Improving Detection of Dim Targets:
Optimization of a Moment-based Detection Algorithm

DISSERTATION

Shannon R. Young, Capt, USAF

AFIT-ENP-DS-18-D-010

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENP-DS-18-D-010

IMPROVING DETECTION OF DIM TARGETS:
OPTIMIZATION OF A MOMENT-BASED DETECTION ALGORITHM

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Shannon R. Young, B.S., M.S.

Capt, USAF

December 2018

AFIT-ENP-DS-18-D-010

IMPROVING DETECTION OF DIM TARGETS:

OPTIMIZATION OF A MOMENT-BASED DETECTION ALGORITHM

DISSERTATION

Shannon R. Young, B.S., M.S.
Capt, USAF

Committee Membership:

Dr. Kevin C. Gross
Chair

Dr. Bryan J. Steward
Member

Dr. Christine Schubert Kabban
Member

Dr. Juan R. Vasquez
Member

ADEDEJI B. BADIRU, PhD
Dean, Graduate School of Engineering and Management

Abstract

Wide area motion imagery (WAMI) sensor technology is advancing rapidly. Increases in frame rates, detector array sizes, and communications bandwidth have led to a dramatic increase in the volume of data that can be acquired and exploited. However, a commensurate increase in analytical manpower has not accompanied the WAMI data growth, leaving some of it underutilized. This creates a need for fast, automated, and robust methods for detecting signals of interest. The most difficult targets to detect are unresolved (i.e., sub-pixel in size), dim (i.e., low signal-to-noise (SNR) ratio), and moving. Current tracking methods fall into two categories: detect-before-track (DBT) and track-before-detect (TBD). The DBT methods apply a threshold to reduce the quantity of data to be processed, making real time implementation practical for high-SNR targets (SNR $>$ 10). However, detecting low-SNR targets (2 $<$ SNR $<$ 5) requires the acceptance of high false alarm rates. TBD methods compare temporal and spatial information simultaneously with hypothesized target tracks to make detection of low SNR targets possible. Unfortunately, TBD methods incur substantial computational costs, limiting their use in most real-time detection applications. The performance of both DBT and TBD algorithms depend on many factors associated with target motion, the background environment, and sensor capabilities and configuration. Thus, vast amounts of WAMI truth data spanning sensor, background, and target parameter spaces are required for statistically meaningful performance evaluations of DBT / TBD algorithms. Unfortunately, it is prohibitively costly to collect real truth data spanning the variable space, and many data simulation efforts have ignored important factors which strongly affect an algorithm's detection performance.

This research effort addresses the DBT-TBD performance gap for low-SNR targets, aiming to achieve TBD-like performance with DBT-like computational costs in order to maintain real-time applicability, and its development is target and sensor agnostic,

ensuring wide applicability. However, to ensure realistic characterization of detection performance, this research first focused on the improvement and validation of AFIT Sensor and Scene Emulation Tool (ASSET). ASSET is an electro-optical sensor, target, and scene generation tool which employs a physics-based imaging-chain model and includes realistic sensor artifacts. ASSET simulations were validated against: (1) static visible and infrared images collected by Himawari-8, a recently-launched Japanese weather satellite, to assess various stationary performance elements including spectro-radiometric sensor response, optical blurring, and pixel-to-geospatial coordinate transformations; and (2) laboratory measurements using a visible and infrared camera to further validate stationary performance metrics, and also assess ASSET's temporal simulation aspects, including sensor noise and platform jitter.

Having ensured ASSET can generate sufficiently realistic scenes with sensing artifacts that make dim-target detection difficult under realistic conditions, the focus turned to the detection of low-SNR targets by mathematically undergirding, thoroughly testing, and expanding Higher-Order Moments Anomaly Detection (HOMAD), a method that exploits both spatial and temporal information but is still computationally efficient and massively parallelizable. Improvements include fast background suppression, feature-based filtering for false-alarm suppression, and algorithm parameterization in terms of sensor and target phenomenology and these improvements are packaged into a robust Moments-Based Detection (MBD) code. The expected moment-based detection (MBD) algorithm performance for a physical sensor was derived and validated with a second statistical approach. Optimal MBD parameters for detection performance were determined as a function of a target's physical characteristics, such as size, brightness, and velocity. MBD was shown to detect dim targets with performance comparable to leading TBD methods in less than 1000^{th} of the computational time, enabling near real-time detection at temporal signal-to-clutter-plus-noise ratios (SCNR's) down to 0.5 under realistic conditions.

Table of Contents

Page

Page

List of Figures

Nomenclature

Abbreviations

ASSET	AFIT Sensor and Scene Emulation Tool
DBT	Detect-before-track
FOV	Field of view
GSD	Ground Sample Distance
H-PMHT	Histogram Probabilistic Multiple Hypothesis Tracking
HOMAD	Higher order moments anomaly detector
MBD	Moment based detection
PDF	Probability density function
PSF	Point spread function
SNR	Signal-to-noise ratio
TBD	Track-before-detect
WAMI	Wide area motion imagery

Symbols

α_Z	Significance level used to set threshold in z-scored moment cubes
β	Target theoretical signal to noise ratio
$\Delta\langle M^n \rangle$	Expected difference between background and target moment windows
$\hat{\mu}_{B_{r,c}}$	Estimate of the background mean of a single spatial pixel computed with W_S frames
μ_B	Mean of the background pixels in a data cube
μ_T	Mean of the distribution of target pixels in a data cube
μ_ε	Mean of the noise in a data cube after background removal
σ_B	Global standard deviation of a data cube
$SCNR_T$	Signal to clutter plus noise ratio
SNR_{MX}	Peak maximum global SNR

$SNR_{S_{TOT}}$	Total local spatial SNR
ε	Time dependent noise
C	Total number of columns in a data cube
c	Column index in a data cube
D	Digital number, a single value in a data cube
d	Dimension of the sliding voxel used for filtering
FA	False alarm percentage
K	Total number of frames in a data cube
k	Temporal frame index in a data cube
L_s	Length of a streak in a single frame of a moment cube
L_t	Length of a temporally connected streak in a moment cube
M_B^n	Moment value of a window containing only background
M_T^n	Moment value of a window containing background and target
$M_{r,c}^n$	Moment value for a single pixel computed over a temporal window W about $\hat{\mu}_{B_{r,c}}$
N	Total number of moments computed in the MBD algorithm
n	Moment order
P	Total numbe of temporal moment frames
p	Temporal moment frame index in a moment cube
P_D	Probability of detection
P_{FA}	False alarm probability
R	Dwell time-the number of frames a target spends in a single spatial pixel
R	Total number of rows in a data cube
r	Row index in a data cube
S	Temporal step size for sliding window calculations
S_{MX}	Peak maximum signal for a target (pixel value when the target is centered)

S_{PK}	Peak target signal (value of the pixel containing a target)
S_{TOT}	Total target signal integrated over the PSF
SNR_S	Peak local spatial SNR
SNR_T	Local temporal signal to noise ratio
T_V	Secondary filtering threshold, the minimum required number of detects in a voxel
W	Temporal signal window size
W_S	Temporal background window size
W_{opt}	Optimal window size to maximize detection

For my parents, who constantly encouraged my often destructive curiosity and fostered a lifelong desire to learn

Acknowledgements

I would like to thank my research advisor Dr. Kevin Gross for his time, energy and insights throughout this research effort, helping me pull myself out of bottomless rabbit holes and move forward whenever I seemed to hit a wall, and Dr. Bryan Steward, both for his help developing much of the code necessary for this research and for his constant support and guidance throughout the entire effort. I would also like to thank my family who have inspired and motivated me throughout this process, especially my sister, who made the time to read and edit for me. Finally I would like to thank all the teachers I have had in my life, especially my elementary school teachers Mrs. Beardsley and Mrs. Nelson who sparked my interest in science and the physical world and set me on the path I am on today.

Shannon R. Young

IMPROVING DETECTION OF DIM TARGETS:

OPTIMIZATION OF A MOMENT-BASED DETECTION ALGORITHM

I. Introduction

1.1 Problem History and Motivation

The field of remote sensing is rapidly expanding to meet commercial, defense, and intelligence community demands. Many new wide area (100-10,000km) imaging sensors have been developed in the last few decades, and many more are currently under development. Imagery is constantly collected by a large variety of platforms with varying resolutions, including both space and ground based telescopes, airborne vehicles, and satellites. Within this imagery are many different objects of interest, varying in size and brightness. However, the increasing number of sensors, along with detector array sizes, frame rates, and data transfer rates has not been accompanied by an equivalent increase in the manpower necessary to manually process it. This mismatch between manpower and data volume results in large quantities of data that are underutilized. To prevent this mismatch from causing a significant loss in information, algorithms that are capable of reliably and automatically detecting and tracking objects of interest are required.

In the cases where the imager is of high quality with a high resolution, and objects of interest have high signal to noise ratios (SNRs) or are large in size, the problem of detecting and tracking is often not challenging due to the simple fact that the target of interest is clearly discernible in each frame of data. For these targets, automated detection and tracking methods already exist. However, as the resolution of the camera,

size of the target, or brightness of the target decreases, it becomes increasingly difficult to detect the target. These small, low SNR targets, or hard targets, are present in many fields. Examples include satellites in ground based telescope imagery in the space situational awareness field, and small vehicles or people in low cost unmanned aerial vehicle (UAV) imagery, as might be implemented for border patrol or other large scale surveillance operations. In order to detect hard targets such as these, methods that exploit the information available in many frames of data simultaneously are needed and are therefore the focus of many current research efforts. While there are methods that currently do this, due to the megapixel size and multi-detector nature of today's sensors, these methods are typically too computationally complex, and thus slow, to be of use in a near real-time setting. Additionally, the performance of detection and tracking algorithms will depend greatly on elements such as the sensor used to collect the data, the environment, and the target characteristics, requiring a large volume of relevant data to fully test any algorithm. This research seeks to contribute a solution to the problem of detecting the challenging subset of dim, unresolved targets through development of the Moments-based Detection (MBD) algorithm, while the development and validation of the AFIT Sensor and Scene Emulation Tool (ASSET) provides a method of acquiring the large volumes of data needed for thorough algorithm testing.

1.2 Overview

The motivations behind research in the realm of target detection and tracking and sensor and scene emulation have been discussed. The remainder of this document can be summarized as follows: Chapter II discusses the physics involved in collecting data using an optical detection system, the different approaches to the detection and tracking problem, and the common methods of testing such algorithms. In Chapter III, the development and validation of the sensor emulator, ASSET, used to test and

compare the algorithms researched is presented. Chapter IV discusses the theory behind the MBD algorithm, the algorithm itself, and the optimal algorithm parameters given a set of target and data characteristics. Chapter V compares the performance of the MBD algorithm to existing methods, and Chapter VI provides conclusions and recommendations for future work. In Appendix A, an alternative derivation is provided for a key result presented in Chapter IV. Finally, the final Matlab code developed in this research effort is presented in Appendix B.

II. Background

2.1 Physics of Imaging

The goal of any remote sensing effort is to gather information from a distance. To accomplish this, many different sensors exist for collecting electromagnetic radiation depending on the wavelengths of interest and the applications involved. This research focuses on one subset of these sensors, staring optical detectors, but does not focus on any specific wavebands in an effort to keep the results applicable across a large range of platforms.

2.1.1 Problem Scenario.

Figure 1 depicts the remote sensing scenario in this research. The remote sensing platform is staring at a scene containing primarily background features (water, ice, forest, desert, etc.) as well as some targets of interest, represented here by the tiny dragon breathing a fireball.

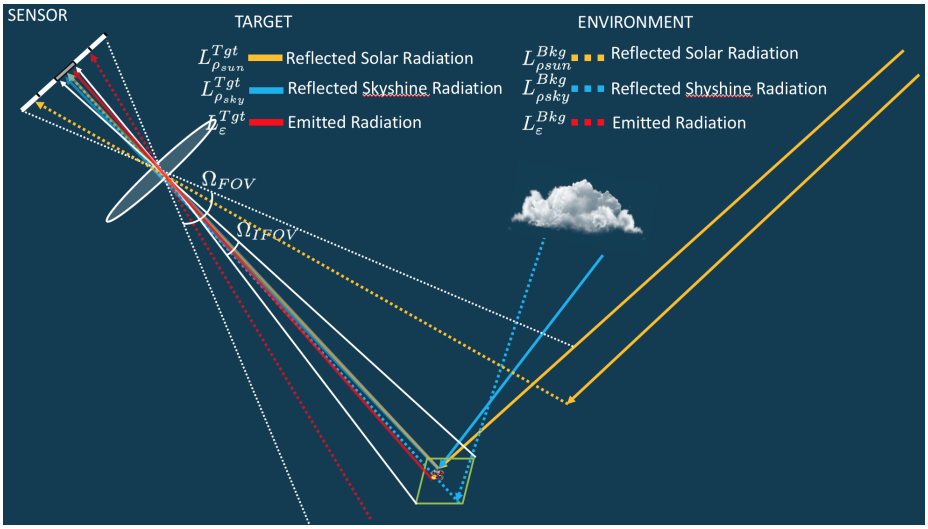


Figure 1. The remote sensing scenario in this research. The green region represents a single spatial pixel while the multi-colored arrows represent various types of radiation.

The sensor collects radiation from all sources within its field of view (FOV) from both the background and targets over the integration time, Δt, and outputs a single digital number for each pixel. While some sensor geometries will produce data containing targets that are multiple pixels in extent, the scenario depicted here and focused on in this research is the sub-pixel, unresolved target case. In this scenario, each pixel will image either solely background, or background plus a target. It is also assumed that the target is emissive and has a signal larger than the background. The *detection problem* is then the problem of determining which pixels contain only background, and which contain target in addition to background. To solve this problem, it is useful to understand how the digital number output from the sensor relates to the original scene that it is replicating.

There are a number of details to consider when looking at this scene to digital output mapping, but generally the problem can be broken down into two parts: propagation to the detector, and the conversion of the signal at the detector to a digital count. The first part depends on the radiance from all sources L, within the field of view of a pixel Ω_{IFOV}, the transmission of the atmosphere τ_{ATM}, the transmission of the optics τ_{OPT}, and the the area of the sensor's aperture A_{OPT}. The second part depends on the quantum efficiency of the detector, η, and the integration time. Putting all these terms together and integrating over the bandwidth $\Delta\lambda$ and integration time results in a signal in terms of total electrons that can be converted to a digital number,

$$D = A/D\left\{\int_{\Delta t}\int_{\Delta\lambda} \tau_{atm}(\lambda)L(\lambda,t)\Omega_{IFOV}A_{OPT}\tau_{OPT}(\lambda)\frac{\eta(\lambda)}{hc/\lambda}d\lambda d\,t\right\}, \tag{1}$$

where A/D represents the analog to digital signal conversion based on the total number of bits. That is, if there are 12 bits, the range of digital numbers, or *counts* would be from zero to $2^{12}-1 = 4095$, with a value of 4095 representing saturation and zero representing no signal. In a perfect system, the only source of electrons would be the incident

photons from the scene, contained within $L(\lambda, t)$, and the number of counts would be proportional to L. Additionally, for a perfect system that is diffraction limited, the energy on the detector could be computed by using an Airy function to represent the point spread function. However, real optical systems suffer from a number of factors that degrade their performance and as a result, have an impact on the detection problem.

2.1.2 Noise and Image Artifact Sources.

Section 2.1.1 discussed the problem of imaging a scene with a staring sensor. While ideally this process would produce an exact representation of the scene and targets, there are a number of noise sources and aspects of the imaging process that corrupt the ideal image, making it more challenging to extract useful information from the data. Photodetectors rely on a detector material suitable for absorption of energy transmitted within a narrow band. For photovoltaic detectors, incident photons are converted to a photogenerated current $i_g = \eta \phi_q q$ where η is the quantum efficiency, ϕ_q is the incident photon flux, and q is the elementary charge. Analog current is linearly converted to a digital number D recorded at each pixel in the image plane. Therefore for ideal detectors, $D \propto \phi_q$. The proportionality constant between D and ϕ_q will depend on the specific characteristics of the detector collecting the radiation, but ϕ_q is not the only source of current in the detector system.

In addition to the photogenerated current, current due to noise sources both internal to the detector and due to the electronics and environment are present. The potential internal detector-generated sources of noise in an optical detection system include Johnson, shot, generation recombination, one over frequency, temperature fluctuation, microphonics and popcorn noise; while the external sources include noise photon flux, and the post-detector electronics involved in converting current to the output signal. While all of these noise sources may be present, some will dominate, and generally only

three or four will be of concern. Here, only an understanding of the noise statistics for the primary noise sources is necessary, but a detailed discussion of these noise sources can be found in [21]. For the sensors of interest to the department of defense and intelligence communities (DOD/IC), the typically dominant internal noise sources are *Johnson noise* (also known as thermal noise), which is a result of the thermal motion of charges, and *shot noise*, which is due to discrete photoelectron generation and is associated with DC current flow across a potential barrier (namely the pn junction in photovoltaic detectors). The dominant external sources are *quantization noise*, which is due to the conversion from analog signal to a digital output and *photon noise*, which is the result of fluctuations in the incident signal and background radiation [14]. For quantization and photon noise, the noise distribution can be approximated with a Gaussian distribution due to the large numbers involved (e.g. current carriers, photons) via the central limit theorem [21, 38].

In addition to noise current, the individual pixels of real imagers will always have slightly different characteristics and therefore responses, producing non-uniformities in the final image that do not trace back to the scene being imaged. Generally, non-uniformity corrections (NUCs) and removal of bad pixels (e.g. pixels with virtually no response to incident radiation, saturation regardless of incident radiation, or other anomalous behavior) is performed prior to any further image analysis. Most detection and tracking methods assume such corrections have been performed. The details of these corrections are beyond the scope of this research, but an understanding of the concept of NUCs will aid in recognizing the advantage of a detection method that is insensitive to such non-uniformities.

Another factor to consider is the point spread function (PSF). The PSF describes how radiation from a point source is physically distributed on the focal plane and is a property of the optical system. For an ideal imaging system that is diffraction limited,

the PSF is an Airy disc, but in reality few systems are diffraction limited. For most well designed imagers the PSF is roughly Gaussian and as such, many detection approaches simplify the problem by using a Gaussian shape for the PSF so that targets have a 2-D Gaussian profile as in [2, 5, 8, 9]. Figure 2 shows an example of a Gaussian PSF. The sensor model described in Chapter III and used throughout this research has the option of loading in a specific PSF, but because the work here is meant to be platform agnostic, a simple 2D Gaussian profile is used.

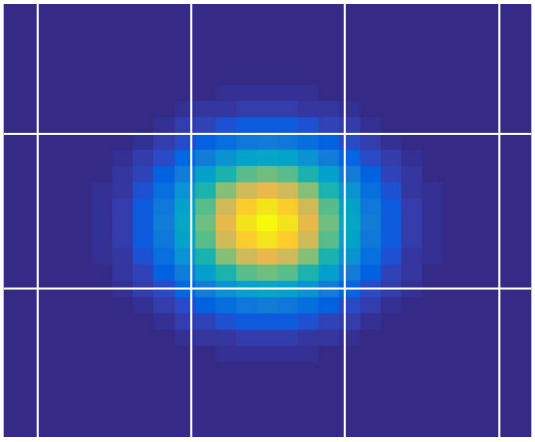


Figure 2. An example of the point spread function of an imager.

Because the energy collected by an imager is first blurred by the PSF and then sampled by the pixels in the detector, even when a target is centered in the field of view of a pixel, its energy will generally be spread across multiple pixels, requiring integration over the PSF to collect all the energy from any given target. This is an important effect to understand when discussing the difficulty of detecting various targets and will be discussed further in Section 2.4.

Finally, the sensor inherent noise and non-uniformity discussed above are not the only factors altering the final digital image; environmental and platform motion effects also contribute to sensor artifacts. For a completely stationary sensor, every frame would image the same portion of the scene in the same pixel as the previous frame; however, small sensor motions known as jitter can cause an object to appear in an adjacent

pixel in consecutive frames. For areas in the scene where there is a large contrast, such as coast lines or cloud edges, this results in clutter, or *clutter noise*. Similarly, sensor drift and yaw also cause objects to appear in different spatial pixels over time. For algorithms that use a large number of temporal frames of data, these are important effects to consider.

2.2 Approaches to the Detection and Tracking Problem

The output of an imager as described in Section 2.1 is a three dimensional array often referred to as an image stack, frame stack, or a data cube. Each pixel in the data cube has a digital number value $D_k(r,c)$ that is proportional to the number of photons collected by that pixel. Here, r and c represent the pixel row and column and k represents the frame number. Each pixel contains structured background signal represented by $B_k(r,c)$, noise represented by $\varepsilon_k(r,c)$ that is randomly varying in time, and potentially a target of interest represented by $T_k(r,c)$. The equation

$$D_k(r,c) = B_k(r,c) + T_k(r,c) + \varepsilon_k(r,c) \tag{2}$$

describes every pixel in such a frame stack. A target is defined here as a moving object that is not part of the background scene. For example, in the case of the Space Based Infrared System (SBIRS) whose mission includes global strategic missile warning, targets are certain classes of missiles [1]. Targets can be grouped into those that are resolved (target is observed in multiple pixel field of views) or unresolved (target completely contained within a single pixel's field of view). This research focuses on unresolved targets.

2.2.1 Detect-Before-Track.

There are two common paradigms for the detection and tracking problem. The first (and simplest) involves using the information contained in either a single frame k or multiple frames combined into a single frame to decide whether to declare a detection. This is generally referred to as the Detect-Before-Track (DBT) approach and is shown in Figure 3, where a blue square represents the data at a particular stage and a rounded green rectangle represents an operation on the data.

Figure 3. DBT process for detecting moving targets

The input of the process is the data cube described in the previous section. The image processing step in the DBT scheme is generally a two part process. The first part involves applying a NUC and removing bad pixels as discussed in Section 2.1.2 while the second is known as background suppression and involves removing the structured background from the frame stack so that ideally only the temporally and spatially varying noise and targets are left in the data. Techniques for background suppression have been a popular research topic for decades and as a result a large number of methods are available. All involve a prediction of the background derived from the data either spatially or temporally. Simple methods involve using spatially or temporally neighboring pixels to create the background prediction as in [24, 10, 29, 25, 26, 27] while more advanced methods involve performing a low-rank and sparse (LRS) decomposition using a method such as Robust Principle Components Analysis (RPCA) [32].

In general, for every frame k, the background estimation method produces a corresponding frame containing only background information ($B_k(r,c)$). The next step involves simply subtracting the background estimation frames from the original frame

stack to produce a new image stack containing only targets and noise, as shown in Figure 4.

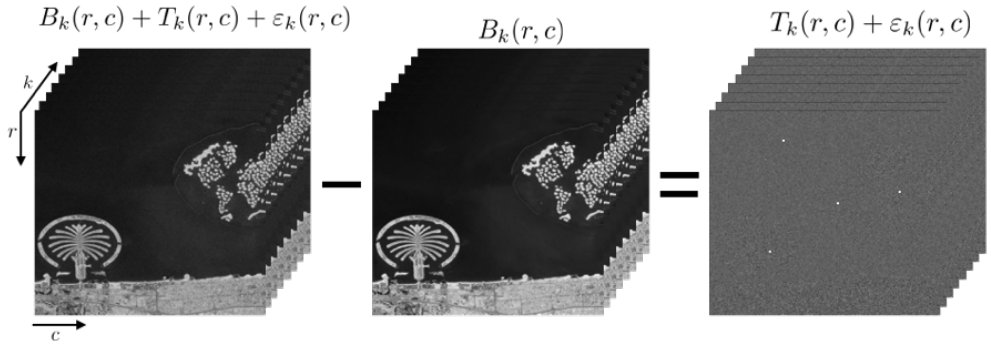


Figure 4. The process of removing the structured background from an image where the background estimation (center) is subtracted from the raw data (left) to produce the final image containing only noise and targets (right)

At this point it is the goal of the detection process to determine which pixels in the background suppressed data cube contain targets and which are simply noise. As with the background suppression step, there are a number of different ways to perform the detection step, but the basic premise is the same for most of them. First a threshold is chosen based on the desired balance of true detections and false alarms and a chosen metric, then all pixels exceeding that threshold are given a value of one and those below it are given a value of zero. The result is the candidate detections cube, a binary cube with the same dimensions as the raw data cube, but with a value of one for a possible target and zero elsewhere. Variations in detection approaches enter with respect to how the threshold is determined. The simplest example is to base the threshold on the overall noise (represented by the standard deviation) in the data cube, but additional methods include using a probability distribution to set a threshold, as in [4] or the use of a sliding window structure as in [39].

The candidate detections cube is then passed through a filtering process. The goal of this step is to reduce the number of false alarms by eliminating candidate detections that are likely to be false. For example, one such type of filtering is spatial filtering.

If the targets are known to be multiple pixels in extent, then it follows that isolated pixels are false detections and can be removed. The result of the filtering step (a more accurate candidate detections cube) is then fed into a tracking algorithm to combine the individual detections into groups of candidate detections corresponding to a single target, known as tracks.

These tracking algorithms all follow the same basic iterative two step approach with varying degrees of complexity. Starting with the first temporal frame of data, each candidate detection or *measurement* is used to predict measurements in subsequent frames. This is known as state estimation and usually involves using equations of motion (in terms of position, velocity and acceleration). After the prediction step, the measurements in future frames are compared to the predictions, and if they fall within some acceptable region, they are then considered potentially part of that particular track. This is illustrated with track A in Figure 5. The circle around frame 4 (green) indicates the acceptable region. Clearly only one of the frame 5 detections falls within this acceptable region and so it is considered the next point in the track. One of the most well known approaches to state estimation involves using the Kalman filter of [19].

The second step involves determining which measurements belong to which tracks when multiple candidates fall within a single predicted region. These *data association* methods fall into two categories, non-Bayesian and Bayesian. The non-Bayesian approaches involve maximizing a cost function to determine data associations while the Bayesian methods use the measurements that are already part of a track to estimate probabilities for each potential new track point. For example, a simple non-Bayesian method involves using the nearest neighbor approach. This means that when deciding between multiple measurements, the closest measurement to the previous time step is chosen (as shown in track D of Figure 5). This process of state estimation and data association is repeated until all measurements have either been grouped into tracks

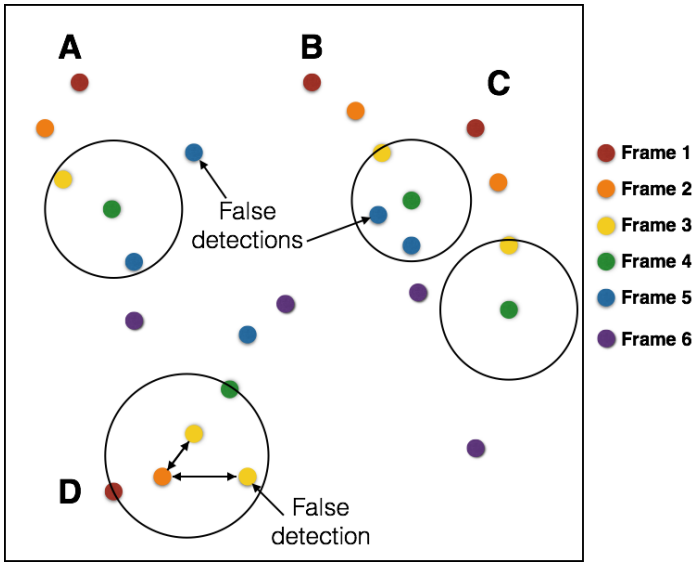


Figure 5. Illustration of the data association and state prediction steps in a tracking algorithm.

or discarded as false. Potential challenges for tracking algorithms include missed detections (as illustrated in track C of Figure 5) as well as false detections that trick the algorithm into choosing the wrong path, as the frame 5 detections of track B in Figure 5 would do for the nearest neighbor approach.

The Detect Before Track scheme works well in situations where the targets of interest are bright with high signal to noise (SNR) ratios, but the decision to threshold based on an individual frame of data presents a problem when signal from targets of interest are barely discernible above the noise floor. This problem is demonstrated in Figure 6: the top row shows a sequence of frames in which the target is moving and bright. The bottom row is a similar sequence of frames but with a target that is only slightly above the noise floor and (as a result) is difficult to discern in a single frame of data.

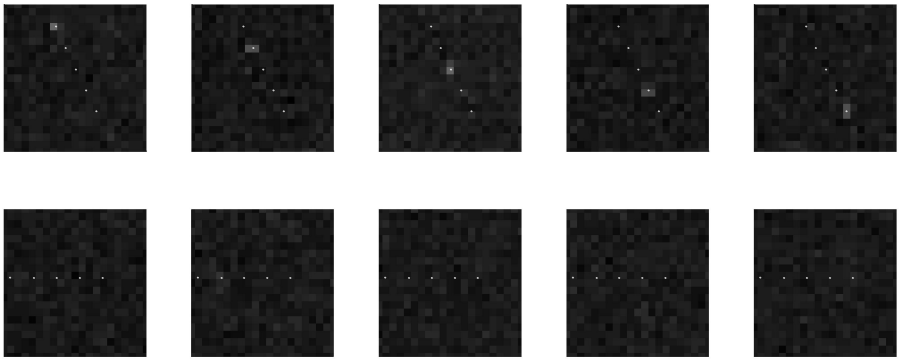


Figure 6. A target with a maximum SNR of 10 (top) and a target of SNR 2 (bottom) moving along the true path indicated by the white dots

In this latter situation it is necessary to lower the threshold value significantly to avoid discarding potential target pixels. As a result, the number of candidate detections passed to the tracking stage is very high, causing an increase in the number of false alarms as well as an increased burden on the tracker. Figure 7 illustrates this problem using a simulated data set containing eighteen targets with average SNRs ranging from 2.5 to 50. The threshold was set at 4σ on the left and 3σ on the right where σ is the standard deviation of the whole data cube and is a simplistic representation of the noise in the scene. While three times the noise level seems like a fairly high threshold, it is clear that there are a large number of false alarms present. This is due to the large number of pixels. For a normal distribution, 99.7% of the data points are within $\pm 3\sigma$ of the mean, which means there is an expected 0.3% false alarm probability. For an array that is 500×500, this means there will be 750 false detections. When the threshold is lowered to detect dim targets, these false detections can overwhelm the tracking algorithm. As a result, detection of low SNR targets with the DBT approach is generally not feasible. In response to this problem, many algorithms that delay the detection decision through partial target tracking across multiple frames and pixels, have been developed and as a body are termed Track-Before-Detect (TBD) methods.

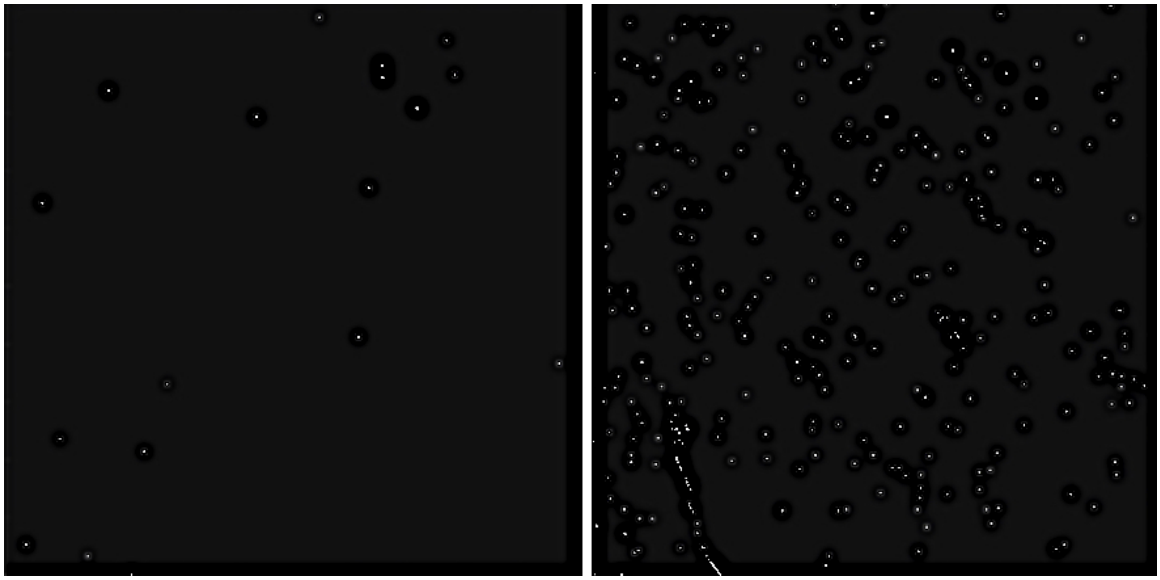


Figure 7. Single frame of detections with a 4σ (left) and 3σ (right) threshold

2.2.2 Track-Before-Detect.

TBD methods leverage the information contained in multiple frames of data as well as spatial information simultaneously, allowing them to perform better for low SNR targets. Figure 8 shows the general processing chain for TBD methods.

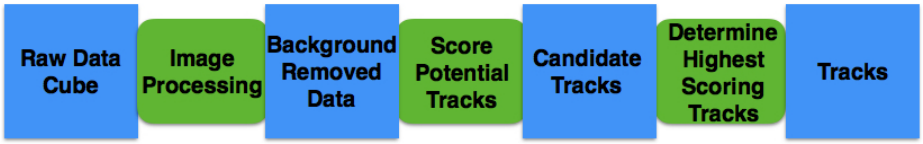


Figure 8. TBD process for detecting moving targets

The principle is the same as that of averaging images to reduce noise, but with the added complication that the target's path must be predicted for each additional frame. Just as with the DBT approach, the first step involves processing the image to remove non-uniformities, bad pixels and the structured background. But instead of thresholding the data at this point, potential target tracks are found and scored using various methods. Examples include Bayesian methods like that of [36], which accumulate the probability from all candidate tracks and determine the most likely track;

and maximum likelihood (ML) methods like that of [6] , which select the single best track at each time step based on a metric. Figure 9 is a notional diagram to illustrate these approaches. The five tracks shown represent the potential paths a target could have taken based on a set of assumptions such as the maximum velocity and acceleration of the target. For the Bayesian approach, the probability that each of the final five tracks is correct is computed and the final track is the one with the highest probability. For the maximum likelihood approach, each step in time is treated separately and the highest scoring trajectory at each step is kept. For example, in Figure 9 if path 2-5 is the most likely of the three paths from frame two to three, then at the next frame, only paths 5-8 and 5-10 are compared and whichever scores higher is kept. This process is repeated over the entire data cube and tracks that score above a set probability threshold (or metric threshold for ML methods) comprise the final output.

In this illustration the potential paths over four frames were limited to 5 for ease of visualization, but in reality, without prior knowledge of the target's motion, the number of potential paths may be as large as $(R \times C)^K$. For our simple example, there would really be $(5 \times 5)^4 \approx 390,625$ possible paths in a data cube with five rows, five columns and four frames. What this means is that prior knowledge of the target's motion is necessary to reduce computation time to a feasible level, but even then the number of computations (at least one for each possible path) is extremely high and thus the resulting TBD algorithms are computationally expensive. In a comparison of four TBD methods, it was found that even the fastest implementation of the four (the histogram probabilistic multiple hypothesis tracking (H-PMHT) method of [37]) took 42 seconds to process a $20 \times 20 \times 20$ data cube [12]. Without considering the size of the data cube, this is a reasonable computation time, but with today's megapixel array sizes, this quickly becomes intractable. To put this into perspective, first note that computer clock speeds have largely plateaued since 2004 [16]. Let's also assume that the H-PMHT method

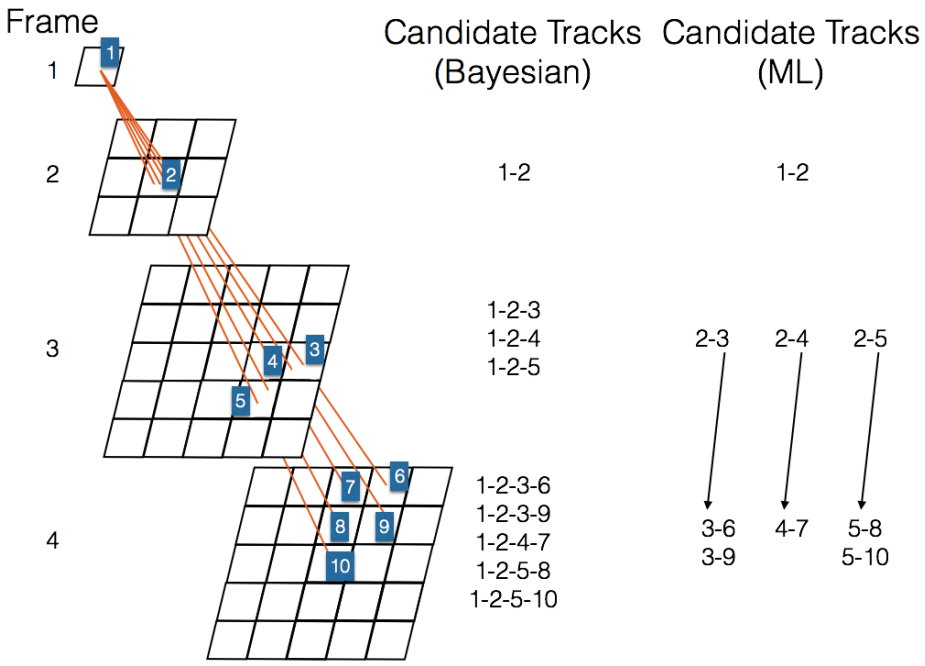


Figure 9. Illustration of the Bayesian and ML track-before-detect approaches

scales linearly with the number of pixels (a poor assumption but the most generous one we can make). For a large data cube with dimensions $1024 \times 1024 \times 500$, the total computation time would be over 31 days, clearly making real-time implementation impossible. As a result, new methods that combine the speed of DBT approaches with the ability of TBD methods to detect low SNR targets are needed. These methods are the focus of this research.

