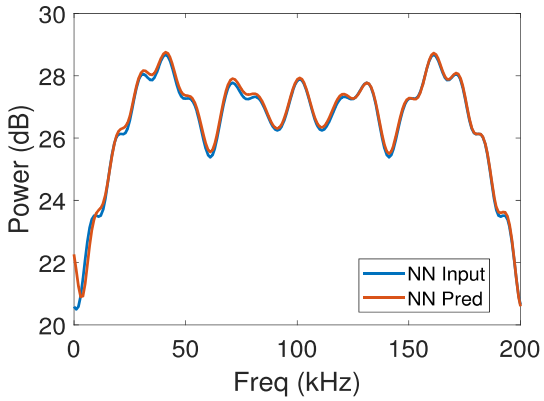


Figure 6.5: Output of LSTM Freq. Domain Response Autoencoder Task

100 varied τ values, which lead to a total of 5000 "feature" LFM waveforms. The results shown in Table 6.3 were a result of computing the relative error for each of the 5000 different cases through which the max and the average were obtained. The relative error is computed using the following equation:

$$\boldsymbol{\sigma}_{Rel,F} = \frac{\left\| \left| \boldsymbol{Y}_{NN}(\boldsymbol{f}) \right| - \left| \boldsymbol{Y}_{truth}(\boldsymbol{f}) \right| \right\|}{\| \boldsymbol{Y}_{truth}(\boldsymbol{f}) \|} \tag{6.5}$$

$$\boldsymbol{\sigma}_{Rel,t} = \frac{\left\| \left| \boldsymbol{y}_{NN}(\boldsymbol{t}) \right| - \left| \boldsymbol{y}_{truth}(\boldsymbol{t}) \right| \right\|}{\| \boldsymbol{y}_{truth}(\boldsymbol{t}) \|} \tag{6.6}$$

$$\tag{6.7}$$

where $\boldsymbol{\sigma}_{Rel,F}$ and $\boldsymbol{\sigma}_{Rel,t}$ are the relative errors for the frequency domain magnitude ($|\boldsymbol{Y}(\boldsymbol{f})|$) and time domain envelope ($\left|\boldsymbol{y}(\boldsymbol{t})\right|$) between the neural network output and the manually generated truth, respectively; and $\|\cdot\|$ is the ℓ_2/Euclidean norm.

Approved for public release; distribution is unlimited.

Table 6.3: AutoEncoder Results on Test Dataset

Performance Metric	$\sigma_{Rel,t} * 100$	$\sigma_{Rel,F} * 100$
Max Relative Error (%)	5.1	5.48
Average Relative Error (%)	2.4	2.5

Notched Waveform Task

For the second task, the network performs spectral notching based on interference presence. Figures 6.6-6.7 show the results for the LSTM in spectral notching. For this task, we have similar plots from the autoencoding tasks with the time domain, envelope and frequency domain plots. In the spectral notching tasks, the plot of most concern is Figure 6.7, where we see the performance of the networks in generating a notch for a given interference presence. The notches generated for the LSTM in Figure 6.7 are low (interference center: ≈15 dB below passband, interference band: ≈8 dB) but the magnitude of the frequency components outside the band are tampered and distorted.

Similar to the autoencoder task, the notched waveform task was evaluated over a test dataset of 5000 LFM waveforms made up of a 50 uniformly distributed interference center frequencies with 100 uniformly distributed τ values and a fixed center frequency for the unperturbed transmit signal at baseband. These waveforms were a set of different interference perturbed waveforms in which the interference center frequency varied uniformly between 50-100 kHz, the interference pulse width (τ) values were varied from 0.1 - 1 ms, the transmit center frequency varied uniformly between 0-50 kHz, and the transmit pulse width (τ) values were varied from 0.1 - 1 ms. This lead to a total of 5000 test LFM waveforms. The results shown in Table

Approved for public release; distribution is unlimited.

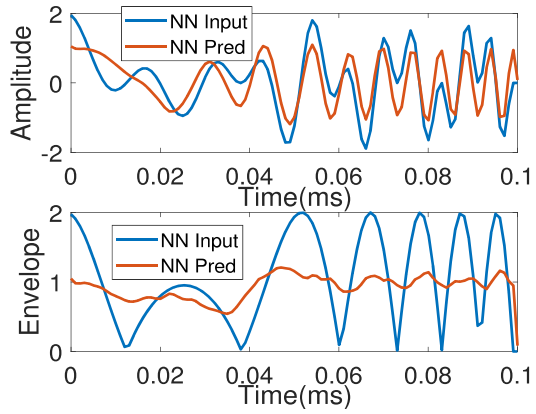


Figure 6.6: Output of LSTM Time Domain Response Notched Waveform Task

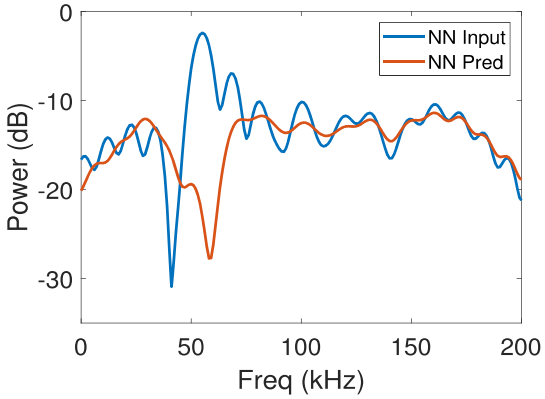


Figure 6.7: Output of LSTM Freq. Domain Response Notched Waveform Task

Approved for public release; distribution is unlimited.

Table 6.4: Notched Waveform Results on Test Dataset

Performance Metric	$\sigma_{Rel,t} * 100$	$\sigma_{Rel,F} * 100$
Max Relative Error (%)	97.74	61
Average Relative Error (%)	66.57	36

6.4 were a result of computing the relative error for each of the 5000 different cases through which the max and the average were obtained. In addition to this, a notch depth metric was also obtained for the notched waveform case, over all 5000 different cases. The notch depth was evaluated by evaluating the average power level at the notch location in frequency, $\bar{\boldsymbol{S}}(\boldsymbol{f}_{notch})$, where $\bar{\boldsymbol{S}}$ is the average normalized power in the frequency domain. Through use of the normalized frequency domain power as a reference, an average notch depth of ~ 2.7 dB was obtained by the neural network versus the average notch depth of the manually generated truth at ~ 200 dB. From this,we see that while the network was able to generate a notch at the interference location its depth and similarity to the truth was variable.

The results of the network should be noted as being done with slight tuning to the LSTM hyper-parameters. With more tuning in the hyper parameters, a better performance for the LSTM's could be obtained however it should be noted that with minimal hyper parameter tuning the LSTM was able to perform within an acceptable tolerance in both tasks. The LSTM network was also able to produce these outputs with an average time of 0.2 seconds, a time variance of $4.2 \cdot 10^{-4}$ seconds and a max time of 0.8 seconds (due to initialization) when run 1000 times. The self-timing of the function used by MATLAB to produce the result from the network given an input was 0.009 seconds which indicates that the main function used to produce the output

Approved for public release; distribution is unlimited.

executes quickly but the children functions used within it are slow. However, when executed in a GPU environment there will be significant improvement to this timing. Overall, the timing results show that the neural network is capable of completing the tasks with a consistently low time after being initialized.

Through the autoencoder and notched waveform tasks, we see that the neural network is capable of reproducing a waveform. However, in the notched waveform task, it is noted that while the network has the ability to generate notches, the NN generated notches compared to the database truth have large variance. The large variance between the notch depth of the neural network and the truth is likely due to the fact that a single neural network was trained on generating both the real and imaginary time domain components. This large domain space may have been too difficult for the network to accurately approximate given the relatively small amount of training data samples, as well as the fact that differences in both the real and imaginary components between the neural network can lead to larger differences in their frequency domain components. Due to this, a reduction to the complexity of the problem approached by the neural network was needed. One consideration would be use of a complex neural network, however in this work we choose to change the approach and make use of algorithmic based phase reconstruction for notched waveform design, particularly the ERA.

6.3.2 ERA-NN Results

Error Reduction Algorithm (ERA)

The error reduction algorithm, also referred to as the Gerchberg-Saxton algorithm [96, 104] is a phase retrieval algorithm which uses a four-step process to find solutions to the phase problem. The ERA can be used for solving for a complex signal, $\boldsymbol{q}$,

Approved for public release; distribution is unlimited.

with measurements of $|\tilde{\boldsymbol{q}}|$. In this section, the algorithm description from [105] is summarized.

To begin the algorithm, we start with a constant modulus complex-valued waveform,

$$\boldsymbol{s} = A\exp(j\boldsymbol{\phi}) \tag{6.8}$$

where A is a constant amplitude scalar value and $\boldsymbol{\phi}$ is the vector of phase components.

After which, the Fourier transform, $\mathcal{F}$, of $\boldsymbol{s}$ is taken using the DFT or *fast Fourier transform* (FFT),

$$\tilde{\boldsymbol{s}} = \mathcal{F}(\boldsymbol{s}) \tag{6.9}$$

Then, the required spectral mask, $|\tilde{\boldsymbol{q}}|$, is specified.

At this point, a four-step process is evaluated recursively till convergence,

Step 1

$$\tilde{\boldsymbol{v}} = |\tilde{\boldsymbol{q}}| \odot \exp\left(j\angle\tilde{\boldsymbol{s}}\right) \tag{6.10}$$

where $\angle$ is the angle operator and $\odot$ is the Hadamard/element-wise product.

Step 2

$$\boldsymbol{v} = \mathcal{F}^{-1}\left(\tilde{\boldsymbol{v}}\right) \tag{6.11}$$

where $\mathcal{F}^{-1}$ is the inverse Fourier transform.

Step 3

$$\boldsymbol{s} = \exp\left(j\angle\boldsymbol{v}\right) \tag{6.12}$$

Step 4

$$\tilde{\boldsymbol{s}} = \mathcal{F}\left(\boldsymbol{s}\right) \tag{6.13}$$

Approved for public release; distribution is unlimited.

The process terminates and reaches convergence when the difference between the squared errors of the magnitude of the designed waveform ($|\tilde{\boldsymbol{s}}|$) and the specification ($|\tilde{\boldsymbol{q}}|$), in two successive passes through the four-step process, is within 10^{-6}.

A comparison was made using the same neural architecture as in Section 6.2.1 for training but changing the input to the neural network from the complex time domain interference perturbed signal to the interference frequency mask and making the output of the network the phase components of the transmit waveform, resulting in a phase-only constant modulus waveform design for simplification. A graphical representation of the process is displayed in Figure 6.8.

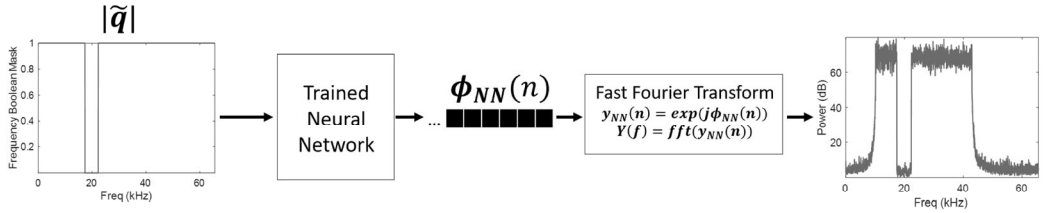


Figure 6.8: Output of LSTM Freq. Domain Response
Notched Waveform Task

The waveform parameters used to generate the ERA solutions are shown in Table 6.5, where the center frequency of the interference changes as a result of a dynamic interference presence:

ERA Result Comparison

Taking on notation defined in Section 6.3.2, we wish to take the desired spectral mask defined as $|\tilde{\boldsymbol{q}}|$, which is a vector of ones and zeros, where the zeros indicate the locations of the interference signal in the spectral domain and the ones indicate

Approved for public release; distribution is unlimited.

Table 6.5: Waveform Parameters

Parameters	TX	Interferer
τ	10 ms	10 ms
B	27.2 kHz	4.2 kHz
f_0	Baseband (0Hz)	0-16.9 kHz

the rest of the band. With this desired spectral mask defined with the nulls at the interference spectral locations, the problem then becomes a signal reconstruction problem where the phase of the time domain waveform which results in the desired spectral mask in the frequency domain is estimated. Given this, the focus is on estimating the time domain phase components, $\boldsymbol{\phi}_{notch}(n)$, for discrete indices n, where the transmit signal created by the neural network can be written as follows:

$$\boldsymbol{y}_{NN}(n) = A\exp(j\boldsymbol{\phi}_{NN}(n)). \tag{6.14}$$

where $\boldsymbol{\phi}_{NN}(n)$ is the neural network estimated phase components and $\boldsymbol{y}_{NN}$ is the complex time-domain neural network waveform

and

$$\boldsymbol{Y}_{NN}(F) = |\boldsymbol{\mathcal{F}}(\boldsymbol{y}_{NN}(n))| = |\tilde{\boldsymbol{q}}| \tag{6.15}$$

where $|\cdot|$ is the Euclidean norm (magnitude), $\boldsymbol{\mathcal{F}}$ is the Fourier transform operation, and $\boldsymbol{Y}_{NN}(F)$ is the neural network discrete frequency domain signal.

The results for the neural network waveform generated using the interference binary mask as an input with a dynamic interference, is shown in Figures 6.9,6.10, 6.11, and 6.12. In these results, we see that when we hold the constant modulus

Approved for public release; distribution is unlimited.

constraint when creating the "truth" signal waveforms using the ERA algorithm, we are able to generate constant envelope notched waveforms that are almost identical to the training/testing database.

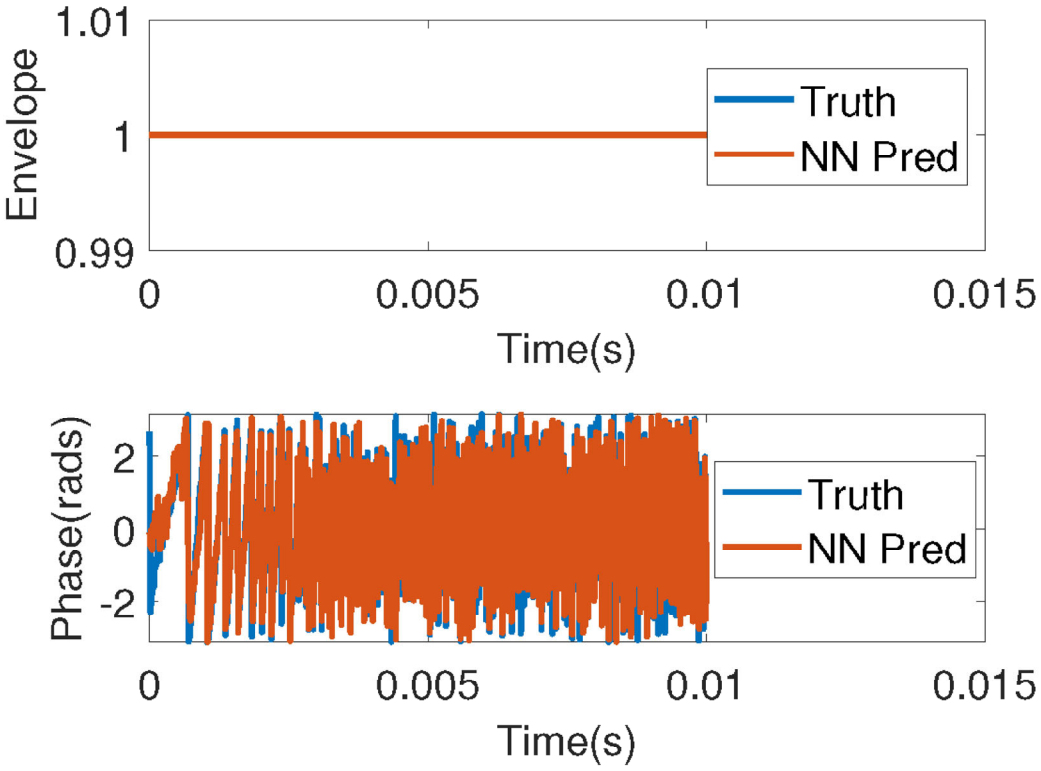


Figure 6.9: Output of LSTM Time Domain Response for Dynamic Interference at Location 1
ERA Comparison

Approved for public release; distribution is unlimited.

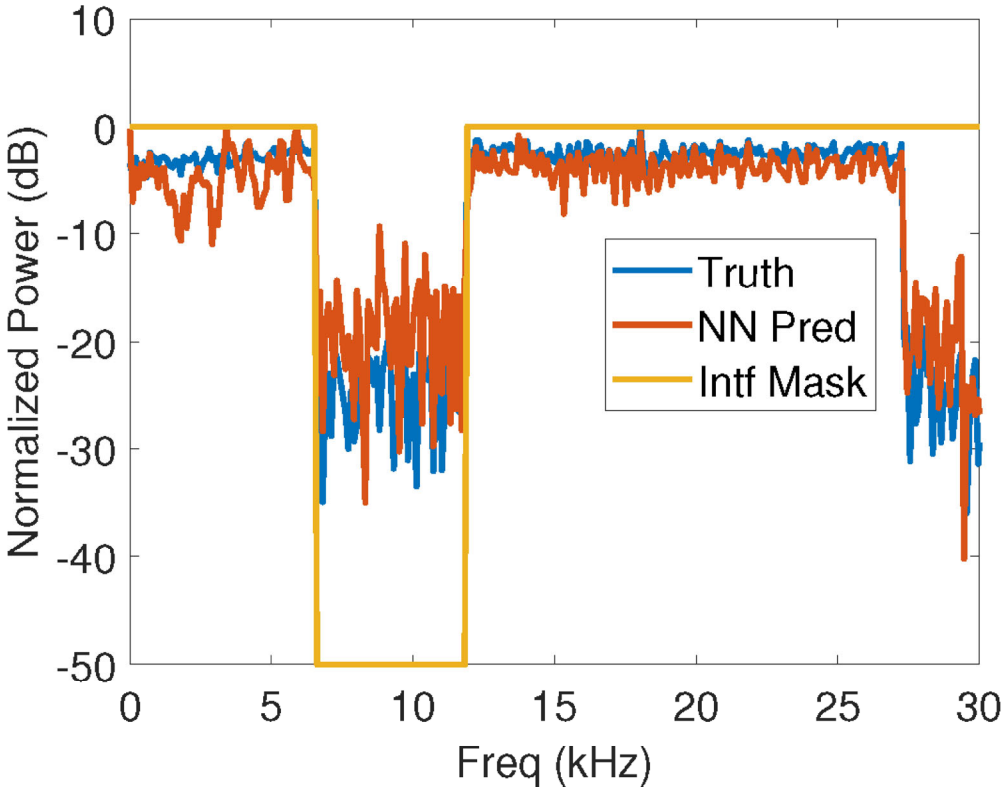


Figure 6.10: Output of LSTM Freq. Domain Response for Dynamic Interference at Location 1
ERA Comparison

Approved for public release; distribution is unlimited.

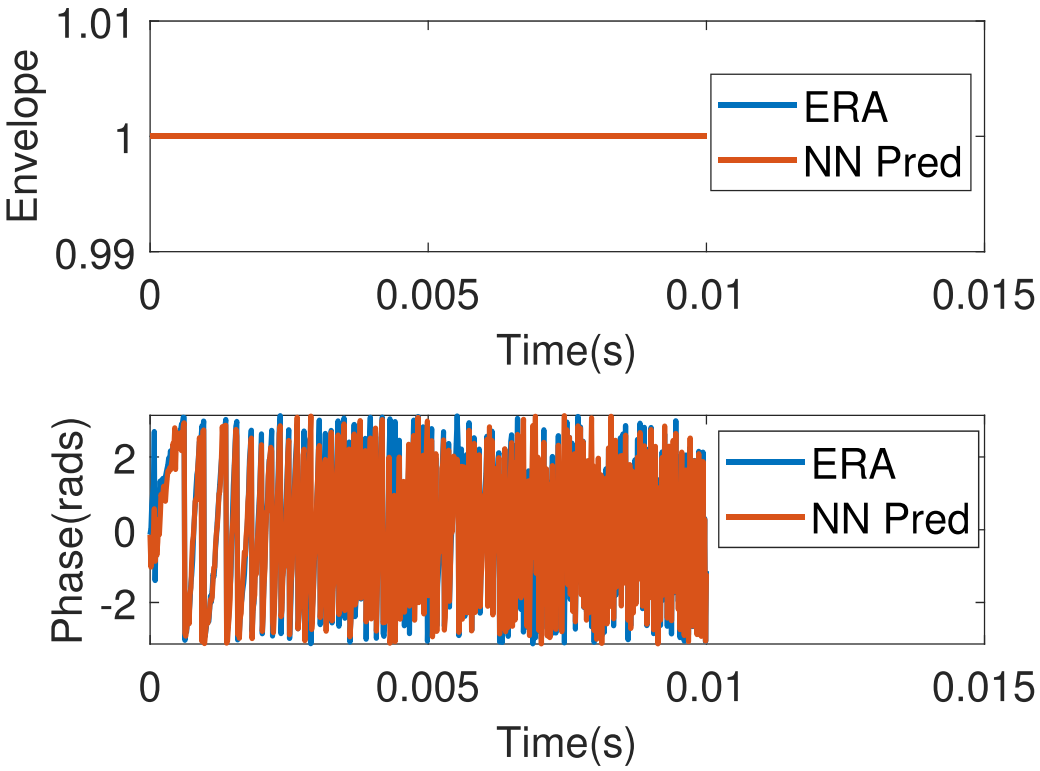


Figure 6.11: Output of LSTM Time Domain Response for Dynamic Interference at Location 2
ERA Comparison

Approved for public release; distribution is unlimited.

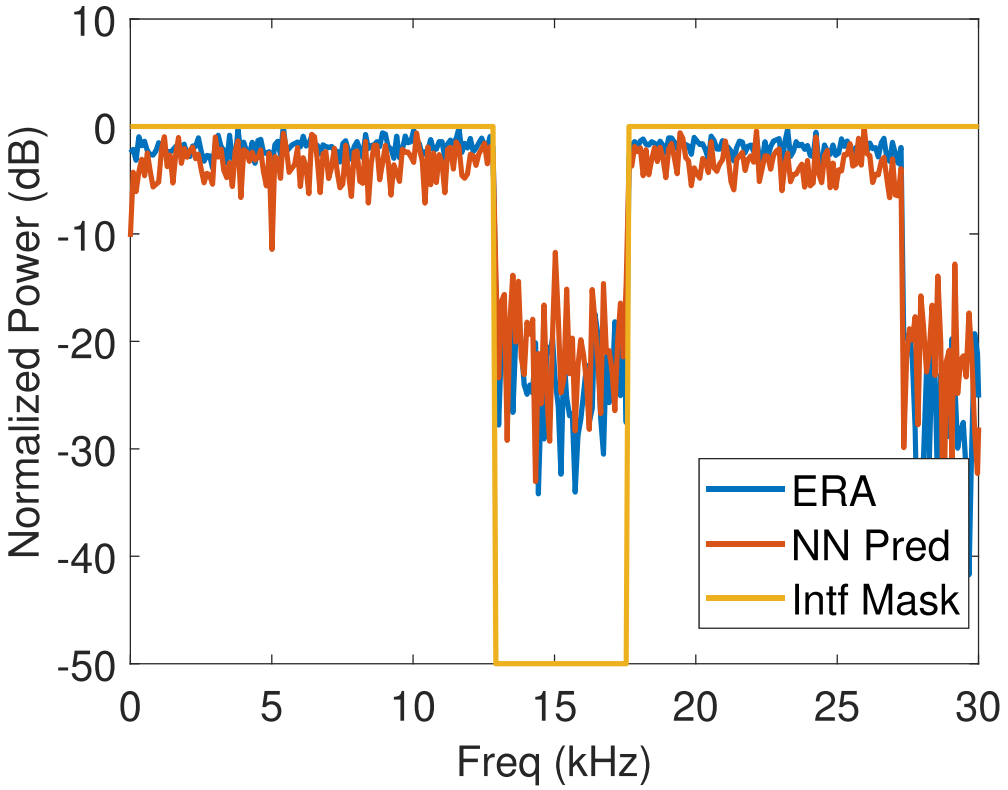


Figure 6.12: Output of LSTM Freq. Domain Response for Dynamic Interference at Location 2
ERA Comparison

For the results generated using the ERA solutions, 200 different waveforms were produced using the waveform parameters specified in Table 6.5 for training/testing. The 200 waveforms were of a dynamic interference set, through which each waveform represents the interference signal at a particular position in the spectral domain according to its varying center frequency. The results shown in Table 6.6 were a result of computing the relative error for each of the 200 different cases through which the max and the average were obtained. In addition to this, a notch depth metric was also obtained for the notched waveform case, over all 200 different cases. An average

Approved for public release; distribution is unlimited.

Table 6.6: Notched Waveform Results on Test Dataset

Performance Metric	$\sigma_{Rel,t} * 100$	$\sigma_{Rel,F} * 100$
Max Relative Error (%)	66	38.3
Average Relative Error (%)	50	21

notch depth of $\sim$ 18.6 dB was obtained by the neural network versus the average notch depth of the ERA generated truth at $\sim$ 23.7 dB. From this,we see that when the notched waveform generation problem was reduced to simpler components, the network was able to generate a notch at the interference location similar to that of the ERA truth. From Table 6.6, we see that the relative error between the neural network and the ERA in frequency domain is high but this is most likely due to the high variation of the spectral content of the signal as seen in Figure 6.10. Most noticeably, we see that the network is capable of producing an average notch depth close to that of the ERA.

Additionally, timing of the ERA algorithm versus the neural network was performed in MATLAB using the `timeit()` function. The `timeit()` function calls the specified function multiple times, and returns the median of the measurements based on wall-clock time. The LSTM execution timing using `timeit()` was found to be 0.052 seconds compared to the 1.18 seconds using the ERA. This means that in a median sense, the LSTM execution time was $\sim$22 times faster than the ERA algorithm. Additionally as the signal sample length increases, the ERA algorithm will require more time, whereas the LSTM-RNN (with the same architecture) will give the same timing.

Approved for public release; distribution is unlimited.

6.4 Conclusion

This section has introduced a neural network based waveform design technique demonstrating auto-encoding capability and interference notching. In addition to this, this section proposes a novel use of LSTM's for the task of waveform design and provides initial results on the feasibility of this through completing an autoencoder and notch waveform design. The trained LSTM's were shown to give acceptable performance for waveform design given minimally tuned networks implying that the capabilities of the network can be optimized with further tuning. Using these tests, it can be seen that LSTM networks prove to be an acceptable method for waveform generation given a finite solution space. While further performance comparisons between the LSTM and existing optimization techniques in spectral notching need to be considered, such as looking at relaxing the constant modulus constraint through use of the PAPR, a method utilizing LSTM networks for waveform design, given truth data designed by any optimization technique for real-time implementation, can begin to become conceivable.

Approved for public release; distribution is unlimited.

Chapter 7: Fully Adaptive Radar for Multiple Target Tracking

Separate from the work discussed in prior chapters, this chapter and the following chapter do not use neural network implementations and are instead focused on advancements made to FAR concepts. This chapter discusses the development of a fully adaptive radar multiple target tracking (FAR-MTT) framework. The framework was tested using the existing FARMS environment. The FAR-MTT framework uses the FAR model for adaptation of the radar waveform parameters in order to optimize track performance in the presence of multiple performance objectives that arise in a multiple target environment. The *Fisher Information Matrix* (FIM) is used for track performance evaluation as well as the cost functions for the MTT case are derived. Previously, the FAR framework has only been applied to single-target tracking scenarios and this work focuses on expanding this framework to a multiple target environment. We will show that the FAR-MTT framework is capable of obtaining low track errors in range and velocity with minimized sensor resource usage all while ensuring target separation in comparison to the fixed parameter case in different multiple target environments.

The remainder of this chapter is organized as follows. Section 7.1 introduces the problem of multiple target tracking and the motivation behind using fully adaptive

Approved for public release; distribution is unlimited.

methods. Section 7.2 reviews the approach for multiple target tracking through discussion of the multiple target model; the theoretical formulations and experimental simplifications for the FAR-MTT recursion, and the new optimization method implemented at the center of the fully adaptive radar framework. Section 7.3 presents and discusses the simulation and experimental results. Concluding remarks on the work are then made in Section 7.4.

7.1 Motivation

Multiple Target Tracking (MTT) is an area of significant research in the radar community because most radar scenes contain multiple targets. Compared to *Single Target Tracking* (STT), MTT has greater complexity as it requires a data association step to assign detections to tracks. The method implemented in this work makes use of the MAP-PF to combine the detection and association steps. The MAP-PF approach jointly performs detection and target state estimation required to obtain target measurements. The MAP-PF method approaches MTT as a target state point estimation problem and the solution is a two-stage estimation process where the penalty function guided detection supplies the measurements to a tracker algorithm for target estimation. First, we extract one measurement for each target from sensor data using a penalty function guided detection; and then we estimate the target state using Bayesian filtering. The MAP-PF method relies on the estimation of the PIM or *Predicted Cramer-Rao Bound* (P-CRB) to implement the penalty portion of the MAP-PF method, in which the PIM/P-CRB controls the strength of the penalty applied in order to perform the target detection and association. In addition to target detection, track estimation is also of concern for MTT because of the risk of

Approved for public release; distribution is unlimited.

target merging from lack of adequate target separation in the measurement space. The FAR framework implements an adaptive algorithm that uses measurement and performance cost functions to perform an executive optimization for sensor parameter selection in order to obtain a better tracking performance [4]. Instantiations of the FAR framework from [4] thus far have only been applied to single-target scenarios, however here its adaptive nature will be demonstrated useful for multiple-target environments also.

Applications of adaptive radar in a multiple target environment that have been investigated include: using an adaptive alpha-beta tracking algorithm for multiple target tracking in a multi-radar environment [106], tracking multiple targets using a particle filter with a resource constrained sensor network by assigning resources to select sensors through minimization of the *Predicted Conditional Cramer-Rao Lower Bound* (PC-CRLB) [107], using adaptive waveform scheduling with look-ahead in a multi-sensor, multipath environment by minimizing the track error covariance to select the optimal waveform for multi-target tracking [108], multi-sensor scheduling for multiple target tracking using a partially observable Markov decision process [109], and using a cognitive radar network to perform adaptive sensor scheduling and power allocation for tracking multiple targets in a multipath environment by minimizing the *Posterior Cramer-Rao Bound* (PCRB) [110].

The central theme of many of these works lies in the derivation and analysis of multi-sensor radar environments for tracking multiple targets through minimization of the Cramer-Rao bound. These works also focus on sensor scheduling and resource allocation approaches to MTT and are mostly realized in numerical simulation. This

Approved for public release; distribution is unlimited.

paper focuses on developing a robust adaptive framework for multi-target tracking that can be used in both simulations and experiments.

This work focuses on the development of a FAR-MTT and demonstrates its benefit over static parameter selections. To achieve this, a new optimization for the FAR framework executive processor is derived in addition to a revised FIM implementation. In applying FAR concepts from the STT problem [4] to MTT, concerns were focused on developing a robust but straightforward optimization scheme/objective function. As with the single target FAR framework FAR-STT, concerns of achieving acceptable target track performance were noted. In addition to this, the development of the *Bayesian Information Matrix* (BIM)/*Cramer-Rao Bound* (CRB) expression for multiple targets needed to be considered to capture the effects of other targets when estimating an individual target's state. A simulation and experimental example is presented to demonstrate the performance of this approach.

7.2 Approach

In order to investigate FAR-MTT, some base assumptions were established to simplify the problem to facilitate experimental validation and comparative analysis. The assumptions are as follows:

1. a *Nearly Constant Velocity* (NCV) model is used for the target dynamics of all targets;

2. number of targets is assumed to be known a priori;

3. number of targets does not change over scan times (no appearance of new targets or disappearance of existing targets);

Approved for public release; distribution is unlimited.

4. multiple target likelihood functions at distinct times are independent given the state;

5. the Markov motion processes for the individual targets are mutually independent;

6. the system is linear and Gaussian (Kalman filter can be implemented);

7. target states to be tracked are range r, range rate $\dot{r}$ (velocity), and SNR χ;

8. A LFM is used for environment sensing with adapted waveform parameters: B, f_{PRF}, τ, and N_{p};

9. a single "monostatic" sensor is used for measurements (single transmit and single receive).

