DISSERTATION

PERFORMANCE-COMPUTATION TRADEOFFS IN DETECTION AND ESTIMATION

Submitted by

Pranav U. Damale

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2023

Doctoral Committee:

Advisor: Edwin K. P. Chong

Ali Pezeshki
Ronald B. Tjlakens
Renzo Cavalieri

ABSTRACT

PERFORMANCE-COMPUTATION TRADEOFFS IN DETECTION AND ESTIMATION

Detection and estimation problems involve challenging tasks that often demand real-time, accurate results. Algorithms able to produce highly accurate results are often computationally expensive or inefficient. Naturally, we need to tailor algorithms to the specific needs of problems to optimally trade off between computation and accuracy. To explore this ever-present tradeoff, this dissertation describes three distinct problems in detection and estimation and our contribution to the decision-making process for choosing the best algorithms for solving these problems.

First, we look at tradeoffs involved in designing a low-cost, camera-based autonomous gait acquisition and analysis system for inspecting gait impairments in mice. Specifically, we give a detailed description of our detection and classification algorithms for gait-event detection and gait-parameter extraction. Using the videos acquired in a live-animal study, we validate the performance of our system for assessing recovery in a mouse model of Parkinson's disease.

Next, we analyze the tradeoffs involved in designing a modified data association algorithm for tracking multiple objects using measurements of uncertain origins, such as radar detection with false alarms and missed detection. Specifically, we explore the performance of the distance-weighting probabilistic data association approach in conjunction with the loopy-sum product algorithm and, using simulation data, we analyze its performance in terms of tracking accuracy and computation against other state-of-the-art data association methods for tracking multiple targets in clutter.

Finally, to address the ill-conditioning of linear minimum mean square error estimation, we develop four approximate Wiener filter formulas that do not directly involve the inverse of the observation covariance matrix. Using real data, we evaluate the performance-complexity tradeoff for our approximated filters.

The common underlying theme that connects our solutions to these distinct problems is that our decisions for selecting various parameters in each solution are based on the performance-computation tradeoff. Throughout this dissertation, we employ various methods to handle this tradeoff, such as receiver operating characteristics analysis and line-search procedure. Our analysis is beneficial for choosing the best algorithm to optimally trade off between performance and computation.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my doctoral advisor Prof. Edwin K. P. Chong. He took a huge risk in accepting me as his doctoral student. He gave me the freedom to explore my research interests. He showed enormous patience while I tried to grasp abstract mathematical concepts and struggled to write scientific papers in my broken English. He listened to my poorly thought-out ideas and kept me on track with his trademark question, "What did I mean exactly?". He provided me with a solid learning platform, and he was always there with me in my research journey. He helped me understand that no mistakes are too small and no tasks demand less than your very best. He pushed me when it was difficult to find the motivation to keep going and showed faith in me when the results were not there. His energy is contagious and work-ethic exemplary. I consider myself lucky to have ever had the chance to be trained by him, and I am forever indebted to him for all his guidance.

I would like to thank Prof. Ronald B. Tjalkens for giving me the opportunity to work on the gait analysis project. He was always willing to innovate and collaborate in his research work. He shared the same joy and pride as me to develop a fully-functioning system in-house. I will always appreciate the knowledge and experience I gained during this project.

I would like to thank my committee members Profs. Ali Pezeshki and Renzo Cavalieri. They were both instrumental in my education and progress throughout graduate school. Fundamental mathematical concepts in linear algebra and topology courses taught by Prof. Renzo Cavalieri helped me develop a solid foundation as I started on my research journey. Various signal processing topics, such as estimation and filter theory, compressed sensing, and inverse problems taught by Prof. Ali Pezeshki helped me expand my knowledge horizon and opened new paths for my research. I feel blessed to have learned under their expert guidance.

The research on developing an autonomous gait analysis system was a collaborative effort between the Department of Electrical and Computer Engineering and the Department of Environmental and Radiological Health Sciences at Colorado State University, Fort Collins, Colorado. I

would like to thank Drs. Sean L. Hammond and Katriana A. Popichak, who are my coauthors on several papers, for their valuable input as well as for conducting multiple experiments during the system's development. I would like to thank the National Institute of Health for funding this project, which was funded by grant R01ES021656 (RBT).

The research on developing a novel target tracking algorithm was a collaborative effort with Sandia National Laboratories, Albuquerque, New Mexico. I would like to thank Dr. Tian J. Ma for the insights he provided. I would like to thank Sandia National Laboratories for funding this project, which was partially supported under Purchase Order 1980525.

Special thanks goes to Prof. Louis L. Scharf. Not only his expertise and guidance were instrumental in our Wiener filter approximations project, but the whole research area of statistical signal processing exists today largely because of him. I am grateful to him for presenting our work at the 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

I would like to thank my colleagues Tushar Ganguli and Drs. Yugandhar Sarkale and Christopher P. Robbiano for creating a positive environment for discussing work and assisting with my research work throughout my tenure at Colorado State University.

Even though acknowledging everyone here is not feasible, I would like to thank my friends for their continued support in the pursuit of Ph.D. I will always be grateful to them for being by my side and never letting me feel alone on this journey.

Finally, I would like to thank my family for always encouraging me to follow my passions and for their unconditional love and unwavering belief in every endeavor that I have undertaken.

DEDICATION

I would like to dedicate this dissertation to my family, my friends, and my advisor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In this dissertation, we explore performance-computation tradeoffs faced in detection and estimation problems. Detection problems deal with discerning between information-bearing patterns and noise, while estimation problems deal with estimating the values of parameters based on measured or empirical data that has a random component. Together, these problems play a critical role in a variety of real-world applications, such as medical imaging, defense, economics, transportation, and life sciences. In practice, these problems demand real-time, accurate results. Algorithms that are able to produce highly accurate results are often computationally expensive or inefficient. Naturally, selecting an optimal algorithm entails a tradeoff between computation and accuracy. This tradeoff is governed by different parameters that are specific to the solutions to individual problems. In this dissertation, we describe three distinct detection and estimation problems: i) design of a low-cost, autonomous gait detection and estimation system for analyzing gait impairments in mice, ii) performance study of a distance-weighting approach with loopy-sum product algorithm for multi-object tracking, and iii) Wiener filter approximations without covariance matrix inversion. We present our solutions to these problems and evaluate the performance-computation tradeoffs encountered in the process of selecting the optimal solutions. Our performance-computation tradeoff evaluation for each of these solutions is what ties together these seemingly distinct detection and estimation problems.

In Chapter 2, we describe the design of our low-cost, camera-based autonomous gait acquisition and analysis system for inspecting gait impairments in mice. Gait impairments are one of the most common traits of many neurodegenerative diseases, such as Parkinson's disease and Huntington's disease, and analysis of gait impairments plays a vital role in the diagnosis of these diseases. However, commercial systems available for performing motion analysis in mice are expensive and implement patented or proprietary algorithms that are unpublished. To address the need for an economic gait assessment system and the accessibility of the motion analysis algo-

rithms, we give a step-by-step process for designing a low-cost, camera-based gait analysis system and give a detailed description of our object-detection algorithms for gait-event detection and gait-parameter extraction in mice. Using the videos acquired in a live-animal study, we present results for validating the performance of our system with receiver operating characteristics (ROC) and Hit:Miss:False (H:M:F) detection analyses.

In Chapter 3, we explore the performance of the distance-weighting probabilistic data association (DWPDA) approach in conjunction with the loopy-sum product algorithm (LSPA) for tracking multiple objects in clutter. Here, we discuss the problem of data association (DA), which is to infer the correspondence between targets and measurements. DA plays an important role when tracking multiple targets using measurements of uncertain origins, such as radar detection with false alarms and missed detection. We describe four data association methods for tracking multiple targets in cluttered environments: probabilistic data association (PDA), joint PDA (JPDA), LSPA, and a DA method that is a modification of PDA by incorporating a weighting scheme based on distances between position estimates and measurements. This distance-weighting approach, when combined with PDA, has been shown to enhance the tracking accuracy of PDA without significant change in the computation burden. Since PDA constitutes a crucial building block of LSPA, which is a state-of-the-art DA algorithm, we present a new integrated approach between DWPDA and LSPA. Using simulation results, we analyze the tradeoff between tracking accuracy and computation time for these DA methods.

In Chapter 4, we address the problem of ill-conditioning of the Wiener filter, the optimal linear minimum mean square error estimator. Computing the Wiener filter involves the inverse of the observation covariance matrix. In practice, the observation covariance matrix may have a large condition number, resulting in unreliable computation of the Wiener filter. To address this issue, we develop four approximate Wiener filter formulas using a truncation technique based on the principal components of a composite covariance matrix. Compared to the Wiener filter, our formulas do not directly involve inverting the observation covariance matrix. An important consequence is that our approximate filters do not suffer from the ill-conditioning of the covariance matrix and are

numerically reliable to compute. In addition, we prove that the approximate formulas converge to the Wiener filter as certain approximating terms vanish. Using empirical data, we evaluate the performance-computation tradeoff for our approximate filters.

We have written the rest of the chapters in this dissertation in a way that they are self-contained. All the remaining chapters are based on our published work, the details of which can be found at the beginning of each chapter. Our contributions to the design of an autonomous gait detection and analysis system problem are summarized as follows:

- We provide a step-by-step process for building a gait acquisition system.
- We describe the necessary image-processing algorithms for autonomous gait-event detection and gait-parameter extraction.
- We also provide the ROC and H:M:F analyses results to show that our described system produces accurate results.

Our contributions to the design of a modified DA algorithm problem are summarized as follows:

- We describe four DA methods for tracking multiple targets in clutter: PDA, JPDA, LSPA, and DWPDA.
- We apply these DA methods for tracking multiple crossing targets in cluttered environments.
- We explore the idea of modifying one of the building blocks for LSPA with the distance-weighting scheme of DWPDA and compare the performance of the modified algorithm to that of the original algorithm.

Our contributions to the ill-conditioning of the Wiener filter problem are summarized as follows:

- We present four approximate formulas for the Wiener filter that do not directly involve inverting the observation covariance matrix.
- We analyze the asymptotic performance of our proposed filters and show that our filters converge to the Wiener filter as certain approximating terms vanish.

- We describe two methods for determining the optimal number of principal components used in the approximate formulas.
- We evaluate the performance of our approximate Wiener filters using real data.

Chapter 2

A Low-Cost, Autonomous Gait Detection and Estimation System for Analyzing Gait Impairments in Mice [1]

2.1 Introduction

Gait impairments are one the most common traits of many neurodegenerative diseases, such as Parkinson's disease (PD), multiple sclerosis, amyotrophic lateral sclerosis, Huntington's disease, and spinal cord injury. Many human neurological diseases are studied using their mouse models [2]. Since preclinical mouse models of these diseases replicate their respective gait abnormalities [3], gait analysis in mice has always been a topic of interest for gaining insight into neurodegenerative diseases and identifying potential treatments. While validated commercial systems for gait analysis in mice exist, they remain expensive and use patented or proprietary implementation of unpublished work. To address the need for an economic gait assessment system, in this chapter we describe a design for a low-cost monitoring and assessment system for mice gait that can provide the necessary data comparable with some of the commercial systems already available in the market.

Traditionally, gait analysis in mice has been performed through visual inspection [4] and open field activity monitoring [5]. With advancements in imaging technology, many commercial systems have been developed for monitoring mice gaits, such as Animal Strideway Systems (Tekscan, Inc., South Boston, MA, USA), DigiGait Imaging Systems (Mouse Specifics, Inc., Framingham, MA, USA), and CatWalk XT (Noldus Information Technology, Wageningen, NL). There even have been individual attempts to develop an independent gait analysis system, such as one made

[1]The material in this Chapter was published in the Hindawi Journal of Healthcare Engineering special issue *Development in Optimization Algorithms for Smart Healthcare* [1].

by Casey Harr of the University of Kentucky [6]. The animal Strideway System uses thin-film force sensors for gait acquisition combined with digital imaging software for displaying the results. The DigiGait Imaging System implements a treadmill with a transparent treadmill belt and digital imaging hardware and software. The CatWalk XT uses a glass trackway equipped with image capturing hardware and software for gait assessment. All these systems provide possible options for gait acquisition in their own right, but they remain quite expensive and are protected with patents that apply closed source implementation.

In recent years, there has been a growing literature on examining the gait kinematics in rodents during preclinical trials. In [7], Maghsoudi et al. proposed a superpixel-based image segmentation method to examine the kinematics of running rodents to improve our understanding of motion control and to aid in treating motor impairments. The method utilizes spatial and color information for image segmentation to extract various features from the images. Wong and Shah [8] proposed a method to collect and analyze three-dimensional kinematics data of quadrupedal locomotion in rodents for preclinical trials. Their method utilizes a six-camera motion capture system to analyze the gait in rodents during treadmill locomotion and studies a variety of motion behaviors in healthy and neurotraumatic rats. In [9], Timotius et al. suggested a gait analysis technique that systematically scales individual gait parameters in mice and rats based on their video-derived silhouette length and area data along with body weight and age. The silhouette-scaled parameters are used for identifying the body length independent motor functional differences in transgenic Huntington's disease mice and PD rats. An open-source gait analysis suite called GAITOR [10] studies gait pattern changes in rodents during preclinical trials. The GAITOR suite is capable of detecting gait changes in rodent models of monoiodoacetate (MIA) injection of joint pain, sciatic nerve injury, elbow joint contracture, and spinal cord injury.

These efforts provide viable solutions for acquiring gait kinematics data in rodents. The gait kinematics data is useful for examining certain aspects of the gait, such as the position of joints and segments through each phase of the gait. However, these are not the only gait parameters indicative of neurodegenerative disease. Quantitative gait pattern analysis used during preclinical

trials calculates parameters such as cadence, stance, step and duty cycle, and inter-leg coordination. Available commercial systems for assessing these gait pattern parameters remain expensive.

To address the need for a cost-effective, robust system for comparing gait abnormalities in mouse models of neurological diseases to a baseline or control group, we have developed and built an inexpensive, camera-based system for gait acquisition and assessment using an LED-lit glass trackway. We have also independently developed algorithms for the extraction of various gait metrics and their analysis, which are implemented using MATLAB R2015b (MathWorks, Inc., Natick, MA, USA). Our system autonomously detects the gait and extracts a large set of kinematic parameters, such as positioning of footfalls, step patterns, contact and non-contact gait metrics, and inter-leg coordination. While we restrict our attention to the development and performance of these algorithms in this chapter, a detailed build of our system is given in [11].

We validate the accuracy of our gait detection algorithms with a receiver operating characteristic (ROC) analysis performed during a live animal study assessing recovery in a classical murine model of PD. We also calculate the Hit:Miss:False (H:M:F) ratio to verify the accuracy of our gait detection system during the same study. The study was conducted over a period of two weeks in three groups of C57BL/6 mice. The first group was administered with Saline. The second group was dosed with Methyl-4-phenyl-1,2,3,6-tetrahydropyridine (MPTP), which induces motor impairments identical to those observed in PD. The third group was treated with MPTP and 1,1-bis(3'indolyl)-1-(p-chlorophenyl)methane (C-DIM12), which has demonstrated efficacy in reducing MPTP-induced neuronal loss in mice [12–15]. The experimental setup and the training and habituation of mice to the trackway are explained in detail in [16]. Various gait metrics data acquired during the study showed general agreement with the known literature.

Apart from studying neurodegenerative diseases with the help of gait analysis, there is a wide variety of real-world problems that can be solved using accurate gait detection and estimation. Gait recognition at a distance is useful for video surveillance [17–20]. Characteristic gait parameters, such as varus instability in the knee or ankle at heel strike, are useful to perform comparisons between disguised perpetrators and suspects. Remote gait detection and analysis can be used for

healthcare monitoring [21–23]. Gait is a good indicator of our overall health status. Remote gait analysis can be effectively used to support the healthcare needs of the elderly in a non-invasive and reliable manner. Also, the unique gait features of a person can be used for biometric identification along with traditional methods such as fingerprint, face, or iris recognition [24–27]. Gait biometrics identifies features of an individual walking style, such as the shape and gesture, and is one of the recent biometrics systems. Clearly, systems analyzing gait patterns in daily life are in demand due to their non-invasive nature and ease of application.

Gait recognition is implemented as a combination of multiple techniques, such as object localization, motion recognition, and spatiotemporal event detection. Object localization refers to identifying the location of one or more objects in an image and drawing a bounding box around their extent. Motion recognition is the process of detecting a change in the position of an object relative to its surroundings or a change in the surroundings relative to an object. Events that can change in both time and space are called spatiotemporal events. Spatiotemporal event detection is used to model dynamic events such as animal movement and traffic control. The most popular approach for human gait recognition is the use of video recordings, where different methods, such as silhouette-based gait recognition and gait feature extraction, are implemented for identifying moving objects in the video [28–35]. Another approach is the use of wearable sensors, where gait data is collected by the sensors attached to the person's body or limb [36–43]. Yet another approach is the use of thermal sensors, where infrared thermal imaging is used for collecting human gait data and to remove noises from the complex background [44–46]. All these efforts highlight the growing interest gait recognition has gained from researchers and the need for an inexpensive, easy-to-implement system that can be reliably used for accurate gait detection and estimation.

The main contribution of this chapter is to provide a step-by-step process for building a gait acquisition system and describe the necessary image-processing algorithms for autonomous gait detection and parameter extraction. We also provide the gait analysis results using our described system from a live animal study that validates the efficacy of the system. A lack of open-source information is one of the major limiting factors for building a low-cost gait analysis system. Gait

analysis is one of the non-intrusive ways of diagnosis. The healthcare industry is experiencing a state of revolution with modern computers providing faster diagnosis and therapeutics. Despite modern electronic equipment and computers getting cheaper each year, healthcare remains expensive in all parts of the world. The detailed build guide provided in [11] and the image-processing algorithms provided in this chapter mitigate this limiting factor, enabling implementation of an inexpensive gait analysis system.

The remainder of this chapter is organized as follows. In Section 2.2, we give a detailed description of the gait acquisition and detection algorithms used in our system. In Section 2.3, we present detection performance, ROC analysis, and gait assessment results of the system performed during an in-lab experimental study. We discuss the overall performance and potential improvements of our system in Section 2.4 and provide concluding remarks in Section 2.5.

2.2 Method

Our gait acquisition system consists of a long trackway across which a single mouse can run and algorithms to detect the position and pressure of footfalls. The trackway is two meters long and made of glass, with a rectangular enclosure, a ventrally located camera to capture footfalls, top and side-mounted LEDs, and a computer with our data processing algorithms and a graphical user interface (GUI) developed in MATLAB. The rectangular enclosure is made of plastic and is placed on the top of the glass trackway to ensure that the mouse continues to run in a particular direction during an experiment. The goal of the system is to autonomously detect and analyze the gait characteristics of the mouse for the experiment. Figure 2.1 shows the proposed gait acquisition system, and Figure 2.2 shows an overview of the gait acquisition and assessment process. Each component is further discussed in detail in the following subsections.

2.2.1 Video Acquisition

LEDs are mounted on the top and side of the glass trackway to illuminate the footfalls of the mouse. Red LEDs are installed on the inside of the top side of the rectangular enclosure, and green

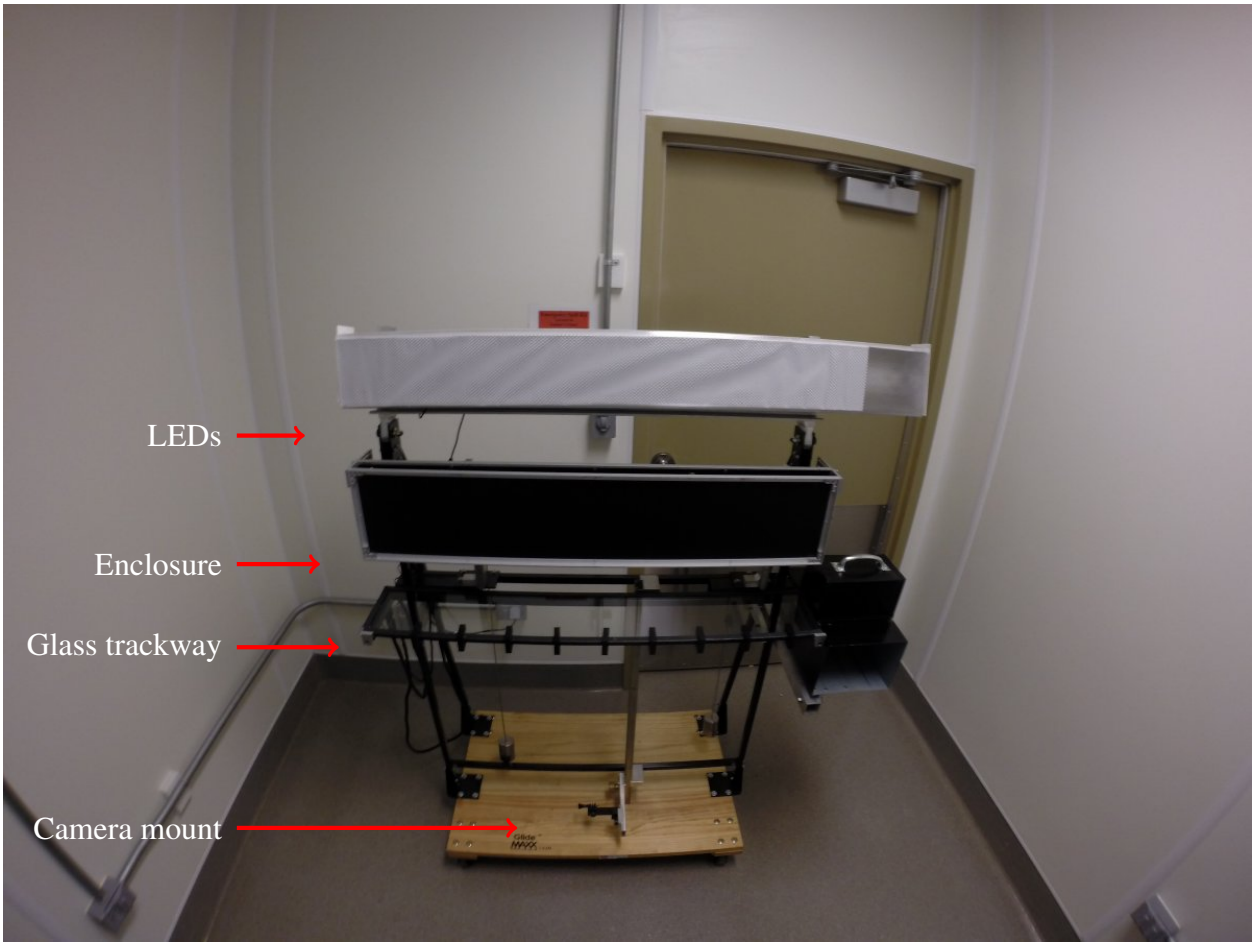

Figure 2.1: Gait acquisition system.

LEDs are installed along the sides of the glass trackway. The LEDs are equipped with adjustable intensity dimmers. The red LEDs illuminate the glass trackway in a red glow. The light from the green LEDs is reflected internally everywhere inside the glass trackway, except at those areas where the footfalls of the mouse make contact with the glass. The green light is refracted towards the camera at the footfalls creating a strong contrast between the footfalls and the background. This is illustrated in Figure 2.3. A high-end camera provides a high sampling rate and resolution, both of which are extremely important in collecting gait data. A higher resolution camera helps in capturing relatively fine details of the footfalls of a mouse, while a higher frame rate ensures that no footfalls are missed while recording a video. The camera of choice was GoPro HERO3+ (GoPro, Inc., San Mateo, CA, USA), which can be installed easily on the adjustable mount to determine the exact field of view of the lens. The videos are recorded at 60 frames per second (fps)

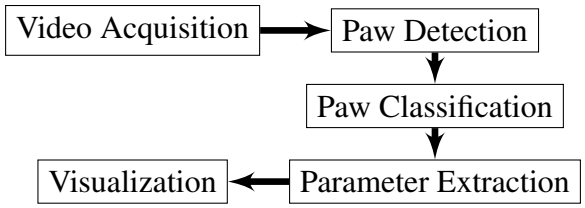

Figure 2.2: System overview.

and 1080×1920-pixel resolution with the help of the GoPro remote user interface. The captured videos are transferred to a computer and are processed using MATLAB with the help of the Image Processing and Computer Vision Toolbox.

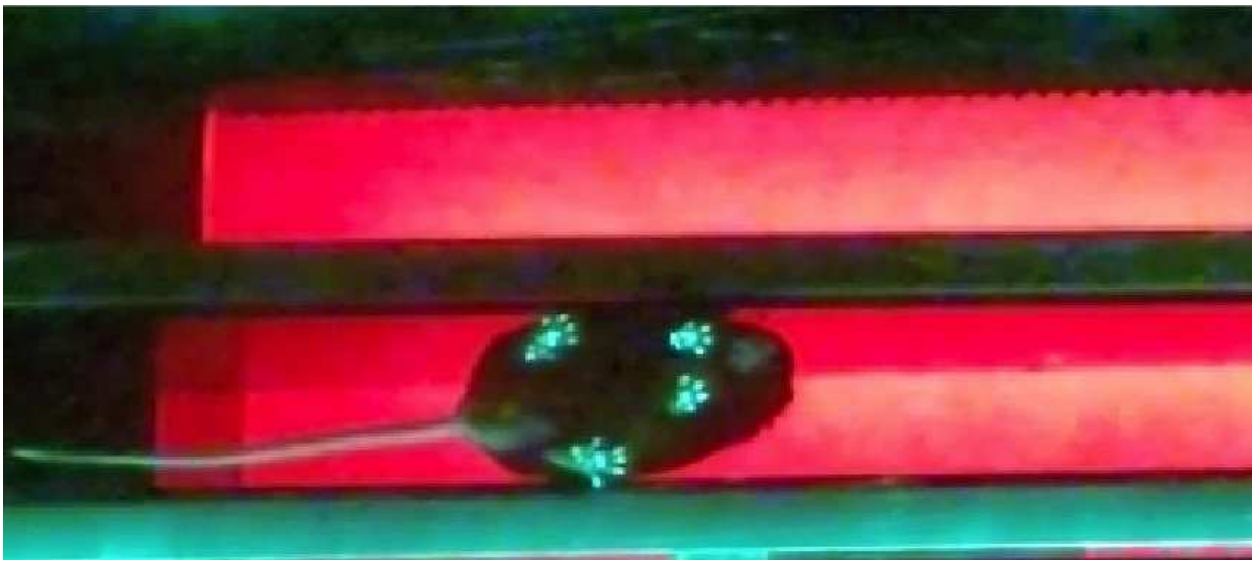

Figure 2.3: A sample image of the trackway, zoomed-in, to show the illuminated footfalls.

Once the videos are acquired, we need to perform some preprocessing on the image sequence before we acquire the gait data. The primary aim of data preprocessing is to extract only the necessary parts of an image and to avoid any extra processing for faster analysis. Also, we need to adjust for any possible distortion or rotation present in an image. For these purposes, we use the system GUI for manually setting the parameters for the preprocessing. First, we load an image from a video to be analyzed. Second, we use the system GUI to remove any barrel distortion in the image. Third, we rotate the image to make sure that the trackway is oriented horizontally in

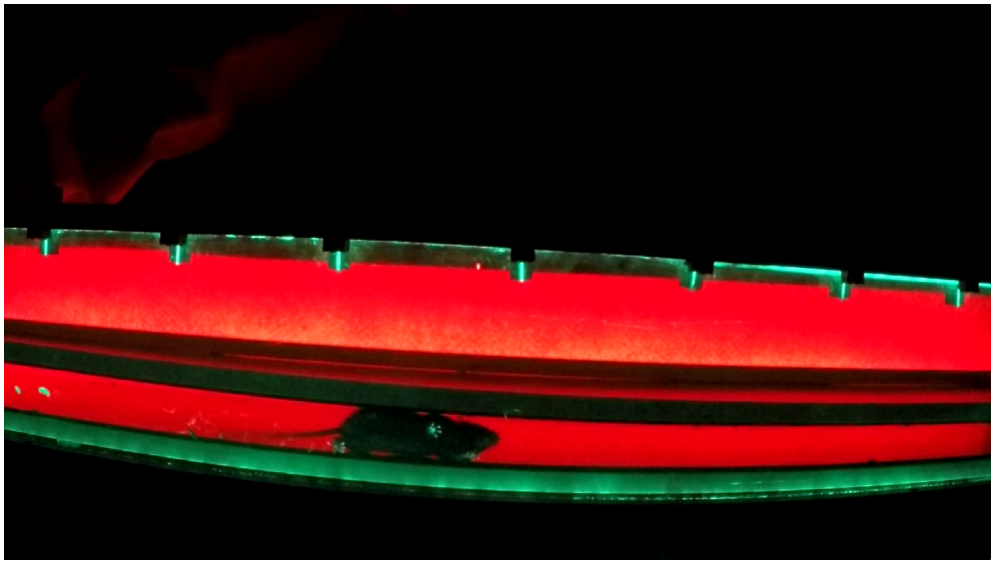

Figure 2.4: Preprocessing a sample image Part 1: Original image.

the image. Last, we mark the top and bottom walls of the glass trackway to crop the image to a fixed dimension and to remove the unnecessary areas. Figures 2.4, 2.5, 2.6 and 2.7 demonstrate the preprocessing steps applied to a sample image. The preprocessing parameters are saved using the system GUI and will be applied to the rest of the image sequences automatically as part of the preprocessing routine.

2.2.2 Paw Detection

In general, any digital image consists of a rectangular grid of pixels, where a pixel is the smallest controllable element of an image represented on the screen. In the *RGB* color imaging system, color is generally represented by the three-component intensities, viz., Red, Green, and Blue. A major task in paw detection is differentiating the pixels in an image representing the footfalls of a mouse from the rest of the pixels in that image. This task is simplified when there is good color contrast between the pixels representing the footfalls and the rest of the pixels. The previously described LED lighting for the gait acquisition helps in creating such contrast. Differentiating these pixels is achieved by detecting the pixels with sufficiently high green (G) values in an RGB image. More specifically, it is adequate to detect the pixels with sufficiently high G values within the rect-

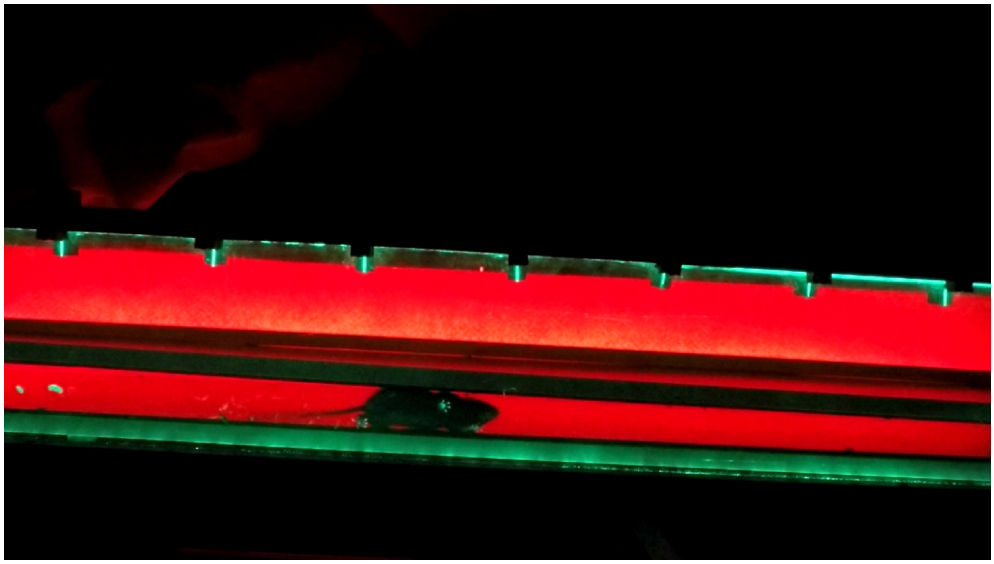

Figure 2.5: Preprocessing a sample image Part 2: Image after correcting for barrel distortion.

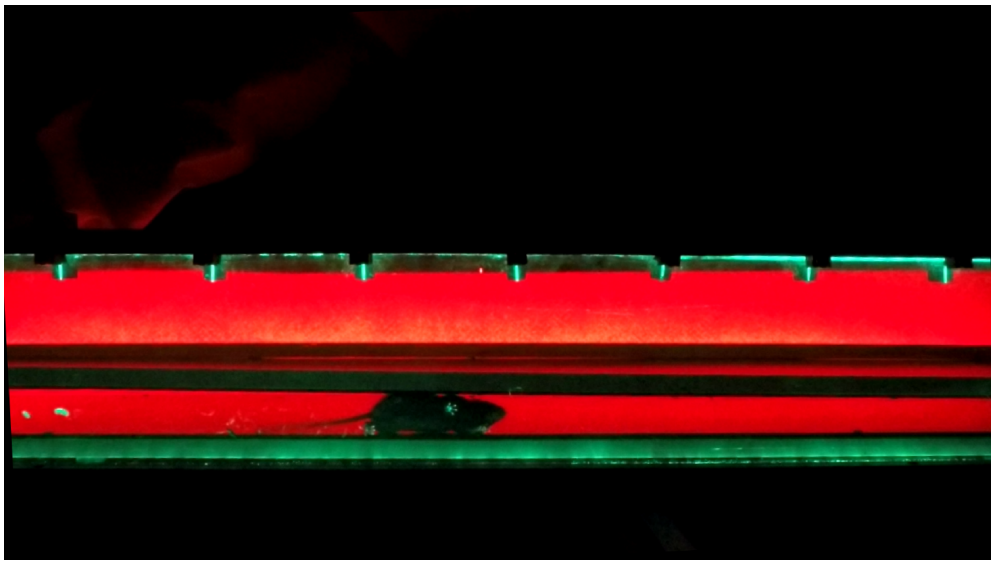

Figure 2.6: Preprocessing a sample image Part 3: Image after correcting for rotation.