2.3 Algorithms for Comparison

Because the focus of this research is on algorithms with the potential to operate in the near real time realm, direct realistic comparisons will be made to two other detect-before-track approaches both in terms of their detection performance and computation time. Both of these algorithms essentially consist of a background suppression step followed by a local spatial detection method, like the one use in [39].

The first algorithm is known as coaddition, or coadd, and implements several basic

local background estimation strategies combined with frame summation. Figure 10 shows a single pixel from a frame stack of 40 frames.

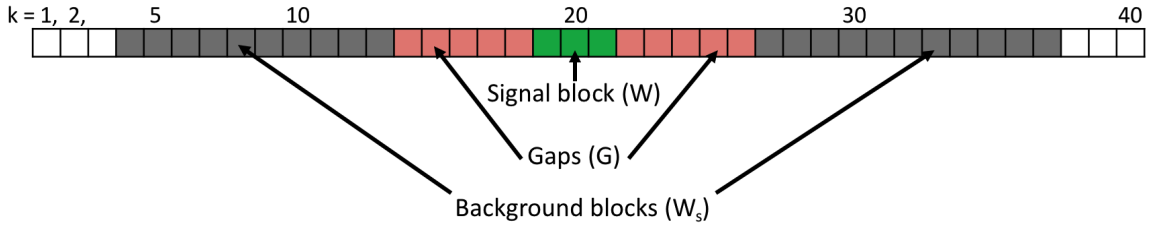


Figure 10. Coaddition algorithm parameters and calculations

The coadd algorithm computes a coadded frame by performing a signal operation on W frames in the signal block, a background operation on the W_S frames in the background blocks, and then subtracting the two. The whole process is then repeated after sliding S frames forward in the frame stack. The available operations are mean, median, minimum and maximum. The result of this process is a background suppressed and signal enhanced frame stack where the level of enhancement depends on how much platform motion is present (in the background case) and how long the target dwells in a single pixel (in the target case). For the background case, this is because the mean, median, minimum and maximum will only provide relevant information when the pixel is imaging the same portion of the scene over the W_S included frames. In the target case, summing frames will enhance the signal as long as the target is still partially in the pixel since the signal will add and the uncorrelated noise will tend to average to zero, resulting in a theoretical SNR increase of $\sqrt{W}$.

The second algorithm used for comparison is a running principle components algorithm. Whereas coadd subtracts a background estimated using simple operations, running principle components is a more complex and accurate method of background estimation. The parameters and implementation of the running principle components algorithm are the same as they are for coadd, with the exception of the background portion. Instead of performing a simple operation over the background frames and

using the result as a background estimate, the running principle components algorithm uses the background frames (W_S) to determine the principle components, and then projects the missing frames that contain the signal (W) into that space to get a background estimate for the current frame. The result of this process is a very good estimate of the background, with the different principle components being used to account for all the different sensor motion that occurs.

For the detection step, the implementation in both these algorithms is the same. A spatial kernel like the one shown in Figure 11 is used to compute a peak local spatial SNR, $SNR_S = (x - \mu_S)/\sigma_S$ where μ_S and σ_S are the mean and standard deviation computed using the gray background pixels and x is the red test pixel.

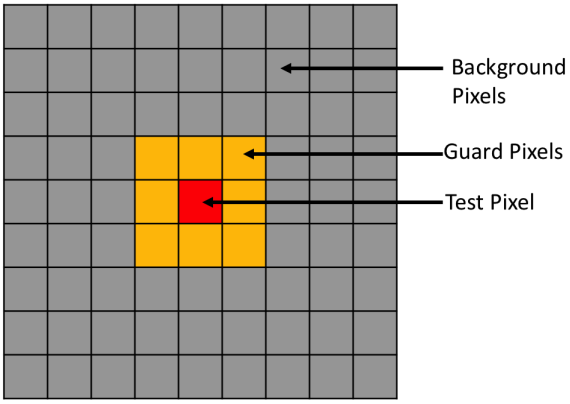


Figure 11. Kernel used to compute local spatial SNR for detect-before-track methods

This computation is done for every pixel in the frame stack, resulting in a new cube of local spatial SNRs. To get detections, a threshold is set and pixels with a local spatial SNR greater than the threshold are *detects* and set to one, while those with lesser values are set to zero.

2.4 Metrics

In order to quantify how well a detection or tracking method is performing, a number of different metrics are used both to quantify how challenging the scenario is and to

measure the detection performance.

2.4.1 Signal to Noise Ratio.

In the previous section, the local spatial signal to noise ratio, SNR_S, was introduced as a metric to describe how easily a target can be detected. Generally speaking, the signal to noise ratio is defined as the ratio of the signal of interest to the background noise level [22], but there are numerous definitions for both the signal and the noise terms in the ratio that can greatly impact the calculated value. Here we will lay out the many different methods of computing SNR in the context of targets in wide area motion imagery, and discuss how they relate to each other, allowing the results presented here to be put in the context of existing work.

As discussed in Section 2.1.2, the energy from a target is smeared across multiple pixels by the point spread function. As a result, rather than a single bright pixel, several adjacent pixels will also appear brighter than the background. There are three different ways to describe the signal of the target. The first is in terms of a *peak maximum* signal, S_{MX} which is the value of the pixel containing the target when the target is perfectly centered. This value is of use in describing generally how bright a target is, but since targets are rarely perfectly centered, it is not a practical method of quantifying target brightness in real data. The second approach is to integrate over the point spread function to obtain a *total* signal, S_{TOT}. This will produce a much higher value than the peak maximum signal but is of use in scenarios where spatial features of the target are an important element. It also will be consistent as a target moves since it does not depend heavily on the position of the target relative to a pixel. The third approach is to use the actual value of the pixel containing the target as the signal, known as the *peak* signal, S_{PK}. This is the approach used for the local spatial SNR discussed in Section 2.3. Similar to the peak maximum signal, the peak signal is based on a single pixel, but it is

derived from the data and will change significantly depending on the target's position within the pixel. Since the MBD algorithm does not make use of spatial features in the frame stack, this third metric is the most logical choice given that it will quantify the instantaneous brightness of the target and provide a measure of the difficulty of detecting a target in any given frame.

The signal is only one component of the signal to noise ratio. There are also three common ways of quantifying the background noise. The first approach is to use the global standard deviation, σ_B, a metric that captures the variation across the whole data cube. For this to be a meaningful metric, the structured background must be removed first (as shown in Figure 4), because it is the time-varying noise that is of interest, not the spatial variation in the scene. Letting ε_{r_k,c_k} represent a pixel in the data cube after the background has been removed,

$$\sigma_B = \sqrt{\frac{1}{r\,c\,k}\sum_{g=1}^{r}\sum_{i=1}^{c}\sum_{j=1}^{k}(\varepsilon_{g_j,i_j}-\mu_\varepsilon)^2}, \tag{3}$$

where μ_ε is the mean over all noise pixels ε_{r_k,c_k}. For this approach to be accurate, the structured background must be removed effectively and the noise must be fairly uniform across the scene. In cases where there is platform motion and the background cannot be easily removed, a localized estimate of the variation in the background is of more use. This can be done either temporally, computing the standard deviation for a single spatial pixel over time, or spatially, computing the mean and standard deviation over a local spatial region as discussed in Section 2.3. For the temporal case, the standard deviation for a pixel can either be computed using its entire temporal history, or it can be computed using an approach analogous to the local spatial approach, computing the standard deviation over a local *temporal* region and excluding the frames containing the signal. The latter approach is much more accurate as it excludes the target, preventing

the target's signal from artificially inflating the background variation. Figure 12 is the temporal analog of Figure 11. In this figure, σ_T is the standard deviation of the gray pixels.

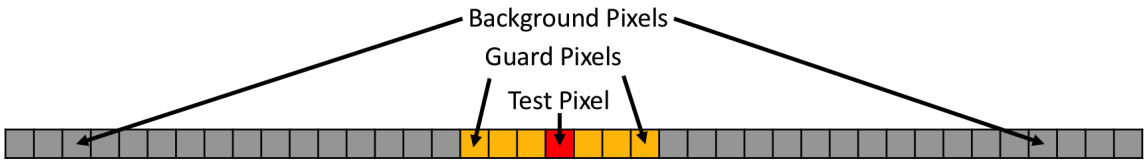
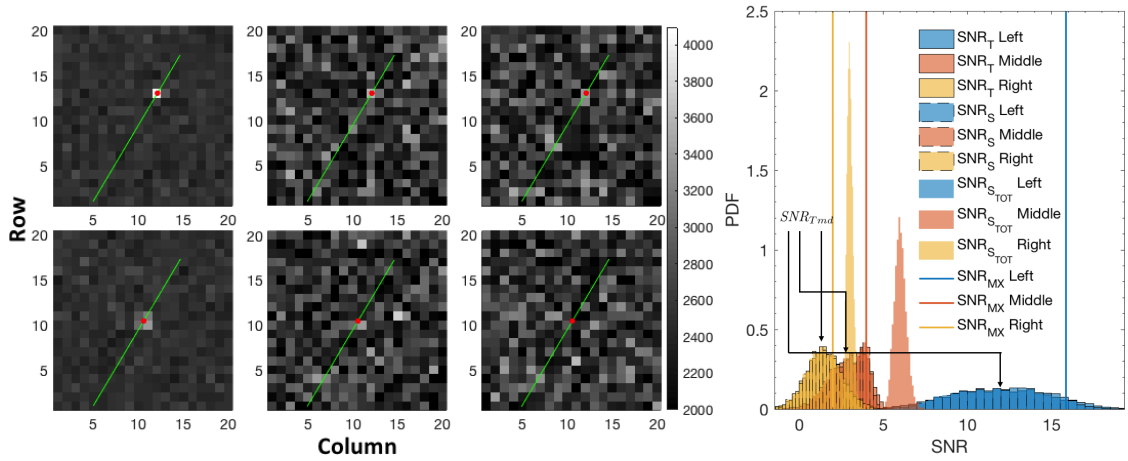


Figure 12. Regions for computing temporal SNR for a single spatial pixel

Now that we have discussed the various methods of computing the signal and noise in a data cube, we turn to combining these two elements into some common SNR metrics, and discuss how these will be used in the context of this research. The first of these, the local spatial SNR, SNR_S, has already been discussed. A second useful localized metric is the local temporal SNR, $SNR_T = S_{PK}/\sigma_T$, which quantifies the instantaneous difficulty of detecting a target in a single spatial pixel and frame using the pixel's time history. In the case of perfect background suppression, as more temporal frames are included, these two metrics should approach each other. Additional useful metrics are the peak maximum global SNR, $SNR_{MX} = S_{MX}/\sigma_B$ and the total local spatial SNR, $SNR_{S_{TOT}} = S_{TOT}/\sigma_S$. Other combinations of the signal and background calculations discussed previously are possible, but this research will primarily use the local spatial and temporal SNRs and will make a comparison to work that uses the peak maximum global metric.

To illustrate how these metrics differ, consider a scenario with a target moving in a straight line in Gaussian distributed noise as shown in Figure 13a. One hundred such scenarios were created at a single input signal level with the target persisting for 20 frames in each case for a total of 2000 data points containing target. The target's SNR was then decreased twice to create 2 more sets of data with the only difference being a lowered target signal. Two individual frames from each of these 3 scenarios are shown

from left to right in Figure 13a. Figure 13b shows histograms of the calculated local temporal, local spatial, and total local spatial SNRs along with a line representing the peak maximum global SNR for each scenario.



(a) Target Examples

(b) SNRs

Figure 13. Examples of variation in SNR of targets depending on position and the method used to calculate signal and noise components

As expected, the local spatial and local temporal SNR calculations align very well, but show significant spread. This is due to the variation in the position of the target in a pixel and is noticeable when comparing the top and bottom rows in the first column of Figure 13a. In the top row the target is mostly centered on a pixel, while in the bottom row it is at the corner of a pixel. The difference in brightness is significant, and it is this difference that causes the spread in the histograms of Figure 13b. As mentioned previously, one way to account for this is to use the total local spatial SNR instead, and as expected, these histograms are not as spread out. The spread that is present is due to the small variations in the local spatial noise over the scene. However, for this research, this metric is not as meaningful since single spatial pixels are operated on independently to produce detects. Finally, the lines indicating the peak maximum global SNR appear on the right tail side of the local temporal and spatial SNR distributions, but are clearly

not the maximum values for these distributions. This is due to the variation in the local noise statistics. When the local temporal or spatial noise is lower than the global noise and the target is relatively well centered in a pixel, the local metrics will likely produce higher values than the peak maximum global SNR.

Clearly, there are a large number of methods of quantifying how difficult a target is to detect. For this research, since the MBD algorithm operates primarily temporally and because it does not depend on how the background is removed, we will use the median of the distribution of a target's local temporal SNR, SNR_T, as a single metric when discussing the SNR of targets within a scene. Also note that in cases where there is considerable platform motion, and therefor clutter noise, the value computed for SNR_T becomes a measure of the signal to clutter plus noise ratio, $SCNR_T$, since it is not possible to separate clutter noise from other noise sources. When the background is perfectly removed, clutter noise goes to zero and $SNR_T = SCNR_T$. Throughout the following chapters, when the local temporal SNR is computed *before* background suppression, it will be referred to as the local temporal signal to clutter plus noise ratio $SCNR_T$, and when the same metric is computed *after* background suppression, it will be referred to as the local temporal signal to noise ratio, SNR_T. In the latter case, imperfect background suppression may still leave some artifacts, but these will have the same impact on target detectability as other noise sources, and thus we make no distinction between the two types of noise.

2.4.2 Reciever Operating Characteristic Curves.

Now that we have a method of quantifying a target's brightness in a meaningful and repeatable way, we need metrics with which to compare and quantify the performance of a detection algorithm. As discussed previously, the detection problem involves determining if any given pixel contains either only background, or background with the

addition of a target. To illustrate this concept, consider a target embedded in normally distributed noise with mean μ_B and standard deviation σ_B. Using the framework of a statistical hypothesis test and assuming that the presence of a target causes only a shift in the mean of the background normal distribution from μ_B to some higher value, μ_T,

$$H_0: \quad X \backsim \frac{1}{\sqrt{2\pi\sigma_B}} \exp \frac{-(x-\mu_B)^2}{2\sigma_B^2} \tag{4a}$$

$$H_a: \quad X \backsim \frac{1}{\sqrt{2\pi\sigma_B}} \exp \frac{-(x-\mu_T)^2}{2\sigma_B^2}, \tag{4b}$$

where X is a random variable representing a pixel in a data cube and the magnitude of μ_T depends on the SNR of the target. Figure 14 show plots of the null and alternative hypotheses in blue and orange respectively. Since the detection problem is binary, a threshold is chosen and realizations of X that are greater than that threshold will result in rejecting the null hypothesis while lesser values will accept it. In the example in Figure 14, the threshold is set at five.

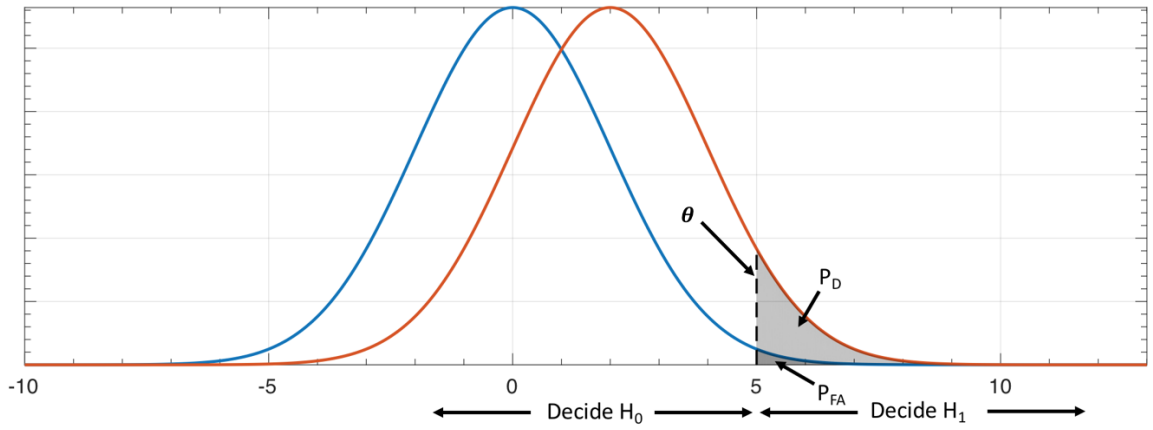


Figure 14. Null (blue) and alternative (orange) hypothesis probability distributions with $\sigma_B = 2$, $\mu_B = 0$ and $\mu_T = 2$, resulting in $SNR_{MX} = 1$

We estimate the *probability of detection*, $\hat{P}_D$ as the number of accurate decisions, or true

positives, divided by the total number of trials or truth pixels,

$$\hat{P}_D = \frac{\text{\# True Positive Detects}}{\text{\# Total Truth Pixels}}. \tag{5}$$

This metric quantifies how effectively a target is detected, with zero representing no detects, and one representing perfect detection. The lighter shaded region in the figure indicates the probability of detection with a threshold set at 5. Clearly, a large portion of the pdf is not included in this region (in this example, $\hat{P}_D = 0.07$). Sliding the threshold to the left will increase the probability of detection, but at the cost of more incorrect decisions, where the null hypothesis is accepted while a target is actually present. This trade off is quantified with a second metric, the *false alarm probability*, $\hat{P}_{FA}$. In the context of detection, there are a number of similar ways to quantify this property, but for this research we estimate it as the number of false detections divided by the total number of pixels in the data cube,

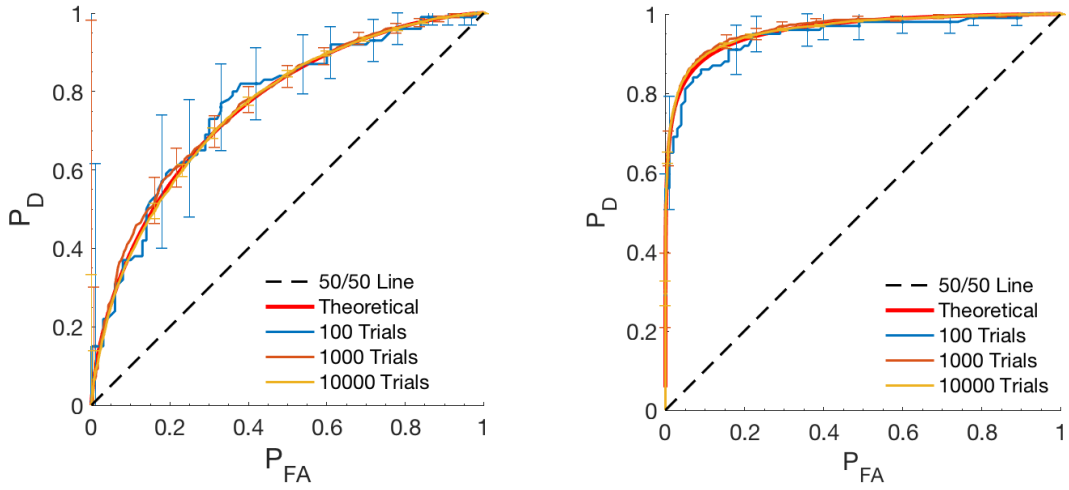
$$\hat{P}_{FA} = \frac{\text{\# False Positive Detects}}{\text{\# Total Pixels } (R \times C \times K)}. \tag{6}$$

Different applications will have different requirements when it comes to acceptable false alarm and detection probabilities. As a result, it is useful to examine the performance of a detection method over the whole space of possible false alarm and detection probabilities. This is traditionally done using a *reciever operating characteristics curve* or *ROC curve*, which is a plot of the probability of detection versus the false alarm probability. Each point on the curve represents a different possible operating point for the detector, providing a complete picture of the detector's performance. If the probability distribution functions for the null and alternative hypothesis are known (as is the case in Figure 14), the ROC curve can be computed directly from their cumulative distribution functions with $\hat{P}_D = 1 - CDF_{Tgt+Bkg}(\theta)$ and $\hat{P}_{FA} = 1 - CDF_{Bkg}(\theta)$, but when

testing an algorithm on a set of data, we estimate $\hat{P}_D$ and $\hat{P}_{FA}$ through equations 5 and 6. Clearly, the accuracy of this approach will depend on the quantity of data used. To quantify this, the relative absolute error at some significance level α given N trials can be computed for P_D as

$$\epsilon = \sqrt{\frac{[Q^{-1}(\alpha/2)]^2(1-P_D)}{NP_D}}, \tag{7}$$

where Q is the right tailed probability for the standard normal distribution and $100(1-\alpha)$ is the confidence level [20]. Figure 15a shows the ROC curves for the scenario in Figure 14 and Figure 15b shows the same scenario with a target that is three times as bright.



(a) $SNR_{MX} = 1$ (b) $SNR_{MX} = 3$

Figure 15. ROC curves computed theoretically and numerically for two targets with $SNR_{MX} = 1$ (a) and $SNR_{MX} = 3$ (b) with error bars computed using equation 7 with $\alpha = 0.05$.

The red curve represents the theoretical ROC curve based on the cumulative distribution functions while the other curves are numerically computed using varying numbers of trials. As the number of trials or probability of detection increase, the error bars shrink and the numerically determined curves approach the theoretical curve as expected. Equation 7 will be used for all error bars in the following sections, where the number of trials will simply be the number of truth points (the denominator in P_D).

Finally, while ROC curves are an excellent method of comparing a small number of detection approaches, the large number of data points makes it difficult to visualize and compare the results when comparing many different sets of algorithm parameters. To solve this problem, two approaches will be used to compare the information contained in ROC curves using a single metric. The first involves computing the minimum distance to the top left corner of the plot containing the ROC curve,

$$D_{ROC} = \sqrt{P_{FA}^2 + (1 - P_D^2)}. \tag{8}$$

This metric is a representation of overall performance for a detector, but in some cases the minimum distance may have an unacceptable false alarm probability for an application. The second approach involves choosing a specific false alarm probability and finding the probability of detection at that single operating point. Both of these methods will be used to quantify and compare the detection performance of algorithms in the following chapters.

2.5 Algorithm Test Data

In order to truly understand the performance of MBD compared to the two algorithms described in Section 2.3, a direct comparison is necessary. Because the algorithms will perform differently depending on elements such as noise, sensor motion, target dynamics, and target signal, to make a meaningful comparison, sets of data that span the relevant parameters are needed. Additionally, to compute the metrics discussed in Section 2.4, absolute target truth must also be known. There are three primary ways of obtaining such data sets. The generally preferred method is to obtain data from real sensors; however, these data sets are often difficult to obtain, the truth is estimated and not necessarily accurate, and the scene content cannot be easily

controlled to test algorithms under a large variety of relevant conditions. All of these elements make using real data in the initial development and testing of algorithms very challenging. The second method is to use real data sets for the background imagery and embed moving targets to detect. This approach solves the problem of obtaining absolute truth, but still does not allow for a great deal of flexibility for testing algorithms in a variety of scenarios, and again presents the problem of obtaining good data sets to work with. The third and final option is to use synthetically generated data. With this approach, all elements of the scene and target motion are controllable, allowing for thorough studies of automatic detection algorithm performance across a range of scenarios. When considering ease of use and the ability to evaluate algorithms over a large number of scene parameters, the option of generating synthetic data is the most practical, as long as it is truly representative of real data.

III. ASSET

This chapter is a repetition of [42], a conference paper presented at the 2017 international society for optics and photonics, defense and commercial sensing (SPIE DCS) conference. Minor changes have been made to improve grammar and clarity but the content is the same as originally presented and published in the conference proceedings.

3.1 ASSET Model Description

ASSET was originally developed for internal use at the Air Force Institute of Technology (AFIT) to support student research where absolute knowledge of object position and radiometric signature (i.e. *truth*) is needed, as is the case in detection and tracking efforts [43], as well as research where a large number of data sets are required, such as machine learning. It is intended to be used to model sensor response to at-aperture irradiance, generate data sets suitable for algorithm development and testing, and allow for the investigation of sensor configurations in a way that is computationally efficient and easily accessible to the user. As a result, it is not suitable for work where exact radiometric properties of specific scenes are needed: for example, in absolute band comparisons in various weather applications – for those types of applications, tools such as the Digital Imaging and Remote Sensing Image Generation (DIRSIG) model [34] or the Monte Carlo Scene (MCScene) model [33, 3] may be better suited.

ASSET is a sensor *emulator*, not a simulator, and as such is not designed to replace high fidelity models such DIRSIG or MCScene, but rather to operate in the problem space where plausible and realistic but time consuming calculations, such as using the bidirectional reflectance distribution function (BRDF) and ray tracing, do not provide a significantly improved result for the application at hand, i.e. where the signal statistics

and sensor artifacts are much more important than absolute radiometric representation. ASSET's value is its ability to reproduce the imperfections commonly associated with real sensors but not often observed in simulated data, and to do so efficiently and with a shallow learning curve.

3.1.1 Model Overview.

ASSET is a physics-based image-chain model in which a high resolution *source image* provides the basis for the background scene, various remote sensing processes are emulated, and a dynamic (i.e. time-varying) array of data frames is output. The overall process flow is shown in Figure 16. For clarity, we refer to the input as the *source image*; intermediate steps as *oversampled arrays* (for reasons which will be later explained); and the sensor's output as a *data cube* comprised of multiple *detector frames*.

Characteristics of the sensor and scene emulated in ASSET are specified with an ASCII text configuration file containing all of the tunable parameters in the emulator. A representative subset of these parameters are identified in Table 1. To start, ASSET takes the high resolution source image as an input. ASSET was originally developed to use Landsat-8, which is approximately 6000×6000 samples at 30m ground sample distance (GSD), but any generic image can be used with additional user-defined spatial and radiometric properties. For Landsat-8 inputs, the data are calibrated to convert from digital numbers (DN) to top of atmosphere spectral radiance (L_λ) using the Landsat-8 metadata; for generic images, the user defines the dynamic range of the scene radiance, and the minimum to maximum DN are linearly mapped to the minimum and maximum radiance values. The user may instead specify the range of surfaces reflectivity (or its complement, emissivity) which are likewise linearly interpolated to DN. For the latter, surface radiance is then computed as the sum of Lambertian-reflected seasonally-

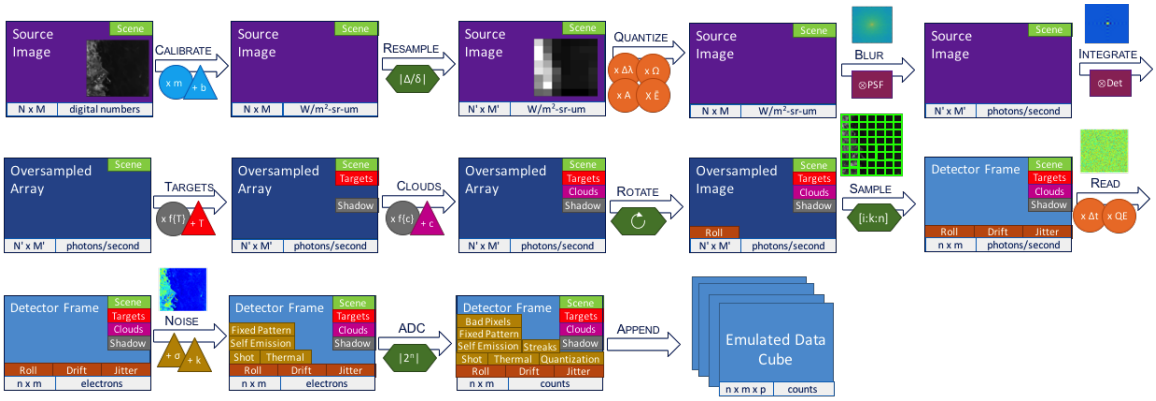


Figure 16. Process flow diagram for ASSET. The large box colors represent changes in the dimensions of the source image as it is propagated through the model, as noted in the lower left of each box; the inset boxes indicate which scene characteristics are included at each point in the process; and the bottom right of each box identifies the units associated with each pixel.

adjusted solar irradiance at earth's surface for the user-specified scene location, date, and time [28]; and thermal graybody emissions at surface temperature T. For both reflected and emitted components, spectral radiance is attenuated by atmosphere and path radiance is added,

$$L_\lambda = \tau_{atm} \cdot \left(\frac{\rho}{\pi} \cdot E_{sol} + \epsilon \cdot B(T) \right) + L_{path} \tag{9}$$

where τ_{atm} and L_{pth} are derived from a user-defined atmosphere, the default values of which are currently obtained from MODTRAN standard atmospheres [7].

The source image is then resampled via 2D linear interpolation so that the ratio of the model sensor's GSD (Δ) to the source image's GSD (δ) is an integer value equal to an user-specified oversample factor. This oversampling conserves energy and is performed in order to facilitate sensor jitter later in the process. In the case that the the source image is not of sufficient spatial resolution, i.e. the ratio Δ/δ is less than the specified oversample factor, a sharpening algorithm is used to artificially introduce higher spatial frequency content. This is necessary to provide sufficient spatial variation to generate realistic clutter statistics from sensor motion.

Table 1. ASSET Configuration Parameters

Source Image Properties		**Sensor Configuration**			
Rotation angle	[deg]	Frame rate	[Hz]	Quantum efficiency	[e/ph]
Scene dimensions	[km × km]	Integration Time	[μs]	Well depth	[e]
Pixel dimensions	[m × m]	Focal plane dimensions	[pix × pix]	Dark current	[e/s]
Radiance range	[W/m2-sr]	Ground sample distance	[m]	Dark current variance	[%]
Surface reflectivity range	[min, max]	Focal length	[cm]	Thermal	[e/s]
Emissivity range	[min, max]	Platform altitude	[km]	Read	[e]
Environmental Characteristics		Platform yaw rate	[deg/hr]	Flicker β	
Cloudyness	[%]	Platform jitter	[μrad/s]	Flicker strength	[e/s^{β}]
Cloud velocity	[km/hr]	Optical transmission	[%]	Dead pixels	[%]
Cloud warp rate	[m/s]	Aperture diameter	[cm]	Hot pixels	[%]
Cloud base	[m]	Point source width	[pix]	Happy pixels	[%]
Cloud depth	[m]	Cold stop temperature	[K]	Aimpoint drift	[μrad/s]
Cloud albedo	[%]	Cold stop emissivity		Vertical non-uniformity	[%]
Surface temperature	[K]	Optics temperature	[K]	Horizontal non-uniformity	[%]
Date & Time		Optics emissivity		Random non-uniformity	[%]
Location	[lat, lon]	Bit depth			

At this point it is helpful to introduce additional terminology to avoid confusion. The source image has $N \times M$ *samples* (or $N' \times M'$ *samples* after resampling). We use the term *samples* instead of *pixels* because the latter is reserved for the modeled focal plane, which is of dimensionality $n \times m$ *pixels*. The ratio $N'/n = M'/m = \Delta/\delta$ is the oversample factor, the reciprocal of which is the minimum sub-pixel shift in sensor pointing that can be accommodated by the model. A larger oversample factor results in finer spatial quantization of scene, and consequently a more highly resolved frame-to-frame jitter and drift, but at the cost of greater memory requirements. In summary, the source image is comprised of *samples*, the detector frames are comprised of *pixels*, and the ratio of pixel to sample linear dimension is the oversample factor.

The sensors emulated by ASSET do not respond directly to energy (e.g. joules). Rather, they are photon detectors, and thus source spectral radiance is converted to at-aperture photon flux,

$$\Phi_p = L_\lambda \frac{\lambda}{h \cdot c} \Omega \cdot A \tag{10}$$

where Ω is the solid angle subtended by the model sensor's aperture and A is the area of a source image sample (after resampling). In the case that Landsat-8 images are used as the input, the conversion from Watts to photons/second is accomplished using the

band-average photon energy.

To account for the effects of the optical system and sampling by detectors with finite spatial extent, the image is convolved with the point spread function (PSF) of the sensor and also with the detector response,

$$\text{Oversampled Array}_{N' \times M'} = \mathscr{F}^{-1}[\mathscr{F}(\text{Source Image}_{N \times M}) \times \mathscr{F}(PSF) \times \mathscr{F}(Det)] \quad (11)$$

Equation 11 shows how these two steps are accomplished. The Fourier Transform (FT) of the source image is computed and multiplied by the FTs of the PSF and detector response. Currently, ASSET uses a Gaussian as the PSF and a *rect* function as the detector response, but future development will allow the user to directly specify any arbitrary *PSF* or *Det* function. Taking the inverse transform produces an oversampled representation of the source image in which the PSF has introduced blurring (energy is conserved) and the detector response has spatially integrated (energy is summed). The oversampled array has a factor $(\Delta/\delta)^2$ greater energy than the source image (hence the term *oversampled*) because this array represents sampling of the scene at multiple sub-pixel positions – this will be used later to model sub-pixel jitter and drift.

Whereas the background scene is treated as static (i.e. background scene radiance does not vary) any objects or clouds in the scene may be dynamic, both moving and potentially changing shape over the duration of the sensor collection. Rather than injecting objects and clouds into the scene and repeating the process up to this point for every detector frame, for computation efficiency the background scene is computed only once, and each detector frame is obtained by sampling the oversampled array based on the jitter and drift sub-pixel shift of the sensor's pointing for the frame. Objects and clouds are processed in a similar manner and then fused with the background scene, accounting for obscuration and effects of blurring at edges, as will be described below.

ASSET currently has three methods of injecting objects: user-defined, pre-defined,

or random path. For all cases, the object position on the emulated sensor's focal plane for each frame is obtained from its projection from row, column, and altitude position in the source image. The object's radiometric signature (i.e. time-varying radiance) is either user-specified; derived from its projected area, temperature, and surface reflectivity and/or emissivity; or scaled to a desired signal-to-noise ratio (SNR) relative to the global scene noise. For the case where the surface properties are specified, the apparent object signal is modeled as a combination of solar reflection and graybody at the specified temperature, which is then weighted by atmospheric transmission and the sensor's relative spectral response (RSR) and integrated across the band pass,

$$\Phi_{tgt} = \int \tau_{atm} \cdot (\epsilon_{tgt} \cdot B(\lambda, T_{tgt}) + \frac{\rho}{\pi} \cdot E_{sol}) \cdot RSR(\lambda) d\lambda. \tag{12}$$

If the SNR was provided, rather than the surface properties, the object signal is scaled so that its peak signal – after accounting for the system response as in Equation 11 – is equal to the SNR times an estimate of the global noise. In all cases the object is currently modeled as spherical[1] if larger than or equal to that of a sample in the source image (30m for Landsat); or if the object area is smaller than that of a sample, then the object is treated as a point source. Although modeling the object shape these ways seems unconventional, for WFOV sensors with GSD on the order 100-1000m, the object is often much smaller than a pixel and the shape is unimportant. Nonetheless, future development will allow for non-spherical object shapes.

In an application where the user needs full control over the object, all object information can be provided as an input to ASSET. However, since many situations do not require specific object motion and generation of such data can be tedious, the user can instead provide kinematic limits on the object's velocity, lateral acceleration, and axial

[1]The version of ASSET validated here modeled the object as rectangular, but the version used in Chapter IV is the newer version with spherical objects

acceleration; and ASSET will produce a random object path that meets the provided standards. Additionally, since repeating the same object path across test data sets is often desirable, e.g. when evaluating different sensor configurations for the same scene and objects, the user can re-use previously simulated objects.

Once the object signal and path are defined, the object must be inserted into the scene. This is accomplished as in Equation 11 using a peak-normalized object *profile* (shape). Here, however, the convolutions are performed in two steps so that PSF blurring of the object profile can be used to also generate an obscuration mask which is used to remove signal from the background scene proportional to obscuration by the object: complete obscuration at the center of objects that are of multi-sample size, and partial obscuration for sub-sample size objects and at all object edges. The latter ultimately results in a blending of scene and object signal which is physically representative of the blurring of adjacent scene irradiance into the object, and vice versa, due to the optical system. In all cases, energy is conserved and the result is identical to – albeit more computationally efficient than – injecting the object into the source image for every frame.

Whereas the obscuration mask is applied after convolution with the PSF only (it's a blurring of the irradiance incident on the focal plane), the detector response must be included in the object signal to account for spatial integration of incident irradiance over each pixel's detector dimension. The object profile is convolved with the detector response, and its area-under-the-curve is scaled to the signal level as discussed above. This object signal profile is added to the oversampled array at the sub-pixel position for the corresponding frame. The result is a object whose signal profile properly represents changes in shape and pixel-to-pixel signal level due to changes in sub-pixel position frame-to-frame. This is shown in Figure 17 where the top set of *chips* (small region about the object) is the object signal profile and the bottom set is after insertion into

the data cube. These chips demonstrate the utility of ASSET, i.e. exact truth is known for every aspect of the data cube, allowing the algorithm developer to investigate whether anomalous results are due to their algorithm, object signal, sensor noise, artifacts, etc.

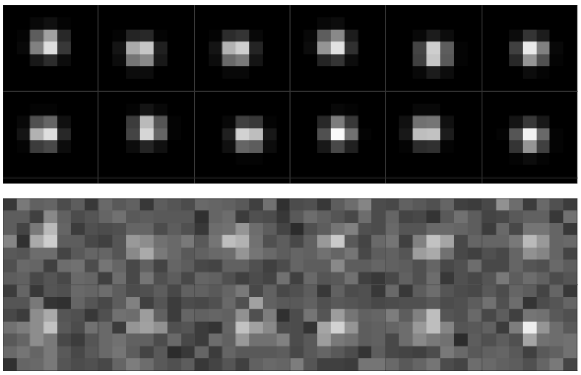


Figure 17. Objects as seen by the emulated sensor alone (top rows) and after insertion into the data cube (bottom rows).