7.2.1 Multiple Target Tracking (MTT) Model

For FAR-MTT, we develop a generic RD radar sensor model for multiple targets plus clutter and noise. The target kinematic parameters that we want to estimate and track are the two-dimensional range and velocity, where $r_{m,k}$ and $\dot{r}_{m,k}$ denote the range and velocity of the m^{th} target at time t_k, respectively. Using this, we define the kinematic parameter vector, $\boldsymbol{\rho}_{m,k}$ as:

$$\boldsymbol{\rho}_{m,k} = [r_{m,k} \;\; \dot{r}_{m,k}]^T \tag{7.1}$$

The radar receiver observes the target using the natural radar parameters of range and Doppler frequency. We denote $r_{m,k}$ and $f_{m,k}$ as the range and Doppler of the m^{th} target at time t_k, respectively. Using this, we define the natural radar parameter vector, $\boldsymbol{\eta}_{m,k}$ as:

Approved for public release; distribution is unlimited.

$$\boldsymbol{\eta}_{m,k} = [r_{m,k} \;\; f_{m,k}]^T \tag{7.2}$$

The natural radar parameters are related to the kinematic parameters by the following transformation:

$$\boldsymbol{\eta}_{m,k} = \boldsymbol{\zeta}_t(\boldsymbol{\rho}_{m,k}) = \begin{bmatrix} 1 & 0 \\ 0 & \frac{-2f_c}{c} \end{bmatrix} \boldsymbol{\rho}_{m,k} \tag{7.3}$$

We assume the observed data is complex fast-time radar samples. Let f_c denote the radar operating frequency, c denote the speed of propagation, f_s denote the fast-time sample rate, and $t_s = \frac{1}{f_s}$ denote the sampling interval. Let T_p denote the length of the PRI and the f_{PRF} be denoted as $f_{PRF} = \frac{1}{T_p}$. There are $N_r = \lfloor T_p f_s \rfloor$ fast-time samples per PRI, and N_p pulses per coherent processing interval CPI, where $\lfloor \cdot \rfloor$ reflect the floor function. The waveform has pulse length τ and B.

Let $w(t)$ denote the continuous-time waveform defined on $0 \leq t \leq \tau$. There are $N_w = \lfloor \tau f_s \rfloor + 1$ fast-time waveform samples per pulse. The unambiguous range is:

$$R_{unamb} = \frac{cT_p}{2} = \frac{c}{2f_{PRF}} \tag{7.4}$$

Let R_{max} denote the maximum range. Let N_s denote the maximum number of fast time samples that we need to collect and process. The number of fast time samples has the following form:

$$N_s = \lfloor (\frac{(2R_{max})}{c} + \tau) f_s \rfloor + 1 \tag{7.5}$$

We define $\mathbf{z}_k$ to be the $N_s N_p \times 1$ observation vector obtained by stacking the fast-time samples in the k^{th} CPI across PRI's and pulses. Let $\boldsymbol{\theta}_k$ denote the sensor

Approved for public release; distribution is unlimited.

parameters that we can control, which will be the following parameters f_{PRF}, N_{p}, τ, and B. Let $\boldsymbol{\nu}(\boldsymbol{\eta}_{m,k};\boldsymbol{\theta}_k)$ be the $N_sN_p \times 1$ radar response vector for radar parameters: $r_{m,k}$ and $f_{m,k}$. The radar response vector has the form:

$$\boldsymbol{\nu}(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k) = u(f_{m,k};\boldsymbol{\theta}_k) \otimes w(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k) \tag{7.6}$$

where $\boldsymbol{w}(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k)$ is the $N_s \times 1$ waveform vector of fast time samples in a PRI, $u(f_{m,k};\boldsymbol{\theta}_k)$ is the $N_p \times 1$ Doppler response vector and $\otimes$ is the Kronecker product. Let $T_{m,k} = \frac{2r_{m,k}}{c}$ denote the round-trip travel time for a range of $r_{m,k}$ and $N_{m,k} = \lfloor T_{m,k}f_s \rfloor$ denote the corresponding number of fast time samples. The waveform vector has the form:

$$\boldsymbol{w}(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k) = \begin{bmatrix} \mathbf{0}_{N_{m,k}} \\ w(N_{m,k}t_s - T_{m,k})e^{j2\pi f_{m,k}N_{m,k}t_s} \\ \vdots \\ w((N_{m,k}+N_w-1)t_s - T_{m,k})e^{j2\pi f_{m,k}(N_{m,k}+N_w-1)t_s} \\ \mathbf{0}_{N_s-N_w-N_{m,k}} \end{bmatrix} \tag{7.7}$$

and the Doppler response vector has the form:

$$\boldsymbol{u}(f_{m,k};\boldsymbol{\theta}_k) = \begin{bmatrix} 1 & e^{j2\pi T_p f_{m,k}} & \cdots & e^{j2\pi(N_p-1)T_p f_{m,k}} \end{bmatrix}^T \tag{7.8}$$

Let $\alpha_{m,k}$ denote the complex scattering response of the m^{th} target at time t_k, and let $\boldsymbol{n}_k$ denote a random realization of the clutter and noise at time t_k. Assuming there are M targets, the data vector $\mathbf{z}_k$ has the form:

$$\mathbf{z}_k = \sum_{m=1}^{M} \alpha_{m,k}\boldsymbol{\nu}(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k) + \boldsymbol{n}_k \tag{7.9}$$

We assume that the $\boldsymbol{n}_k$ are independent samples from a zero-mean random process with positive definite covariance matrix $\boldsymbol{R}_n(\boldsymbol{\theta}_k) \sim \sigma_w^2\boldsymbol{I}$, which depends on the sensor

Approved for public release; distribution is unlimited.

parameters and where $\boldsymbol{I}$ is the $N_s N_p \times N_s N_p$ identity matrix. We assume that the target responses $\alpha_{m,k}$ are independent across dwells, with zero mean and variance $\gamma_{m,k}$, and are independent from the clutter and noise.

We define the following $M \times 1$ vectors:

$$\boldsymbol{r}_k = [r_{1,k} \cdots r_{M,k}]^T \tag{7.10}$$

$$\boldsymbol{f}_k = [f_{1,k} \cdots f_{M,k}]^T \tag{7.11}$$

$$\boldsymbol{\gamma}_k = [\gamma_{1,k} \cdots \gamma_{M,k}]^T \tag{7.12}$$

the $M \times M$ matrix

$$\boldsymbol{\Gamma}_k = diag(\boldsymbol{\gamma}_k) \tag{7.13}$$

and the $N_s N_p \times M$ matrix

$$\boldsymbol{V}(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = [\boldsymbol{\nu}(r_{1,k}, f_{1,k}; \boldsymbol{\theta}_k) \cdots \boldsymbol{\nu}(r_{M,k}, f_{M,k}; \boldsymbol{\theta}_k)] \tag{7.14}$$

$$\boldsymbol{R}_{\mathbf{z_k}}(\boldsymbol{r}_k, \boldsymbol{f}_k, \boldsymbol{\gamma}_k; \boldsymbol{\theta}_k) = \sum_{i=1}^{M} \gamma_{m,k} \boldsymbol{\nu}(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k) \boldsymbol{\nu}(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k)^H + \boldsymbol{R}_n(\boldsymbol{\theta}_k) \tag{7.15}$$

which can be written as:

$$\boldsymbol{R}_{\mathbf{z_k}}(\boldsymbol{r}_k, \boldsymbol{f}_k, \boldsymbol{\gamma}_k; \boldsymbol{\theta}_k) = \boldsymbol{V}(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) \Gamma_k \boldsymbol{V}(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k)^H + \sigma_w^2 \boldsymbol{I} \tag{7.16}$$

Assuming that the data samples have a Gaussian distribution, then the pdf of the received data is completely characterized by this covariance matrix. The covariance matrix in (7.16) is a function of the range vector $\boldsymbol{r}_k$, the Doppler vector $\boldsymbol{f}_k$, the signal powers $\boldsymbol{\gamma}_k$, the white noise power σ_w^2, and parameterized by the sensor parameter vector $\boldsymbol{\theta}_k$.

Approved for public release; distribution is unlimited.

Define the post-processing SNR $\chi_{m,k}$ as:

$$\chi_{m,k} = \frac{N_p N_w \gamma_{m,k}}{\sigma_w^2} \tag{7.17}$$

We define the target state vector as the kinematic parameters in $\boldsymbol{\rho}_{m,k}$ augmented by the post-processing SNR

$$\mathbf{x}_{m,k} = \left[\boldsymbol{\rho}_{m,k}^T \;\; \chi_{m,k}\right]^T \tag{7.18}$$

and we define the target natural parameter vector as:

$$\boldsymbol{y}_{m,k} = \left[\boldsymbol{\eta}_{m,k}^T \;\; \gamma_{m,k}\right]^T \tag{7.19}$$

The natural parameter vector is related to the target state vector by the mapping

$$\boldsymbol{H}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{-2f_c}{c} & 0 \\ 0 & 0 & \frac{\sigma_w^2}{N_p N_w} \end{bmatrix} \tag{7.20}$$

$$\boldsymbol{y}_{m,k} = \boldsymbol{H}_t(\mathbf{x}_{m,k}) = \left[\boldsymbol{\zeta}_t(\boldsymbol{\rho}_{m,k}^T) \;\; \chi_{m,k} \frac{\sigma_w^2}{N_p N_w}\right]^T \tag{7.21}$$

We define the total target state and natural parameter vectors as:

$$\mathbf{x}_k = \left[\mathbf{x}_{1,k}^T \cdots \mathbf{x}_{M,k}^T\right]^T \tag{7.22}$$

$$\boldsymbol{y}_k = \left[\boldsymbol{y}_{1,k}^T \cdots \boldsymbol{y}_{M,k}^T\right]^T \tag{7.23}$$

The total natural parameter vector is related to the total target state vector by the mapping

$$\boldsymbol{y}_k = \boldsymbol{H}(\mathbf{x}_k) = \begin{bmatrix} \boldsymbol{H}_t & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{H}_t \end{bmatrix} \mathbf{x}_k = \boldsymbol{H}\mathbf{x}_k \tag{7.24}$$

Approved for public release; distribution is unlimited.

The likelihood function of the observation at time t_k is the complex Gaussian density:

$$f\left(\mathbf{z}_k|\boldsymbol{y}_k = \boldsymbol{h}(\mathrm{x}_k);\boldsymbol{\theta}_k\right) \sim \mathcal{CN}\left(\mathbf{0}, \boldsymbol{R}_{z_{\boldsymbol{k}}}(\boldsymbol{y}_{\boldsymbol{k}};\boldsymbol{\theta}_k)\right) \tag{7.25}$$

where $\mathbf{0}$ is a $N_s N_p \times 1$ zero vector.

Due to these assumptions made about the system, a Kalman filter as discussed in Chapter 2.1.2 can be implemented. In implementing the Kalman filter tracker, estimates of the target natural parameters, $\hat{\boldsymbol{y}}_k$, are substituted from the MAP-PF detection process discussed in Chapter 2.1.3. The PIM can now be stated as the inverse of the predicted error covariance, $\boldsymbol{P}_k^-$.

The PIM has the form:

$$\boldsymbol{U}_k^-(\boldsymbol{\theta}) = \left[\boldsymbol{P}_k^-\right]^{-1} \tag{7.26}$$

Target Motion Model

For this work, we use a linear Gauss-Markov motion model for the targets. The motion model consists of a prior density $q(\mathrm{x}_0)$ and a transition density $q(\mathrm{x}_k|\mathrm{x}_{k-1};\boldsymbol{\theta}_k)$. We assume an initial target state distribution of the form:

$$q(\mathbf{x}_0) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \tag{7.27}$$

Let $\Delta t(\boldsymbol{\theta}_k) = t_k - t_{k-1}$, through which parameter selection may affect t_k. For each target, we assume an AWGN NCV motion model of the form:

$$\mathbf{x}_{m,k} = \boldsymbol{F}_t(\boldsymbol{\theta}_k)\mathbf{x}_{m,k-1} + \boldsymbol{e}_{m,k}, \tag{7.28}$$

Approved for public release; distribution is unlimited.

where

$$\boldsymbol{F}_t(\boldsymbol{\theta}_k) = \begin{bmatrix} 1 & \Delta t(\boldsymbol{\theta}_k) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \beta(\boldsymbol{\theta}_k) \end{bmatrix} \tag{7.29}$$

$$\boldsymbol{\beta}(\boldsymbol{\theta}_k) = \frac{N_{p,k}}{N_{p,k-1}} \frac{\tau_k}{\tau_k - 1} \frac{P_{t,k}}{P_{t,k-1}}, \tag{7.30}$$

and $e_{m,k} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_t)$ is a zero-mean random vector with covariance matrix:

$$\boldsymbol{Q}_t(\boldsymbol{\theta}_k) = \begin{bmatrix} \sigma_r^2(\Delta t(\boldsymbol{\theta}_k)) & 0 & 0 \\ 0 & \sigma_v^2(\Delta t(\boldsymbol{\theta}_k)) & 0 \\ 0 & 0 & \sigma_s^2(\Delta t(\boldsymbol{\theta}_k)) \end{bmatrix} \tag{7.31}$$

We assume the motion models for the targets are independent, therefore the joint motion model has the form:

$$\mathbf{x}_k = \boldsymbol{F}(\boldsymbol{\theta}_k)\mathbf{x}_{k-1} + \boldsymbol{e}_k \tag{7.32}$$

where

$$\boldsymbol{e}_k = \left[\boldsymbol{e}_{1,k}^T \cdots \boldsymbol{e}_{M,k}^T\right]^T \tag{7.33}$$

$$\boldsymbol{F}(\boldsymbol{\theta}_k) = \begin{bmatrix} \boldsymbol{F}_t(\boldsymbol{\theta}_k) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & 0 \\ 0 & 0 & \boldsymbol{F}_t(\boldsymbol{\theta}_k) \end{bmatrix} \tag{7.34}$$

and $\boldsymbol{e}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$ is a zero-mean random vector with covariance matrix:

$$\boldsymbol{Q}(\boldsymbol{\theta}_k) = \begin{bmatrix} \boldsymbol{Q}_t(\boldsymbol{\theta}_k) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & 0 \\ 0 & 0 & \boldsymbol{Q}_t(\boldsymbol{\theta}_k) \end{bmatrix} \tag{7.35}$$

Thus,

$$q(\mathbf{x}_k|\mathbf{x}_{k-1}; \boldsymbol{\theta}_k) \sim \mathcal{N}(\boldsymbol{F}(\boldsymbol{\theta}_k)\mathbf{x}_{k-1}, \boldsymbol{Q}(\boldsymbol{\theta}_k)) \tag{7.36}$$

Approved for public release; distribution is unlimited.

7.2.2 Fisher Information Matrix

As mentioned in Section 7.2.1, the inverse of the FIM is used as the measure noise covariance, $\boldsymbol{R}_k$. In prior work [4, 71], the FIM was based on the single target case but in this work we develop and implement a multiple target FIM.

The FIM in [111] is developed for the parameter vectors $\boldsymbol{r}_k$, $\boldsymbol{f}_k$, and $\boldsymbol{\gamma}_k$, and has the form:

$$\boldsymbol{J}_{rf\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) = \begin{bmatrix} \boldsymbol{J}_{rr}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{rf}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{r\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \\ \boldsymbol{J}_{fr}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{ff}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{f\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \\ \boldsymbol{J}_{\gamma r}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{\gamma f}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{\gamma\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \end{bmatrix} \tag{7.37}$$

In this work the multiple target FIM is a $3M \times 3M$ matrix, based on the one developed in [111], which takes the following form:

$$\boldsymbol{J}_{\boldsymbol{y}}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) = \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{y}}^{11}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{\boldsymbol{y}}^{12}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \cdots & \boldsymbol{J}_{\boldsymbol{y}}^{1M}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \\ \boldsymbol{J}_{\boldsymbol{y}}^{21}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{\boldsymbol{y}}^{22}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \cdots & \boldsymbol{J}_{\boldsymbol{y}}^{2M}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{J}_{\boldsymbol{y}}^{M1}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{\boldsymbol{y}}^{M2}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \cdots & \boldsymbol{J}_{\boldsymbol{y}}^{MM}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \end{bmatrix} \tag{7.38}$$

where each block ($\boldsymbol{J}_{\boldsymbol{y}}^{\boldsymbol{ij}}$) is a 3×3 matrix with elements ordered by range, Doppler, and signal power for each of the M targets.

The required FIM $\boldsymbol{J}_{\boldsymbol{y}}(\boldsymbol{y}_k;\boldsymbol{\theta}_k)$ in (7.38) can be obtained from $\boldsymbol{J}_{rf\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k)$ in (7.37) by a linear transformation, $\boldsymbol{\Xi}$:

$$\boldsymbol{J}_{\boldsymbol{y}}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) = \boldsymbol{\Xi}\boldsymbol{J}_{rf\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \tag{7.39}$$

where $\boldsymbol{\Xi}$ is a linear transformation matrix that rearranges the elements of $\boldsymbol{J}_{rf\gamma}$ to the representation shown in (7.38). Further details of the FIM derivation can be found in Appendix B.

Approved for public release; distribution is unlimited.

Pre-Whitening Raw Data

In the optimization problem previously outlined in the background discussing the MAP-PF method in Chapter 2.58, the optimization was performed over a single target through considering the likelihood function for one target based on assuming the other targets' parameters fixed/known. This formulation also assumes that the penalty function is a sum of individual target penalty functions, which is true if $\boldsymbol{\Sigma}_k$ from (2.50) is a block diagonal matrix, following from the predicted error covariance $\boldsymbol{P}_k^-$ being a block diagonal matrix. Making these assumptions results in a "pre-whitened" detection surface in which the other targets are suppressed.

Using the simplification outlined in (2.58), this results in a whitened range/Doppler ($\boldsymbol{RD}_{\tilde{\mathbf{z}}_k}$) surface being computed for each target. Computing multiple range/Doppler surfaces would be computationally intensive, so examples of the whitened surface are limited to simulation.

Following the derivations from Appendix B, we obtain the whitening filter as:

$$\tilde{\boldsymbol{R}}_{\mathbf{z}}^{-1/2} = \sum_{i=1}^{M-1} (\lambda_i + \sigma_w^2)^{-1/2} - (\sigma_w^{-1/2})\boldsymbol{E_i}\boldsymbol{E_i}^H + \frac{1}{\sqrt{\sigma_w^2}}\boldsymbol{I} \tag{7.40}$$

where $\boldsymbol{E}$ and $\boldsymbol{\lambda}$ are the eigenvectors and eigenvalues, respectively, of the signals one wishes to suppress. See Appendix B.2 for further details.

The whitened range-Doppler surface can then be obtained at every coherent processing interval CPI using the whitened data vector:

$$\tilde{\mathbf{z}}_{\boldsymbol{k}} = \tilde{\boldsymbol{R}}_{\mathbf{z}}^{-1/2}\mathbf{z}_{\boldsymbol{k}} \tag{7.41}$$

where $\tilde{\boldsymbol{R}}_{\mathbf{z}}^{-1/2}$ is the whitening filter; and $\mathbf{z}_{\boldsymbol{k}}$ is the data vector.

Approved for public release; distribution is unlimited.

In the whitened data vector, the signal responses from the targets considered to be interference are suppressed.

Multi-Target Fisher Information Matrix Simplification for CREW Implementation

In order to compute the FIM expression in real time and gather experimental results, simplifications and approximations were made to the equations mentioned in Chapter 7.2.2 and further detailed in Appendix B.1. The FIM expressions implemented in the FAR-MTT algorithm take the following form:

$$\boldsymbol{S} \rightarrow \hat{\boldsymbol{S}} = \frac{1}{N_p N_w} diag(\hat{\boldsymbol{\chi}}_k) \tag{7.42}$$

$$\boldsymbol{A} \rightarrow \hat{\boldsymbol{A}} = [(\boldsymbol{V}^H \boldsymbol{V} + \hat{\boldsymbol{S}}^{-1})]^{-1} \tag{7.43}$$

$$\begin{aligned} \boldsymbol{J}_y^{rr} = C_R 2Re\{[\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_r] \odot [\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_r]^T + \\ [\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{V}\hat{\boldsymbol{S}}] \odot [\boldsymbol{D}_r^H\boldsymbol{D}_r - \boldsymbol{D}_r^H\boldsymbol{V}\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_r]^T\} \end{aligned} \tag{7.44}$$

$$\begin{aligned} \boldsymbol{J}_y^{ff} = C_F 2Re\{[\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_f] \odot [\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_f]^T + \\ [\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{V}\hat{\boldsymbol{S}}] \odot [\boldsymbol{D}_f^H\boldsymbol{D}_f - \boldsymbol{D}_f^H\boldsymbol{V}\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_f]^T\} \end{aligned} \tag{7.45}$$

$$\begin{aligned} \boldsymbol{J}_y^{rf} = \sqrt{C_F C_R} 2Re\{[\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_r] \odot [\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_f]^T + \\ [\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{V}\hat{\boldsymbol{S}}] \odot [\boldsymbol{D}_r^H\boldsymbol{D}_f - \boldsymbol{D}_r^H\boldsymbol{V}\hat{\boldsymbol{A}}\boldsymbol{V}^H\boldsymbol{D}_f]^T\} \end{aligned} \tag{7.46}$$

Following from the derivations in [111] and formulations in Appendix B.1, we assume that r and f are uncoupled from γ, resulting in their cross-terms being:

$$\boldsymbol{J}_y^{r\gamma} \approx 0 \tag{7.47}$$

$$\boldsymbol{J}_y^{f\gamma} \approx 0 \tag{7.48}$$

Approved for public release; distribution is unlimited.

$$\boldsymbol{J}_y^{\gamma\gamma} = C_\gamma \hat{\boldsymbol{S}}^{-1} \odot [\hat{\boldsymbol{S}}^{-1}]^T \tag{7.49}$$

where C_R, C_F, and C_γ are empirical constants; $Re\{\cdot\}$ is the operator used to find the real elements of a given input; $\hat{\boldsymbol{S}}$ is the SNR estimate; $\boldsymbol{D}_r$ is the range derivative of the radar response vector; and $\boldsymbol{D}_f$ is the Doppler derivative of the radar response vector.

For FAR-MTT, we will look at adapting the following sensor parameters for each k^{th} index of, $\theta_{k,i}$.

$$\boldsymbol{\theta_k} = [\theta_{k,1} \;\; \theta_{k,2} \;\; \theta_{k,3} \;\; \theta_{k,4}] \tag{7.50}$$

where $i = 1$ to 4 with 1 denoting pulse repetition frequency, f_{PRF}, 2 denoting number of pulses, N_{p}, 3 denoting the pulse-width, τ, and 4 denoting bandwidth, B.

Restating elements from Section 7.2.1, for each target, the following states , $\text{x}_{m,k}$, are estimated: range $r_{m,k}$; range rate (velocity) $\dot{r}_{m,k}$; and SNR $\chi_{m,k}$ for each CPI.

The estimated target state is represented by a concatenated vector, $\hat{\mathbf{x}}_k$, composed of $\{\hat{\mathbf{x}}_{1,k},\ \hat{\mathbf{x}}_{2,k},\ ...,\ \hat{\mathbf{x}}_{M,k}\}$, where M is the number of targets, of the following form shown in (7.22) and restated here:

$$\begin{aligned} \hat{\mathbf{x}}_{m,k} &= \left[\boldsymbol{\rho}_{m,k}^T \;\; \chi_{m,k}\right]^T \\ \hat{\mathbf{x}}_k &= \left[\hat{\mathbf{x}}_{1,k}^T \;\; \hat{\mathbf{x}}_{2,k}^T \;\; ... \;\; \hat{\mathbf{x}}_{M,k}^T\right]^T \end{aligned} \tag{7.51}$$

The motion model assumed for all the targets was a NCV model:

$$\hat{\mathbf{x}}_k = F(\boldsymbol{\theta}_k)\hat{\mathbf{x}}_{k-1} + \boldsymbol{e}_k \tag{7.52}$$

For each of the targets, the following natural parameter vector was captured, $\boldsymbol{y}_{m,k}$: range $r_{m,k}$; Doppler frequency $f_{m,k}$; and power $\gamma_{m,k}$. The estimated natural parameter

vector for all targets can be represented by a concatenated vector, $\boldsymbol{y}_k$ composed of $\{\hat{\boldsymbol{y}}_{1,k}, \hat{\boldsymbol{y}}_{2,k}, ..., \hat{\boldsymbol{y}}_{M,k}\}$, in terms of $\hat{\mathbf{x}}_k$, through the following form:

$$\hat{\boldsymbol{y}}_k = \boldsymbol{H}\hat{\mathbf{x}}_k + \boldsymbol{w}_k \tag{7.53}$$

where $\boldsymbol{w}_k$ is the measurement noise modeled as a normal Gaussian ($\boldsymbol{w}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_y(\boldsymbol{y}_k; \boldsymbol{\theta}_k))$).

The measurement noise covariance of $\boldsymbol{w}_k$, $\boldsymbol{R}_y$, is defined as the inverse of an empirically derived approximation of the predicted FIM, $\boldsymbol{R}_{\boldsymbol{y}} = \left[J_{\boldsymbol{y}}^{-}(\boldsymbol{y}; \boldsymbol{\theta})\right]^{-1}$.

7.2.3 Fully Adaptive Radar Multiple Target Tracking Optimization

Following the approach taken by [7], for FAR-STT, the development of the optimization for FAR-MTT expanded on the range and velocity track standard deviation objectives used in the single target application.

For the work presented here, we will only look at a single radar application. The application will be radar timeline minimization, where we want to have short dwells and acceptable track performance, while minimizing sensor resource usage. In the single target outlook from [7], the effects of the waveform parameters impact have been considered, through which we recall that changes in waveform bandwidth impact the nominal system range resolution $\delta R = \frac{c}{2B}$ and the associated tracking performance; it is noted that this performance is also dependent on the target SNR [7]. Changes in f_{PRF} and N_{p} in a CPI affect the velocity precision $\delta V = \frac{\lambda}{2}\frac{1}{CPI} = \frac{\lambda f_{\mathrm{PRF}}}{2N_{\mathrm{p}}}$, while the f_{PRF} alone affects the maximum Doppler frequency amplitude to avoid Doppler aliasing $f_{D_{max}} = \frac{PRF}{2}$. The number of pulses in a CPI further impacts the SNR of the target due to the coherent processing performed within a CPI. Keeping these relations

Approved for public release; distribution is unlimited.

in mind, there is additional condition that is necessary in the multiple target outlook: separability of targets in range and velocity.

In viewing this additional concern for MTT, we can begin to see that target separation can be best represented through our performance and measurement cost functions. However, the constrained optimization formulation of the executive optimization in [7] may prove to be infeasible for achieving optimal performance in the multiple target tracking environment due to the establishment of hard objectives represented through the constraints, which sparingly allow for constraint violation [112]. Therefore, we propose a weighted penalty function executive optimization approach, where we introduce new forms of the performance and measurement costs.

MTT Performance Cost Function

The performance cost function takes a different form than the single target case in [7]. In designing the cost function, focused its design on capturing the costs associated with *not* achieving tracking performance objectives. For range and velocity performance, the tracking performance objectives were defined as: 1) individual tracking performance in range and velocity ($\sigma_{R,i}$, $\sigma_{V,i}$: for $i = 1{:}M$, where M is the number of targets); and 2) inter-target separability in the measurement space.

For individual target tracking performance, the focus was on maintaining the track performance of the individual target's below a user-defined goal. The individual target tracking performances were maintained in order to evaluate the confidence of the estimate of each target's state. The track performance was evaluated using the inverse of the *Predicted Conditional Bayesian Information Matrix* (PC-BIM), known as the *Predicted Conditional Cramer-Rao Bound* (PC-CRB). Maintaining a low PC-CRB

Approved for public release; distribution is unlimited.

(in comparison to a user-defined goal) translated into having a consistently low error on the estimation of the tracking states.

Target separability was the other objective realized through the individual tracking performances of each target. The contributions of closely spaced targets were captured in the inverse of the PC-BIM where existing cross-correlations between the range and velocity measurements of individual targets resulted in an increase in measurement error.

The PC-BIM can be shown in the following form:

First, we state the PIM as follows using the predicted target error covariance from Section 2.1.3:

$$\boldsymbol{U}_k^-(\boldsymbol{\theta}_k) = \left(\boldsymbol{P}_k^-(\boldsymbol{\theta}_k)\right)^{-1} \tag{7.54}$$

Then, we state the FIM/Expected FIM as follows using the FIM from Section 7.2.2:

$$\boldsymbol{J}_k^-(\boldsymbol{\theta}_k) = \boldsymbol{H}^T \boldsymbol{J}_y \left(\boldsymbol{h}\left(\hat{\mathbf{x}}_k^-(\boldsymbol{\theta}_k)\right); \boldsymbol{\theta}_k\right) \boldsymbol{H} \tag{7.55}$$

Finally, we end up with the PC-BIM, $\boldsymbol{U}^-(\boldsymbol{\theta}_k)$, as:

$$\begin{aligned} &\boldsymbol{U}^-(\boldsymbol{\theta}_k) = \boldsymbol{U}_k^-(\boldsymbol{\theta}_k) + \boldsymbol{J}_k^-(\boldsymbol{\theta}_k) = \\ &\left(\boldsymbol{P}_k^-(\boldsymbol{\theta}_k)\right)^{-1} + \boldsymbol{H}^T \boldsymbol{J}_y \left(\boldsymbol{h}\left(\hat{\mathbf{x}}_{k|k-1}^-(\boldsymbol{\theta}_k)\right); \boldsymbol{\theta}_k\right) \boldsymbol{H} \end{aligned} \tag{7.56}$$

Recalling that the PC-CRB, $\boldsymbol{O}_k^-(\boldsymbol{\theta}_k)$, is the inverse of the PC-BIM , we get the following for the PC-CRB:

$$\begin{aligned} &\boldsymbol{O}_k^-(\boldsymbol{\theta}_k) = \boldsymbol{P}_k^-(\boldsymbol{\theta}_k) - \boldsymbol{P}_k^-(\boldsymbol{\theta}_k)\boldsymbol{H}^T \\ &\left[\boldsymbol{H}\boldsymbol{P}_k^-(\boldsymbol{\theta}_k)\boldsymbol{H}^T + \left(\boldsymbol{J}_y(\boldsymbol{\theta}_k)\right)^{-1}\right]^{-1} \boldsymbol{H}\boldsymbol{P}_k^-(\boldsymbol{\theta}_k). \end{aligned} \tag{7.57}$$

The elements along the diagonal of the PC-CRB contain the squared range and velocity performance of the targets captured through the variance, by which we can take the square root and obtain $\sigma_{R,i}$ and $\sigma_{V,i}$ for $i = 1{:}M$ for the range and velocity

Approved for public release; distribution is unlimited.

standard deviation, respectively. Due to the assumptions made for implementing the MAP-PF that allow us to separate the penalty function into individual penalty functions for each target, we have to assume that $\boldsymbol{P}_k^-$ is a block diagonal matrix and therefore assume that $\boldsymbol{O}_k^-(\boldsymbol{\theta}_k)$ is a block diagonal matrix. This approximation removes the off-diagonal elements of $\boldsymbol{O}_k^-(\boldsymbol{\theta}_k)$ to approximate it as a block diagonal matrix while still capturing the mutual information associated between targets when taking the inverse of the PC-BIM.

Considering the performance objectives, the performance cost function takes the following form:

$$C_P(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k^-) = \sum_{i=1}^{M} q_i(\boldsymbol{\theta}_k, \hat{\mathbf{x}}_k^-) + \upsilon_i(\boldsymbol{\theta}_k, \hat{\mathbf{x}}_k^-) \tag{7.58}$$

$$\begin{aligned} q_i(\boldsymbol{\theta}_k, \hat{\mathbf{x}}_k^-,) &= \frac{w_{TP_i}(\hat{\mathbf{x}}_k^-; \boldsymbol{\theta}_k)}{2} \left[\frac{\sigma_{R,i}(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k^-) - \sigma_R^*}{\sigma_R^*} u(\sigma_{R,i}(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k^-) - \sigma_R^*) \right]^2 \\ \upsilon_i(\boldsymbol{\theta}_k, \hat{\mathbf{x}}_k^-) &= \frac{w_{TP_i}(\hat{\mathbf{x}}_k^-; \boldsymbol{\theta}_k)}{2} \left[\frac{\sigma_{V,i}(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k^-) - \sigma_V^*}{\sigma_V^*} u(\sigma_{V,i}(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k^-) - \sigma_V^*) \right]^2 \end{aligned} \tag{7.59}$$

where q_i and υ_i represents the individual tracking performance cost for the range and velocity track error of the i^{th} target with i =1:M, $u(\cdot)$ is the unit step function and $w_{TP_i}(\hat{\mathbf{x}}_k^-; \boldsymbol{\theta}_k)$ is a weight meant to further emphasize target importance based on the predicted SNR, $\hat{\boldsymbol{\chi}}$.