Figure 2.7: Preprocessing a sample image Part 4: Cropped image showing just the glass trackway.

angular enclosure in an image of the glass trackway. At the beginning of the analysis, the top and bottom ends of the rectangular enclosure are manually set from a random frame in the video. To avoid detecting any possible noise, we use the thresholding technique to ensure that only the pixels representing the mouse's footfalls are differentiated. Algorithm 1 summarizes this technique. After thresholding for the G value, there are still many pixels remaining in the image that needs

Algorithm 1 Green Pixel Threshold

```
for Each input frame do
  for Each pixel within the rectangular enclosure do
    if Pixel's green value is greater than a given threshold T_g then
      Mark pixel as green-pixel
    end if
  end for
end for
```

to be identified as paws or otherwise. Proximity clustering, which links closely related pixels to form a larger object, is used for joining the digits of the paws together which are disjointed from the palm area. The distance between two pixels located at 2-D positions p and q is measured by the Euclidean norm, represented by $||q - p||$. Pixels within a predefined threshold distance are then grouped. This is summarized in Algorithm 2. Finally, the clusters that are assumed to be

Algorithm 2 Proximity Clustering

```
if ||green-pixel-cluster_2 - green-pixel-cluster_1|| is less than a given threshold T_d then
  Merge green-pixel-cluster_2 and green-pixel-cluster_1 as green-pixel-cluster_1
end if
```

paws, but are smaller or larger than some predefined thresholds, are filtered out. This eliminates the clusters formed by unwanted areas such as noise, dust, and mouse feces on the glass trackway. This procedure is summarized in Algorithm 3. After finalizing the paw clusters, the centroids of these clusters are recorded for each frame of an acquired video. The set of N pixels belonging to cluster k at the frame t is denoted $p_{k_t} = \{p_{k_{1,t}}, p_{k_{2,t}}, \ldots, p_{k_{N,t}}\}$, with coordinates of the pixel $p_{k_{i,t}}$

Algorithm 3 Cluster Sizing

if Cluster size is greater than a given threshold T_s **and** cluster size is smaller than a given threshold T_l **then**

Mark cluster as detected-paw

Return centroid for the detected-paw

end if

at t being (x_i, y_i). Here, the X direction is along the trackway while Y direction is perpendicular to the trackway. The centroid for the cluster k at time t is

$$C_{k_t} = \frac{1}{N} \sum_{i=1}^{N} p_{k_{i,t}} \,. \tag{2.1}$$

The centroid represents a 2-D location of the paw cluster at a given time.

Figure 2.8 and Figure 2.9 depict two sample images from a test video after the application of Algorithms 1– 3. Our system correctly identifies two paw clusters in Figure 2.8 and four paw clusters in Figure 2.9. The system automatically marks the detected paw clusters with bounding boxes and also indicates their positions inside the image by highlighting their respective X and Y coordinates. Later, in Subsection 2.3.1, we show that our system very rarely misses a paw cluster present in an image or falsely identifies some area in an image as a paw cluster. And, in Subsection 2.3.2, we explain how to choose particular threshold values in Algorithms 1–3 and how they affect the false alarms and missed detections. The system saves these values of X and Y coordinates for all detected paw clusters along with their respective frame numbers for further classification.

2.2.3 Paw Classification

The detected paw clusters need to be classified into four groups: viz., *Left-Front*, *Right-Front*, *Left-Rear*, and *Right-Rear*. There is a minimum of one and a maximum of four paw clusters present in each frame, and these paw clusters need to be matched with their corresponding paws in the previous and future frames. There may also be a mouse's tail or nose detected in some frames in addition to some random noise that may be present. This redundant data needs to be eliminated.

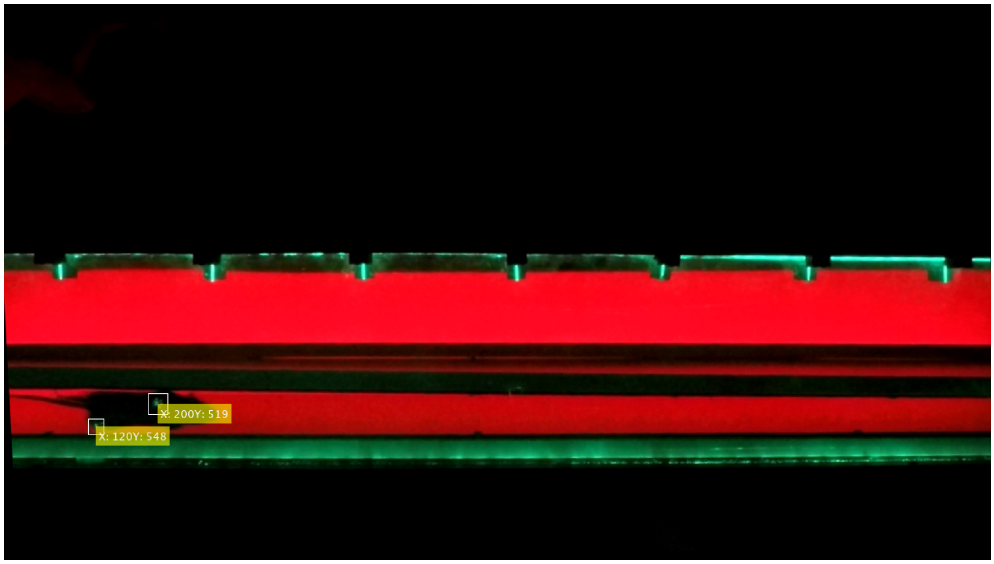

Figure 2.8: Sample image showing detected paws after application of Algorithms 1–3: Image with two detected paw clusters.

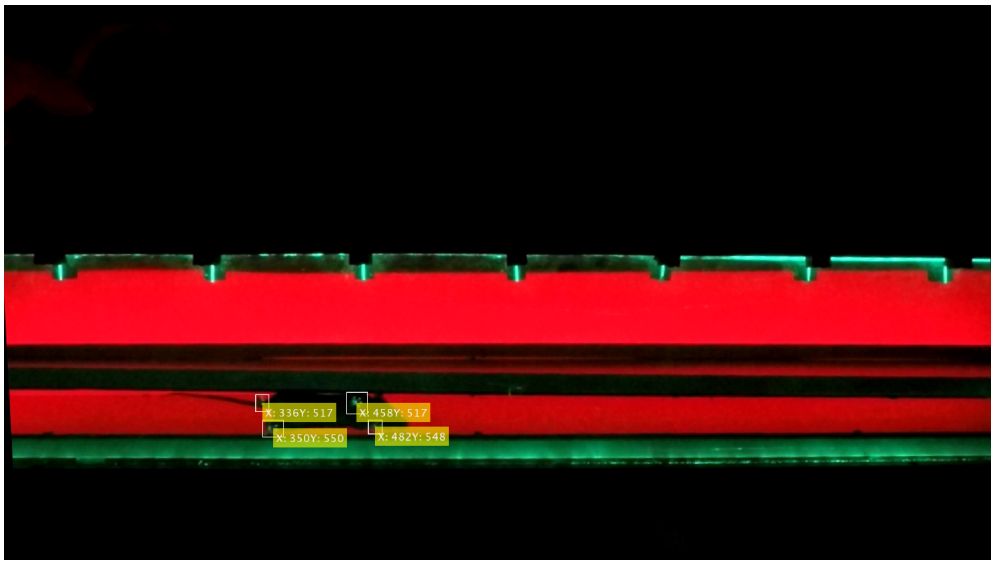

Figure 2.9: Sample image showing detected paws after application of Algorithms 1–3: Image with four detected paw clusters.

The main difficulty with the paw classification is that the rear paws often occupy similar locations that the front paws have previously occupied. This leads to many repeated coordinate entries in the recorded data even for different paws.

To solve this problem, first, we categorize the images from a video by the number of paw clusters detected in each frame. In the frames with four paw clusters, classification of individual clusters into *Left-Front*, *Right-Front*, *Left-Rear*, and *Right-Rear* is fairly easy. These are usually the frames where a mouse is standing on all four of its paws. The classification is done by using the X and Y coordinates for the detected paw clusters. Given the fixed left to the right direction with which a mouse is allowed to run on the trackway, the two clusters with higher X coordinates are classified as the front paws, and the remaining two are classified as the rear paws. The front and the rear paws are then further classified into the left and right paws based on their Y coordinates. The fixed direction of the mouse movement ensures that the right paws are the ones with the smaller Y coordinates and vice-versa. This is summarized in Algorithm 4. We move recursively from

Algorithm 4 Four Paw Classification

if Number of paw clusters detected in the frame is equal to four **then**
  Arrange detected paw clusters in increasing order of their X coordinates
  Arrange first two and last two detected paw clusters in increasing order of their Y coordinates

  Classify the first paw cluster as the *Right-Rear* paw
  Classify the second paw cluster as the *Left-Rear* paw
  Classify the third paw cluster as the *Right-Front* paw
  Classify the fourth paw cluster as the *Left-Front* paw
end if

the frames with classified paw clusters to their previous and successive frames. Any unclassified paw cluster within a predefined threshold distance from a classified paw is assigned to the same category as that of the classified paw. This is summarized in Algorithm 5. The process is repeated until there are no more new classifications. The remaining unclassified clusters at the end of this process are eliminated as redundant data. The thresholds for the distance and the frames were determined based on a large number of trial runs and cross verification.

Algorithm 5 Unclassified Paw Identification

A classified paw P_0 in frame F_0

for Each frame F with an index that is within a given threshold T_f of the index of the above frame F_0 **do**

  if There is an unclassified paw P in the frame F such that its location is within a given threshold T_p of the given classified paw P_0 **then**

    Set the class of the unclassified paw P to be equal to that of the classified paw P_0

  end if

end for

2.2.4 Parameter Extraction

Various gait parameters can be calculated from the classified paws. These gait parameters are useful for tracking any irregularities in gait. The gait parameter definitions commonly used in clinical gait analysis are listed for the reader's convenience in Table 2.1. These definitions are used by the CatWalk XT system and are only used to guide our system's proper estimation of these parameters.

Run Duration

The GoPro HERO3+™ camera is able to capture frames at a rate of 60 fps. The run duration (RD) is calculated as

$$RD = \frac{|t_f - t_l|}{60}\,, \tag{2.2}$$

where t_f and t_l are the times of the first and last frames with at least one detected paw.

Step Recognition

To compute the gait parameters, the steps must first be recognized accurately. The X coordinates of each classified paw are projected into a *Distance vs. Time* graph as shown in Figure 2.10. Each run is composed of two phases: *Contact phase* as recognized by the horizontal line parts, and *Swing phase* as recognized by diagonal line parts. The number of steps S is obtained directly from

Table 2.1: Definitions of gait parameters.

Parameter	Definition
Run duration	Time of finishing an entire run
Stride length	Distance between two consecutive travels in the same paw
Swing duration	Duration of no contact of a paw with the glass
Swing speed	Ratio of the stride length to the swing duration
Stance	Time the paws are in contact with the glass
Step cycle	Time between two consecutive initial contacts of the same paw
Duty cycle	Percentage of the stance over the sum of the stance and the swing duration
Cadence	Number of steps per time interval in the trial
Base of support	Distance between the fore limbs and the hind limbs at the maximum area
Diagonal dual support	Relative duration of simultaneous contact of two limbs with the glass
Three point support	Relative duration of simultaneous contact of three limbs with the glass
Four point support	Relative duration of simultaneous contact of four limbs with the glass
Average area	Average area of a paw contacting the glass
Average intensity	Average pressure of a paw contacting the glass

the number of contact phases during one entire run. The cadence (Cad) is calculated as

$$Cad = \frac{S}{RD}\,. \tag{2.3}$$

Stride Length, Swing Duration, Stance, Duty Cycle

Let t_f^j, t_m^j, and t_l^j represent first, mean, and last frames respectively for step j, and let the of centroids associated with step j for a particular paw be $C_{t_f^j}, C_{t_m^j}$, and $C_{t_l^j}$. Now stride length is calculated as

$$L_j = ||C_{t_m^j}(x,y) - C_{t_m^{j+1}}(x,y)||\,, \tag{2.4}$$

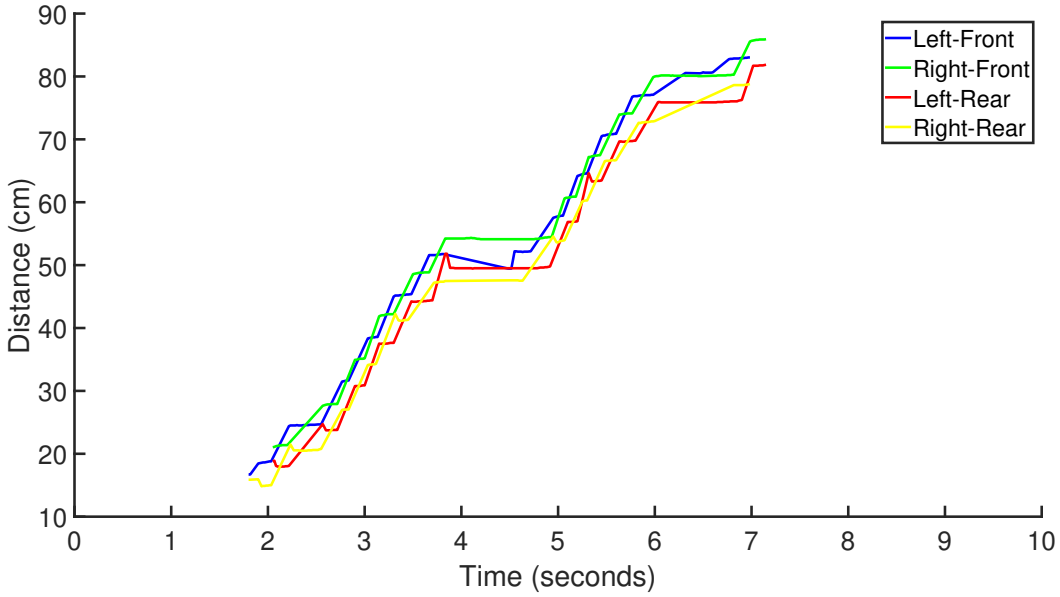

Figure 2.10: Distance travelled by each paw during a sample run.

and the *average stride length* for a particular paw is calculated as

$$L = \frac{1}{S}\sum_{i=1}^{S} L_i \,. \tag{2.5}$$

The *swing duration* is calculated as

$$D_j = \frac{|t_l^j - t_f^{j+1}|}{60} \,, \tag{2.6}$$

and the *average swing duration* for a particular paw is calculated as

$$D = \frac{1}{S}\sum_{i=1}^{S} D_i \,. \tag{2.7}$$

The *stance* is calculated as

$$R_j = \frac{|t_f^j - t_l^j|}{60} \,. \tag{2.8}$$

and the *average stance* for a particular paw is calculated as

$$R = \frac{1}{S} \sum_{i=1}^{S} R_i \,. \tag{2.9}$$

Once swing duration and stance are obtained, the *duty cycle* is calculated as

$$DC = \frac{\sum_{i=1}^{S} R_i}{\sum_{i=1}^{S} R_i + \sum_{i=1}^{S} D_i} \,. \tag{2.10}$$

Paw Support

Paw support indicates the relative duration of simultaneous contact of two or more limbs with the trackway. Different paw supports as defined in Table 2.1 are calculated by comparing the frame numbers for the steps of the required paws. Once we get the shared frames for the paws, we obtain the duration of their simultaneous contact simply by adding the total number of shared frames and dividing it by 60.

Paw Pressure

Paw pressure is represented in the form of an image by creating a surface plot for the green pixel values in a paw cluster. Smoothing of the curve is achieved by interpolating the values using a spline filter. The area of a paw cluster is calculated as the total number of green pixels in a paw cluster. The *average area* for a particular paw is then calculated as the mean of areas of all the paw clusters of the same category. The intensity of a paw cluster is calculated as the mean of green pixel values in a paw cluster. The *average intensity* of a particular paw is then calculated as the mean of intensities of all the paw clusters of the same category.

2.2.5 Visualization

The system GUI provides all control options and shows visualization results on the extracted gait parameters. Visualizations for some of the gait parameters acquired during the live animal study are shown in Figures 2.11, 2.12, 2.13, and 2.14. Visualization plays an important role in the

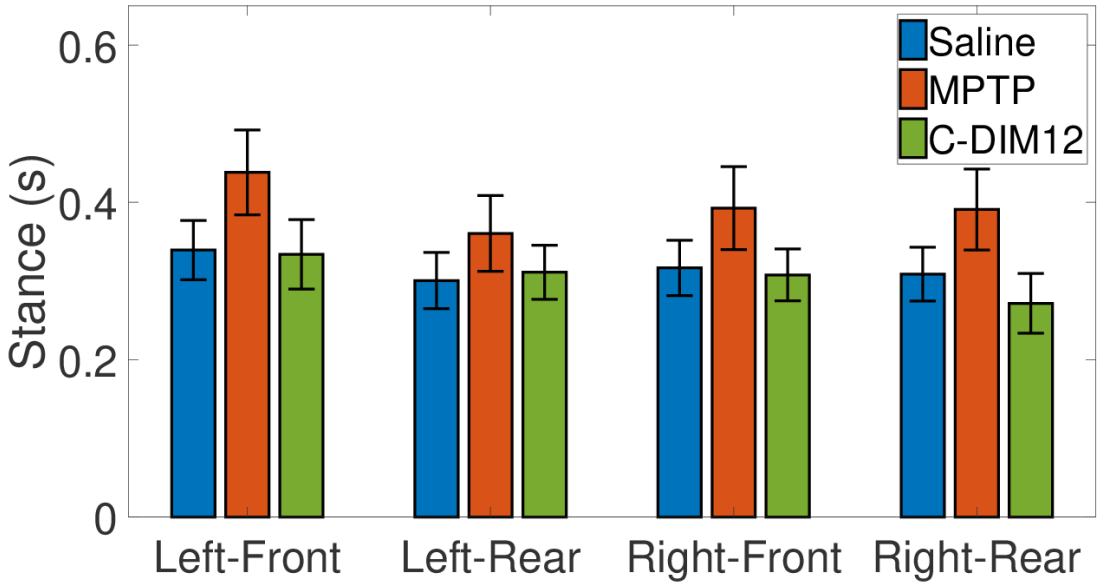

Figure 2.11: Graphical visualization of *Stance period* for the three mice groups on the 13th day. Error bars indicate 95% confidence intervals calculated over 20 measurements in 24 videos (n = 480).

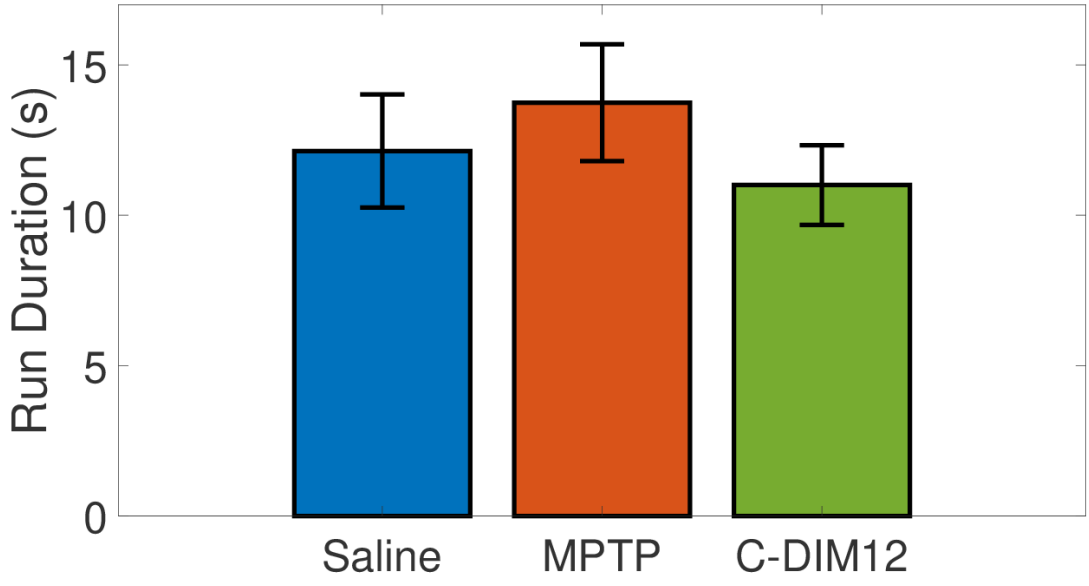

Figure 2.12: Graphical visualization of *Run duration* for the three mice groups on the 13th day. Error bars indicate 95% confidence intervals calculated over 20 measurements in 24 videos (n = 480).

quick and efficient analysis of large data. The system GUI can also be used to save the analyzed data in *.mat* or *.csv* formats for future use.

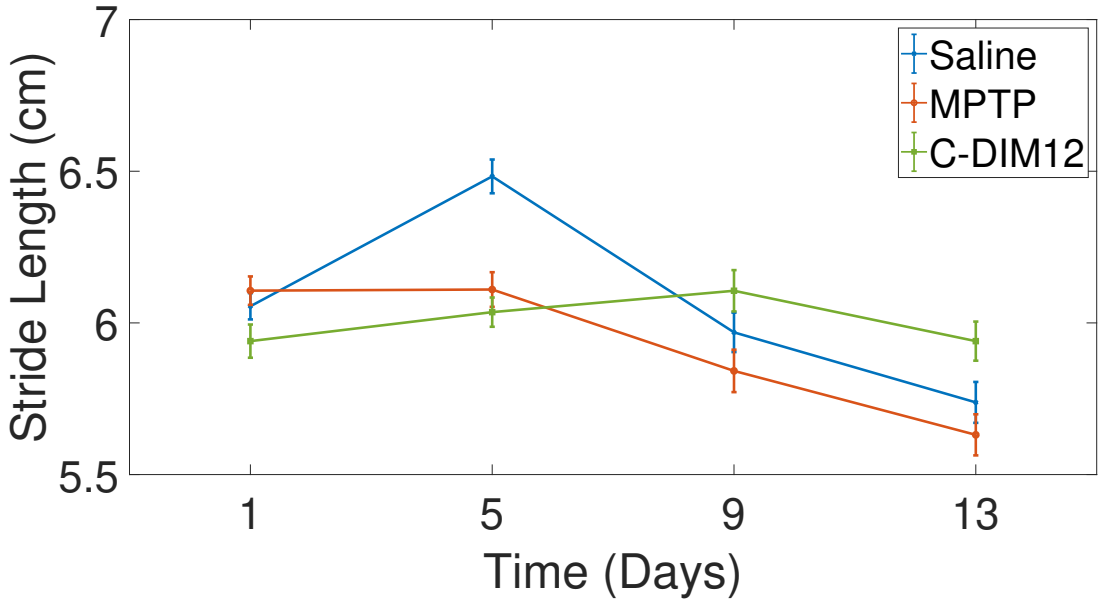

Figure 2.13: Graphical visualization of changes in *Stride lengths* for Left-Rear paw. Error bars indicate 95% confidence intervals calculated over 20 measurements in 24 videos (n = 480).

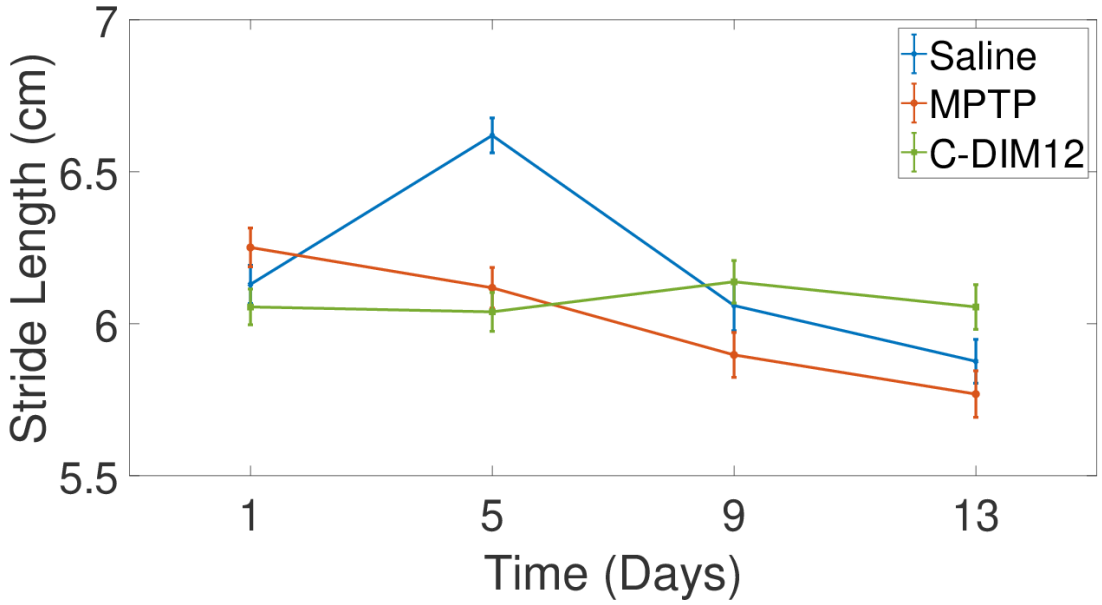

Figure 2.14: Graphical visualization of changes in *Stride lengths* for the Right-Rear paw. Error bars indicate 95% confidence intervals calculated over 20 measurements in 24 videos (n = 480).

2.3 Results

The system described above was used during a live animal study as one of the tools for assessing the recovery in C57BL/6 mice acquired from Charles River Laboratories (Wilmington, MA,

USA), which were medically induced with PD using MPTP and treated with C-DIM12. The study started with three different groups, each consisting of 12 healthy mice, and was conducted over a period of 14 days. The first group was treated with *Saline* throughout the experiment. The second group was injected with *MPTP* on days 1, 5, 9, and 13. The third group was treated with *MPTP* on the same days, along with oral gavage of *C-DIM12* compound dissolved in corn oil (or corn oil vehicle control). On the 14th day, the mice were terminated. Behavior was assessed at these time points using the gait acquisition system. The detailed training and habituation of mice to the trackway is given in [16]. After the treatment, each mouse was allowed to run on the trackway with optimally present lightning. The trackway was wiped with alcohol solution after each run to clean any excretion left by the mice during a run. Each mouse was allowed to run twice for the sake of obtaining better test results. A run with the least number of stops was selected for the analysis using the gait detection system. A total of 328 videos or approximately 196,800 frames were analyzed using our system.

From the video analysis, we acquired the footfall dataset for each frame in the video in terms of the categories Left-Front, Right-Front, Left-Rear, and Right-Rear, and X and Y coordinates for each footfall indicating the centroid of the footfall. Algorithms 1– 3 detect footfalls in the video and return a set of X and Y coordinates associated with the footfalls along with their respective frame numbers. Algorithms 4 and 5 classify the detected footfalls into the appropriate categories Left-Front, Right-Front, Left-Rear, and Right-Rear. To label the footfalls, we used a graphical image annotation tool called LabelImg [47]. LabelImg is an open-source tool written in Python and uses Qt for its graphical interface. First, we annotated every footfall in each frame of the video with its correct paw class and X and Y coordinates. Then we saved the annotations in XML files using LableImg. After saving the annotations, we cross-referenced the gait-detection data acquired using our autonomous gait-detection algorithms with the annotated gait data using LabelImg as described above. Finally, we used the system GUI to extract the gait parameters as defined in Table 2.1.

In this section, we show the results for the evaluation of the gait detection algorithms using H:M:F detection ratio, ROC analysis, and gait metrics acquired for the gait assessment.

2.3.1 Hit/Miss/False Detection Ratio

H:M:F detection ratio is the ratio of the percentage of positive paw identification (hit) to missed paw identification (miss) to false identification (false). A *positive* paw identification denotes correct identification and classification of paws present in each frame. A *missed* detection is when the system fails to detect one or more paws present in any frame. A *false* detection is incorrect identification of an object as a paw or misclassification of a detected paw. The goal of the gait acquisition and detection system is to generate close to 100% positive detection.

A missed detection can be caused by low contrast between a paw and the glass trackway. This happens when the area where a paw makes contact with the glass plate is not illuminated enough to be detected through the paw detection threshold explained in Algorithm 1. It may also happen when a paw is being lifted, and the area of the contact is so small that it does not get detected through the threshold in Algorithm 3. A missed detection causes a gap in the acquired gait data and can compromise the recorded gait metrics.

A false detection can be caused by some random noise or a large enough object other than a paw to be falsely labeled as a paw cluster. Most commonly it is caused by a mouse's nose or tail making contact with the glass trackway. False detection can also mean misclassification of some detected paw clusters. This usually happens when a mouse is trying to turn on the trackway during a run, making Algorithm 4 incorrectly classify one or more of the detected paw clusters.

We allowed the system to perform autonomous gait detection and paw classification as explained by the algorithms given in the previous section. For evaluation, we manually labeled all the footfalls present in each frame. Our described system correctly detected and classified each footfall across all the frames with an average H:M:F score of 92.1:2.3:5.6. The minimum H:M:F score was 84.3:0.9:14.9, where a large number of footfalls were misclassified due to the mouse turning in the

opposite direction midway through a run. The maximum H:M:F score was 96.0:1.4:2.6, where the mouse completed a near-perfect run with minimum stops.

2.3.2 Receiver Operating Characteristic Analysis

To evaluate how the system performs with the various threshold values for the variables used in the Algorithm 1– 3, we did an ROC analysis for each of these algorithms.

The gait detection system performs the role of a binary classifier by analyzing each pixel in each image of a video and returning a value of Positive or Negative for whether the pixel is a part of a paw cluster or not. For the ROC analysis, we chose a test set of carefully selected images from multiple videos. These images represent different possible scenarios while a mouse is walking on the glass trackway: all paws were present, only two paws were present, etc. After performing gait detection on the test set for each threshold, we analyzed the accuracy of gait detection using the manually labeled data. For the system performing as a binary classifier with a given threshold, there are four possible outcomes. Each correctly detected paw cluster is counted as 'true positive (TP)', incorrectly detected paw cluster as 'false positive (FP)', missed detection of a paw cluster as 'false negative (FN)', and the rest of the pixels as 'true negative (TN)'. In each image, only the center part of the image consisting of the glass trackway was analyzed, making it a total of 1720×70 possible positive and negative pixels in an image. Each bounding box of TP, FP, and FN is assumed to have a size of 20×20 pixels, and the rest of the pixels are labeled as TN. The *true positive rate (TPR)* for the classifier with a given threshold is then estimated as

$$TPR \approx \frac{TP}{TP + FN}.$$

The *false positive rate (FPR)* of the classifier is

$$FPR \approx \frac{FP}{FP + TN}.$$

ROC curves depict how a model's TPR (shown on the vertical axis) varies as a function of its FPR (shown on the horizontal axis). For a binary classifier, each threshold value corresponds to a single point (i.e., (FPR, TPR) pair) in the ROC plane. Conceptually, we may imagine varying threshold values for a variable from $-\infty$ to $+\infty$ and tracing a curve through ROC space. Computationally, this is not an optimal way of generating an ROC curve [48]. Instead, an (empirical) ROC curve for each variable in Algorithms 1– 3 is constructed by computing FPR and TPR at each threshold value, varied within a reasonable finite range, and interpolating by drawing a straight line between points corresponding to consecutive thresholds. The area under an ROC curve (AUC) is a commonly used summary statistic, typically ranging between 0.5 (random model) and 1 (perfect model) [49].

The ROC curves for Algorithm 1 and Algorithm 2 are shown in Figures 2.15 and 2.16, respectively, while Figures 2.18 and 2.20 depict the ROC curves for the minimum and maximum cluster size obtained using Algorithm 3. We can see that the AUC values for all the plots are above 0.95, which indicates an almost perfect detection model. Here, we also show the optimal threshold (OT) values in each ROC plot that achieved the maximum TPR and minimum FPR for the respective ROC curve. Then we construct confusion matrices for each of the OT values. The confusion matrices for the green pixel threshold, proximity cluster threshold, minimum cluster size, and maximum cluster size are shown in Figure 2.21(a), Figure 2.21(b), Figure 2.22(a), and Figure 2.22(b), respectively. The confusion matrices show that there were 0 FN detections for each of the OT values and the maximum FPR was 0.028 for the maximum-cluster-size threshold. Again, all these confusion matrices indicate that the performance of our described gait detection algorithms was near perfect.

2.3.3 Experimental Gait Analysis

The gait analysis system was used to assess locomotor function in the different mice groups during the experiment. The analysis showed an increase in stance period and run duration for the mice treated with MPTP, which were largely restored to control levels by day 13 in mice

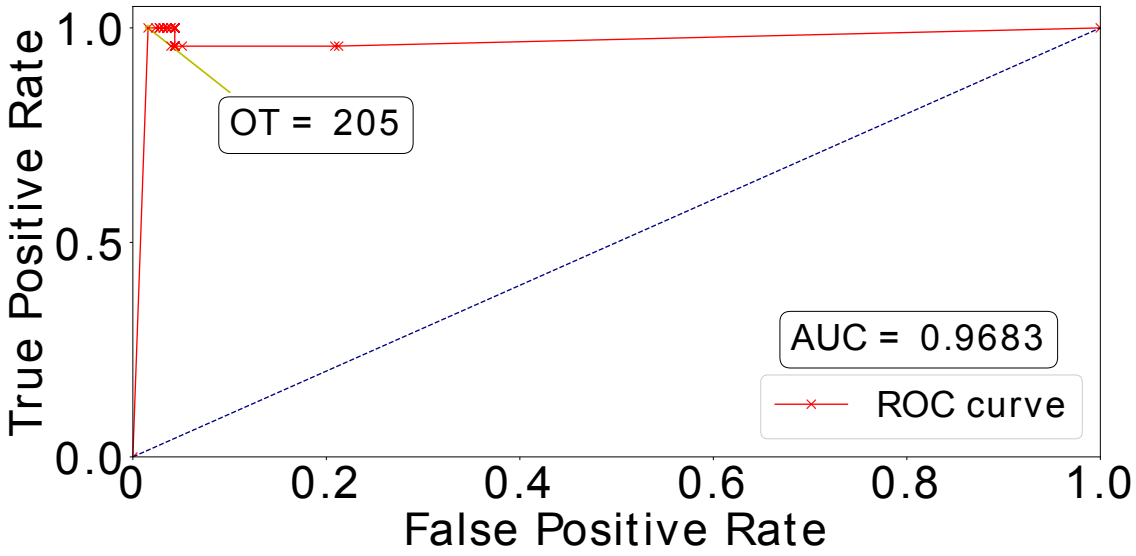

Figure 2.15: ROC curve with the optimal threshold (OT) value for *Green pixel threshold*.

