REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 24-10-2006	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From – To)* 1 September 2003 - 01-Sep-06

4. TITLE AND SUBTITLE Spatial and Temporal Point Tracking in Real Hyperspectral Images	**5a. CONTRACT NUMBER** FA8655-03-1-3077
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Dr. Stanley Rotman	**5d. PROJECT NUMBER**
	5d. TASK NUMBER
	5e. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Ben-Gurion University of the Negev, PO Box 653, Beer-Sheva 84 105, Israel	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD, PSC 821 BOX 14, FPO AE 09421-0014	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) Grant 03-3077

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking Ben-Gurion University of the Negev as follows: The Grantee will investigate developing a full target detection module which analyzes a temporal sequence of hyperspectral datacubes for the acquisition of infrared targets. They will test this module as a function of target strength and background clutter to test for its ruggedness in its ability to detect targets with low false alarm rates. They will document its improvement over systems which use the temporal and hyperspectral data separately. This is the final report.

15. SUBJECT TERMS

EOARD, Hyperspectral Imagery, Optical Sensors, Optical sensing

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18, NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			GEORGE W YORK, Lt Col, USAF
UNCLAS	UNCLAS	UNCLAS	UL	153	**19b. TELEPHONE NUMBER** *(Include area code)* +44 (0)20 7514 4354

BEN-GURION UNIVERSITY OF THE NEGEV

FACULTY OF ENGINEERING SCIENCES

UNIT OF ELECTROPTICS ENGINEERING

Spatial and temporal point tracking in real hyperspectral images

Stanley R. Rotman

Ben-Gurion University of the Negev

Dept. of Elec. And Comp. Eng.

Final Report for United States AFOSR IRI Grant 033077

26 August 2006

Abstract

The scope of this project addresses the problem of tracking a dim moving point target from a sequence of hyperspectral cubes. The resulting tracking algorithm is useful for many staring technologies such as the ones used in space surveillance and missile tracking applications. In these applications, the images consist of targets moving at sub-pixel velocity and noisy background consisting of evolving clutter and noise. The demand for a low false alarm rate (*FAR*) on one hand and a high probability of detection (P_D) on the other makes the tracking a challenging task. The use of hyperspectral images should be superior to current technologies using broadband IR images due to the ability of exploiting simultaneously two target specific properties: the spectral target characteristics and the time dependent target behavior.

The proposed solution consists of three stages: the first stage transforms the hyperspectral cubes into a two dimensional sequence, using known point target detection acquisition methods; the second stage involves a temporal separation of the 2D sequence into sub-sequences and the usage of a variance filter (*VF*) to detect the presence of targets from the temporal profile of each pixel in each group, while suppressing clutter specific influences. This stage creates a new sequence containing a target with a seemingly faster velocity; the third stage applies the Dynamic Programming Algorithm (*DPA*) that proves to be a very effective algorithm for the tracking of moving targets with low SNR at around pixel velocity.

The system is tested on both synthetic and real data.

Acknowledgements

We would like to thank *Dr. Jerry Silverman* and *Mrs. Charlene Caefer* for supplying us with the IR sequences, [16] .

Ben-Gurion University graduate students participating and contributing to this project were: Ms. Roni Succary, Mr. Eran Ohel, Ms. Vardit Adler, Ms. Louisa Versano, Mr. Yuval Cohen, Mr. Shlomi Buganim, Mr. Simon Adar, Mr. Yoni Simson, Mr. Boris Efros, Ms. Irena Yatskaer, Mr. Ofir Nichtern, and Mr. Benjamin Aminov.

This work was sponsored by an United States AFOSR IRI Grant 033077.

Glossary and Acronyms

AM	-	Anti-median Metric
CF	-	Cloud Filter
CFAR	-	Constant False Alarm Rate
DC	-	Direct Current
DPA	-	Dynamic Programming Approach
FF	-	Fusion Filter
GLRT	-	Generalized Likelihood Ration Test
HSI	-	Hyperspectral Imagery
IID	-	Independent Identically Distributed
IR	-	Infra Red
IRST	-	Infra Red Search and Track
LR	-	Likelihood Ratio
LRT	-	Likelihood Ratio Test
LSE	-	Least Squares Estimation
MF	-	Match Filter
NF	-	Noise Filter
PCA	-	Principle Components Analysis
PDF	-	Probability Density Function
PDF	-	Probability Density Function
ROC	-	Receiver Operating Characteristics
RX	-	Reed Xiaoli algorithm
SNR	-	Signal to Noise Ratio
STD	-	Standard Deviation
TBD	-	Track Before Detect
TTF	-	Triple Temporal Filter
UMP	-	Uniformly Most Powerful
VF	-	Variance Filter
WGN	-	White Gaussian Noise

Table of Contents

List of Figures

List of Tables

Keywords

Hyperspectral target detection, tracking point targets, temporal filters, point target detection, point target tracking, temporal filter, IRST, variance filter, DPA, metrics, cloud clutter, staring camera, consecutive frames, real-world data, hyperspectral imagery.

1 Introduction

The goal of the research in the scope of the project is to propose a unique system for the tracking of dim point targets moving at sub-pixel velocities in a sequence of hyperspectral cubes or, simply put, a hyperspectral movie. Our research incorporates algorithms from two different areas – target detection in hyperspectral imagery and target tracking in IR sequences. Numerous works have addressed each of these problems separately, but to the best of our knowledge no attempts were made, so far, to combine the two fields.

We chose the most intuitive approach to tackle the problem, namely divide and conquer – separation of the problem into three sub-problems and solving each one separately using a combination of approaches. Thus, we first transform each hyperspectral cube into a two dimensional image using a hyperspectral target detection method. The next step involves a temporal separation of the movie (images sequence) into sub movies and the usage of a variance filter (*VF*). The filter detects the presence of targets from the temporal profile of each pixel in each group while suppressing clutter specific influences. Afterwards, an accumulation of the results from several consecutive images is done using a Track Before Detect (*TBD*) approach, implemented by the Dynamic Programming Algorithm (*DPA*), to perform detection in the time domain. Performance metrics are defined for each step, and are used in the analysis and optimization of each step.

In order to evaluate the complete system, we need to obtain a hyperspectral movie. Since this kind of data is not available to us yet, an algorithm is developed for the creation of a hyperspectral movie, based on a real world IR sequence and real-world signatures, including an implanted synthetic moving target.

This paper is organized as follows: Sections 2 and 3 provide a short overview of the basic methods for target detection in hyperspectral imagery and the time and spatial processing methods for target tracking in IR imagery on which we base our algorithm. Section 4 presents the suggested system. Section 5 presents the evaluation of the full system on real and synthetic data. Section 6 ends the report describing conclusions and ideas for future research.

2 Point target detection in hyperspectral imagery

2.1 *Hyperspectral imagery overview*

2.1.1 Hyperspectral imagery basics

Hyperspectral imagery (*HSI*) exploits the fact that different materials reflect, absorb and emit electromagnetic radiation in ways characteristic of their molecular composition and structure [1] . The radiation arriving at the hyperspectral sensor is measured at each wavelength over a sufficiently broad spectral band, and the resulting spectral signature, or simply spectrum, can be used to uniquely characterize and identify any given material.

Hyperspectral sensors represent an advance on technology from the earlier multispectral sensors, which were able to collect information in only a few discrete and noncontiguous bands. Current hyperspectral sensors sample the reflective portion of the electromagnetic spectrum that extends from the visible region (0.4 - 0.7 μm) through the near infrared (about 2.4 μm) in hundreds of narrow contiguous bands about 10 *nm* wide. Other types of hyperspectral sensors exploit the emissive properties of materials by collecting data in the mid-wave and long-wave infrared regions of the spectrum. From a spatial point of view, current sensors are capable of spatial sampling (or ground pixel size) in the resolution of several meters to tens of meters, depending on the sensor aperture and platform altitude (mainly space borne vs. airborne sensor). For example, the sensor system called *Hyperion*, carried on the experimental *NASA EO-1* spacecraft launched in November 2000, has 220 spectral bands, 30 *m* pixels and 10 *bit* data system [2] .

The spatially and spectrally sampled information can be described as a data cube, whose face is a function of the spatial coordinates and depth is a function of spectral band, as shown in Figure 1.

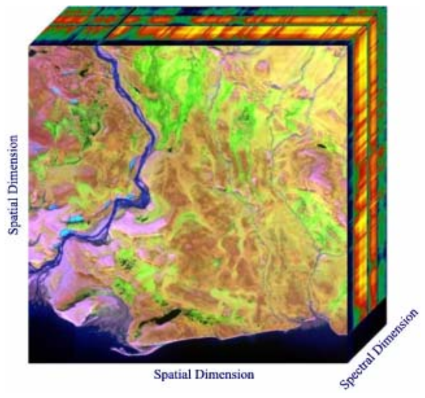


Figure 1: Illustration of the hyperspectral cube data.

There are many effects which influence the data collected at the sensor, besides of course the materials themselves. Some of these are:

- The angle of the sun.
- The viewing angle of the sensor.
- The upwelling solar radiance from atmospheric scattering.
- The secondary illumination of the material by light reflected from adjacent objects in the scene.
- Shadowing.
- The scattering and absorption of the reflected radiance by the atmosphere.
- Spatial and spectral aberrations in the sensor.

All these make the processing of the hyperspectral data a challenging task requiring sophisticated signal processing algorithms and models.

2.1.2 The basic steps of hyperspectral signal processing

The basic hyperspectral processing chain is given in Figure 2. There are two important preprocessing steps which should be performed before the actual processing of the data begins. These are shortly described in the following subsections.

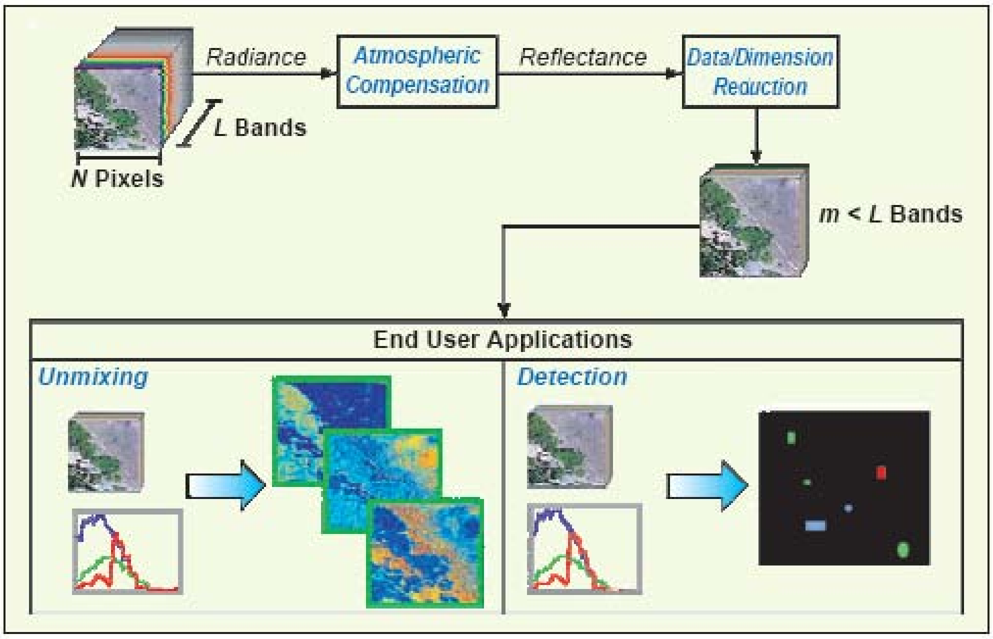


Figure 2: <u>Illustration of hyperspectral imagery processing chain</u>.