The weight for emphasizing target performance, w_{TP}, is an exponential function of the predicted SNR state. The weight assigns high importance (high weight value) to low SNR predictions and low importance (low weight value) to high SNR predictions. This weight value assignment translates into importance being placed on a target with lower SNR versus higher SNR at each time t_k. This adaptive weight choice was

Approved for public release; distribution is unlimited.

inspired by the notion of weighted goal programming which is often applied in multi-objective optimizations [113, 114] in order to balance the needs of achieving multiple desired objectives. Due to the approximations made for the PC-CRB, this weight is meant to further emphasize target performance importance.

Weight selection in goal programming can be understood as an indication of relative importance of each objective [113]. The relative importance of each objective can be realized through establishing preference towards one objective over another and this preference is usually determined by the decion-maker/user. In further considering a multiple target environment where the importance of a target's track performance may shift according to its position in the state space and its response, SNR can be viewed as a strong contributor to target performance. Holding this view, preference of target tracks can be seen as functions of SNR (due to the inverse relation SNR has with the precision of an estimate). Inspired by the idea of priority/preference in goal programming and an understanding of the influence of SNR in the precision our track estimates, the weight w_{TP} was developed to act as a target preference.

This preference selection comes with the possibility of lower SNR targets receiving more preference than higher SNR targets, however this simply translates into higher resource consumption due to the lower SNR target's performance needs. While the higher resource consumption is not ideal, as we want to minimize resource usage, the performance for both targets will still be optimized and when the performance needs of each target is low, resource consumption concerns will be the focus of the optimization.

Approved for public release; distribution is unlimited.

MTT Measurement Cost Function

The measurement cost function was defined similar to the single target tracking case because the same measurement resources were being considered. In developing the measurement cost function, we focused on defining its objective. The objective of the measurement cost function was to minimize radar timeline usage by adapting the waveform parameters. For this work, an LFM chirp waveform was used with parameters (f_{PRF}, N_{p},τ, and B). We augmented the radar timeline minimization measurement objective with an additional power usage control objective. The intention was to avoid overspending power resources once an acceptable SNR level for a task was achieved.

For radar timeline minimization, we want to complete a multitude of tasks within a set amount of time. This application requires short dwells and low processing times. The short dwells were realized by using a smaller number of pulses (low N_{p}) and higher f_{PRF}, which minimized the CPI. The low processing time was enforced by favoring low τ and B, which gave a smaller number of fast time samples to be operated on.

For power usage control, we wanted to disfavor over-spending measurement resources once an acceptable SNR for the radar task was achieved. This acceptable SNR was an empirically defined user value considering the SNR needed to detect a target's presence and maintain acceptable track performance. This control was intended to be realized through maintaining τ and N_{p} when an acceptable SNR for a radar application is obtained. These two resource usage considerations dictated how the radar system should manage its resources from a conservation standpoint. Using these considerations, the measurement cost function took the following form:

Approved for public release; distribution is unlimited.

$$
\begin{aligned}
C_M(\boldsymbol{\theta}_k) &= \sum_{i=1}^{4} \frac{w_{\mathrm{MTT},i}}{2} \left(\frac{\theta_{k,i} - \theta_i^{\dagger}}{\theta_i^{\dagger}} \right)^2 \\
w_{\mathrm{MTT},i} &= \begin{cases} w_{\mathrm{M},i}\ w_{\mathrm{SNR},i}(\hat{\mathbf{x}}_k^-; \boldsymbol{\theta}_k) & , \text{for } i = 2(N_{\mathrm{p}}), 3(\tau) \\ w_{\mathrm{M},i} & , \text{for } i = 1(f_{\mathrm{PRF}}), 4(B) \end{cases}
\end{aligned}
\tag{7.60}
$$

where $\theta_{k,i}$ is the i^{th} parameter selection at the k^{th} time index, $\theta_i^{\dagger}$ is the goal value for the i^{th} parameter, $w_{\mathrm{MTT},i}$ is the weight for the i^{th} parameter, $w_{\mathrm{M},i}$ is the radar application resource weight that captures the importance of resource usage from the radar application perspective, and $w_{\mathrm{SNR},i}(\hat{\mathbf{x}}_k^-; \boldsymbol{\theta}_k)$ is the power usage resource weight that captures the importance of power usage control using the target state estimate, $\hat{\mathbf{x}}_k^-$, as an input and parameterized by $\boldsymbol{\theta_k}$.

In viewing the formation of $w_{\mathrm{MTT},i}$, we can see that the radar application resource weights ($w_{\mathrm{M},i}$) are applied to indices 1 and 4 representing f_{PRF} and B as these parameters are only concerned with resource consumption for radar timeline minimization. However, for indices 2 and 3, an additional weight is multiplied by the usual resource consumption weight to emphasize the power control. Indices 2 and 3 represent N_{p}, and τ respectively. While the weights $w_{\mathrm{M},i}$ were set following the guidelines outlined in [7], they were modified such that $0 \leq w_{\mathrm{M},i} \leq 1$.

The weight function for $w_{\mathrm{SNR},i}$ is an exponential function that increases $w_{\mathrm{MTT},i}$ for SNR values that are greater than the SNR necessary for the radar application. This SNR threshold was an empirically defined user value considering the SNR needed to detect a target's presence and maintain acceptable track performance. The weight, $w_{\mathrm{MTT},i}$ represents the need to have low resource consumption for the given radar application of radar timeline minimization and avoid wasting resources when acceptable SNR is achieved.

Approved for public release; distribution is unlimited.

Table 7.1: Measurement Cost Function Parameters

Parameter	**Measurement Cost Weight** ($w_{M,i}$))	$\theta_i^\dagger$
f_{PRF}(i=1)	1	12kHz
N_{p}(i=2)	0.42	128 pulses
τ(i=3)	0.009	0.075 μs
B(i=4)	0.05	100 MHz

MTT Executive Cost Function

The cost functions for both the measurement and performance take on a quadratic form to enforce stricter objectives by quantifying how much an improvement in our objectives is worth and how much a poorer value penalizes the objective [115] as well as to promote convexity. The unit step function, $u(\cdot)$, is used for the performance objectives to enforce a penalty only when the objectives are below the objective goal value. The form of these objective functions use the penalty function method [33] in order to form a weighted sum form. This weighted sum form was used for the executive cost as it is known to be one of the simplest, computationally efficient approach to multi-objective optimization in goal programming [114].

The executive optimization for FAR-MTT took the following form:

$$\begin{aligned}
\boldsymbol{\theta_k} &= C_E(\boldsymbol{\theta}_k) \\
\boldsymbol{\theta_k} &= \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}_k}{\arg\min}\, C_M(\boldsymbol{\theta}_k) + C_P(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k^-) \\
\boldsymbol{\Theta}_k &= \boldsymbol{\theta}_k \in \boldsymbol{Z}^N : \begin{cases} \boldsymbol{\theta}_{k,1}^{lb}(\hat{\mathbf{x}}_k^-) \leq \boldsymbol{\theta}_{k,1} \leq \boldsymbol{\theta}_1^{ub} \\ \boldsymbol{\theta}_2^{lb} \leq \boldsymbol{\theta}_{k,2} \leq \boldsymbol{\theta}_2^{ub} \\ \boldsymbol{\theta}_3^{lb} \leq \boldsymbol{\theta}_{k,3} \leq \boldsymbol{\theta}_3^{ub} \\ \boldsymbol{\theta}_4^{lb} \leq \boldsymbol{\theta}_{k,4} \leq \boldsymbol{\theta}_4^{ub} \end{cases}
\end{aligned} \tag{7.61}$$

Approved for public release; distribution is unlimited.

where $\boldsymbol{\Theta}_k$ is the set of available sensor parameter selections; $\boldsymbol{Z}^N$ is the set of continuous values, $\boldsymbol{\theta}^{lb}$ & $\boldsymbol{\theta}^{ub}$ are the lower and upper bounds for the sensor parameter selections, respectively; $\boldsymbol{\theta}^{lb}_{k,1}(\hat{\mathbf{x}}^-_k)$ is the parameterized lower bound for the f_{PRF} sensor parameter selection for which the available sensor parameter selections for f_{PRF} may change at a given time t_k. This changing lower bound was included in order to prevent Doppler-aliasing by limiting the set of available sensor parameter selections for f_{PRF} at a given time t_k based on the maximum predicted target Doppler response. **Note**: In implementation, the parameter selections for the sensor parameters are optimized based on a set of bounded continuous values but the selections are rounded to correspond to discrete indices.

7.2.4 FAR-MTT Recursion

In establishing, the detection and tracking component enforced through the MAP-PF methodology and the optimization method for selecting the new sensor parameters to best observe the multiple target environment. We can now state the FAR-MTT recursion as follows:

1. Initialize target motion statistics for perceptual processor using $\hat{\mathbf{x}}_0$ and $\boldsymbol{P}_0$
2. Perform executive optimization using target motion update (prediction) and the PC-BIM to find optimal sensor parameters, $\boldsymbol{\theta}_k$;
3. Obtain hardware sensor measurement through which we obtain $\mathbf{z}_k$ according to $\boldsymbol{\theta}_k$;
4. Perform motion update in perceptual processor, obtaining $\hat{\mathbf{x}}^-_k$ and $\boldsymbol{P}^-_k$;

Approved for public release; distribution is unlimited.

5. Obtain detector measurement of natural parameters using the penalty term $\boldsymbol{\phi}(\boldsymbol{y}_{m,k}, \mathbf{x}_{m,k})$ and the range-Doppler surface $10\log_{10}(\boldsymbol{RD}_{\mathbf{z}_k})$ approximating the log-likelihood function for each target ;

6. Perform an information update with the tracker through updating the measurement covariance $\boldsymbol{R}_k$ with the FIM estimation, obtaining $\hat{\mathbf{x}}_k^+$ and $\boldsymbol{P}_k^+$.

7.3 Results

Using the cost functions, the target models and following the FAR-MTT recursion in section 7.2.4, the following simulation was generated with the FARMS MATLAB environment as well as tested experimentally using the CREW at The Ohio State University ElectroScience Laboratory with targets moving within a 10 meter range.

Three different environments were captured: Case 1 & 2 which are both two target simulations with different target orientations; and Case 3 which is a two target lab experiment. The radar operating parameters that were used are outlined in Tables 7.2 and 7.3. Case 1 consists of two targets: Target 1 (T1) moving from a range of 4m to 6.1m and Target 2 (T2) moving from a range of 9m to 6.8m, with both targets moving at complementary sinusoidal speeds. Case 2 consists of two targets: T1 moving from a range of 2m to 7m and T2 moving from a range of 7m to 2m, with both targets overlapping at 5m and moving in opposite directions at complementary sinusoidal speeds. Case 3 consists of two targets: T1 moving from a range of 2m to 7m and T2 moving from a range of 7m to 2m, with both targets overlapping at 5m and moving in opposite directions at complementary "*square-wave*" speeds. Case 4 consists of three targets T1 moving from a range of 2m to 4m, T2 moving from a range of 5m to 3m, and T moving from a range of 8m to 6m; with T1 moving at a

Approved for public release; distribution is unlimited.

Table 7.2: Adaptive Radar Operating Parameters

Parameter	**Number of Steps**	**Value Range**
F_c	1	95.5 GHz
Transmit Power	1	Exp:16.5dBm
f_{PRF}	10	12kHz - 2kHz
N_{p}	10	128 - 1024 pulses
B	10	100-1000 MHz
τ	10	0.075- 0.6 μs

Table 7.3: Static Radar Operating Parameters

Parameter	**Value Range**
F_c	95.5 GHz
Transmit Power	Exp:16.5dBm
f_{PRF}	4.04 kHz
N_{p}	128 pulses
B	500 MHz
τ	0.15 μs

complementary velocity to T2 and T3. For Case 1, 3 & 4 both the fixed and adaptive parameter cases are shown, whereas for Case 2 only the adaptive parameter case is shown. An illustration of the two target cases are shown in Figures 7.1 & 7.2, whereas the three target case is shown in Figure 7.3.

7.3.1 Case 1

Figures 7.4, 7.5, and 7.6 display the results for Case 1 in which the fixed parameter case is held as a baseline to compare with the four parameter (f_{PRF},B,τ, and N_{p}) adaptation case. Figure 7.4 displays a plot of the parameter adaptation over time for the adaptive parameter case. Due to the fact that the parameters for the fixed parameter case remain fixed in time and can be referenced from Table 7.3, the parameter adaptation over time is only shown for the adaptive parameter case. Figure

Approved for public release; distribution is unlimited.

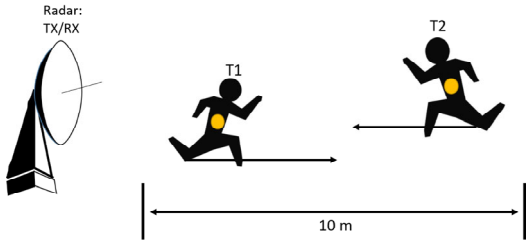


Figure 7.1: Target Model: Two Targets (for Case 1)

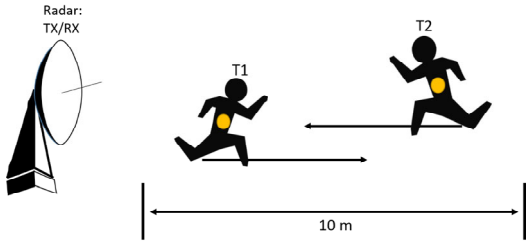


Figure 7.2: Target Model: Two Targets (for Case 2 & 3)

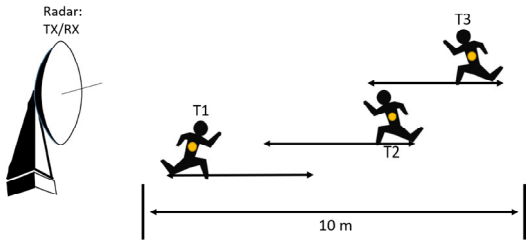


Figure 7.3: Target Model: Two Targets (for Case 4)

Approved for public release; distribution is unlimited.

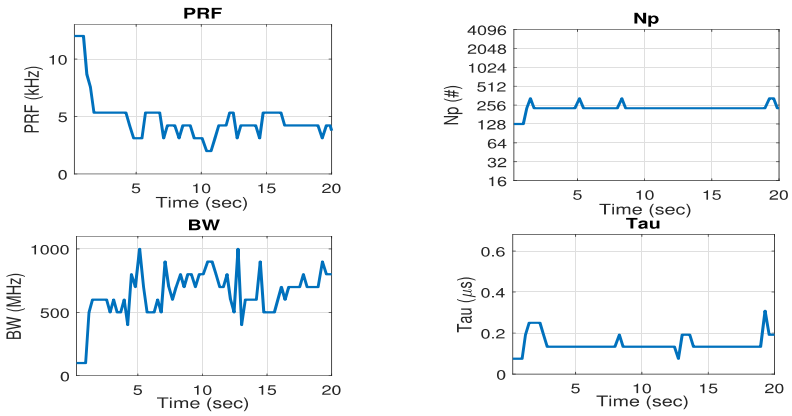
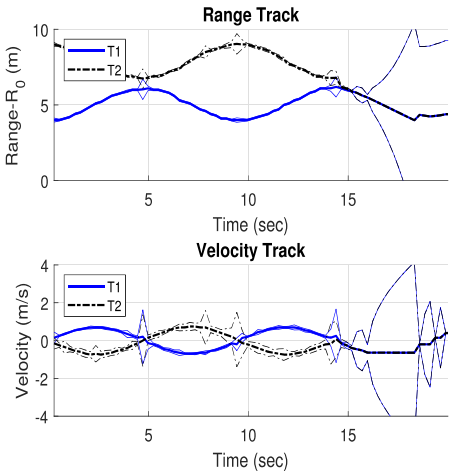


Figure 7.4: Parameters over Time: Adaptive Parameters **Case 1** [**Sim**]

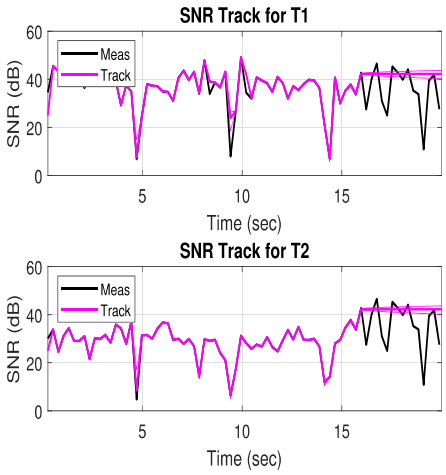
7.5 contains 8 plots: sub-figures (a/c) which display the overlaid range & velocity tracks estimates and deviations of both targets for the fixed and adaptive parameter cases & sub-figures (b/d) which display the individual SNR tracks for each target for the fixed and adaptive parameter cases. Figure 7.6 displays the normalized Doppler frequency and performance goals for both the fixed and adaptive parameter cases, in which the performance goals are realized through the track error in range and velocity standard deviation.

Using the fixed parameter case as a baseline, we refer to Figures 7.5 (a)&(b), where it can be seen that with the fixed parameter values we are able to maintain a noisy track in range and velocity up until $\sim$ 14.9 seconds. After this point, the target's are merged resulting in T2 being merged to T1's track in range and velocity resulting in an increase in error for both target's due to uncertainty shown in Figure 7.5 (a)&(b) and Figure 7.6 (a). When looking at Figure 7.5 (a), it can be seen that the track merge happens when both target's are reducing their speed running risk of entering the zero Doppler region.

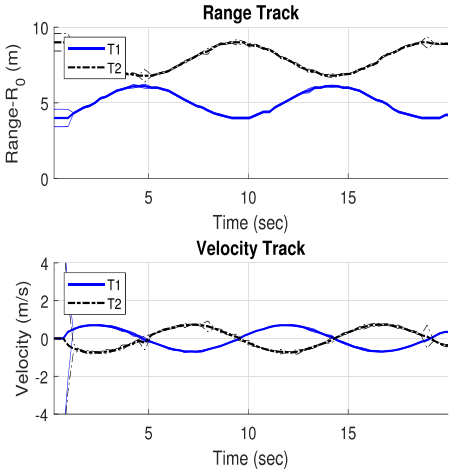
Approved for public release; distribution is unlimited.



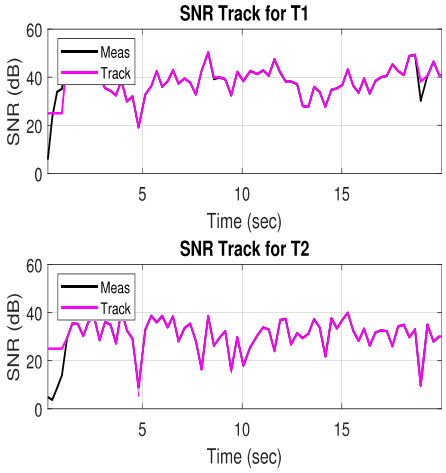
(a) Fixed Parameters



(b) Fixed Parameters



(c) Adaptive Parameters



(d) Adaptive Parameters

Figure 7.5: Tracks over Time: **Case 1 [Sim]**

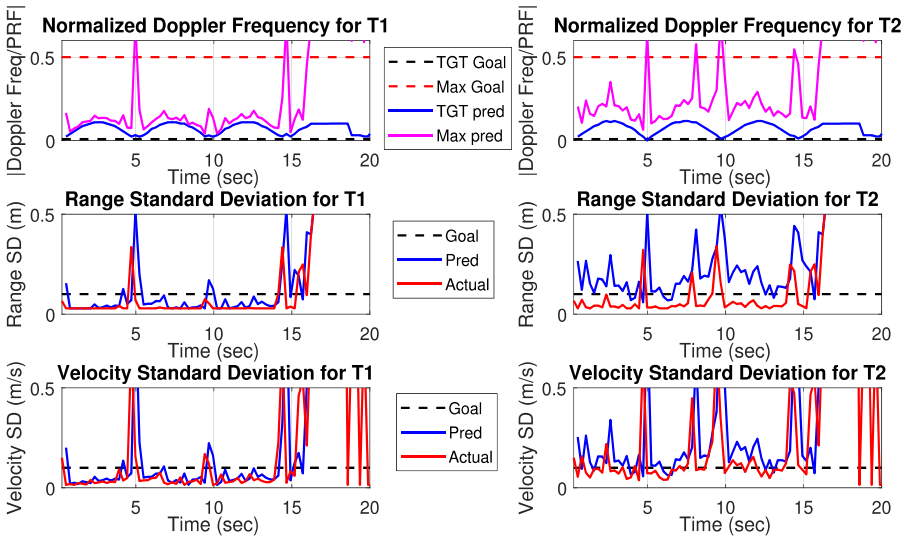
Approved for public release; distribution is unlimited.

In this work, the zero-Doppler region is nulled such that targets within the zero-Doppler region may result in a possible loss of detection measurement, reduced SNR and increase in track error. In looking at the SNR track for T2 in Figure 7.5(b), it can also be seen that it has a lower SNR response than T1 over time, due to the fact that T2 remains further from the radar than T1. The impact of this increased distance is also seen in Figure 7.6 (a) over time, through which the range and velocity standard deviation of T2 tends to be above the user-defined goal. T1 in contrast tends to remain below the user-defined goal. The higher errors in track for T2 in combination with the proximity of T1 result in the track merge for the fixed parameter case.

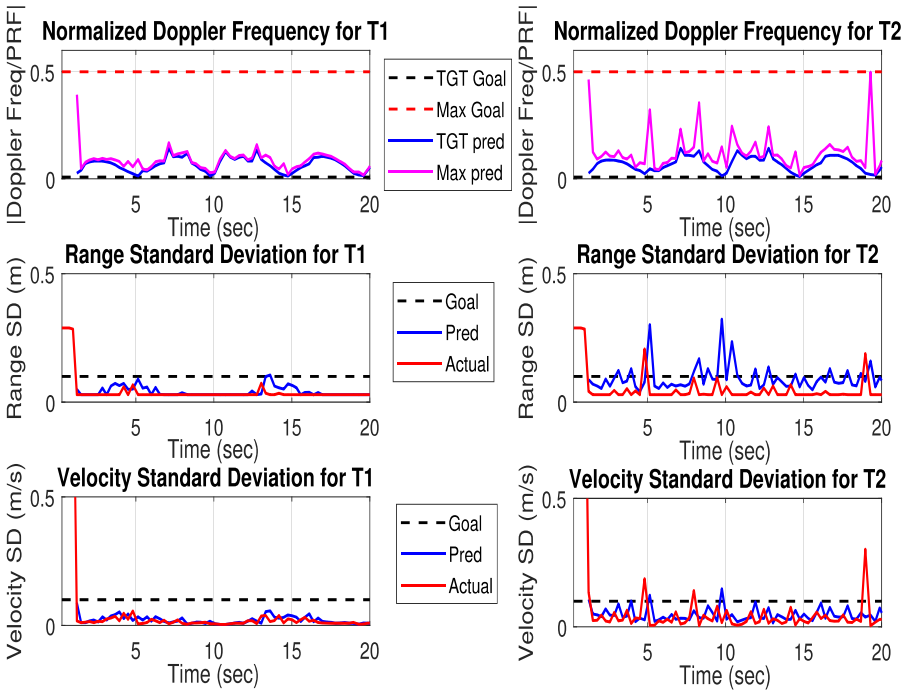
In comparison to the fixed parameter case, when we view Figures 7.5 (c)&(d) and Figure 7.6 (b) we see that the adaptive parameter case maintains a low error track in range and velocity for both targets. In viewing Figure 7.4, we see that when allowed to adapt, the radar favors low f_{PRF} values, high B values, relatively low τ values and maintains a mid-level N_{p} value. The lower f_{PRF} combined with the mid N_{p} value result in an improved Doppler resolution as Doppler resolution is inversely proportional to the CPI. The high B values result in improved range resolution as range resolution is inversely proportional to B.

Specifically,when viewing Figure 7.4, we see that around the same point in time where the fixed parameter case merges the targets ($\sim$ 14.9 seconds), the adaptive parameter case selects a low f_{PRF} and high B to be able to resolve the target in both range and velocity. In addition to avoiding a track merge, the adaptive parameter case maintains a range and velocity error below the user-specified threshold at most points in time as seen in Figure 7.6 (b). The relatively constant N_{p} value is a result of T2's distance from the radar and the need to maintain an acceptable resolution

Approved for public release; distribution is unlimited.



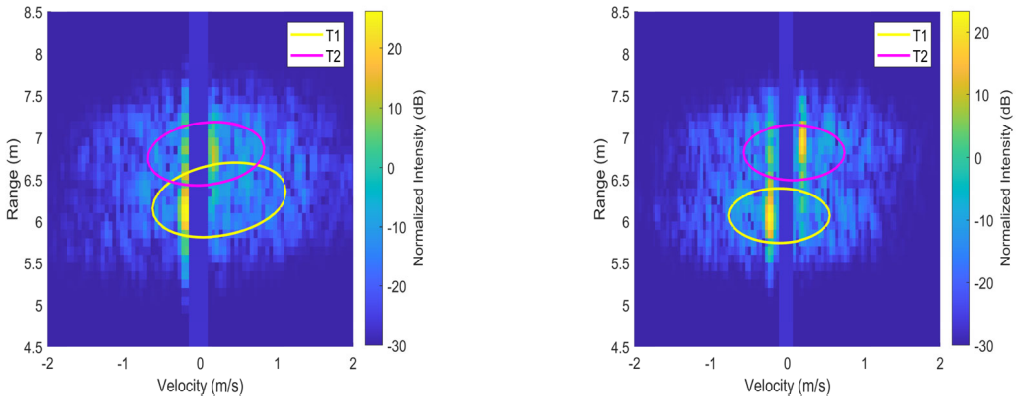
(a) Fixed Parameters



(b) Adaptive Parameters

Figure 7.6: Normalized Doppler and Performance Goals: **Case 1 [Sim]**

Approved for public release; distribution is unlimited.



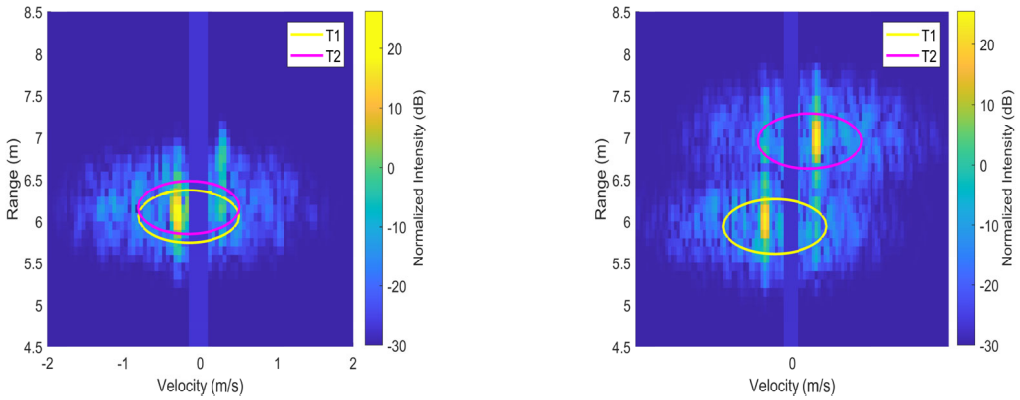
(a) Fixed Parameters (b) Adaptive Parameters

Figure 7.7: Penalized RD Surface at $\sim$ 14.9 sec runtime: **Case 1 [Sim]**

for the two targets through which we recall that N_{p} has an effect on both the SNR and Doppler resolution. The SNR tracks in Figure 7.5 (b)&(d) are of particular note, as we see that when T2 is further away from the radar, the SNR of T2 seems to be consistently maintained at $\sim$ 30 dB for the adaptive parameter case whereas the SNR of T2 in the fixed parameter case varies more towards a lower SNR value $\sim$ 20 dB.

In order to further promote the benefit of the adaptive parameter case in maintaining the target track, we turn to the RD plots in Figures 7.7 and 7.8, where sub-figures (a) and (b) show the penalized RD surfaces for the fixed parameter and adaptive parameter case, respectively. Figure 7.7 is the range/Doppler surface at the instance in time before the track merge in the fixed parameter case, which is around $\sim$ 14.9 seconds. In Figure 7.7 (a) we see that the separation of the targets in the measurement space is poor and the targets are entering the zero-Doppler region. Note that the ellipses drawn around each target correspond to the predicted track error using the PIM. However, when viewing Figure 7.7 (b), there is better separation

Approved for public release; distribution is unlimited.



(a) Fixed Parameters: Penalized RD Surface

(b) Adaptive Parameters: Penalized RD Surface

Figure 7.8: Penalized RD Surface at $\sim$ 15.1 sec runtime: **Case 1 [Sim]**

between the targets as well as improved resolution in the measurement space. In Figure 7.8 (a), the fixed parameter case merges the track of T2 with T1, whereas in Figure 7.8 (b), the separation between the targets is maintained. Furthermore, using a 1000 trial Monte Carlo run, it was found that the fixed parameter case would merge the targets about 50% of the time, whereas the adaptive parameter case would only merge the targets about 28% of the time promoting the benefit of an adaptive radar in preventing track merging in a multiple target environment.

7.3.2 Case 2

Moving on to Case 2, we view Figures 7.9, 7.10, 7.11, and 7.12. In Case 2, we evaluate the impact of considering both targets when optimizing our sensor parameter selection as well as the effect of the parameterized weight applied to the performance cost objectives, $w_{TP_i}(\hat{\boldsymbol{x}}_k^-; \boldsymbol{\theta}_k)$. The fixed parameter results are not shown for Case 2 as this case focuses on the allocation of resources when optimizing for both targets versus

Approved for public release; distribution is unlimited.

optimizing for a single target when other targets are present. Figure 7.9 displays the overlaid parameter adaptations over time for the different adaptive cases: optimizing for T1 only, T2 only and both targets (the default optimization).

Figure 7.10 displays the range and velocity tracks for the adaptive parameter cases, respectively. In Figure 7.10, the range and velocity tracks corresponding to optimizing for T1 only (a-b), T2 only (c-d) and both T1 and T2 (d-f) are shown individually in their corresponding sub-figures. Figure 7.11 displays the overlaid normalized Doppler frequency and performance goals (in terms of the actual velocity and range standard deviation obtained) for the adaptive parameter cases when we choose to optimize for T1 only, T2 only and for both targets (the default optimization).

In viewing Figure 7.9, we see that when optimizing for both targets (the dashed green line), the parameter selections seem to settle on values between the ones selected when optimizing for either T1 only (the solid blue line) or T2 only (the dotted red line). It also worth noting that when better performance is needed for optimizing either T1 or T2 only, resulting in low f_{PRF}, high B, high τ, or high N_{p} , the case where we optimize for both T1 and T2 settles on values closer to those high measurement cost values to achieve better performance. This claim can be substantiated when viewing the range and velocity tracks in Figure 7.10. Due to the complementary sinusoidal nature of the targets' range motion, when T1 is close to the radar then T2 is further away and vice-versa.

Given this, we begin by looking at Figures 7.10 (a) & (b) which are the range, velocity and SNR tracks corresponding to when we optimize for T1 only. In this figure, we see that the range & velocity tracks for T1 are less noisy than T2; the SNR for T2 is only high when the target is close to the radar and low otherwise; and when

Approved for public release; distribution is unlimited.

T2 is further away from the radar the range and velocity track increases in error due to an increase in uncertainty from low SNR.

In looking at Figures 7.10 (c) & (d) which are the range, velocity and SNR tracks corresponding to when we optimize for T2 only, similar behavior is seen as when adapting for T1 only. In this case, we see that when T1 is further away from the radar, its uncertainty (error) in its range and velocity track increase. The increase in error for T1 and T2 in the T1 only optimization and the T2 only optimization is a result of focusing on a single target without regard for the other target and not needing to overspend resources. In the cases when we optimize for only one target, as the target begins approaching the radar, we begin relaxing our need for improved performance due to the target's SNR increasing. Similarly, when the target begins moving away from the radar our need for improved performance increases.