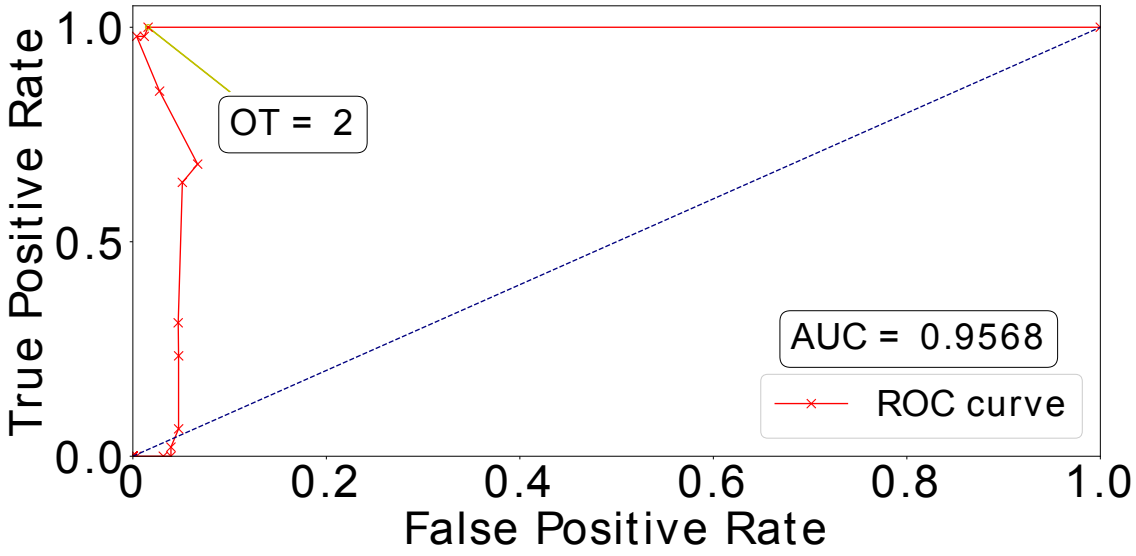

Figure 2.16: ROC curve with the optimal threshold (OT) value for *Proximity cluster threshold*.

treated with MPTP + C-DIM12. The comparison of stance period and run duration between the three groups is shown in Figure 2.11(a) and Figure 2.11(b), respectively. Stride length analysis also showed significantly shorter rear limb stride lengths in the MPTP group compared to stride lengths in the MPTP + C-DIM12 group from day 9 onward. The change in stride lengths for

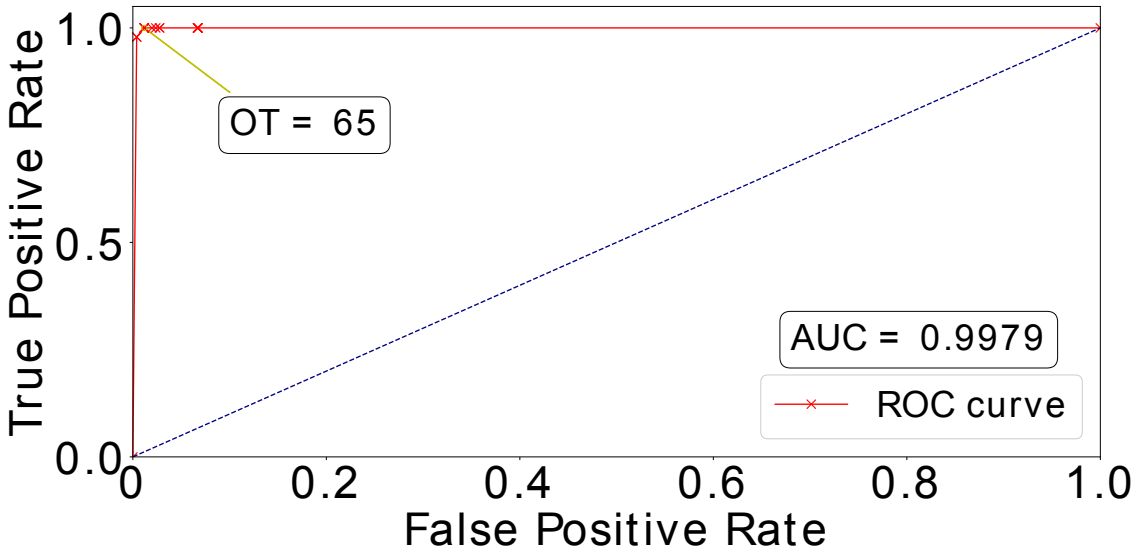

Figure 2.17: Caption

Figure 2.18: ROC curve with the optimal threshold (OT) value for *Minimum cluster size*.

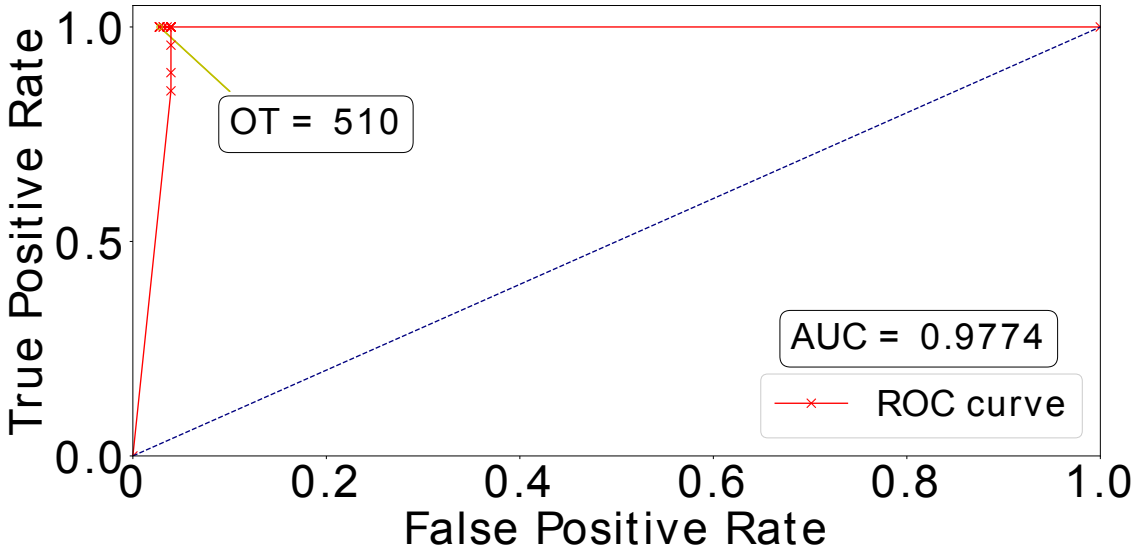

Figure 2.19: Caption

Figure 2.20: ROC curve with the optimal threshold (OT) value for *Maximum cluster size*.

Left-Rear and Right-Rear paws are shown in Figure 2.12(a) and Figure 2.12(b), respectively. The detailed gait analysis results are presented in [16]. Another recent study utilized the stride length

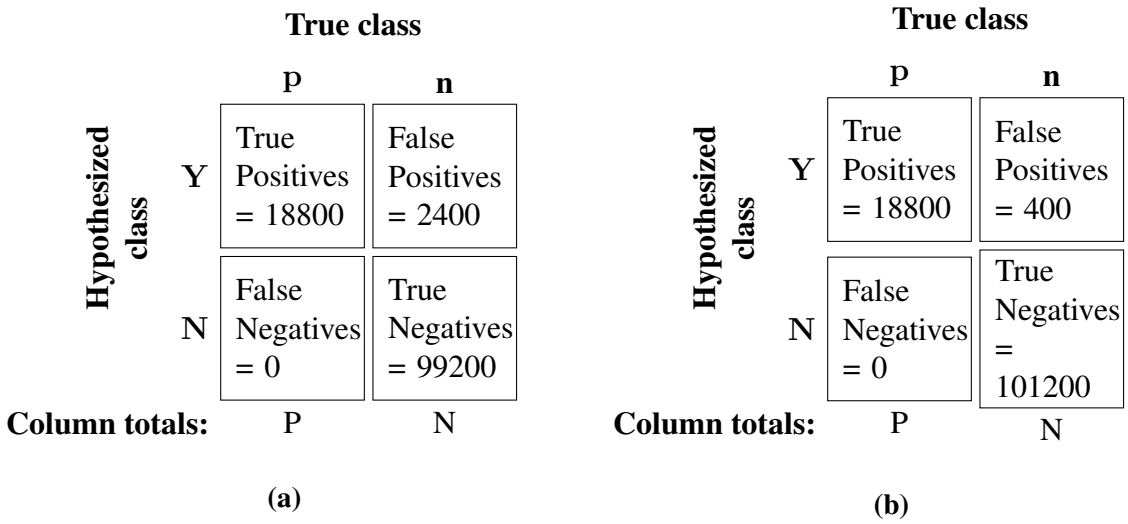

Figure 2.21: Confusion matrices for (a) Green pixel threshold = 205. (b) Proximity cluster threshold = 2.

calculated using our described system as one of the indicators to show that chronic exposure to Manganese is associated with neuroinflammation and extrapyramidal motor deficits resembling features of PD [50]. The values obtained for these gait metrics are in good agreement with the known literature [51–54]. These results show that our system can be reliably used to indicate gait abnormalities in mice.

2.4 Discussion

Quantitative gait analysis in mice during preclinical trials has always been one of the go-to methods for studying neurodegenerative diseases, such as PD or Huntington's disease. The commercial systems available in the market for such gait analysis are expensive and implement patented software. To address this issue, we developed an economic rodent gait analysis system. A detailed build guide for constructing this low-cost system was provided in [11]. In this work, we described the necessary image-processing algorithms for accurate gait detection and estimation. Our system autonomously detected and classified mice gait with very high accuracy when compared to the manually labeled actual footfalls. The gait metrics data acquired during the experiments indicate the robustness of the system for assessing mice gait abnormalities. The average processing time for one video using our described algorithms is approximately five minutes; thus,

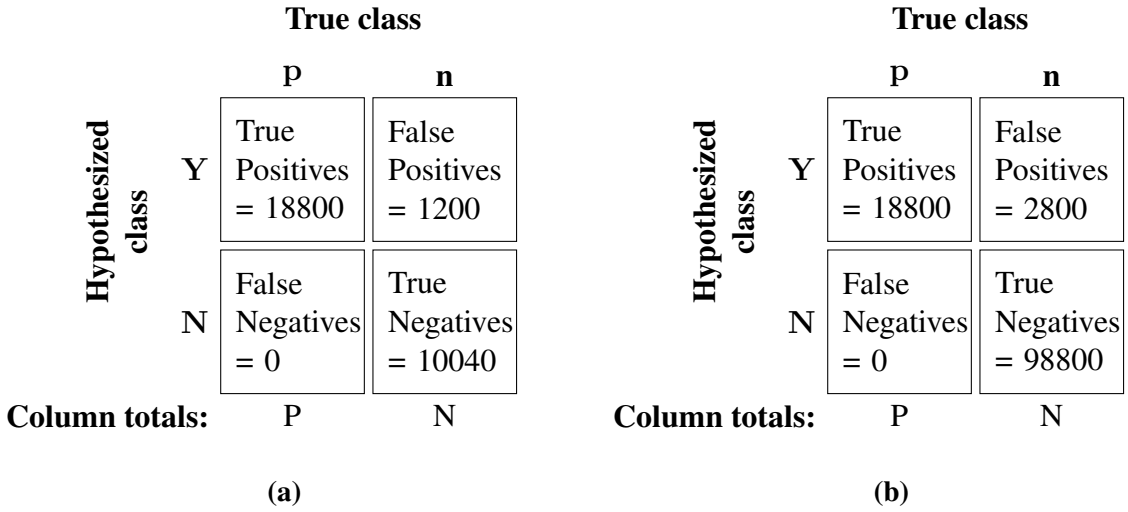

Figure 2.22: Confusion matrices for (a) Minimum cluster size = 65. (b) Maximum cluster size = 510.

it can replace the time-consuming efforts of manual labeling and measurements. Overall, our system has the potential to offer a cost-effective, autonomous gait assessment tool that can be used for studies involving tracking disease progression related to locomotor function and planning treatment strategies.

2.4.1 Economic Factors

We reiterate that our main goal is to build an economic gait analysis system that can provide a reliable alternative to the commercial systems available in the market. The typical cost of a commercial system for gait analysis in rodents is around $50,000. With the help of the detailed build guide described in [11] and the gait detection and estimation algorithms described in this chapter, a robust gait analysis system can be built for under $5,000. This cost includes an academic license for MATLAB. Although our software was written in MATLAB, the purpose of this chapter is to describe the underlying gait analysis algorithms. Indeed, our algorithms are also easily implemented in open-source systems such as Python. The system can be upgraded even further as needed by replacing the camera module, though prices for these high-speed, high-resolution cameras vary widely ($\approx$ $1,000–$25,000). Nonetheless, our described system provides a viable option for constructing a reliable gait analysis system under 10% of the cost of commercial systems.

2.4.2 Current Limitations

There are a few limitations to our described gait detection algorithms. One limitation is the dependence of Algorithms 1– 3 on adequate lighting conditions at the time of recording to create sufficient contrast between the mice footfalls and the rest of the image. Inadequate lighting conditions at the time of recording result in difficulty detecting all footfalls present in the videos, which cause loss of data and incorrect gait parameter calculations. This can be rectified by recording multiple sample videos before the actual experiment to ensure that the lighting conditions are adequate. Another limitation is that Algorithm 4 relies on the mouse moving in the designated left-to-right direction. If the mouse turns in the opposite direction midway through a run, the footfalls might be misclassified, resulting in incorrect gait parameters. To avoid such scenarios, all mice in our experiments were trained and habituated to the trackway [16]. Algorithm 5 relies on a predefined threshold to correctly associate unclassified footfalls with those of correctly classified footfalls. We determined this threshold manually to minimize the number of unclassified paw identification. Still, in certain scenarios, some detected footfalls might not be identified, resulting in loss of data and incorrect gait parameter calculations. We can tradeoff the probability of correctly identifying footfalls with that of false alarms by updating the threshold value in Algorithm 5. Overall, by training and habituating the mice to the trackway, ensuring the lighting conditions are adequate at the time of video acquisition, and updating the threshold value in Algorithm 5, we remedy the limitations of our described gait detection algorithms for reliable gait detection and estimation.

2.4.3 Future Scope

To further improve the results, one can easily replace the existing camera with an improved camera that can capture videos at a higher resolution and frame rate. In practice, the supporting threshold values for Algorithms 1– 3 need to be adjusted for optimal autonomous gait detection. With better resolution cameras, the detection algorithms can be improved to approach the ultimate goal of a 100:0:0 H:M:F detection score. Also, with faster computing processors and optimized software implementation, one can achieve real-time video processing. The GUI can be modified

as desired. Finally, machine learning techniques can be implemented for assessing irregularities in the mice gait. This is part of our ongoing study.

2.5 Conclusion

In this chapter, we have described a low-cost gait acquisition and analysis system for assessing gait abnormalities in mice. Our system automatically detects mice footfalls and calculates gait parameters, such as cadence, stride length, bodyweight distribution. Using this system we were able to extract various gait parameters to track the progression of PD in groups of mice under controlled dosages of a drug. We validated the detection performance of the system by performing an ROC analysis and calculating H:M:F detection ratio. The ROC and H:M:F analyses show that our system produces accurate gait detection. The results observed from the gait assessment study are in agreement with the known literature. This shows that our system can be reliably used for further studies involving mice gait abnormalities related to disease progression and treatment strategy.

2.6 Ethical Approval

All procedures were approved by the Colorado State University Institutional Animal Care and Use Committee (Protocol Number 17-7715A) and were conducted in compliance of National Institute of Health guidelines.

2.7 Funding

This research was funded by National Institutes of Health (NIH) grant R01ES021656 (RBT).

Chapter 3

Performance Study of Distance-Weighting Approach with Loopy Sum-Product Algorithm for Multi-Object Tracking [2]

3.1 Introduction

Since probabilistic data association (PDA) was introduced for tracking in a cluttered environment [56], engineers have been trying to optimize the data association (DA) technique for implementing in a Kalman filter (KF) [57] tracker. With the inception of the sum-product algorithm (SPA) [58], a new graphical approach for KF tracking was developed that is more efficient in dealing with the complex problem of multiple target tracking (MTT). To develop a robust algorithm that is also scalable for tracking multiple objects in clutter, in this chapter we examine the performance of the distance-weighting probabilistic data association (DWPDA) [59] in conjunction with the loopy sum-product algorithm (LSPA) [60].

The problem of DA is finding the correspondence between the targets and the measurements of uncertain origins. There are several approaches to tackle this problem, such as the nearest neighbor (NN), PDA, and joint probabilistic data association (JPDA) [56, 61, 62]. The NN approach is one of the easiest and uses at any time only the nearest measurement to the predicted measurement as if it were the one originated from the target of interest [61]. The NN approach is suboptimal and can only work well in case of widely spaced targets, accurate measurements, and few false alarms. Finding some association between all the targets and measurements is computationally expensive. Therefore a gate is formed around the predicted measurement based on some predefined threshold called the validation region. The PDA algorithm obtains the probability of each measurement

[2]The material in this chapter was published in the MDPI Sensors special issue *Multi-Sensor Fusion for Object Detection and Tracking* [55].

lying inside the validation region as being the correct measurement and updates the state estimates according to an appropriately modified tracking filter, called PDAF [56]. One major drawback of the PDA algorithm is that it treats every measurement inside the validation region of a target of interest as if it were originated from the said target or as a Poisson-distributed false alarm. PDA does not take into account the possibility that a validated measurement for one target might be a measurement originated from another nearby target. The JPDA algorithm improves this deficiency of the PDA algorithm by computing the association probabilities for the validated measurements from the joint likelihood functions corresponding to all feasible joint events such that no more than one measurement originates from each target [62]. However, with an increasing number of targets and/or clutter, the JPDA algorithm becomes impractical for real-time applications due to its combinatorial complexity because it considers all feasible joint events of measurements to targets to obtain the joint association probabilities [63].

PDA and JPDA are zero-scan algorithms, meaning that all hypotheses are combined after computation of the probabilities, for each target at each time step [56, 62]. Alternatively, multiple hypothesis tracking (MHT) is a deferred decision logic algorithm. In the case of a conflicting target-measurement association, the MHT algorithm formulates alternative data association hypotheses instead of choosing the best-combined hypothesis. The ambiguities between the alternative hypotheses are resolved using the measurements that arrive later into the future [64]. JPDA is shown to produce reasonable results compared to the computationally expensive multi-scan MHT algorithm [62–65]. Both the PDA and the JPDA algorithms utilize known targets for forming validation gates in the measurement space to compute the posterior probabilities and are therefore categorized as target-oriented approaches. There exist measurement-oriented algorithms, such as the one described in [66] and others, where hypotheses are formed for each measurement to have originated from a known target, a new target, or clutter. Both sets of algorithms have shown to yield an equivalent expression for posterior probabilities with appropriate assumptions [62, 66, 67].

As we realize that calculating DA probabilities for MTT is a process that involves a complicated global function of many variables that can be broken down into a product of local functions each

of which depends on a subset of the variables, we look for alternate methods that can exploit this trait. SPA is one such method that operates in a factor graph [68] and attempts to compute, either exactly or approximately, various marginal functions associated with the global function [58]. SPA operates by passing messages, called beliefs, between the nodes of a factor graph, i.e., variables and local functions, that involve summations and products of factors. Implementation of a factor graph for the MTT DA problem requires a loopy sum-product solution which is neither guaranteed to converge nor produce the correct marginal functions if convergence occurs. A simultaneous target-oriented and measurement-oriented factor graph formulation of the DA problem has been shown that is guaranteed to converge and results in accurate association probabilities [60, 69]. A simplified implementation of LSPA results in a significant reduction in computational complexity without loss of accuracy [69, 70], which makes LSPA more appealing than PDA or JPDA for DA.

In recent years, the use of LSPA for DA in MTT has been gaining traction. A belief-propagation approach to multi-target multi-sensor tracking is proposed in [71] by formulating a detailed factor graph where every single target and data-association variable is modeled as an individual node. To tackle the problem of tracking an unknown number of targets, where targets appear and disappear, an LSPA-based method is proposed in [72] that creates augmented target states for keeping track of existing and non-existing targets. A more comprehensive derivation of the message-passing algorithm for multi-sensor multi-target tracking is given in [73, 74]. An extension of the LSPA-based MTT framework for target estimation is given in [75] that exploits additional target information provided by a classifier. All these efforts highlight the potential of SPA for solving the complicated problem of DA in MTT.

The data association probabilities obtained using PDA are a crucial part of the beliefs being passed between nodes during the convergence of LSPA. Modifying the DA probabilities according to a weighting scheme based on distances between the predicted and validated measurements has been shown to enhance the tracking accuracy of PDAF while tracking a single target in a densely cluttered environment [59]. In this chapter, we want to explore whether this distance-weighting approach for PDA, when integrated with LSPA, would improve the performance of LSPA even

further. To evaluate this possibility, we formulate a distance-weighting LSPA (DWLSPA) and compare its performance in terms of tracking accuracy and computation time against DWPDA, JPDA, and LSPA for tracking multiple targets crossing at a small angle in different density clutter. Our results turn out to be contrary to expectations.

The main contribution of this performance study is to explore the idea of modifying one of the building blocks for a state-of-the-art data-association algorithm [70] for multi-target tracking and to compare the tracking accuracy of the modified algorithm to that of the original algorithm. The distance-weighting modification analyzed in this chapter is based on a recent successful implementation of a similar modification to one of the earliest data-association frameworks for tracking targets in cluttered environments [59]. In doing so, we develop the mathematical formulation for each data-association filter being considered and evaluate its performance in terms of tracking accuracy and computation time over a wide range of easy-to-replicate multi-object-tracking scenarios.

We introduce our notations and the target tracking system in Section 3.2. In Section 3.3, we describe the problem of PDA, DWPDA, JPDA, and LSPA. We present simulation results for these methods and our analysis in Section 3.4. In Section 3.5, we conclude the chapter with our observations and final remarks regarding the performance of these methods.

3.2 Target Tracking Dynamic System Model and Assumptions

We describe the classic data association problem in which a single sensor surveils a large number of targets. The number of targets under surveillance is assumed to be known and is denoted by N_T. The measurements are comprised of possible target detections and false alarms. A target is detected with a known probability of detection P_D and is independent of time. The false alarms, modeled according to the Poisson point process with a known spatial density λ, are uniformly distributed in the measurement space. A validation region, with the threshold γ corresponding to certain gate probability P_G, is set up at every sampling time around the predicted measurement and possibly several measurements fall in it. Each algorithm differs in how these measurements

are used (or not) in the estimation of the state of the target. We assume that each target can generate at most one measurement and each measurement can have only one source.

We denote by $x_i(k)$, $i \in \{1, \ldots, N_T\}$, the state of i-th target of dimension n_x at time k. The complete set of target states at time k is denoted by $X(k) = (x_1(k), \ldots, x_{N_T}(k))$. At time k, the total number of measurements is denoted by $M(k)$. We denote by $z_j(k)$, $j \in \{1, \ldots, M(k)\}$, the value of j-th measurement of dimension n_z. The complete set of measurements at time k is denoted by $Z(k) = (z_1(k), \ldots, z_{M(k)}(k))$, and the complete set of measurements up to and including time k is denoted by $Z^k = (Z(1), \ldots, Z(k))$.

The state and measurement equations are assumed linear with additive zero-mean white noise with known covariances. The state of target t evolves in time according to the equation

$$x_t(k+1) = F(k)x_t(k) + G(k)u(k), \tag{3.1}$$

and the true measurement for target t is given by

$$z_t(k) = H(k)x_t(k) + w(k) \tag{3.2}$$

where $u(k)$ and $w(k)$ are zero-mean mutually independent white Gaussian noise sequences with known covariances $Q(k)$ and $R(k)$, respectively. Functions $F(k), G(k)$, and $H(k)$ are known matrices for state transition, noise gain, and sensor, respectively. The past information (through time $k-1$) about the target t is assumed to be known and summarized approximately by the Gaussian posterior

$$p[x_t(k-1)|Z^{k-1}] = \mathcal{N}[x_t(k-1); \hat{x}_t(k-1|k-1), P_t(k-1|k-1)] \tag{3.3}$$

where $\hat{x}_t(k-1|k-1)$ and $P_t(k-1|k-1)$ are the state estimate and covariance for target t.

3.3 Algorithm Description

3.3.1 Probabilistic Data Association Filter

The PDA algorithm calculates the association probabilities for each validated measurement at the current time for the target of interest. PDA assumes that all the validated measurements are generated by either the target of interest or clutter. The association probabilities are used for calculating the mean squared error (MSE) estimate and covariance of the target's state. An appropriately modified KF, called PDAF, is used to account for the uncertainty of origins for the validated measurements while estimating the state of the target. The algorithm can be given as follows.

Prediction

The prediction of the state and measurement of target t at time k is done as in the KF, i.e.,

$$\hat{x}_t(k|k-1) = F(k-1)\hat{x}_t(k-1|k-1) \tag{3.4}$$

$$\hat{z}_t(k|k-1) = H(k)\hat{x}_t(k|k-1). \tag{3.5}$$

The covariance of the predicted state for target t is

$$P_t(k|k-1) = F(k-1)P_t(k-1|k-1)F(k-1)' + G(k-1)Q(k-1)G(k-1)'. \tag{3.6}$$

Here, $\hat{x}_t(k-1|k-1)$ and $P_t(k-1|k-1)$ are available from Equation (3.3). The innovation covariance of the target t (for the correct measurement) is

$$S_t(k) = H(k)P_t(k|k-1)H(k)' + R(k). \tag{3.7}$$

Measurement Validation

The validation region for target t at time k is the elliptical region given by

$$\mathcal{V}_t(k,\gamma) = \{z \in Z(k) : [z - \hat{z}_t(k|k-1)]' S_t(k)^{-1} [z - \hat{z}_t(k|k-1)] \leq \gamma\}. \tag{3.8}$$

Here, γ can be obtained according to

$$V(k) = c_{n_z} |\gamma S(k)|^{1/2} \tag{3.9}$$

where $V(k)$ is the volume of the validation region given by Equation (3.8) and c_{n_z} is the volume of the unit hypersphere of dimension n_z [67]. The validated measurements for target t according to Equation (3.8) are

$$Z_t(k) \triangleq \{z_i(k)\}_{i=1}^{m_t(k)} \tag{3.10}$$

where $m_t(k)$ is the number of validated measurements for target t at time k.

Data Association Probabilities

The association probability β_i^t for each validated measurement of target t is obtained as

$$\beta_i^t(k) = \begin{cases} \frac{\mathcal{L}_i(k)}{1 - P_D P_G + \sum_{j=1}^{m_t(k)} \mathcal{L}_j(k)} & i = 1, \ldots, m_t(k) \\ \frac{1 - P_D P_G}{1 - P_D P_G + \sum_{j=1}^{m_t(k)} \mathcal{L}_j(k)} & i = 0 \end{cases} \tag{3.11}$$

where $i = 0$ indicates probability of associating none of the validated measurements to the target. In Equation (3.11), $\mathcal{L}_i(k)$ the likelihood ratio (LR) of measurement $z_i(k)$ originating from the target t vs. from clutter and is obtained as

$$\mathcal{L}_i(k) \triangleq \frac{\mathcal{N}[z_i(k); \hat{z}_t(k|k-1), S_t(k)] P_D}{\lambda}, \quad z_i(k) \in Z_t(k). \tag{3.12}$$

State Estimation

The state estimation for target t is done according to PDAF by

$$\hat{x}_t(k|k) = \hat{x}_t(k|k-1) + W_t(k)v_t(k) \tag{3.13}$$

where the combined innovation for target t is

$$v_t(k) = \sum_{i=0}^{m_t(k)} \beta_i^t(k)v_i^t(k), \tag{3.14}$$

and

$$v_i^t(k) = z_i(k) - \hat{z}_t(k|k-1) \tag{3.15}$$

is the innovation of measurement for $z_i(k) \in Z_t(k)$. The filter gain is calculated as

$$W_t(k) = P_t(k|k-1)H(k)'S_t(k)^{-1}. \tag{3.16}$$

The covariance estimation for target t associated with the updated state is

$$P_t(k|k) = \beta_0(k)P_t(k|k-1) + [1-\beta_0^t(k)]P_t^c(k|k) + \tilde{P}_t(k) \tag{3.17}$$

where the covariance of the state updated with the correct measurement, P^c, and the innovation spread, $\tilde{P}$, for target t are given by

$$P_t^c(k|k) = P_t(k|k-1) - W_t(k)S_t(k)W_t(k)' \tag{3.18}$$

and

$$\tilde{P}_t(k) = W_t(k)\left[\sum_{i=1}^{m_t(k)} \beta_i^t(k)v_i^t(k)v_i^t(k)' - v_t(k)v_t(k)'\right]W_t(k)', \tag{3.19}$$

respectively.

The estimated state in PDAF from Equation (3.13) is for a single target x_t. To estimate the state of every target x_t, $t \in \{1, \ldots, N_T\}$, we need to apply PDAF to the targets one by one sequentially. This means that for each target x_t, $t \in \{1, \ldots, N_T\}$, we perform state and measurement prediction, form a validation region around the predicted measurement and prune off potentially unrelated measurements, obtain data-association probabilities for the validated measurements, and, finally, update the state estimate according to Equation (3.13). The order in which PDAF is applied to the targets is irrelevant because the outcome is independent of the order.

3.3.2 Distance-Weighting Probabilistic Data Association Filter

In the PDA algorithm, the association probability β is calculated as the likelihood ratio of a validated measurement to have originated from a target vs. from clutter. Chen et al. [59] have pointed out that a true measurement from a target of interest is more likely to be near the target's predicted measurement. Since false alarms are uniformly distributed in the measurement space, the measurement nearest to that of the predicted measurement from a target of interest should carry more weight while calculating the association probabilities for the target. The distance weight proposed in [59] is calculated as

$$\Delta_i^t(k) = \frac{1/\delta_i^t(k)}{\sum_{i=1}^{m_t(k)} 1/\delta_i^t(k)} \tag{3.20}$$

where $\delta_i^t(k)$ is the Mahalanobis distance between validated measurement i and predicted measurement of target t at time k. The Mahalanobis distance is calculated as the norm of the innovation squared and is given by

$$\delta_i^t(k) = v_i^t(k)' S_t(k)^{-1} v_i^t(k) \tag{3.21}$$

where $v_i^t(k)$ is the innovation as defined in Equation (3.15).

The new data association probabilities for target t at time k are calculated as

$$\mathcal{B}_i^t(k) = \beta_i^t(k)\Delta_i^t(k) \quad i = 1, \ldots, m_t(k) \tag{3.22}$$

and are normalized according to

$$\mathcal{B}_i^t(k) = \mathcal{B}_i^t(k) / \sum_{j=0}^{m_t(k)} \mathcal{B}_i^t(k) \quad i = 0, 1, \ldots, m_t(k). \tag{3.23}$$

The data association probabilities obtained in Equation (3.23) are used for estimating target state as explained previously in PDAF Section 3.3.1. Similar to PDAF, the DWPDA filter is designed for tracking a single target. For the purpose of tracking multiple targets, we need to update the states of the targets one by one.

3.3.3 Joint Probabilistic Data Association Filter

The PDA algorithm is designed for tracking a single target in clutter. Because the PDA algorithm assumes all the incorrect measurements in the validation region of a target of interest are clutter, it is susceptible to scenarios where these incorrect measurements might have originated from another nearby target. Situations may arise in MTT where the validation regions of nearby targets may overlap for several time frames in a row and cause persistent interference that can lead to track deterioration or track loss altogether with PDA. JPDA improves on PDA by calculating joint association probabilities at each time step. Marginalized association probabilities for each target can be obtained using these joint association probabilities, which are then used for estimating the state of each target.

Measurement Validation

A major difference between the PDA and the JPDA algorithm is that no individual validation gates will be assumed for the various targets. A uniform validation region for all targets is obtained by taking a union of individual validation gates. This way each measurement is assumed to be validated for each target and false alarms will be assumed to be uniformly distributed across the entire validation region. Predictions for each target are done at each time step similar to the PDA algorithm using Equations (3.4)–(3.7).

Once the validated measurements are obtained for each target using Equations (3.8) and (3.10), the combined validated measurements at time k are given according to

$$Z_{N_T}(k) = \bigcup_{i=1}^{N_T} Z_i(k). \tag{3.24}$$

The number of combined validated measurements is $m_{N_T}(k)$ at time k.

The Validation Matrix

A validation matrix Ω of size $m_{N_T} \times (N_T + 1)$ with binary elements is created using the combined validated measurement.

$$\Omega(k) = [\omega_{jt}] \quad j = 1, \ldots, m_{N_T}(k); \quad t = 0, 1, \ldots, N_T \tag{3.25}$$

where

$$\omega_{jt} = \begin{cases} 1 & \text{if } z_j(k) \in Z_t(k) \\ 0 & \text{otherwise.} \end{cases} \tag{3.26}$$

The first column of $\Omega(k)$ corresponding to $t = 0$ is all unity indicating that each measurement could be a false alarm.