Clouds can present a significant signal processing challenge due to their motion and changes in shape over the course of a sensor collection, and they are often a source of significant error when algorithms that have been developed on simplistic simulated data are applied to real sensors. Because of the need to develop algorithms that function in the presence of clouds, ASSET allows the user to introduce artificial clouds into a scene. A fractal model is used to generate cloud textures and depth which are converted to reflectance and altitude maps, respectively. Cloud radiance is computed as a combination of in-band solar-reflected irradiance and thermal emission based on the altitude-derived temperature of the International Standard Atmosphere [18]. The clouds are inserted into the oversampled array using the same process described above for objects. As indicated in Figure 16, we intend to include object and cloud shadows that are dependent on solar position, sensor viewing geometry, and shape and altitude. However, at present this functionality is still under development.

With all scene content incorporated, each frame of sensor data is obtained iteratively by first rotating the oversampled array (if the sensor is rotating) and then sampling the

array according to the current frame's aimpoint error. Rotation accounts for yaw of the satellite about the line-of-sight vector from sensor to aimpoint and is accomplished by rotating the image about its center (the aimpoint) by an angle calculated per frame equal to the product of the accumulated yaw rate and time since the beginning of the collection. Aimpoint error is the sum of sensor drift and jitter and may be specified directly by the user as a vector of aimpoint error per frame, or it may be defined by the user as a drift rate and random jitter motion. Jitter motion can further be specified as either distributed randomly about the ideal aimpoint or as a random shift from the previous frame's actual aimpoint (i.e. "random walk"). Regardless of how aimpoint error is specified, a frame's aimpoint error is rounded to the nearest integer sample (recall that δ/Δ is the minimum allowed sub-pixel shift, which corresponds to a single sample). The aimpoint error is a shift in row and column of the oversample array from its center, (E_N, E_M).

Sampling is accomplished by starting at some offset (N_0+E_N, M_0+E_M) that accounts for aimpoint error and edge padding of the oversampled array (the latter to allow for sensor drift and yaw without the emulated focal plane moving off of the image) and taking every $(\Delta/\delta)^{th}$ sample, in both rows and columns, until the focal plane size is reached. This reduces the number of samples from $(N \times M)$ to $(n \times m)$. These samples are the pixels of the modeled sensor's focal plane that comprise the *detector frame* for the current iteration. For example, the n^{th} row of a detector frame is the set of samples $\{\Phi_{n,M_0+E_M}, \Phi_{n,M_0+E_M+(\Delta/\delta)}, \Phi_{n,M_0+E_M+2\cdot(\Delta/\delta)}, ..., \Phi_{n,M_0+E_M+m\cdot(\Delta/\delta)}\}$ where $\Phi_{i,j}$ is the at-aperture photon flux in the $(i, j)^{th}$ sample of the oversampled array. It should now be evident why jitter and drift are quantized by the oversample factor. To obtain finer aimpoint error requires increasing the oversample factor, which introduces the trade-off between computational limits and model fidelity.

Self-emission photon flux, modeled as graybody radiances from low emissivity but

warm optics plus high emissivity but cold cryo chamber or cold stop, are added to the detector frame to obtain the total photon flux incident on the detector. Optics and cryo temperatures, emissivities, and relative proportions of the field of view of each pixel are all user-defined parameters that allow the user to include self-emission radiation, which is often important in MWIR through LWIR applications.

To emulate the detector response to the total incident photon flux, the detector frame is multiplied by the integration time to obtain the total number of photons collected during a single frame. The per-pixel photon numbers serve as the means of Poisson distributions that are randomly sampled to determine the total number of photons actually detected during the integration time – this is the *shot* or *photon* noise. The detector frame is then multiplied by a global quantum efficiency term perturbed by user-defined random and patterned non-uniformities to obtain a total number of electrons per pixel. Bias due to dark current is also added with a global value and random and patterned perturbations per pixel. The non-uniformities represent pixel-to-pixel variance in detector response due to either the detector material or read-out integrated circuit (ROIC). Figure 18 shows two examples of these non-uniformities.

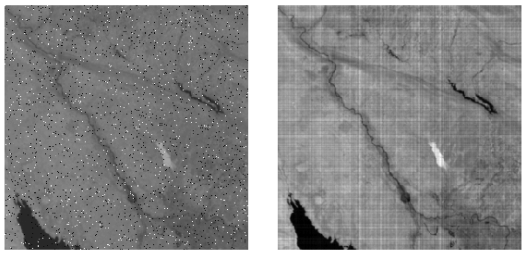


Figure 18. Examples of non-uniformities in ASSET including bad, happy and dead pixels (left) and fixed pattern noise (right).

Read and thermal noise are next added as random draws from time-independent and time-dependent normal distributions, respectively, with user-defined variance. Flicker noise is likewise added with a user-defined strength and frequency dependence (e.g.

$\beta = 1$ for Pink, $\beta = 2$ for Brownian, etc.). The detector frame, now in units of electrons and including most noise sources, is quantized with a conversion gain (electrons/count) derived from user-defined well-depth, bit-depth, and analog gain factor. The resulting *count* values (equivalent to DNs) are rounded down, introducing quantization error.

The final step is to add bad pixels and other focal plane artifacts to the detector frame. Bad pixels include dead, hot, and happy and are randomly distributed across the frame with user-defined clustering, fraction of the focal plane, minimum and maximum signal values, and fraction of frames fixed or flickering. Additional artifacts include streaks (in the case of a scanning sensor) and short-duration streaks (for staring or scanning sensors, representing a temporary fault in the ROIC) of configurable frequency and length.

Once all of these elements have been included, the detector frame is added to the data cube. The process continues iteratively until the full $n \times m \times p$ data cube is formed, where n is rows, m is columns p is frames, and the data cube's units are counts. In addition, nearly all of the elements can be output (or saved to files) independently, e.g. values of all noise sources per pixel and per frame, non-uniformities, bad pixel types and locations, object's sub-pixel positions and signal profiles as a function of frame, cloud shape and altitude per frame, and so on. As stated previously, this provides tremendous value for the algorithm developers who wish to test their code – and more importantly – understand exactly which sensor, scene, or object phenomena are hindrances or enablers for their application.

3.2 Validation

Some of the potential uses for ASSET include: as a tool for training students and analysts on a wide range of sensors, generating data sets for tracking and detection algorithm development, and exploring sensor configurations supporting the development

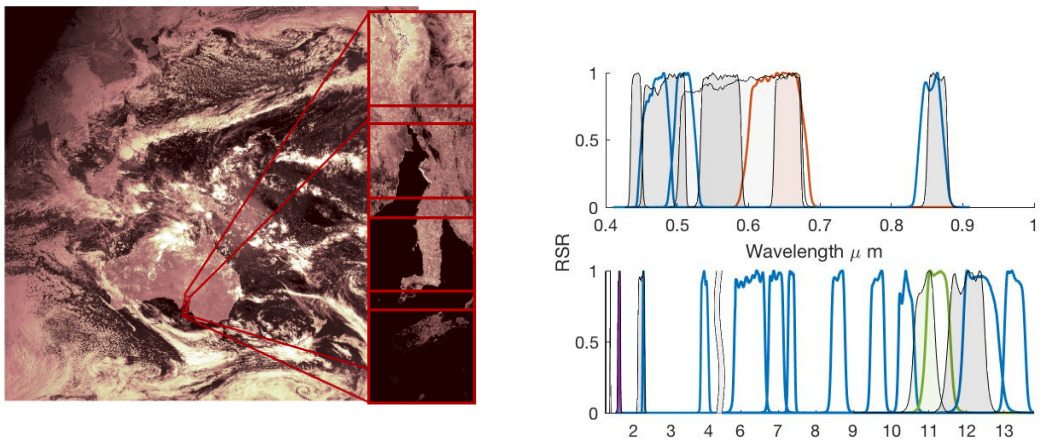
of new sensors. For all of these tasks, and to make the best use of ASSET, it is important to quantify and understand how well ASSET's emulated data represent data from real sensors. Two approaches were taken to achieve this goal: a space-based sensor data comparison, and a lab test with a mid-wave infrared camera and a visible camera. For the space-based sensor comparison, a frame of Himawari-8 data was emulated and then compared to real Himawari-8 data. The purpose of this effort was to quantify how well ASSET can match actual radiance values from a space-based sensor. The lab test was performed to quantify how well ASSET can emulate time-varying sensor characteristics, such as noise and jitter, in staring sensors operating at frame rates ranging from less than 1 to 10s of frames per second.

3.2.1 Satellite Data.

To test ASSET's ability to emulate a specific space-based sensor, synthetic data from the Japanese weather satellite Himawari-8 was generated using high resolution imagery from Landsat-8 as the source image, and the output was compared to actual Himawari-8 data collections. Figure 19a shows an example of a full disc earth image from Himawari with four corresponding high resolution Landsat images to give a sense of the relative areas covered by these two sensors.

For this comparison, two different methods for determining at-aperture irradiance were tested in ASSET. The first is the best case scenario in which a Landsat image and its corresponding calibration coefficients are available in a band similar to that of the sensor being emulated. Looking at Figure 19, it can be seen that this is the case for nine of Himawari-8's sixteen bands. The second case considered is where no such nearby band is available, and a satellite image collected in a different or unknown band must instead be used as the source input. In this latter scenario, the radiance of the source image is calculated from the minimum and maximum reflectance, emittance

or radiance values provided by the user. The lowest and highest DNs in the source image are then linearly scaled to match the user-specified bounds in order to create a distribution of values (reflectance or radiance) that match the real data as closely as possible. While all of Himawari's see to ground bands were examined, the remainder of this chapter will focus on three bands: one each in the visible, short-wave infrared (SWIR), and long-wave (LWIR) regions of the electromagnetic spectrum. These three bands are Himawari bands 3, 5 and 13 which match well with Landsat bands 4, 6 and 10 and are shown in orange, purple and green, respectively, in Figure 19. Additionally, to test the performance of ASSET for the out-of-band case, the same three Himawari bands were used, but this time with neighboring Landsat bands, namely bands 3 (center $\lambda = 0.56\mu m$), 7 (center $\lambda = 2.2\mu m$) and 11 (center $\lambda = 12\mu m$).



(a) Scenes

(b) Bands

Figure 19. Himawari-8 and Landsat 8 Data (a) and relative spectral response curves for Landsat-8 (solid gray) and Himawari-8 (blue line) bands. The three bands used for comparison are shown in orange, purple and green (b) .

In an ideal situation, all of the parameters of the emulated sensor would be known and the user would include them in the configuration file; however, only a subset of Himawari's parameters are releasable to the public. These parameters were obtained from the Advanced Himawari Imager (AHI) manufacturer and their values are listed in

Tables 2 and 3. The remaining required parameters (shown in gray rather than black in Tables 2 and 3) were found by first assuming reasonable values consistent with the known parameters, and then fine tuning the choices using a tuning scene. This process is described further in section 3.2.1.2

Table 2. Himawari Band Specific Parameters

Band #	Center λ (μm)	GSD (km)	Detector	Frame Δt (μs)	Well Depth ($e^- \times 10^6$)	Dark Current ($e^- \times 10^6$)
1, 2, 3	0.47, 0.51, 0.64	1, 1, 0.5	Si	902, 902, 451	1.5, 0.66, 0.413	46, 10, 10.3
4, 5, 6	0.86, 1.61, 2.26	2	HgCdTe	902, 902, 1803	1, 2.28, 2.93	5.23 9.33, 9.74
7-12	3.9, 6.19, 6.95 7.34, 8.5, 9.61	2	HgCdTe	1803	N/A (No matching band to tune)	N/A (No matching band to tune)
13,	10.35,	2	HgCdTe	1803	50	500
14, 15, 16	11.2, 12.3, 13.3	2	HgCdTe	1803		

Table 3. Himawari General Parameters

Parameter	Value
Altitude (km)	35,800
Aperture Diameter (cm)	27
Bits	11
Focal Length (cm)	35.8
Detector Size (μm)	20
Analog Gain (G)	1
Optical Transmission (τ)	0.85
Quantum Efficiency (η)	0.7

3.2.1.1 Baseline Comparison.

In order to know what constitutes good agreement between emulated and actual Himawari data, a baseline (or best-case) scenario for comparing Himawari-8 data to Landsat-8 data is needed. To accomplish this, both data sources were geolocated using information provided with each data set, then the lower resolution Himawari-8 data were upsampled to match the Landsat-8 data, and the two images were further aligned using an affine image registration method that minimized the squared distance between radiance values in the two data sources. Once the two high resolution images were registered, they were both converted back to the resolution of the Himawari data. This

was done by convolving the high resolution image with a rectangle function and then interpolating to find the values for the lower resolution array. Additionally, both data sources were converted to units of radiance ($W/m^2-sr-\mu m$) using the calibration coefficients for Landsat and a digital number to radiance look up table for Himawari.

Once the two scenes were registered and in the same units, their pixel-to-pixel radiance values were compared in two ways: using a point by point radiance comparison and a visual examination of the results in image and histogram form. These visualizations, along with their corresponding bands, are shown in Figures 20 and 21, first for the case where the bands of Himawari and Landsat match well, and second for a less-than-ideal scenario where the bands do not agree or even overlap significantly. The

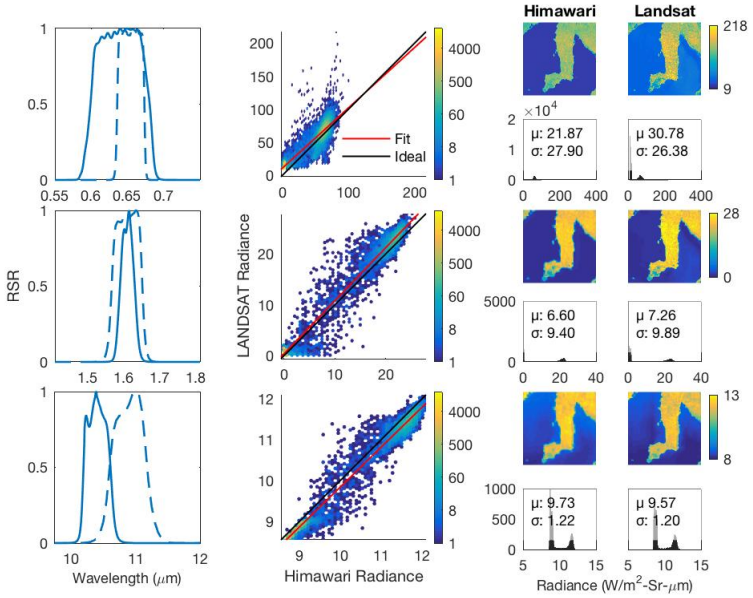


Figure 20. Comparison of Landsat-8 (dashed line) and Himawari-8 (solid line) data in matching bands for the Southern Australia scene used for tuning the parameters of ASSET in Section 3.2.1.2.

second column in both figures shows the radiances plotted point-by-point against each other as a scattered heat map where the radiance from Himawari is on the x-axis, the radiance from Landsat is on the y-axis, and the color representing the number of pixels is in a hexagonal bin of radiances. For a perfect match (i.e. the Himawari and Landsat

radiance were perfectly correlated), all points would lie along a line of slope one and intercept zero, plotted in black. The red line represents a fit to the scattered data.

The third column in the figures is a side-by-side comparison of the two data sources in units of radiance and on the same scale; this allows for a visual comparison of the imagery and their corresponding radiance distributions. This information provides a reasonable expectation of how similar the emulated Himawari data generated using ASSET could be expected to be to the actual Himawari data. This essentially quantifies the effect differences in bands of the source image and emulated sensor have, as well as any errors in calibration between the two sources. The first set of images (Figure 20) represents the best case scenario for a comparison of Himawari data to emulated Himawari data mentioned previously, while the second case (Figure 21) serves to quantify the differences introduced by using an image taken out of band as the input to ASSET.

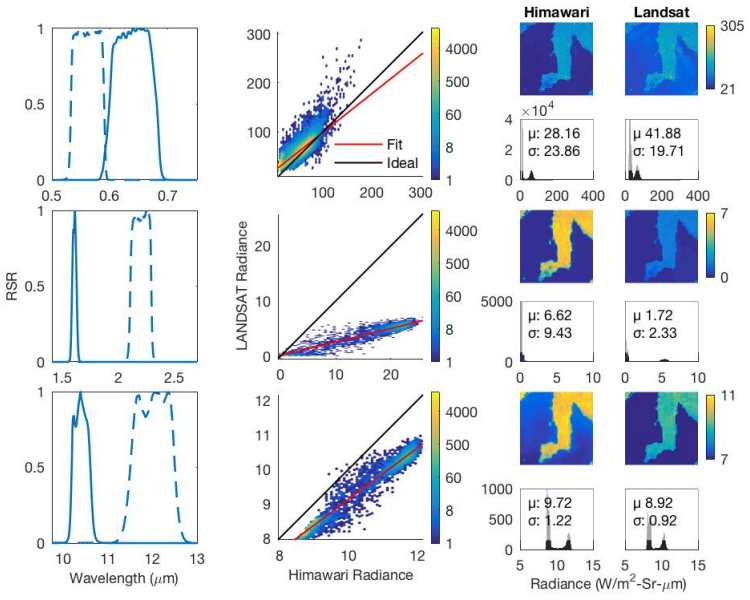


Figure 21. Comparison of Landsat-8 (dashed line) and Himawari-8 (solid line) data in neighboring bands for the Southern Australia scene used for tuning the parameters of ASSET in Section 3.2.1.2.

3.2.1.2 Determining Inputs to ASSET.

To determine the inputs to ASSET for the unknown Himawari parameters, a Landsat data collect was chosen such that there were at least two nearly cloud free images collected adjacent to each other (temporally and spatially) with at least one image containing features that appear very bright as well as very dark in the band (for example, sand and water). This high contrast scene was used to tune the ASSET parameters, while the neighboring scene or scenes were the test scenes. Of the unknown Himawari parameters, the well depth and dark current have the greatest effect on the emulated data since they control the scaling and bias of the digital numbers output by ASSET (although the same results could be achieved by adjusting other parameters, such as the analog gain or integration time, as many parameters have similar effects on the final result). With this choice, all other unknown parameters were decided first with knowledge of typical values, while the well depth and dark current were then adjusted to tune the configuration file to produce data that best matched the actual Himawari data in the tuning scene (using the same comparison approaches described in Section 3.2.1.1). In order to ensure a best possible match, this process was done using the calibration coefficients from the appropriate Landsat band (rather than by setting reflectance or radiance bounds) to calibrate from digital number to top of atmosphere spectral radiance. Figure 22 shows the final result of the comparison of ASSET to Himawari after this tuning has been done.

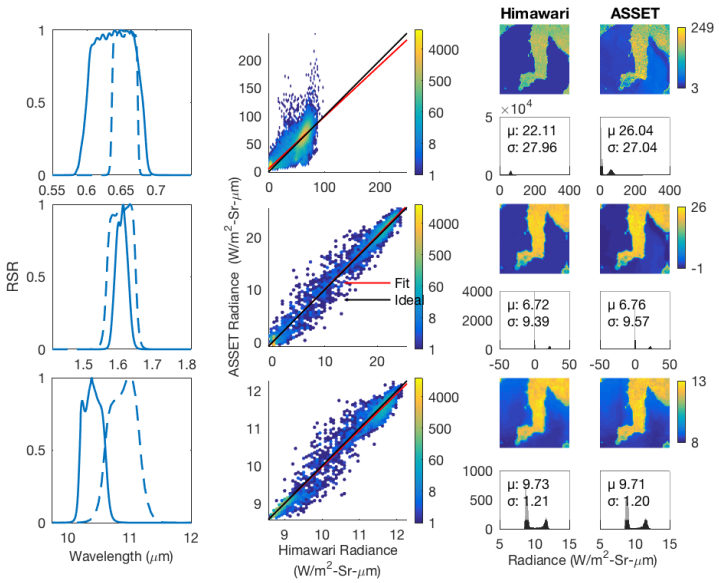


Figure 22. Comparison of ASSET (dashed line) and Himawari-8 (solid line) data in matching bands for the Southern Australia scene after tuning.

3.2.1.3 ASSET Emulated Data Comparison.

Now that a baseline expectation for the differences between Landsat and Himawari data has been established and the ASSET configuration parameters set, a number of scenes were emulated in ASSET and compared to their corresponding Himawari data without changing any of the configuration parameters determined using the tuning scene in Section 3.2.1.2. Figure 23 shows the results of this comparison for the top scene shown in Figure 19a when the Landsat coefficients were used to calibrate the top of atmosphere radiance. This is the best case scenario for ASSET, i.e. when bands match and calibration coefficients are available, and from Figure 23 it is clear that when given all of this information, it is possible for ASSET to match the output of a real sensor very closely in terms of per pixel radiance and overall data appearance. However, because matching bands and calibration coefficients are not always available, Figure 24 shows the results for a more realistic situation. In this case, a Landsat band neighboring to the one ASSET is emulating was used as the source image, and the top of atmosphere

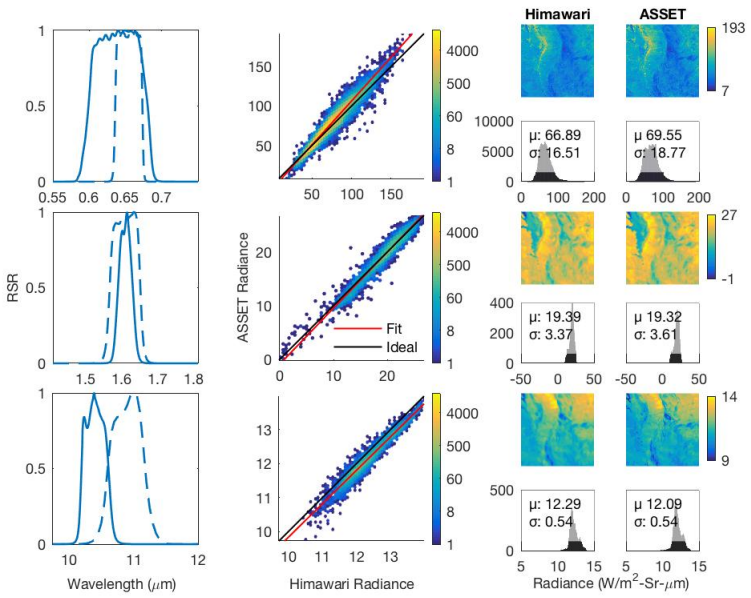


Figure 23. Comparison of ASSET and Himawari-8 data in matching bands and using Landsat coefficients for a Southern Australia test scene two Landsat rows above the tuning scene. The dashed line shows the band for Himawari/ASSET while the solid line is the band in which the source Landsat image was taken.

radiance calibration coefficients were determined by setting bounds on reflectance (for the visible and SWIR bands) or radiance (for the LWIR band). In order to best test ASSET, these bands were calculated directly from the Himawari data, representing the situation where these bounds are known for the scene being emulated in ASSET. While the emulated radiance values do not completely agree in this scenario, the SWIR and LWIR bands match better than the out-of-band baseline. The radiance distributions and overall appearance of the data are quite similar for all three bands.

ASSET is not intended to simulate a scene with absolutely correct radiance values, but rather is meant to *emulate* real data, with the goal of producing truly representative data quickly and easily. These comparisons to real satellite weather data show first, that when radiometrically calibrated source imagery is available in the same band as that being emulated, ASSET is able to emulate the sensor extremely accurately. Second, when no such matching imagery is available, ASSET is still able to produce a result with the

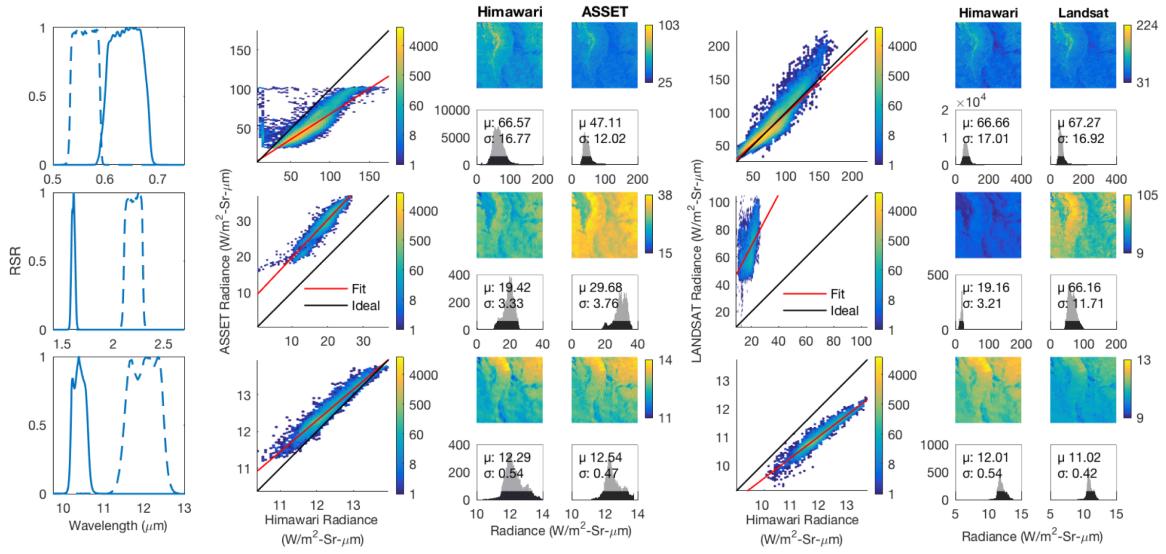


Figure 24. Comparison of ASSET and Himawari-8 data using a neighboring Landsat band source image scaled by user input reflectance (visible and SWIR bands) or radiance (LWIR) bounds for a Southern Australia test scene two Landsat rows above the tuning scene. The rightmost scatter plot and imagery compare the two sensors Himawari-8 and Landsat-8 directly.

same spatial characteristics of the real data, matching the overall shape of the radiance distributions and visual appearance well, even though the absolute radiance values differ. While this study showed that ASSET can match a single frame of a space-based sensor with some tuning of parameters in the configuration file, ASSET is intended to generate data cubes with a temporal dimension, so a second laboratory study was needed to test performance in the temporal domain.

3.2.2 Laboratory Image Comparison.

In order to test ASSET's temporal features and ability to emulate a specific camera where nearly all parameters are known by the user, two cameras were used to collect imagery in a controlled laboratory environment: a broadband MWIR Santa Barbara Focalplane array (SB-FPA) camera with an indium antimonide (InSb) detector, and a visible Canon Powershot G9 camera with a silicon detector operated in raw shooting mode. Figure 25 shows the approximate relative spectral response curves for each

camera based on the responses of a typical InSb detector and a typical visible camera [35, 14]

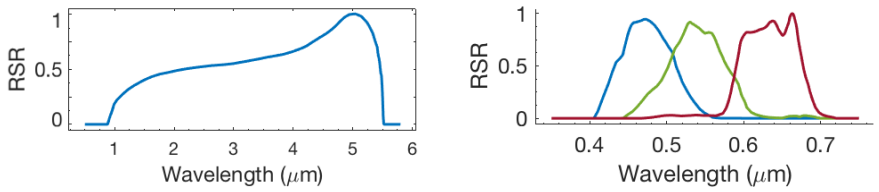


Figure 25. Approximate relative spectral responses used in ASSET to emulate the MWIR (a) and visible (b) cameras.

Both cameras were used to image a fixed scene in the laboratory which contained a variety of objects with varying colors and temperatures to include a resolution bar chart that was cut out of aluminum, painted matte white, and placed in front of a wide area blackbody to provide a highly structured, high contrast object in both the visible and mid-wave portions of the spectrum. Since ASSET requires a high resolution source image and is meant to emulate low spatial resolution, wide field of view sensors, both cameras were placed on the opposite side of the room, 15.24 meters from the scene, when the data was collected. Figure 26 shows the view from the visible camera as well as a single frame of data zoomed in on the region containing the scene.

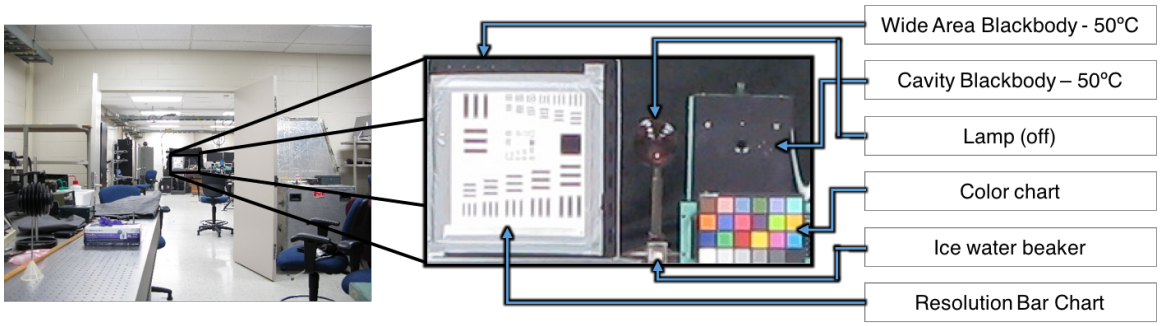


Figure 26. The scene imaged by both cameras in the laboratory as viewed from the camera position.

3.2.2.1 Visible Camera.

For the visible camera, data were taken (as shown in Figure 26) and an ASSET configuration file was created to match the real camera settings as closely as possible. Additionally, an iPhone 6S was used to take a closeup of the scene at a higher resolution for use as the ASSET source image. Table 4 shows the known or calculated (black) and estimated (gray) parameters for the scene and the Canon Powershot G9 camera used for this comparison. The parameters used in this comparison were found using four different methods. First, some of the camera parameters were set by the user or listed in the camera manual, including the frame rate, integration time, focal length, aperture diameter, and number of bits. Second, many parameters were obtained from an open source camera testing site [11] which determined parameters experimentally for a large number of cameras, including the Canon Powershot G9 used here. These parameters include the read noise, quantum efficiency, analog gain and well depth. Third, the scene parameters (including sample dimensions, GSD and distance), as well as the dark current, were measured in the laboratory. For the dark current, 70 frames of data were collected at the same integration time (125 μs) as the test data set and the results were converted from digital counts to electrons using the parameters of Table 4. Finally, the remaining unknown parameters (optical transmission, flicker noise, and energy on detector) were estimated by analyzing the appearance of the emulated data as compared to the real data and choosing values that produced a qualitatively similar result and were physically reasonable.

Table 4. Canon Powershot G9 ASSET Parameters

Parameter	Value	Parameter	Value	Parameter	Value
source image	iPhone 6S	Optical Transmission (τ)	0.99	Distance	15.24 m
Sample Dimensions	0.2 mm	Aperture Diameter	2.64 mm	QE (η)	0.5
Frame Rate	0.7 Hz	Read Noise	5.7 e	Dark Current	10 e/s
Frame Δt	125 μs	Shot noise	TRUE	Analog Gain	1.86
GSD	4.1 mm	Flicker Noise	β = -1.3, σ = 50 e	Well Depth	5562 e
Focal Length	7.4 mm	Bits	8	Energy on Detector	0.8

Since ASSET was primarily designed to use top of atmosphere solar irradiance values when calculating reflectance from a scene, and this test was performed indoors with fluorescent lighting, the option to input radiance bounds was used for this test. While the exact radiance limits were not known for the scene, rough starting estimates were obtained using a lux meter and then adjusted to best fit the real data for each of the three bands. The limits used for the results shown are (0.09,0.58), (0.11,0.72), and (0.11,0.85) $\frac{W}{m^2-sr-\mu m}$ for the red, green and blue bands, respectively.

Three approaches were taken to compare the emulated version of the visible data with the real camera data: a comparison of the distribution of digital count values for all frames, a comparison of the median subtracted digital count values for all 30 frames (which approximately represents sensor noise since all structured scene content is removed), and a visual comparison of the two data sources. Figure 27 shows these results for the camera settings of Table 4. It should be noted that small differences in the scene were introduced through the use of a different camera at a closer position, slightly altering the imaging perspective. However, even with this introduction in error, it can be seen that the ASSET data match the overall distribution of the real data quite well for all three bands. Additionally the noise distributions agree well, all having standard deviations for each band within 6 digital counts (2.4% of the dynamic range) of the real data. Finally, while there are visual differences in the two scenes, it is clear that ASSET is able to capture many of the most prominent features in the real data, including the cloudy and spatially correlated appearance caused by the flicker noise.

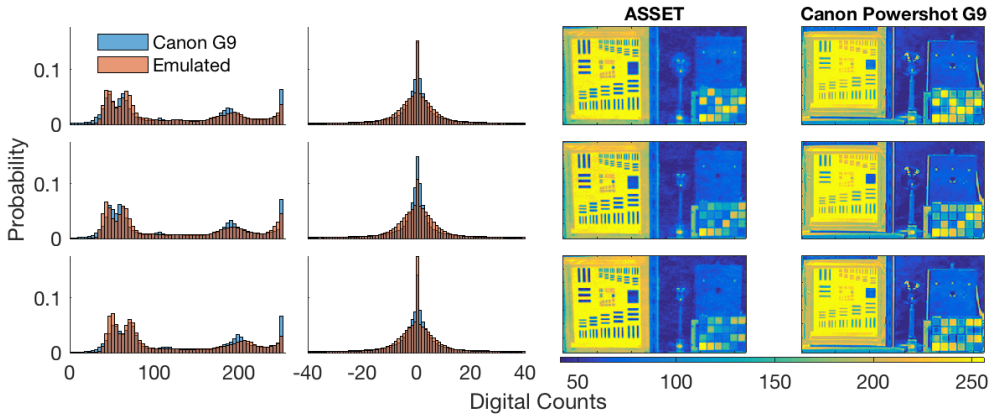


Figure 27. Comparison of ASSET emulated Canon Powershot G9 data to the real data for the red, green and blue bands from top to bottom in rows. From left to right: count values, difference of count values (noise estimate); single frame of data.

3.2.2.2 Mid-wave Camera.

While the visible camera test was able to show that it is possible to closely match the distribution of counts, noise counts, and overall appearance over a temporal window, limitations in the measurements for the radiance in the scene introduce a significant amount of error and a consequent need for user-adjusted input radiance parameters to obtain a good match. In order to test ASSET with a more accurate at-sensor irradiance input, a mid-wave camera was used in the same setup as the visible camera (Figure 26), but the scene was cropped to just the resolution bar chart where the temperatures for the white painted aluminum and blackbody are known. Additionally, in order to test ASSET's ability to accurately include jitter (and as a result clutter noise), the camera was gently moved during the data collection to introduce slight pointing errors. To determine the at-sensor irradiance in ASSET, binary temperature and emissivity maps for the bar chart were defined, with one temperature and emissivity for the white painted aluminum chart itself and another for the holes cut in the chart, revealing the wide area black body. These maps replaced the single ground temperature and the emissivity bounds normally used to estimate emissive radiance in ASSET, producing a more accurate result for the at-aperture irradiance. Figure 28 shows these maps as they

were input into ASSET.

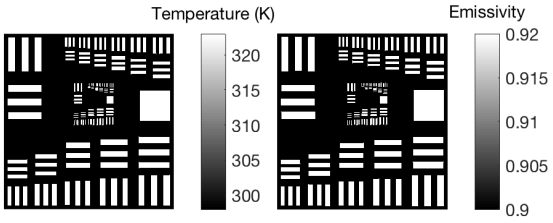


Figure 28. Input temperature (left) and emissivity (right) maps used to emulate the MWIR Santa Barbara Focal plane camera.

The temperature values were determined using an IR thermometer, and the emissivity values were approximated from emissivity tables [30] (for the white paint) and from a past experiment to determine the emissivity of the blackbody using a spectrometer [41]. While there is still some error inherent in the temperature and emissivity values, it is significantly reduced over that of the visible scenario, allowing for the focus to be primarily on ASSET's ability to accurately propagate at-sensor irradiance through to a digital count output. Table 5 shows all of the relevant configuration parameters used to emulate the mid-wave camera at three different integration times. The black parameters were obtained directly from the experiment (for the scene dimensions) or the manufacturer-provided camera specifications, and the gray parameters were estimated based on the sensor type and appearance of the frames of emulated data.

Table 5. Santa Barbara Focalplane Array (SB-FPA) ASSET Parameters

Parameter	Value	Parameter	Value	Parameter	Value
source image	Bar chart draft	Optical Transmission (τ)	0.5	Distance	15.24 m
Sample Dimensions	0.12 mm	Aperture Diameter	2.174 cm	QE (η)	0.8
Frame Rate	72 Hz	Read Noise	10 e	Analog Gain	2
Frame Δt	521, 614, 733 μs	Shot Noise	TRUE	Well Depth	7e6 e
GSD	6.89 mm	1/f Noise	β=-1.3, σ = 100 e	Energy on Detector	0.2
Focal Length	5 cm	Bits	16	Thermal Noise	7000 e$\sqrt{s}$

In the case of the thermal noise, the approximate value was calculated from the RMS thermal noise current using the formula $i_{thermal} = \sqrt{4kT\Delta f/R}$, where k is Boltzmann's constant, T is the temperature of the detector (73.3K), Δf is the noise electrical

bandwidth (approximated as $1/2\Delta t$ where Δt is the integration time), and R is the detector resistance, chosen to be $1M\Omega$ since the actual value is unknown. The results were compared as was done for the visible camera with the addition of a comparison of the offset (or aimpoint) error from the first frame and a visual comparison of the noise, obtained by subtracting the median of the 850 frames collected from a single frame.