However, when looking at Figures 7.10 (e) & (f), it can be seen that the tracks for both targets are maintained with very little error in their estimates and relatively high consistent SNR responses. This point is further illustrated when viewing Figure 7.11, where we see that when optimizing for both targets (the dashed green line), we remain below our goal line for both range and velocity standard deviation with the spikes in the error corresponding to the target entering the zero-Doppler region. As expected, when jointly viewing Figure 7.10 and Figures 7.11 (a), (b), (c) & (d), we see that the large increases in the range and velocity standard deviation of T2 when optimizing for T1 only, corresponds to when T1 was closer to the radar & T2 was further away from the radar. Similarly, when optimizing for T2 only, the large increases in the range and velocity standard deviation of T1 correspond to when T2 was closer to the radar & T1 was further away from the radar.

Approved for public release; distribution is unlimited.

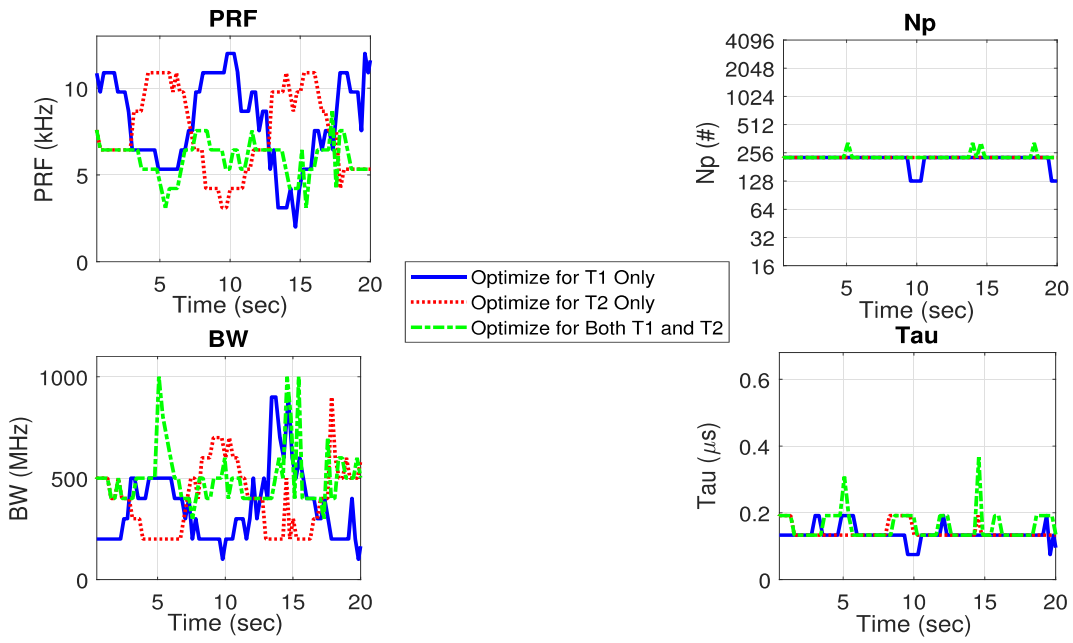
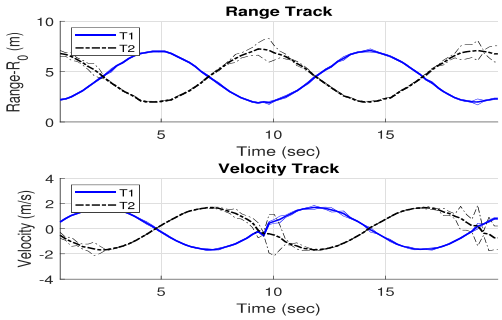


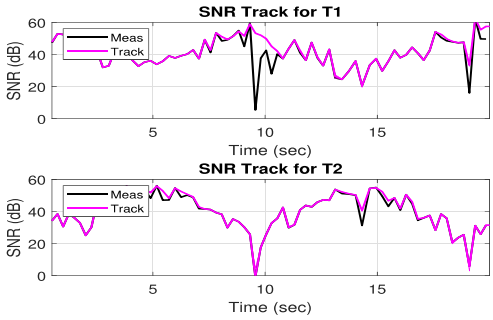
Figure 7.9: Parameters over Time: Adaptive Parameters **Case 2 [Sim]**

In viewing the effects of optimizing for a single target when multiple targets are present, we can begin to understand the idea of target prioritization and the effect of our performance cost weight, w_{TP}. Figure 7.12 diplays three groups of plots which reflect the value of w_{TP} over time when optimizing for T1 only, T2 only and both T1 and T2. In Figure 7.12 (a) and (b), we can see that the importance of improved track performance was decreased (lower value for w_{TP}) when the target being optimized for had a high SNR. Similarly, in Figure 7.12 (c), we see that when a target's SNR is high, the importance of improved track performance is decreased however it is also seen that the target with the lower SNR maintains its importance. This begins to illustrate the notion of prioritizing target performance using SNR mentioned in Section 7.2.3.

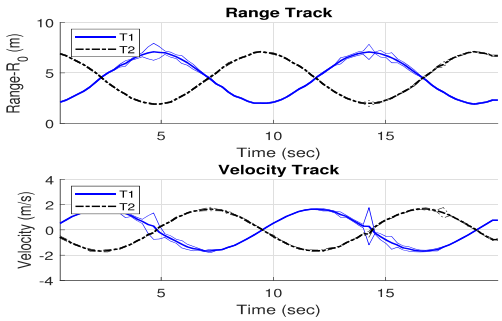
Approved for public release; distribution is unlimited.



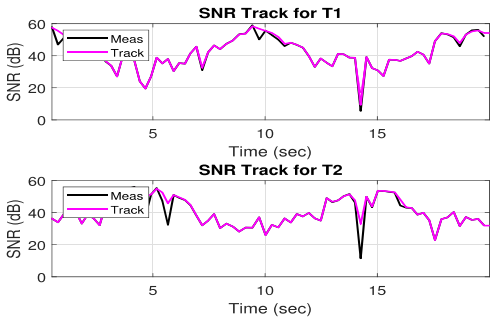
(a) Optimizing for T1 only



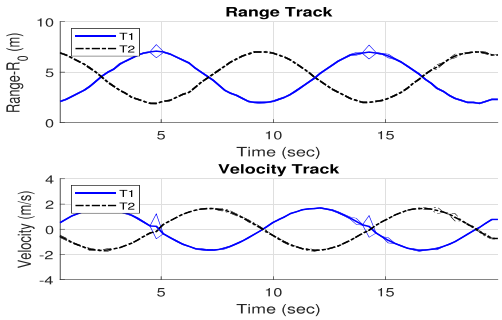
(b) Optimizing for T1 only



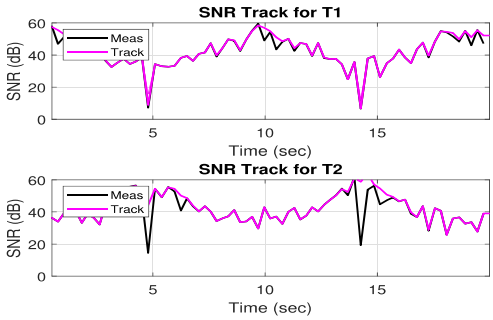
(c) Optimizing for T2 only



(d) Optimizing for T2 only



(e) Optimizing for both T1 and T2



(f) Optimizing for both T1 and T2

Figure 7.10: Tracks over Time: Adaptive Parameters **Case 2** [**Sim**]

Approved for public release; distribution is unlimited.

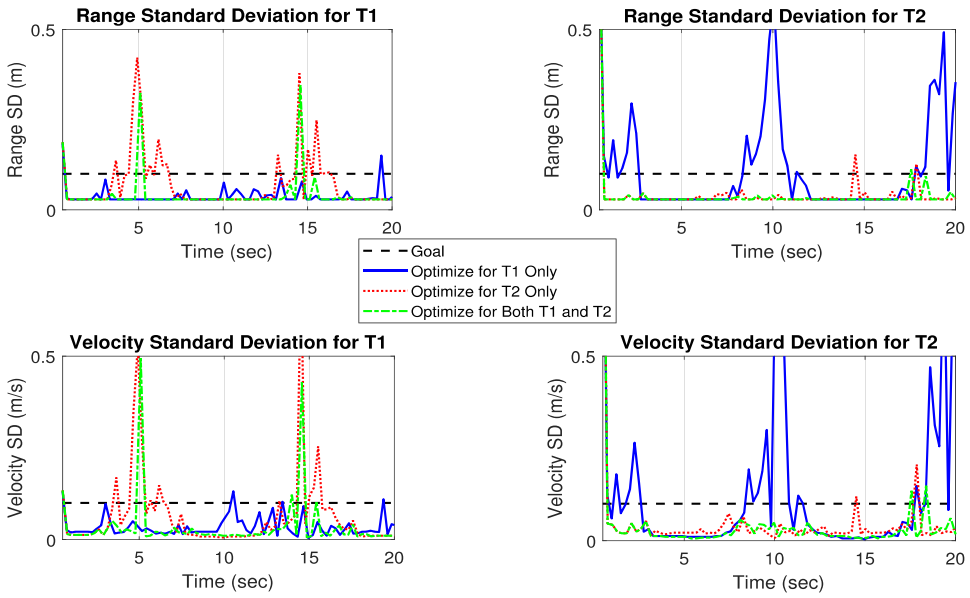
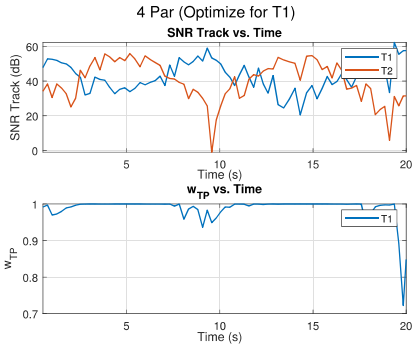


Figure 7.11: Normalized Doppler and Performance Goals: **Case 2 [Sim]**

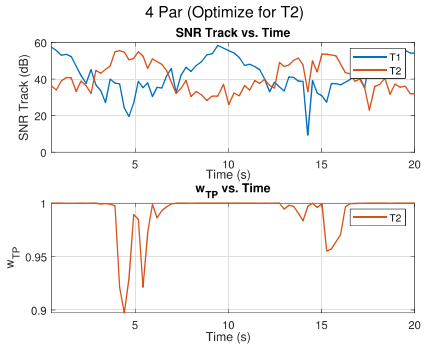
7.3.3 Case 3

Using the first two cases to setup an understanding of the framework's performance through simulation, we can now examine Case 3 which is a two target experimental example using the CREW system at The Ohio State University ElectroScience Laboratory. For Case 3, we focus on Figures 7.13, 7.14, and 7.15 to view the experimental results for the two target environment. It should be noted that the number of samples in the adaptive parameter case is smaller (lower update rate) than the fixed parameter case as the optimization performed consumed processing time. However, the fixed parameter case can still be used as a type of baseline for comparison against the adaptive parameter case. Similar to the figures shown before, we see the parameter variation over time in Figure 7.13. When observing Figures 7.13 and 7.14 together, we can

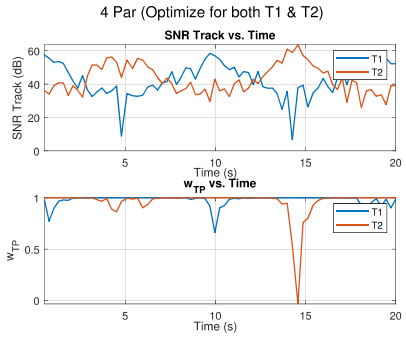
Approved for public release; distribution is unlimited.



(a) Optimizing for T1 only



(b) Optimizing for T2 only



(c) Optimizing for both T1 and T2

Figure 7.12: w_{TP} over Time: Adaptive Parameters

Approved for public release; distribution is unlimited.

see the relations between the parameter adaptations and the target tracks. For the adaptive parameter experimental example, we see that we choose lower f_{PRF} values when the target is slowing down (nearing the zero-Doppler region) and higher f_{PRF} values when the target is increasing in velocity (nearing the maximum unambiguous Doppler region).

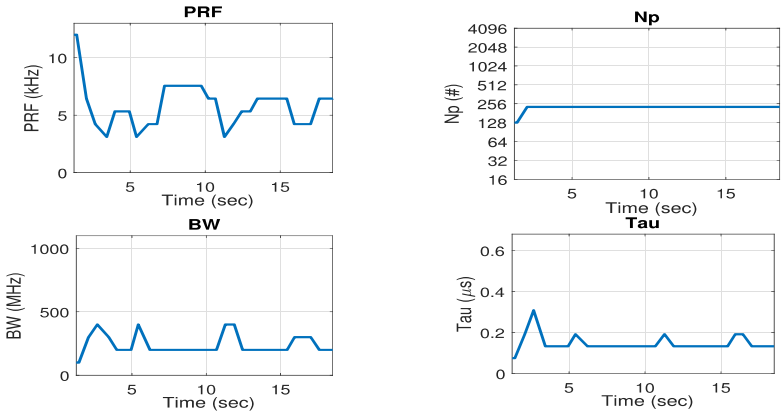


Figure 7.13: Parameters over Time: Adaptive Parameters **Case 3 [Exp]**

We also note that the targets are both moving at opposing velocities, except when near the zero-Doppler region at which the targets approach the same velocity. This observation begins to support the lower f_{PRF} selections because due to dual needs of maintaining adequate target separation in velocity as well as preventing a slow moving target from entering the zero-Doppler region, we aim to improve our Doppler resolution. We also note that both τ and B increase as the target gets further in range and decrease as the targets get closer in range. Our τ adaptations also limit themselves in order to avoid raising the power when the lowest SNR target is already

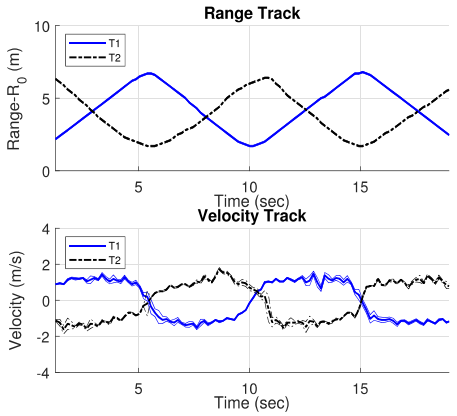
Approved for public release; distribution is unlimited.

high. This feature is especially highlighted when viewing Figure 7.14 (b) in which we see the max SNR for target 2 limited at ∼40dB between approximately 11-13 seconds.

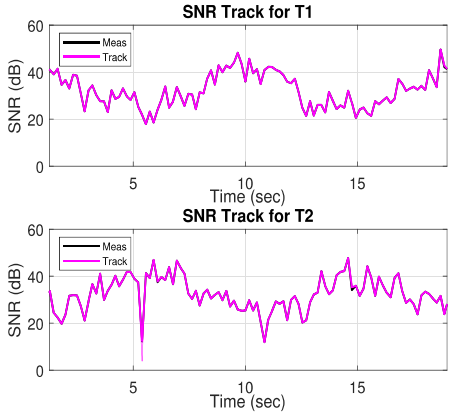
In viewing Figure 7.14 (a), we see that the tracks for velocity are noisy for both target 1 and 2 whereas in Figure 7.14 (c) the velocity tracks have less uncertainty (low track error) due to the lower f_{PRF} and maintained N_{p} selection. The noisy track for the fixed parameter case can be explained when looking at Figure 7.15 (a) where we see that our velocity standard deviation goals are being exceeded for both target 1 and 2 when the targets are further away.

In contrast to the fixed parameter case, we see in Figure 7.15 (b) that the four parameter case maintains a low range and velocity standard deviation, while remaining below the goal values. We see that for the fixed parameter case our range standard deviation remains extremely low, whereas in the four parameter case our range standard deviation is below the goal but slightly higher than the fixed case. This low range error performance for the fixed parameter case is due to the fact that we select a relatively high B value; while, in the adaptive parameter case, we select lower B values in order to save measurement resources. However, it should be noted that even with these lower B selections, we still obtain acceptable range error performance (below or at the goal value).

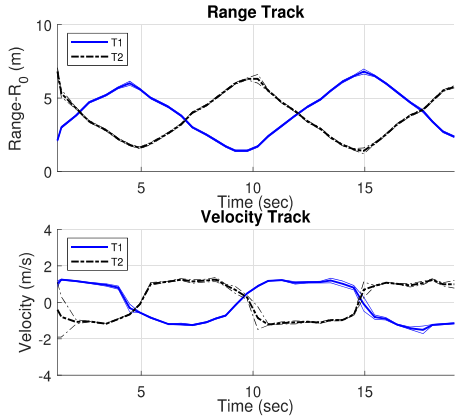
Approved for public release; distribution is unlimited.



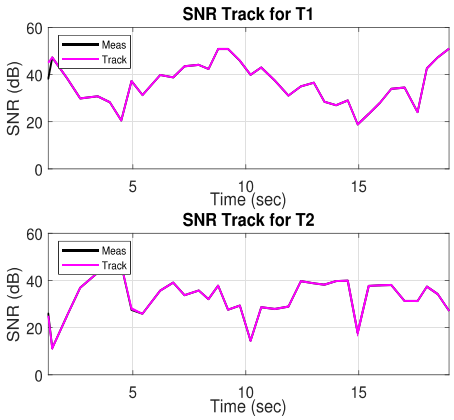
(a) Fixed Parameters



(b) Fixed Parameters



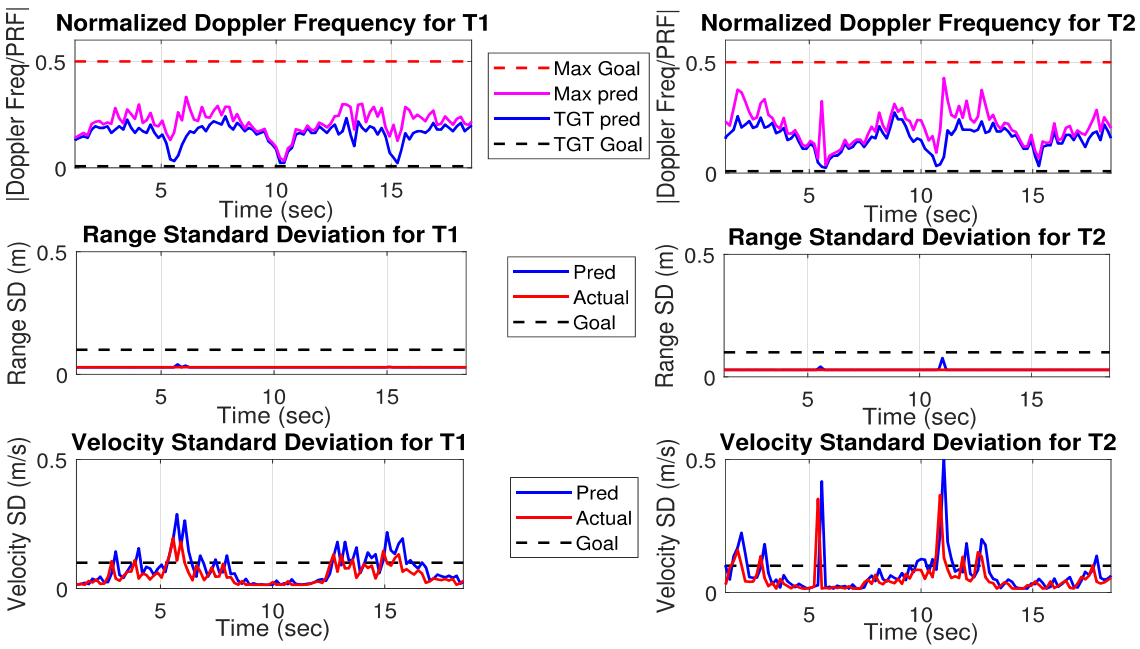
(c) Adaptive Parameters



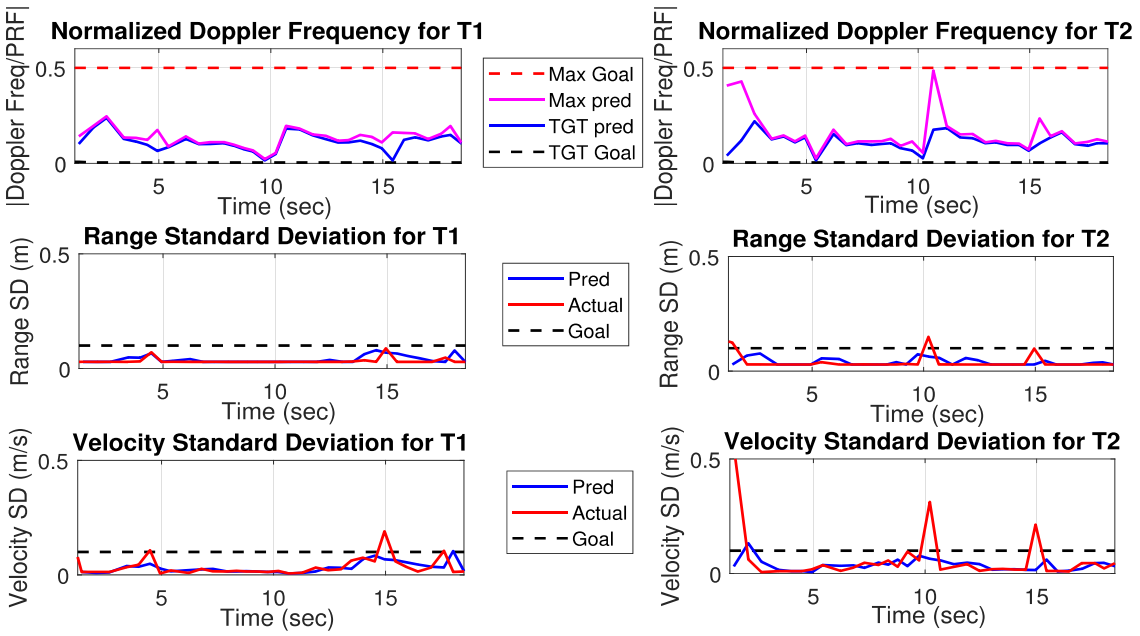
(d) Adaptive Parameters

Figure 7.14: Tracks over Time: **Case 3 [Exp]**

Approved for public release; distribution is unlimited.



(a) Fixed Parameters



(b) Adaptive Parameters

Figure 7.15: Normalized Doppler and Performance Goals: **Case 3 [Exp]**

Approved for public release; distribution is unlimited.

7.3.4 Case 4

Similar to the 2 target environment cases, Figures 7.16, 7.17, and 7.18 display the results for the three target environment (Case 4). In viewing our parameter adaptation over time, we see that in Figure 7.16 that our parameter variation is similar to that of the two target case, where we see lower f_{PRF} selections, higher B selections, relatively mid-level N_{p} selections,and low τ selections. In looking at these parameter value selections, we see that the three target environment results in higher resource needs (better performance) in comparison to the two target environments. This higher resource need is due to the complexity of the environment as well as there being more objectives added to the performance cost with the increased number of targets, resulting in higher costs for poor performance in track of the individual targets. Using the fixed parameter case as a baseline, we begin looking at Figure 7.17 (a) & (b), in which we see that the fixed parameter case results in a track merge of targets T1 and T2, due to their lack of range separation and relatively slow oscillatory motions at around $\sim$ 4.5 seconds. The track merge that occurs results in T1's track being merged with T2, past$\sim$ 4.5 seconds resulting in track uncertainty for both targets. This sustained uncertainty in track for both T1 and T2 in the fixed parameter case result in another track merge occurring between the merged tracks T1/T2 and T3 at around $\sim$ 15 seconds, where we see the merged tracks of T1, T2 and T3 resulting in large oscillatory errors in velocity and large error/uncertainty for T1 in range.

In comparison to the fixed parameter case, when we view the adaptive parameter case in Figures 7.18 and 7.17 (c) & (d), it is seen that the lower parameter adaptations for f_{PRF} , the mid-value selections for N_{p} and high B selections result in improved

Approved for public release; distribution is unlimited.

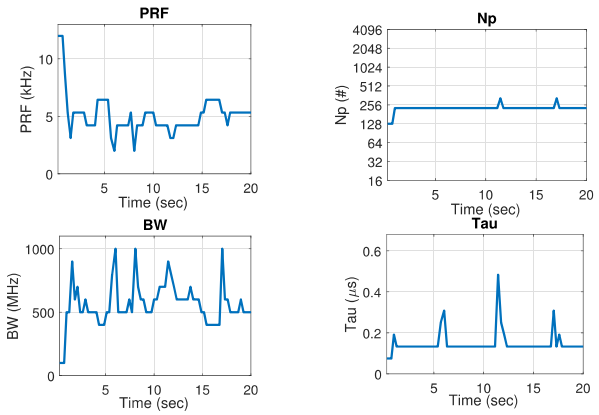
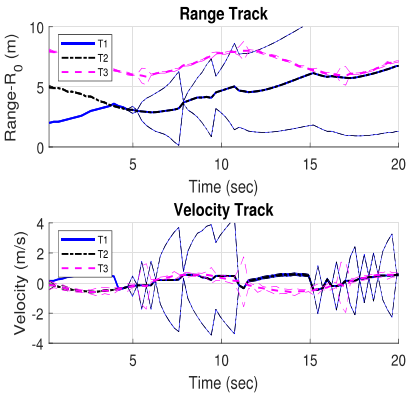
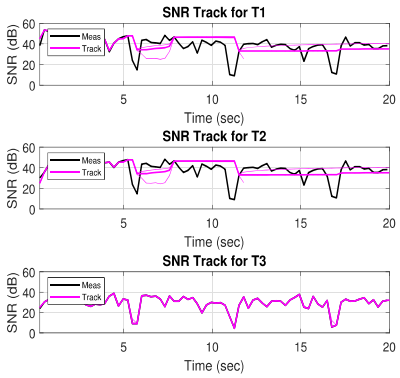


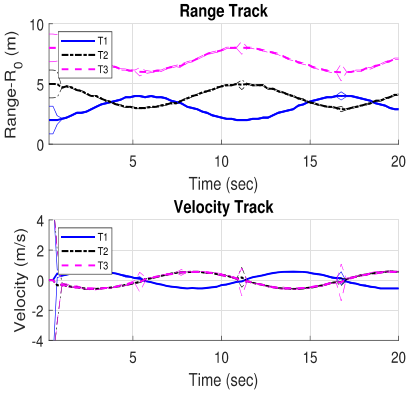
Figure 7.16: Parameters over Time: Adaptive Parameters **Case 4 [Sim]**



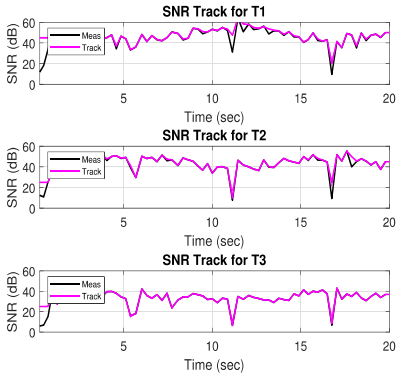
(a) Fixed Parameters



(b) Fixed Parameters



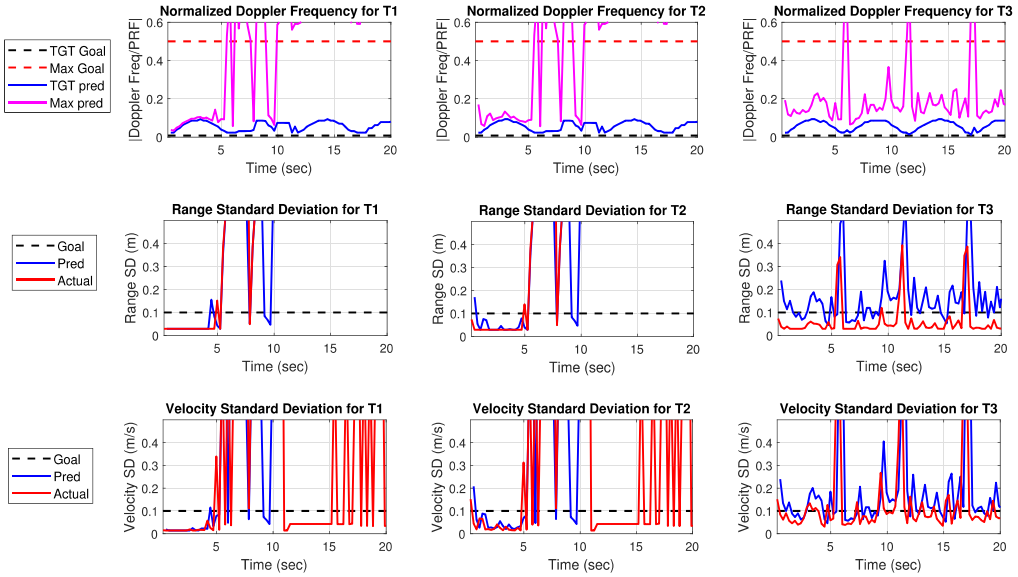
(c) Adaptive Parameters



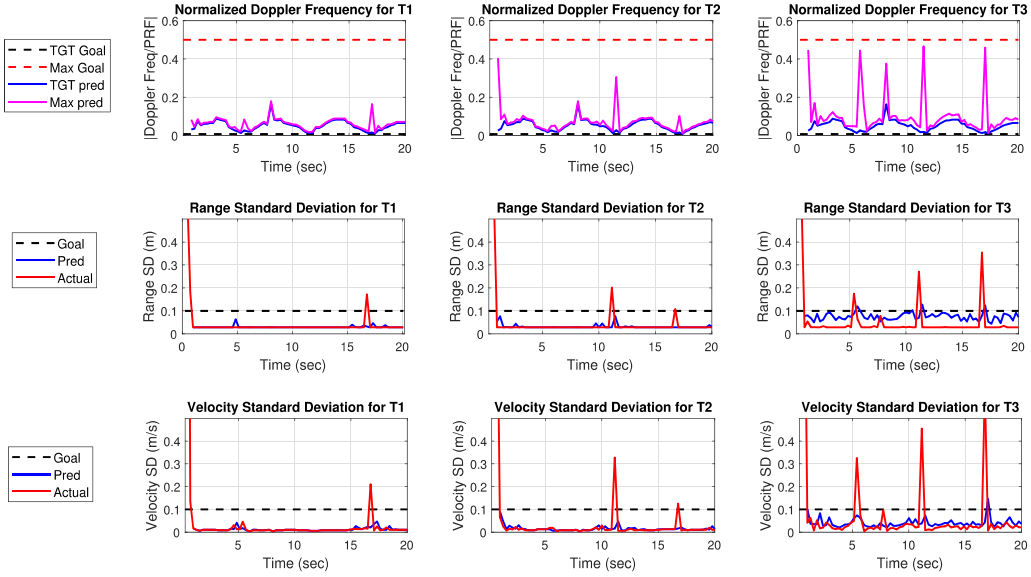
(d) Adaptive Parameters

Figure 7.17: Tracks over Time: **Case 4 [Sim]**

Approved for public release; distribution is unlimited.



(a) Fixed Parameters



(b) Adaptive Parameters

Figure 7.18: Normalized Doppler and Performance Goals: **Case 4 [Sim]**

Approved for public release; distribution is unlimited.

resolution in the measurement space. In looking at the first point where the tracks of T1 and T2 are merged in the fixed parameter case (past ~ 4.5), we see that for the adaptive parameter case the radar parameter selections prevent the targets from merging by maintaining an acceptable SNR level for all three targets at or above $\sim$ 30dB as well as maintaining moderate selections for the parameters resulting in better resolution in the measurement space. Of particular note in Figure 7.16 is at around ~ 5.1 seconds and ~ 8 seconds where we see that the radar selects a low f_{PRF} and high B values. At ~ 5.1, we can see from Figure 7.17 (a) & (b) that T1 and T2 are closely spaced in range, all three targets are getting closer to the zero-Doppler region, and all three targets have a drop in SNR. With these effects in consideration, the requirement for improved performance outweighs resource saving in order resolve the targets. At ~ 8 seconds, we see that a low f_{PRF} and high B value are selected again. In viewing Figure 7.17 (c) & (d) again These selections correspond to the lack of separation between T1 and T2 in range and T3 moving away from the radar. We also note that when all the targets enter the zero-Doppler region resulting in a low SNR response and an increased error in the velocity track, the radar increases the value of N_{p} and τ in order to further improve the Doppler resolution.