The Feasible Joint Events

The joint association events for time k are given by

$$\theta(k) = \bigcap_{j=1}^{m_{N_T}} \theta_{jt_j} \quad j = 1, \ldots, m_{N_T}(k); \quad t = 0, 1, \ldots, N_T \tag{3.27}$$

where in the event of θ_{jt}, measurement j is originated from target t_j. An event matrix $\hat{\Omega}$ consisting of the units in Ω corresponding to the association in θ is used to represent a joint association event θ.

$$\hat{\Omega}(\theta(k)) = [\hat{w}_{jt}(\theta(k))] \quad j = 1, \ldots, m_{N_T}(k); \quad t = 0, 1, \ldots, N_T \tag{3.28}$$

where

$$\hat{\omega}_{jt}(\theta(k)) = \begin{cases} 1 & \text{if } \theta_{jt} \in \theta(k) \\ 0 & \text{otherwise.} \end{cases} \tag{3.29}$$

Not all joint association events are feasible joint events. As per our target tracking model assumptions, feasible association events are only those where no more than one measurement is associated with each target. Therefore a feasible association event is the one which satisfies the following two conditions:

1. A measurement can have only one source, i.e.,

$$\sum_{t=0}^{N_T} \hat{\omega}_{jt}(\theta(k)) = 1 \quad \forall j. \tag{3.30}$$

2. Each target can generate at most one measurement, i.e.,

$$\delta_t(\theta(k)) \triangleq \sum_{j=1}^{m_{N_T}} \hat{\omega}_{jt}(\theta(k)) \leq 1 \quad t = 1, \ldots, N_T. \tag{3.31}$$

$\delta_t(\theta(k))$ in Equation (3.31), called the target detection indicator, indicates if a measurement is associated with target t in event $\theta(k)$. Similarly, we define the measurement association indicator $\tau(\theta(k))$ to indicate if measurement j is associated with a target in event $\theta(k)$.

$$\tau_j(\theta(k)) \triangleq \sum_{t=1}^{N_T} \hat{\omega}_{jt}(\theta(k)) \quad j = 1, \ldots, m_{N_T}(k). \tag{3.32}$$

We can obtain the number of false alarms in event $\theta(k)$ as

$$\phi(\theta(k)) = \sum_{j=1}^{m_{N_T}(k)} [1 - \tau_j(\theta(k))]. \tag{3.33}$$

Joint Data Association Probabilities

The joint data association probabilities are calculated as

$$P\{\theta(k)|Z^k\} = \frac{1}{c}[Z_{m_{N_T}(k)}(k)|\theta(k), m_{N_T}(k), Z^{k-1}]P\{\theta(k)|m_{N_T}(k)\} \tag{3.34}$$

where $P\{\cdot\}$ indicates the probability mass function (PMF) and c is the normalization constant. The marginal association probabilities are obtained from the joint DA probabilities by summing over all the joint events according to Equation (3.29). The marginal association probabilities at time k are then given as

$$\beta_j^t(k) = \sum_{\theta(k)} P\{\theta(k)|Z^k\}\hat{\omega}_{jt}(\theta(k)) \quad j = 1, \ldots, m_{N_T}(k); \quad t = 0, 1, \ldots, N_T \tag{3.35}$$

$$\beta_0^t(k) = 1 - \sum_{j=1}^{m_{N_T}(k)} \beta_j^t(k) \quad t = 0, 1, \ldots, N_T \tag{3.36}$$

where the probability β_j^t represents that the measurement j is associated with target t and β_0^t indicates that none of the validated measurements is associated with target t. The expression for the joint association probabilities in Equation (3.35) can be given in terms of the variables defined in JPDAF Section 3.3.3 as

$$P\{\theta(k)|Z^k\} = \frac{1}{c_1} \prod_{j=1}^{m_{N_T}(k)} \{\lambda^{-1}[\Lambda_j^{t_j}(k)]\}^{\tau_j(\theta(k))} \prod_{t=1}^{N_T} (P_D)^{\delta_t(\theta(k))}(1 - P_D)^{1-\delta_t(\theta(k))} \tag{3.37}$$

where $\Lambda_j^{t_j}$ is the Gaussian density of measurement j associated with target of index t_j given by

$$\Lambda_j^{t_j}(k) = \mathcal{N}[z_j(k); \hat{z}_{t_j}(k|k-1), S_{t_j}(k)] \tag{3.38}$$

and c_1 is the new normalization constant. The marginal association probabilities obtained in Equations (3.35) and (3.36) are used in conjunction with the state estimation equations given in PDAF Section 3.3.1 for estimating state of each target separately.

3.3.4 Loopy Sum-Product Algorithm

Graphical models can be used for representing the joint probability distributions of many variables efficiently by exploiting factorization. For SPA, factor graphs are utilized for representing the DA relations between multiple targets and their validated measurements. SPA is then conducted on the resulting loopy factor graph for obtaining the marginal DA probabilities. The algorithm for the loopy-SPA can be given as follows.

Belief Propagation in Factor Graphs

Kschischang et al. [58] demonstrated how factor graphs can be used to interpret a variety of algorithms such as the KF, the Viterbi algorithm, the Hidden Markov Model, etc. A factor graph is a standard bipartite graphical representation of a mathematical relationship between random variables and local functions. While formulating a factor graph to express the structure of the factorization of a global function of many variables into several local functions, each node represents each random variable $n \in \mathfrak{N}$, each factor represents each local function $f \in \mathfrak{F}$, and an edge connects a node n to a factor f if and only if n is an argument of f.

The SPA algorithm, also called as Belief Propagation (BP), passes messages, called beliefs, between nodes and factors in an iterative manner for conducting optimal inference on a tree-structured factor graph. We denote by $\psi_{\cdot}(\cdot)$ the joint probability distribution function, by $\mu_{n\to f}(x_n)$ the message sent from node $n \in \eta_f$ to factor f, by $\mu_{f\to n}(x_n)$ the message sent from factor f to node $n \in \eta_f$, by $\eta_f \subseteq \mathfrak{N}$ the set of neighboring nodes of f, and by $\eta_n = \{f \in \mathfrak{F} | n \in \eta_f\}$ the factors involving node n. The message computation performed by SPA can be given as

$$\mu_{n\to f}(x_n) = \prod_{\xi\in\eta_n\setminus\{f\}} \mu_{\xi\to n}(x_n) \tag{3.39}$$

$$\mu_{f\to n}(x_n) = \sum_{\sim\{x_n\}} \left(\psi_f(x_{\eta_f}) \prod_{\xi\in\eta_f\setminus\{n\}} \mu_{\xi\to f}(x_\xi) \right) \tag{3.40}$$

where $\sim \{x_n\}$ denotes the summation over all arguments of f except x_n. For factorization involving continuous variables, the summation is replaced with an integral taken over a Lebesgue

measure. The algorithm is known as sum-product because the steps involved are summations (or integrals) and products of factors and messages.

SPA extended to loopy graphs is called LSPA. LSPA simply requires repeated application of SPA until convergence occurs. Practically, this means computing messages continuously using Equations (3.39) and (3.40) until the maximum error between subsequent messages is less than a pre-set threshold. However, LSPA is neither guaranteed to converge to the right answer nor to converge at all.

Factor Graphs for Data Association

We consider the DA problem involving known and fixed N_T targets and their combined validated measurements $Z_{N_T}(k)$ at time k obtained as explained in JPDAF Section 3.3.3. The number of combined validated measurements are $m_{N_T}(k)$ at time k. Formulating a factor graph for the DA problem that is guaranteed to converge according to [60, 69, 70], we need the following two sets of association variables:

1. Target-oriented association variable (a): Create an association variable $a_i(k) \in \{0, 1, \ldots, m_{N_T}(k)\}$ for each target $i \in \{1, \ldots, N_T\}$. The value assigned to $a_i(k)$ is an index to the measurement with which target i is hypothesized to be associated at time k (zero if the target is hypothesized to not have been detected). The complete set of target-oriented association variables at time k is denoted by $a(k)$.

2. Measurement-oriented association variable (b): Create an association variable $b_j(k) \in \{0, 1, \ldots, N_T\}$ for each measurement $j \in \{1, \ldots, m_{N_T}(k)\}$. The value assigned to $b_j(k)$ is an index to the target with which measurement j is hypothesized to be associated at time k (zero if the measurement is hypothesized to be clutter). The complete set of measurement-oriented association variables at time k is denoted by $b(k)$.

Given one set of association variables, the other set can be perfectly reconstructed. As shown in [60, 69, 70], this use of seemingly redundant information while forming a factor graph leads to

the remarkable result of guaranteed convergence of LSPA computed on the said factor graph. A bipartite graphical model formed using the association variables $a(k)$ an $b(k)$ is shown in Figure 3.1.

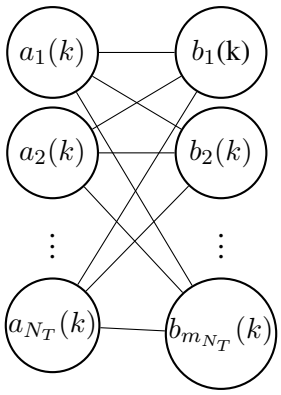

Figure 3.1: Bipartite graphical model formulation for data association at time k. The value assigned to $a_i(k)$ is an index to the measurement with which target i is hypothesized to be associated at time k and the value assigned to $b_j(k)$ is an index to the target with which measurement j is hypothesized to be associated at time k.

Our goal is to estimate state vectors $x_i(k)$, $i \in \{1, \ldots, N_T\}$ from the measurement vectors $Z(k)$ at time k. Introducing the data-association variables $a(k)$ and $b(k)$, a Bayesian approach to state estimation of $x_i(k)$ produces a posterior probability density function (PDF) $f(x_i(k), a(k), b(k)|Z(k))$. This can be achieved by marginalizing the joint posterior PDF $f(X(k), a(k), b(k)|Z(k))$, where $X(k) = (x_i(k), \ldots, x_{N_T}(k))$. However, direct marginalization of $f(X(k), a(k), b(k)|Z(k))$ is not always feasible. Assuming that the posterior PDF of $X(k)$ factorizes into a product of terms associated with each target $x_i(k)$, an efficient marginalization is given as

$$f(X(k), a(k), b(k)) \propto \prod_{i=1}^{N_T} \psi_i(x_i(k), a_i(k)) \prod_{j=1}^{m_{N_T}(k)} \psi_{i,j}(a_i(k), b_j(k)) \tag{3.41}$$

where

$$\psi_i(x_i(k), a_i(k)) = \begin{cases} [1 - P_D(x_i(k)) f(x_i(k)|Z(k-1))] & \text{if } a_i(k) = 0 \\ [P_D(x_i(k)) f(z_{a_i(k)}|x_i(k)) f(x_i(k)|Z(k-1))]/\lambda(z_{a_i(k)}) & \text{if } a_i(k) \neq 0 \end{cases} \tag{3.42}$$

and

$$\psi_{i,j}(a_i(k), b_j(k)) = \begin{cases} 0 & \text{if } a_i(k) = j,\ b_j(k) \neq i \text{ or } b_j(k) = i,\ a_i(k) \neq j \\ 1 & \text{otherwise.} \end{cases} \tag{3.43}$$

Here, $\propto$ indicates equality up to a normalization factor, $\psi_i(x_i(k), a_i(k))$ indicates the dependence of the factors $\psi_i(\cdot)$ on $Z(k)$, $\psi_{i,j}(a_i(k), b_j(k))$ enforces consistency of the redundant association variables $a_i(k)$ and $b_j(k)$ describing the same association configuration, $P_D(x_i)$ gives the probability of detection for target x_i, and $\lambda(z_j)$ gives the PDF value of measurement z_j occurring as a result of false alarm with a Poisson point process.

The factorization structure in Equation (3.41) can be represented by a factor graph. As an example, for $X(k) = (x_1(k), x_2(k))$ and $Z(k) = (z_1(k), z_2(k))$, the factor-graph representation of Equation (3.41) is shown in Figure 3.2. Since the target states $x_i(k)$ in Figure 3.2 are leaf nodes, we can marginalize these and replace the factor $\psi_i(x_i(k), a_i(k))$ with $\psi_i(a_i(k))$. In a factor graph, each parameter variable is represented by a variable node (circle node), and each factor is represented by a rectangle node, as shown in Figure 3.2. Each variable node and each factor node are connected by an edge if the variable is an argument of the factor. For each node, certain messages are calculated according to Equations (3.39) and (3.40). Message passing is started at variable nodes with only one edge (which pass a constant message) and/or factor nodes with only one edge (which pass the corresponding factor). Finally, for each variable node, a belief (posterior PDF value) is calculated as the product of all incoming messages (passed from all adjacent factor nodes) followed by a normalization. For a tree-structured factor graph, these beliefs are exactly equal to marginal posterior PDF values. For a loopy factor graph, the beliefs are in general only approximations of the respective marginal posterior PDF values. A detailed description of the messages passed between each variable node and factor node is given in [74]. A sample MATLAB program for implementing one iteration of loopy SPA is given in [69].

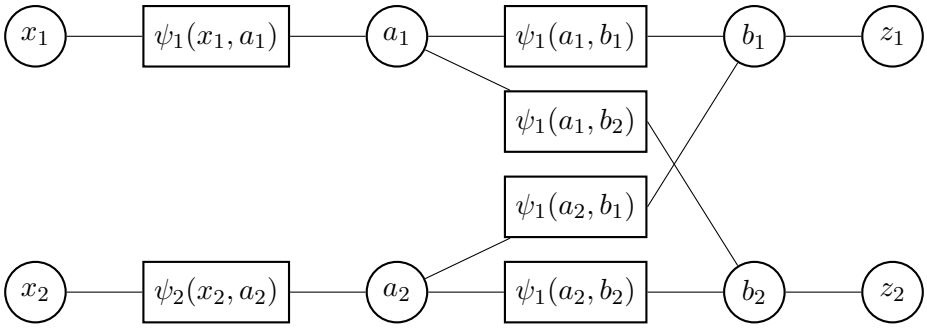

Figure 3.2: Factor graph representing the factorization of the joint posterior probability density function (PDF) $f(x_1, x_2, a_1, a_2, b_1, b_2|z_1, z_2)$ according to Equation (41), depicted for one time-step. For simplicity, the time index k is omitted.

Joint Data Association Probabilities

Once the combined validated measurements are obtained as given in Equation (3.24), the joint data association probabilities under the stated assumptions at time k are calculated as

$$p(a(k), b(k)|Z^k) = \prod_{i=1}^{N_T} \psi_i(a_i(k)) \prod_{j=1}^{m_{N_T}(k)} \psi_{i,j}(a_i(k), b_j(k)) \tag{3.44}$$

where

$$\psi_{i,j}(a_i(k), b_j(k)) = \begin{cases} 0 & \text{if } a_i(k) = j,\ b_j(k) \neq i \text{ or } b_j(k) = i,\ a_i(k) \neq j \\ 1 & \text{otherwise} \end{cases} \tag{3.45}$$

enforce consistency of the redundant association variables $a_i(k)$ and $b_j(k)$ describing the same association configuration, and $\psi_i(a_i(k) = 0) = 1$ and $\psi_i(a_i(k) = j > 0)$ are the (unnormalized) single target data association probabilities obtained as explained in PDAF Section 3.3.1. Equation (3.44) can be expressed as the formulation of SPA for a bipartite model illustrated in Figure 3.1 in which all target association variables $a_i(k)$ are connected to all measurement association variables $b_j(k)$. In this case, LSPA may be implemented via two half iterations, alternating between the two sets of messages $\mu_{a_i(k)\to b_j(k)}(b_j(j))$ and $\mu_{b_j(j)\to a_i(k)}a_i(k)$. The message updating

equations according to Equations (3.39) and (3.40) can be given as

$$\mu_{a_i(k)\to b_j(k)}(b_j(k)) = \sum_{a_i(k)} \psi_i(a_i(k))\psi_{i,j}(a_i(k), b_j(k)) \prod_{j'\neq j} \mu_{b_{j'(k)}\to a_i(k)}(a_i(k)) \tag{3.46}$$

$$= \begin{cases} \psi_i(j) \prod_{j'\neq j} \mu_{b_{j'(k)}\to a_i(k)}(j) & \text{if } b_j(k) = i \\ \sum_{a_i(k)\neq j} \psi_i(a_i(k)) \prod_{j'\neq j} \mu_{b_{j'(k)}\to a_i(k)}(a_i(k)) & \text{if } b_j(k) \neq i \end{cases} \tag{3.47}$$

and

$$\mu_{b_j(k)\to a_i(k)}(a_i(k)) = \sum_{b_j(k)} \psi_{i,j}(a_i(k), b_j(k)) \prod_{i'\neq i} \mu_{a_{i'}(k)\to b_j(k)}(b_j(k)) \tag{3.48}$$

$$= \begin{cases} \prod_{i'\neq i} \mu_{a_{i'}\to j}(i) & \text{if } a_i(k) = j \\ \sum_{b_j(k)\neq i} \prod_{i'\neq i} \mu_{a_{i'}\to j}(b_j(k)) & \text{if } a_i(k) \neq j. \end{cases} \tag{3.49}$$

These messages can be further simplified as shown in [69, 70]. Upon convergence of LSPA, the approximate marginal association probabilities can be given as

$$p(a_i(k) = j|Z^k) = \frac{\psi_i(j)\mu_{b_j(k)\to a_i(k)}(a_i(k))}{\sum_{j'=0}^{m_{N_T}(k)} \psi_i(j')\mu_{b_{j'}(k)\to a_i(k)}(a_i(k))} \tag{3.50}$$

$$p(b_j(k) = i|Z^k) = \frac{\mu_{a_i(k)\to b_j(k)}(b_j(k))}{\sum_{i'=0}^{N_T} \mu_{a_{i'}(k)\to b_j(k)}(b_j(k))} \tag{3.51}$$

where $\mu_{b_j(k)=0\to a_i(k)}(a_i(k)) \triangleq 1$ and $\mu_{a_i(k)=0\to b_j(k)}(b_j(k)) \triangleq 1$. The marginal association probabilities are used to estimate target states as per PDAF Section 3.3.1.

3.3.5 Distance-Weighting Loopy Sum-Product Algorithm

Since the factor-graph representation for the MTT DA problem contains loops, and the convergence process of calculating the marginal data-association probabilities using LSPA is governed by some heuristically determined preset threshold, different initiation messages can lead to different final beliefs. These initiation messages for the convergence process happen to be the (un-

normalized) single-target data-association probabilities. These probabilities directly influence the marginal data-association probabilities at the end of the convergence process and, consequently, the estimation of the target state. We would expect that a more accurate initial set of single-target probabilities leads to either more accurate final beliefs, a faster convergence to the final beliefs, or both. The distance-weight-based association probabilities have been proven to be more accurate for tracking a single target in densely cluttered environments [59]. We would expect this adjustment to the initial condition to change the calculation of the association probabilities and hence the overall tracking process in terms of tracking accuracy or computation time, and therefore worth exploring.

Here we formulate a modification for the joint data association probabilities calculated using LSPA under the stated assumptions as described in Section 3.3.4. The distance-weight based joint data association probabilities at time k can be given as

$$p(a(k), b(k)|Z^k) = \prod_{i=1}^{N_T} \psi_i(a_i(k)) \prod_{j=1}^{m_{N_T}(k)} \psi_{i,j}(a_i(k), b_j(k)) \tag{3.52}$$

where

$$\psi_{i,j}(a_i(k), b_j(k)) = \begin{cases} 0 & a_i(k) = j,\ b_j(k) \neq i \text{ or } b_j(k) = i,\ a_i(k) \neq j \\ 1 & \text{otherwise} \end{cases} \tag{3.53}$$

enforce consistency of the redundant association variables $a_i(k)$ and $b_j(k)$ describing the same association configuration, and $\psi_i(a_i(k) = 0) = 1$ and $\psi_i(a_i(k) = j > 0) = \mathcal{B}_i^t(k)$ are the (unnormalized) distance-weight based single target data association probabilities obtained in Equation (3.22). The approximated marginal association probabilities are obtained from the joint DA probabilities as described in Section 3.3.4.

3.4 Simulation and Analysis

3.4.1 The Dynamic Model

We assume a two-dimensional system where the target state vector consists of position and velocity in each of the two coordinates. For target i at time k, the target state $x_i(k) = [x(k), \dot{x}(k), y(k), \dot{y}(k)]$ has four components: The first and second components are the horizontal location and velocity, respectively, while the third and fourth components are the vertical location and velocity, respectively. The system is equipped with the nearly constant velocity (NCV) model (also sometimes called the constant velocity (CV) model). The system is described by Equations (3.1) and (3.2) with

$$F(k) = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.54}$$

$$G(k) = \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix} \tag{3.55}$$

$$H(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.56}$$

$$Q(k) = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix} \tag{3.57}$$

$$R(k) = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \tag{3.58}$$

where T is the sampling interval.

3.4.2 Simulation Parameters

For our simulations, we set the probability of detection $P_D = 0.9$, the gate threshold $\gamma = 9.21$ corresponding to gate probability $P_G = 0.99$, sampling interval $T = 1$ s, the process noise variance $q = 0.05$, and the measurement noise variance $r = 5$. To better evaluate the accuracy of different algorithms under multiple conditions, we varied the clutter density generated according a Poisson point process from $\lambda = 1.0 \times 10^{-4}/\text{scan}/\text{m}^2$ to $\lambda = 5.0 \times 10^{-4}/\text{scan}/\text{m}^2$. We also varied the number of tracked targets from 1 to 6. The initial state of the first target is always set at $x_1(1) = [100 \text{ m}, 30 \text{ m/s}, 100 \text{ m}, 30 \text{ m/s}]$. We set the initial state of each consecutive target by $x_i(1) = [100 \text{ m}, 30 \text{ m/s}, (100 - i \times 100 \times c(i)) \text{ m}, (30 - i \times 30 \times c(i)) \text{ m/s}]$ where i is the target index and $c(i)$ is a single uniformly distributed random number in the interval $(0, 1)$ independent of each other. To compare the performance, we performed 500 Monte Carlo simulations on MATLAB (Natick, MA, USA) [76].

3.4.3 Results and Discussion

When choosing an optimal tracking algorithm, there is typically a tradeoff between tracking accuracy and computation time. Maintaining a high-level accuracy in complex scenarios where multiple targets need to be tracked simultaneously and the environment is particularly noisy requires significant computation time. Traditionally, the tracking accuracy of an estimator is calculated in terms of a miss-distance, or localization error, between a reference value and its estimated value. In our context, we are also interested in evaluating missed detections and false alarms. The generalized optimal sub-pattern assignment (GOSPA) metric has been designed to reflect this performance [77]. Informally, the GOSPA metric can be defined as

$$\text{GOSPA} = \sum \text{localization error} + \frac{\text{cutoff distance}}{2}(\#\text{ of missed detections} + \#\text{ of false alarms})$$

(for a precise detailed description of GOSPA, see [77]). The localization error is for pairs of true targets and target estimates that are sufficiently close. A missed detection is declared if there is no corresponding target sufficiently close to it, and a false alarm is declared if there is no corre-

sponding true target sufficiently close to it. To evaluate the performance of the different tracking algorithms described in Section 3.3 in terms of the miss-distance, the GOSPA metric, and computation time, we considered multiple scenarios with varying levels of complexities.

Figures 3.3 and 3.4 depict two such scenarios where the trajectories of three targets are shown for clutter densities $\lambda = 1 \times 10^{-4}/\text{m}^2$ and $\lambda = 5 \times 10^{-4}/\text{m}^2$, respectively. The scenario depicted in Figure 3.4 is more demanding because of the increased number of false alarms. Figures 3.5 and 3.6 compare the performance of LSPA and DWPDA in terms of tracking accuracy, calculated using the root mean square (RMS) position error, over a period of 100-time intervals for the above two scenarios, respectively. We see that the RMS position errors for LSPA stay close to 0 over the entire period while they are always increasing for DWPDA as time progresses. We see similar results while tracking six crossing targets with clutter densities $\lambda = 1 \times 10^{-4}/\text{m}^2$ and $\lambda = 5 \times 10^{-4}/\text{m}^2$, as depicted in Figures 3.7 and 3.8. Figures 3.9 and 3.10 show that the computation times for the two clutter scenarios as the number of targets increases from one to six. We see that the computation times required for LSPA are only slightly higher than those of DWPDA. Finally, Tables 3.1 and 3.2 summarize the results for the performance in terms of the GOSPA metric based on the Euclidean distance with a cutoff parameter of 30. Table 3.1 shows the average GOSPA errors as we increase the number of targets from 1 to 6 while keeping the clutter density constant at $\lambda = 3 \times 10^{-4}/\text{m}^2$. We can see that irrespective of the number of targets that are being tracked, GOSPA errors for LSPA are only a fraction of the errors for DWPDA. We see a similar pattern in Table 3.2 where we increase the clutter density from $\lambda = 1 \times 10^{-4}/\text{m}^2$ to $\lambda = 5 \times 10^{-4}/\text{m}^2$ while keeping the number of targets fixed. The poor performance of DWPDA is expected, since the algorithm is ill-equipped to deal with the problem of DA in MTT, and GOSPA appropriately penalizes any missed detections and false alarms. From the above results, it is evident that LSPA is superior to DWPDA in terms of tracking accuracy in all scenarios without trading off much computation time. This superior tracking accuracy can be attributed to the reduction in the association probabilities of false measurements in the overlapping validation regions from multiple targets and, at the same time, the increase in the association probabilities for actual target measurements. From the above

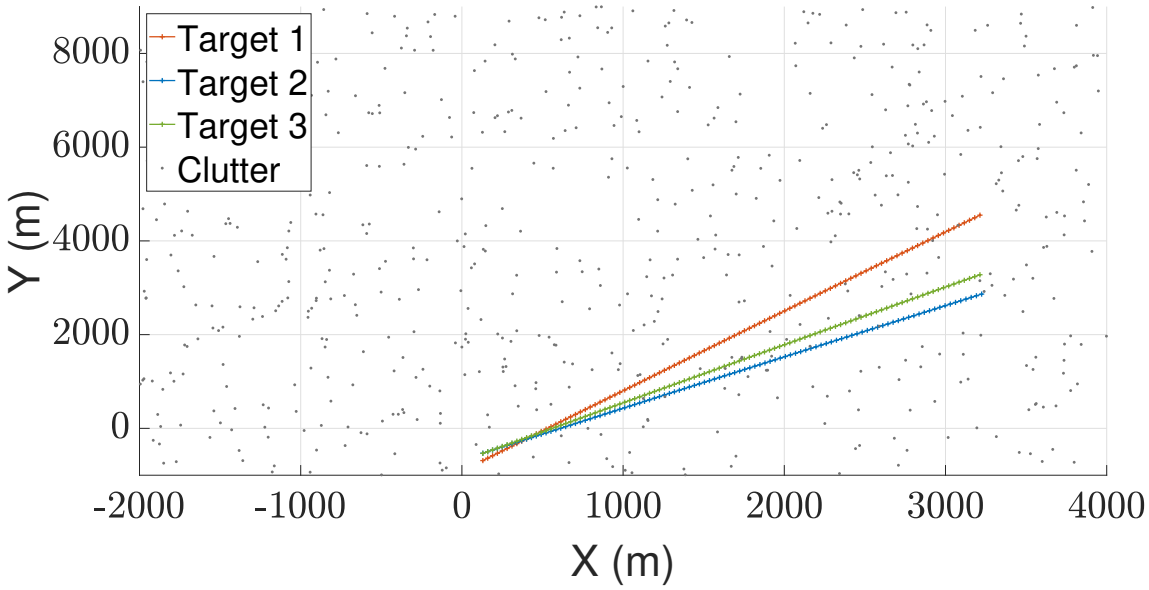

Figure 3.3: True target positions for three crossing targets with clutter density $\lambda = 1 \times 10^{-4}/\mathrm{m}^2$.

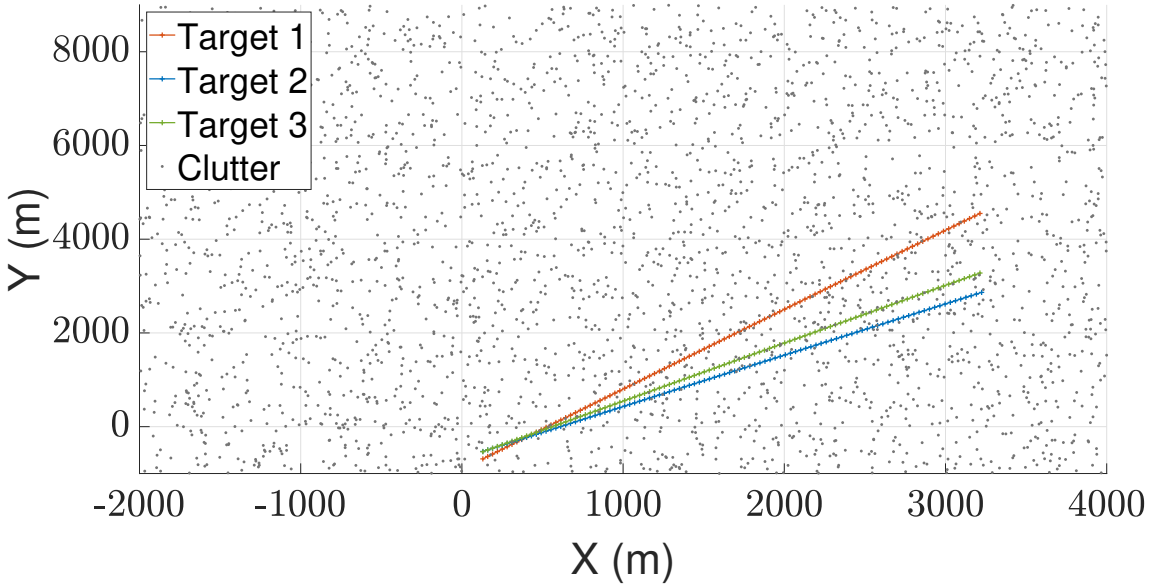

Figure 3.4: True target positions for three crossing targets with clutter density $\lambda = 5 \times 10^{-4}/\mathrm{m}^2$.

results, we know that LSPA can improve the tracking performance for MTT in densely cluttered environments.

Table 3.1: Average generalized optimal sub-pattern assignment (GOSPA) errors for tracking multiple crossing targets with clutter density $\lambda = 3 \times 10^{-4}/\mathrm{m}^2$.

	Number of Targets					
	1	**2**	**3**	**4**	**5**	**6**
LSPA	0.4571	0.9103	1.5033	1.9659	2.3441	2.9000
DWLSPA	0.4435	0.8787	1.5243	1.8601	2.2664	2.8253
JPDA	0.6751	2.5626	4.6425	6.1505	9.3851	11.5240
DWPDA	4.7454	9.1838	14.2879	18.5087	23.3401	27.4684

Table 3.2: Average GOSPA errors for tracking three targets with different clutter densities.

	$\boldsymbol{\lambda}$				
	$\mathbf{1 \times 10^{-4}}$	$\mathbf{2 \times 10^{-4}}$	$\mathbf{3 \times 10^{-4}}$	$\mathbf{4 \times 10^{-4}}$	$\mathbf{5 \times 10^{-4}}$
LSPA	1.4280	1.6058	1.4056	1.6628	1.7690
DWLSPA	1.3718	1.5652	1.3388	1.5289	1.6257
JPDA	3.5367	4.2030	4.3475	4.4828	5.2677
DWPDA	7.4128	11.8464	14.4591	15.0590	15.8318

Next, we compare the performance of LSPA with JPDA, in terms of computation time and RMS position error, as the number of targets increases from one to six. Figures 3.11, 3.13, 3.12, and 3.14 show the results of this comparison for the two clutter scenarios. We can clearly see that the average computation time of LSPA is much smaller than that of JPDA, so much so that the LSPA computation times are not even visible in Figures 3.11 and 3.12. However, in contrast to the comparison of LSPA and DWPDA in terms of the RMS position error, in the case of LSPA compared with JPDA, Figures 3.13 and 3.14 clearly show that there is little difference in RMS

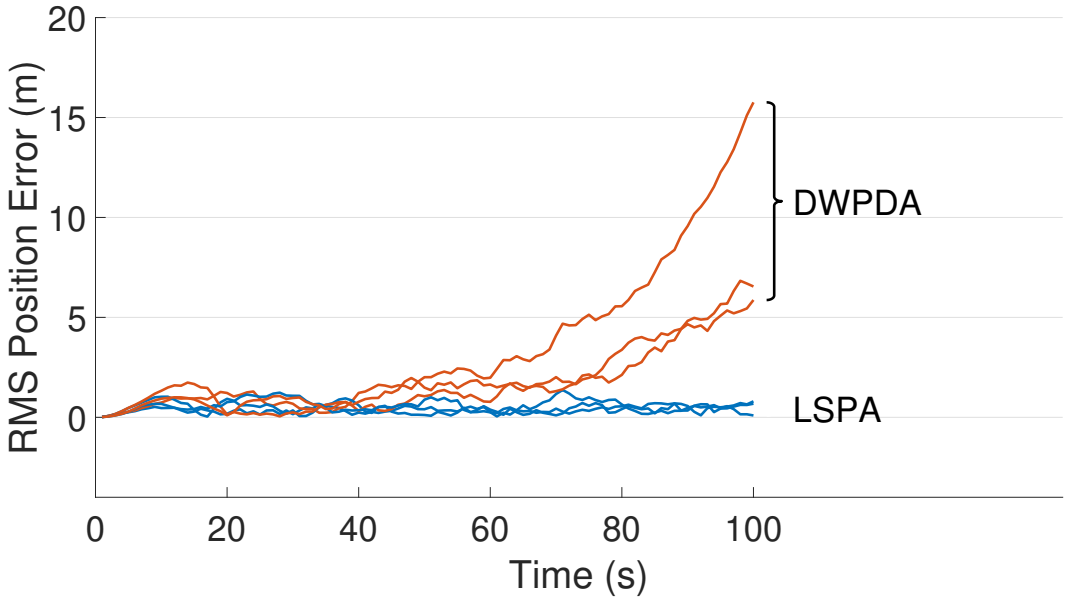

Figure 3.5: RMS position error for three crossing targets using DWPDA and LSPA with clutter density $\lambda = 1 \times 10^{-4}/\text{m}^2$.