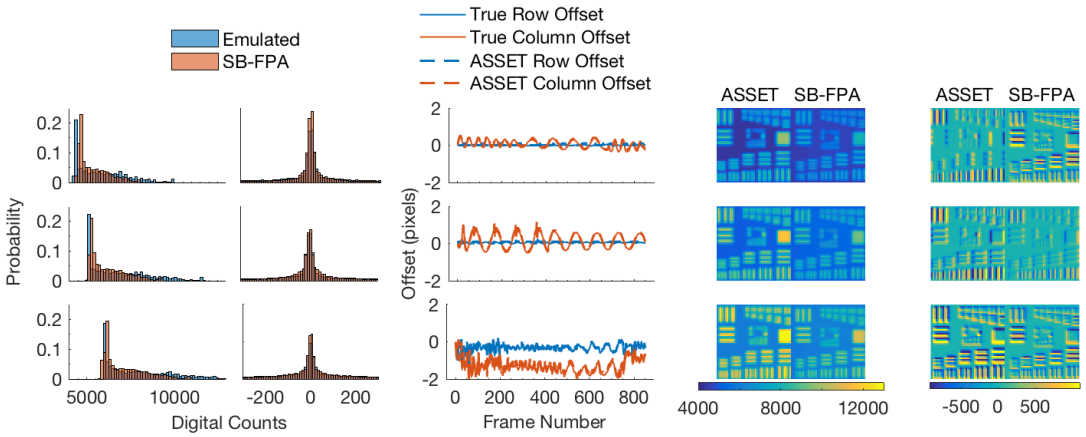


Figure 29. Comparison of emulated data to real mid-wave data for three different collects at three different integration times and levels of jitter, from top to bottom in rows.

Figure 29 shows these results from top to bottom in rows for the three integration times listed in Table 5 in increasing order. The first, second, and fourth columns are the same as for the visible results of Figure 27, and the third and fifth columns show the aimpoint error and a single median-subtracted frame, respectively. While the distributions do not exactly agree, they have the same overall shape, indicating that the emulated data is a relatively good, if not absolutely radiometrically accurate, representation of the real data. The offsets were computed from the first frame (in units of pixels) using a Fourier transform-based cross-correlation technique developed for rigid, sub-pixel image registration [17]. These results show that when given an input file containing the row and column offsets of a sensor for each frame, ASSET can reproduce that same sub-pixel aimpoint error in the emulated data. Finally, visually the two single raw and noise frame images differ due to the differences in peak values (as can be seen in the

histograms), but the structure and general appearances are again similar. This is both an expected and acceptable result because ASSET is not intended to exactly reproduce radiometric scenes, but rather to emulate a sensor and generate data that is plausible and *representative* of that sensor for the purposes of efforts such as tracking and detection algorithm development, machine learning applications, and sensor configuration studies.

3.3 Conclusion

ASSET is a physics-based, image-chain model used to generate synthetic wide field of view EO/IR sensor data containing realistic characteristics and artifacts. It is not intended to accurately reproduce radiometric real scenes or sensors but rather to provide realistic data sets plausibly representative of a wide range of scenes and sensors. Although ASSET includes the ability to model simple scenes, it focuses on the propagation from the aperture through to the digital output, introducing many of the imperfections inherent in real sensors, such as the imagers on unmanned aerial vehicles (UAVs) or weather satellites. With this in mind, it has been shown that ASSET can match single frames of the weather satellite Himawari-8 in the visible, NIR, and LWIR regions of the electromagnetic spectrum under ideal conditions, and the effects of non-ideal inputs have been quantified. Additionally, it has been shown that ASSET can emulate specific sensors both spatially and temporally when provided the relevant input camera parameters and scene information, to include temporally dependent elements such as shot, thermal, and flicker noise sources as well as clutter noise produced by jitter in the sensor. With the capability to quickly and simply generate large quantities of test data for a wide range of scene and sensor parameters, ASSET can give students a tool to better understand the relationships between sensor parameters and resulting imagery, aid in the development of a variety of algorithms whose goal is extracting information

from WFOV temporal imagery, and provide a starting point for designing new sensors optimized to collect imagery or video more relevant to the intended application.

IV. Moment Based Detection

Now that large quantities of realistic synthetic data can be generated quickly and easily using ASSET, we turn to the problem of developing a near real-time detection and tracking algorithm using statistical moments.

4.1 MBD Background and Theory

The DBT and TBD approaches described in Section 2.2 are not always distinct. Recently, progress has been made in the development of hybrid methods that make use of both spatial and temporal information simultaneously, but still follow the approach of feeding candidate detections into a tracker that is characteristic of the DBT methods. One such hybrid method is the moment based detection algorithm, which relies on the promising results of HOMAD found by Borel et al. [8].

HOMAD relies on the fact that as a target passes through a pixel, it momentarily perturbs the pixel's intensity. In order to detect this small perturbation, statistical moments are calculated. The first two statistical moments are the well known mean - a measure of the center of a probability density function (PDF) - and variance, the spread of the PDF. The third and fourth moments are known as skewness and kurtosis which measure asymmetry and peakedness of the PDF respectively. Additionally, an infinite number of higher order centralized moments can be computed using the equation

$$m^n = \int (x-a)^n P\,r(x) d\,x \tag{13}$$

where a is the mean of the PDF and $P\,r(x)$ is the PDF itself. Odd higher order moments are additional measures of asymmetry while even higher order moments are additional measures of peakedness. Note that here and in the following chapters, *moment* refers to a moment that is taken about some value a, not a raw moment. Figure 30 illustrates both

the potential power and difficulty in using higher order moments to detect anomalies.

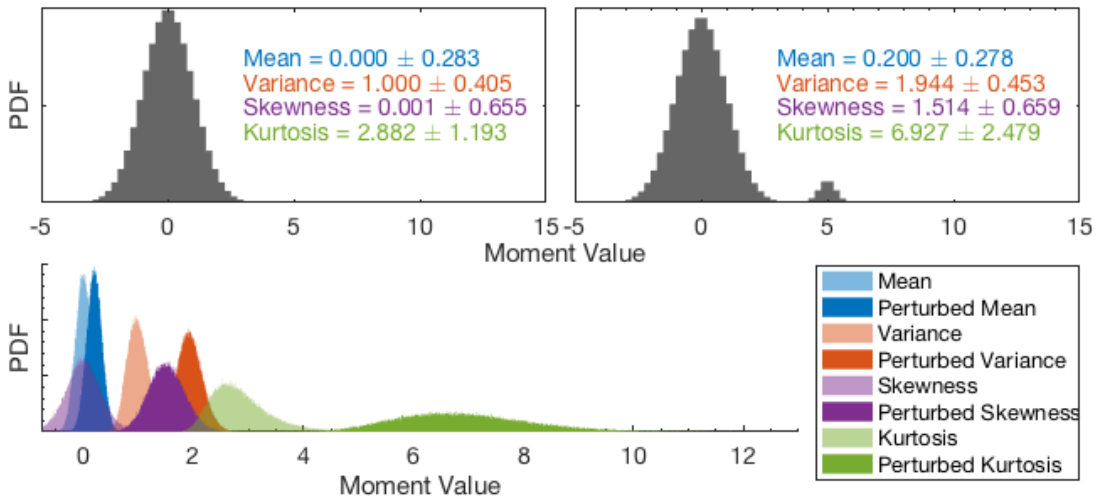


Figure 30. Comparison of the first four moments for 50 unperturbed (top left) and perturbed (top right) Gaussian distributed values with 2σ error, and the sampling distributions for these moments found by repeating the calculation 10^5 times (bottom).

On the top left is a histogram of 50 values from a Gaussian distribution with a mean of zero and a variance of one. On the top right is a histogram of 48 of the 50 values, with the 2 remaining values replaced by values taken from a Gaussian distribution with a mean of 5 instead of zero, representing an anomaly. It is clear from computing the first four moments that the third and fourth moments show a much greater change than the first and second. This is the principle upon which HOMAD and MBD are based. However, while the change in values is higher for higher order moments, so is the uncertainty in those values. The bottom of Figure 30 illustrates this uncertainty. The calculation used in the top two plots was repeated 10^5 times and the distributions of those values are shown in the bottom plot. In the case of the mean, this is the well known student's t distribution and in the case of the variance, this is the chi-squared distribution. Higher order moments do not have a simple closed form equation representing the sample distribution, but they can be investigated numerically as in [23]. The example in Figure 30 shows that even for a relatively large sample size of 50, the spread in the distributions

of skewness and kurtosis values is significant. Thus while increasing the moment order does produce a larger average change between the unperturbed and perturbed cases, there will be a point where the uncertainty outweighs this benefit. In the initial HOMAD publication, the first six moments were computed temporally for each spatially distinct (r, c) pixel in a simulated data cube containing embedded targets moving at a speed of 0.2 pixels per frame. The resulting images were displayed along with receiver operating characteristic (ROC) curves for moments 3-6. Additionally, the results of the global Reed-Xiaoli anomaly detector (RXD) defined as

$$\delta_{RXD} = (x-a)^T \Sigma^{-1} (x-a), \tag{14}$$

where x is the sample vector, a is the sample mean and Σ^{-1} is the inverse of the sample covariance matrix, were shown both visually and as a ROC curve. It was found that computing higher order moments (specifically moments 3-6) allowed for better detection of anomalies than RXD or the variance, both second order methods. This is a promising result, but at this point the moments were computed over the whole temporal vector for each pixel and considered separately. No study was done as to the relative performance of even higher order moments ($n > 6$), combinations of moments, alternative methods of dividing the data temporally to compute the moments. The MBD algorithm described in the following section allows for the adjustment of these elements in an effort to improve the detection of low SNR targets using higher order statistical moments.

4.2 MBD Algorithm

In the previous section the history and theoretical basis for the moment based detection algorithm were presented. Here the algorithm itself is discussed in detail

from beginning to end. Figure 31 shows the MBD algorithm process from the input, consisting of the raw data cube described in Section 2.2, to the final binary data cube output. The first step in the algorithm is to perform any necessary pre-processing of the

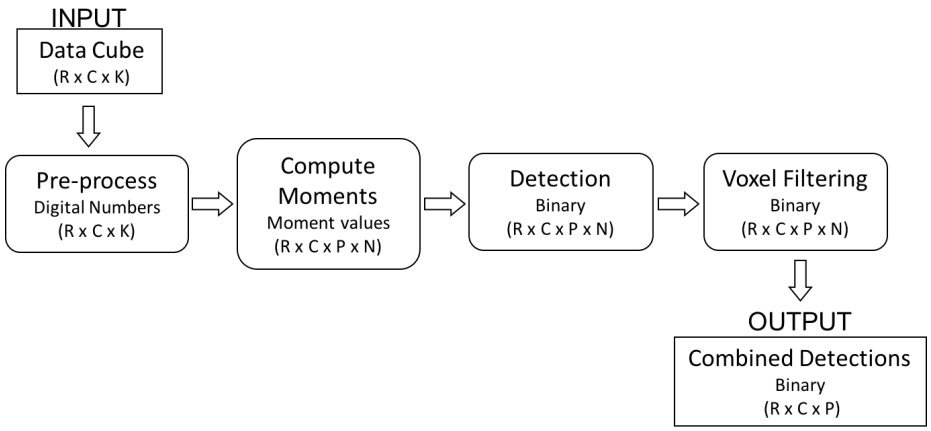


Figure 31. Flow chart of the MBD algorithm taking a frame stack as an input and outputting a final binary detections cube.

data before entering the main portion of the algorithm. For most detection algorithms this step involves a significant effort removing the background and correcting for non-uniformities in the focal plane array. Because the MBD algorithm operates on single spatial pixels independently, this is an optional step and is only necessary when there is significant platform motion or aimpoint error, altering the portion of the scene imaged by a particular pixel over time. In the case of the MBD algorithm, this is done using singular value decomposition (SVD) to model the background and remove artifacts.

The next step is to compute the temporal moments for each spatial pixel. This process is discussed in Section 4.2.1 and the output is a hypercube of moment values with the same spatial dimensions as the input data but with a reduced number of temporal frames P and a fourth dimension N for the number of moments computed. Next, in order to put all values on the same scale for the detection process, the z-score for each spatial pixel in the moments hypercube is computed with respect to its temporal history. This allows for a single threshold to be set based on a significance level choice,

producing a binary detections cube of the same dimensions as the z-scores hypercube. This process of calculating z-scores and thresholding is discussed in detail in Section 4.2.2. Finally, the detections cube is filtered based on the spatio-temporal characteristics of true targets and the moments are combined to produce a final three dimensional binary detections cube. Section 4.2.3 discusses this filtering process in detail.

4.2.1 Computing Moments.

The MBD algorithm computes the n^{th} order moments $M^n_{r,c}$ of windows of W temporal frames independently for each spatial pixel, $\mathbf{x_{r,c}}$. However, rather than compute the commonly used central moments (centered about the mean computed over the same window of W temporal frames, as used previously in [40]) , the moment is computed about an estimate of the background mean calculated using a larger super window W_S that is centered around but excludes the window W. Generally, this can be expressed,

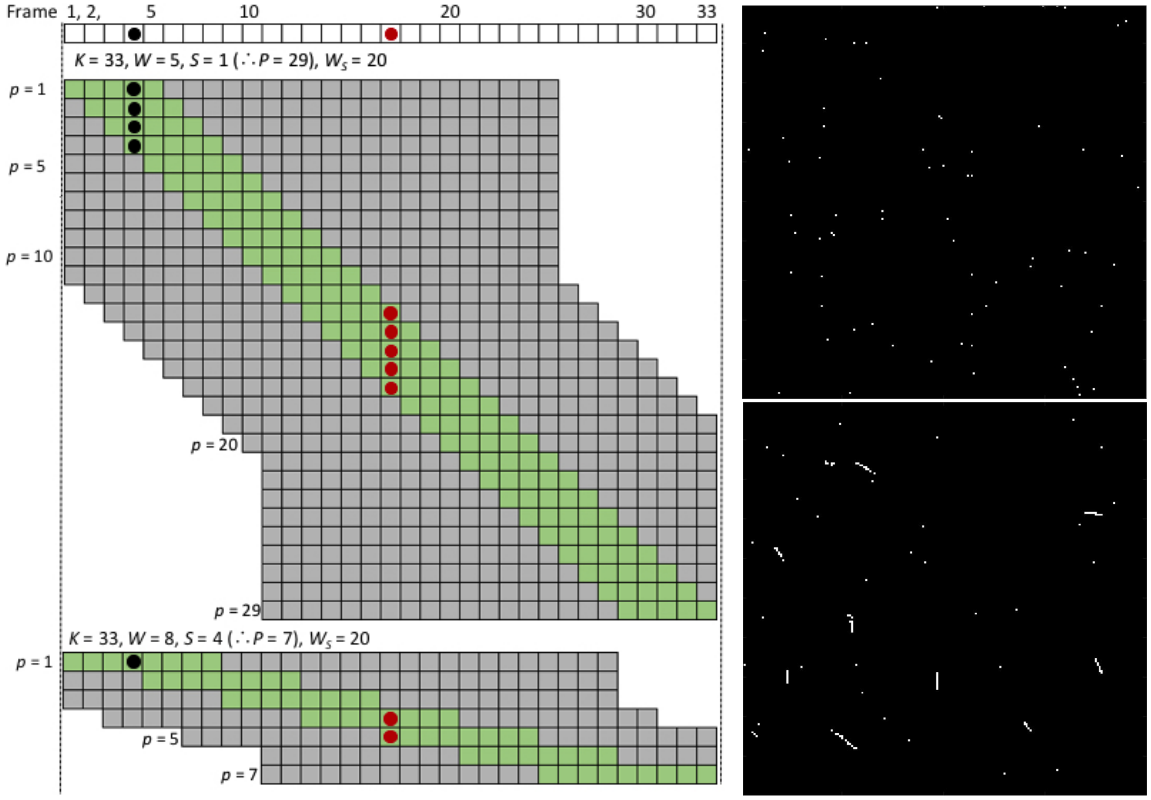
$$M^n_{r,c} = \frac{1}{W}\sum_{k=1}^{W}\left(x_{r_k,c_k} - \hat{\mu}_{B_{r,c}}\right)^n \tag{15}$$

where k is a temporal time step or frame, and $\hat{\mu}_{B_{r,c}}$ is the estimate of the background mean,

$$\hat{\mu}_{B_{r,c}} = \frac{1}{W_S}\left[\sum_{i=k-dk}^{k-1} x_i + \sum_{i=k+W+1}^{k+W+dk} x_i\right] \quad \text{with} \quad dk = \frac{W_S - 1}{2} \tag{16}$$

For cases at the beginning and end of the frame stack, the window is extended towards the center of the stack, keeping the same number of values but resulting in an off-center super window. Figure 32a is a graphic representation of how this calculation is done for a single spatial pixel with 33 frames, two different step sizes ($S = 1$ and 4), and two different window sizes ($W = 5$ and 8). The black and red dots represent targets that are present in the pixel at frames 4 and 17 respectively. Starting with the first pixel in the frame, the moment is computed over a temporal window of W frames, shown in green,

using the estimate of the background computed with the super window W_S, shown in gray, according to Equation 15. Next the window is slid S frames and the moment is computed for this new window, again using its associated super window. In order to perform this computation quickly and to minimize the number of calculations needed, the algorithm uses a convolution to compute $\hat{\mu}_{B_{r,c}}$ first, then the data is reshaped and replicated so that the moment computation can be done in just one line of code without repeating the calculation of $\hat{\mu}_{B_{r,c}}$. It was previously shown that choosing a step size larger than one could decrease computation time at the expense of detection performance [40], but with the implementation of a convolution approach to computing the moments, this increase in speed is no longer a significant factor and a step size of $S = 1$ should be used. The details of this convolution computation can be found in the moment function of the MBD algorithm in Appendix B. The result of this process on every spatial pixel is a new moments cube or moment frame stack containing a reduced number of frames, P. This process is repeated for each moment order, n, separately, resulting in the final four-dimensional hypercube of moment values. Due to the temporal windows, targets that originally only appeared in one pixel and frame now appear as streaks, as shown in the bottom half of Figure 32b. It is the spatially and temporally connected nature of these streaks that will be exploited to filter out many of the false detections.



(a) Method (b) Result

Figure 32. (a) Two examples of the temporal window moment computation used by the MBD algorithm on a single spatial pixel, each with different algorithm parameters. The two dots represent targets present in the pixel at frames 4 and 17. (b) Candidate detections from a standard DBT method (top) compared to candidate detections from MBD (bottom).

4.2.2 Detection.

Once the moments have been computed, the next step is to threshold the moment cubes in a way that maximizes detection for a desired false alarm level. The first step in this process is to compute the z-score for each moment pixel,

$$x^z_{m_{r,c}} = \frac{x_{m_{r,c}} - \mu_{r,c}}{\sigma_{r,c}}, \tag{17}$$

where the mean and standard deviation are computed temporally and independent of the other spatial pixels,

$$\mu_{r,c} = \frac{1}{P}\sum_{i=1}^{P} x_{m_{r_i,c_i}} \tag{18}$$

$$\sigma_{r,c} = \frac{1}{P}\sum_{i=1}^{P}(x_{m_{r_i,c_i}} - \mu_{r,c})^2. \tag{19}$$

The result of this process is a new $R \times C \times P \times N$ z-scores hypercube, where all values in the four dimensional array are z-scores instead of moment values. By computing the z-scores based on the temporal history of each pixel independently, each pixel is treated as an independent detector and all spatial and temporal pixels are on a similar scale. While some pixels will be better detectors than others, that is, a noisier pixel will not detect a target as well as an adjacent clean pixel, this approach allows for a single global threshold to be used to determine where detections are located. This threshold is set by computing the cumulative distribution function (CDF) using all the z-score values ($R \times C \times P \times N$). The CDF, denoted $F_n(x)$, gives the probability that a particular pixel in the z-score hypercube is less than or equal to x [38]. The last step in the detection process is to set the significance level, α_Z, based on the desired confidence level and use it to determine a threshold value. All pixels in the z-scores hypercube exceeding that threshold are considered cadidate detections and set to a value of one in the candidate detections hypercube while all lesser values are set to zero. For example in Figure 33, moments one and three were computed and the significance level was set to $\alpha_z = 0.99$. The latter is shown with the dashed line dropping from the red curve that represents the CDF, to the x-axis. In this case the z-score corresponding to the keeping of the top 1% of values is 2.46. This means that z-scores in the moment hypercube greater than 2.46 will be set to 1 (detects) while z-scores less than 2.46 will be set to zero.

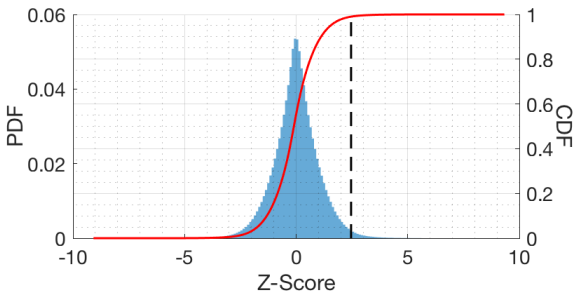


Figure 33. Histogram of z-scores using moments 1 and 3. The right axis and red line represent the cumulative distribution function and the dashed line represents the significance level $\alpha_Z = 0.99$

.

4.2.3 Filtering.

The final step in the algorithm is the filtering step. The temporal window approach to computing moments means that true targets will produce streaks that are connected in both space and time, while false detections will be more isolated. The voxel filtering employed in the MBD algorithm exploits this fact. First, the number of candidate detections in a sliding voxel of dimensions $d \times d \times d$ is computed. This results in an array of the same dimensions as the original candidate detections hypercube, but each element now contains the total number of detects in the voxel it is centered in. For example, for the case where $d = 4$, the maximum value in this new voxel array would be $4^3 = 64$, representing the case where every pixel in every frame within the voxel was a detect. Once this array has been computed, a second threshold is set based on the expected number of counts in a voxel for a real target. As before, values in the voxel array greater than the threshold are set to one and all others are set to zero. This creates a mask, like the one shown fourth from the left in Figure 34, where regions that have low numbers of detects are zeros and regions with higher numbers of detections are ones. The final step is to take the voxel array and multiply it by the candidate detections array. This has the effect of removing isolated candidate detections and leaving only those that are potentially streaks. This process is done separately on each moment in the candidate detections hypercube. The steps of this process are shown for a single

moment and frame of data in Figure 34.

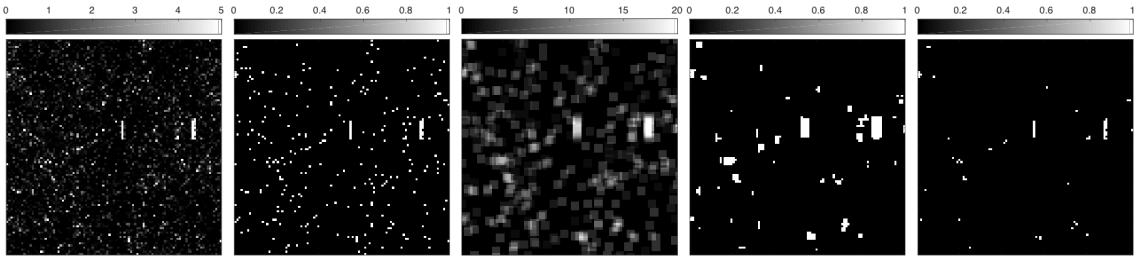


Figure 34. Filtering process frames from left to right, z-scores, detects after α_Z, voxel array, voxel mask after second threshold, and the final filtered detects for two targets with $R = 1$, $SNR \approx 3.5$. The algorithm parameters are $W = 9$, $S = 1$, $d = 4$, $\alpha_Z = 0.025$.

Within the filtering process are three main parameters that control the final maximum false alarm probability as well as impact the performance of the voxel filter. These parameters are the initial significance level α_Z, the second threshold creating the voxel mask, T_V, and the voxel dimension d. While the significance level and voxel dimension are parameters specified by the user, the second threshold is chosen based on a calculation of the expected number of detects in a voxel for a target with a given dwell time and signal to noise ratio, both of which are input by the user. A detailed discussion of these parameters and how they are chosen is presented in Section 4.4.3, but because they differ depending on the moment order, the whole filtering process is done independently for each moment. This results in N separate filtered binary detection cubes. The final step in the algorithm is to combine these using a logical 'or' operation. That is, a detect in any of the filtered N moments will result in a detect in the final combined detections binary data cube. While this does tend to increase the number of false alarms, it will be shown in Section 4.4.2 that the power of a moment to detect a target depends heavily on the target's speed and SNR. Combining moments in this way allows the MBD algorithm to work effectively for a broad range of target speeds and SNRs, making accurate *a priori* target information less critical to the implementation of the algorithm.

4.3 Parameter Optimization Approaches

Now that we have described how the MBD algorithm works, we turn to optimizing the various parameters and determining how target and scene characteristics impact those optimal choices. There are four main parameters that drive the performance of the MBD algorithm and that need to be chosen intelligently. These are the larger super window size W_S, the smaller temporal window size W, the moment order n, and the voxel dimension d. The optimal parameter choices will depend on a number of factors including the target characteristics, scene characteristics, and the other parameters that are chosen. A combination of theoretical development of equations for the expectation values of target and background window moments, simple simulations using Gaussian distributions, and ASSET simulations were used to optimize these parameters.

4.3.1 Theoretical Development.

To gain a clearer picture of how the various algorithm parameters relate and impact the calculated moment values, we need equations that give the expected moment values when a window contains only background, or *background window* moments, M_B^n and when a window contains both background and target, or *target window* moments M_T^n. Note that the spatial and temporal subscripts r, c, and k have been dropped here to simplify notation, but the expressions developed for M_B^n and M_T^n are general and apply to all pixels.

Equation 15 in Section 4.2.1 is the general form for the discrete moment of a temporal window of length W taken about an estimate of the background mean, $\hat{\mu}_{B_{r,c}}$. For a target window of temporal length W with target present in R of those W frames, Equation 15 becomes

$$M_T^n = \frac{1}{W}\left[\sum_{k\in k_0}(x_{b_k}-\hat{\mu}_B)^n + \sum_{k\in k_t}(x_{t_k}-\hat{\mu}_B)^n\right] \tag{20}$$

where R is the dwell time and is approximately equal to $1/v$ where v is velocity in pixels per frame. In cases where the target is moving in a straight line, R is equal to inverse velocity, but when the ground sample distance is large and the target is turning, this relationship is a minimum. A target can dwell within a single pixel for longer of it follows a less direct path across a pixel. In Equation 20 the first term is a summation over all frames in the temporal window containing exclusively background (notated $k \in k_0$) and the second term is a summation over frames containing signal from the target in addition to the background signal (notated $k \in k_t$). The two values x_{b_k} and x_{t_k} represent the raw digital numbers of the frame k for the two cases, $\hat{\mu}_B$ is the sample mean of the background window given by Equation 16, and n is the order of the moment. Similarly, for a background window,

$$M_B^n = \frac{1}{W}\sum_{k=1}^{W}(x_{b_k} - \hat{\mu}_B)^n. \tag{21}$$

At this point the expressions for the moment values are in terms of individual frame digital number values x_{b_k} and x_{t_k}. To simplify the problem and obtain more general expressions, we apply the assumption that the values in each window can be approximated with a Gaussian distribution. If we also assume that the sample background mean $\hat{\mu}_B$ is representative of the population background mean μ_B, we can use the formula for the central moment of a Gaussian distribution,

$$\langle M_B^n \rangle = \begin{cases} \sigma_B^n (n-1)!!, & n \in \mathbb{N} \text{ even} \\ 0, & n \in \mathbb{N} \text{ odd}, \end{cases} \tag{22}$$

to compute the expected value for a background window moment $\langle M_B^n \rangle$, where σ_B is the background standard deviation. For the target window case, if we let β represent the signal to noise ratio of the target and ε_k be a frame-dependent perturbation due to

noise drawn from a population with zero mean and standard deviation σ_B,

$$x_{b_k} = \mu_B + \varepsilon_k \tag{23a}$$

$$x_{t_k} = \beta\sigma_B + \mu_B + \varepsilon_k \tag{23b}$$

and Equation 20 can be rewritten

$$\langle M_T^n \rangle = \frac{1}{W}\left[\sum_{k\in k_0}(\mu_B + \varepsilon_k - \hat{\mu}_B)^n + \sum_{k\in k_t}(\beta\sigma_B + \mu_B + \varepsilon_k - \hat{\mu}_B)^n\right]. \tag{24}$$

Again assuming that the sample background mean is an accurate estimate of the population mean (or in the case that it is not, that any deviation can also be included in ε_k), the first term in Equation 24 goes to zero for odd moments and can be represented using the standard deviation of the background in the case of even moments,

$$\langle M_T^n \rangle = \begin{cases} \frac{W-R}{W}\sigma_B^n + \frac{1}{W}\sum_{k\in k_t}(\beta\sigma_B + \varepsilon_k)^n, & n \in \mathbb{N} \text{ even} \\ \frac{1}{W}\sum_{k\in k_t}(\beta\sigma_B + \varepsilon_k)^n, & n \in \mathbb{N} \text{ odd}, \end{cases} \tag{25}$$

where the $W-R$ term in the numerator represents the number of background pixels and replaces the summation. Finally, assuming the per frame perturbation ε_k is small and using a binomial expansion, the expected value for the moment of a target window is given by

$$\langle M_T^n \rangle = \begin{cases} \frac{W-R}{W}\sigma_B^n + \frac{R}{W}\sum_{i=0,2,4,\ldots}^{n}\binom{n}{i}(\beta\sigma_B)^{n-i}\sigma_B^i & n \in \mathbb{N} \text{ even} \\ \frac{R}{W}\sum_{i=0,2,4,\ldots}^{n}\binom{n}{i}(\beta\sigma_B)^{n-i}\sigma_B^i, & n \in \mathbb{N} \text{ odd}. \end{cases} \tag{26}$$

Here only even values of i are used because $\sum\varepsilon^i = \sigma_B^i$ for even i, and $\sum\varepsilon^i = 0$ for odd i due to the cancellation of positive and negative values.

In order to separate target windows from background windows, the goal is to maxi-

mize the separation between target and background windows, that is, maximize the expected moment difference $\Delta\langle M^n\rangle = \langle M_T^n\rangle - \langle M_B^n\rangle$. Subtracting Equation 22 from 26, and looking at the first five moment orders,

$$\Delta\langle M^1\rangle = \frac{R}{W}\beta\sigma_B \tag{27a}$$

$$\Delta\langle M^2\rangle = \frac{R\sigma_B^2}{W}(\beta^2+1) - \sigma_B^2 \tag{27b}$$

$$\Delta\langle M^3\rangle = \frac{R\sigma_B^3}{W}(3\beta+\beta^3) \tag{27c}$$

$$\Delta\langle M^4\rangle = \frac{R\sigma_B^4}{W}(1+6\beta^2+\beta^4) - 3\sigma_B^4 \tag{27d}$$

$$\Delta\langle M^5\rangle = \frac{R\sigma_B^5}{W}(5\beta+10\beta^3+\beta^5) \tag{27e}$$

From these expressions we can see that the expected moment difference depends on a number of factors including the noise in the scene (σ_B), the target speed and SNR (R and β), and the algorithm parameters (W and n). Up to this point these equations were derived using a fairly intuitive approach that relates the variables clearly to the physical scenario. For additional verification of the results presented in Equation 27, an alternate derivation using moment generating functions is presented in Appendix A. From these equations it is clear that, as expected, targets with longer dwell times and higher SNRs will be easier to detect and that increasing the window size will reduce the moment difference. However, these are expectation values, and variation in estimated moment values arising from finite sample size must also be taken into consideration. Just as targets with high signal compared to the background noise in a data set are easy to detect, when we take moments, optimal detection parameters will be those that maximize signal and minimize noise. As a starting place, we will quantify the uncertainty

in the background and target moment values using their associated variances,

$$Var(M^n) = Var\left[\frac{1}{W}\sum_{k=1}^{W}(x_k - \hat{\mu}_B)^n\right] \tag{28}$$

Note that $\hat{\mu}_B$ is an estimate of the background and x_k is a single spatial pixel in frame k that could contain only background or background plus target radiation. If we assume that the pixels x_k are independent and identically distributed, we can move the variance operation into the summation and rewrite Equation 28,

$$Var[M^n] = \frac{1}{W^2}\sum_{k=1}^{W} Var[(x - \hat{\mu}_B)^n] \tag{29a}$$

$$Var[M^n] = \frac{1}{W} Var[(x - \hat{\mu}_B)^n] \tag{29b}$$

but we know from expanding the definition of the variance of some random variable A, that $Var[A] = E[A^2] - E[A]^2$. Here our random variable is $(x - \hat{\mu}_B)^n$ and following the approach in [15] we have

$$Var[M^n] = \frac{1}{W}E[(x - \hat{\mu}_B)^{2n}] - \left(E[(x - \hat{\mu}_B)^n]\right)^2 \tag{30a}$$

$$Var[M^n] = \frac{1}{W}(M^{2n} - (M^n)^2). \tag{30b}$$

Now that we have expressions for both the signal (Equation 27) and the uncertainty in the moment window values (Equation 30b), we can write an expression for the new signal to noise ratio after the moment computation,

$$\langle SNR_{M^n}\rangle = \frac{\Delta\langle M^n\rangle}{\sqrt{\frac{1}{W}(M_B^{2n} - (M_B^n)^2)}}, \tag{31}$$

where the denominator is the standard deviation of the background moment, M_B^n found using Equation 30b. It is this signal to noise ratio that, when maximized, will produce

the best detection results for the MBD algorithm. Writing out the first 5 terms as before,

$$\langle SNR_{M^1}\rangle = \frac{\beta R\sqrt{W}}{W} \tag{32a}$$

$$\langle SNR_{M^2}\rangle = \frac{\sqrt{W}}{W\sqrt{2}}(R(\beta^2+1)-W) \tag{32b}$$

$$\langle SNR_{M^3}\rangle = \frac{R\sqrt{W}}{W\sqrt{15}}(3\beta+\beta^3) \tag{32c}$$

$$\langle SNR_{M^4}\rangle = \frac{\sqrt{W}}{4W\sqrt{6}}(R(1+6\beta^2+\beta^4)-3W) \tag{32d}$$

$$\langle SNR_{M^5}\rangle = \frac{R\sqrt{W}}{3W\sqrt{105}}(5\beta+10\beta^3+\beta^5), \tag{32e}$$

a several patterns are discernible. First, noting that the maximum value for R is always the window size (there cannot be more target frames than total frames in the window), the peak SNR will always occur when all frames in the window are target frames, or when $W = R$. Second, the SNR will fall off like $\sqrt{W}/W$ from that peak location as the window size is increased. Third, as expected, higher dwell times and target SNRs will result in higher expected moment SNRs. Fourth, the even moments will have lower moment SNRs than the odd moments due to the subtracted value that appears in both. This is because of the difference in even and odd background moments (Equation 22). For odd moments, $\langle M_B\rangle$ is zero but for even moments, $\langle M_B\rangle$ is related to the variation in the background. This decreases the expected moment difference and as a result lowers the moment SNR. Finally, there is a significant dependence on the SNR of the target, β. By setting the different moment SNRs equal to each other and solving for β, we can find the SNR values at which using a higher order moment will result in a higher moment SNR. For example, setting Equations 32a and 32c equal to each other, the dwell and window sizes cancel each other and the result is that for $\beta > 0.93$, $\langle SNR_{M^3}\rangle > \langle SNR_{M^1}\rangle$. Similarly, setting Equations 32c and 32e equal and solving for β, $\langle SNR_{M^5}\rangle > \langle SNR_{M^3}\rangle$ when $\beta > 1.85$. This suggests that for very dim targets, using a higher order moment

will not help because there is not enough signal to enhance, while for brighter targets, a higher order moment will amplify the signal better.

This information provides a starting place for further exploration of the impact and interaction of the scene, target and algorithm parameters that will be explored in Section 4.4, but these are only expected values. While the expression for variance derived in this section does hold when the underlying distribution is a mixture of Gaussians (as is the case for a target window), the sample distributions of higher order moments can be very skewed and as such are not well described by their associated mean and variance. As a result, a different approach is needed to quantify the true expected performance of the algorithm under a variety of target characteristics and algorithm parameters.