In addition to this, when observing Figure 7.18 it can be seen that the range and velocity error remains below the goal line for the adaptive parameter case with error spikes occurring when the target enters the nulled zero Doppler. We further note that the range and velocity standard deviation for target 3 remains below the goal line even though it is the furthest from the radar. In comparing the range and velocity standard deviation results from the fixed parameter case to the adaptive parameter case, we see that our error increases sharply when targets 1 and 2 are merged. In

Approved for public release; distribution is unlimited.

conjunction with the track merge, we also obtain higher errors for target 3 in the fixed parameter case than the adaptive parameter case, specifically in our velocity standard deviation. Also, we note that the adaptive parameter case focuses on not exceeding the maximum velocity goal to keep the targets from being Doppler aliased while the fixed parameter case does not.

Overall, through each of the different cases presented, we were able to obtain better performance for the adaptive parameter cases versus the fixed parameter cases while minimizing the amount of measurement resources used. In addition to this, a real-time experimental verification of an adaptive radar framework for a multiple target environment was investigated to provide further substantiation to the advantage of an adaptive framework for multiple target tracking. Lastly, the utility of a weighted multi-objective penalty function optimization approach for sensor parameter selection in a multiple target environment was demonstrated through which the impact of target prioritization was shown and its benefit in a multiple target environment with changing performance concerns captured.

7.4 Conclusion

From these results, we can conclude that through the development of a FAR-MTT framework including the setup of the cost functions, we were able to see the benefits of adaptive radar parameters in a multiple target environment in comparison to keeping those parameters fixed. Through considerations for velocity and range performance as well as resource consumption and energy usage, the FAR-MTT framework was able to gain acceptable track performance for each target and maintain target separation while using minimal resources. Furthermore an element of target prioritization was

Approved for public release; distribution is unlimited.

captured through the weighted goal programming method in which the performance of targets with lower SNR were prioritized/favored when selecting the "optimal" sensing parameters. Through both simulation and experimental examples, we were able to demonstrate that an extension of the FAR-STT framework to the multiple target environment can provide improved performance to the problem of multiple target tracking. This improved performance was reflected through avoiding track merging by increasing resolution in the measurement space and saving measurement resources while achieving acceptable performance when necessary. Currently, this example is limited to a single mono-static sensor setup however, additional work is being investigated in extending the framework to a multi-static sensor setup.

Approved for public release; distribution is unlimited.

Chapter 8: Fully Adaptive Radar for Multi-Task Radar Functionality

This chapter discusses the development of a hierarchical fully adaptive radar multiple task framework. The work in this chapter applies a HFAR framework to a system needing to engage in balancing multiple tasks: target tracking, classification and target intent discernment (*"friend", "possible foe", or "foe"*). The work presents two frameworks, the first will be termed the *Hierarchical Fully Adaptive Radar for Task Flexibility* (HFAR-TF) and the second is the *Hierarchical Fully Adaptive Radar for Autonomous Decision Making* (HFAR-ADM). HFAR-TF is comprised of two first-tier PAC's along with a central processor that are separately performing tracking and classification along with one second-tier PAC discerning whether to favor improving tracking performance or improving classification performance. HFAR-ADM is similar to HFAR-TF because it also has two first-tier PAC's with a central processor and a second-tier PAC, however the second-tier PAC for HFAR-ADM performs target intent discernment as well as task performance evaluation. Simulations of both frameworks are used to show the benefit of including adaptability in performing multiple radar tasks through the inclusion of "cognitive" decision making and sensor parameter selection properties.

Approved for public release; distribution is unlimited.

The remainder of this chapter is organized as follows. Section 8.1 introduces the problem of a multifunction radar or a radar charged with performing multiple tasks and the motivation behind using fully adaptive methods. Section 8.2 reviews the approach for applying an adaptive radar framework to a radar engaged in multiple tasks by first discussing the two frameworks to be implemented (HFAR-TF & HFAR-ADM), a brief review of the tracking and classification methods implemented, a review of the optimization and decision making employed in the frameworks, discussion on the simulation setup and details on the environment modeling. Section 8.3 presents and discusses the simulation results. Concluding remarks on the work are then made in Section 8.4.

8.1 Motivation

Cognitive radar approaches have allowed radar systems to adapt to their perceived environments whether it be to a spectrally congested or contested environment, to mitigate the effects of clutter, to improve target detection, to improve target tracking, or to manage and forecast system resource usage [4, 89, 116, 117, 118]. An implementation of such a system for tracking has been developed and is known as the FAR framework. The FAR framework [4] provides a method for emulating the PAC of cognition initially developed by Fuster [42], and adapts its sensing based on its perception the environment. The perceived environment is then understood and evaluated according to system performance metrics and goals (performance cost) as well as resource consumption (measurement cost). This FAR framework has been applied to the radar tasks of target detection and tracking [4, 77, 119], classification

Approved for public release; distribution is unlimited.

[72] and synthetic aperture radar imaging [57]. In an attempt to capture the hierarchical nature of the brain, modeled through Fuster's hierarchical cognitive model, the HFAR framework was developed as an extension of the existing FAR framework [28]. The HFAR framework contains multiple PACs and was developed to be applied to the problem of target tracking with distributed sensors, however the hierarchical nature of the framework can be extended to many different problems.

The issue of task scheduling, performing multiple tasks, and resource management is a long existing problem in radar research. The problem of a radar system performing multiple functions can be proposed as a task scheduling problem in which a multifunction radar needs to perform several tasks with differing parameters such as tracking, detection and surveillance in a given timeline such as in [120]. The problem can also be formed as one of resource management in which radar resources such as energy budget, revisit/update time, or frequency bands need to be prioritized and assigned to each task such as in [121, 122, 123]. In [124, 125] a similar view of multiple task radar performance is presented, where an adaptive technique was developed for obtaining tracking or classification measurements to optimize system performance. The implementation used by[124, 125], only took advantage of a single PAC for jointly estimating the target kinematic state and class. In most of these multiple task problems, a series of different tasks are defined and the focus is on placing priority on tasks and assigning portions of the radar timeline to these tasks. The tasks defined are usually considered to be a variety of abstract radar tasks through which less emphasis is placed on actual task performance and instead on resource consumption for the abstract task. The main high-level problems to be addressed in a radar system engaging in multiple radar tasks are resource consumption and task

Approved for public release; distribution is unlimited.

performance. In taking this high-level outlook on a radar engaging in multiple task, the FAR framework seems like a capable implementation for engaging in this as the core components of the framework focus on how to minimize radar resource usage while still being able to obtain optimal task performance.

The cognitive implementation in this work differ from the conventional cognition implementations. In [126], Hakyin discusses the two major types of cognitive implementations for a cognitive radar network as distributed and centralized. Distributed cognition is when the individual radar components as well as the central base station are all cognitive and centralized cognition is defined as confining cognition to the central base station. Extrapolating these definitions from cognitive radar networks to Fuster's cognitive PACs, we can see that a hierarchical nature of Fuster's PACs tends to be more distributed in nature through which each PAC is informed by its local feedback and is connected to other PACs. One issue with this however, is that each PAC has to be aware of the information processed by other PACs which creates a level of inter-dependency between adjacent PACs. One way to resolve this is to limit same-level tier PACs to a central mechanism in order to control the flow of information and remove the dependency of adjacent PACs. In addition to this, the central component can be used to sort information such that only the relevant information is transferred and shared between PACs, while also enabling communication and cooperation with the lower tiers.

In previous HFAR multiple task work, a hierarchical structure was implemented with two first tier PACs separately performing tracking and classification along with one second-tier PAC determining target intent [59]. This work focuses on applying a version of the HFAR framework to a system attempting to perform multiple tasks:

Approved for public release; distribution is unlimited.

target tracking, classification and target intent identification (*"friend", "possible foe", or "foe"*).

8.2 Approach

In the method used in this chapter, we present two different architectures for a system performing multiple radar tasks. Both architectures are hybrid distributed, centralized hierarchical fully adaptive radar frameworks and allow for the adaptation of sensor parameters when performing tracking while classification is permitted through use of a static classifier.

The first architecture is the HFAR-TF, which is comprised of two first-tier PACs that are each separately performing tracking and classification with a shared centralized processor and one second-tier PAC that discerns whether to favor improving track performance or improving classification performance.

The second architecture is the HFAR-ADM, which is similar to HFAR-TF in that it also has two first-tier PACs with a centralized processor for the first tier PACs and a second tier PAC; however this second-tier PAC performs task intent discernment and uses the task discernment to set expected task performance.

The centralized processor allows for a single central control mechanism for transferring sensor data to each of the first tier PACs in order to maintain the separation between the PACs and a centralized sensor parameter selection. The first-tier PACs receive data from the centralized processor and pass their perceptions up to the second-tier PAC. The second-tier PAC receives the perceptions and performance goals from the first tier PAC's and passes hardware sensing requirements down to the centralized processor. Each PAC determines its next action based on its current

Approved for public release; distribution is unlimited.

perception, performance goals and sensor costs. In this paper, we present simulation examples of both architectures along with a baseline to evaluate and demonstrate the performance benefits of a cognitive framework in performing multiple radar tasks.

8.2.1 Framework Implementation

HFAR-TF

The HFAR-TF framework is based on separate first-tier PACs for tracking and classification along with a second-tier PAC for discerning whether to favor improving track performance or improving classification performance and responding appropriately. The central processor of the framework provides the radar hardware interface for transferring sensor data to each of the first tier PAC's and for selecting sensor parameters used for future measurements based on requests from the first-tier PACs and second-tier PAC influence. A figure outlining the architecture is shown in Figure 8.1.

For this framework, sensor data $\mathbf{z}_k$ at time step k is sent to the tracking PAC in order to implement processing such as range-Doppler map creation, target detection, target range/velocity estimation, and determination of future measurement parameters $\boldsymbol{\theta}_{t,k}$ to optimize tracking performance using minimal sensing resources. The range-pulse matrix RP involved in the range-Doppler map creation is sent to the Classification PAC along with the extracted target range/velocity estimates $\hat{\mathbf{x}}_k$.

The Classification PAC extracts the target feature vectors $\tilde{\mathbf{z}}_k$ needed to determine target state classification $\hat{c}_k$ from this data and its associated entropy $\hat{h}_k$. The entropy metric's evolution incorporates a-priori information on the probability of state transitions to represent the degradation in classification confidence that occurs when

Approved for public release; distribution is unlimited.

the classification is not updated. In the example reported here, a classification measurement can be performed only when the scene is observed using sensor parameters $\boldsymbol{\theta}_c$ (which are fixed in time) for which feature vectors $\tilde{\mathbf{z}}_k$ are available from training data; if sensor parameters deviate from $\boldsymbol{\theta}_c$, the current target classification statistics are updated using an assumed model of the class transition. Tracking, in contrast, is always performed regardless of sensor parameters, and is expected to be optimized if state-dependent sensor parameters $\boldsymbol{\theta}_{t,k}$ are used (more details on how $\boldsymbol{\theta}_{t,k}$ is determined can be found in [7]). T The 2^{nd}-tier task priority processor accepts current information from both the Tracking and Classification PACs and based on this information selects (by specifying output $\boldsymbol{\eta}^*_{\boldsymbol{TF}}$) whether to favor classification or tracking, respectively, that result in a selection of sensor parameters corresponding to $\boldsymbol{\theta}_c$ (classification parameters) or $\boldsymbol{\theta}_{t,k}$ (optimal tracking sensor parameters with low resource usage) by the centralized processor. The selected set of sensor parameters $\boldsymbol{\theta}^*_k$ is then used by the sensor to observe the scene.

HFAR-ADM

Similar to the HFAR-TF framework, the HFAR-ADM framework is also based on separate 1^{st}-tier PACs for the tracking and classification tasks along with a 2^{nd}-tier PAC. However, the 2^{nd}-tier PAC is used for discerning target intent and responding appropriately. The central processor of the framework provides the radar hardware interface for transferring sensor data to each of the 1^{st}-tier PAC's and for selecting sensor parameters used for future measurements based on requests from the 1^{st}-tier PACs and on the discerned target intent. A figure outlining the architecture is shown in Figure 8.2.

Approved for public release; distribution is unlimited.

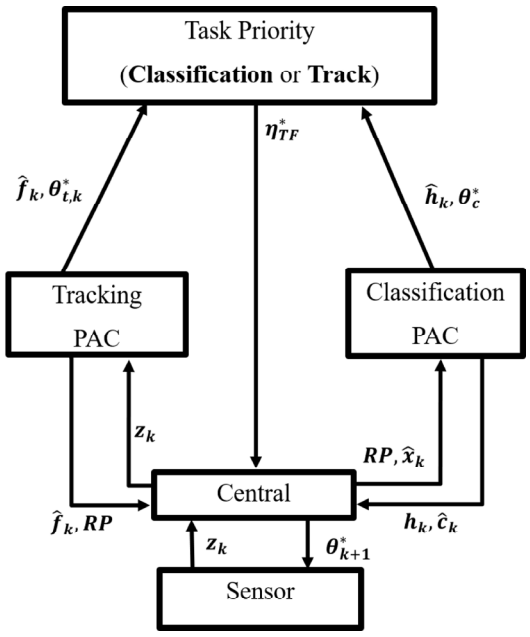


Figure 8.1: HFAR-TF Diagram

The functional flow of the HFAR-ADM framework remains the same as the HFAR-TF framework, however the difference between the two lies in the implementation of the 2^{nd}-tier. In HFAR-ADM, the 2^{nd}-tier target intent processor accepts current information from both the Tracking and Classification PACs and based on this information selects (by specifying output $\boldsymbol{I}$) from three sensing performance states called *"friend"*, *"uncertain"*, and *"foe"* performance, respectively, that vary measurement error goals in entropy H^*, target range σ_R^*, and target velocity σ_V^*. These performance goals influence future radar parameter selections since the process used to select parameters balances sensor resource use versus the desired task performance. The resulting parameter selection is done in the centralized processor and the selected set of sensor parameters θ_k^* is then used by the sensor to observe the scene.

Approved for public release; distribution is unlimited.

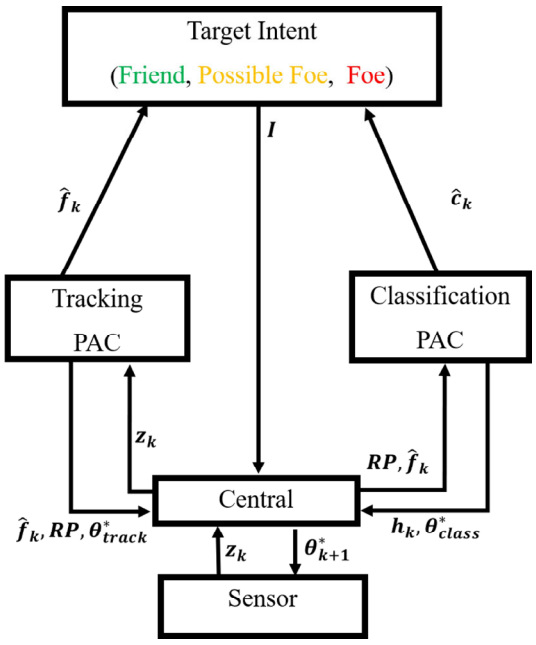


Figure 8.2: HFAR-ADM Diagram

8.2.2 FAR Multi-Task Tracking

The HFAR architecture is can be used for adaptive target tracking systems. Detection for track initiation and the corresponding track association is accomplished by use of the MAP-PF [58] mentioned in Chapter 2.1.3. For the work presented in this chapter, tracking is limited to that of a single target. Based on the Markov motion model, the predicted density $f^-(\mathbf{x}_k) \equiv f(\mathbf{x}_k|\mathbf{Z}_{k-1};\boldsymbol{\Theta}_k)$ is calculated using the Chapman-Kolmogorov equation. The posterior density $f^+(\boldsymbol{x}_k) \equiv f(\mathbf{x}_k|\mathbf{Z}_k;\boldsymbol{\Theta}_k)$ is found using Bayes' rule [15]. The recursion follows as such, with

$$f^+(\mathbf{x}_0) = q(\mathbf{x}_0) \tag{8.1}$$

Approved for public release; distribution is unlimited.

initializing, and then follows [4].

$$\begin{aligned} f^{-}(\mathbf{x}_k) &= f(\mathbf{x}_k|\mathbf{Z}_{k-1};\boldsymbol{\Theta}_k) \\ &= \int q(\mathbf{x}_k|\mathbf{x}_{k-1};\boldsymbol{\theta}_k) f^{+}(\mathbf{x}_{k-1}) d\mathbf{x}_{k-1} \end{aligned} \tag{8.2}$$

$$\begin{aligned} f^{+}(\mathbf{x}_k) &\equiv f(\mathbf{x}_k|\mathbf{Z}_k;\boldsymbol{\Theta}_k) \\ &= \frac{f(\mathbf{x}_k|\mathbf{x}_k;\boldsymbol{\theta}_k) f^{-}(\mathbf{x}_k)}{\int f(\mathbf{x}_k|\mathbf{x}_k;\boldsymbol{\theta}_k) f^{-}(\mathbf{x}_k) d\mathbf{x}_k} \end{aligned} \tag{8.3}$$

We also assume the Markov motion model with an initial target state probability distribution $q(\mathbf{x}_0)$ and a state transition probability $q(\mathbf{x}_k|\mathbf{x}_{k-1};\boldsymbol{\theta}_k)$. The sensor measurement model is then described by the likelihood function, $f(\mathbf{z}_k|\mathbf{x}_k;\boldsymbol{\theta}_k)$ [4].

The system objective is to estimate the target state in the scene at time t_k using the perceptions and sensor parameters up until this time, however given the assumed Markov nature of the motion, only the prior perceptions and sensor parameter selections are needed. The target state estimates are found by minimizing the conditional Bayes risk [4]. The target state is formatted as:

$$\hat{\mathbf{x}}_k = \begin{bmatrix} r_k \\ \dot{r}_k \\ \chi_k \end{bmatrix} \tag{8.4}$$

where r_k is the position state estimate, $\dot{r}_k$ is the range-rate state estimate, and χ_k is the SNR state estimate.

The formulation of the perception at time k is the posterior density $f^{+}(\mathbf{x}_k|\mathbf{Z}_k)$ with the target state $\mathbf{x}_k$ and measurements $\mathbf{Z}_k$. To calculate the state estimate $\hat{\mathbf{x}}_k$, a linear Kalman filter is used. The range and velocity estimates follow a nearly constant velocity model. The full target motion model is as follows :

$$\mathbf{x}_k = \boldsymbol{F}(\boldsymbol{\theta}_k)\mathbf{x}_{k-1} + \boldsymbol{\nu}_k \tag{8.5}$$

Approved for public release; distribution is unlimited.

where $\boldsymbol{F}(\boldsymbol{\theta}_k)$ is the state transition matrix

$$\boldsymbol{F}(\boldsymbol{\theta}_k) = \begin{bmatrix} 1 & \Delta t(\boldsymbol{\theta}_k) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \beta(\boldsymbol{\theta}_k) \end{bmatrix} \tag{8.6}$$

with a revisit time that can be affected by the sensor parameter selection $\Delta t(\boldsymbol{\theta}_k)$ and a scalar factor:

$$\beta(\boldsymbol{\theta}_k) = \frac{N_{p,k}}{N_{p,k}} \frac{\tau_k}{\tau_{k-1}} \frac{P_{T,k}}{P_{T,k}} \tag{8.7}$$

which measures how changes in the sensor parameters $\boldsymbol{\theta}_k$ affect SNR [28]. The process noise $\boldsymbol{\nu}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$ has covariance Q:

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_\rho^2 \Delta t^2 & 0 & 0 \\ 0 & \sigma_{\dot{\rho}}^2 \sqrt{\Delta t} & 0 \\ 0 & 0 & \sigma_\chi^2 \Delta t \end{bmatrix} \tag{8.8}$$

where σ_ρ, $\sigma_{\dot{\rho}}$ and σ_s are empirically determined constants [28].

The measured states can also be represented as a function of the state estimates with the following measurement model:

$$\mathbf{z}_k = \boldsymbol{H}\mathbf{x}_k + \boldsymbol{\eta}_k \tag{8.9}$$

where $\boldsymbol{H}$ is the measurement model matrix

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2\frac{f_c}{c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.10}$$

The measurement noise $\boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ has covariance $\mathbf{R}$ which is the inverse of an empirical derivation of the Fisher information matrix:

$$\boldsymbol{J}_y(\chi; \boldsymbol{\theta}) \approx \begin{bmatrix} C_R \chi (\frac{2B}{c})^2 & 0 & 0 \\ 0 & C_f \chi (\frac{N_p}{f_{PRF}})^2 & 0 \\ 0 & 0 & C_\chi N_p \tau P_T \end{bmatrix} \tag{8.11}$$

with C_R, C_f and C_s as constants [28].

Approved for public release; distribution is unlimited.

8.2.3 FAR Multi-Task Classification

The classifier is used to determine what motion the target is currently performing by using the target's microDoppler response as feature vectors. The three class options are walking, jogging and running. The classifier is trained using fixed high performance sensor parameters $\boldsymbol{\theta}_c$ that were chosen to maximize correct classification probability. As such using the classifier parameters will naturally incur a high measurement cost C_M in the decision making process. The classifier itself is trained using the Doppler spectrum of the target as feature vectors. The feature vectors are acquired from the measured range-Doppler maps by summing across 30 range bins centered at the peak response. This collapses the 2-D range-Doppler map into a 1-D Doppler spectrum used as feature vectors. A *multiple discriminant analysis* (MDA) [127] is performed on the feature vectors to generate a matrix that projects the full length Doppler spectrum feature vector to the 2 dimensional feature space. In this work, the feature vector for the classification PAC can be represented as $\tilde{\mathbf{z}}_k$ with its likelihood function given as $f\left(\tilde{\mathbf{z}}_k|c_k = i; \theta_k = 1\right)$. Figure 8.3 shows the training feature vectors projected into feature space an from this we can see that the training data shows good separation between all three classes.

Another important task the classification PAC performs is calculating the posterior entropy H^+ and the predicted entropy H^- used in the decision making algorithms. The entropy expressions used were discussed in more detail and derived in [72]. However, we summarize the expressions here beginning with the posterior and predicted *probability mass function* (pmf). For each time step, in which a classification

Approved for public release; distribution is unlimited.

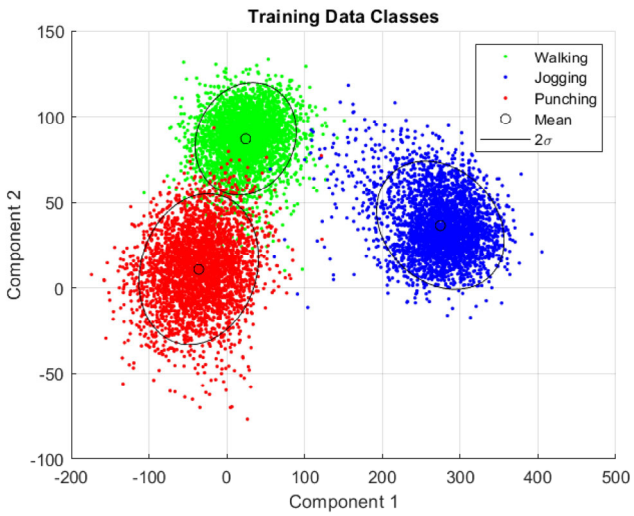


Figure 8.3: Training Data Feature Space

measurement is taken, the posterior pmf $f_i^+(c_k)$ is updated using

$$f_i^+(c_k) = \frac{f(\tilde{\mathbf{z}}_k|c_k = i;\theta_k = 1)\, f_i^-(c_k)}{\sum_{j=1}^{3} f(\tilde{\mathbf{z}}_k|c_k = j;\theta_k = 1)\, f_j^-(c_k)}. \tag{8.12}$$

where $f(\tilde{\mathbf{z}}_k|c_k = i;\theta_k = 1)$ is the likelihood function which is assumed to be a multivariate Gaussian density of $\tilde{\mathbf{z}}_k$ with mean $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma_i}$ determined by the target class i resulting in $f(\tilde{\mathbf{z}}_k|c_k = i;\theta_k = 1) \sim \mathcal{N}(\tilde{\mathbf{z}}_k;\boldsymbol{\mu}_i,\boldsymbol{\Sigma_i})$.

If a classification measurement is not taken, the posterior pmf $f_i^+(c_k)$ takes the values of the predicted pmf $(f_i^-(c_k))$ given by

$$f_i^-(c_k) = \sum_{j=1}^{3} L_{ij} f_j^+(c_{k-1}) \tag{8.13}$$

where $\mathbf{L}$ is a class transition matrix given by

$$\boldsymbol{L} = \begin{bmatrix} 0.99 & 0.005 & 0.005 \\ 0.005 & 0.99 & 0.005 \\ 0.005 & 0.005 & 0.99 \end{bmatrix} \tag{8.14}$$

and models the likelihood of the target switching between the motion classes of walking, jogging or running.

Approved for public release; distribution is unlimited.

The posterior entropy is then given by

$$H^{+}\left(c_k;\tilde{\mathbf{Z}}_k\right)=-\sum_{i=1}^{3}f_i^{+}\left(c_k\right)\log_2\left(f_i^{+}\left(c_k\right)\right) \tag{8.15}$$

where c_k is a scalar class variable, $\tilde{\mathbf{Z}}_k$ is the measured feature vector and $f_i^{+}\left(c_k\right)$ is the posterior pmf. In this case $c_k \epsilon \{1,2,3\}$ where $c_k=1$ is walking, $c_k=2$ is jogging, and $c_k=3$ is punching.

The predicted entropy is given by

$$H^{-}\left(c_k;\tilde{\mathbf{Z}}_{k-1}\right)=-\sum_{i=1}^{3}f_i^{-}\left(c_k\right)\log_2\left(f_i^{-}\left(c_k\right)\right) \tag{8.16}$$

and grows each time step a classification measurement is not taken due to the effect of the class transition matrix $\boldsymbol{L}$ on the predicted pmf. The current entropy H^{+} is used to evaluate the MDA-based classifier performance by keeping it under a goal entropy H^{*}. The predicted entropy H^{-} is used in the decision making process to determining what sensor parameters to be used for the next measurement.

8.2.4 Optimization and Decision Making Discussion

In this work both an optimization and decision making scheme is developed in order to permit parameter selection.

FAR Optimization

Prior implementations of the FAR/HFAR framework [28, 7] implement measurement and performance costs balanced using an executive optimization in order to enable parameter selection. Further details of the cost function setup is discussed in [7] and were reviewed in Chapter 5.2.1. It is these cost functions that will be used in order to perform parameter selection. The measurement cost function is meant

Approved for public release; distribution is unlimited.

to capture the notion of resource consumption for a radar system according to a high-level mission such as radar timeline minimization, low probability of intercept or energy minimization. The measurement cost function takes the following form:

$$C_M(\boldsymbol{\theta}_k) = \sum_{i=1}^{5} w_i \left| \frac{\theta_{k_i} - \theta_i^{\dagger}}{\theta_i^*} \right| \tag{8.17}$$

where $\theta_{k,i}$ represents the ith sensor parameter at time step k, while $\theta_i^{\dagger}$ is a value for this parameter deemed desirable and w_i is a set of weights used to assign costs to distinct parameters. The measurement cost for our example, captures the goal of radar timeline minimization, through which we want to complete a multitude of tasks within a set amount of time. This application requires short dwells and low processing times.

The performance cost function is meant to be a realization of radar task performance such as range and velocity standard deviation for tracking. The performance cost function takes the following form:

$$\begin{aligned} C_P(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k) =& \frac{(\sigma_R(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k) - \sigma_R^*)}{\sigma_R^*} u(\sigma_R(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k) - \sigma_R^*) \\ &+ \frac{(\sigma_V(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k) - \sigma_V^*)}{\sigma_V^*} u(\sigma_V(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k) - \sigma_V^*) \end{aligned} \tag{8.18}$$

where σ_R^* and velocity σ_V^* are the goals for the measurement errors in range and velocity, respectively; and $\sigma_R(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k)$ and $\sigma_V(\boldsymbol{\theta}_k; \hat{\mathbf{x}}_k)$ are the expected track errors for range and velocity respectively. It is noted that $C_P(\cdot)$ is not explicitly used in the optimization performed in the executive processor; it is rather computed separately as a constraint for the radar measurement performance as compared to the assigned goals.

Approved for public release; distribution is unlimited.

Since the classifier is constrained to a fixed set of parameters to classify, the parameter adaptation is only considered for tracking. The optimization that takes place to select the set of parameters for optimal tracking performance using minimal sensor resources then takes the following form:

$$\boldsymbol{\theta}_k = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_k} C_M(\boldsymbol{\theta}_k)\ s.t. \begin{cases} \sigma_R(\boldsymbol{\theta};\hat{\mathbf{x}}_k) \leq \sigma_R^\dagger \\ \sigma_V(\boldsymbol{\theta};\hat{\mathbf{x}}_k) \leq \sigma_V^\dagger \end{cases}$$

where

$$\boldsymbol{\Theta}_k = \boldsymbol{\theta}_k \in \boldsymbol{Z}^N : \begin{cases} \theta_{k,1}^{lb}(\hat{\mathbf{x}}_k) \leq \theta_{k,1} \leq \theta_{k,1}^{ub} \\ \theta_2^{lb} \leq \theta_{k,2} \leq \theta_2^{ub} \\ \theta_3^{lb} \leq \theta_{k,3} \leq \theta_3^{ub} \\ \theta_4^{lb} \leq \theta_{k,4} \leq \theta_4^{ub} \end{cases} \tag{8.19}$$

where $\boldsymbol{\Theta}_k$ is the set of available waveform parameters. Here we see that the optimization here enforces the desired radar performance through the constraints applied while seeking the minimum measurement cost. **Note**: In implementation, the parameter selections for the sensor parameters are optimized based on a set of bounded continuous values but the selections are rounded to correspond to discrete indices.

The optimized solution to this objective function results in sensor parameter selection for a LFM waveform used for environment sensing with possible adapted waveform parameters being: bandwidth-B, pulse repetition frequency-f_{PRF} , pulse width-τ, number of pulses-N_{p}, and transmit power-P_t.

Decision Making

In addition to the optimization used for optimal tracking sensor parameters with minimal resource usage, a decision making scheme was developed for both HFAR-TF and HFAR-ADM. The decision making scheme is used in order to make the final sensor parameter selection to send to sensor in order to observe the scene for a given

Approved for public release; distribution is unlimited.

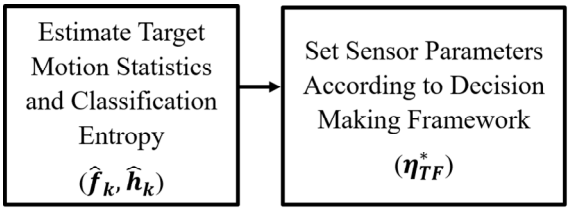


Figure 8.4: HFAR-TF Decision Making Process

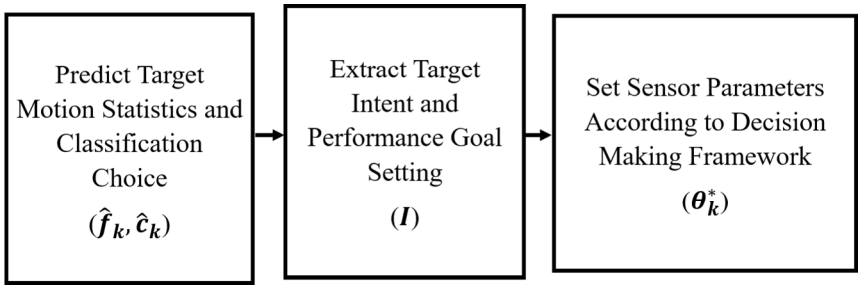


Figure 8.5: HFAR-ADM Decision Making Process

task. The decision making scheme is primarily considered with three metrics: tracking performance, resource usage and classification performance. The set of heuristics used for the decision making for HFAR-TF and HFAR-ADM are as outlined in Algorithm 1.

The process for decision making is outlined in Figures 8.4 and 8.5 for HFAR-TF and HFAR-ADM, respectively.

Target Priority Setting

For HFAR-ADM, an additional target priority aspect was added in order to adjust the goals of the tracking (velocity and range standard deviation) and classification (entropy) performance. The adjustment of the goals takes place before we enter the

Approved for public release; distribution is unlimited.