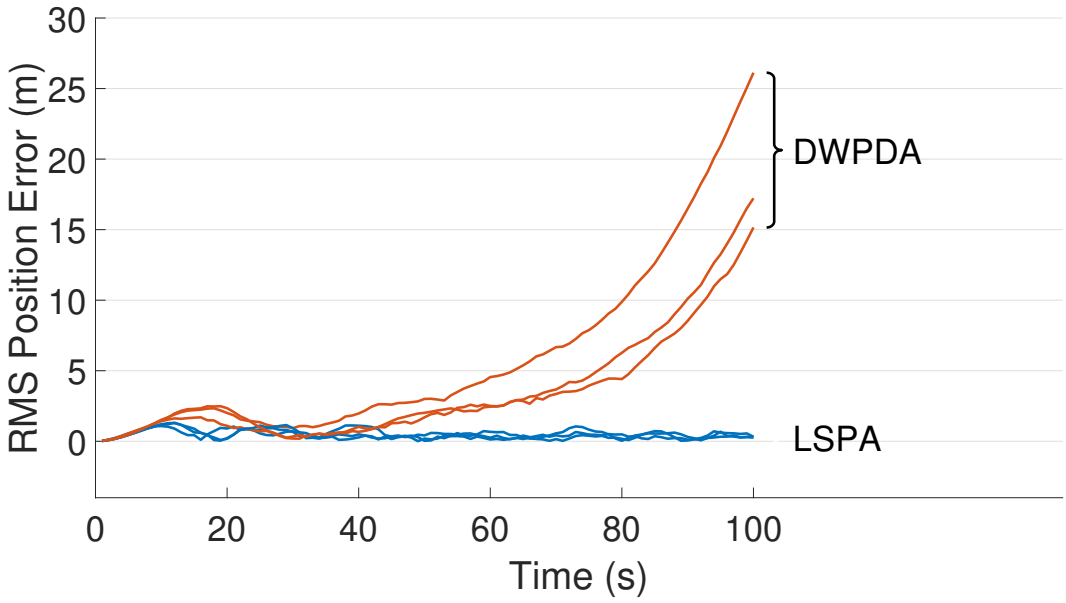

Figure 3.6: RMS position error for three crossing targets using DWPDA and LSPA with clutter density $\lambda = 5 \times 10^{-4}/\text{m}^2$.

position error. The same observation applies in terms of computation time and RMS position error in Figures 3.15 and 3.15 respectively when we compare LSPA with JPDA in scenarios involving a fixed number of targets as the clutter density increases from $\lambda = 1 \times 10^{-4}/\text{m}^2$ to $\lambda = 5 \times$

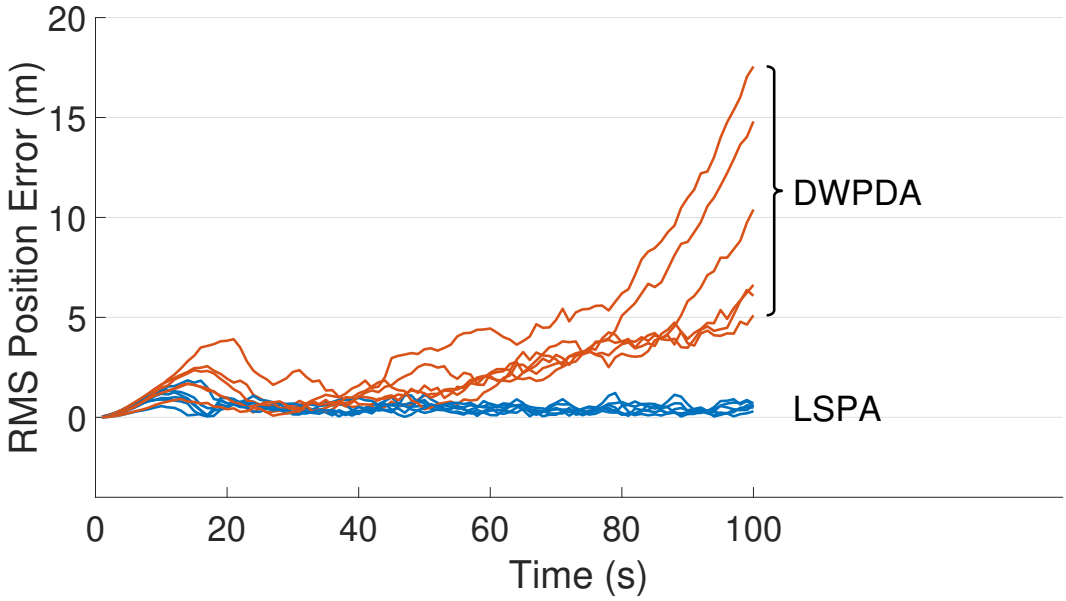

Figure 3.7: RMS position error for six crossing targets using DWPDA and LSPA with clutter density $\lambda = 1 \times 10^{-4}/\text{m}^2$.

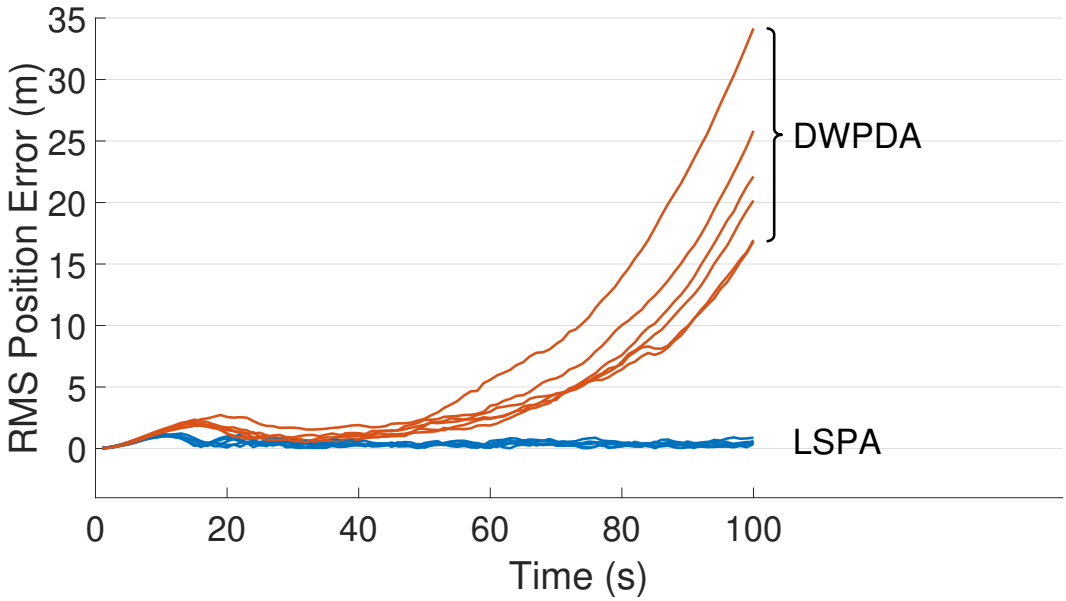

Figure 3.8: RMS position error for six crossing targets using DWPDA and LSPA with clutter density $\lambda = 5 \times 10^{-4}/\text{m}^2$.

$10^{-4}/\text{m}^2$. While there is little difference between LSPA and JPDA in terms of RMS position errors, Tables 3.1 and 3.2 show that GOSPA errors for LSPA are less than 1/3rd of GOSPA errors for JPDA across almost all scenarios. These relatively higher GOSPA errors for JPDA can be

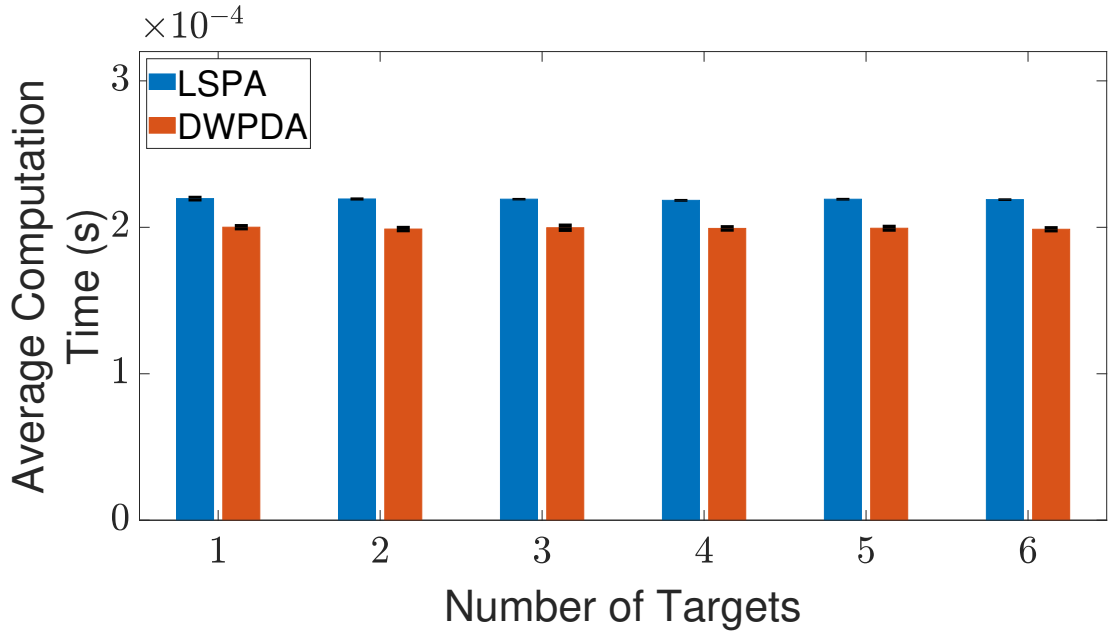

Figure 3.9: Average computation time for obtaining association probabilities using DWPDA and LSPA for tracking multiple crossing targets with clutter density $\lambda = 1 \times 10^{-4}/\text{m}^2$.

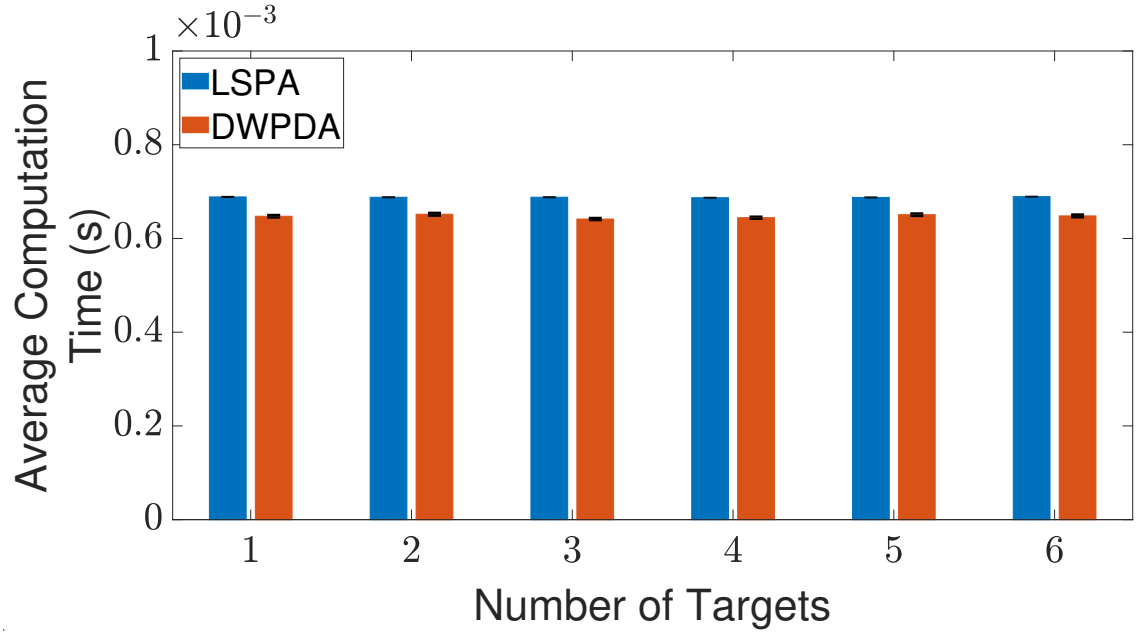

Figure 3.10: Average computation time for obtaining association probabilities using DWPDA and LSPA for tracking multiple crossing targets with clutter density $\lambda = 5 \times 10^{-4}/\text{m}^2$.

explained by a few missed detections when multiple target paths overlap. These missed detections are rightly penalized in the GOSPA metric. These results show that LSPA dominates JPDA in terms of computation time while maintaining a high level of tracking accuracy. This dramatic reduction

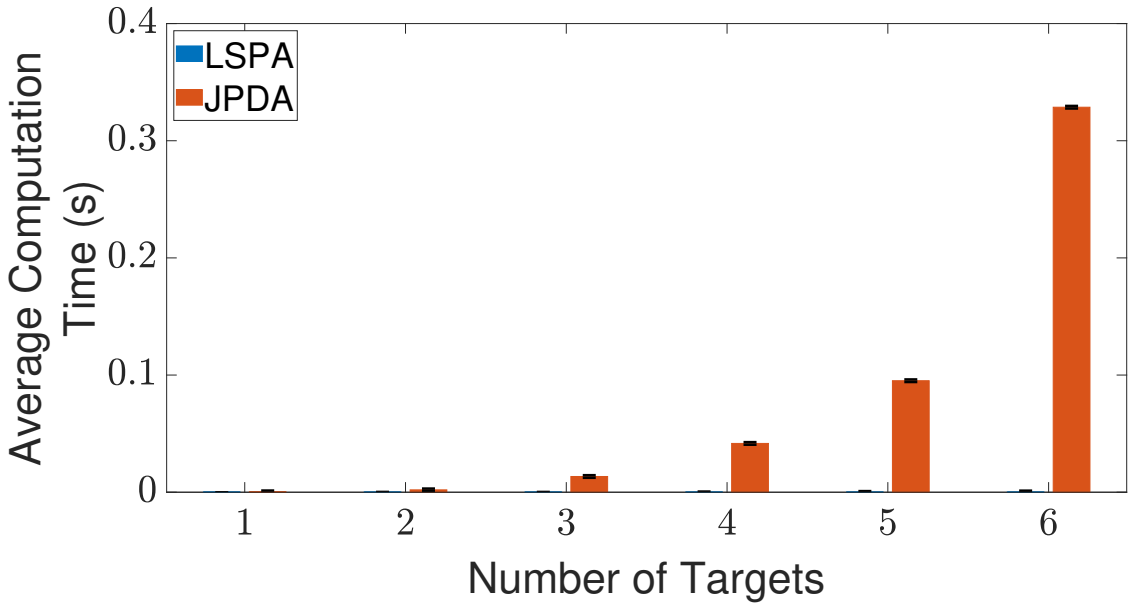

Figure 3.11: Average computation time for obtaining association probabilities while tracking multiple crossing targets using LSPA and JPDA with clutter density $\lambda = 1 \times 10^{-4}/\text{m}^2$. Error bars indicate 95% confidence intervals.

in computation time for LSPA can be explained by the implementation of the loopy factor graph, resulting in the simultaneous updating of the joint association probabilities for multiple targets during each iteration of LSPA. The results show that LSPA scales well for real-time applications involving complex tracking scenarios.

Finally, to see whether the integration of LSPA with the distance-weighting scheme has any effect on its performance, we compare LSPA with DWLSPA in terms of the same two metrics as above. Figures 3.17, 3.18, 3.19, and 3.20 show the results of this comparison for two clutter scenarios respectively, as the number of targets increases from one to six. There is little difference between LSPA and DWLSPA in terms of computation times, as shown in Figures 3.17 and 3.18. Similarly, we can see that the RMS position errors for LSPA and DWLSPA are almost identical in Figures 3.19 and 3.20. Figure 3.21 and 3.22 show that the difference between LSPA and DWLSPA remains negligible, in terms of both computation time and RMS position error, for scenarios with a fixed number of targets and varying clutter densities. Tables 3.1 and 3.2 show that GOSPA errors for LSPA and DWLSPA are comparable across all tested scenarios. This means that in addition

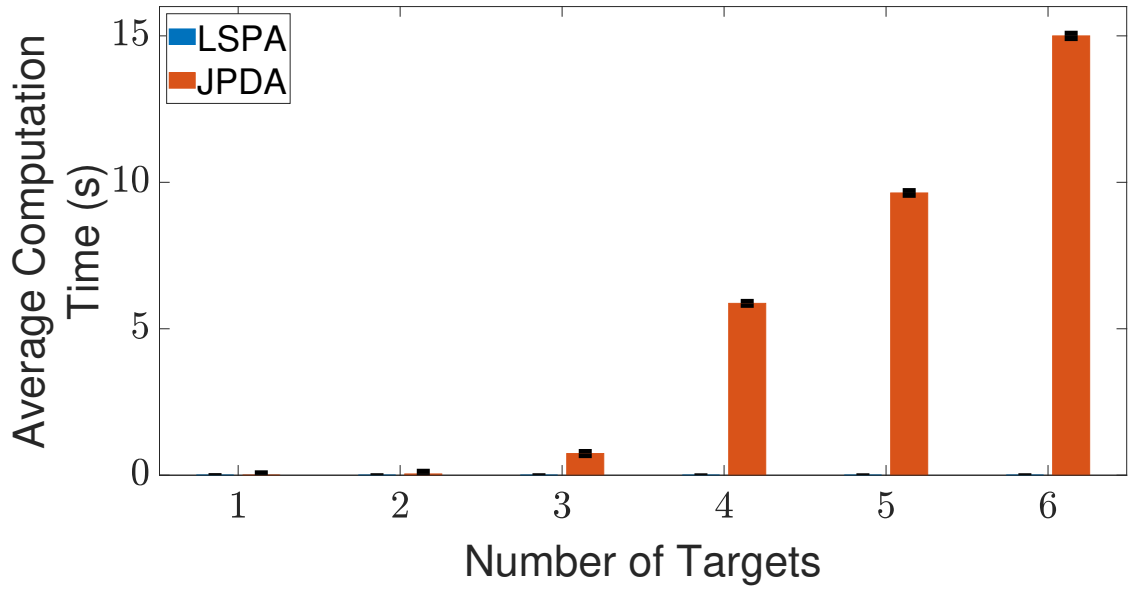

Figure 3.12: Average computation time for obtaining association probabilities while tracking multiple crossing targets using LSPA and JPDA with clutter density $\lambda = 5 \times 10^{-4}/\text{m}^2$. Error bars indicate 95% confidence intervals.

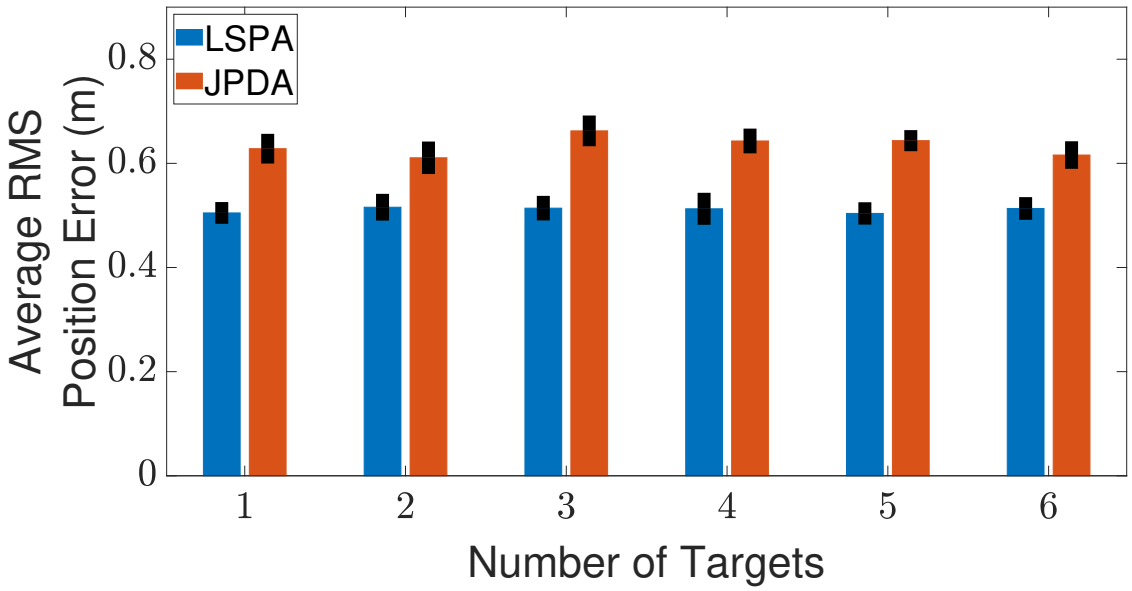

Figure 3.13: Average RMS position error while tracking multiple crossing targets using LSPA and JPDA with clutter density $\lambda = 1 \times 10^{-4}/\text{m}^2$. Error bars indicate 95% confidence intervals.

to the localization errors, the missed detections and false alarms remain consistent between LSPA and DWLSPA, and the potential advantage of DWLSPA with the additional distance-weighting

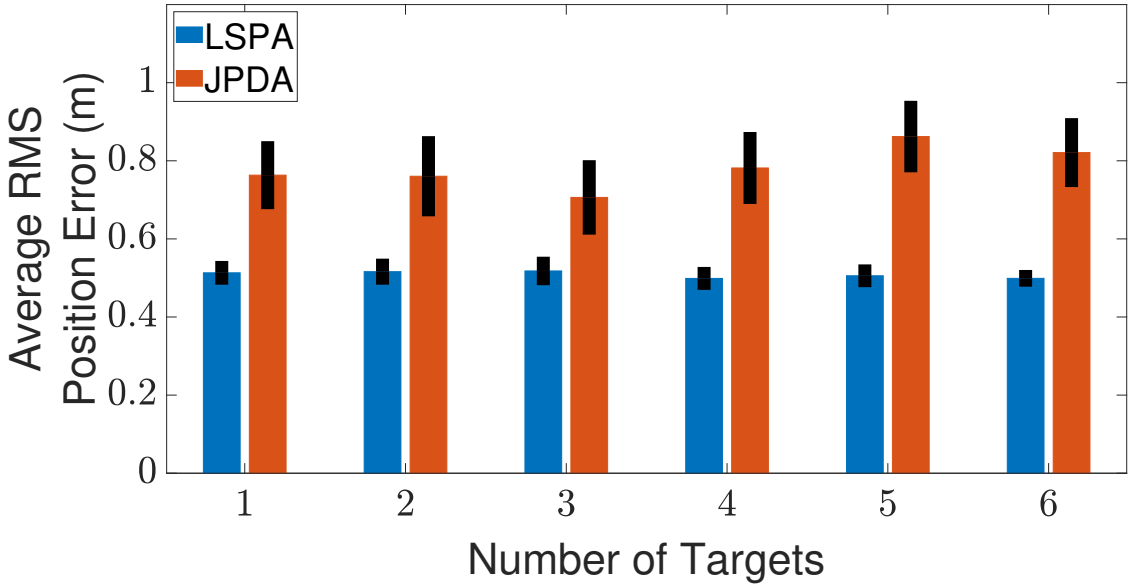

Figure 3.14: Average RMS position error while tracking multiple crossing targets using LSPA and JPDA with clutter density $\lambda = 5 \times 10^{-4}/\text{m}^2$. Error bars indicate 95% confidence intervals.

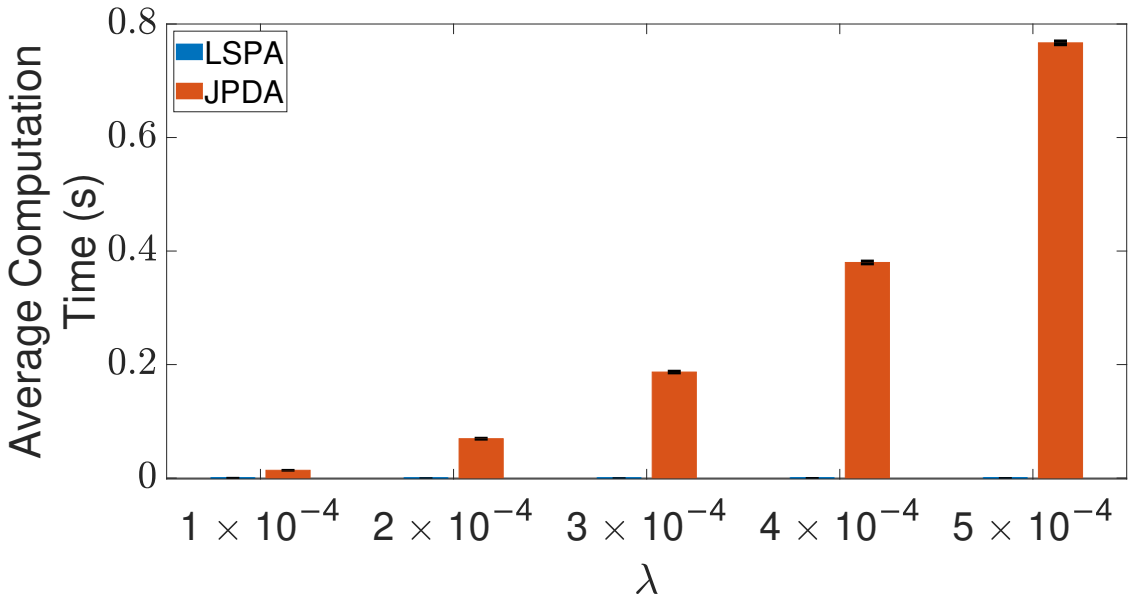

Figure 3.15: Average computation time for obtaining association probabilities while tracking three crossing targets using LSPA and JPDA.

information is not apparent. Surprisingly, LSPA and DWLSPA perform equally well in every scenario in terms of both tracking accuracy and computation time. The unchanged performance of DWLSPA can be explained by the initiation of LSPA with small improvements in single-target

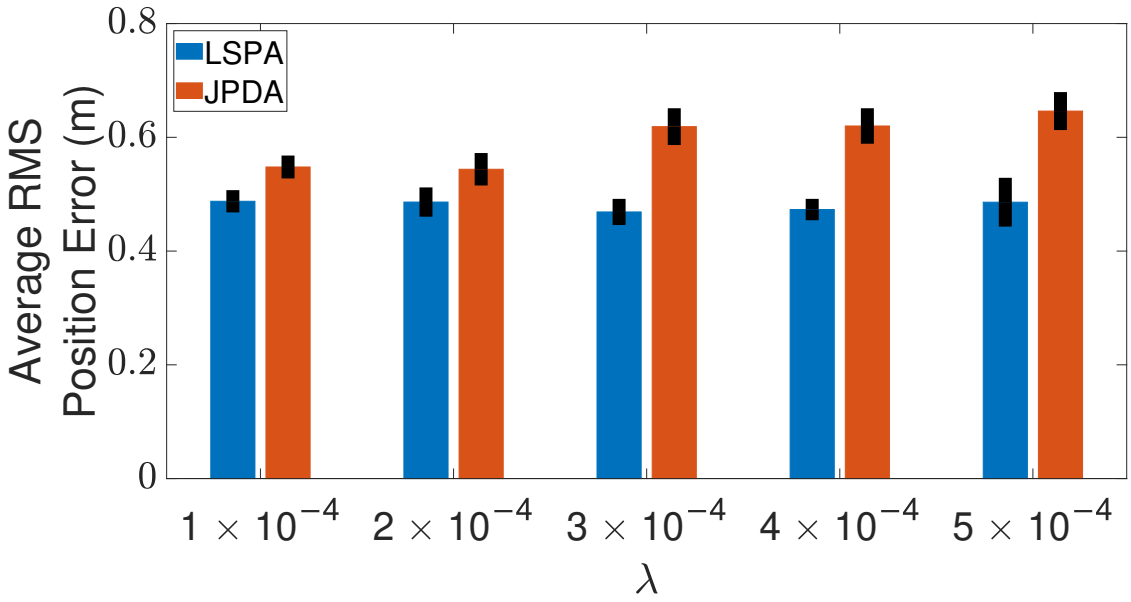

Figure 3.16: Average RMS position error while tracking three crossing targets using LSPA and JPDA.

association probabilities having an insignificant effect on joint association probabilities calculated at the end of a large number of iterations. However, this unexpected lack of improvement contrasts sharply with results reported in [59] showing significant improvement when introducing distance weighting relative to PDA. This result is interesting and useful because we can see that distance weighting does not always lead to better performance. Moreover, the following important observation remains: LSPA is reliable and efficient for tracking multiple objects in cluttered environments.

3.5 Conclusion

In this chapter, we formulated a distance-weighting PDA approach for LSPA and examined its effect for tracking multiple objects in cluttered environments. It has been previously shown that a modification of PDA according to a weighting scheme based on distances between predicted and true target positions improves the tracking accuracy of PDA. LSPA is known to be better than PDA and JPDA and, since PDA constitutes a crucial building block of LSPA, we expected the integration of DWPDA with LSPA to boost the overall performance even further. We studied

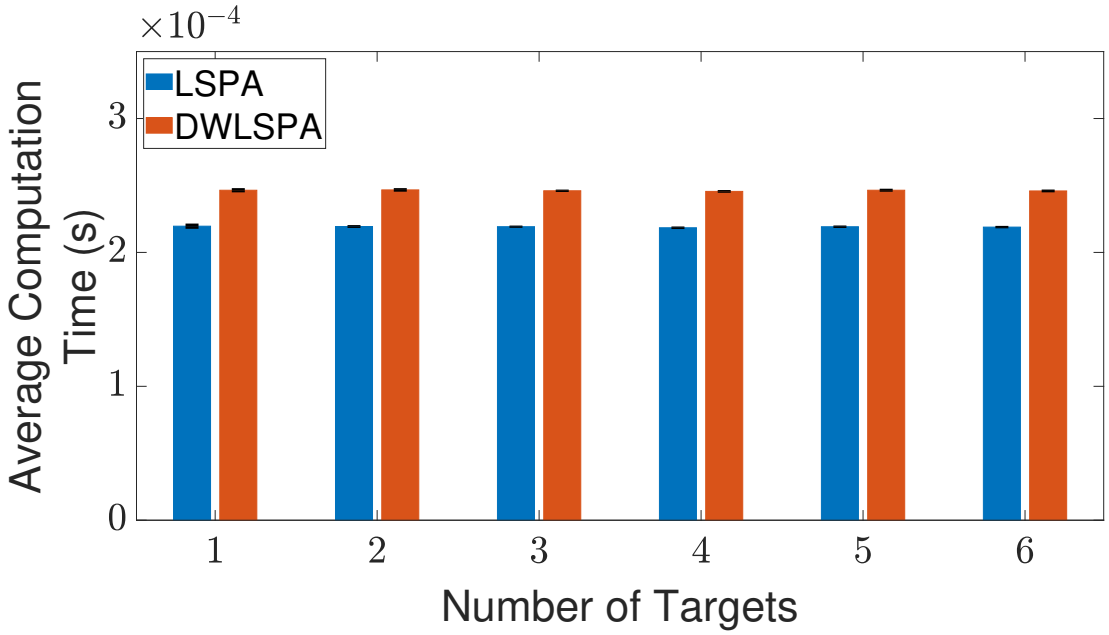

Figure 3.17: Average computation time for obtaining association probabilities while tracking multiple crossing targets using LSPA and DWLSPA with clutter density $\lambda = 1 \times 10^{-4}/\text{m}^2$. Error bars indicate 95% confidence intervals.

the performance of LSPA against DWPDA, JPDA, and DWLSPA for a wide range of tracking scenarios involving multiple targets and varying clutter densities. Our results confirm that LSPA is superior to DWPDA in terms of tracking accuracy and dominates JPDA in terms of computation time. However, contrary to expectations, we found that the distance-weighting approach, when integrated with LSPA, does not enhance the performance of LSPA in terms of either tracking accuracy or computation time. The simulation scenarios in the experiment could be made more realistic with the addition of appearing and disappearing targets and time-varying target velocities. These scenarios add extra layers of complexity to the DA without affecting the conclusions we draw in this chapter. Regardless, we demonstrated the validity of LSPA having computational requirements suitable for real-time processing and accuracy of tracking multiple targets in cluttered environments.

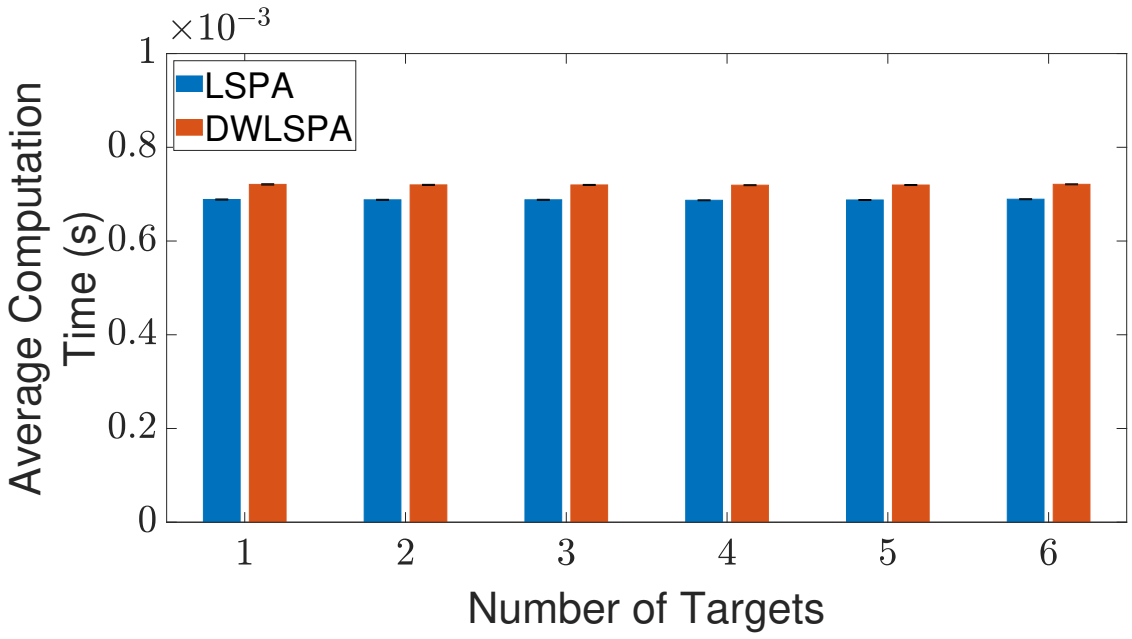

Figure 3.18: Average computation time for obtaining association probabilities while tracking multiple crossing targets using LSPA and DWLSPA with clutter density $\lambda = 5 \times 10^{-4}/\text{m}^2$. Error bars indicate 95% confidence intervals.