2.1.2.1 Atmospheric compensation

The first preprocessing step is atmospheric compensation or atmospheric calibration. This step is needed in order to correct the data for the effects of atmospheric absorption and scattering.

The atmosphere absorbs and scatters light in a wave-length-dependent fashion [3] . Thus, the “raw” data from the sensor cannot be directly compared to either laboratory spectra or “raw” spectra collected at other times or places. In order to apply signal detection algorithm, the reflectance or emissive spectra of the objects of interest must be converted into radiance at the sensor, or the radiance data collected by the sensor must be converted into reflectance or emission. This process is known as atmospheric compensation.

The simplest method to compensate for the environmental and atmospheric effects is to place a calibration panel, with known reflectance in the scene, in an open area, and use the observed radiance spectrum from the panel to develop gain and offset corrections for each waveband of interest. Other more sophisticated methods are the physics-based modeling methods. These include processes to quantify the effect of water vapor on the measurements, estimates of terrain height and aerosol optical depth (visibility).

2.1.2.2 Dimensionality reduction

The next important step is dimensionality reduction. In most cases, hyperspectral sensors over sample the spectral signal to ensure that any narrow features are adequately represented. Dimensional reduction is often desirable because it leads to significant reductions in computational complexity and reduction of the number of pixels required to obtain statistical estimates (the number of pixels required to obtain statistical estimate with a given accuracy increases drastically with the dimensionality of the data). Since the dimensional reduction involves data loss, it is important to make sure that high-quality features needed for detection, discrimination and classification of the data are preserved.

The most common algorithm for dimensionality reduction is principal component analysis (*PCA*) which is based on the *Karhunen - Loeve* linear transformation [4] . Further development of that transformation can be done using a kernel, a method known as K-PCA, which is a non-linear transformation.

2.1.3 Applications

The number and variety of civilian and military applications for hyperspectral remote sensing is enormous. The majority of algorithms used in these applications might be roughly divided into four main classes:

1. Target detection – defined as searching for the pixels of the hyperspectral data cube which exhibit similar spectral properties to the desired spectral signature (in case the desired signature is known) or pixels which differ significantly from their surrounding (anomaly detection) [1] , [5] .

2. Change detection – defined as finding the predefined significant changes between two hyperspectral scenes of the same geographic region [1] .

3. Classification – assigning a label (class) to each pixel of the hyperspectral data cube [1] .

4. Spectral unmixing – estimating the fraction of each material in a given pixel [6] .

This research focuses on the subject of target detection and tracking, which will be further detailed in the next section.

2.2 *Algorithms for target detection in hyperspectral imagery*

2.2.1 Problem Statement

In target detection applications, the goal is to search the pixels of a hyperspectral data cube for the presence of a specific material (target). In surveillance applications, the size of the objects (targets) we are searching for constitutes a very small fraction of the total search area, the targets size is only several pixels. The term “background” is used to refer to all non target pixels of a scene. Usually, targets are man-made objects with spectra that differ from the spectra of natural background pixels.

2.2.2 General framework for target detection algorithms

Most hyperspectral data processing techniques assume that each observation can be modeled as random vector whose dimension is the number of spectral bands and which originates from specific probability distribution, *p(x)*. Given an observed spectrum *x*, the likelihood ratio (*LR*) is given by the ratio of the conditional probability density functions:

(2.1)
$$LR(x) \equiv \frac{p(x|signal\ present)}{p(x|signal\ absent)}$$

If *LR(x)* is larger than the threshold η, the “signal present” hypothesis is accepted; otherwise the “signal absent” hypothesis is accepted:

(2.2)
$$LR(x) \underset{H_0}{\overset{H_1}{\gtrless}} \eta$$

2.2.3 Matched filter algorithms

The expression for the Matched Filter detector is derived here from the likelihood ratio presented in section 2.2.2. In order to build a likelihood ratio detector there is a need to define the statistical model of the data, namely to define the probability distribution functions. Normal probability based models are simple and often lead to good performance.

The multivariate normal distribution of random vector x is denoted by x ~ N (μ,Φ); the mean vector μ and the covariance matrix Φ are given in the following:

(2.3)
$$\begin{aligned}\mu &\equiv E\left[x\right]\\ \Phi &\equiv E\left[\left(x-\mu\right)\left(x-\mu\right)^T\right]\end{aligned}$$

where E [•] denotes the expectation operator.

The probability density function (*PDF*) of *x* is given by (normal distribution function):

(2.4)
$$p\left(x\right)=\frac{1}{\left(2\pi\right)^{N/2}\left|\Phi\right|^{1/2}}\exp\left(-\frac{1}{2}\left(x-\mu\right)^T\Phi^{-1}\left(x-\mu\right)\right)$$

where |Φ| is the determinant of the covariance matrix Φ.

For target and background spectra modeled as multivariate normal vectors, the detection problem can be specified by the following hypotheses:

(2.5)
$$\begin{aligned}H_0 &: x \sim N\left(\mu_b,\Phi_b\right)\\ H_1 &: x \sim N\left(\mu_t,\Phi_t\right)\end{aligned}$$

where H_0 is the null hypothesis assuming absence of target, H_1 assumes presence of target; *b* and *s* denote the background and target parameters, respectively.

The natural logarithm of the *LR* given in (2.1) becomes:

(2.6)
$$R\left(x\right)=\frac{1}{2}\left(x-\mu_b\right)^T\Phi_b^{-1}\left(x-\mu_b\right)-\frac{1}{2}\left(x-\mu_t\right)^T\Phi_t^{-1}\left(x-\mu_t\right)$$

This expression in fact compares the *Mahalanobis distances* of the observed spectrum from the centers of the two classes.

If the target and background classes have the same covariance matrix, that is, $\Phi_t = \Phi_b \equiv \Phi$, the LR detector becomes:

(2.7)
$$R(x) = (\mu_t - \mu_b)^T \Phi^{-1} x = c^T x$$
$$c \equiv \Phi^{-1}(\mu_t - \mu_b)$$

This detector is known as the *Fisher's linear discriminant* and is very useful in pattern recognition applications. In the communications and signal processing areas, as well as hyperspectral imagery processing, the term Matched Filter (*MF*) is used as reference to any filter of the form:

(2.8)
$$R(x) = c_{MF}^T x$$
$$c_{MF} = k\Phi^{-1}(\mu_t - \mu_b)$$

where *k* is a normalization constant.

In practical applications, the mean vectors and the covariance matrix, required for the *MF* calculation, are unavailable and have to be estimated from the available data. Under the assumption of low probability targets, given *N* vectors *x(n)*, *n*=1,2, …, *N*, the maximum likelihood estimates for the mean and covariance matrix of the background are given in the following:

(2.9)
$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x(n) \approx \hat{\mu}_b$$
$$\hat{\Phi} = \frac{1}{N}\sum_{n=1}^{N}\left[x(n) - \hat{\mu}\right]\left[x(n) - \hat{\mu}\right]^T \approx \hat{\Phi}_b$$

As for the target, usually there is not enough training data to determine its mean and covariance. Thus the target spectral signature *s* is often taken from a spectral library, or it is chosen to be the mean of a small number of known target pixels observed under the same conditions.

The resulting adaptive matched filter (*AMF*) is given by:

(2.10)
$$R(x) = \frac{s^T \Phi^{-1} x}{s^T \Phi^{-1} s}$$

where the data cube mean is normally removed from the target and test pixel spectra.

2.2.4 Anomaly detection algorithms

Anomaly detection is based on seeking out the pixels exhibiting strong spectral differences from the surrounding background [7] .

There are situations, besides the trivial case of unknown target signature, in which the anomaly detection is advantageous to matched detection methods. Detection algorithms that presume a target signature are subject to signal mismatch losses because of the complications of converting sensor data into material spectra – in order to apply the known-signal detection algorithm, the reflectance or emissive spectra of the objects of interest must be converted into radiance at the sensor, or the radiance data collected by the sensor must be converted into reflectance or emission, this process is known as atmospheric calibration and was described in section 2.1.2.1. In this process, errors in the estimate of reflective spectra may occur, which will lead to signal processing mismatch losses. Target matching approaches are further complicated by the large number of possible objects of interest and the uncertainty as to the reflectance or emission spectra of these objects. For example, the surface of the object in interest may consist of several materials, and the spectra may be effected by weathering or other unknown factors.

Thus, the multiplicity of possible spectra associated with the objects of interest and complications of atmospheric compensation have lead to the development of anomaly detectors that seek to distinguish observations of unusual materials from typical background materials without reference to target signatures or target subspaces.

Anomalies are defined with reference to a model of the background. Background models are developed adaptively using reference data from either a local neighborhood of the examined pixel or a larger section of the image. Local anomalies are defined as observations that deviate in some way from the estimated background.

2.2.4.1 The RX Algorithm

The RX algorithm is the most common representative of anomaly detectors. It is based on the Generalized Likelihood Ratio Test (*GLRT*) presented in section 2.2.2, assuming normal probability distribution of the data.

In order to calculate the *GLRT*, the *PDF* of the data needs to be estimated. Here the *PDF* is assumed to have a parametric form, thus the *PDF*'s parameter θ can be estimated from reference data. The assumption is that the reference data set consists of *N* independent identically distributed (*IID*) samples of dimension *K* (the model allows *N*=0) denoted by $\left\{ \mathrm{v_j} \in C^K \middle| 1 \le j \le N \right\}$, the *PDF* of the reference data set is denoted by $p_0\left(y,\theta_0\right)$. The test data set is denoted by $\left\{ \mathrm{x_j} \in C^K \middle| 1 \le j \le M \right\}$. The data set is to be classified as arising from either *PDF* $p_1\left(y,\theta_1\right)\left(H_1\right)$ or $p_0\left(y,\theta_0\right)\left(H_0\right)$. The *GLRT* is given by:

(2.11)
$$G(x)=\frac{\max\limits_{\theta_1}\left(p_1\left(\left\{x_l\right\},\theta_1\right)p_0\left(\left\{v_j \middle| 1\le j\le N\right\},\theta_1\right)\right)}{\max\limits_{\theta_0}\left(p_0\left(\left\{x_l\right\},\theta_0\right)p_0\left(\left\{v_j \middle| 1\le j\le N\right\},\theta_0\right)\right)} \underset{H_0}{\overset{H_1}{\gtrless}} \eta$$

The data is modeled as following:

(2.12)
$$\begin{aligned} H_0 &: x \sim N\left(\mu,\Phi\right) \\ H_1 &: x \sim N\left(s,\Phi\right) \end{aligned}$$

where *N* (μ,Φ) denotes normal distribution with mean μ and covariance Φ. *s* and Φ are unknown parameters.