Table 8.1: Target Priority Table

		Tracker			
		Fast		Slow	
		Close	Far	Close	Far
Classifier	Walk	Possible Foe	Possible Foe	Possible Foe	Friend
	Jogging	Foe	Foe	Possible Foe	Possible Foe
	Punching	Foe	Possible Foe	Foe	Possible Foe

decision making scheme, through which target priority indicated by extracted target intent of friend, potential foe and foe. The setting of target priority uses the predicted classification and target state estimates according to Table 8.1.

By defining subjective metrics for indicating a fast, slow, far and close target based on the target state estimates, we can then use the target priority setting in Table 8.1 to set the goals for the track and classification performance. The goal setting is as follows:

- Friend
  - Favor performance relaxation
    - Increase track goals: $\sigma^*_{R,inc}, \sigma^*_{V,inc}$
    - Keep current classification goals: H^*
- Potential Foe
  - Favor performance maintenance
  - Keep current track goals: σ^*_R, σ^*_V

Approved for public release; distribution is unlimited.

  - Decrease classification goals: H^*_{dec}

- Foe
  - Favor improved performance
  - Decrease track goals: $\sigma^*_{R,dec}, \sigma^*_{V,dec}$
  - Increase classification goals: H^*_{inc}

The target priority settings are set according to the performance we wish to obtain when the target is classified as a friend, potential foe or foe. For targets that are considered a friend, we want to relax our tracking performance goals while still maintaining our default tracking performance. This is due to our choice of reducing resource usage for observing a friend but still needing to gain more information to discern if the friend intent may change.

For targets that are considered a potential foe, we want to maintain our default tracking performance and improve our classification performance through decreasing its goals. This is due to our choice of wanting to keep informed about the target's state as well as needing to gain more information to discern if the friend intent may change. For targets that are considered a foe, we want to improve our tracking performance through decreasing its goals and relax our classification performance goals. This is due to our choice of wanting increase knowledge of the target's state but not needing more information to verify if a target is still a foe.

The framework specifically determines sensor parameters θ_{k+1} for the next measurement interval through use of the function:

$$C_{DM}(H_k, \theta_c, \theta_{t,k}) = w_1 \frac{H^- - H^*_{pred}}{H^*_{pred}} + w_2 \Delta_{tc} \tag{8.20}$$

Approved for public release; distribution is unlimited.

$$\Delta_{tc} = \begin{cases} \frac{C_E(\boldsymbol{\theta}_c;\hat{\mathbf{x}}_k) - C_E(\boldsymbol{\theta}_{t,k};\mathbf{x}_k)}{C_E(\boldsymbol{\theta}_{t,k};\hat{\mathbf{x}}_k)}, \text{adaptive case} \\ \frac{C_P(\boldsymbol{\theta}_c;\hat{\mathbf{x}}_k) - C_P(\boldsymbol{\theta}_{t,k};\mathbf{x}_k)}{C_P(\boldsymbol{\theta}_{t,k};\hat{\mathbf{x}}_k)}, \text{fixed case} \end{cases}$$

which combines a first term for classification performance and a second term for the relative tracking performance between measurements with the optimal tracking or classification sensor parameters. It should be noted that when the adaptive parameter case is chosen, Δ_{tc} evaluates the relative tracking cost as a function of the executive cost when choosing the classification and tracking sensor parameters which counts in measurement resource usage (measurement cost); whereas when the fixed parameter case is chosen, Δ_{tc} evaluates the relative tracking performance purely as a function of the performance cost. The weights w_1 and w_2 for these terms set the relative importance of these terms. The weight w_1 is limited to values less than 0 and the weight w_2 is limited to values greater than 0. A classification measurement is performed (i.e. $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_c$) on the next measurement interval if $C_{ADM} \leq 0$, otherwise a tracking measurement ($\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_{t,k}$) occurs.

This results in the function favoring a tracking measurement when the predicted entropy is seen to be below the desired goal (i.e. H_k less than the desired H^* value) and when the executive tracking cost (sum of measurement and processor cost) using the optimal tracking sensor parameters is less than the tracking performance using the optimal classification sensor parameters (i.e. $C_E(\boldsymbol{\theta}_c;\hat{\mathbf{x}}_k) > C_E(\boldsymbol{\theta}_{t,k};\mathbf{x}_k)$). However, when there is degradation in the predicted entropy in comparison to the desired goal (i.e. H_k greater than the desired H^* value) and when the tracking performance when using the optimal classification sensor parameters is comparable or less than when using the tracking sensor parameters (i.e. $C_E(\boldsymbol{\theta}_c;\hat{\mathbf{x}}_k) < C_E(\boldsymbol{\theta}_{t,k};\hat{\mathbf{x}}_k)$), the function results in favoring a classification measurement.

Approved for public release; distribution is unlimited.

Algorithm 1 Decision Making Structure

procedure DECISION PRIORITY
  if $\sigma^2_{R,V} > \sigma^2_t$ **then** ▷ σ^2_t is a threshold
    $\boldsymbol{\theta}^*_k = \boldsymbol{\theta}_{t,k}$
  else
    $C_{DM}(H_k, \theta_c, \boldsymbol{\theta}_{t,k}) = w_1 \frac{H^- - H^*_{pred}}{H^*_{pred}} + w_2 \Delta_{tc}$
    if $C^{DM}_E \geq 0$ **then**
      $\boldsymbol{\theta}^*_k = \boldsymbol{\theta}_{t,k}$
    else
      $\boldsymbol{\theta}^*_k = \boldsymbol{\theta}_c$

8.2.5 Simulation Setup

The FARMS simulator was used to generate the simulation. The real world effects that were captured in the simulation were random RCS response, reducing power received as a function of the radar range equation and noise. The simulation is of a scene in which a target is moving towards and away (sinusoidal motion) from the radar within a range of 10 meters as seen in Figure 8.6, in which the circles in the silhouette represents a distributed point target for the simulation run. The simulation is a theoretical realization of the utility of the FAR framework which will be eventually compared with the experimental using a millimeter wavelength, multi-static, SDR system at the ElectroScience Laboratory known as the CREW. The operational parameters for the radar in the adaptive and fixed parameter cases are shown in Table 8.2 and Table 8.3, respectively. The parameters for the "no adapt" case are also the parameters in which the classification training data were collected and corresponding projection matrix created for classification. This means that the parameters in Table 8.3 correspond to the fixed set of classification sensor parameters, $\boldsymbol{\theta}_c$ and tracking sensor parameters, $\boldsymbol{\theta}_{t,k}$.

Approved for public release; distribution is unlimited.

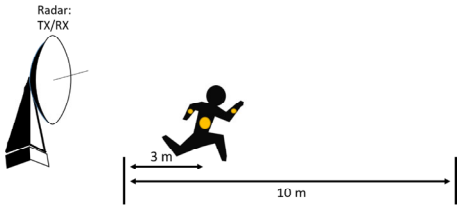


Figure 8.6: Target Motion

For both the *Autonomous Decision Making* (ADM) and *Task Flexibility* (TF) examples, the following holds: the maximum entropy for three classes is $\log_2(3) = 1.5850$; and the default values for the performance goals are set as: $\sigma^*_{R,def} = 0.1$, $\sigma^*_{V,def} = 0.1$ and $H^*_{def} = 0.3$.

For the TF example, we hold the goals fixed at the default values.

For the ADM example, we apply limits to the performance goals adaptations. The performance decrease goals are limited to $\sigma^*_{R,dec} = \frac{\sigma^*_{R,def}}{2}$, $\sigma^*_{V,dec} \geq \frac{\sigma^*_{V,def}}{2}$, and $H^*_{dec} \geq \frac{H^*_{def}}{2}$ and the performance increase goals are limited to $\sigma^*_{R,inc}$,$\sigma^*_{V,inc} \leq 0.5$ for the tracking performance and $H^*_{inc} \leq 0.6$ for the classification performance. The lower limits for the goals were set such in order to avoid unrealistic performance requirements in track error and classification entropy which can possibly limit the feasible solution space. Similarly, the upper limits for the track goals were set such that excessively large track errors were unacceptable to avoid possible loss of track and the upper limits for the entropy were set to be lower than the maximum entropy value for three classes. Lower goals for track error result in better resolution of the target in the measurement space as the ability to track the target becomes imperative, inversely, the higher goals for track error result in a lowering concern for the target position through accepting higher track errors. In a similar sense, lower goals for the

Approved for public release; distribution is unlimited.

Table 8.2: Radar Operating Parameters: Fixed Case

Parameters	**Classification Sensor Values**	**Tracking Sensor Values**
f_{PRF} (khz)	15	7.78
N_{p} (pulses)	512	128
τ (μs)	0.5	0.2
B (MHz)	1000	500

Table 8.3: Radar Operating Parameters: Adapt Case

Parameters	**Number of Steps**	**Value Range**
f_{PRF} (kHz)	10	2-15
N_{p} (pulses)	10	64-640
τ (μs)	10	0.1 - 1
B (MHz)	10	100-1000

classification entropy translate into a higher need to discern the target's class due to lack of information and inversely, the higher goals for the classification entropy result in a reduced concern for determining the target's class through accepting higher entropy values.

8.2.6 Environment Modeling

The simulation scene utilizes three point scatterers to model a human target with microDoppler motions as shown in Figure 8.6. The target is comprised of a central point scatterer with large RCS representing the body surrounded by two point

Approved for public release; distribution is unlimited.

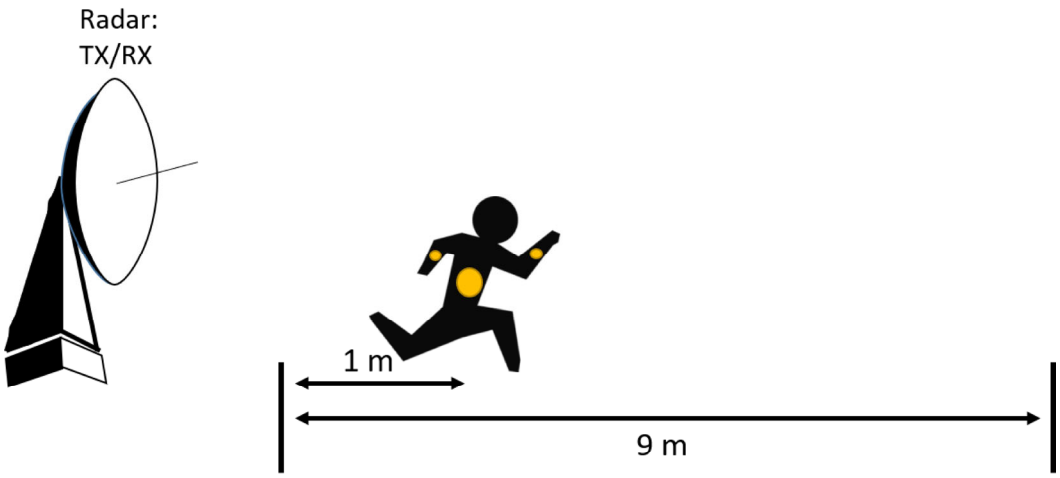


Figure 8.7: Target motion over time

scatterers with lower RCS representing the arms. Smith et al. showed in [128] that modeling a human target's microDoppler with a set of point scatterers is comparable to experimental measurements. Modeling the target as multiple point scatterers with independent motions allows for different types of motions to be more realistically simulated. In particular, three motions are modeled: walking, jogging and punching. The central point scatterer moves at a constant velocity in the walking and jogging cases and is stationary in the punching case. In all cases the side point scatterers oscillate 180 degrees out of phase in downrange, but at different rates and distances depending on which motion is currently taking place. Periodically the target randomly changes which motion it is performing. In Figure 8.7 each plotted line shows a target point scatterer positions with respect to time.

The differences in the target point scatterers with respect to the current motion is clearly seen in Figure 8.7. In this scene the target moves back and forth between 1 m and 8.5 m downrange. Every 4 seconds the target performs a different motion as

Approved for public release; distribution is unlimited.

Table 8.4: Target Scatterer Motion Parameters

Motion Parameters	**Walking**	**Jogging**	**Punching**
Average Velocity (m/s)	0.5	1.5	0
Oscillation rate (rad/s)	1	2	1.5
Oscillation magnitude (m)	0.2	0.3	0.5

labeled in the figure. The three parameters that distinguish the different motion types are the average velocity of the target, and the rate and magnitude of the surrounding point scatterers' oscillations. The parameters for each motion class are listed in Table 8.4. The differences in these parameters create different microDoppler signatures for the three motions allowing for a classifier trained with microDoppler feature vectors as discussed in Section 8.2.3.

8.3 Results

In the work of [4], analysis of track performance for an adaptive radar is viewed using the tracks of the range, velocity, and SNR accompanied by the parameter variation and performance goals in terms of velocity & range standard deviation. In addition to this, in [72], analysis of classification performance is viewed using the entropy (predicted and posterior) of the classes. In an attempt to combine both of these works, [59] uses the track error for range and velocity as well as the entropy for the classification in order to determine whether to take a classification or track measurement providing a combination of entropy and track plots for analysis. In order to begin analyzing the results, a baseline performance needs to be established.

Approved for public release; distribution is unlimited.

In order to view the benefits of an adaptive radar versus a static radar when presented with multiple tasks, two baselines will be used.

The baseline results assume that the radar system has two sets of fixed sensor parameters: one for classification and another for tracking. This choice of baseline was made in order to capture the benefit of an adaptive sensor parameter selection versus a fixed parameter case when engaging in multiple radar tasks. This baseline also allows gives the radar system the flexibility to select another set of parameters specified towards tracking rather than using the classification sensor parameters for both tracking and classification. The figure corresponding to the baseline results is Figure 8.8, through which we can use as a baseline to compare to the adaptive cases of both HFAR-TF and HFAR-ADM since the only difference between the two is an adaptation of the goals based on target intent (ADM) and the parameters themselves are fixed for the baseline.

Figure 8.8 contains three sub-figures: (a), (b), and (c). In sub-figure (a), eleven plots are shown. The first column of three plots are the tracks for the range, velocity and SNR, respectively; the second column of three plots are the goals for the Doppler frequency, range and velocity standard deviation performance which varies according to parameter selection; the third and fourth column of five plots are the parameter adaptations versus time. Zoomed in plots of the range and velocity standard deviation are shown in sub-figure (b) for the predicted and actual standard deviations. Sub-figure (c) displays the predicted and actual entropy results which vary according to parameter selection; the decision of whether to take a measurement using the optimal tracking parameters (track meas.) or to take a measurement using the classification

Approved for public release; distribution is unlimited.

parameters (class & track meas.); and the overlaid classification decision at each point the class changes.

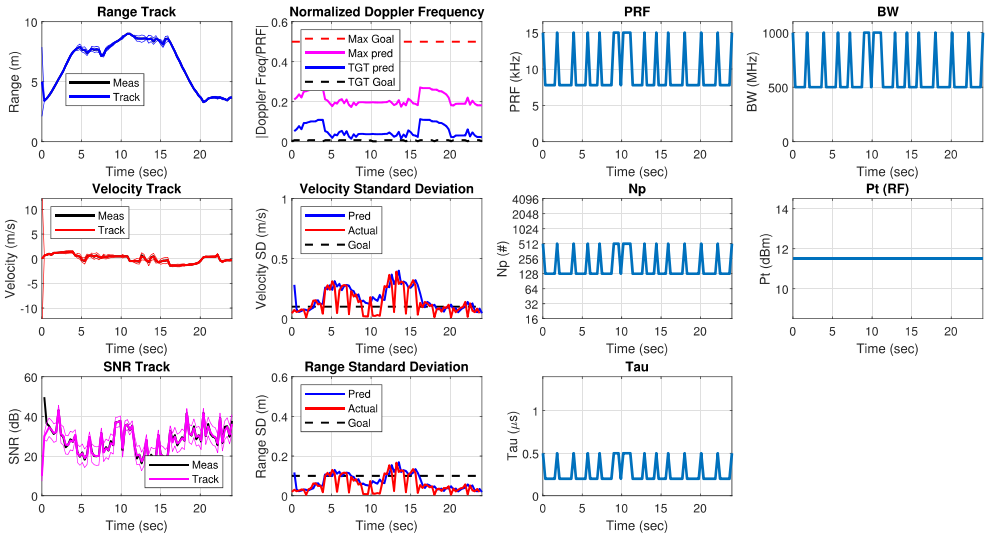
8.3.1 HFAR-TF Results

Comparing the performance between the fixed parameter set performing classification and tracking versus the adaptive parameter set, we can see that better performance is obtained through the adaptive framework. In the fixed parameter case, when viewing Figure 8.8 (a), (b) and (c), we can see that we remain above the goal line for our velocity and range standard deviation. It can also be noted that we obtain a high entropy value at around $\sim$ 10s as well as multiple classifications to lower the entropy.

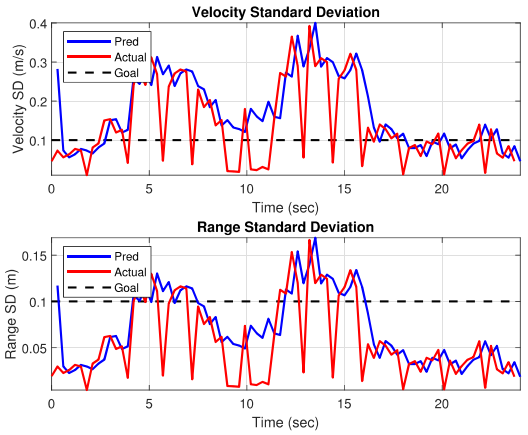
In Figure 8.9 (a) and (b), we that the velocity and range standard deviation performance remains below or at the goal line. In addition to this, in Figure 8.9 (c), we see that classification performance also remains closer to the goal line and when deemed necessary we perform classification.

In comparing the fixed parameter case against the adaptive parameter case, we begin to see the adaptive framework decide between classification and tracking in order to save measurement resources and achieve performance goals. In Figure 8.9 (b), we see that we manage to remain below our velocity and range standard deviation goals and in Figure 8.9 (c) we make an attempt to remain below our entropy goals at most times. The lowest entropy performance for our actual entropy value is obtained when we classify using our classification sensor parameters, otherwise we obtain slightly high predicted and actual entropy performance. However, we notice that at most times, when we predict that our entropy exceeds the goal line significantly,

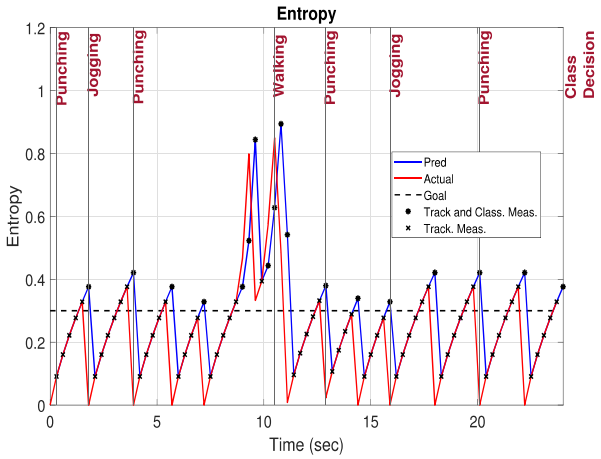
Approved for public release; distribution is unlimited.



(a) Tracks, Track Performance and Parameter vs Time



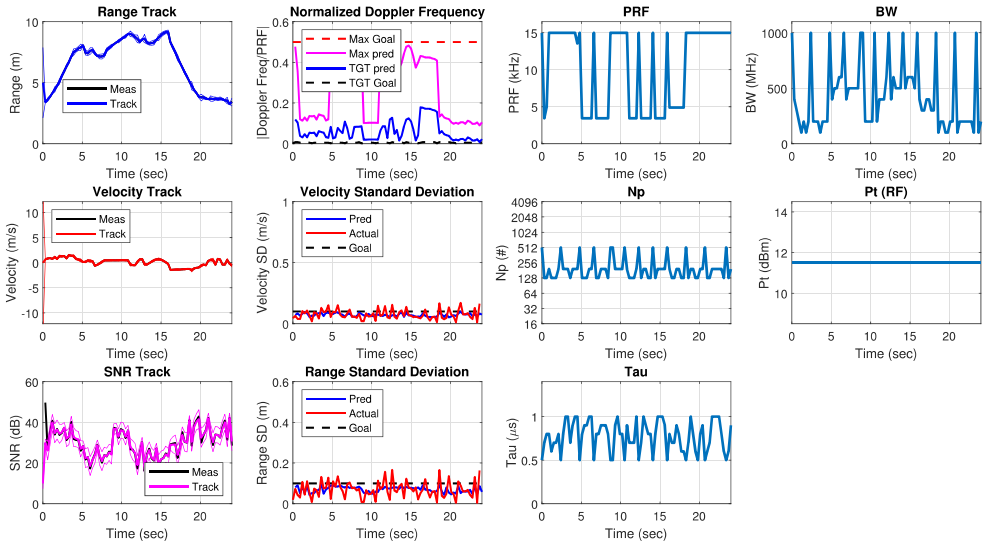
(b) Predicted and Actual Range and Velocity Standard Deviation (zoomed)



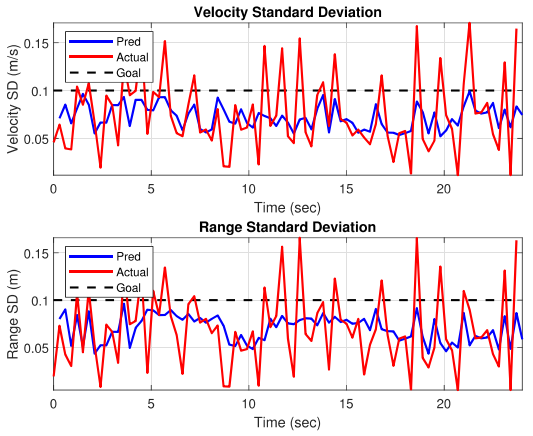
(c) Predicted and Actual Entropy

Figure 8.8: Fixed Parameter: Track and Class Sensor Parameters [TaskFlex]

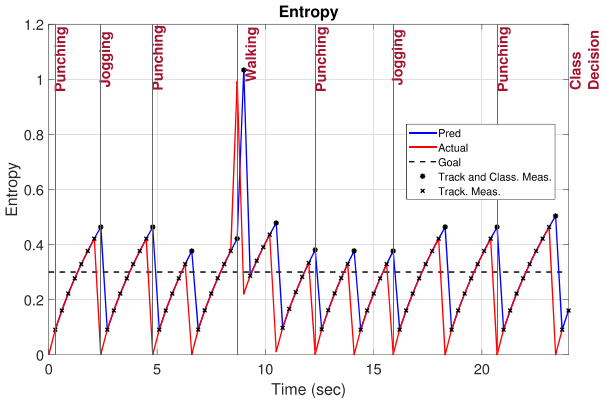
Approved for public release; distribution is unlimited.



(a) Tracks, Track Performance and Parameter vs Time



(b) Predicted and Actual Range and Velocity Standard Deviation (zoomed)



(c) Predicted and Actual Entropy

Figure 8.9: 4 Parameter Adaptation: Track and Class Sensor Parameters [TaskFlex]

Approved for public release; distribution is unlimited.

the framework chooses to classify in order to reduce the entropy. In addition to this we see that we perform that the classifier performs well in looking at Figure 8.9 (c) with minimal confusion for our target class whereas we see that there is confusion for the fixed parameter case, particularly around $\sim$ 10s when looking at Figure 8.8 (c).

From this analysis, we see that when the task is to select between tracking or classification parameters based on when better performance is needed for one or the other, an adaptive parameter case outperforms the fixed parameter case in velocity and range standard deviation while still performing adequately in classification.

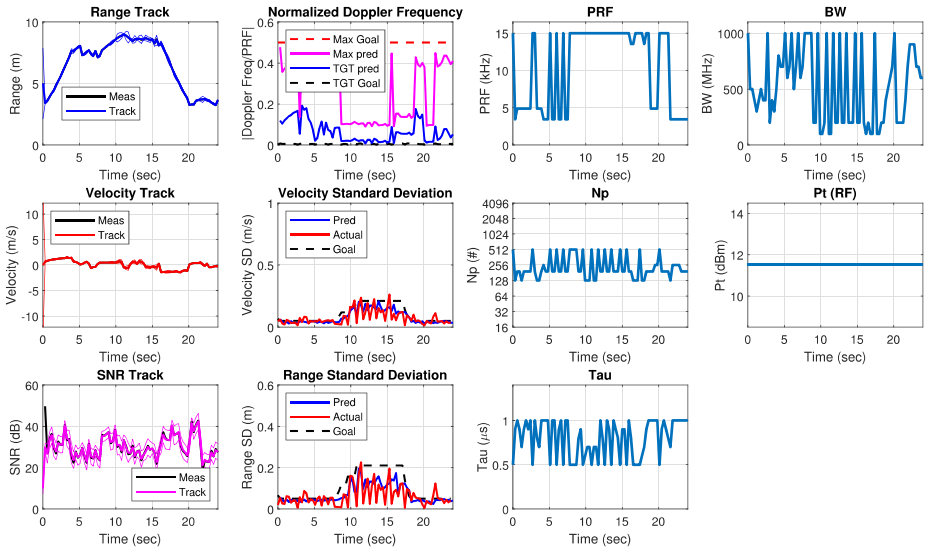
8.3.2 HFAR-ADM Results

For HFAR-ADM, we compare the performance between the fixed parameter set performing tracking and classification against an adaptive parameter set. Comparing the performance between the two, we see that better performance is again obtained through the adaptive framework. For ADM comparison, we use Figure 8.8. While the goals are adaptive for the ADM case and not shown in Figure 8.8, we can still use this as comparison due to the fact that the performance in range & velocity standard deviation as well as entropy will remain the same due to the fixed parameter selection.

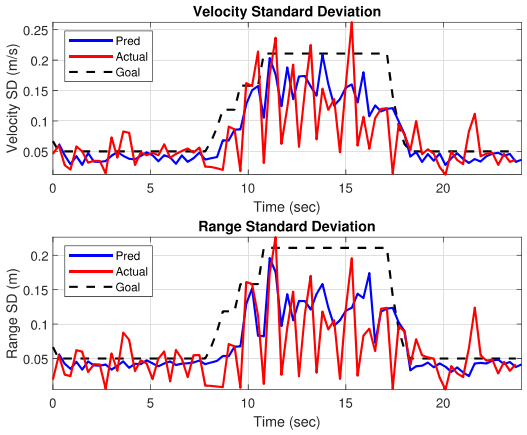
In Figure 8.10 (a) and (b), we that the velocity and range standard deviation performance remains below or at the goal line (even when adaptive). In addition to this, in Figure 8.10 (c), we see that classification performance also remains closer to the goal line and when deemed necessary we perform classification.

In comparing the fixed parameter case against the adaptive parameter case, we begin to see the framework decide target intent and set goals for classification and tracking performance in order to obtain a better perception of the environment. In

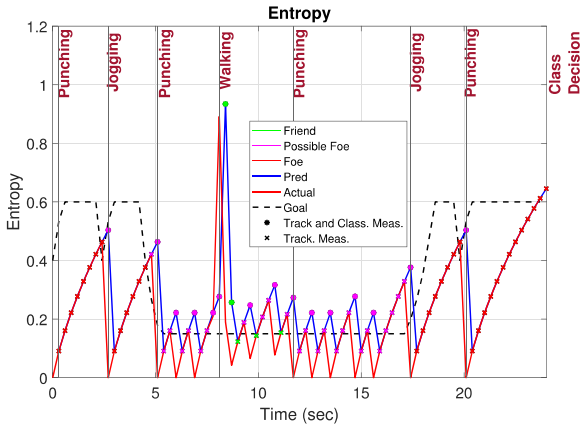
Approved for public release; distribution is unlimited.



(a) Tracks, Track Performance and Parameter vs Time



(b) Predicted and Actual Range and Velocity Standard Deviation (zoomed)



(c) Predicted and Actual Entropy

Figure 8.10: 4 Parameter Adaptation [ADM]

Approved for public release; distribution is unlimited.

Figure 8.10 (c), we see that the 4 parameter adaptation case seems to set the target priority to foe and possible foe more than friend, as expected. Due to the task priority being limited to foe and possible foe, we tend to want to classify more when we have a possible foe and obtain better tracking performance when a foe is present.

In Figure 8.10(b), we see that we are able to meet our tracking performance goals as well as maintaining our classification performance by attempting to remain below our adaptive entropy threshold value as shown in Figure 8.10 (c). In Figure 8.10 (c), we see that whenever we predict that our entropy will surpass the goal line significantly, we choose to classify resulting in a low posterior/actual entropy value. When choosing to track instead of classify, we begin to see a benefit in measurement resource consumption as the classification parameters lead to high resource consumption due to the radar mission being to minimize the radar timeline and the classification parameters having a large N_{p}, B, and τ. This additional benefit of tracking without classification allows the framework to meet its track performance goals while using less resources. It should be noted that due to this high measurement cost a classification measurement is taken only when truly necessary (when the entropy is substantially greater than its goal value), especially when the target is deemed a foe (indicated by the red markings) as this results in favoring a tracking measurement; however when the target is deemed a potential foe (indicated by the magenta markings), there becomes an increasing need for more information about the target which results in more classifications.

Approved for public release; distribution is unlimited.

8.4 Conclusion

The 4 parameter adaptation examples for both HFAR-TF and HFAR-ADM demonstrate task assignment and decision making through choosing to assign task/target priorities based on extracted information from the tracker and classifier. The frameworks developed for this work are inspired by the idea of maximizing information in order to improve environment perception. The amount of information represented through the tracker can be understood as the expected Fisher information matrix, the expected Cramer Rao lower bound or the expected range and velocity track error. Correspondingly, the amount of uncertainty present through the classifier can be understood using the predicted entropy. In order to improve environment perception, we wish to maximize our information through minimizing our track error and minimizing our entropy. The minimization of the track error is undertaken through the tracker and its sensor parameters whereas, the minimization of the entropy is undertaken by the classification performed using the classifier sensor parameters. In order to maximize information, we want to balance the cost of decreasing both these information metrics, while also using the minimum amount of system resources. For HFAR-ADM, the target priorities of foe and friend are understood to demand a level of certainty about the target intent, thus when the target is a potential foe more information is required leading to favoring more frequent classifications. However, both frameworks display and show the benefit of an adaptive radar system engaged in multiple radar tasks.

The adaptive radar frameworks of both HFAR-TF and HFAR-ADM displayed that adaptability in radar systems performing multiple tasks allows for better performance in each task when needed. These frameworks were also able to demonstrate the

Approved for public release; distribution is unlimited.

possibility of using tracking and classification tasks to determine target intent and that adaptability in radar systems can allow better performance through influence of extracted target intent. In addition to this, it was demonstrated that a simple decision making scheme can aid in developing an autonomous decision making system as well as guide performance of a system using a semblance of task prioritization.

Approved for public release; distribution is unlimited.

Chapter 9: Conclusions and Future Work

In summary, this dissertation has discussed developments and advances made to the existing FAR and HFAR frameworks through: the inclusion of neural network methodologies in order to reduce computational complexity in a cognitive radar architecture and introduce a stronger memory component than the existing Markov model based short-term memory from the prior, in order to allow for development of a *"truly"* cognitive system which learns from previously perceived or familiar environments; demonstrating and identifying different areas of benefit for neural networks in cognitive/adaptive processes; and expanding the applications of a cognitive system to different radar tasks such as multiple target tracking and resource management for a multi-function radar.

Chapter 2 provided background information on radar basics, state estimation, the MAP-PF algorithm used within the FAR framework, neural networks (static and dynamic), the FAR & HFAR frameworks, optimization and statistical theory that was used throughout the research presented in this dissertation. Chapter 3 provides a survey of cognitive science, cognitive/FAR, neural networks for radar applications and recent literature on the combination of neural networks and cognitive radar as related to the work in this dissertation.

Approved for public release; distribution is unlimited.

Chapter 4 describes the existing FARMS architecture, the CREW test-bed used for laboratory experiments and the flow of data within the framework in order to build an understanding of the cognitive framework inspired by cognitive structures proposed by Fuster and Haykin and discussed in Chapter 3.