4.3.2 Gaussian Distribution Simulation.

A successful set of parameters in the MBD algorithm can be defined as one that separates target windows from background windows effectively. This means that it is not just the expected value of the moment SNR derived in the previous section that is of importance, but also the distributions of background and target moment values. To explore how various parameters impact this separation, background and target windows were simulated using Gaussian distributions. Figure 35 shows an example where the background is represented by a Gaussian distribution with a mean of 2150 and a standard deviation of 10, while the target distribution is the same, but shifted by 40 representing an SNR 4 target. To form background windows, W samples are pulled from the background distribution while to form target windows, a different set of $W-R$ values are pulled from the background distribution while R values are pulled from the target distribution. Once the windows have been created, the moment over the window is computed about an estimate of the mean, $\hat{\mu}_B$, with W_S samples pulled from the background distribution (as in equation 15). This process is repeated 10^5 times

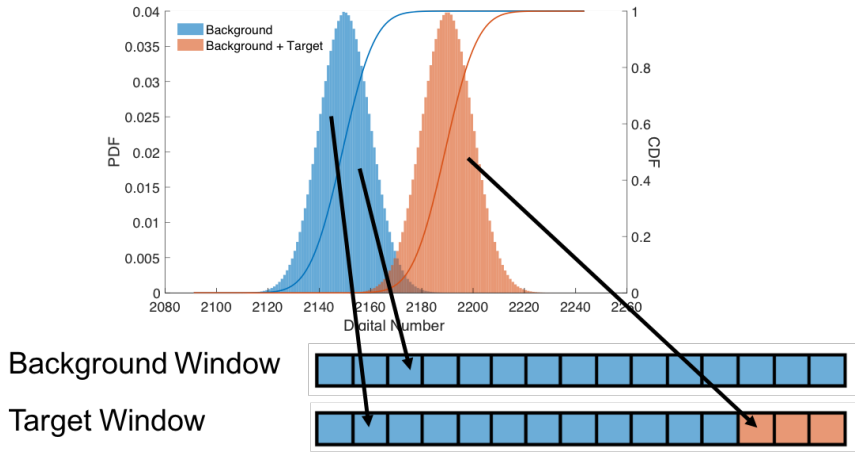
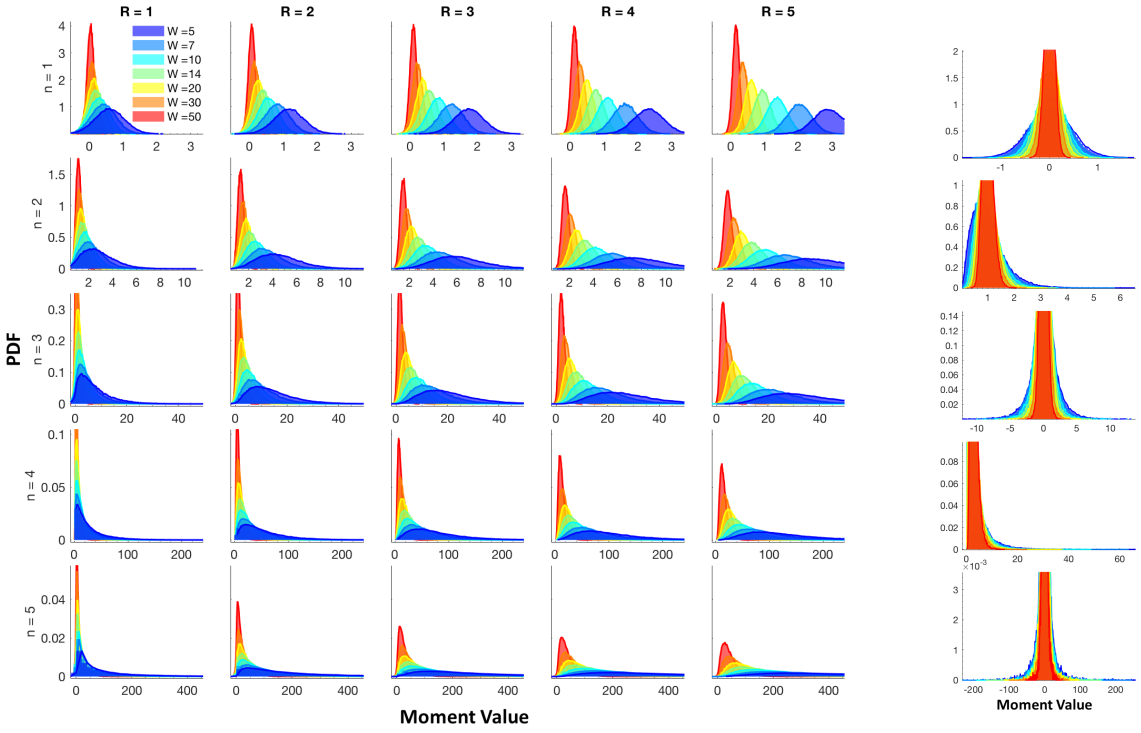


Figure 35. Gaussian simulation of target and background windows. The parameters for this example are $W = 15$, $R = 3$, $\sigma_B = 10$, and $\beta = 4$

for each group of parameters, varying the the target characteristics (SNR and R) and the moment order.

The result of this process is a new set of 10^5 target window and background window moment values for each set of parameters, revealing the *sample* mean, variance, skewness and higher order distributions. Figure 36 shows the first five target and background moments for a target with an SNR of 3 at five different dwell times. The different colors represent different window sizes (number of samples). Of note here are the shapes of the different distributions. Looking at Figure 36b, it is clear that in the background case, the distributions are fairly normal in appearance even at high moment orders, and as such can be described effectively with the expected values and variances derived in Section 4.3.1. In fact, the closed form expressions for the first two distributions are known, even for small sample sizes (namely the student's t distribution for sample mean and the chi-squared distribution for sample variance). However, the distributions in the target case (Figure 36a) have no such known expressions and can be very skewed, especially at small window sizes. This is of critical importance because in Section 4.3.1 it was shown that the moment SNR falls off with window size after $W = R$, indicating that small window sizes will generally detect targets better, eliminating the option of simply



(a) Target Moment Distributions

(b) Background Moment Distributions

Figure 36. a) Target moment window distributions for the first 5 moments and dwell times 1-5 $\sigma_B = 1$, and $\beta = 3$ and b) Background moment distributions for the first 5 moments with the same σ_B

using a large number of samples to obtain a normal distribution. Skewed distributions are not effectively described by their means and variances, so while we can compute these theoretically, a different metric is needed to truly understand the relationships between target characteristics, algorithm parameters, and likely detection performance.

Since the goal of a detection algorithm is the separation of target and background distributions from each other, a metric describing this separation is of great interest. Figure 37 shows the background (blue) and target (orange) distributions together on the same axes for a single case ($\beta = 2.5$, $R = 2$, $n = 3$, $W = 10$). The shaded black region represents the combination of both type I and type II errors. A type I error is a false positive (i.e. a pixel is shown as a detect in the absence of a target) while a type II error

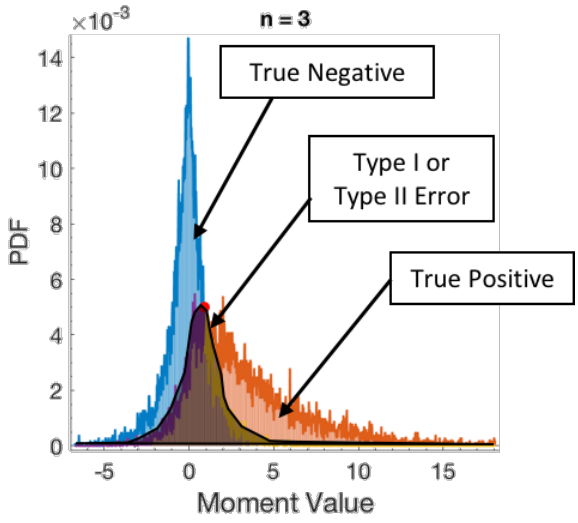


Figure 37. Background and target moment values for skewness with $\beta = 2.5$, $R = 2$, $W = 10$. The overlapping region represents all errors while the other two regions represent true positives and negatives

is a false negative, or missed detection (a pixel containing a target is not a detect). The other two regions are the true negatives and true positives. A true negative occurs when there is no target present and there is no detect, while a true positive occurs when a target is present and is correctly identified as a detect. In the case of perfect detection, there would be no overlap of the distributions and the combined error would go to zero. Therefore, in order to evaluate the performance of a set of parameters using a single metric, we compute the fraction of the total area that is contained in the shaded region and look to minimize it. This is done by finding the intersection of the two histograms (shown with a red dot in Figure 37) and integrating first from the left edge of the target distribution to the intersection (shown in purple), and then from the intersection point to the right edge of the background distribution (shown in yellow). The intersection was found by using the edge and value outputs of the histogram to define the curves, locating segments that intersect using [31], and then keeping only the intersection with the highest associated PDF value. The combination of these two areas represents the total error. Using this process, any set of target, scene and algorithm parameters can be

compared to another using a single value, allowing for a large number of cases to be looked at simultaneously. This is of particular interest when it comes to determining which moments should be used, as will be discussed in Section 4.4.2.

4.3.3 ASSET Simulation.

The Gaussian simulation in Section 4.3.2 is helpful for determining how various parameters increase or decrease the likelihood that a target will be detected as a result of increasing the SNR after computing the moments. However, it does not provide information on the potential impact of filtering based on expected target characteristics as discussed in Section 4.2.3. For this, an approach that involves targets with realistic motion across a range of speeds and SNRs was needed. To accomplish this, ASSET was used to generate scenes containing sixteen targets with the same median dwell times of R = 0.5, 0.75, 1, 2, 3 and 5 and median temporal SNRs of 2, 2.5, 3, 4 and 6 for a total of 30 scenes. Additionally, to maximize comparability across parameters, for each dwell time, all 6 scenes with varying SNRs have the same target paths. Figure 38 shows the target paths for each of these six speeds laid over a single frame of data. With these 30 scenes, MBD can be run using a variety of parameters and the results tabulated using the truth data available from ASSET, allowing for quantitative comparisons of different sets of algorithm parameters.

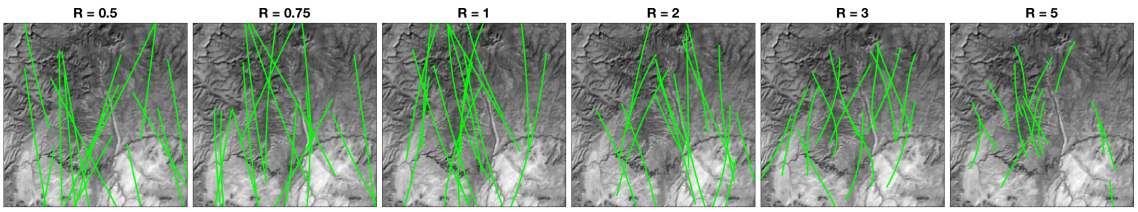


Figure 38. Paths for the six different target speeds over a single frame from six of the $200 \times 200 \times 300$ **data cubes generated for the ASSET simulation approach to parameter optimization. These paths are repeated at 5 different SNRs (changing only the target signal) for a total of 30 scenes.**

While the scenes look quite crowded with targets, rarely do targets actually overlap

in space and time due to the 300 frame time history. The large number of targets was necessary to obtain enough truth points to keep the error at a level that will provide meaningful results. Using this approach, all scenes contained at least 1000 truth points, a minimum number set based on the results shown in Figure 15 of Section 2.4.

4.4 Parameter Optimization

4.4.1 Window Sizes.

In previous versions of the moment based detection algorithm such as [40] and [43], only one window size was used to compute both the sample mean $\hat{\mu}_B$ and the moment value itself. That is, the mean used in the moment computation was simply the mean over the window W. This introduced an artificial limitation on the algorithm and led to a window size choice based on balancing the need for enough samples to provide a good estimate of the mean with the signal decrease associated with including too many background samples in the window. Implementing two separate window sizes removes this limitation by removing the need for this balance, allowing the super window size W_S to be chosen based entirely on the optimal number of samples for estimating the sample mean for a scene, while the smaller window size W can be chosen based on maximizing both the contrast between a target and background window and the length of the streak.

4.4.1.1 Super Window.

The super window is used solely to compute the sample mean of the background $\hat{\mu}_{B_{r,c}}$ given by Equation 16 in Section 4.2.1. If we assume that the time history of a pixel is Gaussian distributed with a population standard deviation σ_B, then the equation for the 95% confidence margin of error given a super window size W_S is given by $E = 2\sigma_B/\sqrt{W_S}$. If we define E in terms of the background standard deviation and a multiplicative factor,

$E = \sigma_B E_f$, we can rewrite this equation to get the necessary super window size for the sample background mean to be within E_f standard deviations of the population background mean. That is,

$$W_S = \frac{4\sigma_B}{E_f} \tag{33}$$

will give a sample mean that is within E_f standard deviations of the mean with 95% confidence. Figure 39 shows Equation 33 with a number of different background standard deviations plugged in. Clearly more samples will reduce the error and allow for

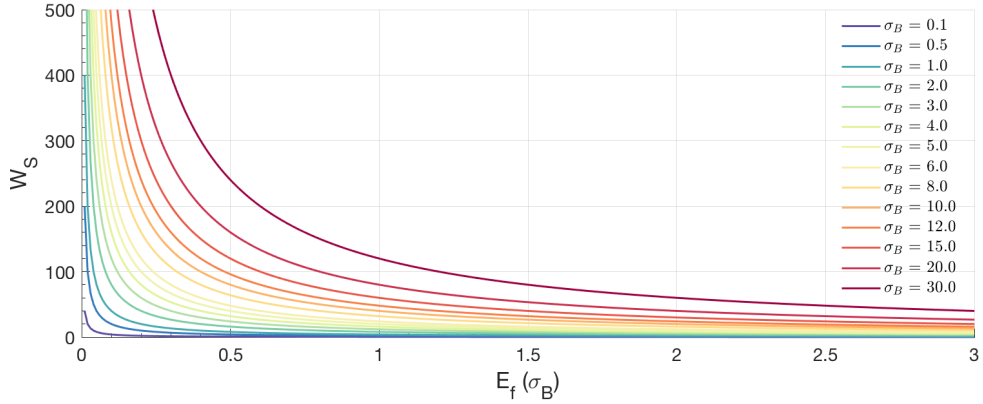


Figure 39. Super window size needed as a function of error in units of background standard deviation for 14 different background standard deviations

better detection, however this is derived using the assumption that the values in the super window are normally distributed and consistent over time. In cases where this tends to hold true, using a larger super window will improve the overall estimate of the background, however in cases where there is a lot of platform motion, bias drift, or clouds moving through pixels, using too many samples will make the sample mean less accurate. In these cases W_S should be chosen to be as large as the expected consistent number of frames.

4.4.1.2 Moment Window.

While the super window is chosen to best estimate the mean of the background in a pixel, the moment window W should be chosen to balance two factors: maximizing the contrast between target windows and background windows, and maximizing the length of the streak that is detected. Looking at Equation 32, the presence of the term $\sqrt{W}/W$ in the denominator makes it clear that increasing the window size will decrease the expected moment difference, $\Delta\langle M^n \rangle$. Figure 40 shows the moment SNR as a function of window size for four different dwell times and three different moment orders. The dashed lines represent the expected value using Equation 32 while the solid lines represent the mode of the distribution of moment SNRs from the simulation results in Section 4.3.2. The mode is used here as opposed to the mean or median because it is the most likely value and is a better representation than the mean or median when the distribution has a heavy right tail, as is the case for the target moment window distributions with small window sizes (as shown in Figure 36a). The shaded regions indicate 95% confidence bounds for the empirically estimated mode of the moment SNR from the simulation. These were computed by finding the 0.025 and 0.975 fractional quantiles for the target moment distributions numerically. As expected, due to the heavy right tail in the target moment window distributions, the mode is a lower moment SNR value than the expected value computed analytically. Additionally, as the moment order is increased, the uncertainty also increases rapidly, again as expected due to the heavy right tails of the distributions in Figure 36a. From the analytical development of the problem, the peak moment SNR is expected when $W = R$, and this result is visible in all four cases. If this were the only consideration, using a temporal window beyond the expected target dwell time would seem to have little benefit, however Equation 27 represents only the expected moment difference under ideal conditions, where the background estimate is accurate and all noise terms are very small, and does not take

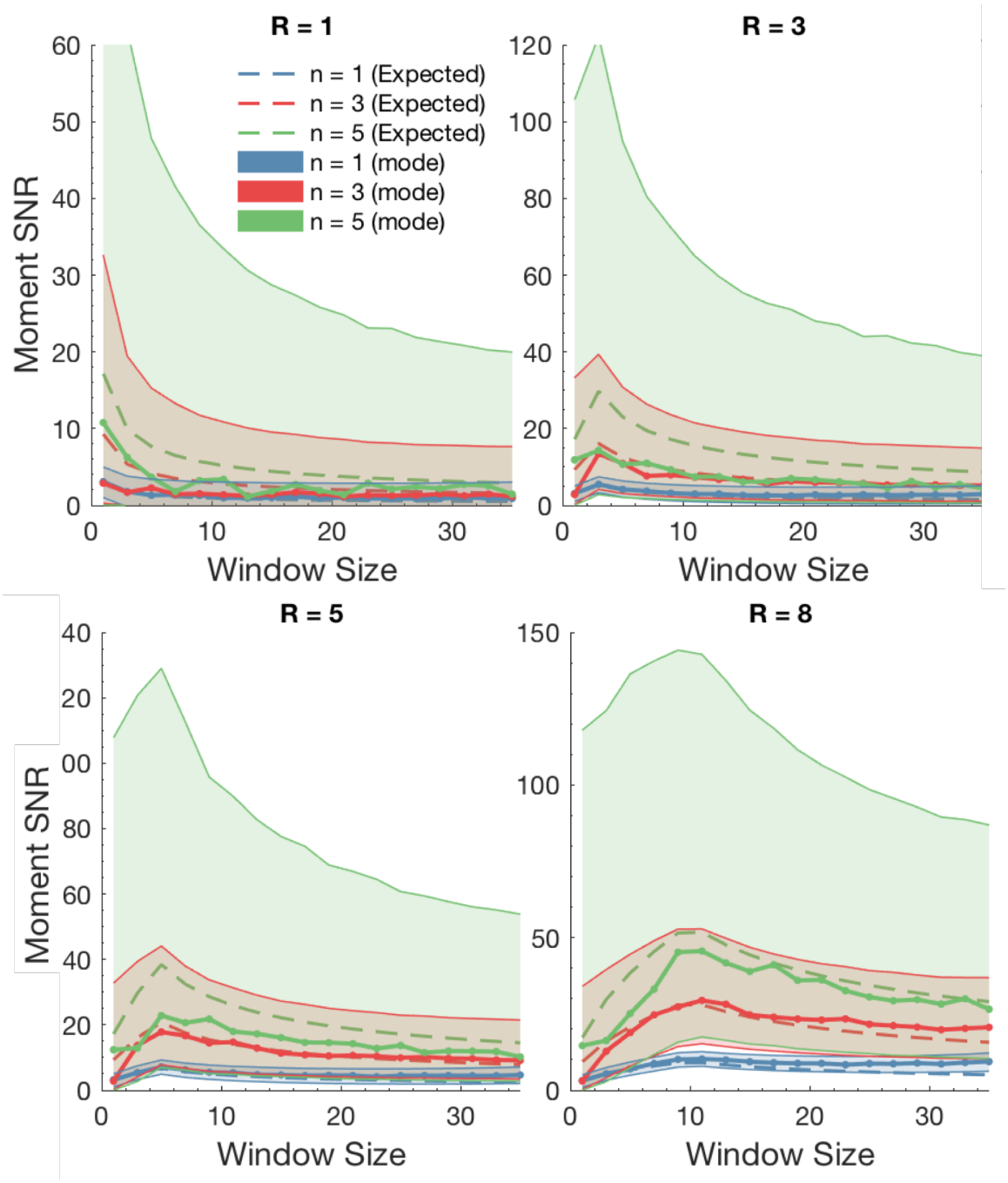


Figure 40. Moment SNR from from Equation 32 (dashed lines), using the mode of the distribution of moment SNRs based on the Gaussian simulation approach (solid lines) with 95% confidence bounds (shaded region) for an SNR 3 target.

into account the benefits of filtering. Choosing a larger window size will increase the length of the streak produced, trading off contrast between target and background

windows for streak length. Thus, the ideal window size for a target will be a balance between maximizing the streak length within a voxel and minimizing the wash out effect of adding additional frames after $W = R$.

The implementation of the voxel filtering introduces a maximum ideal window size, since extending the streak length beyond the diagonal of a voxel will no longer increase the detectability of the target. Thus, the first step in determining the optimal window size, W_{opt}, is to determine the expected spatial and temporal lengths of a streak for a particular window size and dwell time. The spatial streak length, L_s is the length of a streak present in a single temporal moment frame, while the temporal streak length is the number of moment frames associated with a target in a single spatial pixel. To simplify the problem we will assume the target is moving in a straight line over the dimension of the voxel and start by considering the distance traveled y for a linear target in continuous space, $y = v\,t$ where v is the velocity and t is time. Noting that the distance traveled will be the spatial streak length L_s, the velocity is the inverse of dwell, and the time is the temporal window size W, this equation becomes $L_s = W/R$. However, the discrete nature of the problem means that both time and distance are integer values.

Consider the two examples of targets moving in space (represented by the boxes) and time (represented by the colored dots) in Figure 41. In the example on the left, the dwell time is 4 so the target moves one quarter of a pixel per frame. This means that for a window size of ten, it will have moved from the first red dot, to the second red dot, a total distance of $\frac{W-1}{R} = 2.25$ pixels. For some starting points within the pixel, this will result in the target being present for 4 total pixels (the yellow and gray cases) while for others it will only appear in 3 pixels. For cases where the target is moving faster than one pixel per frame ($R < 1$) like the example on the right, the total number of pixels it appears in for a single moment frame will be W, but in some cases these pixels will not

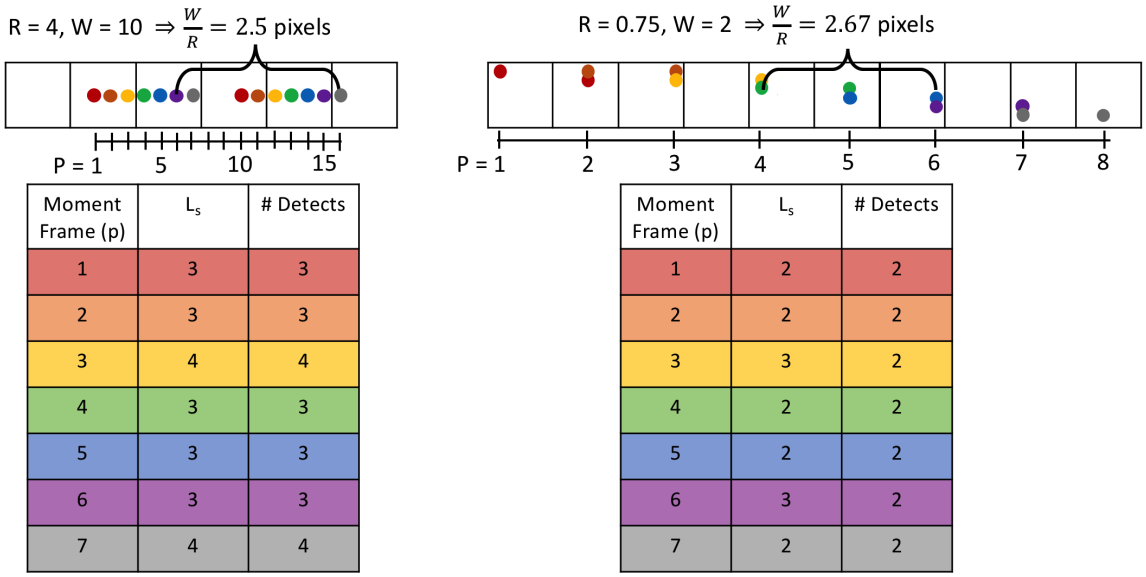


Figure 41. Graphical explanation of the spatial streak length L_s for two different scenarios. Boxes represent spatial pixels, colored dots represent the target position at the beginning and end of a temporal window, and the different colors represent the temporal moment frames p.

be adjacent (as in the yellow moment frame three and purple moment frame 6). For the purpose of determining the optimal window size, it is the extent of the streak that matters, so we consider only L_s and not the number of detects. Now the spatial streak length is

$$L_s = \left\lceil \frac{W-1}{R} + 1 \right\rceil, \tag{34}$$

where the ceiling operation is used to ensure this is an upper limit on the streak length. For the temporal case, since it is the number of frames associated with a target in a single spatial pixel, we multiply Equation 34 by the dwell time to get the temporal streak length,

$$L_t = \left\lceil \frac{W+R-1}{S} \right\rceil = \lceil W+R-1 \rceil, \tag{35}$$

where S is the temporal step size and is set to 1 as discussed in Section 4.2.1. Since the filtering voxels are cubic and we generally want to maximize the fraction of a voxel that is expected to contain a target, we will use the smaller spatial dimension to compute

the optimal window size. This avoids adding additional spatial pixels to the filtering voxel that will not contain a target. Setting $L_s = d\sqrt{2}$ for the diagonal of the spatial box with side length d used for filtering, and solving for W,

$$W_{opt} = [R(d\sqrt{2}-1)+1]. \tag{36}$$

Equation 36 will give the optimal window size in cases where the target is bright enough to remain detectable when the window size is increased past $W = R$. Figure 42 shows this optimal window size as a function of dwell time and voxel dimension. While the

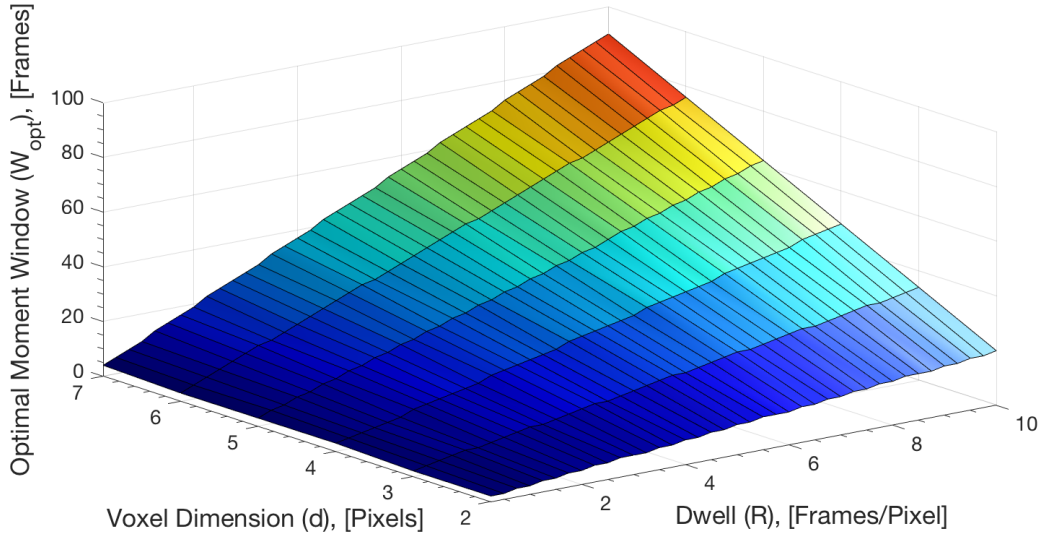


Figure 42. Optimal window size as a function of voxel dimension and dwell time based on equation 36

target's speed cannot be controlled, the voxel dimension is a user parameter and can be adjusted to help obtain an optimal window size that will maximize the target signal. This will be discussed further in Section 4.4.3.

To verify Equation 36, the approach described in Section 4.3.3 was used, varying the window size, voxel dimension, and significance levels to create ROC curves for each scene and set of parameters. To better compare the results, the probabilities of

detection at a number of false alarm points were examined. Figure 43 shows these results for a false alarm percentage of 1.25 ($P_{FA} = 0.0125$) for all six dwell times and two different SNRs using a voxel dimension $d = 3$ (which will be discussed in Section 4.4.3) for moments one, three and five.

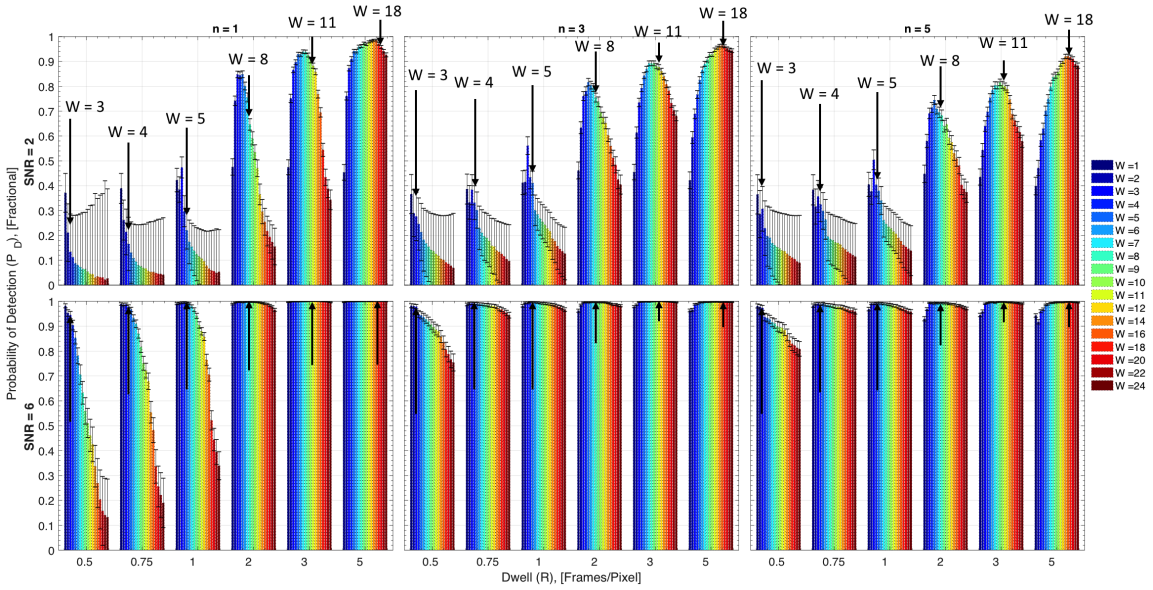


Figure 43. Probability of detection at a single false alarm level of 1.25% for six different dwell times and four different SNRs using moments $n = 1, 3$ and 5 with $d = 3$ for a range of window sizes.

From this figure it is clear that the optimal window size depends heavily on the dwell time as expected. Additionally, while there is some variation in the optimal window size across moments, these are also fairly consistent. The theoretical optimal window sizes from Equation 36 are represented by the arrows in the figure. For the brighter scene (second row), the predicted optimal window sizes are generally accurate for the slower targets ($R \geq 1$), but they are over predictions for the faster targets. This is as expected based on the theoretical and Gaussian simulation approaches. Looking at Figure 40, the moment SNR falls off faster after the $W = R$ point with window size for targets with shorter dwell times. This means that a smaller window size is needed to maintain a high moment SNR for the target to be detected. For the dimmer scene, only the slowest target has a peak window size that matches well with the prediction. For the rest of the

targets, Equation 36 is consistently over predicting. This is likely because for dimmer targets, the washout effect of using additional frames past $W = R$ is dominating, and it suggests that an adjustment must be made to the window size when looking for fast targets with low SNRs. Specifically, for very fast targets with an SNR less than six and a dwell time of 0.5 or less, a window size of one should be chosen because the target is not bright enough to be detected in the skipped adjacent pixels, significantly reducing the number of detects within a voxel. When the number of skipped pixels decreases, as is the case for $0.5 < R < 1$, the window size can be increased slightly for targets with SNRs of about 3 to 6. Once the target is bright enough that these skipped pixels are detects, the optimal window size equation becomes accurate. For targets with a dwell time $R = 1$, no pixels are skipped, so a slightly larger window size should be chosen.

Figure 44 shows the results from the ASSET simulation for $n = 3$ and all five SNRs at the two middle dwell times, $R = 1$ and $R = 2$. The optimal window sizes from Equation 36 for these two cases are $W_{opt} = 5$ and $W_{opt} = 8$, indicated by the red arrows. The green arrows indicate which window size actually has the highest probability of detection. It is clear that for $R = 1$, the target would need to be very bright for Equation 36 to give the true optimal window size. In this case the streak length cannot be extended significantly without reducing the moment SNR to the point where it cannot be detected, and a smaller window size of $W = 3$ should be chosen instead of the value suggested by Equation 36. For the $R = 2$ targets, the same explanation holds and a smaller window size of $W = 5$ should be chosen for a target SNR less than 3. For the brighter targets, while there is a difference between P_D at $W = 7$ and $W = 8$, it is not significant and Equation 36 can be used to compute the window size. Table 6 summarizes these results.

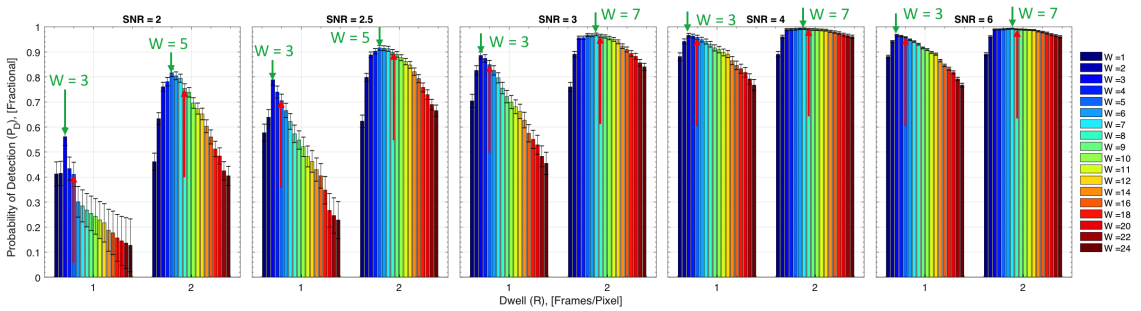


Figure 44. Probability of detection at a single false alarm level of 1.25% for two dwell times and five SNRs using $n = 3$ and $d = 3$. The red arrows show the theoretical optimal window size to maximize streak length from Equation 36 while the green arrows and labels indicate which window size produces the highest probability of detection.

Table 6. Window size for different target dwell times and SNRs. W_{opt} indicates the value can be calculated by inputting the dwell and voxel dimension into Equation 36

SNR \R	≤ 0.5	(0.5,1)	$[1-2)$	$[2-3]$	>3
<3	1	1	3	5	W_{opt}
$3-6$	1	3	W_{opt}	W_{opt}	W_{opt}
>6	W_{opt}	W_{opt}	W_{opt}	W_{opt}	W_{opt}

To test how accurate the recommendations in this table are, the optimal window size was computed based on Table 6 for all of the 30 scenarios run and a voxel dimension of 3. For each SNR and dwell time pair, the window size that produced the highest probability of detection is shown in Table 7. The probability of detection at this window size and a false alarm level of 1% was found, along with the probability of detection at the window size predicted by Table 6. The percent difference in these values was computed and used for the color code. A green cell indicates this value was less than or equal to 1%, yellow indicates a value between 1% and 5%, orange indicates a value between 5% and 10%, and red indicates a larger difference.

Table 7. Optimal window sizes based on Table 6 color coded by their accuracy.

	n = 1						n = 3					
SNR\R	0.5	0.75	1	2	3	5	0.5	0.75	1	2	3	5
2	1	1	3	5	7	14	1	3	3	5	9	16
2.5	1	1	3	5	9	16	1	3	3	5	9	18
3	1	1	3	5	5	16	1	3	3	7	9	18
4	1	1	3	6	10	14	1	3	3	7	12	18
5	1	1	4	6	10	24	1	3	4	7	12	12

In the next section the relationship between moment order and dwell time will be discussed, but to put the results in Table 7 in context, $n = 1$ will be used for slow targets and $n = 3$ will be used for fast targets. With this information, there are only two orange cells, (the SNR 2.5, R = 0.75 and SNR 3, R = 1 cases) in relevant positions. The remaining cells of interest are green and yellow, indicating that this is an acceptably accurate way of determining the appropriate window size. While the optimal window sizes are theoretically the same across moment orders, the rate of change in P_D as a function of window size depends on both the target dwell time and the moment order, suggesting that while the detection performance for the optimal window size is approximately the same across moments, there are more factors to consider.

4.4.2 Moment Order.

Now that the two window sizes have been explored, the relationships between the moment order and relevant target and algorithm parameters are needed. To explore these, both the theoretical approach of Section 4.3.1 as well as the Gaussian distribution simulation approach of Section 4.3.2 will be used. First, looking at Equation 32, the different moment orders can be set equal to each other and solved for β, resulting in the expected target SNR at which one moment would be expected to out perform another. For example, setting Equations 32a and 32b equal to each other and solving yields $\beta = 0.5$, which indicates that on average, $n = 2$ will detect targets with an $SNR > 0.5$

better than $n = 1$. However, this is only in terms of the expected SNR, and as we saw in Figure 36a, the expected target moment value often will not be a good indicator of the actual target moment value in a real scenario with a small window size. This in turn means that the expected moment SNR is also not giving the full picture as to the relationship between moments. To better explore this, the method of Section 4.3.2 was used. Figure 45 shows the results of running such a simulation with a range of SNR and window sizes for four different dwell times. The z-axis here represents the probability of a true outcome, or in other words, the sum of the probabilities of a true positive and a true negative. Each star represents a single window size; SNR combination and the colored surfaces are a simple surface fit to these data points for moments 1-9. These plots confirm the initial result of the previous section (that the optimal window size prior to filtering will occur at $W = R$), but they also show that for brighter targets, it is possible to use a larger window size without decreasing the probability of a true outcome by using a larger odd moment order. Additionally, the color that is visible in each surface plot indicates which moment will produce the highest probability of a true outcome for the corresponding window size, target SNR and dwell.