The *GLRT* for this model is given by:

(2.13)
$$RX(x)=(x-\mu)^T\left(\frac{N}{N+1}\hat{\Phi}+\frac{1}{N+1}(x-\mu)(x-\mu)^T\right)^{-1}(x-\mu) \underset{H_0}{\overset{H_1}{\gtrless}} \eta$$
$$\hat{\Phi}=\frac{1}{N}\sum_{j=1}^{N}\left(v_j-\mu\right)\left(v_j-\mu\right)^T$$

where $\hat{\Phi}$ is the covariance matrix estimated from the reference data.

As $N \to \infty$, *RX* converges to:

(2.14)
$$RX(x)=(x-\mu)^T\,\hat{\Phi}^{-1}(x-\mu) \underset{H_0}{\overset{H_1}{\gtrless}} \eta$$

RX is a single pixel form of the *Reed-Xiaoli* algorithm that is often approximated by equation (2.14).

This expression for the *RX* algorithm is closely related to the expression given in equation (2.10) for the Matched Filter detector. The denominator in (2.10) can be ignored, since it is a normalizing constant term. The *RX* now resembles the Matched Filter, but instead of matching to the target signature (which is unknown for the *RX*), the match is performed on the ($x - \mu$) term. The intuitive explanation for this is that the best estimation of the actual pixel's content is its value subtracted by the background. If a target is present then the residue will be stronger and the overall response of the filter will be higher. If the target is absent, the residue after the background subtraction should be close to zero (depending on the noise level) and the filter output will be low indicating less likeliness of target presence.

It is important to note, that the distribution of the test statistic for H_0 is independent of the unknown parameters, and thus the test statistic has the constant false alarm rate (*CFAR*) property.

2.2.5 Evaluation of detection algorithms

As stated in the previous subsection the detection is based on determining appropriate threshold against which the *LR* is compared [5] . A practical question of great importance to a detection algorithm user is where to set the threshold to keep the number of detection errors (target misses and false alarms) small. There is always a compromise between choosing a low threshold to increase the probability of detection P_D and a high threshold to keep the probability of false alarm P_{FA} low. For any given detector, the trade-off between P_D and P_{FA} is described by the receiver operating characteristic (*ROC*) curves, which plot $P_D(\eta)$ versus $P_{FA}(\eta)$as a function of threshold $-\infty<\eta<\infty$. The calculation of the *ROC* curves or the threshold requires specifying the distribution of the observed spectra x under each of the two hypotheses specified for the *LR*. In most practical situations, these conditional probability densities depend on some unknown target and background parameters (composite hypotheses). Therefore, the *ROC* curves of any detector depend on the unknown parameters. In this case, it is almost impossible to find a detector whose *ROC* curves remain an upper bound for the whole range of the unknown parameters (uniformly most powerful (*UMP*) detector).

Practical target detection systems should function automatically, that is, without operator intervention. This requires an automatic strategy to set a "proper" detection threshold. In certain applications, for example military target detection, a high false alarm rate is very undesirable. Therefore it is critical to keep the false alarm rate constant at the maximal affordable level by using a constant false alarm rate (*CFAR*) processor. The task of a *CFAR* algorithm is to provide detection thresholds that are relatively immune to noise and background variation and allow target detection with a constant false alarm rate.

In addition, the performance evaluation of detection algorithms in practice is challenging due to the limitations imposed by the limited amount of target data. As a result the establishment of accurate ROC curves is quite difficult. A practical mean for comparing various detection algorithms is their ability to operate in *CFAR* mode and the enhancement of the separation between targets and background they provide. The *CFAR* property depends on the capability to accurately model the detection statistics of the background pixels for a given algorithm.

3 Point target detection and tracking in consecutive frame in IR imagery

3.1 Overview

3.1.1 Problem statement

In order to develop advanced staring IR technologies in surveillance applications, there is a need for algorithms to detect weak point target moving at pixel/sub-pixel per frame velocities. These algorithms must handle scenes containing both clutter and noise dominated backgrounds. Clutter dominated background is a background in which the largest non-target detections (false alarms) are from the structured background. In the context of the temporal processing, evolving clouds are the most severe cause of such leakage. In noise dominated background random noise provides the largest non-target detections. Scenes with only blue sky, night scenes, or scenes with temporally stationary clutter are well modeled by the white Gaussian noise in the temporal dimension and hence are noise-dominated backgrounds in the context of temporal processing.

An efficient algorithm must thus address these dual and often conflicting aspects of the problem: the need for effective clutter suppression for minimizing false alarm rates in structured backgrounds, and the need for good signal-to-noise sensitivity for maximum range of weak target detection. A possible solution to the problem is to use a *dynamic programming algorithm* (*DPA*) which gives scores to pixels according to their likelihood of being a target and accumulates scores from frame to frame for all the pixels. Using that algorithm enables the tracking of low SNR targets. It is also advantageous to use a preprocessing stage, which whitens the background, lowers the clutter and emphasized the target to achieve a lower false alarm rate.

3.1.2 Application

Point target detection is particularly useful for staring IR technologies, e.g., IR search and track (*IRST*), such as the ones used in surveillance application or missile detection and tracking. These are applications in which the sensor, in this case the camera, is fixed on the ground and pointed towards the sky. The camera is not moving; the temporal variation of the signal originates from clutter (clouds) and target movement and, of course, noise.

3.2 Algorithms for moving point target detection in IR imagery

3.2.1 The track before detect approach

Track before detect (*TBD*) is a technique for target detection in which the detection is not declared at each frame; instead a number of frames of data are processed after which the estimated track is returned when the detection is declared. Since target trajectories are not known prior to detection, *TBD* schemes typically require that multiple frame (scan-to-scan) processing be performed against multiple, simultaneous target trajectory hypotheses. Detections are declared for those hypotheses for which the integrated energy exceeds appropriate threshold levels.

3.2.2 Dynamic Programming Algorithm (*DPA*)

The *DPA* is an implementation of the *TBD* approach that aids in the tracking of a dim point target maneuvering in a noisy background, [17] ,[18] , [19] . It is based on the assumption that a real target moves in a trajectory while noise appears and disappears in a random fashion. In every frame, each pixel is considered as a potential target and is checked for its origin in the previous frame. By doing so, all possible trajectories with high probability are built up, and it is assumed that with the progression in the image sequence and the addition of data, the probability of a single trajectory will grow above the others. This trajectory will be the target's path.

The algorithm uses the following principles as guidelines:

1. Overcoming low *SNR* using prior knowledge of the way the target moves; an apparent unlikely change of target speed and location will lower the probability of the target to be part of the trajectory we would like to find.
2. Processing is done not on a single frame but rather on a given sequence so that the nth frame will contain accumulated information from the previous frames.
3. Priority is given to the detection of the target's trajectory.

The *DPA* algorithm uses a *Markov Process Model* in order to give scores to the image sequence; when the current frame is processed, all the vital information from the preceding frames lies in the accumulated score matrix (*ASM*) of the previous frame (state). At the end of the *DPA* stage, the target is acquired from the last frame, and its trajectory is found.

3.2.3 Temporal processing

Temporal processing exploits the change in a pixel's intensity as moving point target traverses it. The temporal profile of target affected pixel raises and falls as the target enters and exits the pixel, while clutter affected temporal profiles show more monotonic intensity changes over an equivalent time period, (Figure 3).

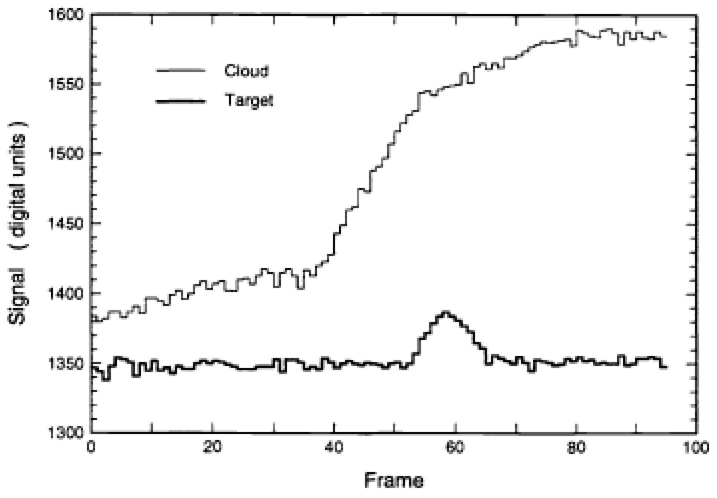


Figure 3: Example of clutter and target temporal profiles.

References [8] and [9] introduce a zero-mean damped sinusoid filter. The *Triple Temporal Filter* (*TTF*) has been designed to have a strong response to relatively narrow peaks and a weak response to monotonic changes. The filter is based on applying three sinusoid filters serially - a sequence of two zero-mean damped sinusoids followed by an exponential averaging filter along with an edge suppression factor. The implementation of the *TTF* is recursive; the output result is updated with each new frame. Figure 4 shows the impulse response of a zero-mean damped sinusoid filter.

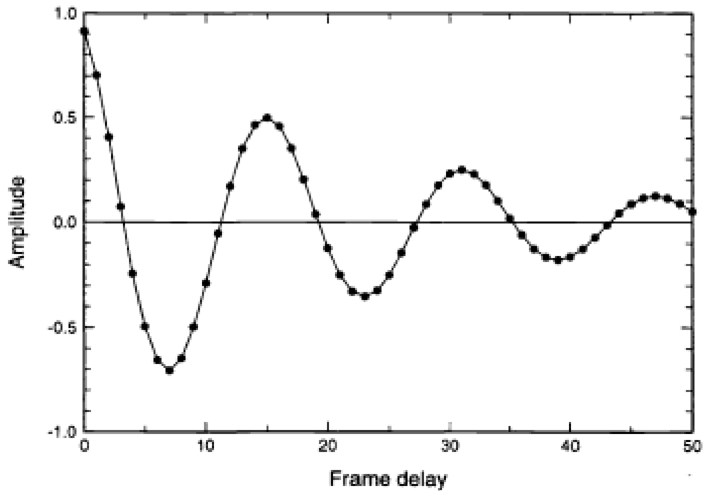


Figure 4: Impulse response of zero-mean damped sinusoid temporal filter.

Each filter is recursively implemented using the following expression:

$$Z_{k+1} = Z_k \cdot \alpha \cdot e^{i\theta} + d_{k+1} \cdot e^{i\phi} \tag{3.1}$$

where α is the damping factor, θ and ϕ are the angle and initial phase of the sinusoid respectively and d_{k+1} is the current sample examined.