Chapter 5 presents and discusses the benefits of neural network-based control of an adaptive framework. Due to the inherent recognition ability of neural networks, their ability to provide lower impact to system processing through faster time to solution convergence, and their ability to learn *"on-the-fly"*, a set of neural networks were implemented in the FAR and HFAR frameworks. Comparisons between the network-based control and the solver-based control displayed that neural networks were not only capable of performing parameter selection for radar systems but could also do so at faster speeds through use of a static network and obtain similar/better performance than the solver in both simulation and lab experiments. Results presented in a single target tracking example displayed the benefit of a static neural network in both the FAR and HFAR frameworks through their lower impact on system processing and ability to obtain similar performance to the solver-based approach; as well as the benefits of a dynamic network over a solver and a static network through its ability perform *"on-the-fly"* learning by using relevant memory of its inputs and outputs to adapt its internal states for better performance over time.

Chapter 6 reviews a neural network approach to rapid waveform generation and relates it to the problem of spectral congestion through interference avoidance. A feasibility verification was performed by evaluating two tasks: auto-encoding and nothched waveform generation using a LSTM-RNN. The LSTM-RNN was able to

Approved for public release; distribution is unlimited.

reproduce the given waveform in the auto-encoding task and design a notched waveform given an interference perturbed signal. By performing these tasks and obtaining acceptable results, the LSTM-RNN proved to be feasible for waveform generation and was further compared against an existing notched waveform design algorithm known as the ERA in which the neural network was able to obtain visibly comparable results to the algorithm with a faster timing in producing the solution. This provided further promotion of the benefit in neural network-based approaches on adaptive radar tasks.

In Chapter 7, the FAR framework was extended from a single target tracking example to a multiple target tracking environment (FAR-MTT) which lead to the development of new cost functions and the introduction of the penalty function method to allow for softer constraints and a more feasible solution space. In addition to this, a revised formulation of the FIM was developed in order integrate the effects of multiple targets in the track error calculation. Results obtained for the FAR-MTT framework displayed the benefit of an adaptive radar framework for tracking multiple targets and captured elements of resource allocation due to the needs of the targets.

Lastly, an extension of the HFAR framework to a radar needing to perform multiple tasks was presented through which two different frameworks based on the HFAR framework were developed : HFAR-TF and HFAR-ADM. Through developments of a decision making framework based on heuristics and cost evaluation, an adaptive radar framework for performing both classification and tracking was developed. The HFAR-TF framework demonstrated the benefit of adaptation in the tracking parameters of a radar system when needing to perform multiple tasks through selecting between tracking a target through adapted waveform parameters or classifying a target with fixed parameters when either performance needed to be improved when

Approved for public release; distribution is unlimited.

evaluating the expected entropy and track errors. Similarly, the HFAR-ADM framework demonstrated the benefit of adaptation in the tracking parameters as well as the tracking and classification goals when engaged with a radar system needing to perform tracking and classification while also discerning the intent of a target between *"foe", "possible foe", or "friend"*. When compared against a radar system with fixed set of tracking and classification parameters, both frameworks outperformed the fixed set example while conserving measurement resources when necessary.

Overall, the work presented in this dissertation provides evidence for the benefit of using neural networks within adaptive systems to either achieve better adaptive/cognitive performance (*"truly cognitive systems"*), lower impact to system processing in comparison to solver/algorithmic-based methods, allow real-time performance to be feasible and provide more autonomy in radar tasks, placing the radar operator in a more advisory role. In addition to this, advancements in the area of cognitive/FAR were made through extending the existing framework to a multiple target environment and development of an adaptive radar frameworks (HFAR-TF & HFAR-ADM) for a radar system engaged in multiple tasks as well as notions of resource management according to target needs represented in FAR-MTT and task needs or target priority represented through HFAR-TF and HFAR-ADM, respectively. These advancements establish a base for the development of cognitive radar systems applied to more complex and heterogeneous tasks with the possibility to offset impact to real-time performance through the use of machine learning approaches such as neural networks as shown for the single target tracking adaptive radar application.

Approved for public release; distribution is unlimited.

9.0.1 Future Work

Given the developments made in the work presented in this dissertation, there are possible areas for further refinement and continuing research using the cognitive/FAR concepts shown in the FAR & HFAR frameworks:

- Given the benefit that neural networks can provide in sensor parameter selection, it should also be noted that there is research in the machine learning and control system community on using neural networks for state model approximation. Due to the function approximation nature of neural networks and their ability to learn time-dependent information, research in applying dynamic/recurrent neural networks to learning different state models of targets in order to obtain flexibility in prediction for a single target or multiple target's with varying types of motions. In addition to learning the state models, there is also validity in being able to predict and assess performance based on the current k index, to $k+n$ indices in the future and adapt the current action selection based on future performance (where n is user defined number of future steps)

- With the surge of research in neural networks and machine learning, there have been many concerns with data-sparse problems due to the data driven nature of deep learning, however investigations into unsupervised learning or reinforcement learning allow for one to shift from data-driven approaches and labeled data to learning driven by statistical correlations & similarity in data distribution (unsupervised learning) and reward functions (reinforcement learning). Application of a reinforcement learning approach to the FAR framework would

Approved for public release; distribution is unlimited.

be beneficial as the simulation component of the framework allows for ease of incorporation to reinforcement learning models, especially given the fixed action space of the problem.

- The FAR-MTT framework is currently limited to a single sensor so an obvious extension is to a multi-static hierarchical framework. With this, there are multiple areas of research:
  - Extending the FAR-MTT to a hierarchical multiple target tracking framework with multiple sensors to perform Cartesian tracking and adapting the measurement covariance goals for the HFAR-MTT framework to consider target prioritization
  - Combining the target intent concept discussed in Chapter 8 with the multiple target environment and performing radar resource assignment based on target of highest *"threat"*.
  - Using a centralized multi-static HFAR framework for task assignment given a multiple target environment, in which each sensor is responsible for performing a different task based on the needs of the radar system and the evaluated benefit the sensor provides conditioned on its proximity to targets and available resources.
- Implementation of a feed-forward neural network in the HFAR framework shows scope for extension. One such is training and implementing neural networks for HFAR applications to perform parameter selection at the lower tier level and decision making/high level action at the higher tiers (ex: setting goals for track error/classification error or upper bounds for measurement covariance).

Approved for public release; distribution is unlimited.

An extended example of this is the possible use of static neural networks for parameter selection given a fixed parameter space and a dynamic/adaptable neural network for the decision making to capture the notion of "learning" taking place at the higher tiers.

- The question of task-based versus information theoretic methods of optimization for radar adaptation is one of contention in the adaptive radar community, with some taking a QoS approach and others using information based methods. An examination of these two methods for radar resource management similar to that discussed by Kreucher in [125] for the multi-tier, multi-PAC HFAR framework would be of great benefit in developing a flexible cognitive framework capable of implementing different adaptive radar approaches.

Approved for public release; distribution is unlimited.

Appendix A: FAR NN Appendix

A.1 Optimization Reliability

In order to further guarantee the optimality of the local solver performance versus the global solution, a comparison between a global solver and the local solver was made. While the `fmincon()` toolbox was used to implement the sqp algorithm for real time and simulation functionality, the global search method implemented using MATLAB's `GlobalSearch()` function was used for evaluating the reliability of the local solutions produced by the `fmincon()` sqp implementation.

The global search method was implemented in order to find the global minima. The global search algorithm is a scatter search method of finding global minima by generating multiple trial/potential start points and performing a search of the solution space until the best points that provide the lowest objective function value is found that satisfies the given constraints. For more information on the global search method, the reader is referred to [129].

Comparisons were made between the the local solver (`fmincon()` sqp) and the global solver (`GlobalSearch()`) solutions for the four parameter adaptation simulation case using a 2000 trial Monte Carlo FARMS 20 second run, resulting in 80 indices per 20 second run (for a total of 160000 indices/samples). The performance of

Approved for public release; distribution is unlimited.

Table A.1: Summary of Monte Carlo Trials for Optimization Reliability

Metric	**Primary Solver: Local Solver**	**Primary Solver: Global Solver**
Average of Function Evaluation Difference	3.13×10^{-2}	4.99×10^{-2}
Standard Deviation of Function Evaluation Difference	3.78×10^{-1}	2.36×10^{-1}
l_1 Norm of Average Theta Selection Difference	$8.21 \times 10^{-}1$	$3.14 \times 10^{-}2$
l_1 Norm of Standard Deviation of Theta Selection Difference	1.65	1.86

the local solver is compared to the global solver in order to show that the local solver is capable of producing comparatively optimal selections to the sqp solver. Table A.1 provides a summary of the comparison between the local and global solver over 2000 Monte Carlo trials.

In Figures A.3 and A.4, there are two plots at the top and bottom of these figures, respectively. The figure at the top is the overlaid histograms of the function evaluation of the primary and background solver, whereas the figure to the bottom is the histogram of the difference in function evaluations between the two. The process is illustrated in Figures A.1 and A.2. The difference in function evaluations are calculated by first evaluating the objective function value given the solutions from each solver and then taking the difference between the two:

$$\text{Feval Diff} = C_{E_{solv1}}(\boldsymbol{\theta}^*_{primary}) - C_{E_{solv2}}(\boldsymbol{\theta}^*_{background}). \tag{A.1}$$

where $C_{E_{solv1}}(\boldsymbol{\theta}^*_{primary})$ is the function evaluation of the primary solver and $C_{E_{solv2}}(\boldsymbol{\theta}^*_{background})$ is the function evaluation of the background solver using the solutions obtained by minimizing (5.2).

In Figure A.1, the FAR algorithm was run in four parameter simulation mode using the solutions from the local solver as the primary solution with the global solver running in the background, producing solutions to the inputs that the local

Approved for public release; distribution is unlimited.

solver receives. In Figure A.2, a similar run was performed except with the global solver solutions used as the primary solution choice.

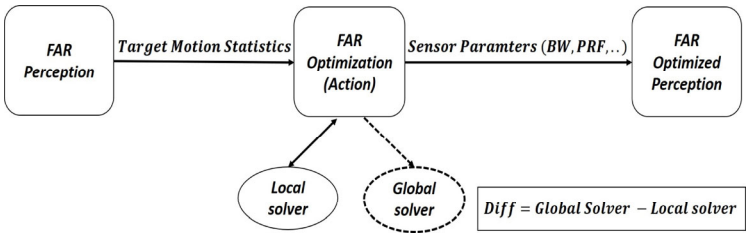


Figure A.1: Primary Solver: Local Solver

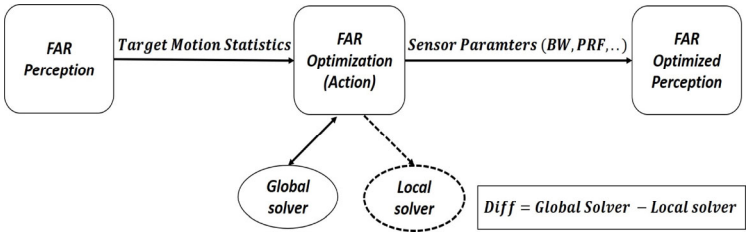


Figure A.2: Primary Solver:Global Solver

From Figures A.3 and A.4, we see that the local solver is able to obtain similar function evaluations as the global solver. A function evaluation value, $C_E(\boldsymbol{\theta}_k, \hat{\mathbf{x}}_k)$, represents the value of $C_M(\boldsymbol{\theta_k})$ with the optimized values of $\boldsymbol{\theta_k}$ given the constraints obtained from $C_P(\boldsymbol{\theta}_k, \hat{\mathbf{x}}_k)$ at the k^{th} index, obtained when performing (5.2). In Figure A.3, we see that when the parameter selections from the local solver are used in the framework, with the global solver running in the background (not used in the framework), the function evaluations between the two are pretty similar when viewing their overlaid histograms in addition to having very little difference in the function

Approved for public release; distribution is unlimited.

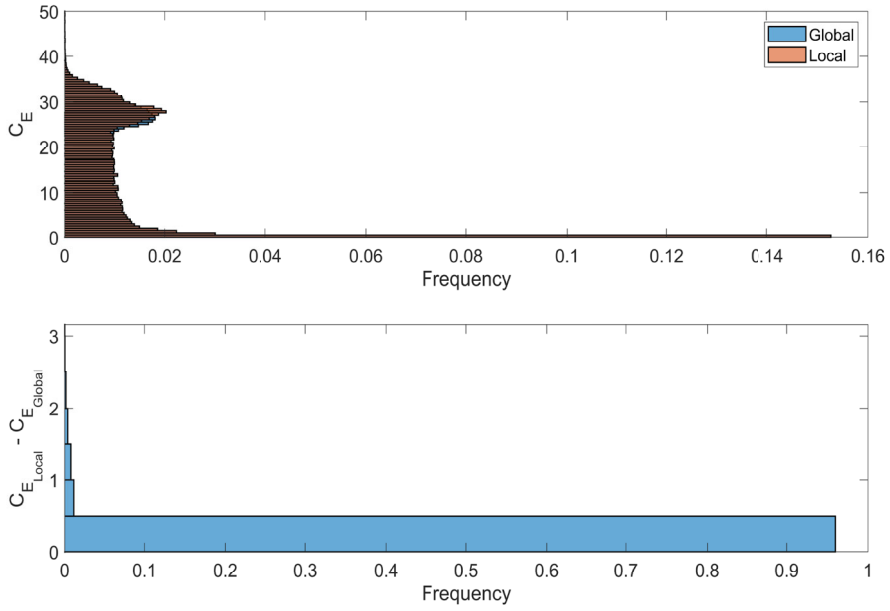


Figure A.3: Primary Solver: Local Solver [Overlaid histograms of Function Evaluation (top) , Function Difference between solvers (bottom)]

evaluations (seen in the plot to the bottom of Figure A.3). Additionally, it should be noted that the difference in function evaluation is very minute between the two solvers with the local solver obtaining similar function evaluations to the global solver throughout the trials. In looking at Figure A.4, we see that when the global solver solutions are used in the framework, with the local solver running in the background (not used in the framework), we have similar performance with very little difference in the function difference between the two solvers and similar frequency in function evaluation.

From these results, we can empirically see that when using the global solver parameter selections, we arrive at improved solutions more often, but the difference between

Approved for public release; distribution is unlimited.

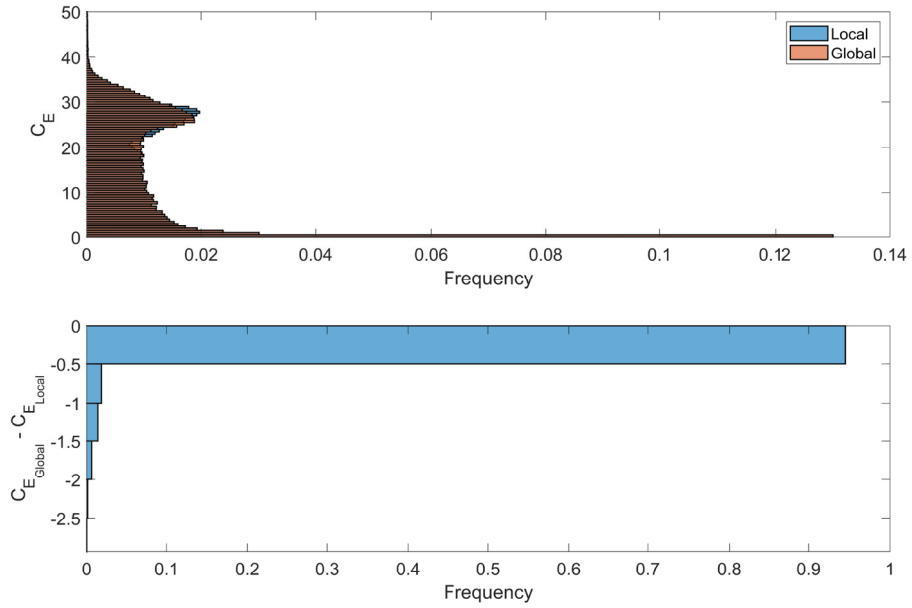


Figure A.4: Primary Solver:Global Solver [Overlaid histograms of Function Evaluation (top) , Function Difference between solvers (bottom)]

the function evaluations for the solvers are small and few in number. This quantitative analysis promotes the idea that the local solver solutions, while not capable of guaranteeing global minimums, are still acceptable to use for the FAR optimization.

Approved for public release; distribution is unlimited.

Appendix B: MTT Appendix

B.1 Fisher Information Matrix Derivation

Following formulations from [111], we set up the formulations of the Fisher Information Matrix. The FIM in [111] is developed for the parameter vectors $\boldsymbol{r}_k$, $\boldsymbol{f}_k$, and $\boldsymbol{\gamma}_k$, and has the form:

$$\boldsymbol{J}_{rf\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) = \begin{bmatrix} \boldsymbol{J}_{rr}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{rf}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{r\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \\ \boldsymbol{J}_{fr}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{ff}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{f\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \\ \boldsymbol{J}_{\gamma r}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{\gamma f}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) & \boldsymbol{J}_{\gamma\gamma}(\boldsymbol{y}_k;\boldsymbol{\theta}_k) \end{bmatrix} \tag{B.1}$$

where each block is an $M \times M$ matrix with elements ordered by target number for each parameter vector.

First, we define $\dot{\boldsymbol{w}}_r(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k)$ to be the $N_s \times 1$ vector which is the derivative of $\boldsymbol{w}(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k)$ with respect to $r_{m,k}$, $\dot{\boldsymbol{w}}_f(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k)$to be the $N_s \times 1$ vector which is the derivative of $\boldsymbol{w}(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k)$ with respect to $f_{m,k}$.

Next, we define $\boldsymbol{d}_r(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k)$ and $\boldsymbol{d}_f(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k)$ to be the $N_sN_p \times 1$ vectors which are the derivatives of $\boldsymbol{\nu}(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k)$ with respect to $r_{m,k}$ and $f_{m,k}$, respectively.

$$\begin{aligned} \boldsymbol{d}_r(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k) &= \frac{\partial}{\partial r_{m,k}}\boldsymbol{\nu}(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k) \\ &= \boldsymbol{u}(f_{m,k};\boldsymbol{\theta}_k) \otimes \dot{\boldsymbol{w}}_r(r_{m,k}, f_{m,k};\boldsymbol{\theta}_k) \end{aligned} \tag{B.2}$$

Approved for public release; distribution is unlimited.

$$\begin{aligned}\boldsymbol{d}_f(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k) &= \frac{\partial}{\partial f_{m,k}} \boldsymbol{\nu}(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k) \\ &= \dot{\boldsymbol{u}}_f(f_{m,k}; \boldsymbol{\theta}_k) \otimes \boldsymbol{w}(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k) + \\ &\quad \dot{\boldsymbol{u}}(f_{m,k}; \boldsymbol{\theta}_k) \otimes \dot{\boldsymbol{w}}_f(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k)\end{aligned} \tag{B.3}$$

where $\frac{\partial}{\partial(.)}$ is the partial derivative operator.

Finally, define $\boldsymbol{D}_r(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k)$ and $\boldsymbol{D}_f(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k)$ to be the $N_s N_p \times M$ matrices whose columns are the range and frequency derivative vectors for the M targets, respectively,

$$\boldsymbol{D}_r(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = [\boldsymbol{d}_r(r_{1,k}, f_{1,k}; \boldsymbol{\theta}_k) \cdots \boldsymbol{d}_r(r_{M,k}, f_{M,k}; \boldsymbol{\theta}_k)] \tag{B.4}$$

$$\boldsymbol{D}_f(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = [\boldsymbol{d}_f(r_{1,k}, f_{1,k}; \boldsymbol{\theta}_k) \cdots \boldsymbol{d}_f(r_{M,k}, f_{M,k}; \boldsymbol{\theta}_k)] \tag{B.5}$$

Rewriting the FIM expressions with the parameters from Section 7.2.1, we have the following the FIM expressions are:

$$\begin{aligned}\boldsymbol{J}_y^{rr} = 2Re\{&[\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_r] \odot [\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_r]^T + \\ &[\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{V}\boldsymbol{\Gamma}] \odot [\boldsymbol{D}_r^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_r]^T\}\end{aligned} \tag{B.6}$$

$$\begin{aligned}\boldsymbol{J}_y^{ff} = 2Re\{&[\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_f] \odot [\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_f]^T + \\ &[\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{V}\boldsymbol{\Gamma}] \odot [\boldsymbol{D}_f^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_f]^T\}\end{aligned} \tag{B.7}$$

$$\begin{aligned}\boldsymbol{J}_y^{rf} = 2Re\{&[\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_r] \odot [\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_f]^T + \\ &[\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{V}\boldsymbol{\Gamma}] \odot [\boldsymbol{D}_r^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_f]^T\}\end{aligned} \tag{B.8}$$

$$\boldsymbol{J}_y^{r\gamma} = 2Re[\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{V}] \odot [\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_r]^T \tag{B.9}$$

$$\boldsymbol{J}_y^{f\gamma} = 2Re[\boldsymbol{\Gamma}\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{V}] \odot [\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_f]^T \tag{B.10}$$

Approved for public release; distribution is unlimited.

$$\boldsymbol{J}_y^{\gamma\gamma} = [\boldsymbol{V}^H \boldsymbol{R}_z^{-1} \boldsymbol{V}] \odot [\boldsymbol{V}^H \boldsymbol{R}_z^{-1} \boldsymbol{V}]^T \tag{B.11}$$

$$\boldsymbol{J}_y^{fr} = [\boldsymbol{J}_y^{rf}]^{\boldsymbol{T}} \tag{B.12}$$

$$\boldsymbol{J}_y^{\gamma r} = [\boldsymbol{J}_y^{r\gamma}]^{\boldsymbol{T}} \tag{B.13}$$

$$\boldsymbol{J}_y^{\gamma f} = [\boldsymbol{J}_y^{f\gamma}]^{\boldsymbol{T}} \tag{B.14}$$

where $\odot$ is the Hadamard (element-wise) product.

Under the white noise assumption, we can simplify the expressions, giving us:

$$\boldsymbol{R}_{\mathrm{z}} = \boldsymbol{V}\boldsymbol{\Gamma}\boldsymbol{V}^H + \sigma_w^2 \boldsymbol{I} \tag{B.15}$$

$$\boldsymbol{R}_z^{-1} = \frac{1}{\sigma_w^2}[\boldsymbol{I} - \boldsymbol{V}[\boldsymbol{V}^{\boldsymbol{H}}\boldsymbol{V} + \sigma_w^2 \boldsymbol{\Gamma}^{-1}]^{-\mathbf{1}}\boldsymbol{V}^{\boldsymbol{H}}] \tag{B.16}$$

$$\begin{aligned} &\boldsymbol{V}^H \boldsymbol{R}_{\mathrm{z}}^{-1} \boldsymbol{V} = \\ &\boldsymbol{\Gamma}^{-1}[\boldsymbol{V}^{\boldsymbol{H}}\boldsymbol{V} + \sigma_w^2 \boldsymbol{\Gamma}^{-1}]^{-\mathbf{1}}\boldsymbol{V}^{\boldsymbol{H}}\boldsymbol{V} \end{aligned} \tag{B.17}$$

$$\begin{aligned} &\boldsymbol{V}^H \boldsymbol{R}_{\mathrm{z}}^{-1} \boldsymbol{D}_r = \\ &\boldsymbol{\Gamma}^{-1}[\boldsymbol{V}^{\boldsymbol{H}}\boldsymbol{V} + \sigma_w^2 \boldsymbol{\Gamma}^{-1}]^{-\mathbf{1}}\boldsymbol{V}^{\boldsymbol{H}}\boldsymbol{D}_r \end{aligned} \tag{B.18}$$

$$\begin{aligned} &\boldsymbol{D}_r^H \boldsymbol{R}_{\mathrm{z}}^{-1} \boldsymbol{D}_f = \frac{1}{\sigma_w^2}[\boldsymbol{D}_r^H \boldsymbol{D}_f - \\ &\boldsymbol{D}_r^H \boldsymbol{V}[\boldsymbol{V}^H \boldsymbol{V} + \sigma_w^2 \boldsymbol{\Gamma}^{-1}]^{-1} \boldsymbol{V}^H \boldsymbol{D}_f]. \end{aligned} \tag{B.19}$$

Approved for public release; distribution is unlimited.

Now, we define:

$$\boldsymbol{S} = \frac{1}{\sigma_w^2}\boldsymbol{\Gamma} \tag{B.20}$$

$$\boldsymbol{A} = [(\boldsymbol{V}^H\boldsymbol{V} + \boldsymbol{S}^{-1})]^{-1}. \tag{B.21}$$

Then,

$$\boldsymbol{V}^H\boldsymbol{R}_{\mathrm{z}}^{-1}\boldsymbol{V} = \boldsymbol{\Gamma}^{-1}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{V} \tag{B.22}$$

$$\boldsymbol{V}^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_r = \boldsymbol{\Gamma}^{-1}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_r \tag{B.23}$$

$$\boldsymbol{D}_r^H\boldsymbol{R}_z^{-1}\boldsymbol{D}_f = \frac{1}{\sigma_w^2}[bmD_r^H\boldsymbol{D}_f - \boldsymbol{D}_r^H\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_f]. \tag{B.24}$$

Then the FIM expressions become:

$$\begin{aligned}\boldsymbol{J}_y^{rr} = 2Re\{[\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_r] \odot [\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_r]^T + \\ [\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{V}\boldsymbol{S}] \odot [\boldsymbol{D}_r^H\boldsymbol{D}_r - \boldsymbol{D}_r^H\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_r]^T\}\end{aligned} \tag{B.25}$$

$$\begin{aligned}\boldsymbol{J}_y^{ff} = 2Re\{[\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_f] \odot [\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_f]^T + \\ [\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{V}\boldsymbol{S}] \odot [\boldsymbol{D}_f^H\boldsymbol{D}_f - \boldsymbol{D}_f^H\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_f]^T\}\end{aligned} \tag{B.26}$$

$$\begin{aligned}\boldsymbol{J}_y^{rf} = 2Re\{[\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_r] \odot [\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_f]^T + \\ [\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{V}\boldsymbol{S}] \odot [\boldsymbol{D}_r^H\boldsymbol{D}_f - \boldsymbol{D}_r^H\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_f]^T\}\end{aligned} \tag{B.27}$$

$$\boldsymbol{J}_y^{r\gamma} = \frac{2}{\sigma_w^2}Re[\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{V}] \odot [\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_r]^T \tag{B.28}$$

$$\boldsymbol{J}_y^{f\gamma} = \frac{2}{\sigma_w^2} Re[\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{V}] \odot [\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{D}_f]^T \tag{B.29}$$

$$\boldsymbol{J}_y^{\gamma\gamma} = \frac{1}{\sigma_w^4} Re[\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{V}] \odot [\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{V}^H\boldsymbol{V}]^T. \tag{B.30}$$

These expressions can be difficult to implement as $N_s N_p \times M$ matrices. They can be further simplified using properties of the Kronecker and Khatri-Rao products. From [15], we see the Khatri-Rao product $(*)$ of the matrices $\boldsymbol{A} = [\boldsymbol{\alpha}_1 \cdots \boldsymbol{\alpha}_M]$ and $\boldsymbol{B} = [\boldsymbol{b}_1 \cdots \boldsymbol{b}_M]$ defined as:

$$\boldsymbol{A} * \boldsymbol{B} = [\boldsymbol{\alpha}_1 \otimes \boldsymbol{b}_1 \cdots \boldsymbol{\alpha}_M \otimes \boldsymbol{b}_1]. \tag{B.31}$$

Using the following Khatri-Rao Product property:

$$(\boldsymbol{A} * \boldsymbol{B})^H (\boldsymbol{C} * \boldsymbol{D}) = (\boldsymbol{A}^H\boldsymbol{C}) \odot (\boldsymbol{B}^H\boldsymbol{D}). \tag{B.32}$$

If we define,

$$\boldsymbol{U}(\boldsymbol{f}_k; \boldsymbol{\theta}_k) = [\boldsymbol{u}(f_{1,k}) \cdots \boldsymbol{u}(f_{m,k}; \boldsymbol{\theta}_k)] \tag{B.33}$$

$$\dot{\boldsymbol{U}}_f(\boldsymbol{f}_k; \boldsymbol{\theta}_k) = [\dot{\boldsymbol{u}}(f_{1,k}) \cdots \dot{\boldsymbol{u}}(f_{m,k}; \boldsymbol{\theta}_k)] \tag{B.34}$$

$$\boldsymbol{W}(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = [\boldsymbol{w}(\boldsymbol{r}_{1,k}, f_{1,k}; \boldsymbol{\theta}_k) \cdots \boldsymbol{w}(\boldsymbol{r}_{M,k}, f_{M,k}; \boldsymbol{\theta}_k)] \tag{B.35}$$

$$\dot{\boldsymbol{W}}_r(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = [\dot{\boldsymbol{w}}_r(\boldsymbol{r}_{1,k}, f_{1,k}; \boldsymbol{\theta}_k) \cdots \dot{\boldsymbol{w}}_r(\boldsymbol{r}_{M,k}, f_{M,k}; \boldsymbol{\theta}_k)] \tag{B.36}$$

Approved for public release; distribution is unlimited.

$$\dot{\boldsymbol{W}}_f(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = [\dot{\boldsymbol{w}}_f(\boldsymbol{r}_{1,k}, f_{1,k}; \boldsymbol{\theta}_k) \cdots \dot{\boldsymbol{w}}_f(\boldsymbol{r}_{M,k}, f_{M,k}; \boldsymbol{\theta}_k)]. \tag{B.37}$$

Then,

$$\boldsymbol{V}(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = \boldsymbol{U}(\boldsymbol{f}_k; \boldsymbol{\theta}_k) * \boldsymbol{W}(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) \tag{B.38}$$

$$\boldsymbol{D}_r(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = \boldsymbol{U}(\boldsymbol{f}_k; \boldsymbol{\theta}_k) * \boldsymbol{W}_r(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) \tag{B.39}$$

$$\begin{aligned} \boldsymbol{D}_f(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) = \dot{\boldsymbol{U}}(\boldsymbol{f}_k; \boldsymbol{\theta}_k) * \boldsymbol{W}(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) + \\ \boldsymbol{U}(\boldsymbol{f}_k; \boldsymbol{\theta}_k) * \dot{\boldsymbol{W}}_f(\boldsymbol{r}_k, \boldsymbol{f}_k; \boldsymbol{\theta}_k) \end{aligned} \tag{B.40}$$

Dropping arguments, we then have the following equivalent expressions, which are easier to implement:

$$\boldsymbol{V}^H \boldsymbol{V} = \boldsymbol{U}^H \boldsymbol{U} \odot \boldsymbol{W}^H \boldsymbol{W} \tag{B.41}$$

$$\boldsymbol{V}^H \boldsymbol{D}_r = \boldsymbol{U}^H \boldsymbol{U} \odot \boldsymbol{W}^H \dot{\boldsymbol{W}}_r \tag{B.42}$$

$$\boldsymbol{V}^H \boldsymbol{D}_f = \boldsymbol{U}^H \dot{\boldsymbol{U}}_f \odot \boldsymbol{W}^H \boldsymbol{W} + \boldsymbol{U}^H \boldsymbol{U} \odot \boldsymbol{W}^H \dot{\boldsymbol{W}}_f \tag{B.43}$$

$$\boldsymbol{D}_r^H \boldsymbol{D}_r = \boldsymbol{U}^H \boldsymbol{U} \odot \dot{\boldsymbol{W}}^H \boldsymbol{W}_r \tag{B.44}$$

$$\begin{aligned} \boldsymbol{D}_f^H \boldsymbol{D}_f = \boldsymbol{U}_f^H \dot{\boldsymbol{U}}_f \odot \boldsymbol{W}^H \boldsymbol{W} + \boldsymbol{U}_f^H \boldsymbol{U} \odot \boldsymbol{W}^H \dot{\boldsymbol{W}}_f + \\ \boldsymbol{U}^H \dot{\boldsymbol{U}}_f \odot \dot{\boldsymbol{W}}_f^H \boldsymbol{W} + \boldsymbol{U}^H \boldsymbol{U} \odot \dot{\boldsymbol{W}}^H \dot{\boldsymbol{W}}_f. \end{aligned} \tag{B.45}$$

Approved for public release; distribution is unlimited.

$$\boldsymbol{D}_r^H \boldsymbol{D}_f = \boldsymbol{U}_f^H \dot{\boldsymbol{U}}_f \odot \boldsymbol{W}_r^H \boldsymbol{W} + \boldsymbol{U}^H \boldsymbol{U} \odot \boldsymbol{W}_r^H \dot{\boldsymbol{W}}_f. \tag{B.46}$$

The entries in $\boldsymbol{V}^H \boldsymbol{V}$ are the (scaled) ambiguity function of the waveform, and the entries in $\boldsymbol{V}^H \boldsymbol{D}_r$ and $\boldsymbol{V}^H \boldsymbol{D}_f$ are derivatives of the ambiguity function, which have closed form expressions in many cases.