In the previous section the optimal window size was determined including the impact of filtering based on spatially and temporally connected streaks. Using a filtering voxel with dimension $d = 3$ and assuming dim targets with $SNR_T < 3$, these values for $R = 1, 2, 3$ and 5 are $W = 3, 5, 11$ and 18 and are represented by the blue surfaces in the plots. In the case of the last three plots ($R = 2, 3$ and 4), the surfaces are green and transition to red as the SNR is decreased. This indicates that for brighter targets, using the 3rd order moment will produce better results while for dimmer targets, using the mean will work better. For the $R = 1$ case, a similar transition occurs but this time between the 5th order moment and the 3rd. The even moments produce consistently lower detection results as expected based on the theoretical development, and should

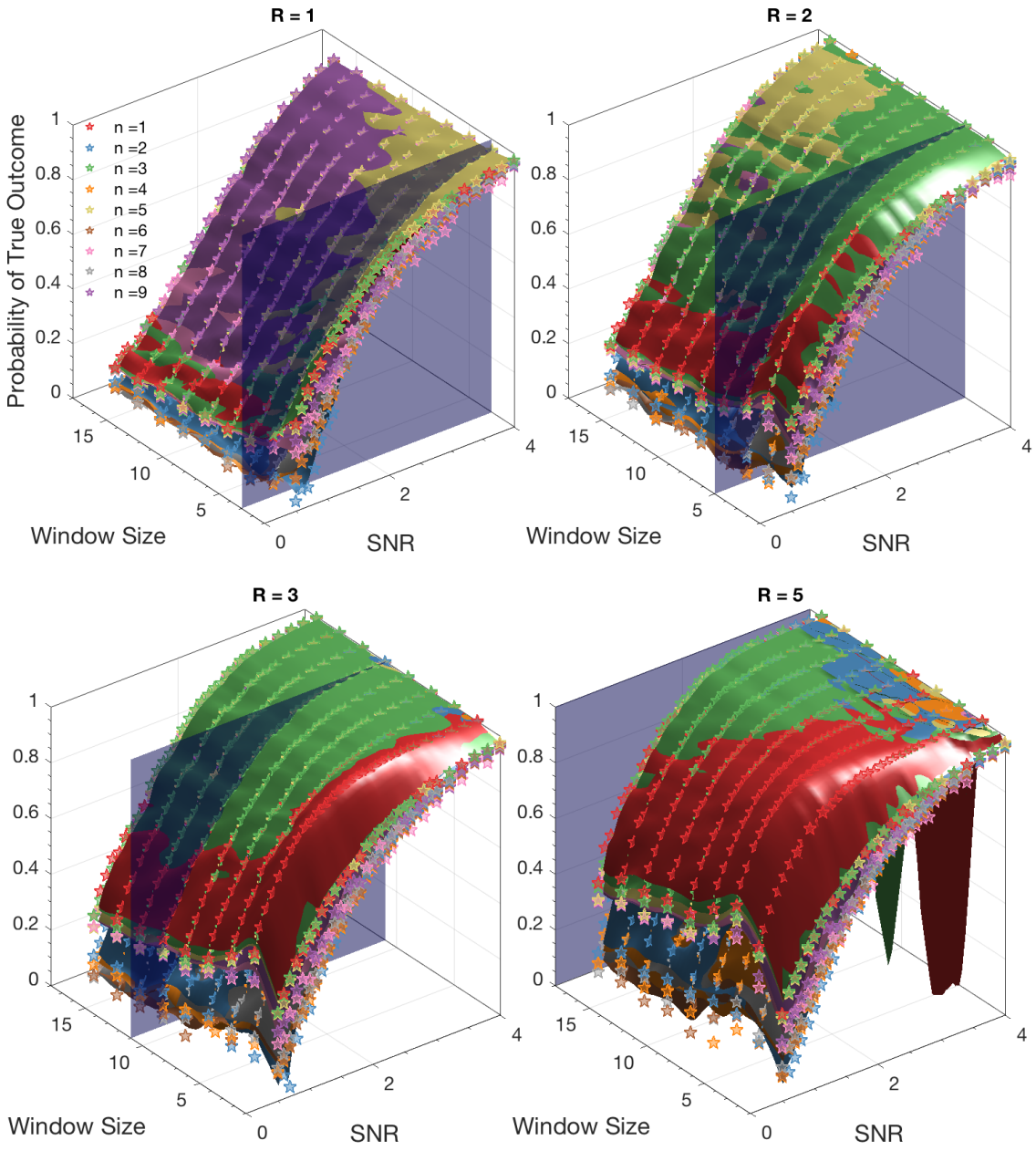


Figure 45. Probability of a true outcome surface fits for the first nine moment orders as a function of moment window size W and target SNR for four different dwell times. The blue surface represents the ideal window size for a voxel dimension $d = 3$ based on table 6.

not be used.

These conclusions are drawn solely from the Gaussian simulation. To verify these conclusions and provide additional insight into the appropriate choice of moment order, the ASSET simulation approach was also used. In Section 4.4.1.2 the probability

of detection was found at a constant false alarm level for a range of window sizes and this was used to determine the optimal window size when filtering using the voxel filter. The results shown in Figure 43 also contain important information with respect to choosing a moment order. Figure 46 is the same type of plot as before, but for SNRs of 2.5 and 4 and only moments 1 and 3. Moment 5 was not included because there is no statistically significant difference between the detection performance of the higher odd moments. While theoretically increasing the moment order will increase the moment SNR, the corresponding increase in uncertainty in the target moment SNR negates this benefit. However, there are significant differences between using the mean ($n = 1$) and using the skewness ($n = 3$). The red lines in Figure 46 indicate the linear portion of

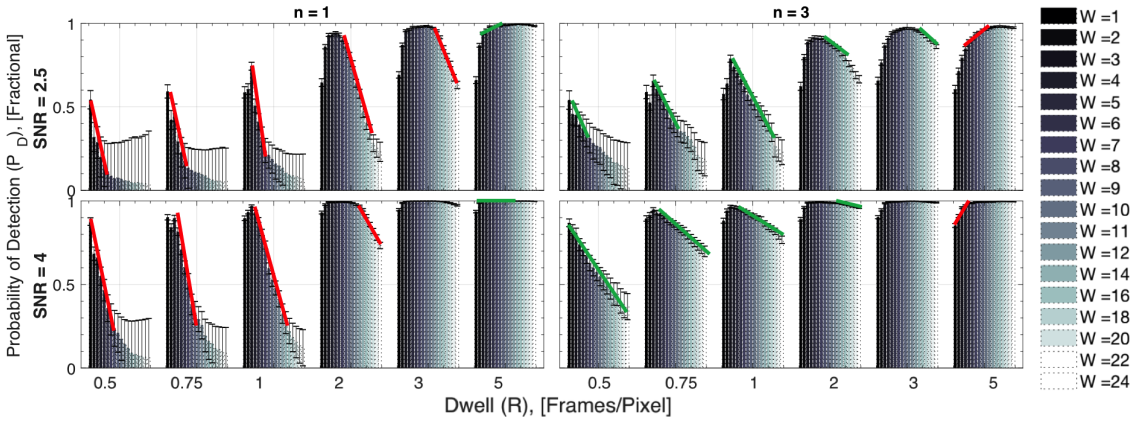


Figure 46. Probability of detection at a 1.25% false alarm level for two SNRs and six different dwell times. The lines highlight the linear portion of the change in P_D as a function of window size.

the decrease in P_D as the window size is changed. For the fastest 4 targets, the slopes are favorable for the 3rd order moment, while the $R = 3$ cases are quite similar, with only the lower SNR showing a less rapid decrease than the mean. For the slowest target, the slopes are steeper for the 3rd moment, indicating that the mean is actually less sensitive to an incorrect window size choice. Examining the target and background sample moment distributions in Figure 36 of Section 4.3.2 provides insight into why this is the case. For all odd moments, the background distributions remain centered at zero

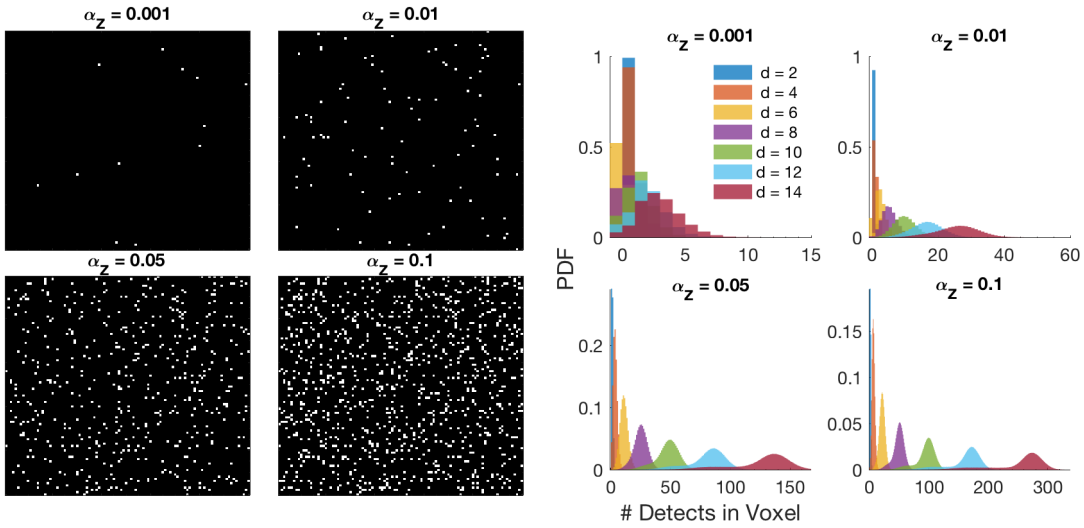
and become more peaked as the moment order increases. For the mean, short dwell times keep the target sample moment distribution fairly normal with a peak not far from zero, and the overlap of the background and target cases is significant. However, for longer dwell times, the peak of the sample moment distribution shifts away from zero significantly while maintaining a fairly Gaussian shape. The result is that now the target and background distributions overlap significantly less, making the mean an effective metric to use. For faster targets, the higher order odd moments have a heavy right tail that allows the targets to be amplified more than the noise, which has a much more peaked distribution centered at zero. This insensitivity to window size is an important factor because the target's speed is often only known to within an approximate range and not exactly, and while an assumption about the general brightness of a target can be made, its true SNR is also frequently unknown. In order to account for this level of uncertainty when using the algorithm, multiple moments can be used in combination, allowing a detection to be in any single moment and thus reducing the algorithm's dependence on accurate *a priori* target information. For slow targets with dwell times greater than about $R = 2$, a combination of moments 1 and 3 should be used, while for faster targets only moment 3 is needed.

4.4.3 Filtering Parameters.

The final set of MBD parameters to optimize are those associated with filtering. There are essentially three filtering parameters: the voxel dimensions d, the initial significance level α_Z which controls the initial threshold set to produce candidate detections, and the secondary threshold T_V, the minimum number of detects in a voxel for that voxel to be kept. Because the initial significance level controls the maximum false alarm probability in any given moment, it's optimal value will depend on the acceptable false alarm probability for a given application. As such, the other two parameters will

be optimized after a choice for α_Z has been made.

The number of detects within any given voxel is the number of detects due to noise plus the number of detects due to a target. In order to separate voxels that are purely noise from those containing target energy, the expected number of detects from each source is needed. This number of detects will depend on the initial significance level α_Z and the voxel dimension d. Figure 47a shows the detects due to noise in a single 100×100 frame of data for four different significance levels while Figure 47a shows probability normalized histograms of the number of detects in a voxel for 7 different voxel sizes and 4 different significance levels. The problem of counting noise detections



(a) Noise (b) Voxel Sums

Figure 47. (a) Noise in a single 100×100 frame with significance level α and (b) the distribution of sums within a voxel of volume d^3 with significance level α_Z

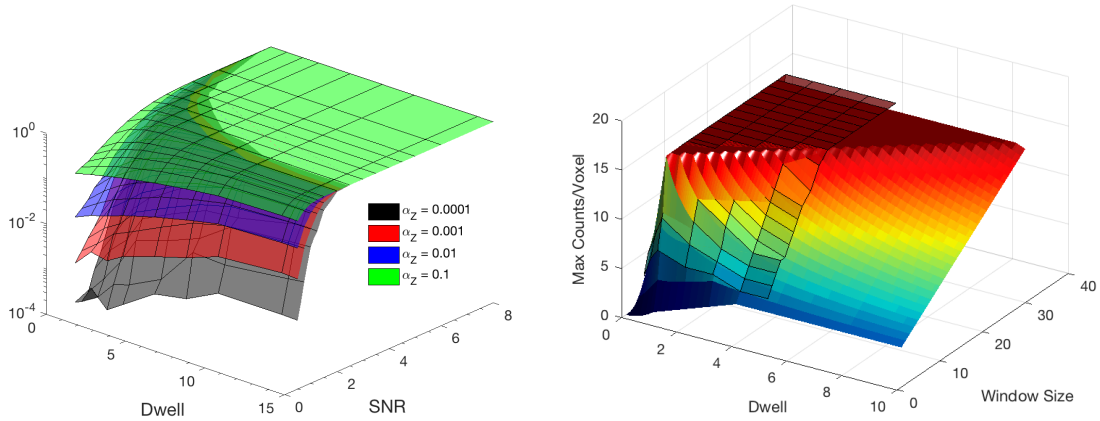
within a voxel can be framed as a sequence of d^3 independent experiments with a true or false result, and thus the distributions in Figure 47b are binomial. The number of detects due to noise can then be found by finding the inverse of the binomial cumulative distribution function with d^3 samples at some probability threshold. Choosing this threshold to be 0.5 would give the average noise in any given voxel, but since the goal of the filtering process is to remove voxels that do not have a high enough density of

detects to contain a target, this probability threshold should be set very low. The default in the MBD algorithm is 0.05, but this can be increased to filter more aggressively at the risk of removing voxels that contain a target but also have lower than average noise.

Now that the noise contribution to a voxel has been determined, it is necessary to determine the minimum expected number of detects in a voxel from a target. To determine this value, two components are needed: the streak length produced by a target with a given speed, and the fraction of the streak that is expected to be detected under a given set of algorithm parameters.

To determine the latter parameter, we use the approach outlined in Section 4.3.2 to generate the PDFs and CDFs for background moments and target moments, like the example shown in Figure 35. These CDFs were then used to generate ROC curves as described in Section 2.4. Using this approach, ROC curves for any given set of target and algorithm parameters can be computed. To help speed up the algorithm, a large set of these parameters were run and stored in a large table with the algorithm code. Figure 48a is an example of the data obtained from this approach. It shows probability of detection as a function of SNR and dwell time for three different significance levels for a window size of 8 and the third order moment. As expected, higher SNRs and significance levels result in a higher probability of detection and increasing the dwell time increases probability of detection up until $R = W$, or in this case $R = 7$. Additionally, the smoothness of these surfaces indicates that in cases where the desired parameters do not match exactly with the pre-run values, these results can be interpolated to find an approximate value.

Now that we know the probability of detection as a function of target and algorithm parameters, if we know the number of detects in a voxel under perfect detection conditions, we can multiply that value by P_D to determine a filtering threshold. This was done in two different ways, first mathematically and second using the ASSET simulated



(a) Probability of detection

(b) Target detects per voxel

Figure 48. a) Probability of detection as a function of SNR and dwell time for skewness with a window size of 7 at three different significance levels b) Maximum number of counts/voxel as a function of dwell and window size computed using Equation 37 and via simulation (lines)

data. There are 3 parameters that will impact the number of detects in a voxel: the voxel size d, the window size W, and the target dwell time R. Since we are attempting to derive a minimum threshold for the number of detects in a voxel, we will consider the case of a target that moves in a straight line through the voxel, as this represents the shortest distance across the voxel and will therefore be the minimum number of ideal detects for a given set of parameters. In order to determine the number of detects in a voxel, the lengths of the streak both spatially and temporally are needed. These were derived in Section 4.4.1.2 and are given by Equations 35 and 34. However, these are based on the extent of the streak and include potential skipped pixels for targets that move more than one pixel per frame. To account for this, if a target has a dwell time less than one, $L_s = W$ instead of the previously derived equation. This gives the actual number of detects instead of the extent of the streak. This is necessary because the goal of the voxel filtering is to remove voxels that do not contain a target due to a low density of pixels. Fast targets will inherently have a lower density of detects in a voxel, so without this adjustment they would be far more likely to be filtered out incorrectly.

With this modification, the total number of detects in a voxel can be computed as,

$$\text{Target Detects} = \begin{cases} L_s \times L_t & L_s, L_t < d \\ d \times L_t & L_s > d, L_t < d \\ L_s \times d & L_s < d, L_t > d \\ d^2 & L_s, L_t > d, \end{cases} \tag{37}$$

where the piecewise function is needed to cap the temporal and spatial streak lengths at the voxel dimension.

To verify this calculation using an ASSET simulation, the truth data for scenes with targets moving in straight lines at various speeds were used to create binary truth cubes. That is, for each set of parameters, instead of running MBD, the truth data was processed in the same manner, producing a binary data cube with a one where the target is expected in the output of MBD, and zeros elsewhere. This cube was then put through the voxel summation in the MBD algorithm, resulting in a new cube containing the voxels for perfect detection. In each case, the maximum number of detects in a voxel was found and stored. Figure 48b shows an example of these results for a voxel size $d = 4$ as a function of dwell time and window size while Figure 49 shows an example of what this calculation looks like visually for a single target. In Figure 49 blue represents truth after MBD processing (the denominator of the formula for P_D), green represents truth in the raw data cube (the true target position), and the color map shows how many counts there are in a voxel with $d = 4$ for this target. It is clear that the true target positions overlap with the largest values in the summation. This is as expected since the voxel sum is done by sliding one pixel at a time, resulting in voxels around the edges capturing only a small portion of the target. It is for this reason that the maximum value from the voxel summation of the truth data is used in the final calculation.

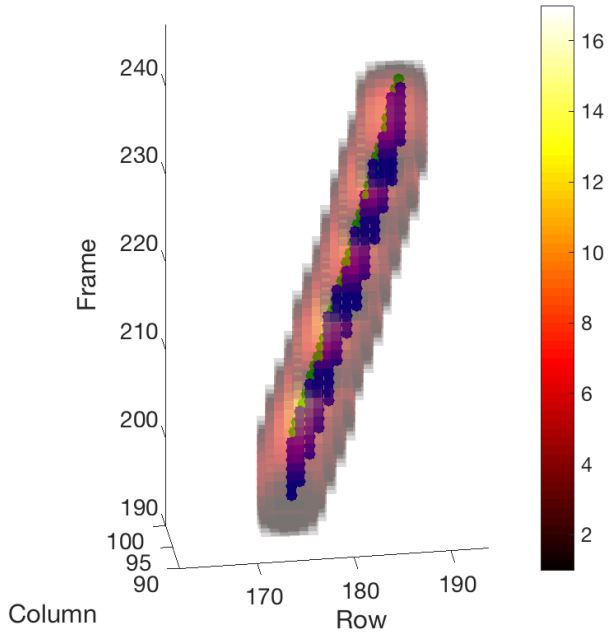
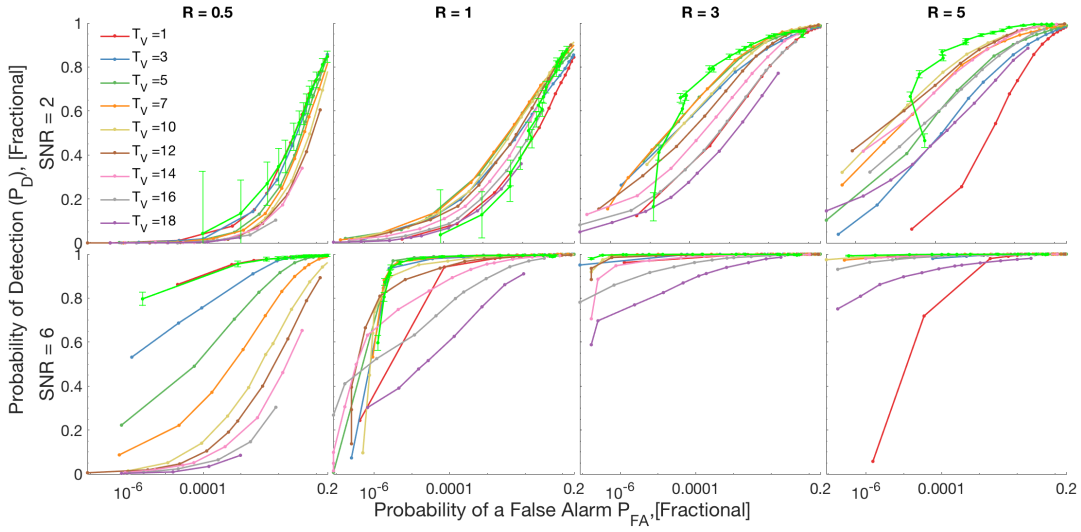


Figure 49. Result of MBD and voxel processing for single target with a dwell time of R = 3 using a window size of 7 and a voxel size of 4. The color map represents the number of detects in a voxel for a perfectly detected target. Green indicates the true target position while blue indicates perfect detection using MBD.

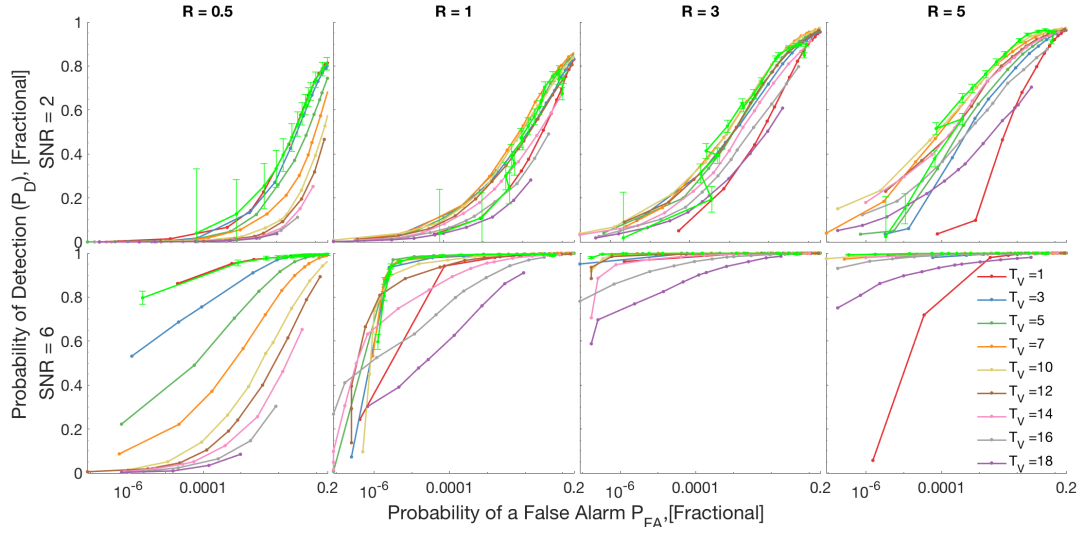
The next step is to multiply the probability of detection by the maximum expected number of detects per voxel to get the expected number of detects due to a target within a voxel. In scenarios where the initial threshold α_Z is small, using this value for T_V would work well, but as the initial threshold is raised, the number of false detects in a voxel also increases. To account for this, the final filtering threshold is obtained by adding the expected number of additional false detects due to noise discussed previously to the expected number of detects due to a target. While the filtering step does require inputs for the target dwell time and SNR, these are only best estimates and should be set at the low end of the range of targets of interest. This results in less aggressive filtering but is less likely to eliminate a real target of interest by accident.

To verify this approach to automatically choosing the threshold T_V, the ASSET simulation approach was used, running the algorithm first with the automatically chosen

threshold for T_V, and then again but this time varying T_V. Figure 50 shows a sampling these results for four different dwell times and the lowest and highest SNRs. For the



(a) $n = 1$



(b) $n = 3$

Figure 50. ROC curves using different filtering thresholds T_V plotted with the curve using the automatic filtering approach (green) for $n = 1$ (top) and $n = 3$ (bottom). Error bars represent 95% bounds using equation 7

bright target, the automatically chosen filtering curves are equal to or outperforming all the chosen thresholds in every scenario. For the dimmer target, for a false alarm probability greater than about 0.0001, all but the $R = 1$ case perform as well as or better

than the test thresholds. The jogs in the ROC curves occur because the filtering thresholds depend on the significance level. At very low significance levels and SNRs, the thresholds stop changing as a function of α_Z because the end of the background CDF used to determine P_{FA} (the end of the histogram in Figure 37 is reached). This isn't of great concern because such a low false alarm level does not produce a high enough probability of detection to be reasonably useful for low SNR targets, thus this is a regime in which the algorithm will rarely, if ever, operate.

The final filtering parameter to optimize is the voxel dimension. In Section 4.4.1.2 it was shown that using a large voxel meant that the optimal window size also was large. Since increasing the window size tends to wash out the target, a small voxel size will likely work best, allowing the streak length within a voxel to be maximized for a larger range of target SNRs and dwell times than would be the case for a large voxel. To help determine how small this voxel should be, the ASSET simulation approach of Section 4.3.3 was used. Figure 51 shows the probability of detection at $P_{FA} = 0.01$ for $n = 3$ at four different dwell times for a target with an SNR of 2.5. While this figure only shows a few of

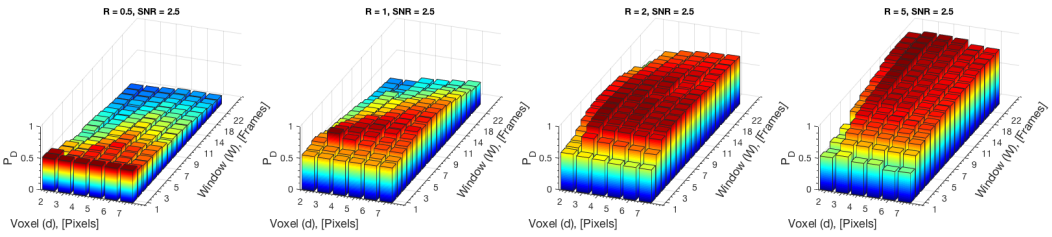


Figure 51. P_D **at** $P_{FA} = 0.01$ **as a function of voxel dimension** d **and moment window size** W **for** $n = 3$ **and a target with an SNR of 2.5 based on the ASSET simulation approach.**

the data points, in nearly every case, the $d = 3$ voxel tied or exceeded the probability of detection of the other voxels across the window size, dwell time, SNR parameter space. This is because it is a small enough region that the streak can be maximized within it even for fairly dim, fast targets. The $d = 2$ voxel likely did not work as well because in

such a small volume (8 total pixels), the distinction between noise and target is more difficult to make as it is far more likely for there to be a target equivalent number of false detects due to noise.

V. Results

Now that the parameters for the MBD algorithm are known for various target characteristics, it is possible to compare MBD to other methods. To summarize, the guidelines for optimizing MBD are:

1. The super window (W_S) should be chosen to be as large as possible while all frames included are expected to remain consistent (elements such bias drift, sensor motion and clouds will reduce the number of frames)

2. The optimal moment window (W) depends on the expected SNR and dwell of the target and can be found using Table 6, repeated here, where

$$W_{opt} = [R(d\sqrt{2}-1)+1] \tag{38}$$

Table 8. Window size for different target dwell times and SNRs. W_{opt} indicates the value can be calculated using Equation 38

SNR \R	≤ 0.5	(0.5,1)	$[1-2)$	$[2-3]$	>3
<3	1	1	3	5	W_{opt}
$3-6$	1	3	W_{opt}	W_{opt}	W_{opt}
>6	W_{opt}	W_{opt}	W_{opt}	W_{opt}	W_{opt}

3. For fast targets with dwell times less than about 2, skewness ($n=3$) should be used by itself. For slower targets, a combination of the mean ($n=1$) and skewness should be used.

4. The significance level α_Z controls the number of false alarms allowed prior to filtering and should be chosen based on an acceptable maximum false alarm probability

5. The best dimension of a voxel for filtering is $d = 3$[1]. The filtering threshold is determined automatically based on the user input SNR and dwell time, so these should be chosen on the low end of the desired range of targets to prevent filtering out true detections.

To put the results in perspective with respect to existing track-before-detect methods, the idealized scenario presented by Davey in [12] will be replicated so that the results of MBD using the optimized parameters can be directly compared to the reported results for four TBD algorithms. However, the idealized scenario is not truly representative of real data. To put the performance of MBD on realistic data in the context of other DBT methods, a direct comparison to the two other detect-before-track approaches described in Section 2.3 is made. This will be done both in terms of computation time and detection performance for a range of target characteristics and a number of different scenes to make the results as relevant as possible to the real problem of detecting targets in a variety of scenarios automatically and without analyst intervention.

5.1 Idealized Comparison

To see how the performance of the MBD algorithm compares to current approaches to detection and tracking, it is necessary to look at the detection performance of leading track-before-detect approaches. To compare to the results of [13], the MBD algorithm was run using the same approach as described in the book, generating similar scenarios using ASSET. Davey ran twelve scenarios consisting of four target speeds repeated at three different SNRs. The metric Davey used to define SNR is the peak maximum global SNR, SNR_{MX} of Section 2.4, where the noise in all 12 scenarios has a Gaussian distribution with standard deviation σ_B. This metric was also used to ensure a true

[1]For unresolved, point-like targets blurred across a 3 x 3 region with peak energy on detector (EOD) of about 0.25 - 0.8)

comparison, but the local temporal median SNR was also calculated to put the results in the context of the work in the previous sections. While Davey's example was for a radar system, the returns of such a system when grouped into spatial cells are equivalent to spatial pixels. That is, once the returns or imagery from each system (radar or staring optical) have been converted to digital numbers, they can be treated the same. While this is not always the case, this scenario ignores elements such as artifacts due to sensor motion that might cause data from these two sources to appear different to a detection or tracking method.

Figure 52 shows an example of a single frame of the data used in this comparison for each target speed. Each target is present for 20 frames of data so the faster targets travel

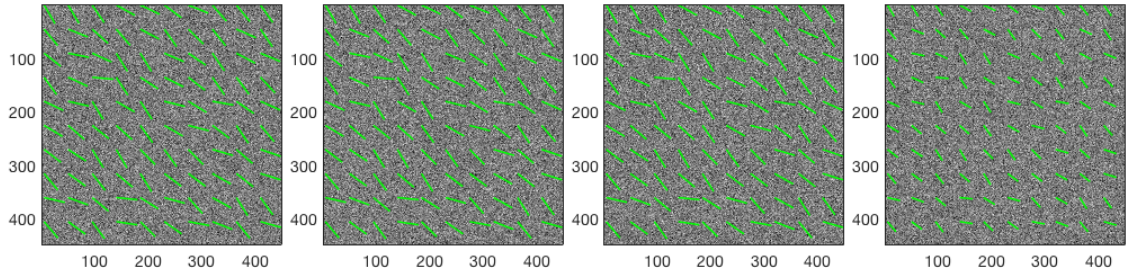


Figure 52. The 100 target paths over a single frame of data for each of the 4 speeds ($R = 0.5, 1, 2$ **and** 4) **from left to right.**

a greater distance than the slower targets. While the scenario presented by Davey only included 20 frames, because MBD relies on the temporal history of a pixel, each data cube was created with an extra 80 frames of target free data. To make the comparison, MBD was run on these 12 data sets using the parameters found in 4.4. Since the H-PMHT, particle filter, Bayes and Viterbi algorithms are all tracking methods, they have an inherent filtering step that is not present in MBD, namely using the motion of targets as a filtering mechanism. This makes the comparable operating point for MBD difficult to determine. To account for this, results at six different operating points are shown along with the TBD results in Figure 53.

The false alarm percentage shown is the percent of false alarms $FA = 100 \times P_{FA}$. To

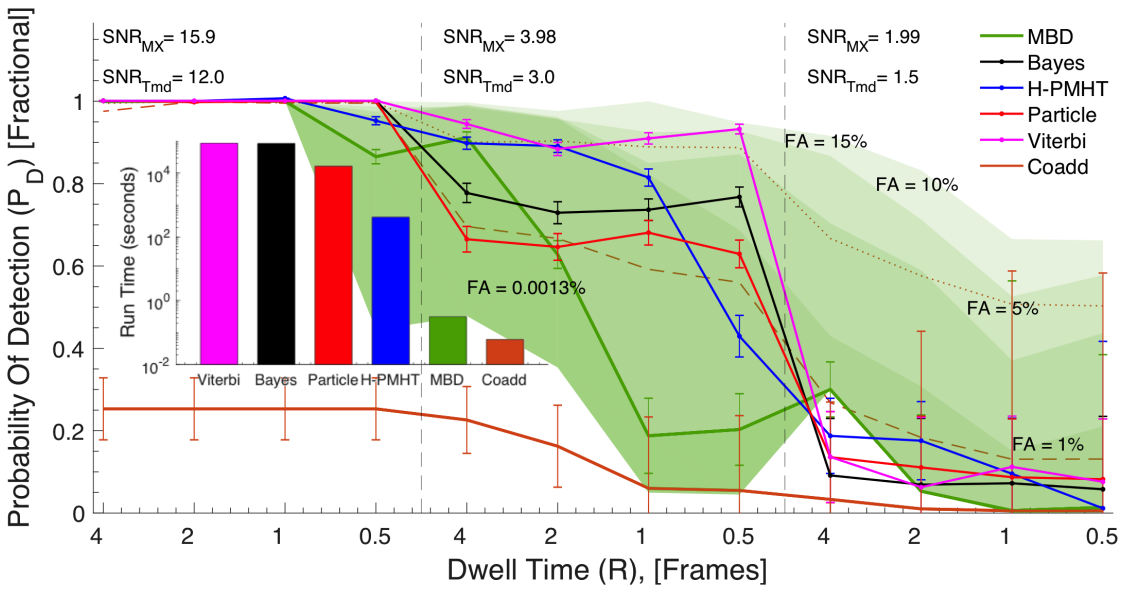


Figure 53. Comparison of detection performance and run time for 4 TBD methods and MBD and coadd at the 0.0125 percent false alarm operating point. The four different shadings above the MBD line represent four different higher false alarm levels while the shading below the line represents the lower 0.00125 percent false alarm operating point. The dashed line is the 1% level for coadd and the dotted line is the 15% level. Error bars are computed using Equation 7 as described in Section 2.4. The run time is for all 100 scenarios, or $20^3 \times 100$ pixels

determine a number of false alarms that is comparable to Davey, $\hat{P}_{FA}$ can be multiplied by the number of pixels in the 2000 frame noise scenario, 800,000. While the very low false alarm operating points do not always produce a probability of detection that is comparable with the TBD methods, allowing even 1% false alarms results in a comparable probability of detection, outperforming the particle filter and H-PMHT in nearly every scenario. While this level of performance in itself is a promising result, it is the difference in algorithm run times that presents the most notable difference. The histogram in Figure 53 shows the run-times for all 100 scenarios on a logarithmic scale. The H-PMHT algorithm is by far the fastest of the TBD algorithms (about 40 times faster than the particle filter), but MBD is over 1000 times faster. Finally, while coaddition is the fastest algorithm, it generally requires a much higher acceptable false alarm level to achieve the same results as the MBD algorithm. Even for very bright targets, to achieve the same probability of detection as MBD, the false alarm threshold for coadd must be

increased by a factor of four.

Due to the large large focal plane arrays and increasing time histories of today's data sets, computation speed is a critical element to consider when discussing the performance of an algorithm. Figure 54 shows algorithm run time as a function of number of pixels for two of the detect-before-track comparison methods, the MBD algorithm, and the two DBT approaches, coadd and PCA. These values were found

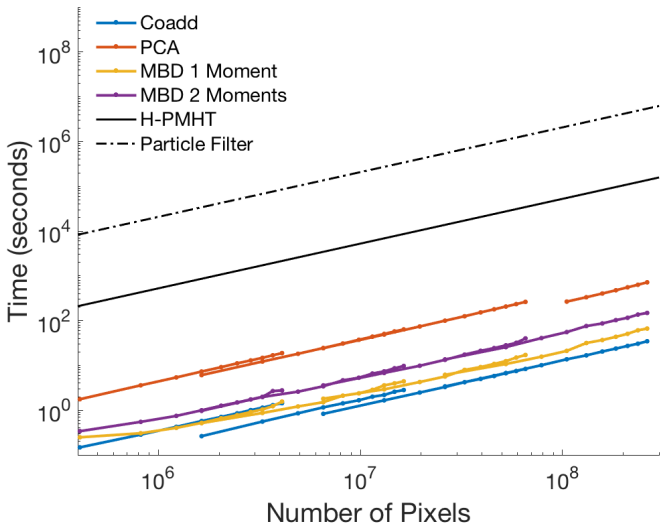


Figure 54. Algorithm run-time in seconds computed directly for coaddition, running PCA and MBD along with a conservative estimate (linear scaling) for the H-PMHT algorithm discussed in section 2.2.2

by running the algorithms for four different focal plane array sizes (64×64, 128×128, 256×256, and 512×512) varying the number of frames from 100 to 1000. The black lines, solid and dashed, represent very conservative estimates that assume linear scaling of the computation time for the track-before-detect H-PMHT algorithm and particle filter discussed in Section 2.2.2 based on the values reported by Davey [12]. Since the methods considered are fairly linear with respect to the number of pixels, and we assumed linearity for the H-PMHT and particle filter to give the most optimistic (albeit very generous) estimates, we can compute a time/pixel metric for each algorithm. These values are shown in Table 9. Additionally, the last two columns of the table show

the frame rates at which the algorithms could achieve real time performance for a 512^2 and a 256^2 pixel focal plane array respectively. The three different MBD times represent

Table 9. Algorithm Times

Algorithm	**Time per pixel (μs)**	512 RT FR(Hz)	256 RT FR (Hz)
MBD Only	1.43	2.67	10.67
Coadd	1.3	2.93	11.74
Running PCA	37.8	0.10	0.40
MBD1 + BG	2.4	1.59	6.36
MBD2 + BG	5.48	0.70	2.78
H-PMHT	525	0.0073	0.029
Particle Filter	20,750	1.84×10^{-4}	7.35×10^{-4}

MBD without any background suppression, MBD with one moment and background suppression, and MBD with two moments and background suppression. While the MBD algorithm is slightly slower than coaddition, it clearly has the potential to operate in the near-real time realm. It is also important to note that while we assumed for calculation purposes that everything scaled linearly, elements of the DBT algorithms including MBD will actually scale much slower with array size due to the potential for parallelization. For MBD and coadd, all operations are on either single pixels (in the case of the moment computation), or on small regions (in the case of the voxel filtering approach). For the background suppression, since the MBD algorithm does not compare spatial pixels across a large area, the data can be segmented and the principle components computed in small spatial regions in parallel as well. As a result of this parallelization, with enough computational resources, the run time and associated real time frame rate for a larger focal plane array will be nearly the same as for a much smaller focal plane array, limited only by the number of processors available. For example, if there are 16 processors available, each with enough memory to hold the data when it is divided into 128×128 data cubes, the new achievable real time frame rate for a single moment with background suppression would be about 25Hz. Whether or not this can

actually be achieved will depend on the overhead involved, but such overhead is likely to be small enough that run time will scale much slower than linearly for the paralellized version of the MBD algorithm. Finally, it should also be noted that the MBD, PCA and coadd algorithms are in the form of research Matlab code that has not been optimized for speed, while "significant effort" was spent in optimizing the TBD methods for speed in the work done by Davey et. al in [12].