The *TTF* is specified by six parameters: the periods and the damping constants of the first two sinusoids, the damping constant of the averaging filter and an edge parameter used in the edge suppression spatial adjunct. The parameters can be derived by applying the simplex algorithm to sets of real data, reference [10] describes this procedure.

The filter parameters define its clutter suppression ability on one hand, and SNR sensitivity on the other. Thus, a certain set of parameters results in a clutter suppressive filter (Cloud Filter - *CF*), while a different set of parameters results in a filter which performs better in noise-dominated scenes (Noise Filter - *NF*).

Reference [9] presents a fusion filter (*FF*) which incorporates the output of two double temporal filters, one designed for clutter suppression and the other for low SNR operation, thus providing a filter capable of dealing with both clutter and noise dominated scenes.
The *FF* extends the range of velocities over which weak targets can be extracted from temporal noise as well as above the clutter leakage. The form of the *FF* is two double temporal filters running in parallel. One channel of the *FF* is derived from the *CF* (unmodified) and the second channel is a modified version of the *NF*. The modified *NF* enables the detection of slow weak targets in clear sky and still control clutter leakage. The final *FF* algorithm result is taken as a maximum of the two channels for each pixel followed by an eight frame average. The *FF* is illustrated schematically below.

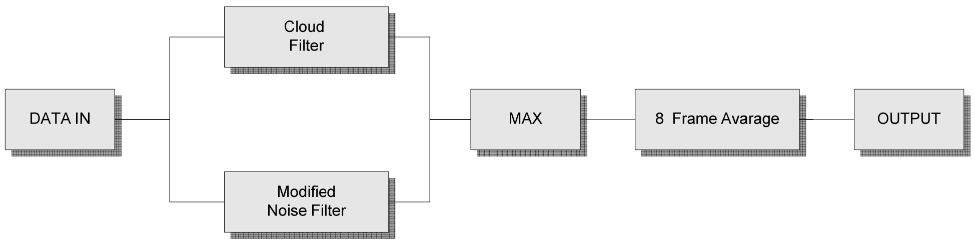


Figure 5: <u>Fusion Filter configuration</u>.

3.2.3.1 Variance filter

Another type of temporal processing filter is the *variance filter* (*VF*) presented in [11] . The noise filter based on damped sinusoids can detect targets roughly as weak as *S/N* = 3, but only over a narrow range of focal plane velocities: 0.05 to 0.1 [*ppf*]. Sensitivity falls off rapidly for velocities slower or faster than this. The variance filter ,*VF*, can detect targets this weak, or weaker in some cases, over the extended range: 0.02 to 0.25 [*ppf*]. The algorithm involves simple recursive steps easily implemented in hardware which provides a recursively updated estimate of the following quantity for each pixel: *(temporal variance) - (long time baseline variance)*. The quantity is zeroed unless positive. The variance filter estimates the change in temporal variance by calculating the short-term variance and subtracting the estimated long-term variance.

The baseline algorithm consists of four steps per pixel done in sequence and recursively updated with each new temporal frame of data. The four steps correspond to the four equations below; the steps involve exponentially weighted averages of 16 or 32 frames (nominally).

- Step 1 updates the 32 frame average M_{32} of the input data for each pixel.
- Step 2 updates the 16 frame average estimate of the variance defined as $(input-M_{32})^2$.
- Step 3 generates a change in variance (zeroed unless positive) by subtracting from Step 2, a baseline estimate of the pixel variance given by the 32 frame average of Step 2.
- Step 4 updates the 16 frame average of Step 3 providing the displayed value of the variance filter (*VF*).
- R_1-R_4 in the next equations represents the recursive output of these steps for each pixel with each new input frame. The brackets with subscripts refer to the recursive frame average computed as explained subsequently.

$$\begin{aligned}
&1.\quad R_1 = \langle input \rangle_{32} \\
&2.\quad R_2 = \left\langle (input - R_1)^2 \right\rangle_{16} \\
&3.\quad R_3 = R_2 - \langle R_2 \rangle_{32} \quad if \geq 0, \quad else \quad 0 \\
&4.\quad R_4 = \langle R_3 \rangle_{16}
\end{aligned} \qquad (3.2)$$

The exponentially weighted average for arbitrary vector of samples *x* is given in the following:

(3.3) $$\langle x \rangle_i = \frac{1}{i}\sum_{j=0}^{i-1}\left(\frac{i-1}{i}\right)^j x[k-j]$$

where *k* is the k^{th} input sample (which is the most recent sample), *i* is the number of previous samples participating in the average calculation (including the most recent sample).

The *VF* responds to moving targets of positive or negative contrast. Typical targets of interest are positive contrast. For such cases, a preprocessing technique based on morphological image processing but implemented by median filters is applied and further improves the SNR performance of the algorithm.

3.2.3.2 Performance metrics

There is a need for defining proprietary performance metrics for point target detection in consecutive frame IR imagery based on temporal processing. This task becomes more challenging when working with real world data. Standard metrics are based on some variations of the following equation for gains in *dB* [12] :

(3.4) $$Gain = 20\log\frac{(S/C)_{out}}{(S/C)_{in}}$$

where *S* and *C* are target and clutter measures respectively, in and out denote the data at the input and at the output of the relevant algorithm respectively. Usually the values of *S* are predetermined since the target is often embedded into the sequence. The values of *C* can be taken either as the standard deviations or the variances of a single frame.

When working with real world data several issues arise: the signal strength for targets with no auxiliary ground truth is not known and must be extracted from the data; the clutter probability density function is rarely Gaussian, because of that the clutter is not accurately characterized by standard statistical measures; the measure of the initial clutter severity should relate to its temporal non-stationarity since stationary clutter can be removed by well-designed temporal processing.

Reference [12] suggests the following desired features for performance metric of temporal algorithm:

1. Have some link to previous practice (for example still employ equation (3.4)).
2. Be relevant to false alarm threats.
3. Capable of operating on real targets without auxiliary ground truth.
4. Be adequate to temporal processing.
5. Not be unduly influenced by simple processing steps like mean or local mean removal.
6. Be symmetric in its treatment of the input and filtered output.

The reference suggests two metrics based on equation (3.4). The first metric is called the Variance Metric (*VM*). The ratio $(S/C)_{out}$ is taken as the ratio between the largest target related pixel output to the largest non-target output from the filtered output frame/s. The symmetric use of the corresponding largest target and non-target input values from the unprocessed frames in the $(S/C)_{in}$ calculation would be inconsistent with the features 4 and 5 defined above and does not address the target in terms of its surroundings. The C_{in} is taken as the maximal value of the temporal standard deviation calculated for each of the non-target pixels. The approach for determining S_{in} is not as straightforward since it is difficult to determine the intensity of a real target when it moves slowly through sampled imagery. Thus the input target strength is characterized as the average of the maximal values of the target pixels after subtracting the local background mean estimated from non-target pixels.

The second metric presented in source [12] is the *Anti-median Metric* (*AM*). An anti-median filter is applied on the input and output frame/frames. The filter takes an absolute difference between the center pixel and the median of 5x5 block about and including this pixel. The ratios $(S/C)_{out}$ and $(S/C)_{in}$ are taken from the largest target value and the largest non-target value from the output and input frame respectively. If there are more than one frame, averaging over the number of frames is performed for the largest values.

None of the two metrics presented satisfies the desired demands listed above. The complicated dynamics of moving targets and evolving clutter recorded on a staring array is hardly captured by a single measure.

In order to evaluate the algorithm a new metric has been defined. Each frame in the sequence is divided into blocks, the size of *HxN* (30*x*30 were used).The algorithm is run over 9 blocks – Target block and eight adjacent blocks. The *SNR* of the target block (*TB*) and its eight adjacent non-target blocks (*NTB*) are calculated. Afterwards, an algorithm score is calculated based on the resulting *SNR*'s.

The block SNR is given as:

$$Block\_SNR(i,j)=\frac{E\left[v_{i,j}\in M\right]-E\left[v_{i,j}\notin M\right]}{\sigma_{v_{i,j}}} \qquad (3.5)$$

where $v_{i,j}$ is the set of pixels belonging to the *(i,j)*th block, *M* is a set containing the five pixels with the highest gray level in that block, and σ is the standard deviation of the block pixels. The formula performs a subtraction between the expectation value of the highest pixels (target) and the expectation value of the rest of the pixels (background), divided by the standard deviation of block pixels. Since the probability matrices introduce the influence of target pixels on adjacent pixels, these influenced pixels might accumulate higher values than uneffected pixels (background), and can be regarded as target pixels. This might lower the expectation value of the target, but will lower the standard deviation of the background, since these high pixels are higher than the statistics of the background, to a more accurate one.

The algorithm score is given as:

$$Score=\frac{Block\_SNR\left(i,j\right)_{(i,j)=TB}-E\left[\left\{Block\_SNR\left(i,j\right)\right\}_{(i,j)=NTB}\right]}{\sigma_{\left\{Block\_SNR\left(i,j\right)\right\}_{(i,j)=NTB}}} \qquad (3.6)$$

The score is calculated by subtracting between the target block *SNR* and the expectation value of the *SNR* of the non-target blocks, divided by the standard deviation of the non-target blocks *SNR*. The algorithm always identifies a target in each given block. Since that is the case, the false detection of targets should achieve less *SNR* in these *NTB*, resulting in higher score. It should be noted that the division to blocks is only for the purpose of algorithm evaluation (score), where in real scenarios, the frames will not be divided. In these cases the algorithm will produce several high pixels - true and false targets. Using the *NTB*'s *SNR* in the algorithm score simulates the non-target scenarios. In the tests, we will vary the number of highest pixels, defined as *Max_Numbers*, which enter the target metric in this way. We will check the influence of the number of high pixels regarded as ‘target pixels’ on the *Block_SNR*.

4 A system for point target tracking in consecutive frame hyperspectral imagery

4.1 General Architecture

The system performs target detection and tracking in three steps. The first step is to transform the three dimensional hypercube into a two dimensional image using one of the well known hyperspectral reduction algorithms. The second stage involves a temporal separation of the images sequence into sub- images sequence and the usage of a variance filter (temporal processing) to detect the presence of targets from the temporal profile of each pixel groups (implementation of a variance filter on the sub-temporal profiles promise a pixel velocity at new sequences), while suppressing clutter specific influences. The third step is target detection and tracking based on the *TBD* approach and implemented using *DPA* that proved to be an effective algorithm for the tracking of moving targets with low *SNR*. The general system architecture is given in Figure 6.