3.6 Funding

This work was partially supported by Sandia National Laboratories under Purchase Order 1980525. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly-owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This chapter describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the chapter do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

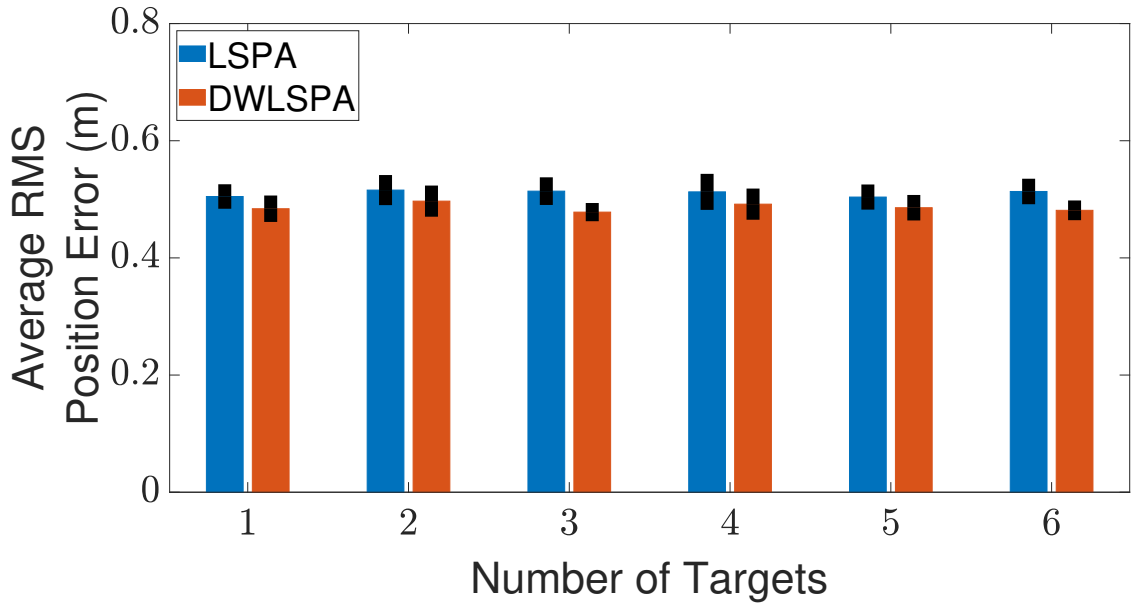

Figure 3.19: Average RMS position error while tracking multiple crossing targets using LSPA and DWLSPA with clutter density $\lambda = 1 \times 10^{-4}/\text{m}^2$. Error bars indicate 95% confidence intervals.

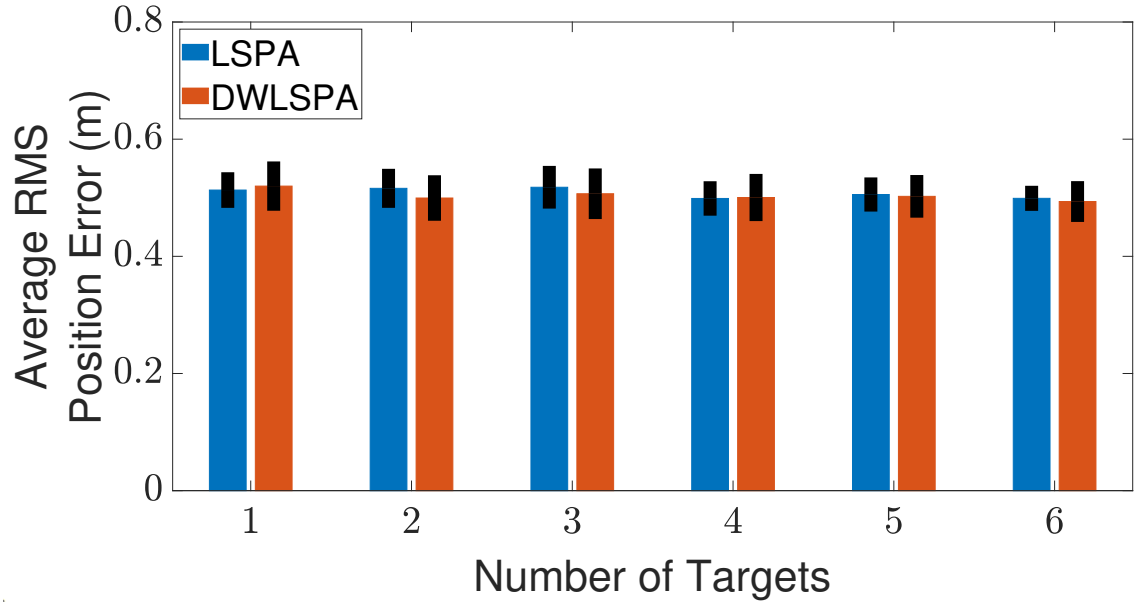

Figure 3.20: Average RMS position error while tracking multiple crossing targets using LSPA and DWLSPA with clutter density $\lambda = 5 \times 10^{-4}/\text{m}^2$. Error bars indicate 95% confidence intervals.

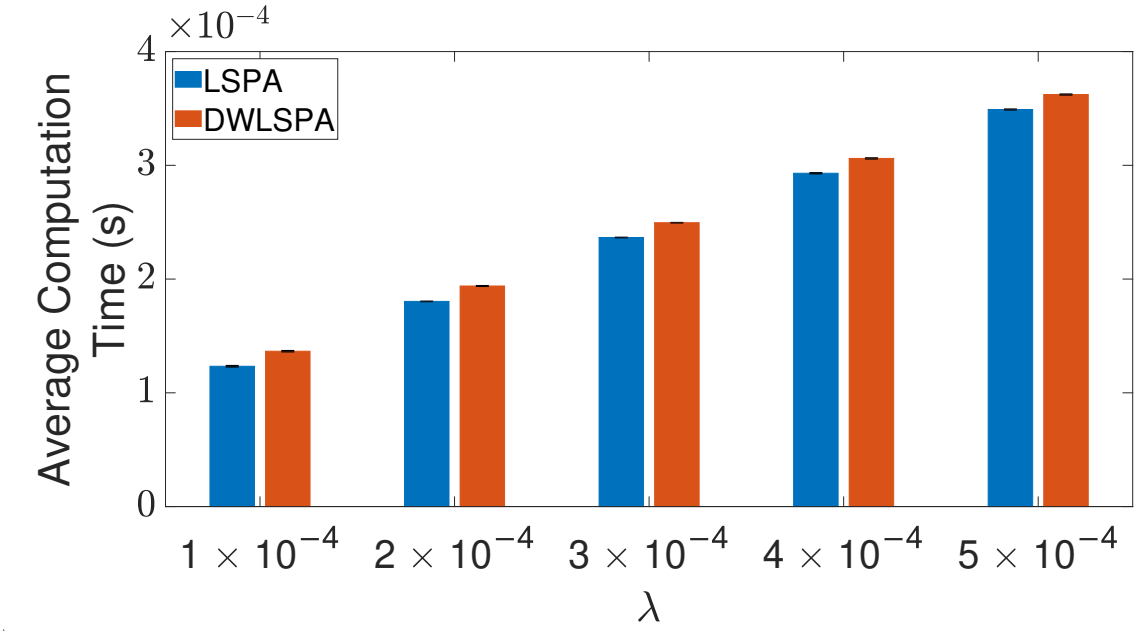

Figure 3.21: Average computation time for obtaining association probabilities while tracking three crossing targets using LSPA and DWLSPA.

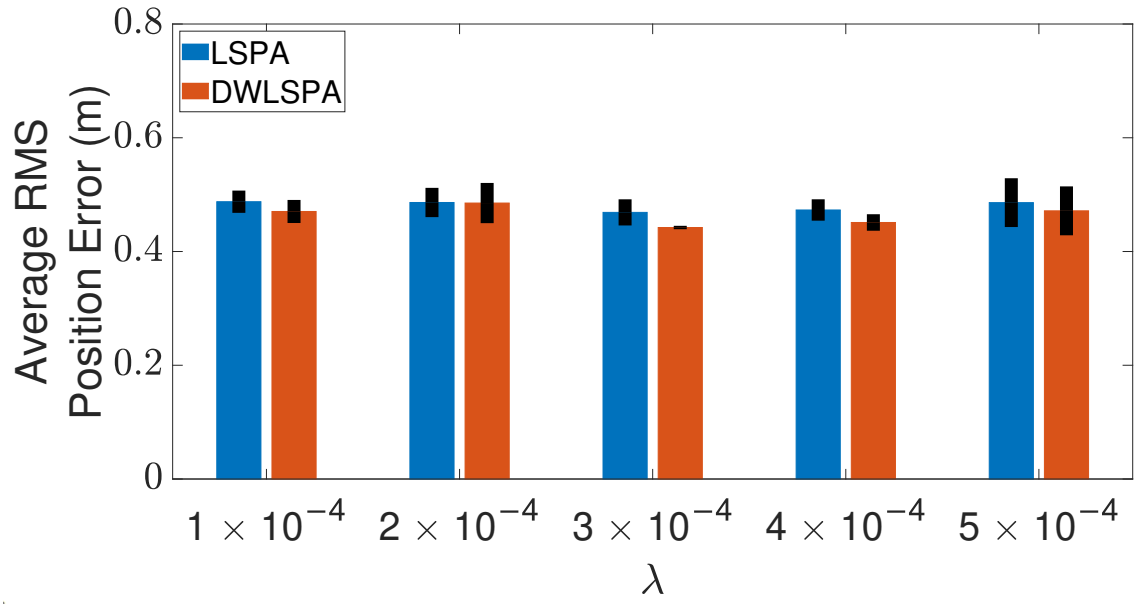

Figure 3.22: Average RMS position error while tracking three crossing targets using LSPA and DWLSPA.

Chapter 4

Wiener Filter Approximation without Covariance Matrix Inversion [3, 4]

4.1 Introduction

Estimating a signal of interest based on an observable related signal is of crucial importance. An optimal *estimator* or *filter* (we use these terms interchangeably) is an algorithm that processes the observable signal to yield an estimate such that a certain error criterion between the estimated and desired signal is minimized. The *Wiener filter* is the optimal linear minimum mean square error (LMMSE) estimator. However, computing the Wiener filter involves inverting the observation covariance matrix. In practice, the dimension of the observation signal might be large, or the number of available observation samples might be small relative to the dimension of the observation signal. This results in the observation covariance matrix having a *large condition number*. Computing Wiener filters with ill-conditioned covariance matrices may be numerically unreliable.

We can circumvent the problem of ill-conditioning by approximating the Wiener filter. One approach is by *reconditioning* the covariance matrix using various well-known techniques: ridge regression [80], the minimum eigenvalue method [81], etc. Another approximation approach is to constrain the rank of the Wiener filter to reduce the computation complexity and enhance the performance [82–89]. However, this does not address the issue of ill-conditioning [90]. More recently, [90] described an alternative approach that explicitly addresses the ill-conditioning instead of constraining the filter rank. The approach described in [90] involves "truncation" based on prin-

[3] A short version of this chapter was published in the Proceedings of the 48th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) [78].

[4] The material in this chapter has been accepted to be published in the IEEE Open Journal of Signal Processing (OJSP) [79].

cipal components of a *composite* covariance matrix, leading to two well-conditioned approximate Wiener filters.

To explore the possibility of a more computation-efficient filter without significantly sacrificing performance, in this chapter, we similarly use truncation to formulate our first two approximate Wiener filters. We approximate our filter formulas even further using the Neumann series expansion and eliminate matrix inversion altogether. The resulting approximate filters are well-conditioned and therefore numerically reliable to compute. The approximations developed here enable a tradeoff between computation and accuracy. We analyze the asymptotic performance of our approximate filters and show that they converge to the Wiener filter as certain approximating terms vanish. To demonstrate the efficacy of our approximation formulas and illustrate the possible computation-accuracy tradeoff, we evaluate the performance of our filters using an empirical dataset: historical daily closing values of the CBOE Amazon VIX Index [91].

The number of principal components of the composite covariance matrix plays a crucial part in truncation and effectively determines the quality of the approximation. To determine the optimal number of principal components used for approximating the filter, [90] uses a line-search procedure to minimize the mean square error. In this chapter, we use the method of [92], based on results from random matrix theory (RMT), to determine the number of principal components of the covariance matrix. We compare the results for the number of principal components computed using the RMT-based method against the method of [90]. It turns out that the number of principal components, and effectively the performance of our filters, computed using both methods are comparable, while the RMT-based method is more efficient in terms of the computation time than the method of [90].

Even though the practical limitations caused by ill-conditioning are not new to signal processing, and similar problems are faced in other fields, such as statistics and machine learning, only a few known solutions are available, and even fewer solutions that directly address the issue at hand. In signal processing, ill-conditioned covariance matrices can produce filters exhibiting significant errors [93]. In statistics, linear least squares (LLS) problems are often ill-conditioned. In [94], the authors study the stability and accuracy of least squares approximations. In [95], the authors

investigate the efficacy of a self-adaptive iterative algorithm for solving severe ill-conditioned LLS problems. In machine learning, ill-conditioning problems occur when dealing with convolutional neural networks [96]. Because the solution to Wiener filters closely relates to the above-mentioned problems, our method of developing well-conditioned filters will be useful for dealing with a variety of ill-conditioned problems.

Our contributions are summarized as follows:

1. To address the problem of ill-conditioning of the Wiener filter, we describe a method based on which we introduce four new approximate Wiener filter formulas that do not directly involve inverting the observation covariance matrix (Section 4.3). We also exploit the peculiar distribution of eigenvalues observed in large covariance matrices using the method of principal component analysis. The approximations developed in this chapter are justified whenever inverting the observation covariance matrix is ill-conditioned.

2. We prove that the approximate formulas converge to the Wiener filter as certain approximating terms vanish. We also characterize the asymptotic scaling laws (Section 4.4). Our asymptotic analysis is based on elementary tools accessible to anyone familiar with the application of linear algebra to statistical signal processing problems and will help the reader not accustomed to these types of analyses to develop an intuitive understanding of how to make calculations more tractable without using any esoteric or heavy machinery.

3. We describe two methods for determining the optimal number of principal components used in the approximate filter formulas (Section 4.5-4.5.4). Furthermore, we evaluate how the performance of our approximate filters varies with the number of principal components used for truncation and characterize the tradeoff between the numerical unreliability of filter computation and power loss due to truncation.

4. We evaluate the performance of our approximate Wiener filter formulas using daily closing values of the CBOE Amazon VIX Index (Section 4.5-4.5.5). Our quantitative results with the empirical data demonstrate that the performance of our filters remains stable as we increase

the dimension of the associated covariance matrix, while the performance of the Wiener filter deteriorates significantly as expected.

4.2 The Wiener Filter

We wish to estimate a zero-mean random signal $\boldsymbol{X} \in \mathbb{C}^M$ based on a related zero-mean random observation $\boldsymbol{Y} \in \mathbb{C}^N$, where $M, N \in \mathbb{N}$. We use $\mathrm{E}[\cdot]$ as either the expectation or empirical mean from data, $\|\cdot\|$ as the standard complex 2-norm, and $\boldsymbol{A}$ as a matrix representing a linear filter or estimator. Then the LMMSE estimation problem can be stated as follows:

$$\underset{\boldsymbol{A}}{\text{minimize}}\ \mathrm{E}[\|\boldsymbol{A}\boldsymbol{Y} - \boldsymbol{X}\|^2]. \tag{4.1}$$

The optimal solution, called the Wiener filter, is given as

$$\boldsymbol{A}_{\mathrm{W}} := \boldsymbol{C}_{\boldsymbol{X}\boldsymbol{Y}}\boldsymbol{C}_{\boldsymbol{Y}\boldsymbol{Y}}^{-1}, \tag{4.2}$$

where $\boldsymbol{C}_{\boldsymbol{Y}\boldsymbol{Y}} = \mathrm{E}[\boldsymbol{Y}\boldsymbol{Y}'] \in \mathbb{C}^{N\times N}$ is the covariance matrix of the observation $\boldsymbol{Y}$, and $\boldsymbol{C}_{\boldsymbol{X}\boldsymbol{Y}} = \mathrm{E}[\boldsymbol{X}\boldsymbol{Y}'] \in \mathbb{C}^{M\times N}$ is the crosscovariance matrix of the desired signal $\boldsymbol{X}$ and the observation $\boldsymbol{Y}$. Throughout this chapter, we use the superscript prime to denote the Hermitian transpose.

Even though the Wiener filter solves the LMMSE estimation problem, the numerical computation of $\boldsymbol{A}_{\mathrm{W}}$ is unreliable because it involves the inverse $\boldsymbol{C}_{\boldsymbol{Y}\boldsymbol{Y}}^{-1}$. If the condition number of $\boldsymbol{C}_{\boldsymbol{Y}\boldsymbol{Y}}$—the ratio of its largest eigenvalue to its smallest eigenvalue—is large, then the computation of $\boldsymbol{C}_{\boldsymbol{Y}\boldsymbol{Y}}^{-1}$ is numerically unreliable. Notice that we do not need to compute the inverse explicitly. We can simply solve (for $\boldsymbol{A}_{\mathrm{W}}$) the normal equation $\boldsymbol{A}_{\mathrm{W}}\boldsymbol{C}_{\boldsymbol{Y}\boldsymbol{Y}} = \boldsymbol{C}_{\boldsymbol{X}\boldsymbol{Y}}$. Unfortunately, irrespective of the way we compute $\boldsymbol{A}_{\mathrm{W}}$, the reliability of a numerical solution for $\boldsymbol{A}_{\mathrm{W}}$ depends on the condition number of $\boldsymbol{C}_{\boldsymbol{Y}\boldsymbol{Y}}$. A matrix is *ill-conditioned* if it has a large condition number; otherwise, it is *well-conditioned*. Our primary motivation is to present a solution to the LMMSE estimation problem that is reliable to compute regardless of the condition number of $\boldsymbol{C}_{\boldsymbol{Y}\boldsymbol{Y}}$.

Now, there are two major factors that cause $\boldsymbol{C_{YY}}$ to have a large condition number. In many practical problems in signal processing, finance, physics, etc., the dimensions of the observation vectors and their corresponding covariance matrices are very large. According to [97, 98], the mean condition number of a random covariance matrix increases with its size. The second reason, which is closely related to the first, is the ratio of the total number of observation samples to the dimension of the observation vector is small (e.g., because the number of observation samples is limited). In such cases, the few largest eigenvalues of the covariance matrix are significantly larger than the remaining eigenvalues [99–104]. This results in the condition number of the covariance matrix being large as well. Empirical evidence shows that the condition numbers of covariance matrices can easily exceed 10^4 [90, 105]. In practice, these ill-conditioned covariance matrices can cause significant errors when used to solve linear equations.

To address this problem of ill-conditioning, in the next section, we describe approximate Wiener filter formulas that do not directly involve the inverse $\boldsymbol{C_{YY}^{-1}}$. These approximations are justified whenever inverting $\boldsymbol{C_{YY}}$ is ill-conditioned. We introduce multiple filter formulas with varying computational requirements to allow trading off performance for computation. Following that, we show the convergence of these approximate filters to the Wiener filter in terms of their asymptotic scaling laws. Then, we describe two methods for choosing the best approximations to optimally trade off between computation and accuracy. Finally, we evaluate the performance of these filters using empirical data.

4.3 Wiener Filter Approximations

To describe our method for approximating the Wiener filter, we start with the method of [90]. First, define $\boldsymbol{Z} \in \mathbb{C}^{(M+N)}$ to be the concatenation of the vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$:

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix}. \tag{4.3}$$

Let $\boldsymbol{C_{ZZ}} \in \mathbb{C}^{(M+N)\times(M+N)}$ be its covariance. Then, the signal $\boldsymbol{X}$ and the observation $\boldsymbol{Y}$ share the composite covariance matrix

$$\boldsymbol{C_{ZZ}} = \begin{bmatrix} \boldsymbol{C_{XX}} & \boldsymbol{C_{XY}} \\ \boldsymbol{C'_{XY}} & \boldsymbol{C_{YY}} \end{bmatrix}. \tag{4.4}$$

Define the eigendecomposition $\boldsymbol{C_{ZZ}} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{V'}$, where $\boldsymbol{V} \in \mathbb{C}^{(M+N)\times(M+N)}$ is the orthogonal eigenvector matrix composed of eigenvectors of $\boldsymbol{C_{ZZ}}$ as its columns, and $\boldsymbol{S} \in \mathbb{C}^{(M+N)\times(M+N)}$ is the diagonal eigenvalue matrix with diagonal elements $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{M+N}$. The matrices $\boldsymbol{V}$ and $\boldsymbol{S}$ depend jointly on $\boldsymbol{C_{XX}}$, $\boldsymbol{C_{YY}}$, and $\boldsymbol{C_{XY}}$. Next, partition $\boldsymbol{V}$ into $\boldsymbol{V_X}$ (top M rows) and $\boldsymbol{V_Y}$ (bottom N rows). The following identities are easy to verify:

$$\boldsymbol{C_{YY}} = \boldsymbol{V_Y}\boldsymbol{S}\boldsymbol{V'_Y}, \qquad \boldsymbol{C_{XY}} = \boldsymbol{V_X}\boldsymbol{S}\boldsymbol{V'_Y}. \tag{4.5}$$

The eigendecomposition, or singular-value decomposition (SVD), here is different from the ones usually associated with Wiener filters. For example, in deriving a low-rank version of the Wiener filter, [82] uses the SVD of $\boldsymbol{C_{XY}}\boldsymbol{C_{YY}^{-1/2}}$; see also [85], [86], and [88].

4.3.1 Truncation Approximation

In many real-world problems, such as stock price prediction and direction-of-arrival estimation, we have to deal with large covariance matrices [99–105]. In these types of problems, the dimension N of $\boldsymbol{Y}$ can be as large as one thousand or even more. The corresponding covariance matrix $\boldsymbol{C_{YY}}$ is of size $N \times N$ and can be transformed using various techniques. The technique of eigendecomposition for transforming a large covariance matrix is of particular interest because the covariance matrix is symmetric by definition, and for a large N, the eigenvalues of covariance matrices vary widely in magnitude. For example, in financial covariance matrices, the first few eigenvalues are often well-separated from the rest of the eigenvalues and account for the bulk of the information, while the remaining eigenvalues decay rapidly toward zero [99–105]. As the dimension N of $\boldsymbol{Y}$ increases, the proportion of the number of eigenvalues corresponding to this bulk of the information decreases.

As the separation between the first and the last eigenvalue increases, so does the ill-conditioning of the corresponding covariance matrix $\boldsymbol{C_{YY}}$. In such situations, computing a numerical solution in a linear system involving the ill-conditioned $\boldsymbol{C_{YY}}$ becomes unreliable. However, as only the first few eigenvalues carry most of the information, we can safely approximate the relatively small eigenvalues of $\boldsymbol{C_{YY}}$ as 0. To elaborate, suppose the first $L \leq N$ eigenvalues account for the bulk of the eigenvalues of $\boldsymbol{C_{YY}}$; we call these the principal eigenvalues. Next, partition $\boldsymbol{V_X}$, $\boldsymbol{V_Y}$, and $\boldsymbol{S}$ as

$$\begin{aligned}
\boldsymbol{V_X} &= \begin{bmatrix} \boldsymbol{V}_{\boldsymbol{X},L} & \boldsymbol{V}_{\boldsymbol{X},\overline{L}} \end{bmatrix}, \\
\boldsymbol{V_Y} &= \begin{bmatrix} \boldsymbol{V}_{\boldsymbol{Y},L} & \boldsymbol{V}_{\boldsymbol{Y},\overline{L}} \end{bmatrix}, \\
\boldsymbol{S} &= \begin{bmatrix} \boldsymbol{S}_L & \boldsymbol{O}_{L\times(M+N-L)} \\ \boldsymbol{O}_{(M+N-L)\times L} & \boldsymbol{S}_{\overline{L}} \end{bmatrix},
\end{aligned} \tag{4.6}$$

where $\boldsymbol{V}_{\boldsymbol{X},L} \in \mathbb{C}^{M\times L}$, $\boldsymbol{V}_{\boldsymbol{Y},L} \in \mathbb{C}^{N\times L}$, $\boldsymbol{S}_L \in \mathbb{C}^{L\times L}$, and $\boldsymbol{O}$ is the all-zero matrix (with dimensions shown in the subscript). Since these L principal eigenvalues constitute the bulk of all the eigenvalues of $\boldsymbol{C_{YY}}$, we can safely approximate $\boldsymbol{S}_{\overline{L}}$ by $\boldsymbol{O}_{(M+N-L)\times(M+N-L)}$, called *truncation* (see, e.g., [85, 90, 105]). Equivalently, we approximate the decomposition by $\boldsymbol{C_{YY}} \approx \boldsymbol{V}_{\boldsymbol{Y},L}\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L}$.

Consider the following transform of $\boldsymbol{Z}$:

$$\tilde{\boldsymbol{Z}} = \boldsymbol{V}'\boldsymbol{Z} = \boldsymbol{V}'_{\boldsymbol{X}}\boldsymbol{X} + \boldsymbol{V}'_{\boldsymbol{Y}}\boldsymbol{Y}. \tag{4.7}$$

Then, the inverse transform is $\boldsymbol{Z} = \boldsymbol{V}\tilde{\boldsymbol{Z}}$, giving $\boldsymbol{X} = \boldsymbol{V_X}\tilde{\boldsymbol{Z}}$ and $\boldsymbol{Y} = \boldsymbol{V_Y}\tilde{\boldsymbol{Z}}$. We now apply the truncation approximation to get the approximate formulas

$$\tilde{\boldsymbol{Z}}_a = \boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{X} + \boldsymbol{V}'_{\boldsymbol{Y},L}\boldsymbol{Y}, \tag{4.8}$$

and

$$\boldsymbol{X} = \boldsymbol{V}_{\boldsymbol{X},L}\tilde{\boldsymbol{Z}}_a. \tag{4.9}$$

Combining (4.8) and (4.9), we get

$$\begin{aligned} \boldsymbol{X} &\approx \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{X} + \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{Y},L}\boldsymbol{Y} \\ \implies \boldsymbol{X} &\approx (\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^{-1}\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{Y},L}\boldsymbol{Y}. \end{aligned} \tag{4.10}$$

This suggests the following filter:

$$\boxed{\boldsymbol{A}_1 = (\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^{-1}\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{Y},L}.} \tag{4.11}$$

Notice that $\boldsymbol{C}_{\boldsymbol{YY}}^{-1}$ or its eigenvalues do not appear in (4.11).

Next, we combine the Neumann series expansion

$$(\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^{-1} = \sum_{k=0}^{\infty}(\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^k \tag{4.12}$$

with the identity $(\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^k\boldsymbol{V}_{\boldsymbol{X},L} = \boldsymbol{V}_{\boldsymbol{X},L}(\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})^k$ to get $(\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^{-1}\boldsymbol{V}_{\boldsymbol{X},L} = \boldsymbol{V}_{\boldsymbol{X},L}(\boldsymbol{I}_L - \boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})^{-1}$. This leads to an alternative filter formula:

$$\boxed{\boldsymbol{A}_2 = \boldsymbol{V}_{\boldsymbol{X},L}(\boldsymbol{I}_L - \boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})^{-1}\boldsymbol{V}'_{\boldsymbol{Y},L}.} \tag{4.13}$$

Though (4.11) and (4.13) are equivalent formulas, their computational burden might differ. We investigate this difference later.

The filters $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are similar to the LSJPC filter in [90], which was derived using a similar truncation operation and is well-conditioned. The LSJPC filter was shown to have stable performance, even when the covariance matrix is ill-conditioned [90]. This indicates propitious performance for $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$.

4.3.2 Inverse Approximation

Though we do not limit ourselves to any particular matrix norm, we assume that the matrix norm is submultiplicative. Many matrix norms, such as the 1-norm, the infinity-norm, and the

Frobenius norm, have the submultiplicative property. However, there are some other matrix norms, such as the max norm, that do not possess the submultiplicative property. Furthermore, we assume that $\|\boldsymbol{I}_M\| = 1$ (otherwise we need to normalize by $\|\boldsymbol{I}_M\|$). Considering $\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L} + \boldsymbol{V}_{\boldsymbol{X},\overline{L}}\boldsymbol{V}'_{\boldsymbol{X},\overline{L}} = \boldsymbol{I}_M$, we see that $\|\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\| < 1$. This means that $\|\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\|^k$ can be made arbitrarily small with sufficiently large k. Hence, we can approximate the Neumann series expansion

$$(\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^{-1} = \sum_{k=0}^{\infty}(\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^k \tag{4.14}$$

by a finite sum:

$$(\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^{-1} \approx \sum_{k=0}^{K}(\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^k. \tag{4.15}$$

Applying this approximation to (4.11), we get the filter

$$\boxed{\boldsymbol{A}_3 = \left(\sum_{k=0}^{K}(\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})^k\right)\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{Y},L}.} \tag{4.16}$$

Similarly, from (4.13), we get

$$\boxed{\boldsymbol{A}_4 = \boldsymbol{V}_{\boldsymbol{X},L}\left(\sum_{k=0}^{K}(\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})^k\right)\boldsymbol{V}'_{\boldsymbol{Y},L}.} \tag{4.17}$$

Finally, note that computing the estimate $\hat{\boldsymbol{X}} = \boldsymbol{A}_3\boldsymbol{Y}$ based on (4.16) can be done iteratively using

$$\begin{aligned} &\hat{\boldsymbol{X}}(k+1) = (\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L})\hat{\boldsymbol{X}}(k) + \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{Y},L}\boldsymbol{Y}, \\ &k = 0, 1, \ldots, K-1, \quad \hat{\boldsymbol{X}}(0) = \boldsymbol{0}. \end{aligned} \tag{4.18}$$

Because $\|\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\| < 1$, (4.18) is a contractive fixed-point iteration, guaranteeing convergence of $\hat{\boldsymbol{X}}$ to (4.11) as $K \to \infty$. Similarly, computing the estimate $\boldsymbol{A}_4\boldsymbol{Y}$ based on (4.17) can be done

iteratively using

$$\begin{aligned}
&\boldsymbol{U}(k+1) = (\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})\boldsymbol{U}(k) + \boldsymbol{V}'_{\boldsymbol{Y},L}\boldsymbol{Y}, \qquad (4.19)\\
&k = 0, 1, \ldots, K-1, \quad \boldsymbol{U}(0) = \boldsymbol{0},\\
&\hat{\boldsymbol{X}} = \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{U}(K).
\end{aligned}$$

We use the notation $\boldsymbol{A}_3(K)$ (or $\boldsymbol{A}_4(K)$) to indicate the Kth Neumann-approximation order for the filter $\boldsymbol{A}_3$ (or $\boldsymbol{A}_4$, respectively). The special case of $K = 0$ gives the simple formula $\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{Y},L}$, which is difficult to imagine simplifying any further.

In the cases where $\boldsymbol{C}_{\boldsymbol{YY}}$ is ill-conditioned and its eigenvalues decrease very quickly, it is possible to find an appropriate L for truncation. Similar truncations have been used in low-rank versions of the Wiener filter [82, 83, 85, 86, 88, 90, 106]. Naturally, in choosing L there is a tradeoff between the quality of the approximation and the computational burden. Therefore, there is no universally applicable value for L. Indeed, this tradeoff affords flexibility in implementing filters under different computational constraints.

It is not yet entirely clear how well our approximate filters perform in practice. Our derivations for the approximate filters described above are found in simple heuristics. To ensure the viability of these filters, we need to carefully analyze and evaluate them further. In Section 4.4, we show how these approximations converge to $\boldsymbol{A}_{\mathrm{W}}$ as the truncation and inverse approximations become exact. Later, in Section 4.5, we evaluate these formulas using empirical data.

Our approach to addressing the problem of ill-conditioning involves dimension reduction. An added benefit of dimension reduction is that we can expect the computational burden of applying the filters to be reduced relative to the Wiener filter. Our empirical results in Section4.5-4.5.5 corroborate this expectation.

4.4 Asymptotic Performance

We now show how each of the approximation formulas derived in Section 4.3 converges to $\boldsymbol{A}_{\mathrm{W}}$ using the *Bachmann-Landau* notation $\mathcal{O}(\cdot)$. To explain, consider a matrix $\boldsymbol{M}(\rho)$ that depends on some parameter $\rho \to 0$. If, for some c, $\|\boldsymbol{M}(\rho)\| \leq c\rho$ for sufficiently small ρ, then we write $\boldsymbol{M}(\rho) = \mathcal{O}(\rho)$. If $\boldsymbol{C}$ and $\boldsymbol{D}$ are bounded as $\rho \to 0$, then the following algebraic rules help us to simplify the calculations:

$$\boldsymbol{C}\mathcal{O}(\rho) = \mathcal{O}(\rho), \tag{4.20}$$

$$(\boldsymbol{C} + \mathcal{O}(\rho))^{-1} = \boldsymbol{C}^{-1} + \mathcal{O}(\rho), \text{ and} \tag{4.21}$$

$$(\boldsymbol{C} + \mathcal{O}(\rho))(\boldsymbol{D} + \mathcal{O}(\rho)) = \boldsymbol{C}\boldsymbol{D} + \mathcal{O}(\rho). \tag{4.22}$$

Whenever we use the truncation operation for some values of L and K, we essentially lose some of the power in the observed signal $\boldsymbol{Y}$. We measure this *truncation-power loss* as ρ_L. The amount of power lost due to truncation is small by design. The parameter ρ_L is a measure of this amount of power being discarded. The exact expression for the power being discarded depends on the filter. For $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, $\rho_L = \|\boldsymbol{S}_{\overline{L}}\|$. For $\boldsymbol{A}_3$, $\rho_L = \|\boldsymbol{S}_{\overline{L}}\| + \|\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\|^{K+1}$. And, for $\boldsymbol{A}_4$, $\rho_L = \|\boldsymbol{S}_{\overline{L}}\| + \|\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}\|^{K+1}$. The truncation approximation relies on $\|\boldsymbol{S}_{\overline{L}}\| \approx 0$. Moreover, the inverse approximation relies on either $\|\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\| \approx 0$ or $\|\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}\| \approx 0$. Thus, in our analysis, we consider $\boldsymbol{S}_{\overline{L}}$ and $\boldsymbol{V}_{\boldsymbol{X}}$ to be variables such that $\|\boldsymbol{S}_{\overline{L}}\| \to 0$, $\|\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\| \to 0$, and $\|\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}\| \to 0$. We treat $\boldsymbol{S}_L$ as fixed. Note that $\boldsymbol{V}_{\boldsymbol{X}}$ and $\boldsymbol{V}_{\boldsymbol{Y}}$, and their submatrices $\boldsymbol{V}_{\boldsymbol{X},L}$ and $\boldsymbol{V}_{\boldsymbol{Y},L}$, are bounded as $\|\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\| \to 0$ and $\|\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}\| \to 0$. This is relevant for applying the Bachmann-Landau rules.