Doppler Response Derivative Vector

To obtain the derivative of the Doppler response vector with respect to $f_{m,k}$, we need:

$$\frac{\partial}{\partial f_{m,k}} e^{j2\pi n T_p f_{m,k}} = (j2\pi n T_p) e^{j2\pi n T_p f_{m,k}} \tag{B.47}$$

Define,

$$\boldsymbol{a}_{N_p,T_p} = (j2\pi T_p)[0 \cdots N_p - 1]^T \tag{B.48}$$

Then,

$$\dot{\boldsymbol{u}}(f_{m,k}; \boldsymbol{\theta}_k) = \boldsymbol{a}_{N_p,T_p} \odot \boldsymbol{u}(f_{m,k}; \boldsymbol{\theta}_k) \tag{B.49}$$

Linear Frequency Modulation Derivative Vectors

For a LFM waveform up-chirp,

$$w(t) = e^{j\pi \frac{B}{\tau} t^2} \qquad 0 \leq t \leq \tau \tag{B.50}$$

$$w(t_n - T_{m,k}) e^{j2\pi f_{m,k} t_n} = e^{j\pi \frac{B}{\tau} (n t_s - \frac{2r_{m,k}}{c})^2} e^{j2\pi f_{m,k} t_n}$$

$$T_{m,k} \leq t_n \leq \tau T_{m,k} \tag{B.51}$$

Approved for public release; distribution is unlimited.

Using the expression for our time-shifted waveform, we then find our derivatives with respect to $r_{m,k}$ and $f_{m,k}$:

$$\frac{\partial}{\partial r_{m,k}} w(t_n - T_{m,k}) e^{j2\pi f_{m,k} t_n} = \\ (-j\frac{4\pi B}{\tau c}(nt_s - \frac{2r_{m,k}}{c}))e^{j\pi\frac{B}{\tau}(nt_s - \frac{2r_{m,k}}{c})^2} e^{j2\pi f_{m,k} nt_s} \tag{B.52}$$

$$\frac{\partial}{\partial f_{m,k}} w(t_n - T_{m,k}) e^{j2\pi f_{m,k} t_n} = \\ (j2\pi nt_s)e^{j\pi\frac{B}{\tau}(nt_s - \frac{2r_{m,k}}{c})^2} e^{j2\pi f_{m,k} nt_s} \tag{B.53}$$

$$\boldsymbol{b}_{N_s,t_s,r_{m,k}} = -j\frac{4\pi B t_s}{\tau c}([0 \cdots N_s - 1]^T - \frac{2r_{m,k}}{ct_s}) \tag{B.54}$$

$$\boldsymbol{a}_{N_s,t_s} = (j2\pi nt_s)[0 \cdots N_s - 1]^T \tag{B.55}$$

Then,

$$\dot{\boldsymbol{w}}_{\boldsymbol{r}}(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k) = \boldsymbol{b}_{N_s,t_s,r_{m,k}} \odot \boldsymbol{w}(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k) \tag{B.56}$$

$$\dot{\boldsymbol{w}}_{\boldsymbol{f}}(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k) = \boldsymbol{a}_{N_s,t_s} \odot \boldsymbol{w}(r_{m,k}, f_{m,k}; \boldsymbol{\theta}_k) \tag{B.57}$$

B.2 Pre-Whitening Derivation

The pre-whitening process is outlined here, where the signals corresponding to the targets we wish to suppress are viewed as interference:

First we define the $N_s N_p \times (M-1)$ matrix

Approved for public release; distribution is unlimited.

$$\tilde{\boldsymbol{V}} = [v_1 \cdots v_m]. \tag{B.58}$$

$$\tag{B.59}$$

Then we define the $(M-1) \times (M-1)$ matrix

$$\tilde{\boldsymbol{\Gamma}} = diag[\gamma_1 \cdots \gamma_M]. \tag{B.60}$$

$$\tag{B.61}$$

In order to give us the interference covariance matrix:

$$\tilde{\boldsymbol{R}}_z = \sum_{m' \neq m} \gamma_{m'} \boldsymbol{v_{m'}} \boldsymbol{v}_{\boldsymbol{m'}}^H + \sigma_W^2 \boldsymbol{I} \tag{B.62}$$

$$= \tilde{\boldsymbol{V}} \tilde{\boldsymbol{\Gamma}} \tilde{\boldsymbol{V}}^H + + \sigma_W^2 \boldsymbol{I} \tag{B.63}$$

$$\tag{B.64}$$

where $\boldsymbol{I}$ is the $N_s N_p \times N_s N_p$ identity matrix.

Now, let $\boldsymbol{E_i}$ with $i = 1 : M-1$ denote the eigenvectors of $\tilde{\boldsymbol{V}} \tilde{\boldsymbol{\Gamma}} \tilde{\boldsymbol{V}}^H$ which can be written as:

$$\tilde{\boldsymbol{V}} \tilde{\boldsymbol{\Gamma}} \tilde{\boldsymbol{V}}^H = \sum_{i=1}^{M-1} \lambda_i \boldsymbol{E_i} \boldsymbol{E_i}^H. \tag{B.65}$$

$$\tag{B.66}$$

We can then state that the eigenvectors of $\tilde{\boldsymbol{R}}_{\boldsymbol{z}}$ are $\boldsymbol{E_i}$ with $i = 1 : M-1$, the eigenvalues are $\lambda_i + \sigma_W^2$, and the rest of the eigenvectors are an orthonormal set that are orthogonal to $\boldsymbol{E_i}$.

279
Approved for public release; distribution is unlimited.

Following the theory developed in [111], we recognize that eigenvectors of $\tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H$ satisfy the following:

$$\lambda_i \boldsymbol{E_i} = \tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H \boldsymbol{E_i} \tag{B.67}$$

$$[\lambda_i I - \tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H]\boldsymbol{E_i} = 0. \tag{B.68}$$

Due to the fact that $\tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H$ is of rank $M-1$, there will be $M-1$ non-zero eigenvalues and corresponding eigenvectors. These eigenvectors span the same subspace as $\tilde{\boldsymbol{V}}$, known as the signal subspace. Therefore, they can be expressed as a linear combination of the columns of $\tilde{\boldsymbol{V}}$, i.e.

$$\boldsymbol{E_i} = \tilde{\boldsymbol{V}}\boldsymbol{c_i} = \sum_{j=1}^{M-1} \tilde{\boldsymbol{v}}_{\boldsymbol{j}} c_{ji} \tag{B.69}$$

where $\boldsymbol{c_i}$ is a $(M-1) \times 1$ vector.

We can then write:

$$\lambda_i \tilde{\boldsymbol{V}}\boldsymbol{c_i} = \tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H \boldsymbol{c_i} \tag{B.70}$$

$$\tilde{\boldsymbol{V}}[\lambda_i I - \tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H\tilde{\boldsymbol{V}}]\boldsymbol{c_i} = 0 \tag{B.71}$$

$$[\lambda_i I - \tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H\tilde{\boldsymbol{V}}]\boldsymbol{c_i} = 0 \tag{B.72}$$

where c_i is an eigenvector of $\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H\tilde{\boldsymbol{V}}$, which is an $(M-1) \times (M-1)$ matrix.

Once we find the $M-1$ eigenvectors of $\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H\tilde{\boldsymbol{V}}$, we can then write the eigenvectors of $\tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^H$ as $\boldsymbol{E_i} = \tilde{\boldsymbol{V}}\boldsymbol{c_i}$.

Using these formulations, we can write the interference covariance matrix, $\tilde{\boldsymbol{R}}_{\text{z}}$, as:

Approved for public release; distribution is unlimited.

$$\tilde{\boldsymbol{R}}_{\text{z}} = \tilde{\boldsymbol{V}}\tilde{\boldsymbol{\Gamma}}\tilde{\boldsymbol{V}}^{H} \tag{B.73}$$

$$= \sum_{i=1}^{M-1}(\lambda_i + \sigma_W^2)\boldsymbol{E_i}\boldsymbol{E_i}^H + \sum_{i=M}^{N_sN_p}(\sigma_W^2)\boldsymbol{E_i}\boldsymbol{E_i}^H \tag{B.74}$$

$$\tilde{\boldsymbol{R}}_{\text{z}}^{-1/2} = \sum_{i=1}^{M-1}(\lambda_i + \sigma_W^2)^{-1/2}\boldsymbol{E_i}\boldsymbol{E_i}^H + \tag{B.75}$$

$$\sum_{i=M}^{N_sN_p}(\sigma_W^2)^{-1/2}\boldsymbol{E_i}\boldsymbol{E_i}^H \tag{B.76}$$

$$= \sum_{i=1}^{M-1}(\lambda_i + \sigma_W^2)^{-1/2} - (\sigma_W^{-1/2})\boldsymbol{E_i}\boldsymbol{E_i}^H + \tag{B.77}$$

$$\sum_{i=1}^{N_sN_p}(\sigma_W^2)^{-1/2}\boldsymbol{E_i}\boldsymbol{E_i}^H \tag{B.78}$$

$$= \sum_{i=1}^{M-1}(\lambda_i + \sigma_W^2)^{-1/2} - (\sigma_W^{-1/2})\boldsymbol{E_i}\boldsymbol{E_i}^H + \frac{1}{\sqrt{\sigma_W^2}}\boldsymbol{I} \tag{B.79}$$

Bibliography

[1] A. E. Mitchell, G. E. Smith, K. L. Bell, and M. Rangaswamy, "Coordinate descent for cognitive radar adaptation," in *2016 CIE International Conference on Radar (RADAR)*, Oct 2016, pp. 1–5.

[2] K. L. Bell, C. J. Baker, G. E. Smith, J. T. Johnson, and M. Rangaswamy, "Fully adaptive radar for target tracking part i: Single target tracking," in *2014 IEEE Radar Conference*, May 2014, pp. 0303–0308.

[3] K. L. Bell, C. J. Baker, G. E. Smith, J. T. Johnson, and M. Rangaswamy, "Fully adaptive radar for target tracking part ii: Target detection and track initiation," in *2014 IEEE Radar Conference*, May 2014, pp. 0309–0314.

[4] K. L. Bell, C. J. Baker, G. E. Smith, J. T. Johnson, and M. Rangaswamy, "Cognitive radar framework for target detection and tracking," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 8, pp. 1427–1439, Dec 2015.

[5] A. F. Martone, "Cognitive radar demystified," *URSI Radio Science Bulletin*, vol. 2014, no. 350, pp. 10–22, 2014.

[6] S. Haykin, Y. Xue, and P. Setoodeh, "Cognitive radar: Step toward bridging the gap between neuroscience and engineering," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3102–3130, Nov 2012.

[7] A. E. Mitchell, G. E. Smith, K. L. Bell, A. J. Duly, and M. Rangaswamy, "Cost function design for the fully adaptive radar framework," *IET Radar, Sonar Navigation*, vol. 12, no. 12, pp. 1380–1389, 2018.

[8] R. Gross, *Psychology: The Science of Mind and Behaviour*, 6th ed. Hachette UK, 2012, includes bibliographical references (p. [824]-874) and index.

[9] R. S. Michalski, J. Carbonell, and T. M. Mitchell, *Machine learning. an Artificial Intelligence approach. Volume 2*, 01 1983, vol. Vol. II.

[10] G. E. Hinton, "How neural networks learn from experience," *Scientific American*, vol. 267, pp. 144–51, 10 1992.

Approved for public release; distribution is unlimited.

[11] P. John-Baptiste and G. E. Smith, "Utilizing neural networks for fully adaptive radar," in *2019 IEEE Radar Conference*, Apr 2019.

[12] "Radar - History of radar," Apr 2020, [Online; accessed 28. Apr. 2020]. [Online]. Available: https://www.britannica.com/technology/radar/History-of-radar

[13] M. Richards, *Principles of Modern Radar: Basic Principles, Volume 1*, 1st ed. SciTech Publishing, Inc., 2010, vol. 1. [Online]. Available: https://books.google.com/books?id=w-MPDQAAQBAJ&source=gbs_similarbooks

[14] L. Stone, "Likelihood ratio detection and tracking," in *2009 12th International Conference on Information Fusion*, 2009, pp. 10–10.

[15] L. Stone, R. Streit, T. Corwin, and K. Bell, *Bayesian Multiple Target Tracking, Second Edition*, ser. Radar/Remote Sensing. Artech House, 2013. [Online]. Available: https://books.google.com/books?id=fJhQAgAAQBAJ

[16] S. G. Nash, *Linear and nonlinear programming*. McGraw-Hill Science, Engineering & Mathematics, 1996.

[17] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, Mar. 1996.

[18] G. Zhang, B. E. Patuwo], and M. Y. Hu], "Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35 – 62, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207097000447

[19] A. Tikkala and M. Juhola, "A neural network simulation of aphasic naming errors: Network dynamics and control," *Neurocomputing*, vol. 13, no. 1, pp. 11 – 29, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0925231295000828

[20] R. Rojas, *The Backpropagation Algorithm*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 149–182. [Online]. Available: https://doi.org/10.1007/978-3-642-61068-4_7

[21] "Neuron3.png (PNG Image, 390 × 205 pixels)," Sep 2018, [Online; accessed 30. Jun. 2020]. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/4/44/Neuron3.png

[22] "fmech-05-00030-g002.jpg (JPEG Image, 765 × 339 pixels)," Jun 2019, [Online; accessed 30. Jun. 2020]. [Online]. Available: https://www.frontiersin.org/files/Articles/447117/fmech-05-00030-HTML/image_m/fmech-05-00030-g002.jpg

Approved for public release; distribution is unlimited.

[23] S. Lowel and W. Singer, "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity," *Science*, vol. 255, no. 5041, pp. 209–212, 1992. [Online]. Available: https://science.sciencemag.org/content/255/5041/209

[24] "Edge.org," Apr 2020, [Online; accessed 28. Apr. 2020]. [Online]. Available: https://www.edge.org/response-detail/10351

[25] R. D. Reed and R. J. Marks, *Neural Smithing : Supervised Learning in Feedforward Artificial Neural Networks.*, ser. Bradford Book. A Bradford Book, 1999. [Online]. Available: https://proxy.lib.ohio-state.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=9366&site=ehost-live

[26] M. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, p. 3, 2012.

[27] Y.-M. Chiang, L.-C. Chang, and F.-J. Chang, "Comparison of static-feedforward and dynamic-feedback neural networks for rainfall–runoff modeling," *Journal of Hydrology*, vol. 290, pp. 297–311, 05 2004.

[28] A. E. Mitchell, G. E. Smith, K. L. Bell, A. J. Duly, and M. Rangaswamy, "Hierarchical fully adaptive radar," *IET Radar, Sonar Navigation*, vol. 12, no. 12, pp. 1371–1379, 2018.

[29] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[30] K. G. Murty, *Linear Programming*. Wiley, Oct 1983. [Online]. Available: https://www.wiley.com/en-us/Linear+Programming-p-9780471097259

[31] R. Byrd, M. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, Jan. 1999.

[32] D. P. Bertsekas, "Convex optimization algorithms," 2015.

[33] A. V. Fiacco and G. P. McCormick, *Nonlinear programming: sequential unconstrained minimization techniques*. Siam, 1990, vol. 4.

[34] S. Rocktäschel, *A Branch-and-Bound Algorithm for Multiobjective Mixed-integer Convex Optimization*. Springer Spektrum, Wiesbaden, 2020.

[35] J. Pearl, "Heuristics: Intelligent search strategies for computer problem solving," 1 1984.

Approved for public release; distribution is unlimited.

[36] X.-S. Yang, "Chapter 23 - optimization," in *Engineering Mathematics with Examples and Applications*, X.-S. Yang, Ed. Academic Press, 2017, pp. 267 – 283. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128097304000318

[37] S. P. Han, "A globally convergent method for nonlinear programming," *J. Optim. Theory Appl.*, vol. 22, no. 3, pp. 297–309, Jul 1977.

[38] A. Smith, D. Coit, T. Bäck, D. Fogel, and Z. Michalewicz, "Penalty functions," 07 1998.

[39] B. L. Welch, "The generalization ofstudent's' problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.

[40] J. Bermúdez, *Cognitive Science: An Introduction to the Science of the Mind*. Cambridge University Press, 2014. [Online]. Available: https://books.google.com/books?id=zB7HAgAAQBAJ

[41] P. H. Winston, *Artificial Intelligence (3rd Ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992.

[42] J. Fuster, *Cortex and Mind*. Oxford, U.K.: Oxford Univ. Press, 2003.

[43] C. Buckner and J. Garson, "Connectionism," in *The Stanford Encyclopedia of Philosophy*, fall 2019 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2019.

[44] J. L. Mcclelland and D. E. Rumelhart, "Distributed memory and the representation of general and specific information." *Journal of Experimental Psychology: General*, vol. 114, no. 2, p. 159–188, 1985.

[45] J. R. Guerci, "Cognitive radar: A knowledge-aided fully adaptive approach," in *2010 IEEE Radar Conference*, May 2010, pp. 1365–1370.

[46] S. Brüggenwirth, M. Warnke, C. Bräu, Simon, T. Müller, P. Marquardt, and F. Rial, "Sense Smart, Not Hard: A Layered Cognitive Radar Architecture," *IntechOpen*, May 2018.

[47] B. Ravenscroft, S. D. Blunt, C. Allen, A. Marione, and K. Sherbondy, "Analysis of spectral notching in FM noise radar using measured interference," in *International Conference on Radar Systems (Radar 2017)*, Oct. 2017.

[48] J. Jakabosky, S. D. Blunt, and B. Himed, "Spectral-shape optimized fm noise radar for pulse agility," *2016 IEEE Radar Conference (RadarConf)*, pp. 1–6, 2016.

Approved for public release; distribution is unlimited.

[49] L. S. Wang, J. P. Mcgeehan, C. Williams, and A. Doufexi, "Application of cooperative sensing in radar-communications coexistence," *IET Communications*, vol. 2, no. 6, pp. 856–868, July 2008.

[50] J. Jakabosky, B. Ravenscroft, S. D. Blunt, and A. Martone, "Gapped spectrum shaping for tandem-hopped radar/communications cognitive sensing," in *IEEE Radar Conf.*, May 2016.

[51] M. J. A. Rahman, M. Krunz, and R. Erwin, "Interference mitigation using spectrum sensing and dynamic frequency hopping," in *2012 IEEE International Conference on Communications (ICC)*, June 2012, pp. 4421–4425.

[52] M. Labib, V. Marojevic, A. F. Martone, J. H. Reed, and A. I. Zaghloui, "Coexistence between communications and radar systems: A survey," *URSI Radio Science Bulletin*, vol. 2017, no. 362, pp. 74–82, Sep. 2017.

[53] A. De Maio and A. Farina, "Transmit adaptivity for cognitive radar," 2017.

[54] J. W. Owen, B. Ravenscroft, B. H. Kirk, S. D. Blunt, C. T. Allen, A. F. Martone, K. D. Sherbondy, and R. M. Narayanan, "Experimental demonstration of cognitive spectrum sensing & notching for radar," in *2018 IEEE Radar Conference (RadarConf18)*. IEEE, 2018, pp. 0957–0962.

[55] C. P. Horne, A. M. Jones, G. E. Smith, and H. D. Griffiths, "Fast fully adaptive signalling for target matching," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 6, pp. 46–62, 2020.

[56] M. S. Greco, F. Gini, P. Stinco, and K. Bell, "Cognitive Radars: A Reality?" *arXiv e-prints*, p. arXiv:1803.01000, Feb 2018.

[57] A. E. Mitchell, J. L. Garry, A. J. Duly, G. E. Smith, K. L. Bell, and M. Rangaswamy, "Fully adaptive radar for variable resolution imaging," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 12, pp. 9810–9819, 2019.

[58] K. L. Bell, J. T. Johnson, G. E. Smith, C. J. Baker, and M. Rangaswamy, "Cognitive radar for target tracking using a software defined radar system," in *2015 IEEE Radar Conference (RadarCon)*, 2015, pp. 1394–1399.

[59] K. Bell, G. Smith, A. Mitchell, and M. Rangaswamy, "Multiple task hierarchical fully adaptive radar," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, Oct 2018, pp. 1344–1348.

[60] X. L. Travassos, S. L. Avila, and N. Ida, "Artificial neural networks and machine learning techniques applied to ground penetrating radar: A

Approved for public release; distribution is unlimited.

review," *Applied Computing and Informatics*, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210832718302266

[61] J. C. Briones, B. Flores, and R. Cruz-Cano, "Multi-mode radar target detection and recognition using neural networks," *International Journal of Advanced Robotic Systems*, vol. 9, no. 5, p. 177, 2012. [Online]. Available: https://doi.org/10.5772/52073

[62] F. Del Frate, A. Petrocchi, J. Lichtenegger, and G. Calabresi, "Neural networks for oil spill detection using ers-sar data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 5, pp. 2282–2287, Sep. 2000.

[63] S. Indolia, A. K. Goswami, S. Mishra, and P. Asopa, "Conceptual understanding of convolutional neural network- a deep learning approach," *Procedia Computer Science*, vol. 132, pp. 679 – 688, 2018, international Conference on Computational Intelligence and Data Science. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050918308019

[64] Y. Xie, W. Dai, Z. Hu, Y. Liu, C. Li, and X. Pu, "A novel convolutional neural network architecture for sar target recognition," *Journal of Sensors*, vol. 2019, 2019.

[65] Rongrui Xiao and V. Chandrasekar, "Development of a neural network based algorithm for rainfall estimation from radar observations," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 35, no. 1, pp. 160–171, Jan 1997.

[66] T. Shevgunov and E. Efimov, "Artificial neural networks implementing maximum likelihood estimator for passive radars," in *Artificial Intelligence and Algorithms in Intelligent Systems*, R. Silhavy, Ed. Cham: Springer International Publishing, 2019, pp. 144–153.

[67] G. Noone, "Radar pulse train parameter estimation and tracking using neural networks," in *Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, Nov 1995, pp. 95–98.

[68] J. Gao, B. Deng, Y. Qin, H. Wang, and X. Li, "Enhanced radar imaging using a complex-valued convolutional neural network," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 1, pp. 35–39, 2018.

[69] M. Shaghaghi and R. S. Adve, "Machine learning based cognitive radar resource management," 04 2018, pp. 1433–1438.

[70] A. M. Elbir, K. V. Mishra, and Y. C. Eldar, "Cognitive radar antenna selection via deep learning," 2018.

Approved for public release; distribution is unlimited.

[71] A. E. Mitchell, G. E. Smith, K. L. Bell, A. Duly, and M. Rangaswamy, "Fully adaptive radar cost function design," in *2018 IEEE Radar Conference (RadarConf18)*, 2018, pp. 1301–1306.

[72] K. L. Bell, G. E. Smith, A. E. Mitchell, and M. Rangaswamy, "Fully adaptive radar for target classification," in *2019 IEEE Radar Conference (RadarConf)*, 2019, pp. 1–6.

[73] F. P. Härter and H. F. de Campos Velho, "New approach to applying neural network in nonlinear dynamic model," *Applied Mathematical Modelling*, vol. 32, no. 12, pp. 2621 – 2633, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0307904X07002296

[74] K. Hornik, M. B. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[75] P. Dorfman and D. Henderson, *Data Management Solutions Using SAS Hash Table Operations.* SAS Institute, 2018.

[76] A. Kumar, *Data Structure for C Programming.* Laxmi Publications Pvt Limited, 2004.

[77] G. E. Smith, Z. Cammenga, A. Mitchell, K. L. Bell, M. Rangaswamy, J. T. Johnson, and C. J. Baker, "Experiments with cognitive radar," in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2015, pp. 293–296.

[78] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design.* PWS Publishing Co., 1996, pgs. 12-19.

[79] P. D. Wasserman, *Advanced Methods in Neural Computing.* Van Nostrand Reinhold, 1993.

[80] R. Kompella and R. Kompella, "Using lstms to forecast time-series," Jan 2018. [Online]. Available: https://towardsdatascience.com/using-lstms-to-forecast-time-series-4ab688386b1f

[81] M. Gonnella, "Bio-inspired rf steganography based joint radar/communication waveform design," Master's thesis, Wright State university, 2018.

[82] M. Bell, "Information theory and radar waveform design," *IEEE Transactions on Information Theory*, vol. 39, no. 5, pp. 1578–1597, September 1993.

[83] M. Wicks, "A brief history of waveform diversity," in *Radar Conference, 2009 IEEE.* IEEE, 2009, pp. 1–6.

Approved for public release; distribution is unlimited.

[84] P. Antonik, M. Wicks, H. Griffiths, and C. Baker, *Principles of Waveform Diversity and Design*, 1st ed. Raleigh, NC: SciTech Publishing, Inc., 2010.

[85] J. Arora, *Introduction to Optimum Design*, 3rd ed. Academic Press, 2012.

[86] A. Aubry, A. DeMaio, A. Farina, and M. Wicks, "Knowledge-aided (potentially cognitive) transmit signal and receive filter design in signal-dependent clutter," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 49, no. 1, pp. 93–117, January 2013.

[87] J. R. Guerci, "Cognitive radar: A knowledge-aided fully adaptive approach," in *Radar Conference*. Washington, D. C.: IEEE, May 2010, pp. 1365–1370.

[88] M. C. Wicks, "Spectrum crowding and cognitive radar," *2010 2nd International Workshop on Cognitive Information Processing*, pp. 452–457, 2010.

[89] B. Ravenscroft, J. W. Owen, J. Jakabosky, S. D. Blunt, A. F. Martone, and K. D. Sherbondy, "Experimental demonstration and analysis of cognitive spectrum sensing and notching for radar," *IET Radar, Sonar and Navigation*, vol. 12, pp. 1466–1475(9), December 2018. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-rsn.2018.5379

[90] L. Ma, A. Doufexi, and J. P. Coon, "Spectrum notching and interference avoidance for power line communications," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Sep. 2013, pp. 1451–1455.

[91] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley, 1973. [Online]. Available: https://books.google.com/books?id=POMGRAAACAAJ

[92] F. Chang, L. C. Chang, and H. H. H. Huang, "Real-time recurrent learning neural network for stream-flow forecasting," 2002.

[93] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," in *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)*, Dec 2014, pp. 1–6.

[94] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *CoRR*, vol. abs/1611.06455, 2016. [Online]. Available: http://arxiv.org/abs/1611.06455

[95] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

Approved for public release; distribution is unlimited.

[96] R. W. GERCHBERG, "A practical algorithm for the determination of phase from image and diffraction plane pictures," *Optik*, vol. 35, pp. 237–246, 1972. [Online]. Available: https://ci.nii.ac.jp/naid/10025518647/en/

[97] T. Higgins, T. Webster, and A. K. Shackelford, "Mitigating interference via spatial and spectral nulls," in *IET International Conference on Radar Systems (Radar 2012)*, 2012, pp. 1–6.

[98] A. M. Jones, "Frequency diverse array receiver architectures," 2011.

[99] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, p. 292, 2019.

[100] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[101] D. Hsu, "Multi-period Time Series Modeling with Sparsity via Bayesian Variational Inference," *arXiv e-prints*, p. arXiv:1707.00666, Jul 2017.

[102] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans, "Nonlinear Systems Identification Using Deep Dynamic Neural Networks," *arXiv e-prints*, p. arXiv:1610.01439, Oct 2016.

[103] T. Lin, T. Guo, and K. Aberer, "Hybrid neural networks for learning the trend in time series," pp. 2273–2279, 2017. [Online]. Available: http://infoscience.epfl.ch/record/262447

[104] J. R. Fienup, "Phase retrieval algorithms: a comparison," *Appl. Opt.*, vol. 21, no. 15, pp. 2758–2769, Aug 1982. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-21-15-2758

[105] C. P. Horne, A. M. Jones, G. E. Smith, and H. D. Griffiths, "Fast fully adaptive signalling for target matching," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 6, p. 46–62, 2020.

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

[107] L. Úbeda-Medina and J. Grajal, "Multiple target tracking in the fully adaptive radar framework," in *2016 IEEE Statistical Signal Processing Workshop (SSP)*, June 2016, pp. 1–5.

Approved for public release; distribution is unlimited.

[108] P. R. Barbosa, Y. Sarkale, E. K. Chong, Y. Li, S. Suvorova, and B. Moran, "Controlled tracking in urban terrain: Closing the loop," *Asian Journal of Control*, 2018.

[109] Y. Li, L. Krakow, E. Chong, and K. Groom, "Approximate stochastic dynamic programming for sensor scheduling to track multiple targets," *Digital Signal Processing*, vol. 19, no. 6, pp. 978 – 989, 2009, dASP'06 - Defense Applications of Signal Processing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1051200407000905

[110] P. Chavali and A. Nehorai, "Scheduling and power allocation in a cognitive radar network for multiple-target tracking," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 715–729, Feb 2012.

[111] H. L. Van Trees, *Optimum array processing: Part IV of detection, estimation, and modulation theory*. John Wiley & Sons, 2004.

[112] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective evolutionary algorithms and applications*. Springer Science & Business Media, 2006.

[113] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Struct. Multidiscip. Optim.*, vol. 41, no. 6, pp. 853–862, Jun 2010.

[114] J. Zgajnar and S. Kavcic, "Weighted Goal Programming and Penalty Functions: Whole-farm Planning Approach Under Risk," *AgEcon Search*, 2011.

[115] J. Andersson, "A survey of multiobjective optimization in engineering design," *Department of Mechanical Engineering, Linktjping University. Sweden*, 2000.

[116] A. Farina, A. D. Maio, and S. Haykin, Eds., *The Impact of Cognition on Radar Technology*, ser. Radar, Sonar and Navigation. Institution of Engineering and Technology, 2017. [Online]. Available: https://digital-library.theiet.org/content/books/ra/sbra520e

[117] R. Klemm, H. Griffiths, and W. Koch, *Novel radar techniques and applications*, ser. Electromagnetics and radar. Raleigh: Institution of Engineering and Technology, 2017. [Online]. Available: https://cds.cern.ch/record/2312970

[118] P. Stinco, M. Greco, F. Gini, and B. Himed, "Cognitive radars in spectrally dense environments," *IEEE Aerospace and Electronic Systems Magazine*, vol. 31, no. 10, pp. 20–27, 2016.

[119] L. Úbeda-Medina and J. Grajal, "Implementation of the fully adaptive radar framework: Practical limitations," *2017 IEEE Radar Conference (RadarConf)*, pp. 0761–0766, 2017.

Approved for public release; distribution is unlimited.

[120] M. Shaghaghi and R. S. Adve, "Task selection and scheduling in multifunction multichannel radars," in *2017 IEEE Radar Conference (RadarConf)*, 2017, pp. 0969–0974.

[121] P. Moo and Z. Ding, *Adaptive radar resource management*. Academic Press, 2015.

[122] A. Charlish, K. Woodbridge, and H. Griffiths, "Phased array radar resource management using continuous double auction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 3, pp. 2212–2224, 2015.

[123] A. M. Ponsford, L. Sevgi, and H. C. Chan, "An integrated maritime surveillance system based on high-frequency surface-wave radars. 2. operational status and system performance," *IEEE Antennas and Propagation Magazine*, vol. 43, no. 5, pp. 52–63, 2001.

[124] C. M. Kreucher, K. D. Kastella, and A. O. Hero III, "Information-based sensor management for multitarget tracking," in *Signal and Data Processing of Small Targets 2003*, vol. 5204. International Society for Optics and Photonics, 2004, pp. 480–489.

[125] C. Kreucher, A. O. Hero, and K. Kastella, "A comparison of task driven and information driven sensor management for target tracking," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 4004–4009.

[126] S. Haykin, "Cognitive radar networks," in *Fourth IEEE Workshop on Sensor Array and Multichannel Processing, 2006.*, 2006, pp. 1–24.

[127] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification, 2nd Edition*. Wiley, Nov 2000. [Online]. Available: https://www.wiley.com/en-us/Pattern+Classification%2C+2nd+Edition-p-9780471056690

[128] G. E. Smith and C. J. Baker, "Multistatic micro-doppler signature processing," pp. 241–272, 2014. [Online]. Available: https://digital-library.theiet.org/content/books/10.1049/pbra034e_ch9

[129] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Martí, "Scatter search and local nlp solvers: A multistart framework for global optimization," *INFORMS Journal on Computing*, vol. 19, no. 3, pp. 328–340, 2007.

Approved for public release; distribution is unlimited.