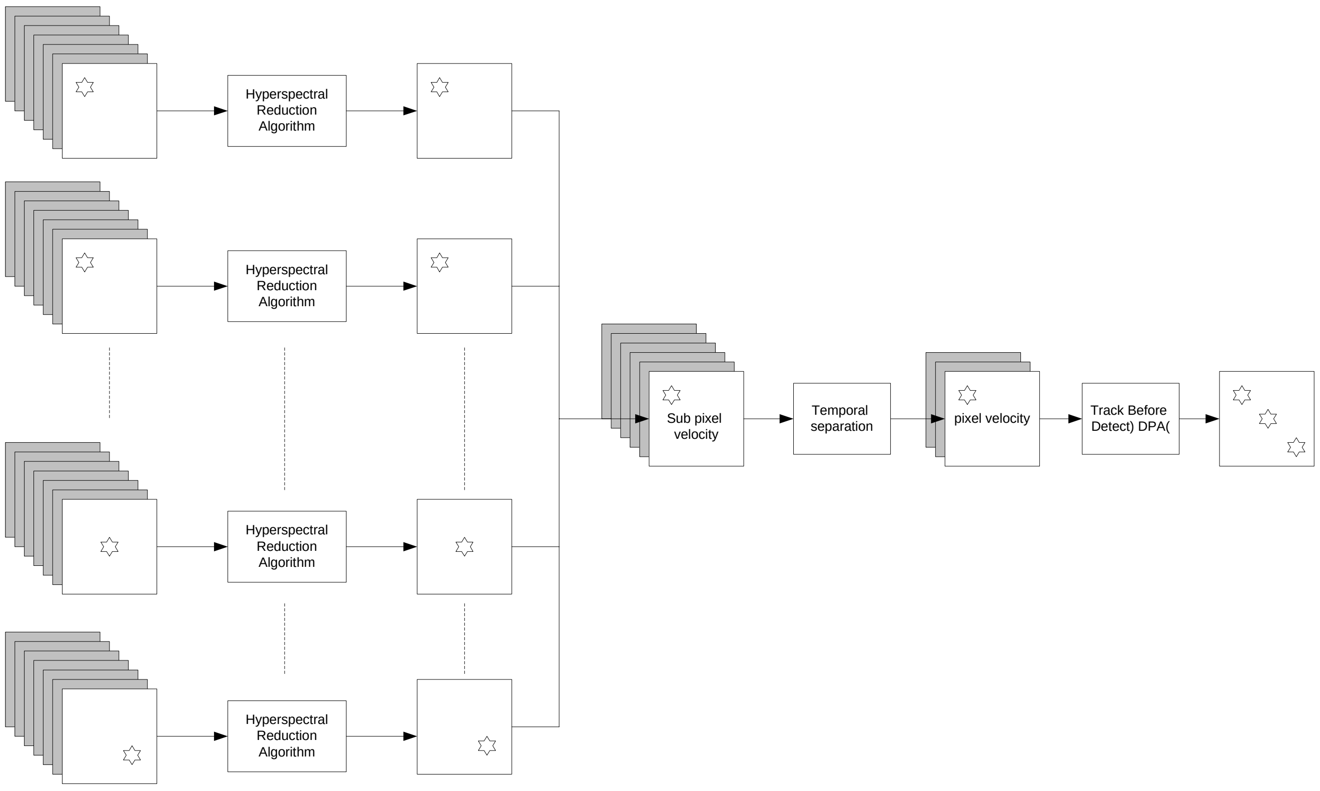

Hyperspectral Reduction Algorithm
Hyperspectral Reduction Algorithm
Hyperspectral Reduction Algorithm
Hyperspectral Reduction Algorithm
Sub pixel velocity
Temporal separation
pixel velocity
Track Before Detect) DPA(

Figure 6: General system architecture.

4.2 Hyperspectral reduction algorithm

Three different reduction tests were applied on each temporal hypercube individually. Each of these methods is characterized by a mathematical operator, which is calculated on each pixel. In every frame, a map of pixel scores is received (the result of the mathematical operator) and used to create the movie.

4.2.1 Test 1 - Spectral Average

Implementation of a simple spectral average of each pixel:

$$E(x) = \frac{1}{N}\sum_n x_n \tag{4.1}$$

where x denotes a pixel's spectrum, x_n the spectral value of the n^{th} band, and N is the number of spectral bands.

4.2.2 Test 2 - Scalar Product

Implies a simple scalar product of the pixel's spectrum (after mean background subtraction) with the known target spectral signature:

$$\text{Scalar Product} = t^T \cdot (x - m) \tag{4.2}$$

4.2.3 Test 3 - Matched Filter

The model assumed is the *linear mixture model* (*LMM*) and a known target signature. The Matched Filter detector given in equation (2.10) is rewritten as follows:

$$MF = t^T \Phi^{-1}(x - m) \tag{4.3}$$

where x is the pixel being examined, t is the known target signature, and m and Φ are the background and covariance matrix estimations, respectively. Compared to (2.10), the expression given in (4.3) ignores the constant denominator and explicitly states the background subtraction procedure prior to applying the filter.

The background estimation is performed on the closest neighbors, which definitely do not contain target; for example, if the target is known to be at most two pixels wide, the background is estimated from the 16 surrounding neighbors as illustrated in Figure 7.

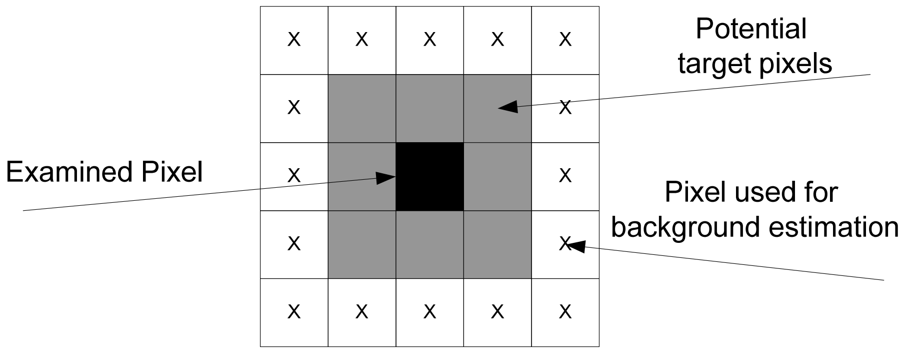


Figure 7: Pixels used for background estimation for 2x2 pixels target.

Each of the tests mentioned above were run with a different target factor (intensity). This intensity can be controlled manually by the hyperspectral data creation algorithm, and it is an external parameter to those test run mentioned above. The higher the target factor value, the stronger the intensity of the implanted target, applying less difficulty to the detection and tracking algorithm. The implantation model of the target is directly proportional to the target factor (intensity of implantation).

4.3 The temporal processing algorithm

Overall, the input to the first stage is a hyperspectral cube; the output of the first stage is a two-dimensional image obtained from the hypercube. A buffering of several such images is needed in order to obtain a sequence long enough to perform temporal processing. The input for the temporal processing algorithm is a temporal profile of a pixel. Figure 8 defines our terminology at this stage.

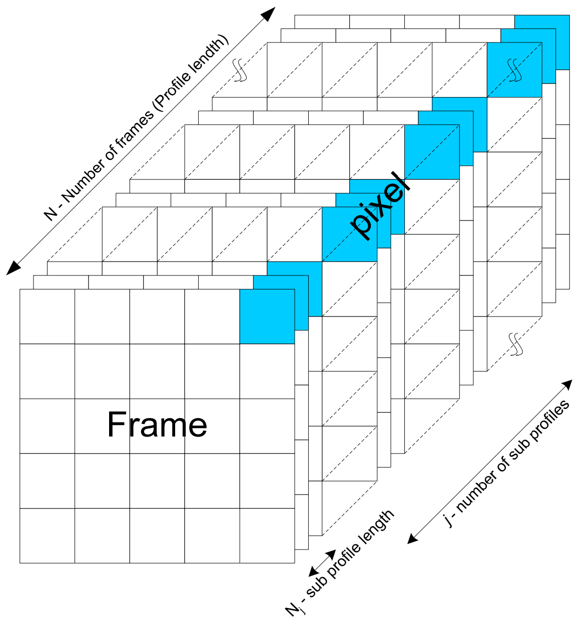

N - Number of frames (Profile length)
pixel
Frame
j - number of sub profiles
N_j - sub profile length

Figure 8: Definitions at sequences after hyperspectral reduction algorithm.

The temporal processing algorithm start with a temporal separation of each temporal profile to sub-temporal profiles to achieve target moving at a pixel velocity from images containing targets moving at a sub-pixel velocity. For example, the profile at Figure 9 (above) is an input to the temporal separation as shown at Figure 9 (below). The number of sub profiles is defined by:

(4.4) $$j = \frac{N - G_0}{G - G_0}$$

where *N* is the number of frames at profile, G_0 is the overlap between each of sub-profiles and *G* is the sub-profile length.

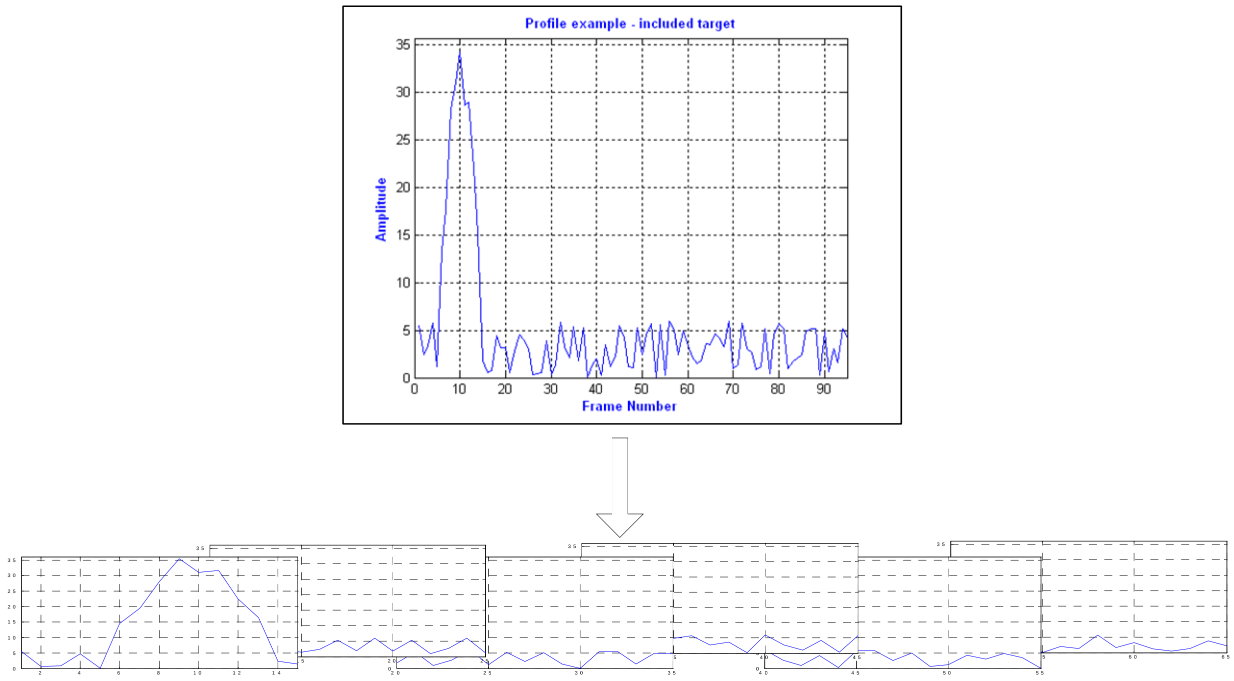


Figure 9: <u>Profile with target before (above) and after (below) temporal separation</u>[1].