5.2 Realistic Comparison

Now that the MBD algorithm performance has been put in the context of existing algorithms in terms of detection performance and run-time, the next step is to move from an idealized scenario with simple target paths, no clutter, and normally distributed noise, to one with realistic scene elements. For this comparison, the two DBT algorithms described in Section 2.3 were run and compared to MBD for a large number of realistic scenes generated using ASSET.

5.2.1 Optimization of DBT Comparison Algorithms.

Because the MBD algorithm parameters have been optimized based on target and scene characteristics, to make a fair comparison between algorithms, the coadd and running principle components algorithm parameters also need to be optimized. While some elements of this optimization are relatively straight forward, such as matching the number of frames in the signal block to the target dwell time, others are less clear. While it is beyond the scope of this research to dive deeply into the optimization of these parameters, an understanding of the optimal choices can be gained using a brute force approach and a large quantity of simulated data. For both algorithms, the key parameters are the background block size W_S, the buffer gap size G and the signal block size W. To keep computation times reasonable, a small selection spanning the most

likely best choices for these parameters were chosen and implemented. These values are shown in Table 10. The algorithms were run for every possible combination of these

Table 10. Coadd and running principle components algorithm parameters

Parameter	Values
Background Block (W_S)	3 5 7 10 15 20 30 40 50
Gap Block (G)	1 3 5 7
Signal Block (W)	1 2 3 4 5 6

19 different values for a total of 216 runs. In each case, the threshold for detection was varied to obtain ROC curves. In order to determine which set of parameters produced the best ROC curve, the minimum distance to the top left corner (as discussed in Section 2.4) was used.

Twenty four variations of the same realistic scene were created in ASSET, varying the target SNR and speed in each scene in order to obtain the relationship between the optimal parameters, target speed and SNR. Figure 55 shows a single frame of data along with the target paths for the four different dwell times. Also shown is the offset from the first frame for one of the sets of data. This second plot shows the amount of drift and jitter present in the scene and is approximately the same for all 24 scenes.

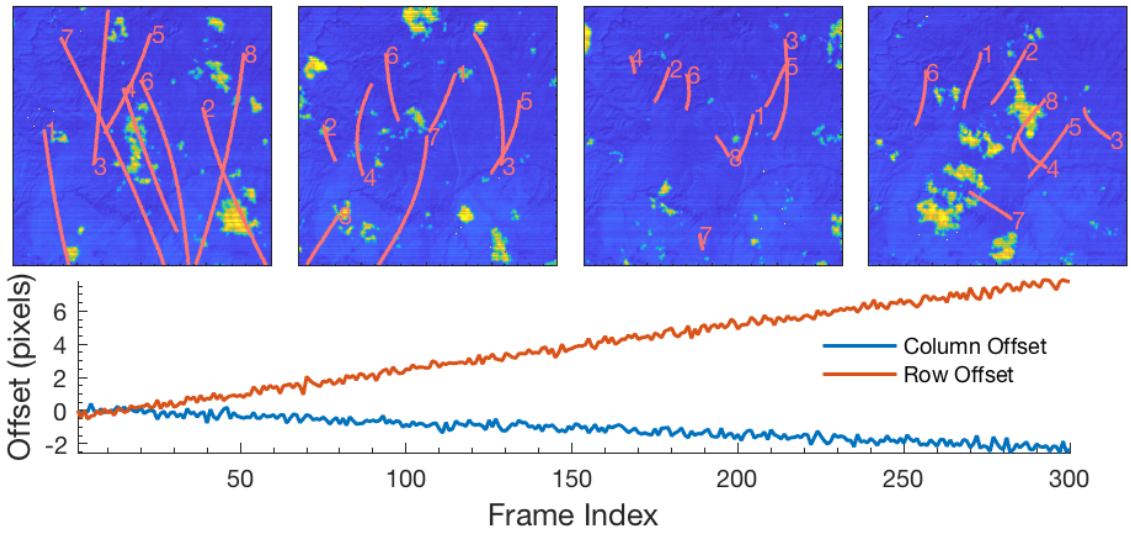


Figure 55. Target paths for dwell times $R = 1, 2, 3$ and 5 from left to right (top) and pixel offset from the first frame of data (bottom).

Tables 12 and 11 show the parameter sets that resulted in the smallest distance to the upper left hand corner of the ROC curves for the PCA and coadd algorithms respectively. For both algorithms, but especially for the PCA algorithm, the SNR of the target had little to no impact on the optimal set of parameters. While there appears to be some variation in the background block size with SNR for the coadd algorithm, $\hat{P}_D$ for values of five, seven and ten are within the error of each other across the four different dwell times.

Table 11. Optimal parameters for coadd

	Background Block				Gap Block				Signal Block			
SNR \ R	1	2	3	5	1	2	3	5	1	2	3	5
2.5	7	10	7	5	1	1	3	5	1	1	2	3
3	7	10	7	7	1	1	3	5	1	1	2	3
3.5	10	10	10	7	1	1	3	5	1	1	2	3
4	10	10	10	7	1	1	3	5	1	1	2	3
4.5	7	10	10	7	1	1	3	5	1	1	2	3
5	10	10	10	5	1	1	3	5	1	1	2	3

Table 12. Optimal parameters for PCA

	Background Block				Gap Block				Signal Block			
SNR \ R	1	2	3	5	1	2	3	5	1	2	3	5
2.5	15	15	15	15	1	1	3	5	1	1	2	3
3	15	15	15	15	1	3	3	7	1	1	1	3
3.5	15	15	15	15	1	1	3	7	1	1	1	3
4	15	15	15	15	1	1	3	7	1	1	1	3
4.5	15	15	15	15	1	1	3	7	1	1	1	3
5	15	15	15	15	1	1	3	7	1	1	1	3

There is a clear and consistent dependence in the gap and signal block sizes on the dwell time for both algorithms, making it necessary to choose different algorithm parameters based on the expected dwell times of targets within the scenes.

5.2.2 Comparison Scenes and Results.

Now that the parameters for the two comparison algorithms have been determined, a fair comparison can be made to the MBD algorithm. To ensure that this comparison is realistic, seven different source images were used with ASSET and a variety of targets included. Because it is reasonable to assume some knowledge of a target's speed, frame stacks were generated with narrow ranges of target speeds but a broad range of SNRs. For each of the seven source images, six data sets with dimensions $400 \times 400 \times 400$ were generated, two for each grouping of target speeds, for a total of 42 scenes. Figure 56 shows histograms for the dwell time and target SNRs by these groupings.

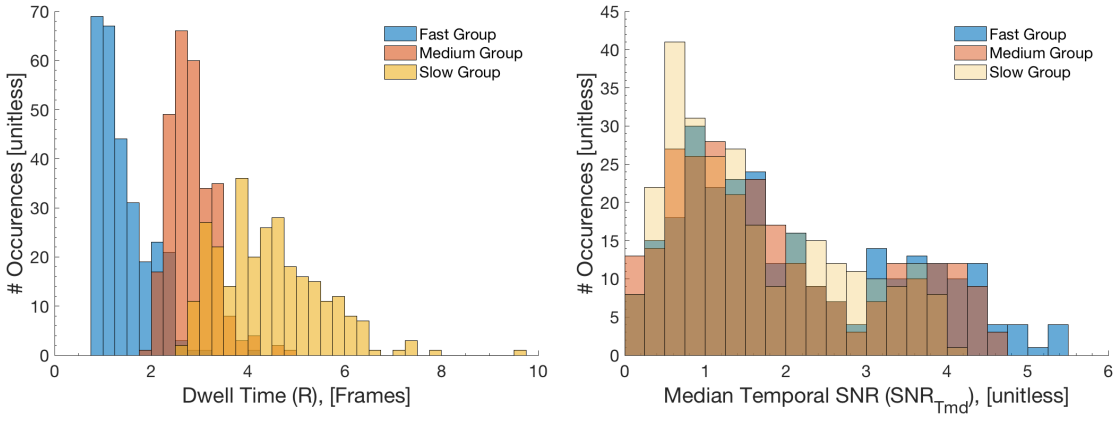


Figure 56. Distribution of target dwell times (left) and SNRs (right) for the three speed groupings of targets in the realistic set of scenes

The fast group has targets from $R = 0.75$ to $R = 2.25$, the medium group overlaps slightly, with most of the targets ranging from $R = 1.75$ to about $R = 3.5$, and the slow group ranges from $R = 3$ to $R = 6$. The SNRs for all three speed groups range from zero to about 5.5. Figure 57 shows examples of these target paths over a single frame of data from a frame stack using each of the seven different source images. The labels indicate the general geographic region for the Landsat 8 image used to generate the data. These locations were chosen with the intent of obtaining a wide variety of scene content to test the MBD algorithm across the scene parameter space. For all 42 scenes, the jitter

and drift profiles are on the order of those used to optimize the algorithm parameters in Section 5.2.1. This was done to ensure these parameters remained close to the optimal choices to keep the comparison fair.

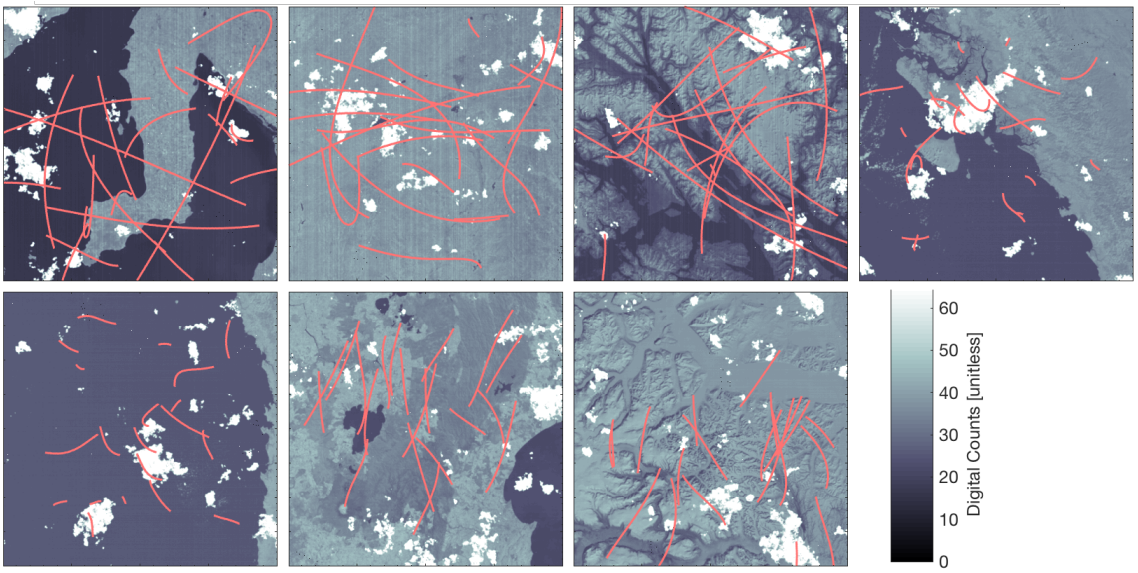


Figure 57. Examples of randomly generated target paths over single frames of data from the seven different scenes used in the final comparison. From left to right and top to bottom, the fist 3 are fast examples, the next two are medium speed examples, and the final two are slow examples.

Table 13 shows the parameters chosen for the two DBT algorithms for each of the three speed groups based on the results of Section 5.2.1. The MBD parameters were chosen based on the results in Section 4.4 using dwell times of 4, 3, and 1 for the slow, medium and fast groups respectively, and an input SNR of 2. This value for SNR seems high (shifted from the minimum) based on Figure 56 because these scenes contains jitter and drift, so the computed median temporal SNR actually includes clutter, therefore the background suppression step will help raise the SNR before the moments are computed. For the slower two groups, both the mean and skewness were used, while for the fastest group only skewness was included. To compare the detection performances of the algorithms, the detection thresholds were varied to obtain ROC curves for each individual target in all 42 scenes for a total of 840 separate ROC curves. To visualize these results, two false alarm levels, $\hat{P}_{FA} = 0.005$ and $\hat{P}_{FA} = 0.01$ were chosen

Table 13. DBT Algorithm parameters for the three target speed groups

	Background Block			Gap Block			Signal Block		
	Fast	Med	Slow	Fast	Med	Slow	Fast	Med	Slow
Coadd	10	10	7	1	3	5	1	2	3
PCA	15	15	15	1	3	7	1	1	3

and the corresponding $\hat{P}_D$ for each target determined. These values were plotted as a function of the target's average dwell time and SNR and a surface was fit to the results. Figure 58 shows these two plots.

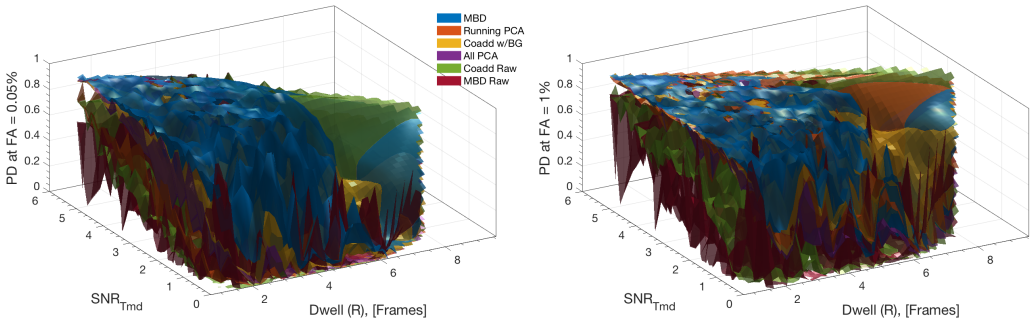


Figure 58. $\hat{P}_D$ at two different false alarm levels as a function of dwell time and local temporal median SNR prior to background suppression. The different colored surfaces represent the results for different versions of the 3 algorithms. All PCA refers to computing PCA over the whole data cube and then thresholding as in the running PCA algorithm. Raw indicates no background suppresion was done prior to running the algorithm.

Clearly, for all but the very slow targets the MBD algorithm is outperforming all other approaches in nearly every case. For the 0.05% false alarm level, MBD outperforms all of the other methods for 82% of the targets while for the 1% false alarm level, MBD produces the highest $\hat{P}_D$ for 77% of the targets. Additionally, at a false alarm level of 1%, nearly all of the targets are detected with a probability of detection very close to one. Figure 59 quantifies this by showing first what fraction of the 840 targets represented by the surface in Figure 58 were detected better using MBD (with the blue bars) than each of the other methods, and second, by indicating on average how much better MBD detected these targets than each of the other methods (black).

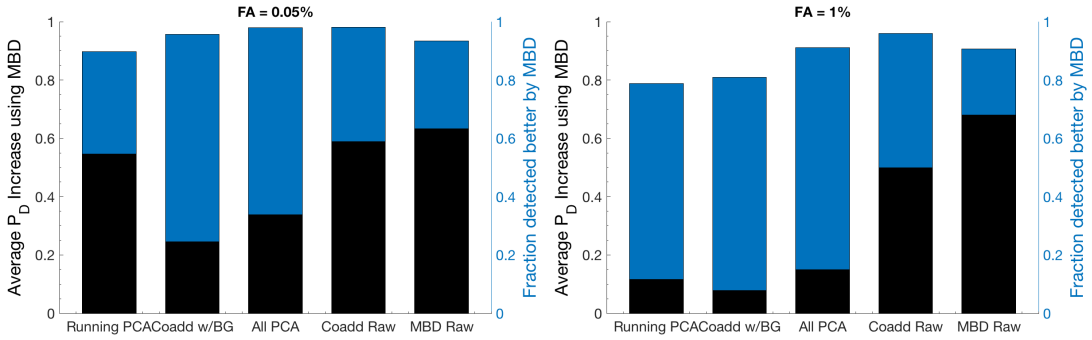


Figure 59. Fraction of the 840 targets in the 42 realistic scenarios used to make the surface in Figure 58 that are detected better using MBD (blue) for each detection method, and by how much in terms of $\hat{P}_D$ (black) at each false alarm level.

For example, for the lower false alarm percentage of 0.05%, of the 840 targets in the 42 scenarios run, MBD detected 754 (90%) of the targets better than running principle components with an average probability of detection 0.55 higher than running PCA. As the false alarm level is increased to 1%, MBD still detects 79 % of the targets better than running principle components, but the average improvement in probability of detection over running PCA drops to 0.12. The majority of the targets that are detected by a method other than MBD are those with long dwell times. This is due to the blocked background suppression approach used and could be mitigated by adding frames together to increase the target's speed and signal prior to running MBD. These results indicate that MBD is still effective when complicating elements such as fixed pattern noise, bad pixels, jitter, drift and clouds are present, showing clear improvement over other DBT methods, especially at low false alarm levels.

Finally, by summing all frames, or a large subset of frames in the final binary detections cubes that are output by these DBT detection methods, an additional advantage of MBD is evident. Figure 60 shows an example of a super frame made from the fast realistic scenario with the New Zealand background for MBD (left) and for running PCA (right).

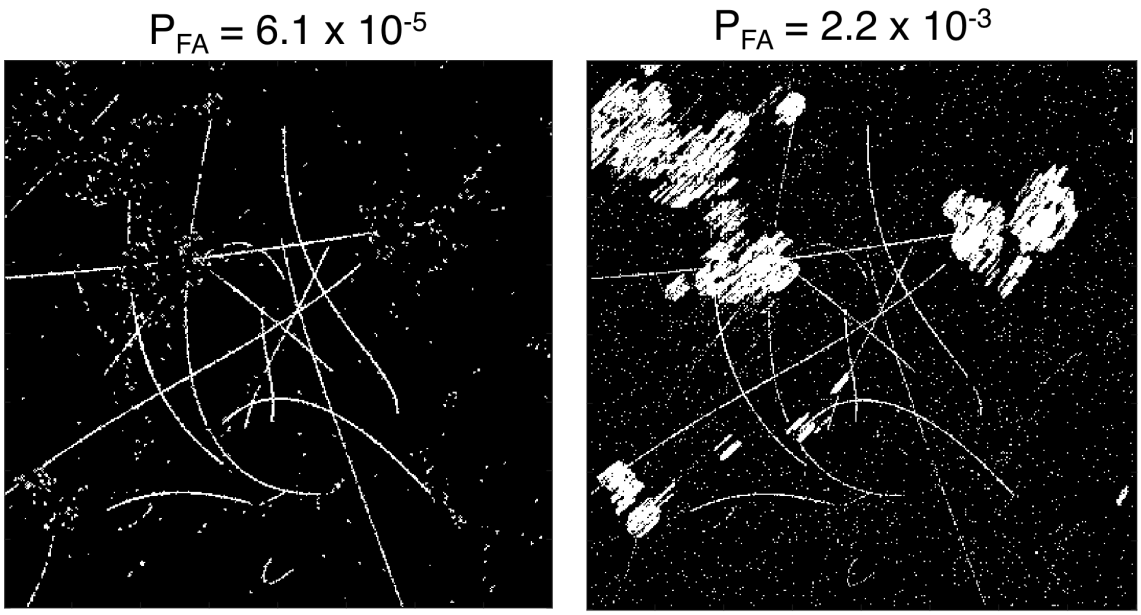


Figure 60. Super frames for the fast New Zealand scene (detects summed over all frames) for MBD (left) and running PCA (right).

Because MBD produces streaks, the target trajectories are more solid and easier to pick out at a much lower false alarm level than in those in a single point detection approach like running PCA. In the example, the target trajectories in the super frame computed from MBD detects are still more solid than those from the running principle components method, even though running PCA has over 30 times as many false alarms. At this point this super frame approach is useful as a tool for an analyst to quickly identify if there is a target like trajectory in the data, however this step could be automated with an algorithm that looks for such trajectories.

5.3 Implications for Sensor Design

At this point the relationships between the MBD algorithm parameters, target characteristics, and detection performance have been thoroughly examined. With this information, it is possible to understand generally the characteristics of a sensor that will work well with MBD based on parameters such as the target's speed, intensity and

area, and the sensor's frame rate and ground sample distance (GSD). As expected, the SNR and dwell time of a target have a significant impact on detection performance when using MBD. It is intuitive that maximizing SNR will maximize detection performance, but because of the way that MBD approaches detection, there is a more nuanced relationship between dwell time and ideal performance of MBD. Very fast dim targets ($R < 1$) will appear as dashed streaks, making filtering difficult, while very slow targets can be integrated into the background during background suppression, making them difficult to detect. Looking at Figure 58 this upper bound on dwell time is at about $R = 5$, suggesting that ideally targets will have dwell times $1 < R < 5$, with the middle of this range ($2 < R < 4$) providing the best performance as this is where streak length is maximized for dimmer signals and sensitivity to window size becomes less significant (as shown in Figure 43). Of these two parameters, SNR will tend to drive detection performance more than dwell time, but only as long as the dwell time is within MBDs optimal range. These two parameters describe the end appearance of a target to a sensor, but they are consequences of a sensor's design

The intensity of a target, its area, the sensor integration time (related to frame rate), and the sensor GSD all impact the final SNR of a target. As discussed in Chapter I, MBD was developed to detect dim, unresolved moving targets and because MBD operates on single spatial pixels individually, extended targets do not significantly increase the algorithm performance. Thus, an ideal sensor in terms of target SNR is one that maximizes the signal of a target within a single pixel, matching the area of the target to the GSD of the sensor. Improving spatial resolution beyond this point will only serve to reduce the intensity of the target within the pixel and lower its detectability. Integration time also has a significant impact on a target's apparent SNR, but an ideal choice will depend heavily on the target's speed, as integrating for a longer is not beneficial if the target has moved out of the pixel.

The dwell time of a target depends on three main factors, the GSD and integration time of the sensor, and the target's velocity. The latter cannot be controlled, but the two sensor parameters could be tailored to specific classes of target speeds. For example, consider a target with an area approximately equal to the GSD of a sensor moving at a speed that covers a quarter of a pixel every second. MBD will operate best if the integration time is chosen to be about 2 seconds, putting the target's dwell time at $R = 2$. This choice strikes a balance between increasing SNR through a longer integration time and obtaining a target dwell time that works well for the MBD algorithm. This is a specific example, but generally choices that increase the target's SNR while keeping the dwell time in the middle of the range of ideal dwell times will result in better detection performance.

Finally, all of these choices assume that sensor alignment has been done well. The most problematic element for MBD is the case where the sensor has moved enough that a pixel is imaging a significantly different portion of the scene, making the estimate of the background, $\hat{\mu}_B$ used in the moment calculation inaccurate. For MBD to work well, pixels must image the same portion of scene for a reasonably large number of frames. In practice this was found to be about 50 frames, but as indicated by figure 39, if the noise in the scene is high, this number will increase.

VI. Conclusion

6.1 Summary of Accomplishments

The goal of this research effort was to contribute a solution to the problem of detecting small, unresolved, low SNR targets automatically in near real-time, reducing the burden on analysts and making better use of the ever increasing quantity of data collected by today's sensors. The key accomplishments of this effort are:

- Demonstrated that ASSET simulation data was spatially and temporally representative of real imagery collected in the electro-optical spectrum (0.4-15μm), making it a viable source of data for detection algorithm development, testing, and evaluation (Chapter III)

- Derived expected MBD performance using a physical sensor model and validated the expressions found with both numerical simulations and a separate moment-generating function derivation (Section 4.3.1, Appendix A)

- Determined optimal MBD parameters for detection performance as a function of the target's physical characteristics: radiometric intensity (via SNR) and velocity (Section 4.4, summarized in Chapter V)

- Demonstrated that MBD was capable of detecting targets with comparable detection performance — at $1/1000^{\text{th}}$ the computational cost — of the fastest TBD approaches, particle filter and H-PMHT (Section 5.1)

- Established that MBD can reliably detect targets with an $SCNR_T$ down to 0.5 under realistic imaging conditions (drift / yaw, jitter, cloud motion) with limited a priori target information, and typically outperformed optimally-tuned PCA and coaddition algorithms (Section 5.2)

- Developed a method of using the filtered MBD results to enable analysts to quickly determine the presence or absence of a moving target in a frame stack

6.2 Conclusion

Using a theoretical approach, a simple target and background moment window simulation, and a brute force approach using a large quantity of simulated data, it is clear that intelligent choices can be made for the algorithm parameters in MBD. It was found that when these rules were used to implement the MBD algorithm for an idealized scenario with targets moving linearly embedded in normally distributed noise and a 1% false alarm level, MBD outperformed both the histogram probabilistic multiple hypothesis tracker and the particle filter for targets with dwell times ranging from R = 0.5 to R = 4 and local temporal SNRs down to $SNR_{Tmd} = 1.5$ with computation times 1337 and 52,854 times faster respectively. This puts MBD firmly in the category of a near real-time approach. However, real data is much more complicated with non-Gaussian artifacts introduced by platform motion combined with clouds and differences in pixel responses.

To test MBD under more relevant conditions, a large quantity of realistic data containing targets with a range of speeds and signal levels as well as a variety of backgrounds was of critical importance. To fill this need, ASSET was developed, tested, and implemented to generate these data sets quickly and easily. Data generated with ASSET was shown to be truly representative of real data through comparisons of lab camera data and weather satellite data with their simulated versions. While the simulated data is not necessarily radiometrically accurate, it does capture the relevant sensor motion and resulting artifacts that impact detection performance, such as cloud edges and coast lines that create clutter noise. Through the use of ASSET data containing a variety of artifacts that frequently pose problems for detection, MBD was tested under realistic

conditions and shown to reliably detect ($P_D > 0.9$) targets with a temporal signal to clutter plus noise ratio, $SCNR_T$, down to about 0.5 for dwell times from about 0.75 to 5. The MBD algorithm has been shown to have great potential for automatically and reliably detecting dim targets in near real-time, without the intervention of an analyst. With further proliferation of high resolution, high frame rate sensors into a large number of department of defense and intelligence community applications, the demand for automatic detection and tracking methods will only grow. The thorough analysis of the potential for moments to contribute to this problem space is just one step in the journey to achieving a fast and reliable automatic detection and tracking process for dim targets.

6.3 Future Work

The next steps in continuing the development of the MBD algorithm include three areas: code efficiency and speed improvements, further development of the filtering, and exploration of how best to use the final streaks produced by the MBD algorithm. The version of the code in Appendix B is research code and while the moments were computed using a convolution approach to reduce run-time, most of the code is not optimized for speed and it does not include parallelization using a GPU. With the combination of parallelization and a focus on improving code speed, the algorithm run-time could be greatly reduced.

The automatic filtering thresholds developed in Section 4.4.3 are only lower thresholds and were found and tested under ideal conditions, with no jitter or clouds. The MBD algorithm does have the capability of putting an upper cut off on voxels as well, but to do this, a clear understanding of the maximum expected number of detects within a voxel due to a target is needed. This will require factoring in the PSF of the imager and the SNR of the target in order to avoid removing voxels that simply have very bright

or large targets with energy spread across multiple pixels. Additionally the voxel size may also need to be adjusted in order to include a large enough region to differentiate between such targets and regions that contain many detects due to clutter. Thus, a large scale study using realistic data across a wide range of target, sensor and scene characteristics is needed to further tune the filtering thresholds in order to remove more clutter artifacts while preserving the targets. Possible avenues of exploration include using two filtering steps, one with a lower threshold and smaller voxel and one with an upper threshold and a larger voxel, and studying the clutter artifacts common in real data to determine features specific to clutter noise that can be used to differentiate them from real streaks. Additionally, super frames made from MBD detects make determining the general location of a target trajectory over time quick for an analyst, but automating the process of determining the potential presence or absence of a target based on these super frames is a logical next step.

Finally, at this point in the research process, the output of MBD is a filtered detections cube containing streaks where targets are likely to appear. The next step in the development of this algorithm is to implement a form of tracking to combine the individual streaks into tracks. Because the streaks contain many detections per target, the assumption that each detect is associated with a single target made by most trackers is violated, requiring the development of an alternative approach. Additionally, as a part of this tracking process, it will be necessary to determine where within the streak the target actually is in any given frame. This has been determined to be a function of a frame's position in the frame stack, with the target appearing at the head of the streak for the first frames, and the tail of the streak for the last frames, but a method to determine this for the filtered detections is needed in order to determine the true position of a detected target in any given frame. For cases where the target is moving linearly with constant velocity, this problem is relatively straightforward, but for targets

with both axial and lateral acceleration, the problem becomes more complicated.

These are the three main areas for the future development of MBD, however there are two smaller items that could help improve the performance of MBD. First, it was shown that because of the block background suppression method, MBD has an upper limit on the dwell time of a target for which it will work well. One potential improvement is to combine frames together to enhance signal and speed up the target in situations where the target is expected to be slow. Ensuring that this process does not alter the statistics in a way that will reduce detection performance and implementing this change will remove the upper limit on the dwell time of a target for MBD. Second, since large sensor motion is one of the most significant challenges for MBD, implementation of fast and accurate frame registration prior to using MBD will help remove many of the clutter artifacts that currently cause most false detects.

Appendices

IMPROVING DETECTION OF DIM TARGETS:

OPTIMIZATION OF A MOMENT-BASED DETECTION ALGORITHM

A. Expected Moment Equations using Moment Generating Functions

The moment generating function $m(t)$ for a random variable X is defined as

$$m(t) = E[e^{tX}]. \tag{39}$$

Additionally, if $m(t)$ exists, the nth moment of the random variable X taken about the origin, $E[Y^n] = M^n$ can be found by taking the derivative of $m(t)$ with respect to t and evaluating at $t = 0$,

$$\left.\frac{d^n m(t)}{dt^n}\right|_{t=0} = m^n(0) = M^n. \tag{40}$$

As a result, if the probability distribution function for the random variable X is known, equations 39 and 40 can be used to find its moments. In the case of pixels belonging to a background window or a target window, these PDFs are described by a single Gaussian distribution and a mixture of Gaussians respectively,

$$X_B \backsim \frac{1}{\sqrt{2\pi\sigma}} \exp \frac{-(x-\mu_B)^2}{2\sigma^2} \tag{41a}$$

$$X_T \backsim \frac{f}{\sqrt{2\pi\sigma}} \exp \frac{-(x-\mu_B)^2}{2\sigma^2} + \frac{1-f}{\sqrt{2\pi\sigma}} \exp \frac{-(x-\mu_T)^2}{2\sigma^2}. \tag{41b}$$

where f represents the fraction of background pixels in the smaller temporal window (size W). In terms of dwell time, (the time a target spends in a single pixel, R) $f = \frac{W-R}{W}$. Using equation 39, the two moment generating functions for background and target

windows are given by

$$m(t)_{X_B} = \int_{-\infty}^{\infty} e^{tx} \mathcal{N}(\mu_B, \sigma_B^2) dx \tag{42a}$$

$$m(t)_{X_T} = \int_{-\infty}^{\infty} e^{tx} \left[\frac{W-R}{W} \mathcal{N}(\mu_B, \sigma_B^2) + \frac{R}{W} \mathcal{N}(\mu_T, \sigma_B^2) \right] dx. \tag{42b}$$

For the simple background case, this integral is easily solved and produces the well known moment generating function for a normal distribution. For the target case, the integral can be separated into two integrals, one for each normal distribution. Additionally, equations 39 and 42 are for the moments taken about the origin, but for the MBD algorithm, central moments are taken about an estimate of the background mean $\hat{\mu}_B$. To account for this difference, we center the normal distributions by setting $\mu_B = 0$. Noting that $\mu_T = \mu_B + \beta\sigma_B$ and evaluating the integrals, the final moment generating functions for the target and background window cases are

$$m_{X_B}(t) = \exp(\frac{1}{2}\sigma_B^2 t^2) \tag{43a}$$

$$m_{X_T}(t) = \frac{W-R}{W} \exp(\frac{1}{2} t^2 \sigma_B^2)) + \frac{R}{W} \exp(\frac{1}{2}(2\beta\sigma_B t + \sigma_B^2 t^2). \tag{43b}$$

Finally, to get from equation 43 to equation 27 in section 4.3.1, apply equation 40 to equation 43 and subtract the background case from the target case,

$$\Delta\langle M^1 \rangle = \frac{R}{W} \beta\sigma_B \tag{44a}$$

$$\Delta\langle M^2 \rangle = \frac{R\sigma_B^2}{W}(\beta^2 + 1) - \sigma_B^2 \tag{44b}$$

$$\Delta\langle M^3 \rangle = \frac{R\sigma_B^3}{W}(3\beta + \beta^3) \tag{44c}$$

$$\Delta\langle M^4 \rangle = \frac{R\sigma_B^4}{W}(1 + 6\beta^2 + \beta^4) - 3\sigma_B^4 \tag{44d}$$

$$\Delta\langle M^5 \rangle = \frac{R\sigma_B^5}{W}(5\beta + 10\beta^3 + \beta^5) \tag{44e}$$

B. MBD Matlab Code

The MBD algorithm is called using RunMBD:

```
function [DetsF,opts,Moms,Dets,F] = RunMBD(Frames,opts,varargin)
    % [DetsF,opts,Moms,Dets,F] = RunMBD(Frames,opts,varargin)
    %
    % Main code to run the Moments-Based Detection algorithm
    %
    % --- Inputs ---
    % Frames - 3D array of staring frame data
    % opts   - options structure controlling how MBD works
    %
    % --- Outputs ---
    % DetsF - filtered detections
    % opts  - updated options structure
    % Moms  - moments array
    % Dets  - unfiltered detections
    % F     - processed frame data that was fed into moment comp
    %
    % ---------------------------------
    % Shannon R. Young
    % POC: Bryan J. Steward
    % POC: Kevin C. Gross
    % Air Force Institute of Technology
    % Wright-Patterson AFB, Ohio
    % Kevin.Gross@afit.edu
    % (937) 255-3636 x4558
    % Version - Dissertation
    % 10-0ct-2018
    % ---------------------------------

    opts = parse_pv_pairs(opts,varargin);
    [Moms,opts,F] = mbd2_moms(Frames,opts);
    [Dets] = mbd2_detect(Moms, opts);
    [~,DetsF] = BoxFilter2(Dets,opts,0);
end
```

These functions and their necessary sub-functions are printed here. First the options function is called to set the default options and change any user input options:

```
function opts = mbd2_opts(varargin)
% opts = mbd2_opts(varargin)
%
% Function MBD2_OPTS generates the default options structure used for
% detecting statistically signficant pixels in frame arrays using running
% windows of moments.
%
% OPTS = mbd2_opts() with no input arguments returns the default OPTS
% structure for use in MBD. OPTS = mbd_opts(FIELD,VALUE) generates the
% default OPTS structure and replaces the default value in FIELD with VALUE.
% Multiple FIELD VALUE pairs are allowed.
%
% --- Inputs ---
% field, value pairs - see code for full set of adjustable options
%
% --- Outputs ---
% opts - options structure
%
% ---------------------------------
% Shannon R. Young
% POC: Bryan J. Steward
% POC: Kevin C. Gross
% Air Force Institute of Technology
% Wright-Patterson AFB, Ohio
% Kevin.Gross@afit.edu
% (937) 255-3636 x4558
% Version - Dissertation
% 10-0ct-2018
% ---------------------------------

% Prepare Inputs
if nargin == 0 || ~isa(varargin{1},'struct')
    opts.gpuFlag = 0;           % Set to 1 to compute on GPU

    % Preprocessing options
    opts.bg_suppress = [];      % RxCxK chunk size for BG removal; no removal if empty

    % Postprocessing options
    opts.zscore = 1;            % zscore the moments after computation
    opts.std_norm = 0;          % divide moments by sigma^n after computation
    opts.power_norm = 0;        % raise moments to 1/order power
    opts.alpha = 0.01;          % threshold for quantiles
    opts.verbose = 1;           % display feedback at command line
    opts.SNRguess = 3;       % Guess at SNR of target, for window size calculation
    opts.moments = [1,3];       % moments to calculate
    opts.dwell = 1.25;             % Guess at # frames a target spends in a single pixel
    opts.window_size = [];      % will be calculated later
    opts.super_window = [];     % Number of frames to use for mean computation
    opts.step_size = 1;         % step between start frames --> DO NOT CHANGE
    opts.frame_size = [];       % NxMxP size or frame array
    opts.moment_size = [];      % NxMxQ size or moment array
    opts.precision = 'single';  % Not used everywhere yet...
    opts.data_mean = [];        % Field to store average temporal mean
    opts.data_std = [];         % Field to store average temporal std

    % box filtering parameters
    opts.box = 3;               % size of box for summation filtering
    opts.FinalFilt = 1;         % turns on 2D convolution with summed dets at the end
    opts.upper_thresh = 1;      % threshold for upper cutoff - 1 for no upper cutoff
    opts.lower_thresh = 0.05;   % threshold for lower cutoff (detn's noise detects in box)
end

% window size as a function of dwell and maximum moment order:
opts.window_size = getBestW(opts);

% update with user inputs:
opts = parse_pv_pairs(opts,varargin);

% re-calculate window size if necessary
if (any(strcmp(varargin(:),'dwell')) || any(strcmp(varargin(:),'SNRguess')))
    opts.window_size = getBestW(opts);
end
```