Theorem 1. *Given a positive integer $L \leq N$,*

$$\boldsymbol{A}_{\mathrm{W}} - \boldsymbol{A}_2 = \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|). \tag{4.23}$$

Proof. We first write

$$\begin{aligned}
\boldsymbol{C}_{\boldsymbol{XY}} &= \boldsymbol{V}_{\boldsymbol{X}}\boldsymbol{S}\boldsymbol{V}'_{\boldsymbol{Y}} = \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L} + \boldsymbol{V}_{\boldsymbol{X},\overline{L}}\boldsymbol{S}_{\overline{L}}\boldsymbol{V}'_{\boldsymbol{Y},\overline{L}} \\
&= \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L} + \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|) \\
\boldsymbol{C}_{\boldsymbol{YY}} &= \boldsymbol{V}_{\boldsymbol{Y}}\boldsymbol{S}\boldsymbol{V}'_{\boldsymbol{Y}} = \boldsymbol{V}_{\boldsymbol{Y},L}\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L} + \boldsymbol{V}_{\boldsymbol{Y},\overline{L}}\boldsymbol{S}_{\overline{L}}\boldsymbol{V}'_{\boldsymbol{Y},\overline{L}} \\
&= \boldsymbol{V}_{\boldsymbol{Y},L}\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L} + \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|). \qquad (4.24)
\end{aligned}$$

Therefore, using the Bachmann-Landau rules,

$$\begin{aligned}
\boldsymbol{A}_{\mathrm{W}} &= \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L}(\boldsymbol{V}_{\boldsymbol{Y},L}\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L})^{-1} + \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|) \\
&= \boldsymbol{V}_{\boldsymbol{X},L}(\boldsymbol{V}'_{\boldsymbol{Y},L}\boldsymbol{V}_{\boldsymbol{Y},L})^{-1}(\boldsymbol{V}'_{\boldsymbol{Y},L}\boldsymbol{V}_{\boldsymbol{Y},L})\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L} \\
&\quad (\boldsymbol{V}_{\boldsymbol{Y},L}\boldsymbol{S}_L\boldsymbol{V}'_{\boldsymbol{Y},L})^{-1} + \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|) \\
&= \boldsymbol{V}_{\boldsymbol{X},L}(\boldsymbol{V}'_{\boldsymbol{Y},L}\boldsymbol{V}_{\boldsymbol{Y},L})^{-1}\boldsymbol{V}'_{\boldsymbol{Y},L} + \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|) \\
&= \boldsymbol{V}_{\boldsymbol{X},L}(\boldsymbol{I}_L - \boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})^{-1}\boldsymbol{V}'_{\boldsymbol{Y},L} + \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|), \qquad (4.25)
\end{aligned}$$

which completes the proof. □

Corollary 1. *Given a positive integer $L \leq N$,*

$$\boldsymbol{A}_{\mathrm{W}} - \boldsymbol{A}_1 = \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|). \qquad (4.26)$$

Theorem 2. *Given positive integers $L \leq N$ and K,*

$$\boldsymbol{A}_{\mathrm{W}} - \boldsymbol{A}_4 = \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|) + \mathcal{O}(\|\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}\|^{K+1}). \qquad (4.27)$$

Proof. Using the Neumann series expansion

$$\begin{aligned}(\boldsymbol{I}_L - \boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})^{-1} &= \sum_{k=0}^{\infty}(\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})^k \\ &= \sum_{k=0}^{K}(\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L})^k \\ &\quad + \mathcal{O}(\|\boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}\|^{K+1}). \end{aligned} \tag{4.28}$$

Combining this with (4.23), we get the desired result. □

Corollary 2. *Given positive integers $L \leq N$ and K,*

$$\boldsymbol{A}_{\mathrm{W}} - \boldsymbol{A}_3 = \mathcal{O}(\|\boldsymbol{S}_{\overline{L}}\|) + \mathcal{O}(\|\boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}\|^{K+1}). \tag{4.29}$$

The results above, stated in terms of the Bachmann-Landau notation, provide only an asymptotic characterization of the difference between each filter and $\boldsymbol{A}_{\mathrm{W}}$, not an exact expression for it. However, inspecting the calculations above gives some idea of how impractical an exact analysis would be.

4.5 Results and Discussion

We now compare, in terms of accuracy and computation time with real data, the performance of the different approximate filters developed in Section 4.3 relative to the Wiener filter. Our performance-complexity evaluation has two parts. First, we test the approximate filters as we vary the conditioning of the observation covariance matrix. Second, we quantify the price paid for well conditioning of $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\boldsymbol{A}_3$, and $\boldsymbol{A}_4$ in the preprocessing steps relative to the Wiener filter. We expect the performance of the Wiener filter to deteriorate as the conditioning of the associated observation covariance matrix worsens while the performance of the approximate filters does not deteriorate. Further, we expect the approximate filters to be computationally more efficient than the Wiener filter because of the reduced dimensions involved. We also describe and compare two

methods for selecting the optimal filter orders. The results from the computational comparison elucidate how the complexities of the approximate filters scale with different dimensions of the associated covariance matrices and approximation orders. Although some of the specific numerical values are data-dependent, they illustrate the general qualitative features of our methods.

4.5.1 Dataset

To demonstrate the performance-complexity tradeoff of each of our approximate filters relative to the Wiener filter, we use the CBOE Amazon VIX index (VXAZN) daily closing values. The historical VXAZN data used in this chapter is freely available at [91]. To check for stationarity in time of the VXAZN data, following the method in [107, 108], we implemented the augmented Dickey-Fuller (ADF) test [109] for up to 40 lags. The ADF test checks for the null hypothesis that the process follows a unit root, i.e., it is non-stationary. Our ADF test results for the VXAZN data, produced using the HypothesisTests package in Julia [110], strongly reject the null hypothesis with a maximum p-value of 0.0006. Therefore, we can safely assume that the VXAZN data sequence is stationary.

4.5.2 Data Processing

For the purpose of our estimation problem, we use a sequence of 2974 daily closing values of the VXAZN dataset, starting on January 7, 2011, and ending on November 4, 2022. Our estimation problem can be stated as follows: given the VXAZN values for N consecutive days, predict the values for M consecutive days immediately after. To do this, we consider vector-valued samples of $M+N$ consecutive days. We get a total of $T = 2974-(M+N)+1$ such vector-valued samples, each one shifted by one lag from the previous one. We center the data by subtracting the empirical mean of the data matrix from each sample. We then construct a data matrix

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{W}_2 & \cdots & \boldsymbol{W}_T \end{bmatrix}$$

$$= \begin{bmatrix} W(1) & W(2) & \cdots & W(T) \\ W(2) & W(3) & \cdots & W(T+1) \\ \vdots & \vdots & \ddots & \vdots \\ W(M+N) & W(M+N+1) & \cdots & W(2974) \end{bmatrix}, \tag{4.30}$$

where $W(i)$ represents the centered closing value of the VXAZN dataset for the day i. Because the dataset is stationary, the columns of $\boldsymbol{W}$ have a common $(M+N)$-variate mean and covariance. Then, we compute the sample covariance matrix $\boldsymbol{C_{WW}}$ as

$$\boldsymbol{C_{WW}} = \frac{1}{T-1} \sum_{i=1}^{T} \boldsymbol{W}_i \boldsymbol{W}_i'. \tag{4.31}$$

We use 80% of the data samples for computing the sample covariance matrix and reserve the remaining 20% for conducting out-of-sample test experiments. We obtain the 80% above randomly from the data matrix.

4.5.3 Ill-Conditioning of $\boldsymbol{C_{YY}}$

Our aim for the performance-complexity analysis is to test how the different approximate filters derived in Section 4.3 perform relative to the Wiener filter when the conditioning of the observation covariance matrix is varied. To test this, we fix $M = 7$ and vary N from 100 to 1300. Because the total number of data values is fixed (2974 in our case), as we increase the dimension of our vector-valued data sample, the total number of available data samples T decreases. We use $\gamma = (M+N)/T$ to denote their ratio. As the dimension of the data sample approaches the total number of available data samples (i.e., $\gamma \to 1$), the resulting covariance matrix becomes more ill-conditioned. Figure 4.1 shows how the condition number of $\boldsymbol{C_{YY}}$ changes as $\gamma \to 1$. We can see that, as γ rises from 0.04 to 0.97, the resulting condition number of $\boldsymbol{C_{YY}}$ increases by more than three orders of magnitude. This allows performance evaluation across many scenarios.

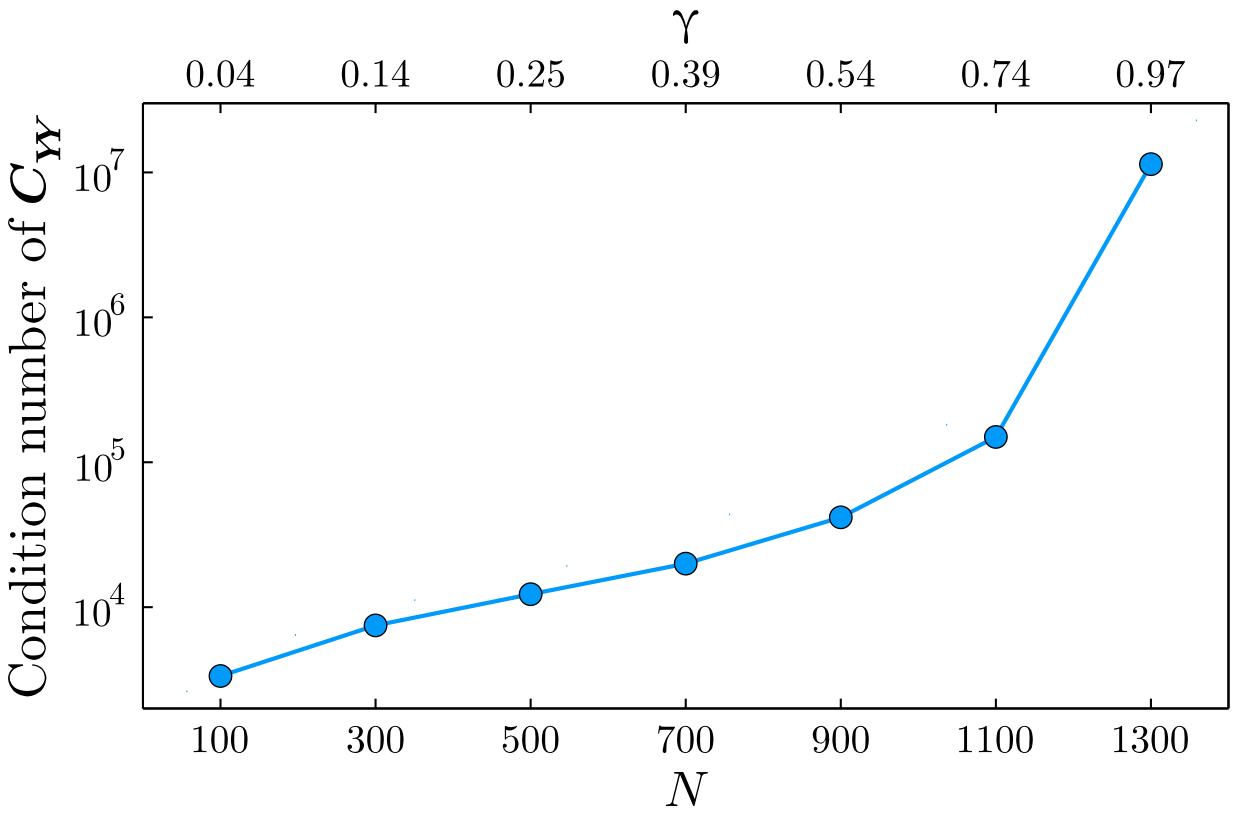

Figure 4.1: Condition number of $\boldsymbol{C_{YY}}$ vs. N and γ.

To keep the performance metric directly comparable as we vary N, we evaluate the performance according to the normalized root-mean-square error (nRMSE), calculated as

$$\sqrt{\frac{1}{T}\sum_{i=1}^{T}\|\boldsymbol{A}\boldsymbol{W}_i(\boldsymbol{Y})-\boldsymbol{W}_i(\boldsymbol{X})\|^2}\Bigg/\sqrt{\frac{1}{T}\sum_{i=1}^{T}\|\boldsymbol{W}_i(\boldsymbol{X})+\overline{\boldsymbol{W}}\|^2},$$

where $\boldsymbol{W}_i \in \mathbb{R}^{(M+N)}$ $(i = 1, \ldots, T)$ is the vector-valued sample, $\boldsymbol{W}_i(\boldsymbol{Y})$ is the N-subvector of $\boldsymbol{W}_i$ corresponding to $\boldsymbol{Y}$, $\boldsymbol{W}_i(\boldsymbol{X})$ is the M-subvector corresponding to $\boldsymbol{X}$, and $\overline{\boldsymbol{W}}$ is the average M-vector. Better performance values are closer to 0.

4.5.4 Best L

The performance of $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\boldsymbol{A}_3$ and $\boldsymbol{A}_4$ varies with the number of principal components L used for truncation. To illustrate the effect of different values of L on the performance, Figure 4.2

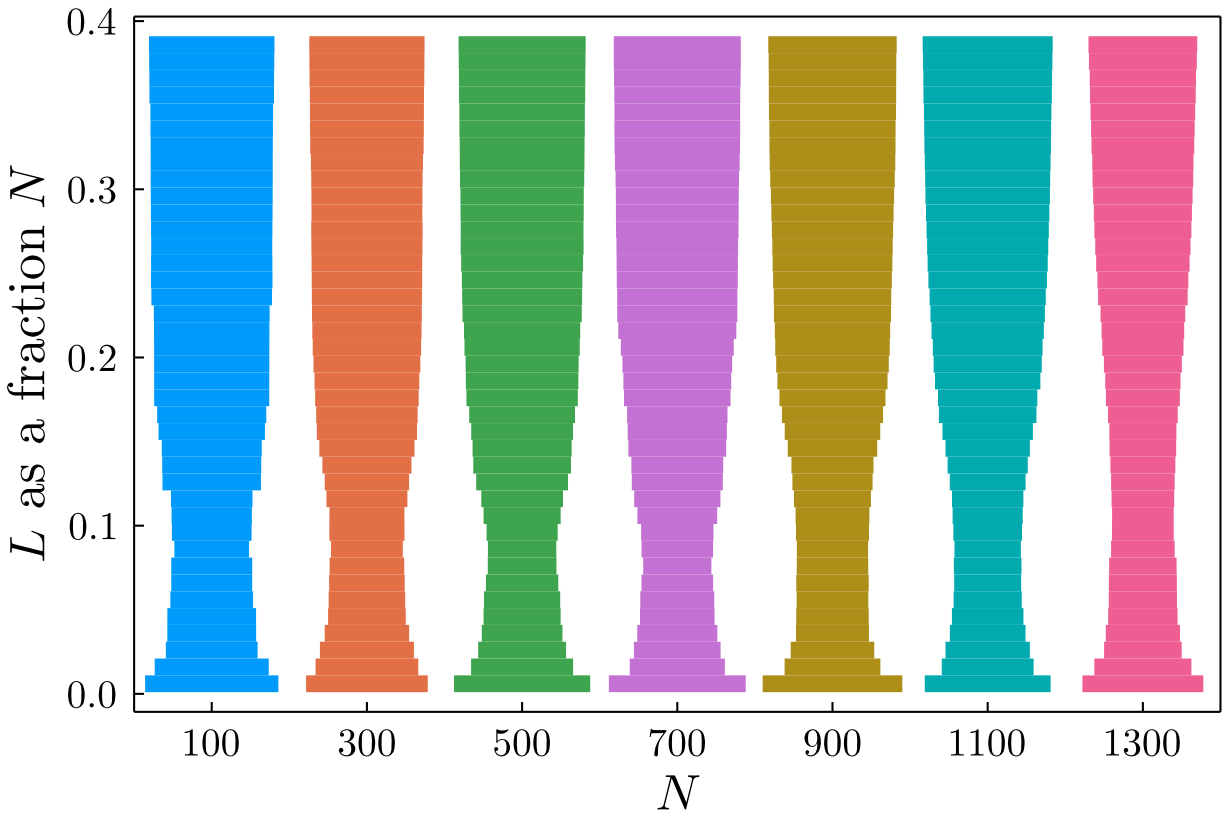

Figure 4.2: Change in filter performance as L varies.

depicts the performance of $\boldsymbol{A}_3(5)$ for different values of N as we vary L. Here, the thickness of each column rung for different values of N indicates the performance in terms of nRMSE, with the maximum thickness of a column rung showing an nRMSE value of 0.28. We can see that the performance of $\boldsymbol{A}_3(5)$ first improves as we increase L and then deteriorates as we keep increasing L. Clearly, there is some optimal value of L for which the filter performs best. This optimal value is governed by the tradeoff between the numerical unreliability of filter computation and the truncation-power loss as we increase L. The filter performance is more sensitive to this optimal value of L for smaller values of N, but for larger values of N, we can vary L by a relatively wide margin without affecting its performance by much. The performance of $\boldsymbol{A}_3$ and $\boldsymbol{A}_4$ also varies with the Neumann-approximation order. In our experiments, their performance is essentially constant beyond $K = 5$. Even when we change the dimension of M, $K = 5$ is the smallest Neumann-

approximation order of $\boldsymbol{A}_3$ and $\boldsymbol{A}_4$ that matches the best performance of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, beyond which there is little to separate between their performance.

Next, we describe two methods for computing the best L value to get the optimal filter performance.

Line-Search Method

Our objective function value, the mean square error, for any given $\boldsymbol{A}$ can also be expressed as

$$\mathrm{E}[\|\boldsymbol{AY} - \boldsymbol{X}\|^2] = \mathrm{tr}(\boldsymbol{C_{XX}} - 2\boldsymbol{C_{XY}}\boldsymbol{A}' + \boldsymbol{A}\boldsymbol{C_{YY}}\boldsymbol{A}'). \tag{4.32}$$

Following the method in [90], we can find a suitable value of L for any filter $\boldsymbol{A}$ using a simple line-search (LS) procedure [106] together with the mean-square-error formula in (4.32).

The LS method for choosing the best L is computationally costly because we need to compute $\boldsymbol{A}$ multiple times for different values of L.

Marchenko-Pastur Method

Following the method in [92], we can find the best L from the data matrix without the explicit need for computing $\boldsymbol{A}$. Recall that $\boldsymbol{W} \in \mathbb{R}^{(M+N)\times T}$ is the mean-subtracted data matrix and $\boldsymbol{C_{WW}}$ is its covariance matrix. Without loss of generality, we assume $(M + N) < T$. If $\boldsymbol{W}$ is a random matrix, i.e., $\boldsymbol{W}$ has independent and identically distributed (i.i.d.) entries with mean 0 and variance σ^2, we can view the eigenvalues $\lambda_1, \ldots, \lambda_{M+N}$ of $\boldsymbol{C_{WW}}$ as random variables. Then, in agreement with an asymptotic universal law from RMT, the eigenvalues of $\boldsymbol{C_{WW}}$ have the Marchenko-Pastur (MP) probability density function [111], which is fully characterized by the dimensions of the data matrix ($(M + N)$ and T) and the variance (σ^2). According to [112], if $\boldsymbol{C_{WW}}$ has a non-zero number of eigenvalues that are well-separated from the rest of the eigenvalues, then we can infer that the entries of $\boldsymbol{W}$ are not i.i.d. According to [92, 112–116], if a data matrix can be synthesized by $L < (M + N)$ principal components, then the MP distribution still applies to the remaining $M + N - L$ eigenvalues of its covariance matrix. In many real-world problems, the

Table 4.1: Computation time for the line search (LS) method and the Marchenko-Pastur (MP) method for calculating the best L as N varies.

N	LS:$\boldsymbol{A}_1$ Time (ms)	LS: $\boldsymbol{A}_3(5)$ Time (ms)	MP Time (ms)
100	3.61	4.12	0.0039
300	17.1	17.9	0.0101
500	48.8	50.0	0.0256
700	91.5	92.4	0.0352
900	161.0	163.0	0.0930
1100	252.0	253.0	0.1305
1300	383.0	386.0	0.1655

largest L eigenvalues, corresponding to the principal components of the covariance matrix, are well-separated from the remaining eigenvalues. The latter are packed together within the support of the MP density [112, 116–124]. For such covariance matrices, we can jointly estimate L and σ^2 by determining the minimum value of L such that the λ_{L+1} to λ_{M+N} eigenvalues are well described by the MP distribution. Following the method in [92], we can easily find the minimum L that satisfies the inequality

$$\frac{\lambda_{L+1} - \lambda_{M+N}}{4\sqrt{(M+N-L)/T}} \leq \frac{1}{(M+N-L)} \sum_{i=L+1}^{M+N} \lambda_i. \tag{4.33}$$

This method works because if at least one of the eigenvalues corresponding to the principal components is part of the eigenvalues grouped together in the support of the MP density, then the range (left-hand side of (4.33)) will exceed the mean (right-hand side of (4.33)). Once we identify the minimum L for which the test inequality in (4.33) is satisfied, we can substitute $\lambda_{L+1}, \ldots, \lambda_{M+N}$ with 0 and approximate the Wiener filter using the formulas derived in Section 4.3.

Figure 4.3 shows the best L values for $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$ as we vary N, calculated using the LS and MP methods. Because the LS method depends on the individual filter, we need to calculate the best L values for $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$ separately. In contrast, the MP method is independent of the filter and only depends on the data matrix. From Figure 4.3, we can see that the best L values for both the LS and MP methods are relatively close to each other.

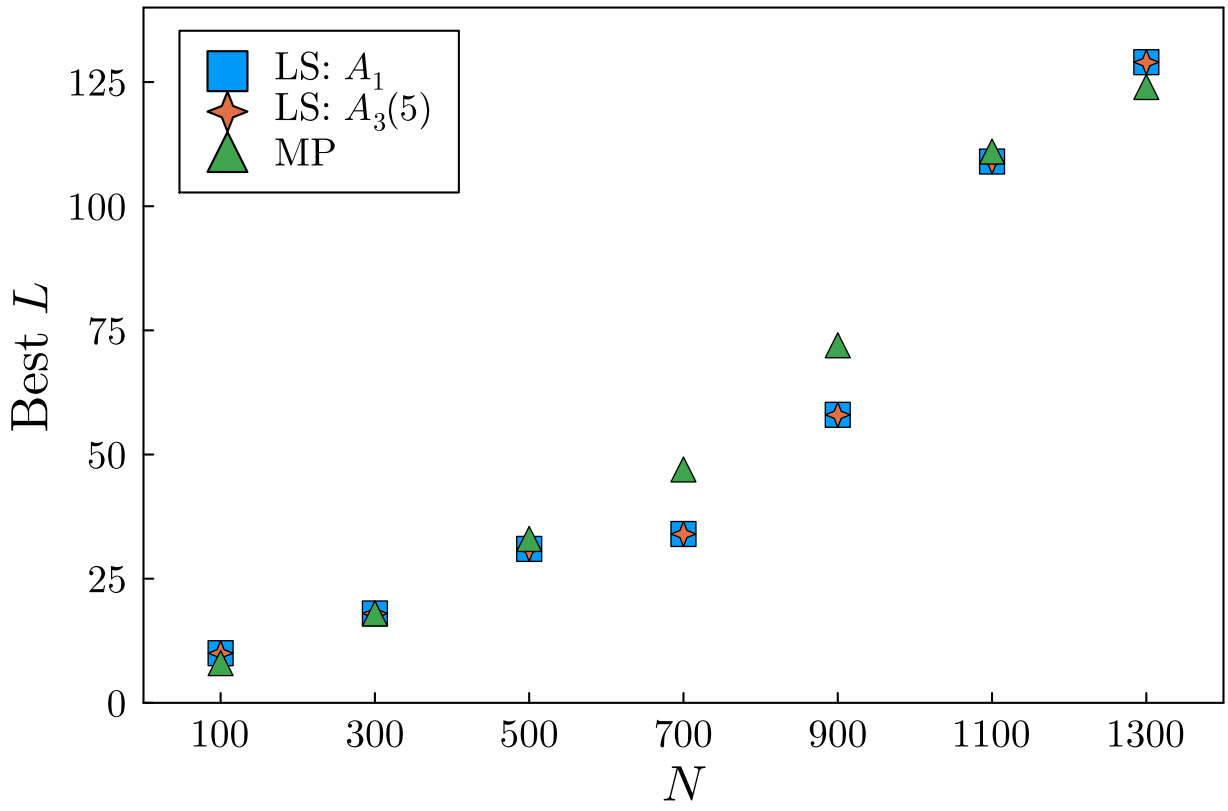

Figure 4.3: Best L value vs. N.

To better differentiate these methods, we analyze the computation time required for each in calculating the best L values. Table 4.1 shows the computation times for the LS and MP methods for calculating the best L value. We can see that the MP method is approximately three orders of magnitude faster than the LS method. As seen earlier in Figure 4.3, the optimal L values computed using both of these methods are comparable. Therefore, henceforth we will use the MP method for computing the optimal L value without compromising the filter performance.

4.5.5 Performance-Complexity Tradeoff

Figure 4.4 compares the performance of $\boldsymbol{A}_1$, $\boldsymbol{A}_3(5)$, and $\boldsymbol{A}_{\mathrm{W}}$ in terms of nRMSE as N increases and $\boldsymbol{C}_{\boldsymbol{YY}}$ becomes more ill-conditioned (see Figure 4.1). For $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$, we used the best L values computed according to the MP method. All filters have comparable performance for smaller values of N, or until the total number of samples is more than double the dimension of

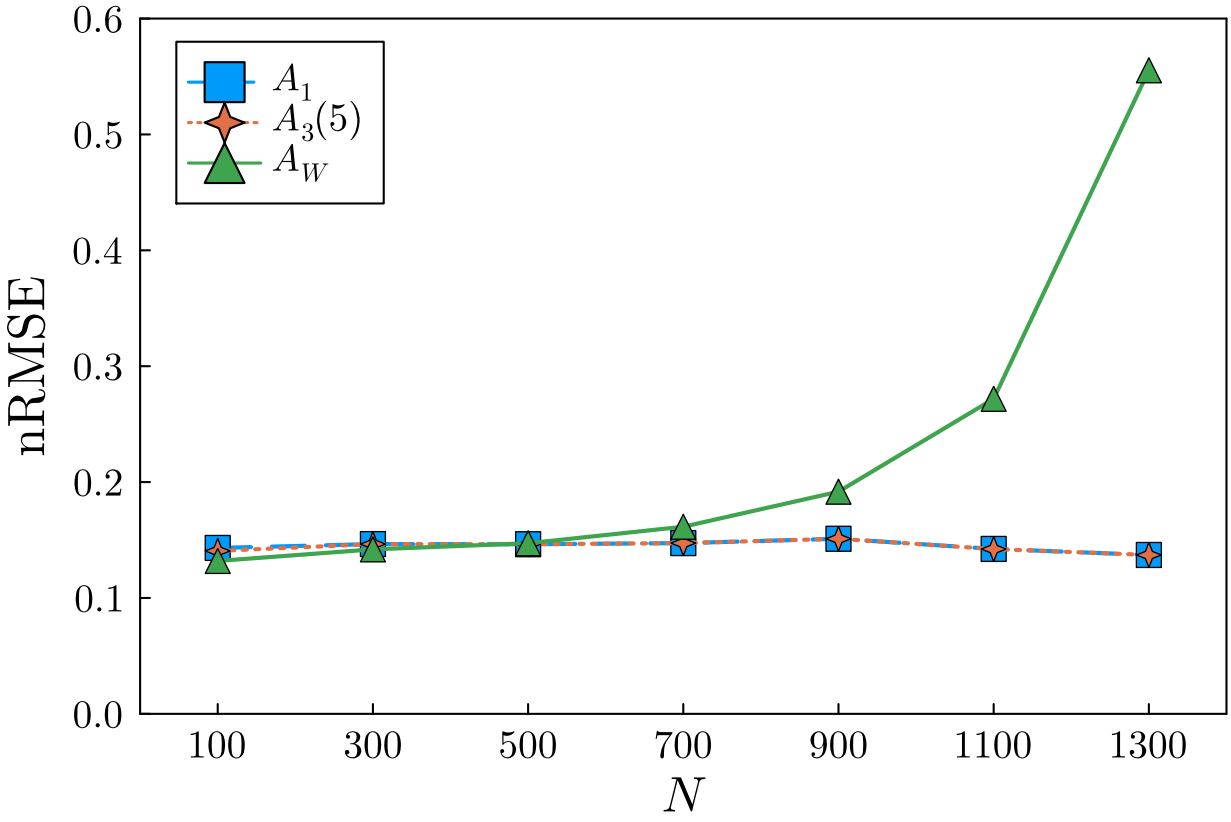

Figure 4.4: nRMSE vs. N.

the data vector. After $N = 700$, when the ill-conditioning of $\boldsymbol{C_{YY}}$ starts to increase rapidly, the computation of $\boldsymbol{A}_{\mathrm{W}}$ becomes more unreliable. This is evidenced by the deteriorating performance of $\boldsymbol{A}_{\mathrm{W}}$ after $N = 700$. But the same is untrue for filters $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$. The nRMSE values for $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$ never exceed 0.15. This demonstrates that our filters have consistently stable performance even when $\boldsymbol{C_{YY}}$ is ill-conditioned.

We also implemented the recursive least squares (RLS) algorithm with a growing window, which is a well-known adaptive filter and employs an iterative approach for filter computation. However, the resulting RLS filter converges to $\boldsymbol{A}_{\mathrm{W}}$ and has the same deteriorating performance as the Wiener filter after $N = 700$, as illustrated in Figure 4.4.

Note that, although there are matrices that involve inverses in computing filters $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ (namely $\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}$ and $\boldsymbol{I}_L - \boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}$), these matrices are well-conditioned. Table 4.2 compares the condition numbers of these matrices in computing filters $\boldsymbol{A}_{\mathrm{W}}$, $\boldsymbol{A}_1$, and $\boldsymbol{A}_2$. We can

Table 4.2: Condition numbers of matrices involved in computing $\boldsymbol{A}_{\mathrm{W}}$, $\boldsymbol{A}_1$, and $\boldsymbol{A}_2$, respectively.

N	Cond. no. of $\boldsymbol{C}_{\boldsymbol{YY}}$	Cond. no. of $\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}$	Cond. no. of $\boldsymbol{I}_L - \boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}$
100	3349.86	7.35	7.35
300	7484.85	2.53	2.53
500	12289.47	2.53	2.53
700	19972.04	2.50	2.50
900	41678.75	4.07	4.07
1100	149494.81	5.38	5.38
1300	1.13e+07	10.79	11.05

Table 4.3: Computation time for $\boldsymbol{A}_1$ and $\boldsymbol{A}_{\mathrm{W}}$ as N varies.

N	$\boldsymbol{A}_1$ w/o preprocessing Time (ms)	$\boldsymbol{A}_1$ w/ preprocessing Time (ms)	$\boldsymbol{A}_{\mathrm{W}}$ Time (ms)	# of applications of $\boldsymbol{A}_1$ to break even
100	0.0039	3.0095	0.4578	7
300	0.0102	16.0639	2.4300	7
500	0.0257	38.4283	5.7700	7
700	0.0351	77.9418	13.900	6
900	0.0859	131.4415	24.000	6
1100	0.1329	209.5738	36.700	6
1300	0.1723	326.5040	56.300	6

see that, as we increase N from 100 to 1300, the resulting condition number of $\boldsymbol{C}_{\boldsymbol{YY}}$ increases by more than three orders of magnitude while the condition numbers of matrices $\boldsymbol{I}_M - \boldsymbol{V}_{\boldsymbol{X},L}\boldsymbol{V}'_{\boldsymbol{X},L}$ and $\boldsymbol{I}_L - \boldsymbol{V}'_{\boldsymbol{X},L}\boldsymbol{V}_{\boldsymbol{X},L}$ remain relatively small, which shows that these matrices are well-conditioned.

Using the approximated filters $\boldsymbol{A}_1$ to $\boldsymbol{A}_4$ requires some preprocessing, which includes computing the eigendecomposition of the covariance matrix and selecting the best L using the LS or MP methods. Indeed, this preprocessing involves some computational costs. Fortunately, the preprocessing has to be done upfront only once, after which we can repeatedly use the formulas for $\boldsymbol{A}_1$ to $\boldsymbol{A}_4$ with very little extra computation. i.e., once we obtain $\boldsymbol{V}_{\boldsymbol{X},L}$ and $\boldsymbol{V}_{\boldsymbol{Y},L}$, computing the approximate filters $\boldsymbol{A}_1$ to $\boldsymbol{A}_4$ is more efficient than computing $\boldsymbol{A}_{\mathrm{W}}$ due to the reduced dimensions of $\boldsymbol{V}_{\boldsymbol{X},L}$ and $\boldsymbol{V}_{\boldsymbol{Y},L}$. Table 4.3 shows the computation times for $\boldsymbol{A}_1$ with and without the preprocessing. We can see that the computation times for $\boldsymbol{A}_1$ with the preprocessing are greater than that of $\boldsymbol{A}_{\mathrm{W}}$, while the computation times for $\boldsymbol{A}_1$ without the preprocessing are two orders of magnitude

Table 4.4: Computation time for all approximate filters as N varies.