Afterwards, the temporal processing algorithm is applied. The temporal processing algorithm is based on a comparison of an estimation of the overall *DC* estimation of the sub-profile (*DC* estimation calculated for the overall profile in order to achieve best or more accurate *DC* estimation, the *DC* estimation of the sub-profile chosen by the relation location at the overall profile) and the single highest fluctuation which occurs within the sub-profile. In order to estimate the overall temporal DC estimation (baseline background estimation) was performed using a long term window of samples. The background estimation is performed by calculating a linear fit by means of least squares

[1] The specific profile was chosen with target – peak at frame 10; the profile length is 95 frames; the sub-profile length is 10 frames and overlap is equal to 5 frames.

estimation (*LSE*) [13] . The fluctuation or short-term variance estimation is performed on a shorter term window of samples, after removing the estimated baseline background. The algorithm is presented in the following two steps, although in practice the processing can be performed in real time using a finite size buffer:

4.3.1.1 Background estimation using a linear fit model

The background can be referred as the *DC* level of the temporal profile: the *DC* level is constant for noise dominated temporal profiles but time varying when a clutter is present. The *DC* is estimated in a piecewise fashion by using a long term sliding window and performing estimation on each set of samples separately. The number of samples of each long term window is denoted by *M*. The following linear model is used for estimating the *DC*; for the sake of simplicity the description of the estimation is relevant for a single window:

(4.5) $$y = ax + b + n;\; x = \begin{bmatrix} 1 & 2 & \dots & M \end{bmatrix}^T$$

where *n* is the noise, *a* and *b* are the coefficients which must be estimated, *y* is the *DC* signal. The goal of this step is to estimate the long-term *DC* baseline using a least squares fit to the linear model represented by coefficients vector $\begin{bmatrix} \hat{a} & \hat{b} \end{bmatrix}^T$.

(4.5) can be rewritten as follows:

(4.6) $$y = X\beta + n$$

where $\beta = \begin{bmatrix} b \\ a \end{bmatrix}$ and $X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & M \end{bmatrix}$

Using least squares estimation the following equation for $\hat{\beta}$ is obtained:

(4.7) $$\beta = \left(X^T X\right)^{-1} X^T y;\; \hat{\beta} = \begin{bmatrix} \hat{b} \\ \hat{a} \end{bmatrix}$$

The estimated *DC* of a single window becomes:

(4.8) $$\hat{y} = X\hat{\beta}$$

The estimated *DC* of the complete signal is obtained after performing the above calculations for each window separately. Figure 10 shows two synthetic temporal profiles (one with the target implanted and a second with identical noise but without a target) and their estimated *DC* signals.

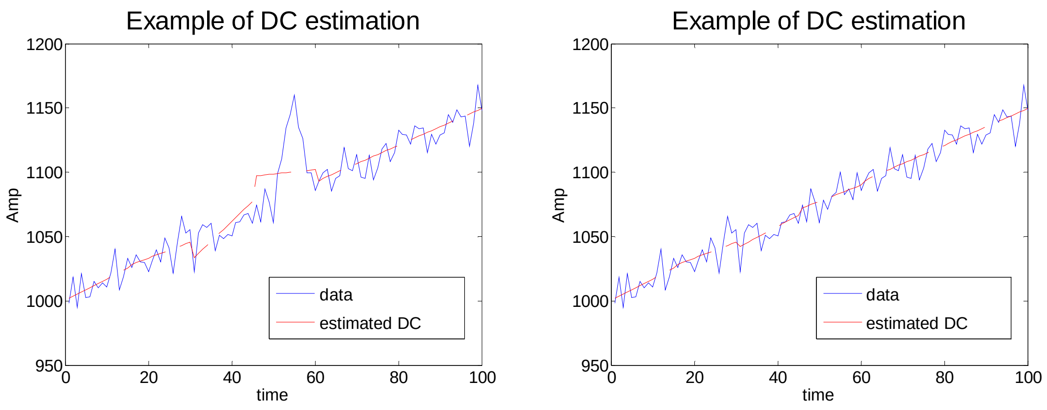


Figure 10: DC estimation on a signal with target and the same signal without a target.

The estimated *DC* of the sub-profile is chosen by the relation location at the complete pixel, Figure 11 are show the *DC* estimation on a signal with a target and same signal without the target at the sub-profiles separations point of view.

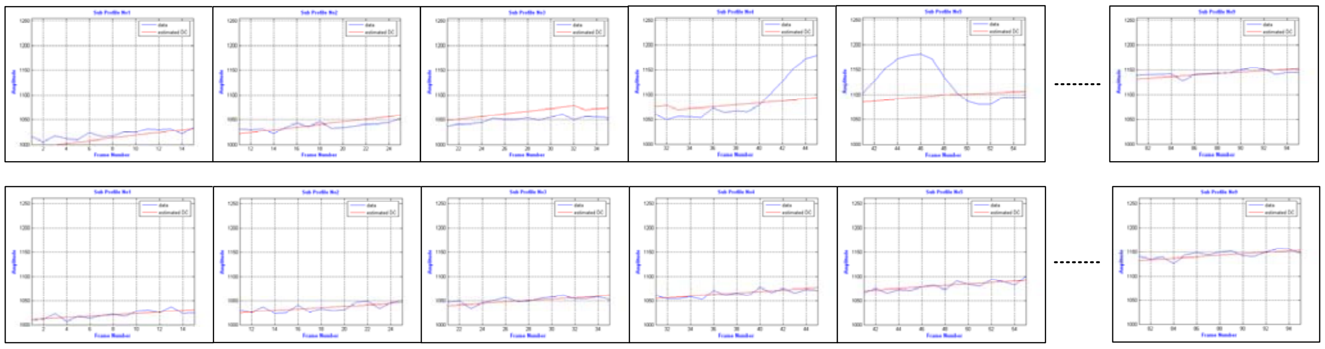


Figure 11: Sub profile DC estimation on a signal with target and the same signal without a target.

4.3.1.2 Short term variance estimation

Step 2 calculates the short-term variance after subtracting the estimated long term *DC* at each sub-profile. The complete *DC* signal obtained in the previous step is denoted by DC_j, where *j* denotes the index of sub-profile, the number of sub-profiles is defined at (4.4). The DC_j is subtracted from the temporal sub-profile P_j:

(4.9) $$\hat{P}_j = P_j - DC_j$$

The variance estimation is calculated by using a sliding short term window and performing estimation on each set of samples separately. *L* denotes the number of samples of each short term window. The short term variance of each window is estimated as follows:

(4.10)
$$\sigma = \frac{1}{L}\sum_{i=1}^{L}\hat{P}_j(i)^2$$

For window size of *L* samples, overlap of L_o samples and sub-temporal profile of *G* samples, the number of windows *W* is given by:

(4.11)
$$W = \left\lfloor \frac{G-L_0}{L-L_0} \right\rfloor + 1$$

Finally, the maximum variance value of a given temporal profile is given by:

(4.12)
$$\sigma_{\max}^2 = \max_{1\le i\le W}\left\{\sigma_i^2\right\}$$

where σ_i^2 is the estimated variance of the i^{th} window.

An example of the variance response to the presence of a target is shown in Figure 12. The assumption is that the presence of the target will lead to a short-term variance increase. The *DC* subtraction has a clutter suppression effect since the long-term *DC* tracks the clutter influence on the temporal profile. The graphs of Sub Profile 4 and 5 were scaled to the range of the pixel's values in the profile. The scaling was done in order to show the range of the variance estimation values.

Finally, a likelihood ratio based metric is used to evaluate the final score of each sub-temporal profile. The likelihood ratio in this case is given by:

(4.13)
$$\begin{aligned} &H_0: \quad \hat{P} = n \\ &H_1: \quad \hat{P} = t + n \\ &LRT = \frac{\hat{\sigma}_1^2}{\hat{\sigma}_0^2} \end{aligned}$$

where $\hat{P}$ is the zero-mean temporal profile, n is noise and t is the target signal. $\hat{\sigma}_1^2$ is the estimated variance when assuming a target is present; $\hat{\sigma}_0^2$ is the variance estimated assuming the absence of target.

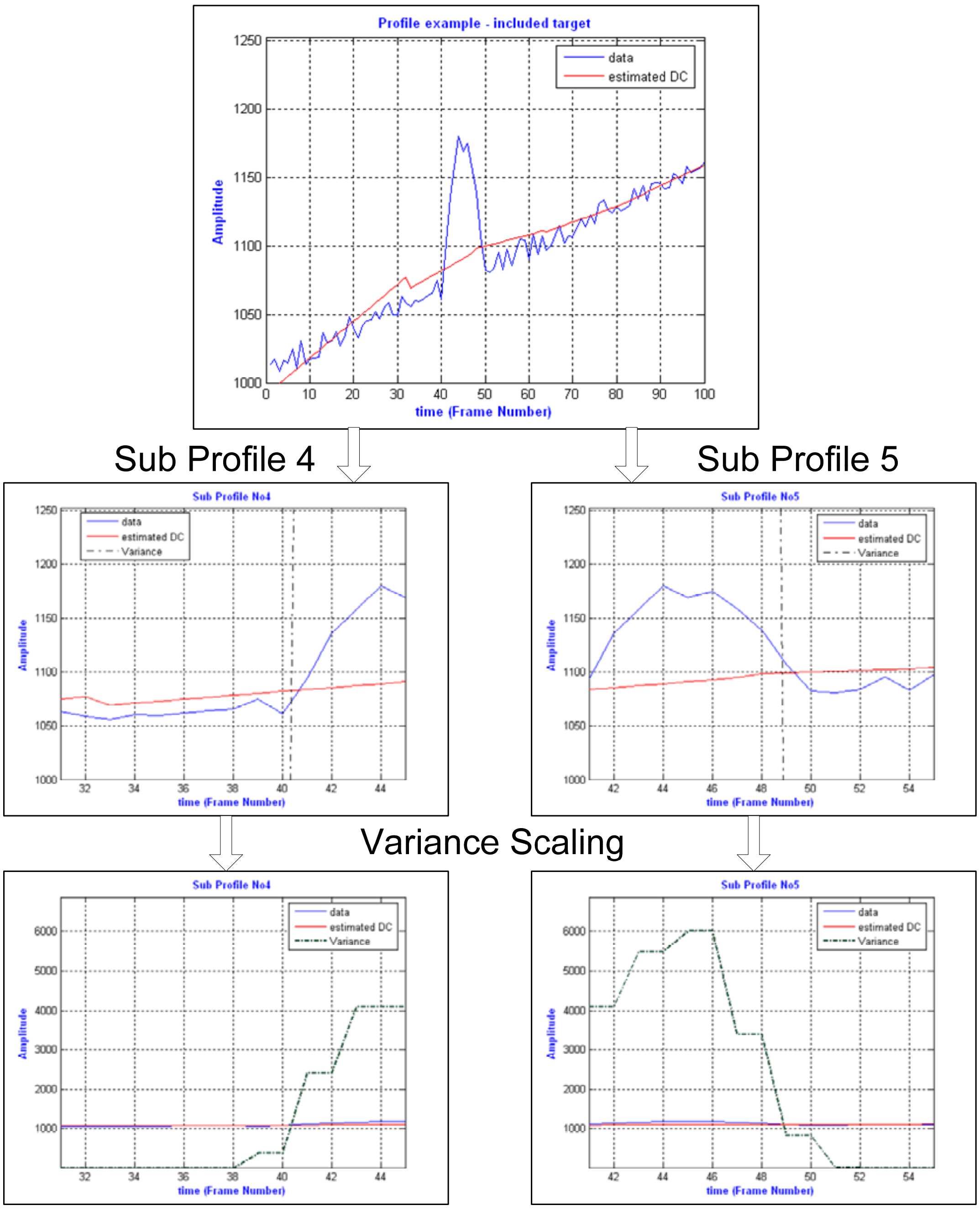


Figure 12: Example of variance estimation on a synthetic signal.