Now that the options have been set, the moments are computed:

```
function [Moms,opts,F] = mbd2_moms(Frames,opts,varargin)
    % [Moms,opts,F] = mbd2_moms(Frames,opts,varargin)
    % MBD2 function to do preprocessing, compute moments, and postprocessing
    %
    % --- Inputs ---
    %   Frames - Frame stack (ASSET or other data, N x M x K 3D cube format)
    %   opts   - opts structure generated with mbd2_opts
    %
    % --- Outputs ---
    %   Moms - 4D moment array with dims [Row, Col, Moment Value, Moment Order]
    %   opts - opts structure updated with mean and stdev of pre-processed frames
    %   F    - processed frames data that was fed into the moment computation
    %
    % ---------------------------------
    % Shannon R. Young
    % POC: Bryan J. Steward
    % POC: Kevin C. Gross
    % Air Force Institute of Technology
    % Wright-Patterson AFB, Ohio
    % Kevin.Gross@afit.edu
    % (937) 255-3636 x4558
    % Version - Dissertation
    % 10-0ct-2018
    % ---------------------------------

    % process options
    opts = parse_pv_pairs(opts,varargin);
    opts.frame_size = size(Frames);

    % preprocess as dictated by opts
    if ~isempty(opts.bg_suppress)
        % Apply background suppression
        F = BG_suppress(Frames, opts.bg_suppress, 0,[],[],opts);
    else
        if ~opts.zscore
            % Mean subtraction if no zscore later
            F = bsxfun(@minus, Frames, mean(Frames,3));
        else
            % Do nothing if it will be z-scored
            F = Frames;
        end
    end

    % update opts structure to include average per pixel mean and std
    opts.data_mean = mean(vec(mean(F,3)),'omitnan');
    opts.data_std = mean(vec(std(F,[],3)),'omitnan');

    % set super-window parameter if not / inappropriately supplied
    test1 = isempty(opts.super_window)
    test2 = opts.super_window>=(opts.frame_size(3)-opts.window_size)
    if test1 || test2
        % calculate super window so error is less than 0.1 std
        opts.super_window = ceil(4*opts.data_std/0.1);
        if opts.super_window >= opts.frame_size(3)
            opts.super_window = opts.frame_size(3)-opts.window_size;
        % set a minimum super window size of 5
        elseif opts.super_window < 5
            opts.super_window = 5;
        end
    end

    % Compute moments
    W = opts.window_size;
    BigW = opts.super_window;
    [Data,dims] = rs2D(F);
    nT  = dims(end);
    nK  = nT-W+1;
    ix  = repmat(0:W-1,nK,1)+cumsum(ones(nT-W+1,W));
    ix  = ix(1:opts.step_size:end,:);
    nK  = size(ix,1);

    % reshape to N*M by P by W (row times col, reduced moment frames, window)
    dat = reshape(Data(:,ix),prod(dims(1:2)),nK,W);

```

```
    % First copy BigW/2 frames to the front and back to extend window to center
    DataPlus = cat(2,cat(2,Data(:,W+1+ceil(BigW/2):W+1+BigW-1),Data),...
        Data(:,end-(W+BigW-1):end-(W+floor(BigW/2))));

    % compute mean using convolution with or without gpu
    kernel = [ones(floor(BigW/2),1); zeros(W,1); ones(ceil(BigW/2),1)];
    sm = conv2(DataPlus, kernel', 'valid')./BigW;

    % compute std over super window if needed
    if opts.std_norm
        if opts.gpuFlag
            sd2 = conv2(DataPlus.^2,kernel','valid')./BigW-sm.^2;
            % remove (set to zero) negative values and take the square root
            sd2(sd2<0) = 0; sd2 = sqrt(sd2);
            sd2 = gather(sd2(:,1:opts.step_size:end));
            sd2 = repmat(sd2(:,1:size(dat,2)),1,1,size(dat,3));
        else
            sd2 = conv2(DataPlus.^2,kernel','valid')./BigW-sm.^2;
            % remove (set to zero) negative values and take the square root
            sd2(sd2<0) = 0; sd2 = sqrt(sd2);
            sd2 = sd2(:,1:opts.step_size:end);
            sd2 = repmat(sd2(:,1:size(dat,2)),1,1,size(dat,3));
        end

    end
    sm = sm(:,1:opts.step_size:end);

    % repeat mean to match size of x
    sm = repmat(sm(:,1:size(dat,2)),1,1,size(dat,3));

    m = opts.moments;
    Moms = zeros(prod(dims(1:2)),nK,numel(m),'like',Data);
    % split number of pixels up into 12 groups?
    for mm = 1:numel(m)
        % compute moment
        if opts.std_norm
            Moms(:,:,mm) = mean(((dat-sm)./sd2).^m(mm),3);
        else
            Moms(:,:,mm) = mean((dat-sm).^m(mm),3);
        end
        if opts.verbose
            fprintf('Moment %d done\n',m(mm));
        end
    end

    if length(m) == 1
        Moms = rsND(Moms,[dims(1:2) nK]);
    else
        Moms = rsND(Moms,[dims(1:2) nK numel(m)]);
    end

    [N,M,K,P] = size(Moms);
    opts.moment_size = [N,M,K,P];

    % Normalize / standardize moments
    if opts.power_norm
        L = N*M*K
        p = repmat(1./opts.moments,L,1)
        f = @(x,m) sign(x)*abs(x).^m
        Moms = reshape(f(reshape(Moms,L,P),p),N,M,K,P)
    end
    if opts.zscore, Moms = myzscore(Moms,3); end
end
```

If background suppression is turned on, svd is used to remove the background:

```
function [Fn,Fb,out] = BG_suppress(F,BS,CompFactor,k,minK,opts)
    % [Fn,Fb,out] = BG_suppress(F, BS, k, minK)
    %
    % Function to model and remove static background (BG) from image stack. Can
    % operate on entire stack or break image into spatio-temporal blocks for
    % parallel processing of BG modeling. Uses SVD to model and remove BG
    % artifacts from the data in order to reveal dim moving targets. Unless k
    % specified, number of singular values kept for BG model is determined
    % automatically.
    %
    % --- Inputs ---
    % F    - frame stack [nRow,nCol,nFrame]
    % BS   - (*) block size [iRow, jCol, kFrame]
    % k    - (*) number of singular vectors for BG model {[], i.e. auto-detn'd}
    % minK - (*) minimum number of singular vectors to use in BG model {[1]}
    %
    % --- Outputs ---
    % Fn  - BG-suppressed, standardized data [nRow,nCol,nFrame]
    % Fb  - BG model [nRow,nCol,nFrame]
    % out - output structure with details about computation
    %
    % ---------------------------------
    % Shannon R. Young
    % POC: Bryan J. Steward
    % POC: Kevin C. Gross
    % Air Force Institute of Technology
    % Wright-Patterson AFB, Ohio
    % Kevin.Gross@afit.edu
    % (937) 255-3636 x4558
    % Version - Dissertation
    % 10-0ct-2018
    % ---------------------------------

    % Handle optional arguments
    if nargin < 3, BS = []; end
    if nargin < 4, k = [];  end
    if nargin < 5 || isempty(minK), minK = 1; end

    % Estimate and remove background
    ll = 1;
    tic; % track timing
    if isempty(BS) % Process the entire frame stack
        [Fn,Fb,K] = estimate_bg(F,k,minK);
    else % Loop to build indices for spatio-temporal blocks
        S = size(F);
        for ii = 1:BS(1)-1:S(1)
            for jj = 1:BS(2)-1:S(2)
                for kk = 1:BS(3)-1:S(3)
                    ixR{ll} = ii:min([ii+BS(1) S(1)]);
                    ixC{ll} = jj:min([jj+BS(2) S(2)]);
                    ixT{ll} = kk:min([kk+BS(3) S(3)]);
                    BLK{ll} = double(F(ixR{ll},ixC{ll},ixT{ll}));
                    ll = ll+1;
                end
            end
        end
        nBlk = ll-1;
        BLK = BLK(1:nBlk);

        % Parallel loop to perform background estimation on independent blocks
        K = zeros(1,nBlk);
        parfor ii = 1:nBlk
            [FN{ii},FB{ii},K(ii)] = estimate_bg(BLK{ii},CompFactor,k,minK);
        end

        % Reassemble blocks into background-suppressed frame stack
        [Fn,Fb] = deal(zeros(S,'like',F)); % Pre-allocate
        for ii = 1:nBlk
            Fn(ixR{ii},ixC{ii},ixT{ii}) = FN{ii};
            Fb(ixR{ii},ixC{ii},ixT{ii}) = FB{ii};
        end
    end


```

```
    % Build output structure
    out.k = K;
    out.ixR = ixR;
    out.ixC = ixC;
    out.ixT = ixT;
    out.nBlk = nBlk;
    if opts.verbose, time = toc; fprintf('BG suppress took %.3f s\n',time); end
end


% --------------------------------------------------------------------
% subfunction estimate_bg
% --------------------------------------------------------------------

function [Fn,Fb,k] = estimate_bg(F,CompFactor,k,minK)
    % code to estimate / suppress background for spatio-temporal chunk

    % Handle optional input
    if nargin < 4 || isempty(minK), minK = 1; end

    % Use SVD to remove background
    [F,dims] = rs2D(F);
    if CompFactor>0
        [U,S,V] = rsvd(F,round(dims(end)/CompFactor));
    else
        [U,S,V] = svdecon(F);
    end

    % Automate finding the number of singular vectors needed to model BG
    if isempty(k) % but only do this if k not supplied by user
        try
            s = diag(S);
            ix = find(s >= median(s) + 2*1.486*mad(s,1))';
            ix = [ix max(ix)+1];
            if isempty(ix), ix = 1:minK; end
        catch
            fprintf('Error finding number of vectors, default to all\n')
            ix = 1:size(F,2);
        end
    else
        ix = 1:min([k round(dims(end)/2)]);
    end
    k = numel(ix);

    % Hard-coded hack if k too big
    if k>round(dims(end)/2)
        ix = 1:10;
    end

    % Another check if k too big
    if k>size(U,2) || k>size(S,2) || k>size(V,2)
        k = size(U,2);
        ix = 1:k;
    end

    % Compute background and remove it
    Fb = U(:,ix)*S(ix,ix)*V(:,ix)'; % BG model
    Fn = F - Fb;                    % remove BG
    Fn = rsND(Fn,dims);             % reshape
    Fb = rsND(Fn,dims);             % reshape
end
```

Randomized SVD code used in background suppression:

```
function [U,S,V] = rsvd(A,K)
    % Code written by Antoine Liutkus and posted on MathWorks FileExchange.
    % 15-Sep-2014
    % https://www.mathworks.com/matlabcentral/fileexchange/47835-randomized-singular-value-decomposition
    %
    %-------------------------------------------------------------------------------------
    % random SVD
    % Extremely fast computation of the truncated Singular Value Decomposition, using
    % randomized algorithms as described in Halko et al. 'finding structure with ...
      randomness
    %
    % usage :
    %
    %  input:
    %  * A : matrix whose SVD we want
    %  * K : number of components to keep
    %
    %  output:
    %  * U,S,V : classical output as the builtin svd matlab function
    %-------------------------------------------------------------------------------------
    % Antoine Liutkus  (c) Inria 2014

    [M,N] = size(A);
    P = min(2*K,N);

    X = randn(N,P);

    Y = A*X;
    W1 = myorth(Y);
    B = W1'*A;
    [W2,S,V] = svdecon(B);
    U = W1*W2;
    K=min(K,size(U,2));
    U = U(:,1:K);
    S = S(1:K,1:K);
    V=V(:,1:K);
end

function Q = myorth(A)
    %ORTH   Orthogonalization.
    %   Q = ORTH(A) is an orthonormal basis for the range of A.
    %   That is, Q'*Q = I, the columns of Q span the same space as
    %   the columns of A, and the number of columns of Q is the
    %   rank of A.

    [Q,S] = svdecon(A); %S is always square.
    s = diag(S);
    tol = max(size(A)) * eps(max(s));
    r = sum(s > tol);
    Q(:, r+1:end) = [];
end
```

Economic SVD code used in background suppression:

```
function [U,S,V] = svdecon(X)
    % Code written by Vipin Vijayan and posted on MathWorks FileExchange.
    % 07-Jul-2014
    % https://www.mathworks.com/matlabcentral/fileexchange/47132-fast-svd-and-pca
    %
    % Input:
    % X : m x n matrix
    %
    % Output:
    % X = U*S*V'
    %
    % Description:
    % Does equivalent to svd(X,'econ') but faster
    %
    % Vipin Vijayan (2014)

    % X = bsxfun(@minus,X,mean(X,2));
    [m,n] = size(X);

    if  m <= n
        C = X*X';
        [U,D] = eig(C);
        clear C;

        [d,ix] = sort(abs(diag(D)),'descend');
        U = U(:,ix);

        if nargout >=2
            V = X'*U;
            s = sqrt(d);
            V = bsxfun(@(x,c)x./c, V, s');
            S = diag(s);
        end
    else
        C = X'*X;
        [V,D] = eig(C);
        clear C;

        [d,ix] = sort(abs(diag(D)),'descend');
        V = V(:,ix);

        U = X*V; % convert evecs from X'*X to X*X'. the evals are the same.
        % s = sqrt(sum(U.^2,1))';
        s = sqrt(d);
        U = bsxfun(@(x,c)x./c, U, s');
        S = diag(s);
    end
end
```

Use the significance level to determine which pixels are detects:

```
function [Dets]=mbd2_detect(Moms,opts,varargin)
    % Dets = mbd2_detect(Moms,opts,varargin)
    %
    % MBD function to take moments from mbd2_moms and find detects based on a
    % global threshold, treating each pixel as a detector
    %
    % --- Inputs ---
    % Moms - Moments array
    % opts - options structure with field alpha and zscore
    %
    % --- Outputs ---
    % Dets - threshold-based detections
    %
    % ---------------------------------
    % Shannon R. Young
    % POC: Bryan J. Steward
    % POC: Kevin C. Gross
    % Air Force Institute of Technology
    % Wright-Patterson AFB, Ohio
    % Kevin.Gross@afit.edu
    % (937) 255-3636 x4558
    % Version - Dissertation
    % 10-0ct-2018
    % ---------------------------------

    % process extra inputs
    opts = parse_pv_pairs(opts,varargin);

    % zscore if it hasn't been done already
    if ~opts.zscore, Moms = myzscore(Moms,3); end

    % get pixels that are above the threshold based on alpha
    Dets = Moms >= quantile(Moms(:), 1-opts.alpha);
end
```

Filter the detections using the voxel filtering approach:

```
function [Summed,DetsF] = BoxFilter2(Dets, opts, FigFlag, Summed)
    % [Summed,DetsF] = BoxFilter2(Dets,opts,FigFlag,Summed)
    %
    % Function to perform box filtering.
    %
    % --- Inputs ---
    % Dets    - binary data hypercube or cube with detections
    % opts    - opts structure from mbd2_opts used to generate Dets
    % FigFlag - if 1, make histogram of the summed data
    % Summed  - (*) if input, skips summing step and uses this instead
    %
    % --- Outputs ---
    % Summed - the array containg the sum of detects in each voxel
    % DetsF  - Filtered detections
    %
    % ---------------------------------
    % Shannon R. Young
    % POC: Bryan J. Steward
    % POC: Kevin C. Gross
    % Air Force Institute of Technology
    % Wright-Patterson AFB, Ohio
    % Kevin.Gross@afit.edu
    % (937) 255-3636 x4558
    % Version - Dissertation
    % 10-0ct-2018
    % ---------------------------------

    % get ideal lower and upper thresholds for the given options
    [Lower,Upper] = getBxThresh3(opts);

    % handle case of VERY bright target (blur across pixels causes this problem)
    if opts.SNRguess > 9, opts.upper_thresh = 1; end

    % change Dets to single to save space on the GPU
    Dets = single(Dets);

    % if user didn't supply Summed, we need to compute it
    if nargin < 4
        Box = opts.box;
        BS = [Box, Box, Box];
        kernel = ones(BS);
        Summed = zeros(size(Dets));
        for ii = 1:size(Dets,4)
            Summed(:,:,:,ii) = convn(Dets(:,:,:,ii),kernel,'same');
        end
    end

    % prepare figure, if necessary
    if FigFlag, figure; hold on; end

    % threshold
    SubArray = zeros(size(Dets));
    for ii = 1:size(Dets,4)
        SubArray(:,:,:,ii) = Summed(:,:,:,ii)>Lower(ii) & Summed(:,:,:,ii)<Upper(ii);
        if FigFlag, histogram(Summed(:,:,:,ii),'BinWidth',1,'linestyle','none'); end
        DetsF = any(Dets.*SubArray,4);
    end

    % apply final filtering, if necessary
    if opts.FinalFilt && nargout > 1
        Box = opts.box;
        BS = [Box, Box, Box];
        % check that initial alpha is low enough for this to work:
        if opts.alpha > 0.5 && opts.verbose
            fprintf('Final 2D summed filter not used; initial threshold too high\n')
        else
            Map = sum(DetsF,3);
            FlatSum = conv2(Map,ones(BS(1:2)),'same'); % convolve box size over 2D map
            DetsF = bsxfun(@times,DetsF,FlatSum>min(Lower));
        end
    end
end
```

Use a helper function to determine the filtering threshold:

```
function [Lower,Upper,Ls,Lt] = getBxThresh3(opts)
    % [Lower,Upper,Ls,Lt] = getBxThresh3(opts)
    %
    % Function to compute lower and upper bounds for filtering box detects
    % based on the mbd2_opts structure (the only input).
    %
    % --- Inputs --- (required fields from the opts structure)
    % window_size - moment temporal window
    % dwell       - guess as to how long target spends in a pixel (choose
    %               R = 1 if totally unknown)
    % upper_limit - qunatile of binomial distribution used to set upper
    %               threshold. Set to 1 if no upper threshold is desired
    % box         - dimension of voxel over which detects are summed
    % SNRguess    - used to calculate how much of streak should be detected
    %
    % --- Outputs ---
    % Lower - lower threshold for detects
    % Upper - upper threshold for detects
    % Ls    - spatial streak length
    % Lt    - temporal streak length
    %
    % ---------------------------------
    % Shannon R. Young
    % POC: Bryan J. Steward
    % POC: Kevin C. Gross
    % Air Force Institute of Technology
    % Wright-Patterson AFB, Ohio
    % Kevin.Gross@afit.edu
    % (937) 255-3636 x4558
    % Version - Dissertation
    % 10-0ct-2018
    % ---------------------------------

    % rename important variables
    W = opts.window_size;
    R = opts.dwell;
    Bx = opts.box;

    % define the necessary streak length functions
    Lspat = @(R,W) (W+R-1)/R;
    LspatFast = @(R,W) W; % don't divide by R yet if R<1
    Ltemp = @(R,W) W+R-1;
    LtempFast = @(R,W) W+R-1;

    if R < 1
        if LspatFast(R,W)>=Bx
            Ls = Bx;
        else
            Ls = LspatFast(R,W);
        end
        if LtempFast(R,W)>= Bx
            Lt = Bx;
        else
            Lt = LtempFast(R,W);
        end
    else
        if Lspat(R,W)>=Bx
            Ls = Bx;
        else
            Ls = Lspat(R,W);
        end
        if Ltemp(R,W)>=Bx
            Lt = Bx;
        else
            Lt = Ltemp(R,W);
        end
    end

    % total expected detects for a straight line
    ExpectedMin = Ls.*Lt;

    % total expected detects for a diagonal target
    % (assume stair style, not solid diagonal)
    ExpectedMax = Ls.*Lt*2;
```

```
75
76      % now load in data and compute expected percent detected
77      try
78          % try to use pre-generated data for speed
79          load('PercDetStats5.mat','MomVec','Rvec','SNRVec','AlphaVec',...
80               'WindowVec','truePositive')
81          if opts.alpha < 0.001
82              Error('need to recompute')
83          end
84
85          % set limits to the edges of computed zone
86          if opts.dwell<1
87              opts.dwell = 1;
88          end
89
90          PercMn = interpn(AlphaVec,MomVec,Rvec,WindowVec,SNRVec,truePositive,...
91              double(opts.alpha),double(opts.moments),double(opts.dwell),...
92              double(opts.window_size),double(opts.SNRguess),'spline');
93
94      catch
95          % if not available, compute PercMn from scratch
96          W = opts.window_size;
97          SNR = opts.SNRguess;
98          Dwell = opts.dwell;
99          Std = opts.data_std;
100         muB = opts.data_mean;
101         NumRuns = 10^5; NumBins = 5*10^3; Ws = opts.super_window;
102         TgtMean = muB+SNR*Std;
103         BkgPix = randn(NumRuns,1)*Std+muB;
104         TgtPix = randn(NumRuns,1)*Std+TgtMean;
105         PercMn = zeros(length(opts.moments),1);
106         for nn = 1:length(opts.moments)
107             IdxBkg = randi(length(BkgPix(:)),NumRuns,Ws);
108             X_Bkg = BkgPix(IdxBkg);
109             IdxTgtBkg = randi(length(TgtPix(:)),NumRuns,int32(Dwell));
110             tmp2 = TgtPix(IdxTgtBkg);
111             X_Tgt = cat(2,X_Bkg(:,1:end-Dwell),tmp2);
112             smBkg = mean(X_Bkg,2);
113             smTgt = smBkg; % Exclued target window completely in mean calc
114             stdBkg = std(X_Bkg,[],2);
115             stdTgt = stdBkg;
116             Mom = opts.moments(nn);
117
118             BkgMom = mean(((X_Bkg(:,1:W)-repmat(smBkg,1,W))./...
119                 repmat(stdBkg,1,W)).^(Mom),2);
120             TgtMom = mean(((X_Tgt(:,end-W+1:end)-repmat(smTgt,1,W))./...
121                 repmat(stdTgt,1,W)).^(Mom),2);
122
123             % set bins based on quantiles?
124             Bins = linspace(quantile(BkgMom(:),.001),quantile(TgtMom(:),.99),NumBins);
125
126             [Values1, Edges1] = histcounts(BkgMom,Bins,'normalization','cdf');
127             [Values2, Edges2] = histcounts(TgtMom,Bins,'normalization','cdf');
128             [~, Edges3] = histcounts([TgtMom;BkgMom],Bins,'normalization','cdf');
129             tmp1 = interp1(Edges1(1:end-1),Values1,Edges3(1:end-1),'linear','extrap');
130             tmp2 = interp1(Edges2(1:end-1),Values2,Edges3(1:end-1),'linear','extrap');
131
132             tmp1(tmp1<0)=0;
133             tmp1(tmp1>1)=1;
134             tmp2(tmp2<0)=0;
135             tmp2(tmp2>1)=1;
136
137             PF = 1-tmp1'; PD = 1-tmp2';
138             if ~isempty(find(PF<opts.alpha,1,'first'))
139                 PercMn(nn) = PD(find(PF<opts.alpha,1,'first'));
140             else
141                 PercMn(nn) = PD(end);
142             end
143         end
144     end
145
146     % Now add in the opts.lower_thesh quantile noise for the minimum and
147     % opts.upper_thresh noise for the maximum
148     Lower = PercMn.*ExpectedMin + binoinv(opts.lower_thresh,Bx^3,opts.alpha);
149     Upper = PercMn.*ExpectedMax + binoinv(opts.upper_thresh,Bx^3,opts.alpha);
150 end
```

Function to apply a z-score transformation of the data:

```
function [z,mu,sigma] = myzscore(x, dim)
    % [z,mu,sigma] = myzscore(x, dim)
    %
    % Function that returns standardized (z-score) data, but modified to throw
    % out outliers in the standard deviation to prevent wild fluctuations when
    % clouds are present in BG-suppressed data.
    %
    % --- Inputs ---
    % x   - array to be standardized
    % dim - dimension along which standardization occurs
    %
    % --- Outputs ---
    % z   - standardized data
    % mu  - mean of x along dim
    % std - std of x along dim
    %
    % ---------------------------------
    % Shannon R. Young
    % POC: Bryan J. Steward
    % POC: Kevin C. Gross
    % Air Force Institute of Technology
    % Wright-Patterson AFB, Ohio
    % Kevin.Gross@afit.edu
    % (937) 255-3636 x4558
    % Version - Dissertation
    % 10-0ct-2018
    % ---------------------------------

    % compute mean and standard deviation
    mu = mean(x, dim);
    sigma = std(x,[],dim);

    % handle data values that might have almost no variance
    Thresh = quantile(vec(sigma), 0.01);
    sigma(sigma<Thresh) = Thresh;

    % handle data values that have no variance
    sigma0 = sigma;
    sigma0(sigma0==0) = 1;

    % apply z-score transform
    z = bsxfun(@rdivide, bsxfun(@minus,x, mu), sigma0);
end
```

Convenience functions for reshaping data:

```
function x = vec(x)
    % reshape to vector
    x = x(:);
end
```

```
function [y,dims] = rs2D(y)
    % reshape to 2D array
    dims = size(y);
    y = reshape(y,[prod(dims(1:end-1)) dims(end)]);
end
```

```
function y = rsND(y,dims)
    % reshape to ND array
    dims(end) = size(y,ndims(y));
    y = reshape(y,dims);
end
```

Helper function for parsing optional field/value pairs:

```
function params=parse_pv_pairs(params,pv_pairs)
    % Code written by John D'Errico and posted on MathWorks FileExchange.
    % 16-Jan-2006
    % https://www.mathworks.com/matlabcentral/fileexchange/9082-parse_pv_pairs
    % Small modification made by Kevin Gross 13-Oct-2007 on lines 67-71
    %
    % parse_pv_pairs: parses sets of property value pairs, allows defaults
    % usage: params=parse_pv_pairs(default_params,pv_pairs)
    %
    % arguments: (input)
    %  default_params - structure, with one field for every potential
    %             property/value pair. Each field will contain the default
    %             value for that property. If no default is supplied for a
    %             given property, then that field must be empty.
    %
    %  pv_array - cell array of property/value pairs.
    %             Case is ignored when comparing properties to the list
    %             of field names. Also, any unambiguous shortening of a
    %             field/property name is allowed.
    %
    % arguments: (output)
    %  params   - parameter struct that reflects any updated property/value
    %             pairs in the pv_array.
    %
    % Example usage:
    % First, set default values for the parameters. Assume we
    % have four parameters that we wish to use optionally in
    % the function examplefun.
    %
    %  - 'viscosity', which will have a default value of 1
    %  - 'volume', which will default to 1
    %  - 'pie' - which will have default value 3.141592653589793
    %  - 'description' - a text field, left empty by default
    %
    % The first argument to examplefun is one which will always be
    % supplied.
    %
    %   function examplefun(dummyarg1,varargin)
    %   params.Viscosity = 1;
    %   params.Volume = 1;
    %   params.Pie = 3.141592653589793
    %
    %   params.Description = '';
    %   params=parse_pv_pairs(params,varargin);
    %   params
    %
    % Use examplefun, overriding the defaults for 'pie', 'viscosity'
    % and 'description'. The 'volume' parameter is left at its default.
    %
    %   examplefun(rand(10),'vis',10,'pie',3,'Description','Hello world')
    %
    % params =
    %     Viscosity: 10
    %        Volume: 1
    %           Pie: 3
    %   Description: 'Hello world'
    %
    % Note that capitalization was ignored, and the property 'viscosity'
    % was truncated as supplied. Also note that the order the pairs were
    % supplied was arbitrary.

    if nargin == 1, return; end

    npv = length(pv_pairs);
    n = npv/2;

    % MODIFIED KCG 13-Oct-2007
    if npv == 1 && isstruct(pv_pairs)
        params = pv_pairs;
        return;
    end

    if npv == 1 && iscell(pv_pairs) && isstruct(pv_pairs{1})
        params = pv_pairs{1};
```

```
        return;
    end

    if n~=floor(n),
        error 'Property/value pairs must come in PAIRS.'
    end
    if n<=0,
        % just return the defaults
        return
    end

    if ~isstruct(params)
        error 'No structure for defaults was supplied'
    end

    % there was at least one pv pair. process any supplied
    propnames = fieldnames(params);
    lpropnames = lower(propnames);
    for i=1:n
    p_i = lower(pv_pairs{2*i-1});
    v_i = pv_pairs{2*i};

    ind = strmatch(p_i,lpropnames,'exact');
    if isempty(ind)
        ind = find(strncmp(p_i,lpropnames,length(p_i)));
        if isempty(ind)
        error(['No matching property found for: ',pv_pairs{2*i-1}])
        elseif length(ind)>1
        error(['Ambiguous property name: ',pv_pairs{2*i-1}])
        end
    end
    p_i = propnames{ind};

    % override the corresponding default in params
    params = setfield(params,p_i,v_i);
end
```

Bibliography

1. Air Force Space Command - Fact Sheet: Space Based Infrared Systems. https://www.afspc.af.mil/About-Us/Fact-Sheets/Display/Article/1012596/space-based-infrared-system/, 2017.

2. N. Acito, G. Corsini, M. Diami, and G. Pennucci. Experimental performance analysis of clutter removal techniques in IR images. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–561–4. IEEE, 2005.

3. Steven Adler-Golden, Steven C. Richtsmeier, Alexander Berk, and James W. Duff. Fast Monte Carlo-assisted Simulation of Cloudy Earth Backgrounds. page 85340F. International Society for Optics and Photonics, nov 2012.

4. H. Askar, Xiaofeng L, and Zaiming L. Background clutter suppression and dim moving point targets detection using nonparametric method. In *Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on*, volume 2, pages 982–986. IEEE, 2002.

5. T. Bae. Small target detection using bilateral filter and temporal cross product in infrared images. *Infrared Physics and Technology*, 54(5):403–411, 2011.

6. Yair Barniv. Solution for Detecting Dim M oving Targets. *Ieee Transactions On Aerospace And Electronic Systems*, (1), 1985.

7. Alexander Berk, Patrick Conforti, Rosemary Kennett, Timothy Perkins, Frederick Hawes, and Jeannette van den Bosch. MODTRAN6: a major upgrade of the MODTRAN radiative transfer code. page 90880H. International Society for Optics and Photonics, jun 2014.

8. Christoph C Borel, David J Bunker, and Lori A Mahoney. Statistical Moments Based Methods for Detecting Sub-pixel Target Tracks In Large Image Sequences, 2014.

9. Edmund Førland Brekke. *Clutter mitigation for target tracking*. PhD thesis, 2010.

10. Jiah Chen and Irving Reed. A Detection Algorithm for Optical Targets in Clutter. *IEEE Transactions on Aerospace and Electronic Systems*, AES-23(1):46–59, 1987.

11. J. Claff, William. Photons to Photos, 2017.

12. S. J. Davey, M. G. Rutten, and B. Cheung. A comparison of detection performance for several track-before-detect algorithms. *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008*, 2008, 2008.

13. Samuel J. Davey and Han X. Gaetjens. *Track-Before-Detect Using Expectation Maximisation*. Signals and Communication Technology. Springer Singapore, Singapore, 2018.

14. E. L. Dereniak and G. D. Boreman. *Infrared Detectors and Systems*, volume 24. Wiley-Interscience, New York, 1996.

15. Krzysztof Domino, Piotr Gawron, and Łukasz Pawela. Efficient Computation of Higher Order Cumulant Tensors. *SIAM Journal on Scientific Computing*, 40(3):A1590–A1610, 2018.

16. Kenneth Flamm. Measuring Moore's Law: Evidence from Price, Cost and Quality Indexes. *National Bureau of Economic Research*, pages 1–46, 2018.

17. Manuel Guizar. Efficient subpixel image registration by cross-correlation. https://www.mathworks.com/matlabcentral/fileexchange/18401-efficient-subpixel-image-registration-by-cross-correlation, 2016. Retrieved 3 March 2017.

18. Graham Gyatt. The Standard Atmosphere, 1976. A mathematical model of the U.S. Standard Atmosphere.

19. R E Kalman. A New Approach to Linear Filtering and Prediction Problems 1. 82(Series D):35–45, 1960.

20. Steven M. Kay. *Fundamentals of Statistical Signal Processing: Detection Theory*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1998.

21. Christopher Keefer. *Infrared Target Detection: Signal and Noise Sensitivity Analysis*. PhD thesis, Air Force Institute of Technology, 1989.

22. Robert Kieser, Pall Reynisson, Timothy J Mulligan Kieser, R Kieser, and T J Mulligan. Definition of signal-to-noise ratio and its critical role in split-beam measurements. *ICES Journal of Marine Science*, (62):123–130, 2005.

23. D.A. Lacher. Sampling Distribution of Skewness and Kurtosis. Technical Report 2, 1989.

24. Sergei Leonov. Nonparametric methods for clutter removal. *IEEE Transactions on Aerospace and Electronic Systems*, 37(3):832–848, 2001.

25. H Leung. Nonlinear clutter cancellation and detection using a memory-based predictor. *IEEE Transactions on Aerospace and Electronic Systems*, 32(4):1249–1256, 1996.

26. H Leung and S Haykin. Detection and estimation using an adaptive rational function filter. *Signal Processing, IEEE Transactions on*, 42(12):3366–3376, 1994.

27. Alyssa Leung, Henry; Young. Small Target Detection in Clutter Using Recursive Nonlinear Predicton. *Aerospace and Electronic Systems Magazine, IEEE*, 36(3):713–718, 1999.

28. D. M. Lindholm, A. Ware DeWolfe, A. Wilson, C. K. Pankratz, M. A. Snow, and T. N. Woods. Solar Irradiance Data Products at the LASP Interactive Solar IRradiance Datacenter (LISIRD). *AGU Fall Meeting Abstracts*, pages GC23A–0931, December 2011.

29. a. Margalit, I.S. Reed, and R.M. Gagliardi. Adaptive Optical Target Detection Using Correlated Images. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21(3):394–405, 1985.

30. Stephen J. Marshall. We Need To Know More About Infrared Emissivity. pages 119–128. International Society for Optics and Photonics, mar 1982.

31. NS. Fast computation of intersections and self-intersections of curves using vectorization. https://www.mathworks.com/matlabcentral/fileexchange/22441-curve-intersections, 2010.

32. Rodriguez Paul and Brendt Wohlberg. Fast Principal Component Pursuit via Alternating Minimization. *Image Processing (ICIP), 2013 IEEE International Conference on*, pages 69–73, 2013.

33. Steven Richtsmeier and Robert Sundberg. Recent Advances in the Simulation of Partly Cloudy Scenes. volume 7827, pages 78270S–78270S–8, oct 2010.

34. J.R. Schott, S.D. Brown, R.V. Raqueño, H.N. Gross, and G. Robinson. An Advanced Synthetic Image Generation Model and its Application to Multi/Hyperspectral Algorithm Development. *Canadian Journal of Remote Sensing*, 25(2):99–111, jun 1999.

35. Christopher Schwartz, Ralf Sarlette, Michael Weinmann, Martin Rump, and Reinhard Klein. Design and Implementation of Practical Bidirectional Texture Function Measurement Devices focusing on the Developments at the University of Bonn. *Sensors*, 14(5):7753–7819, apr 2014.

36. Lawrence D. Stone, Carl A. Barlow, and Thomas L. Corwin. *Bayesian Multiple Target Tracking*. Artech House, 1999.

37. Roy L. Streit, Marcus L. Graham, and Michael J. Walsh. Multitarget Tracking of Distributed Targets Using Histogram-PMHT. *Digital Signal Processing*, 12(2-3):394–404, jan 2002.

38. Dennis D. Wackerly, William Mendenhall, and Richard L. Scheaffer. *Mathematical Statistics with Applications*. Thomson Learning, 7 edition, 2008.

39. Xin Yang, Yanpei Zhou, Dake Zhou, Ruigang Yang, and Yinji Hu. A new infrared small and dim target detection algorithm based on multi-directional composite window. *Infrared Physics and Technology*, 71:402–407, 2015.

40. Shannon Young. *Improving Detection of Dim Targets: Optimization of Moment-Based Detection Using Statistical Confidence*. PhD thesis, Air Force Institute of Technology, 2016.

41. Shannon Young and Jeffrey Sovern. Characterizing the Temperature and Angle Effects on Emissivity of a Wide Area Blackbody, 2015.

42. Shannon R Young, Bryan J Steward, and Kevin C Gross. Development and Validation of the AFIT Sensor and Scene Emulator for Testing (ASSET). In *Infrared Imaging Systems: Design, Analysis, Modeling and Testing XXVIII*, page 101780A, 2017.

43. Shannon R. Young, Bryan J. Steward, Michael Hawks, and Kevin C. Gross. Improving detection of low SNR targets using moment-based detection. page 98280K, may 2016.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
10-18-2018	Dissertation	Jan 2015 — Oct 2018

4. TITLE AND SUBTITLE

Improving Detection of Dim Targets:
Optimization of a Moment-based Detection Algorithm

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Young, Shannon R., Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENP-DS-18-D-010

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Geospatial-Intelligence Agency
Ms. Lori Mahoney
4180 Watson Way
Wright-Patterson AFB, OH 45433-7765
(937)-672-2229 lori.mahoney.2@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

NGA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A: Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States

14. ABSTRACT

Wide area motion imagery (WAMI) sensor technology is advancing rapidly. Increases in frame rates and detector array sizes have led to a dramatic increase in the volume of data that can be acquired. Without a corresponding increase in analytical manpower, much of these data remain underutilized. This creates a need for fast, automated, and robust methods for detecting dim, moving signals of interest. Current approaches fall into two categories: detect-before-track (DBT) and track-before-detect (TBD) methods. The DBT methods use thresholding to reduce the quantity of data to be processed, making real time implementation practical but at the cost of the ability to detect low signal to noise ratio (SNR) targets without acceptance of a high false alarm rate. TBD methods exploit both the temporal and spatial information simultaneously to make detection of low SNR targets possible, but at the cost of computation time. This research seeks to contribute to the near real time detection of low SNR, unresolved moving targets through an extension of earlier work on higher order moments anomaly detection, a method that exploits both spatial and temporal information but is still computationally efficient and massively parallellizable. The MBD algorithm was found to detect targets comparably with leading TBD methods in 1000^{th} the time.

15. SUBJECT TERMS

tracking, detection, remote sensing, track-before-detect (TBD), detect-before-track (DBT), statistics, higher order moments

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kevin Gross, AFIT/ENP
U	U	U	U	166	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4558; kevin.gross@afit.edu