N	$\boldsymbol{A}_1$ Time (ms)	$\boldsymbol{A}_2$ Time (ms)	$\boldsymbol{A}_3(5)$ Time (ms)	$\boldsymbol{A}_4(5)$ Time (ms)
100	0.0039	0.0048	0.0300	0.0387
300	0.0102	0.0148	0.0364	0.1011
500	0.0257	0.0419	0.0520	0.3463
700	0.0351	0.0507	0.0608	0.3902
900	0.0859	0.1489	0.1236	1.6000
1100	0.1329	0.7079	0.167	8.4700
1300	0.1723	0.8610	0.1972	11.500

smaller than $\boldsymbol{A}_{\mathrm{W}}$. These results show that $\boldsymbol{A}_1$ is computationally more expensive than $\boldsymbol{A}_{\mathrm{W}}$ when we need to use the filters just once, but their marginal computational burden quickly diminishes over multiple filtering operations. In our experiments, it never takes more than 7 applications of $\boldsymbol{A}_1$ to break even with $\boldsymbol{A}_{\mathrm{W}}$. As shown later, the computation times for $\boldsymbol{A}_2$ to $\boldsymbol{A}_4$ are comparable to $\boldsymbol{A}_1$. Because the best L values are approximately 10% of the corresponding N values, computing the approximate filters post-processing is more efficient compared to $\boldsymbol{A}_{\mathrm{W}}$. It is worth emphasizing that our primary purpose is to address the ill-conditioning of $\boldsymbol{C}_{\boldsymbol{YY}}$. The computation times in Table 4.3 along with the nRMSE performance of the filters in Figure 4.4 demonstrate that our filters are numerically reliable to compute without significantly increasing the computational burden (and even decreasing it in some use cases).

Finally, Table 4.4 compares the computation times for all four approximate filters as we increase N. Table 4.4 does not include the times to compute eigendecompositions and the MP method to select L. Even though $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$ are algebraically equivalent to $\boldsymbol{A}_2$ and $\boldsymbol{A}_4(5)$, respectively, they differ in terms of their computational burden. We can attribute this added computation to the extra floating point operations (FLOPs) needed to compute $\boldsymbol{A}_2$ and $\boldsymbol{A}_4(5)$ compared to their respective algebraic counterparts. The number of FLOPs for $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$ increases with M, while the number of FLOPs for $\boldsymbol{A}_2$ and $\boldsymbol{A}_4(5)$ increases with only L. For smaller values of N, the values of L and M are comparable. This is demonstrated by their comparable computation times for small values of N. As N increases, so does L, whereas M remains constant throughout. Therefore, for large values of N, the computation times for $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$ are much lower

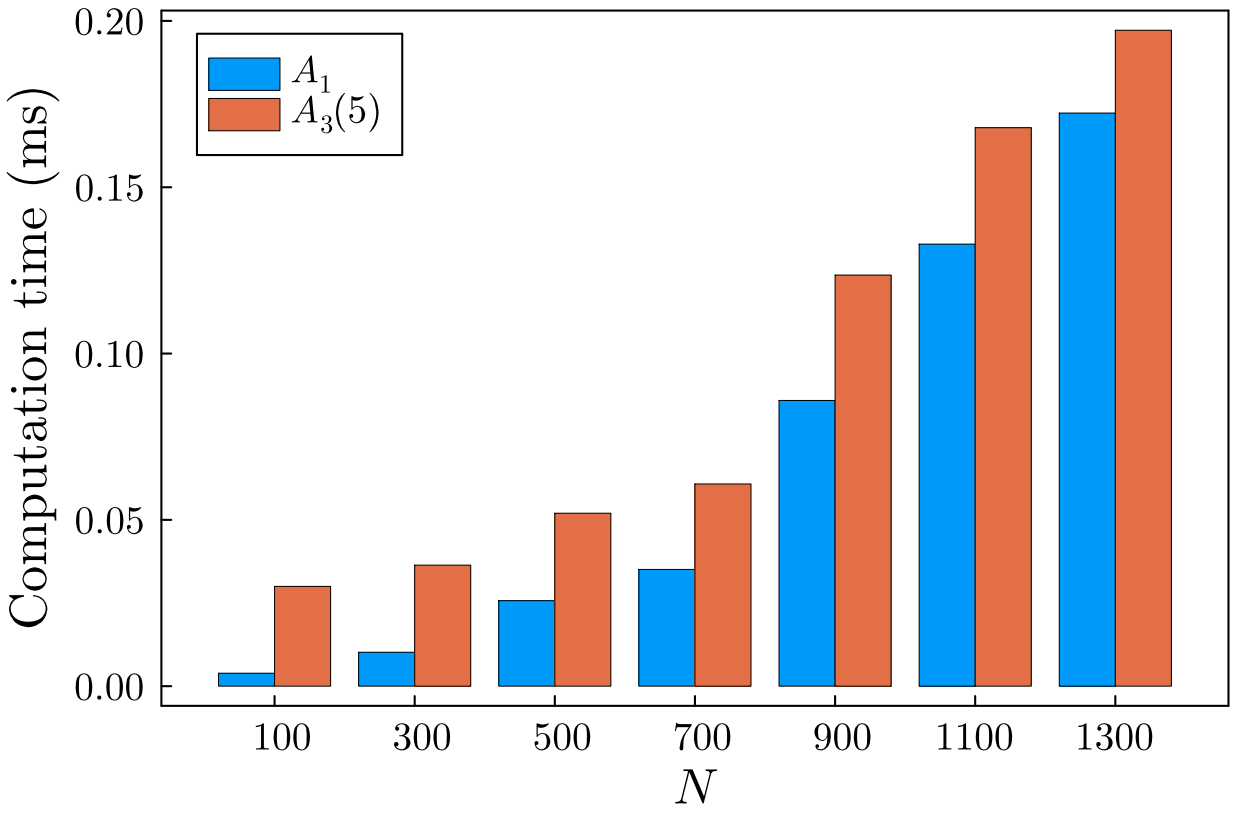

Figure 4.5: Computation time vs. N.

than their respective counterparts. This ordering would reverse whenever $L < M$. The perceptive reader will have noticed that we can easily estimate how the computation cost for the approximate filters would change as M increases by comparing the computation costs for $\boldsymbol{A}_1$ and $\boldsymbol{A}_3$ with those of $\boldsymbol{A}_2$ and $\boldsymbol{A}_4$. The difference between $\boldsymbol{A}_1$ and $\boldsymbol{A}_3(5)$ remains relatively small for all values of N, as illustrated more clearly in Figure 4.5. With additional order of approximation (meaning additional computation), $\boldsymbol{A}_3$ becomes more like $\boldsymbol{A}_1$. But recall that computing $\boldsymbol{A}_1$ involves an inversion of a matrix, albeit of a smaller size, whereas computing $\boldsymbol{A}_3$ does not involve inversion of any kind. This tradeoff is beneficial in many cases.

4.6 Conclusion

In this chapter, we derive four approximate Wiener filter formulas using a truncation technique based on the principal components of a composite covariance matrix. Our approximate filters do

not directly involve the inverse of the covariance matrix. This results in the stable performance of our filters even as the covariance matrix becomes increasingly ill-conditioned, as demonstrated by our results with real data. These results show deteriorating performance for the Wiener filter, demonstrating its numerical unreliability for covariance matrices with large condition numbers. We describe two methods with varying complexity for optimally trading off computation for accuracy using our filters. We also show that the approximate filter formulas converge to the Wiener filter in terms of asymptotic scaling laws. The numerical comparison of computation times demonstrates that the computation of our filters comes at a small extra cost compared to the Wiener filter and is even more efficient in some use cases.

Finally, it has not escaped our notice that our method of approximating Wiener filters in eigen-subspaces spanned by the joint principal components (i.e., those of the composite covariance matrix) suggests a comparison with reduced-rank Wiener filters in Krylov subspaces composed by $\boldsymbol{C_{YY}}$ and $\boldsymbol{C_{YX}}$ [87, 88]. Their performance relative to our filters, as well as the two well-conditioned approximate filters derived using the joint principal components in [90], warrants further investigation.

Bibliography

[1] Pranav U Damale, Edwin K P Chong, Sean L Hammond, and Ronald B Tjalkens. A low-cost, autonomous gait detection and estimation system for analyzing gait impairments in mice. *J. Healthc. Eng.*, 2021, 2021.

[2] Majid Hafezparast, Azlina Ahmad-Annuar, Nicholas W Wood, Sarah J Tabrizi, and Elizabeth M C Fisher. Mouse models for neurological disease. *Lancet Neurol.*, 1(4):215–224, August 2002.

[3] Nir Giladi, Fay B Horak, and Jeffrey M Hausdorff. Classification of gait disturbances: distinguishing between continuous and episodic changes. *Mov. Disord.*, 28(11):1469–1473, September 2013.

[4] D Michele Basso, Lesley C Fisher, Aileen J Anderson, Lyn B Jakeman, Dana M Mctigue, and Phillip G Popovich. Basso Mouse Scale for locomotion detects differences in recovery after spinal cord injury in five common mouse strains. *J. Neurotrauma*, 23(5):635–659, May 2006.

[5] Kathleen S Tatem, James L Quinn, Aditi Phadke, Qing Yu, Heather Gordish-Dressman, and Kanneboyina Nagaraju. Behavioral and locomotor measurements using an open field activity monitoring system for skeletal muscle diseases. *J. Vis. Exp.*, 91, 2014.

[6] Casey Harr. Design of a gait acquisition and analysis system for assessing the recovery of mice post-spinal cord injury. Master's thesis, University of Kentucky, Lexington, Kentucky, 2005.

[7] O Haji Maghsoudi, Annie Vahedipour, Benjamin Robertson, and Andrew Spence. Application of superpixels to segment several landmarks in running rodents. *Pattern Recognit. Image Anal.*, 28(3):468–482, July 2018.

[8] Jeffrey Wong and Prithvi K Shah. 3D kinematic gait analysis for preclinical studies in rodents. *JoVE*, 150:e59612, August 2019.

[9] Ivanna K Timotius, Sandra Moceri, Anne-Christine Plank, Johanna Habermeyer, Fabio Canneva, Jürgen Winkler, Jochen Klucken, Nicolas Casadei, Olaf Riess, Bjoern Eskofier, and Stephan von Hörsten. Silhouette-length-scaled gait parameters for motor functional analysis in mice and rats. *Eneuro*, 6(6), November 2019.

[10] Brittany Y Jacobs, Emily H Lakes, Alex J Reiter, Spencer P Lake, Trevor R Ham, Nic D Leipzig, Stacy L Porvasnik, Christine E Schmidt, Rebecca A Wachs, and Kyle D Allen. The open source GAITOR suite for rodent gait analysis. *Sci. Rep.*, 8(1):1–14, June 2018.

[11] Pranav U Damale. Design of a gait acquisition and analysis system for assessing the recovery in a classical murine model of Parkinson's disease. Master's thesis, Colorado State University, Fort Collins, Colorado, 2015.

[12] Briana R De Miranda, James A Miller, Ryan J Hansen, Paul J Lunghofer, Stephen Safe, Daniel L Gustafson, Dorothy Colagiovanni, and Ronald B Tjalkens. Neuroprotective efficacy and pharmacokinetic behavior of novel anti-inflammatory para-phenyl substituted diindolylmethanes in a mouse model of Parkinson's disease. *J. Pharmacol. Exp. Ther.*, 345(1):125–138, April 2013.

[13] Briana R De Miranda, Katriana A Popichak, Sean L Hammond, James A Miller, Stephen Safe, and Ronald B Tjalkens. Novel para-phenyl substituted diindolylmethanes protect against MPTP neurotoxicity and suppress glial activation in a mouse model of Parkinson's disease. *Toxicol. Sci.*, 143(2):360–373, November 2014.

[14] Briana R De Miranda, Katriana A Popichak, Sean L Hammond, Bryce A Jorgensen, Aaron T Phillips, Stephen Safe, and Ronald B Tjalkens. The Nurr1 activator 1, 1-bis (3'-indolyl)-1-(p-chlorophenyl) methane blocks inflammatory gene expression in BV-2 microglial cells by inhibiting nuclear factor κB. *Mol. Pharmacol.*, 87(6):1021–1034, June 2015.

[15] Sean L Hammond, Stephen Safe, and Ronald B Tjalkens. A novel synthetic activator of Nurr1 induces dopaminergic gene expression and protects against 6-hydroxydopamine neurotoxicity in vitro. *Neurosci. Lett.*, 607:83–89, October 2015.

[16] Sean L Hammond, Katriana A Popichak, Xi Li, Lindsay G Hunt, Evan H Richman, Pranav U Damale, Edwin K P Chong, Donald S Backos, Stephen Safe, and Ronald B Tjalkens. The Nurr1 ligand, 1, 1-bis (3'-indolyl)-1-(p-chlorophenyl) methane, modulates glial reactivity and is neuroprotective in MPTP-induced Parkinsonism. *J. Pharmacol. Exp. Ther.*, 365(3):636–651, June 2018.

[17] Lszl Havasi, Zoltn Szlavik, and Tams Sziranyi. Detection of gait characteristics for scene registration in video surveillance system. *IEEE Trans. Image Process.*, 16(2):503–510, January 2007.

[18] Teddy Ko. A survey on behavior analysis in video surveillance for homeland security applications. *37th IEEE Applied Imagery Pattern Recognition Workshop*, pages 1–8, October 2008.

[19] Peter K Larsen, Erik B Simonsen, and Niels Lynnerup. Gait analysis in forensic medicine. *J. Forensic Sci.*, 53(5):1149–1153, September 2008.

[20] Jay Prakash Gupta, Nishant Singh, Pushkar Dixit, Vijay Bhaskar Semwal, and Shiv Ram Dubey. Human activity recognition using gait pattern. *Int. J. Comput. Vis. Image Process.*, 3(3):31–53, July 2013.

[21] Yanzhi Ren, Yingying Chen, Mooi Choo Chuah, and Jie Yang. Smartphone based user verification leveraging gait recognition for mobile healthcare systems. *2013 IEEE International Conference on Sensing, Communications and Networking (SECON)*, pages 149–157, June 2013.

[22] Sumit Majumder, Tapas Mondal, and M Jamal Deen. A simple, low-cost and efficient gait analyzer for wearable healthcare applications. *IEEE Sens. J.*, 19(6):2320–2329, December 2018.

[23] Zhiming Lin, Zhiyi Wu, Binbin Zhang, Yi-Cheng Wang, Hengyu Guo, Guanlin Liu, Chaoyu Chen, Yuliang Chen, Jin Yang, and Zhong Lin Wang. A triboelectric nanogenerator-based smart insole for multifunctional gait monitoring. *Adv. Mater. Technol.*, 4(2):1800360, February 2019.

[24] David Cunado, Mark S Nixon, and John N Carter. Using gait as a biometric, via phase-weighted magnitude spectra. *International Conference on Audio-and Video-based Biometric Person Authentication*, pages 93–102, March 1997.

[25] Liang Wang, Huazhong Ning, Tieniu Tan, and Weiming Hu. Fusion of static and dynamic body biometrics for gait recognition. *IEEE Trans. Circuits Syst. Video Technol.*, 14(2):149–158, March 2004.

[26] Imed Bouchrika, Michaela Goffredo, John Carter, and Mark Nixon. On using gait in forensic biometrics. *J. Forensic Sci.*, 56(4):882–889, July 2011.

[27] Yu Zhong and Yunbin Deng. Sensor orientation invariant mobile gait biometrics. *IEEE International Joint Conference on Biometrics*, pages 1–8, September 2014.

[28] Ping S Huang, Chris J Harris, and Mark S Nixon. Human gait recognition in canonical space using temporal templates. *IEE P.-Vis. Image Sign.*, 146(2):93–100, August 1999.

[29] Liang Wang, Tieniu Tan, Huazhong Ning, and Weiming Hu. Silhouette analysis-based gait recognition for human identification. *IEEE PAMI*, 25(12):1505–1518, December 2003.

[30] Galina V Veres, Layla Gordon, John N Carter, and Mark S Nixon. What image information is important in silhouette-based gait recognition. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2:II–II, June 2004.

[31] Dong Xu, Shuicheng Yan, Dacheng Tao, Lei Zhang, Xuelong Li, and Hong-Jiang Zhang. Human gait recognition with matrix representation. *IEEE Trans. Circuits Syst. Video Technol.*, 16(7):896–903, July 2006.

[32] Toby HW Lam, King Hong Cheung, and James NK Liu. Gait flow image: A silhouette-based gait representation for human identification. *Pattern Recognit.*, 44(4):973–987, April 2011.

[33] Ahmad Jalal, Naeha Sarif, Jeong Tai Kim, and Tae-Seong Kim. Human activity recognition via recognized body parts of human depth silhouettes for residents monitoring services at smart home. *Indoor Built Environ.*, 22(1):271–279, February 2013.

[34] Maria Mahmood, Ahmad Jalal, and Kibum Kim. WHITE STAG model: Wise human interaction tracking and estimation (WHITE) using spatio-temporal and angular-geometric (STAG) descriptors. *Multimed. Tools. Appl.*, pages 1–32, December 2019.

[35] Kibum Kim, Ahmad Jalal, and Maria Mahmood. Vision-based human activity recognition system using depth silhouettes: A Smart home system for monitoring the residents. *J. Electr. Eng. Technol.*, 14(6):2567–2573, November 2019.

[36] Jani Mantyjarvi, Mikko Lindholm, Elena Vildjiounaite, S-M Makela, and HA Ailisto. Identifying users of portable devices from gait pattern with accelerometers. *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, 2:ii–973, March 2005.

[37] Liu Rong, Zhou Jianzhong, Liu Ming, and Hou Xiangfeng. A wearable acceleration sensor system for gait recognition. *2007 2nd IEEE Conference on Industrial Electronics and Applications*, pages 2654–2659, May 2007.

[38] Davrondzhon Gafurov and Einar Snekkenes. Gait recognition using wearable motion recording sensors. *EURASIP J. Adv. Signal Process.*, 2009:1–16, December 2009.

[39] Wei Wang, Alex X Liu, and Muhammad Shahzad. Gait recognition using wifi signals. *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 363–373, September 2016.

[40] Omid Dehzangi, Mojtaba Taherisadr, and Raghvendar ChangalVala. IMU-based gait recognition using convolutional neural networks and multi-sensor fusion. *Sensors*, 17(12), December 2017.

[41] Ahmad Jalal and Kibum Kim. Wearable inertial sensors for daily activity analysis based on adam optimization and the maximum entropy Markov model. *Entropy*, 22(5):579, May 2020.

[42] Majid Ali Khan Quaid and Ahmad Jalal. Wearable sensors based human behavioral pattern recognition using statistical features and reweighted genetic algorithm. *Multimed. Tools. Appl.*, 79(9):6061–6083, March 2020.

[43] Sheikh Badar ud din Tahir, Ahmad Jalal, and Mouazma Batool. Wearable sensors for activity analysis using smo-based random forest over smart home and sports datasets. *2020 3rd International Conference on Advancements in Computational Sciences (ICACS)*, pages 1–6, February 2020.

[44] Michael Otero. Application of a continuous wave radar for human gait recognition. *Signal Processing, Sensor Fusion, and Target Recognition XIV*, 5809:538–548, May 2005.

[45] Zhaojun Xue, Dong Ming, Wei Song, Baikun Wan, and Shijiu Jin. Infrared gait recognition based on wavelet transform and support vector machine. *Pattern Recognit.*, 43(8):2904–2910, August 2010.

[46] Brian DeCann, Arun Ross, and Jeremy Dawson. Investigating gait recognition in the short-wave infrared (swir) spectrum: dataset and challenges. *Biometric and Surveillance Technology for Human and Activity Identification X*, 8712:87120J, 2013.

[47] Tzutalin. Labelimg. https://github.com/tzutalin/labelImg, 2015.

[48] Tom Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[49] Marc Claesen, Jesse Davis, Frank De Smet, and Bart De Moor. Assessing binary classifiers using only positive and unlabeled data. *arXiv:1504.06837 [stat.ML]*, April 2015.

[50] Sean L Hammond, Collin M Bantle, Katriana A Popichak, Katie A Wright, Delaney Thompson, Catalina Forero, Kelly S Kirkley, Pranav U Damale, Edwin K P Chong, and Ronald B Tjalkens. NF-κB signaling in astrocytes modulates brain inflammation and neuronal injury following sequential exposure to manganese and mptp during development and aging. *Toxicol. Sci.*, 117(2):506–520, October 2020.

[51] K A Clarke and J Still. Gait analysis in the mouse. *Physiol. Behav.*, 66(5):723–729, July 1999.

[52] Dorien H Vrinten and Frank F T Hamers. 'CatWalk' automated quantitative gait analysis as a novel method to assess mechanical allodynia in the rat; a comparison with von frey testing. *Pain*, 102(1-2):203–209, 2003.

[53] Susann Hetze, Christine Römer, Carena Teufelhart, Andreas Meisel, and Odilo Engel. Gait analysis as a method for assessing neurological outcome in a mouse model of stroke. *J. Neurosci. Methods*, 206(1):7–14, April 2012.

[54] Claudia Pitzer, Rohini Kuner, and Anke Tappe-Theodor. Voluntary and evoked behavioral correlates in inflammatory pain conditions under different social housing conditions. *Pain Rep.*, 1(1), July 2016.

[55] Pranav U Damale, Edwin K P Chong, and Tian J Ma. Performance study of distance-weighting approach with loopy sum-product algorithm for multi-object tracking in clutter. *Sensors*, 21(7):2544, 2021.

[56] Yaakov Bar-Shalom and Edison Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, 1975.

[57] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[58] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *Trans. Inf. Theory*, 47(2):498–519, 2001.

[59] Xiao Chen, Yaan Li, Yuxing Li, Jing Yu, and Xiaohua Li. A novel probabilistic data association for target tracking in a cluttered environment. *Sensors*, 16(12):2180, 2016.

[60] Jason L Williams and Roslyn A Lau. Data association by loopy belief propagation. *2010 13th International Conference on Information Fusion*, pages 1–8, 2010.

[61] X Rong Li and Yaakov Bar-Shalom. Tracking in clutter with nearest neighbor filters: analysis and performance. *IEEE Trans. Aerosp. Electron. Syst.*, 32(3):995–1010, 1996.

[62] Thomas Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE J. Ocean. Eng.*, 8(3):173–184, 1983.

[63] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic data association revisited. *Proceedings of the IEEE international conference on computer vision*, pages 3047–3055, 2015.

[64] Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Trans. Aerosp. Electron. Syst.*, 19(1):5–18, 2004.

[65] JA Roecker and GL Phillis. Suboptimal joint probabilistic data association. *IEEE Trans. Aerosp. Electron. Syst.*, 29(2):510–517, 1993.

[66] Donald Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automat. Cont.*, 24(6):843–854, 1979.

[67] Yaakov Bar-Shalom, Peter K Willett, and Xin Tian. *Tracking and data fusion*, volume 11. YBS publishing Storrs, CT, USA:, 2011.

[68] Niclas Wiberg, Hans-Andrea Loeliger, and Ralf Kotter. Codes and iterative decoding on general graphs. *Eur. Trans. Telecomm.*, 6(5):513–525, 1995.

[69] Jason L Williams and Roslyn A Lau. Convergence of loopy belief propagation for data association. *2010 Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 175–180, 2010.

[70] Jason Williams and Roslyn Lau. Approximate evaluation of marginal association probabilities with belief propagation. *IEEE Trans. Aerosp. Electron. Syst.*, 50(4):2942–2959, 2014.

[71] Florian Meyer, Paolo Braca, Peter Willett, and Franz Hlawatsch. Scalable multitarget tracking using multiple sensors: A belief propagation approach. *2015 18th International Conference on Information Fusion (Fusion)*, pages 1778–1785, 2015.

[72] Florian Meyer, Paolo Braca, Peter Willett, and Franz Hlawatsch. Tracking an unknown number of targets using multiple sensors: A belief propagation method. *2016 19th International Conference on Information Fusion (FUSION)*, pages 719–726, 2016.

[73] Florian Meyer, Paolo Braca, Peter Willett, and Franz Hlawatsch. A scalable algorithm for tracking an unknown number of targets using multiple sensors. *IEEE Trans. Signal Process.*, 65(13):3478–3493, 2017.

[74] Florian Meyer, Thomas Kropfreiter, Jason L Williams, Roslyn Lau, Franz Hlawatsch, Paolo Braca, and Moe Z Win. Message passing algorithms for scalable multitarget tracking. *Proc. IEEE*, 106(2):221–259, 2018.

[75] Domenico Gaglione, Giovanni Soldi, Paolo Braca, Giovanni De Magistris, Florian Meyer, and Franz Hlawatsch. Classification-aided multitarget tracking using the sum-product algorithm. *IEEE Signal Process. Lett.*, 27:1710–1714, 2020.

[76] The MathWorks, Inc., Natick, MA. *MATLAB version 9.9.0.1467703 Release 2020b*.

[77] Abu Sajana Rahmathullah, Ángel F García-Fernández, and Lennart Svensson. Generalized optimal sub-pattern assignment metric. *2017 20th International Conference on Information Fusion (Fusion)*, pages 1–8, 2017.

[78] Pranav Ulhas Damale, Edwin K P Chong, and Louis L Scharf. Wiener filtering without covariance matrix inversion. In *Proc. 48th IEEE Int. Conf. Acoust. Speech Signal Process.*, Rhodes Island, GR, 2023, to be published.

[79] Pranav U Damale, Edwin K P Chong, and Louis L Scharf. Wiener filter approximations without covariance matrix inversion. *IEEE Open J. Signal Process. (OJ-SP)*, 4:364–374, 2023, to be published.

[80] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, Feb. 1970.

[81] Jemima M Tabeart, Sarah L Dance, Amos S Lawless, Nancy K Nichols, and Joanne A Waller. Improving the condition number of estimated covariance matrices. *Tellus A: Dynamic Meteorology and Oceanography*, 72(1):1–19, 2020.

[82] Louis L Scharf. *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison-Wesley, Reading, MA, 1991.

[83] J Scott Goldstein and Irving S Reed. Reduced-rank adaptive filtering. *IEEE Trans. Signal Process.*, 45(2):492–496, Feb. 1997.

[84] J Scott Goldstein, Irving S Reed, and Louis L Scharf. A multistage representation of the Wiener filter based on orthogonal projections. *IEEE Trans. Inf. Theory*, 44(7):2943–2959, Nov. 1998.

[85] Yingbo Hua, Maziar Nikpour, and Petre Stoica. Optimal reduced-rank estimation and filtering. *IEEE Trans. Signal Process.*, 49(3):457–469, Mar. 2001.

[86] Guido Dietl and Wolfgang Utschick. On reduced-rank approaches to matrix Wiener filters in MIMO systems. In *Proc. 3rd IEEE Int. Symp. Signal Process. Inf. Technol.*, pages 82–85, Darmstadt, Germany, Dec. 2003. IEEE.

[87] Hongya Ge, Louis L Scharf, and Magnus Lundberg. Reduced-rank multiuser detectors based on vector and matrix conjugate gradient Wiener filters. In *Proc. 5th IEEE Workshop Signal Process. Advances Wireless Commun.*, pages 189–193, Jul. 2004.

[88] Guido KE Dietl. Reduced-rank matrix Wiener filters in Krylov subspaces. In *Linear Estimation and Detection in Krylov Subspaces*, volume 1, chapter 4, pages 71–109. Springer, Berlin, Heidelberg, 2007.

[89] Louis L Scharf, Edwin KP Chong, Michael D Zoltowski, J Scott Goldstein, and Irving S Reed. Subspace expansion and the equivalence of conjugate direction and multistage Wiener filters. *IEEE Trans. Signal Process.*, 56(10):5013–5019, Oct. 2008.

[90] Edwin K P Chong. Well-conditioned linear minimum mean square error estimation. *IEEE Control Syst. Lett.*, 6:2431–2436, 2022.

[91] VXAZN Historical Price Data. Accessed: Nov. 6, 2022.

[92] Jelle Veraart, Dmitry S Novikov, Daan Christiaens, Benjamin Ades-Aron, Jan Sijbers, and Els Fieremans. Denoising of diffusion MRI using random matrix theory. *Neuroimage*, 142:394–406, Nov. 2016.

[93] RJ O'Dowd. The Wiener-Levinson algorithm and ill-conditioned normal equations. *Geophys. J. Int.*, 106(2):399–406, 1991.

[94] Albert Cohen, Mark A Davenport, and Dany Leviatan. On the stability and accuracy of least squares approximations. *Found. Comput. Math.*, 13:819–834, 2013.

[95] Xingsheng Deng, Liangbo Yin, Sichun Peng, and Meiqing Ding. An iterative algorithm for solving ill-conditioned linear least squares problems. *Geod. Geodyn.*, 6(6):453–459, 2015.

[96] Jehan Ghafuri, Hongbo Du, and Sabah Jassim. Sensitivity and stability of pretrained CNN filters. In *Proc. SPIE 11734, Mult. Image Expl. & Learning 2021*, volume 11734B, pages 79–89. SPIE, 2021.

[97] Steve Smale. On the efficiency of algorithms of analysis. *Bull. Amer. Soc.*, 13(2):87–121, Mar. 1985.

[98] Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM J. Matrix Analysis*, 9(4):543–560, Oct. 1988.

[99] Laurent Laloux, Pierre Cizeau, Jean-Philippe Bouchaud, and Marc Potters. Noise dressing of financial correlation matrices. *Phys. Rev. Lett.*, 83(7):1467, Aug. 1999.

[100] Laurent Laloux, Pierre Cizeau, Marc Potters, and Jean-Philippe Bouchaud. Random matrix theory and financial correlations. *Int. J. Theor. Appl. Finance*, 3(03):391–397, 2000.

[101] Gilles Zumbach. Empirical properties of large covariance matrices. *Quant. Finance*, 11(7):1091–1102, Mar. 2011.

[102] Jean-Philippe Bouchaud and Marc Potters. Financial applications of random matrix theory: A short review. In *The Oxford Handbook of Random Matrix Theory*, chapter 40, pages 824–850. Oxford University Press, Sept. 2015.

[103] Alex Bloemendal, Antti Knowles, Horng-Tzer Yau, and Jun Yin. On the principal components of sample covariance matrices. *Probab. Theory Relat. Fields*, 164(1):459–552, Feb. 2016.

[104] Joël Bun, Jean-Philippe Bouchaud, and Marc Potters. Cleaning large correlation matrices: Tools from random matrix theory. *Phys. Rep.*, 666:1–109, Jan. 2017.

[105] Mahsa Ghorbani and Edwin K P Chong. Stock price prediction using principal components. *PLOS One*, 15(3):e0230124, Mar. 2020.

[106] Edwin K P Chong and Stanislaw H Żak. *An Introduction to Optimization*. John Wiley & Sons, Hoboken, NJ, fourth edition, 2013.

[107] Atanu Saha, Burton G Malkiel, and Alex Rinaudo. Has the VIX index been manipulated? *J. Asset Manag.*, 20(1):1–14, 2019.

[108] Arman Hasanzadeh, Xi Liu, Nick Duffield, and Krishna R Narayanan. Piecewise stationary modeling of random processes over graphs with an application to traffic prediction. In *Proc. IEEE Int. Conf. Big Data (Big Data)*, pages 3779–3788, Los Angeles, CA, USA, 2019.

[109] David A Dickey and Wayne A Fuller. Distribution of the estimators for autoregressive time series with a unit root. *J. Amer. Statist. Assoc.*, 74(366a):427–431, 1979.

[110] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[111] Vladimir A Marčenko and Leonid Andreevich Pastur. Distribution of eigenvalues for some sets of random matrices. *Math. USSR Sb.*, 1(4):457–486, 1967.

[112] Jinho Baik and Jack W Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *J. Multivar. Anal.*, 97(6):1382–1408, July 2006.

[113] Jelle Veraart, Els Fieremans, and Dmitry S Novikov. Diffusion MRI noise mapping using random matrix theory. *Magn. Reson. Med.*, 76(5):1582–1593, Nov. 2016.

[114] Wenzhao Zhao, Yisong Lv, Qiegen Liu, and Binjie Qin. Detail-preserving image denoising via adaptive clustering and progressive PCA thresholding. *IEEE Access*, 6:6303–6315, Dec. 2017.

[115] Mark D Does, Jonas Lynge Olesen, Kevin D Harkins, Teresa Serradas-Duarte, Daniel F Gochberg, Sune N Jespersen, and Noam Shemesh. Evaluation of principal component

analysis image denoising on multi-exponential MRI relaxometry. *Magn. Reson. Med.*, 81(6):3503–3514, Feb. 2019.

[116] Iain M Johnstone. High dimensional statistical inference and random matrices, 2006.

[117] Jack W Silverstein. The smallest eigenvalue of a large dimensional Wishart matrix. *Ann. Probab.*, 13(4):1364–1368, Nov. 1985.

[118] Jack W Silverstein. Strong convergence of the empirical distribution of eigenvalues of large dimensional random matrices. *J. Multivar. Anal.*, 55(2):331–339, Nov. 1995.

[119] Jack W Silverstein and Zhi Dong Bai. On the empirical distribution of eigenvalues of a class of large dimensional random matrices. *J. Multivar. Anal.*, 54(2):175–192, Aug. 1995.

[120] Jack W Silverstein and Sang-Il Choi. Analysis of the limiting spectral distribution of large dimensional random matrices. *J. Multivar. Anal.*, 54(2):295–309, Aug. 1995.

[121] Iain M Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Ann. Statist.*, 29(2):295–327, Apr. 2001.

[122] Alexander Soshnikov. A note on universality of the distribution of the largest eigenvalues in certain sample covariance matrices. *J. Statist. Phys.*, 108:1033–1056, Sept. 2002.

[123] Vasiliki Plerou, Parameswaran Gopikrishnan, Bernd Rosenow, Luis A Nunes Amaral, Thomas Guhr, and H Eugene Stanley. Random matrix approach to cross correlations in financial data. *Phys. Rev. E*, 65(6):066126, June 2002.

[124] Sandrine Péché. *Universality of local eigenvalue statistics for random sample covariance matrices*. PhD thesis, Dept. Statist. Stoch. Model., Univ. Paris-Sud, Paris, FR, 2003.