In our model these variances are estimated as follows:

(4.14)
$$\hat{\sigma}_1^2 = \hat{\sigma}_{\max}^2$$
$$\hat{\sigma}_0^2 = \frac{1}{K}\sum_{i=1}^{K}\tilde{\sigma}_i^2$$

where $\tilde{\sigma}_i^2$ for $1 \le i \le K$ denotes the K minimal variance values of each temporal profile. The value of K is chosen to be smaller than W, so as not to include values which might be caused by the presence of the target.

In this case, the final score of each sub-profile is given by:

(4.15)
$$Score_j = \frac{\hat{\sigma}_{\max}^2}{\frac{1}{K}\sum_{i=1}^{K}\hat{\sigma}_i^2} \qquad K = \text{Defined by 5 highest or lowest values}$$

The algorithm performance depends on a wise choice of parameters: the short-term, long-term window size and sub-profile length. The long-term window size serves the baseline *DC* estimation. Since the pixel might be affected by clutter, the baseline *DC* is not constant but rather changing. It is assumed that the entrance of clutter will cause a monotonic rise or fall pattern in the values of the pixel's temporal profile. Thus, the long-term window should be long enough to perform an accurate estimation, on one hand, and short enough to closely track clutter influence, on the other. In any case, the long-term window should be at least longer than the target base width to avoid suppressing it [14] , long-term window is calculated to the overall profile to achieve best estimation of a *DC* level, the long-term window of the sub-profile chosen by the relation location at the overall profile). The short-term window is used for variance estimation. It should be matched (or reduced) to the target width (which depends on the target velocity). If the short-term window is significantly longer than the target width, the variance change caused by the target will be diminished. Sub-profile length services to achieve a pixel target velocity. It should be matched (higher) to the target width. The importance of these three window sizes and the overall window set parameters will be further discussed. It is important to emphasis that the temporal algorithm presented here doesn't assume a particular target shape and width. It does assume a maximum spatial size of the target (affecting the target temporal profile) and a positive adding of the target to the background.

4.4 Performance metric

As stated in section 3.2.3.2, defining a performance metric for the temporal processing algorithm for both synthetic and real data is a challenging task. In this work, a spatial statistic metric is defined for evaluating performance on the real data sequences. The metric divides the output image into spatially equal blocks and grades each block with a statistically based score. This process will be described in the following sections.

4.5 Dynamic Programming Algorithm

The algorithm is implemented using the following assumptions:

1. The target is the size of a pixel or less.
2. Only one target exists.
3. The target moves in any possible direction.
4. Target velocity is within 0-2 [*pixel/frame*].
5. Images are too noisy for using a detect threshold on a single frame.
6. Jitter is up to 0.5 [*pixel/frame*] in horizontal and vertical directions only, uniformly distributed.

Since the target velocity is within the range of 0-2 [*ppf*] and a possible jitter of 0.5 [*ppf*], the pixel can move up to 2.5 [*ppf*] in the horizontal and vertical directions, hence a valid area from which a pixel might origin from in the previous frame is a 7x7 matrix. Such a search area can be resized according to a different velocity range and jitter, in the fashion described above. This search area will define the probability matrices which contain the probabilities of pixels in the previous frames being the origin of the pixel in the current frame. To take into account the unreasonable change of direction, *penalty matrices* have been introduced which are used to build the *probability matrices* for the different possible directions of movement. These matrices give high probabilities to pixels in the estimated direction and decreasing probabilities (punishment) as the direction vary from the estimated direction.

4.5.1 Accumulated Score Matrix

The *Accumulated Score Matrix* (*ASM*) is not based on a single frame, but rather collects information for each pixel along the sequence. The *ASM* contains the following information:

- The pixel's accumulated score.
- Direction (angle) from which the suspected pixel has arrived.
- Summation of the multiplications between the accumulated score from the previous frame with the punishment matrix in the estimated direction (will be elaborate later on).
- The maximum value of the multiplications between the accumulated score from the previous frame with the punishment matrix in the estimated direction (will be elaborated on later).
- The index of the pixel in the previous frame from which the pixel originated, maintaining the tracks with the highest probabilities.

The *ASM* is of size [*N,M,frames,5*] , where *NxM* is the frame size, frames is the number of frames, and 5 is the number of variables we save – *score, direction, sum, max* and *index*. The *sum* and *max* can be discarded and are saved only for purpose of later investigation of the algorithm performance.

The accumulated score matrix consists of two parts – the current frame score and the accumulated score, and is defined as:

(4.16)
$$Score_n(x,y) = a\cdot Oscol_n(x,y) + b\cdot\left[\begin{matrix} g\cdot\left\{\sum\limits_{i=-3}^{3}\sum\limits_{j=-3}^{3} w(direction)_{i,j}\cdot Score_{n-1}(i+x,j+y)\right\} \\ +(1-g)\cdot\left\{\max\left(w(direction)_{i,j}\cdot Score_{n-1}(i+x,j+y)\right)_{i=-3\ldots3,\, j=-3\ldots3}\right\}\end{matrix}\right]$$

where:

$Score_n$ (x,y) - the accumulated score for trajectory that ends at the current pixel,

$Anti\text{-}mean_n$ (x,y) - the score given by the preprocessing stage (anti-mean) to the current pixel at the current frame,

a - the anti-mean co-efficient determining the amount of influence on the total score,

$w_{i,j}$ - an element of the probability matrix, determining the probability of the *(i,j)* pixel to be the origin pixel from the previous frame, based on target speed, jitter and approximated direction of movement,

b - the memory persistence co-efficient, determining the influence of the accumulated score, $\sum w\cdot Score_{n-1}$ - the summation value of the multiplications between the accumulated scores in the 7x7 matrix and the probabilities,

g - the sum co-efficient determining the amount of influence of the summation on the accumulated score,

$max(w\cdot Score_{n-1})$ - the maximum value of the multiplications between the accumulated scores in the 7x7 matrix and the probabilities,

(1 – g) - the sum co-efficient determining the amount of influence of the maximization on the accumulated score.

The new accumulated score for each pixel is given as a combination of the score from the whitening stage of the current frame and a function of the accumulated score from the previous frame. To calculate the function of the accumulated score, a 7x7 temporary matrix is created by multiplying the probability matrix in the assumed direction, *W(direction)*, with the elements of the *ASM* of the previous frame. The function is composed of two parts: the sum of all the 49 values of the calculated temporary matrix, and the maximum value of the calculated temporary matrix. Several parameters have been introduced into the formula to allow for changes in the weight of each part on the calculated accumulated score, enabling flexibility in the tuning of the system to the optimal condition, if one exists. A summary of the parameters will be given later on in Section 4.5.4.

4.5.2 The probability matrix W

This matrix contains the probabilities of the pixel under investigation (suspected to be a target), to originate from a certain location in the previous frame. There are nine different probability matrices for each of the possible directions – 0° ÷ 315° at jumps of 45° (directions 1-8), and standing in place (direction 9). The probability matrix is given as:

$$W\left(direction\right) = p \cdot W_{basic} + \left(1 - p\right) \cdot \frac{W_{punishment}\left(direction\right) \cdot W_{frame}}{\sum W_{punishment}\left(direction\right) \cdot W_{frame}} \tag{4.17}$$

where

W (direction) - the probability matrix calculated for the specific direction,

W_{basic} - the basic probability matrix that includes the effects of speed and jitter,

p - the W_{basic} co-efficient determining the influence of W_{basic} on *W(direction)*,

$W_{punishment}$ *(direction)* - the punishment matrix of a given direction,

W_{frame} - the matrix used to fit $W_{punishment}$ to the set of velocities,

(1-p) - the $W_{punishment}$ *(direction)* co-efficient determining the influence of $W_{punishment}$ *(direction)* on *W(direction)*.

Figure 13 below shows the 7x7 punishment matrix $W_{punishment}$ *(→)* (0°). The estimated region of arrival gets the highest values, 12+24 in this case. The values decrease until reaching the cells opposite to the estimated direction. As can be seen, the punishment on change of direction can be controlled by the parameter y that changes the ratio of punishment. Setting y=24 achieves the highest ratio, whereas y=0 achieves a very low ratio. By using negative values for y the ratio can be further reduced. There is a tradeoff between the need for punishment on change of direction and the need for adaptation to maneuvering targets.

(9+24)-y	(8+24)-y	(7+24)-y	(6+24)-y	(5+24)-y	(4+24)-y	(3+24)-y
(10+24)-y	(9+24)-y	(7.5+24)-y	(6+24)-y	(4.5+24)-y	(3+24)-y	(2+24)-y
(11+24)-y	(10.5+24)-y	(9+24)-y	(6+24)-y	(3+24)-y	(1.5+24)-y	(1+24)-y
(12+24)-y	(12+24)-y	(12+24)-y	(0+24)-y	(0+24)-y	(0+24)-y	(0+24)-y
(11+24)-y	(10.5+24)-y	(9+24)-y	(6+24)-y	(3+24)-y	(1.5+24)-y	(1+24)-y
(10+24)-y	(9+24)-y	(7.5+24)-y	(6+24)-y	(4.5+24)-y	(3+24)-y	(2+24)-y
(9+24)-y	(8+24)-y	(7+24)-y	(6+24)-y	(5+24)-y	(4+24)-y	(3+24)-y

Figure 13: 7x7 punishment matrix in the estimated direction →.

The *W(direction)* matrix is a superposition of the basic probability matrix, W_{basic}, encompassing the target and the allowed jitter, and the punishment matrix, $W_{punishment}$ *(direction)*, in the assumed direction with normalized elements. Figure 14 below shows the basic probability matrix, the eight punishment matrices for the different directions, and the resulting nine probability matrices. The basic probability matrix encompasses the target speed and the allowed jitter, and represents standing targets. The punishment matrices give high probability to the group of pixels in the assumed direction, and 'punish' pixels as they differ from that direction. In the figure below, the target is assumed to be at a velocity range of [0..1], thus only a 5x5 matrix is needed (including the assumed jitter) and the border pixels are zeroes accordingly using W_{frame}. The p parameter was set to 0.5, giving equal weight to W_{basic} and $W_{punishment}$*(direction)*, and the punishment parameter *y* was set to 24.

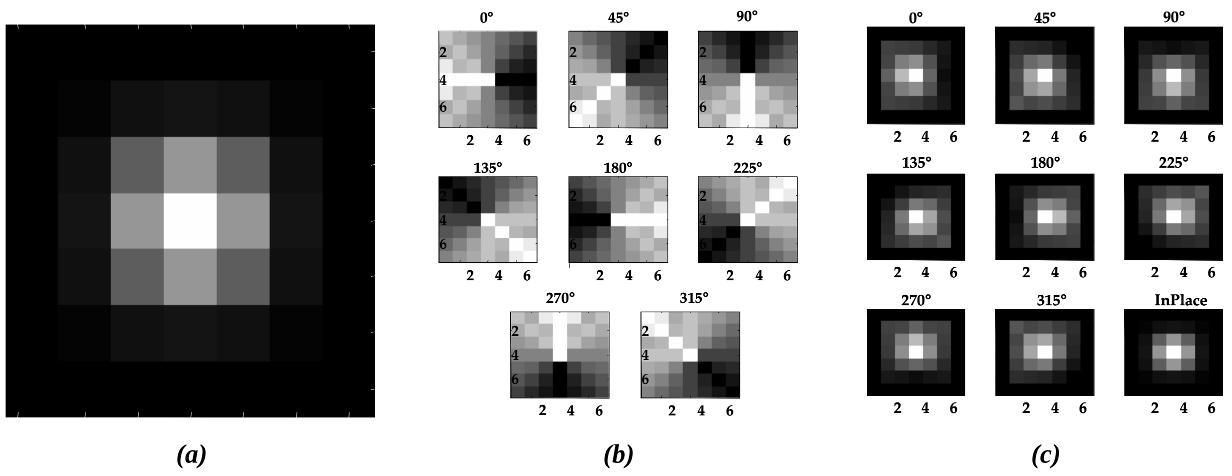


Figure 14: (a) the 7x7 W_{basic} matrix, after using the frame matrix (b) the eight 7x7 $W_{punishment}$ matrices for the different directions (c) the nine *W(direction)* matrices for *p* set to 0.5, and velocity at range [0..1].

4.5.3 Direction Calculation

To use *W(direction)*, the direction from which a pixel originated has to be found. The calculation of the direction starts by creating a temporary 7x7 matrix in which the adjacent pixels scores (including the pixel under investigation) will be saved. Afterwards, the direction of each pixel from the previous frame is found (the direction is retrieved from the *ASM*). Each pixel is given a parameter *Var* that gets a value according to the pixel's direction in the previous frame and its direction relative to the pixel under investigation (central pixel). The parameter gives a maximal value for identical directions and minimal value for opposite directions. The final score of each pixel in the temporary matrix is defined as:

$$Source\_Pixel\_Score(x,y) = Score_{n-1}(x,y) \cdot w_{x,y}(direction) \cdot Var \tag{4.18}$$

where $Score_{n-1}(x+i,y+j)$ is the pixel's accumulated score from the previous frame, *w(i,j)* is the value of *W(direction)* in the *(i,j)* coordinate and in a direction relative to the pixel under investigation, and *Var* is the direction difference parameter that was introduced above. The pixel with the highest score is chosen as the 'source pixel'. Since each group of pixels in the matrix belongs to a specific direction, the direction is now known, and is saved in the *ASM*.

To summarize, to calculate the pixel’s direction:

1. Create a 7x7 temporary matrix.
2. Find the direction of the pixels in the previous frame.
3. Calculate *Var* according to the direction consistency.
4. Calculate the temporary matrix score, *Source_Pixel_Score(x+i,y+j).*
5. Find the pixel with the highest score, and deduce the direction.

4.5.4 Parameter Summary

In the previous sections, several parameters that control the algorithm functionality were introduced. These parameters are of high significance since they allow checking the influence of changes in the weights each part is given on the determination of the *ASM*, allowing the algorithm to work better, and eventually track and detect targets in an optimal fashion. A summary of the parameters used in the algorithm (*a,b,p,g,y*):

a - the whitening co-efficient determining the amount of influence on the total score,

b - the memory persistence co-efficient, determining the influence of the accumulated score,

p (and *(1-p)*) - the W_{basic} ($W_{punishment}(direction)$) co-efficient determining the influence of the 7x7 Wbasic ($W_{punishment}(direction)$) matrix on *W(direction)* probability matrix,

g (*1-g*) - the sum co-efficient determining the amount of influence of the summation (maximum) on the accumulated score,

y - the punishment parameter.

Previous tests on the algorithm have shown that optimal results occur for the following parameter values: *a*=1, *p*=0.5, *y*=24. The next section deals with the issue of the memory coefficient and discusses the other two parameters, *b* and *g*.

4.5.5 The effective memory coefficient (*EMC*)

As can be seen from the accumulated score formula, *b* determines the memory persistency or the decaying influence of the scores from the previously processed frames. To prevent the algorithm from 'exploding' an *effective memory coefficient* (*EMC*) has to be within the range of values [0..1]. The *EMC* is introduced due to the fact that *b* itself is not the memory coefficient, since it is multiplied by *g* or *(1-g)*. The smaller *b* is the less effect the previous frames have on the current score and the overall score will be lower. A high value of *b* will cause high memory persistency and the system might suffer from low adapting capabilities, which is important in cases of altering direction targets. Correct value of EMC can make the detection process much more effective by getting rid of old data, which is no longer needed for the updated decision making, thus providing faster detection capabilities of well maneuvering targets.

In previous research a preferred memory coefficient was obtained: β=0.8. The formula for the accumulated score in the *DPA-1D* was: $Score_n = anti\text{-}mean_n + \beta \cdot max\{w \cdot Score_{n-1}\}$ where β=0.8/*max{w}* so that $0 \leq \beta \leq 1$. In the current *DPA-2D*, the score formula is: $Score_n = anti\text{-}mean_n + \beta \cdot [g \cdot sum\{w \cdot Score_{n-1}\} + (1-g) \cdot max\{w \cdot Score_{n-1}\}]$.

Similarly to *DPA-1D* we demand: $b = \beta = 0.8 / [g \cdot sum\{w\} + (1-g) \cdot max\{w\}]$. By substitution of *g* = 0, *g* = 1, we get: $\beta_{g=0} = 0.8 / max\{w\}$, $\beta_{g=1} = 0.8 / sum\{w\}$. After rearrangement, we get the following formula for *EMC*:

$$b = \beta = \left[\frac{g \cdot sum\{w\} + (1-g) \cdot \max\{w\}}{0.8} \right]^{-1} = \left[g \cdot \beta_{g=1}^{-1} + (1-g) \cdot \beta_{g=0}^{-1} \right]^{-1} \tag{4.19}$$

Using the probability matrices achieved empirically an appropriate range for *b* can be found so that $0 \leq EMC \leq 1$. From the calculation *sum{w}* = 1 for all directions, *max{w}* = 0.1054 for directions 1 to 8, and *max{w}* = 0.1719 for direction 9 (standing in place). Substituting that we get: $\beta_{g=0}$ = 7.5913, for direction 1 to 8, and $\beta_{g=0}$ = 4.6545 for direction 9, and $\beta_{g=1}$ = 0.8 for all directions. Putting that into β, we get : $\beta = [1.1183 \cdot g + 0.1317]^{-1}$ for direction 1 to 8, and $\beta = [1.0352 \cdot g + 0.2148]^{-1}$ for direction 9. So now we have a formula that tells us what values to give *b* for a given value of *g*. Since $0 \leq g \leq 1$, a graph of *b* vs. *g* can be plotted, as in the graph below, and a valid range of values for *b* can be found.

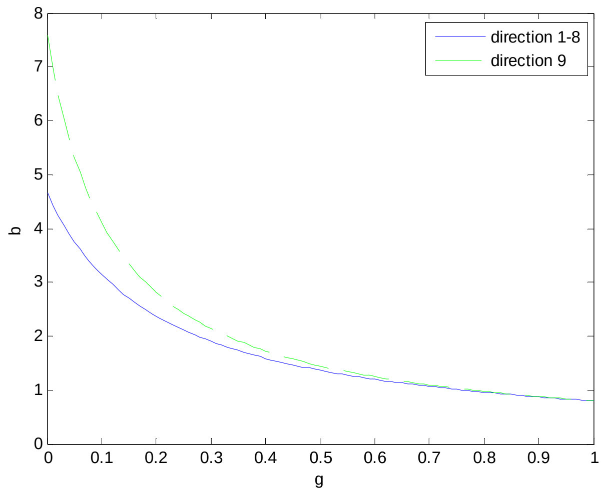


Figure 15: <u>b vs. g for the direction 1-8 (solid line) and direction 9 (dashed line)</u>

As can be seen from the graph, a valid range of values for *b* is about [0..8] for *g*=0 and [0..1] for *g*=1. Since $\beta_{g=1}$ = 0.8 for all directions, memory values of around 0.8 should give the optimal performance for *g*=1, regardless of the scenery. Since the targets in the tested IR sequences move at sub-pixel velocities, mainly 0.2-0.3 [*ppf*], most of the time they are on direction 9 (standing in place), thus we expect the algorithm to be most effective for $\beta_{g=1} = 0.8$ and $\beta_{g=0}$ close to 4.65.

Finding the optimal *b*'s for the different scenes will be done by optimizing an algorithm score, given by a metric that will be defined in the following section.

5 System evaluation

5.1 Evaluation of the temporal algorithm on synthetic data

5.1.1 Synthetic IR frames creation

In order to evaluate the performance of the spatial and temporal tracking algorithm, synthetic temporal profiles, which simulate different types of clutter and background behavior, are created. Target signal is implanted into these background signals in order to simulate target traversing both clutter and noise dominated scenes. [11] states that the temporal noise is closely matched to *white gaussian noise* (*WGN*), therefore *WGN* at various *SNR*s is used to test the algorithm. *SNR* is defined as:

(5.1)
$$SNR = \frac{MaxT}{\sigma}$$

Where σ^2 is the noise variance, *MaxT* is the target's maximum peak amplitude.

Figure 16 below shows the different types of signals used to test the algorithm.

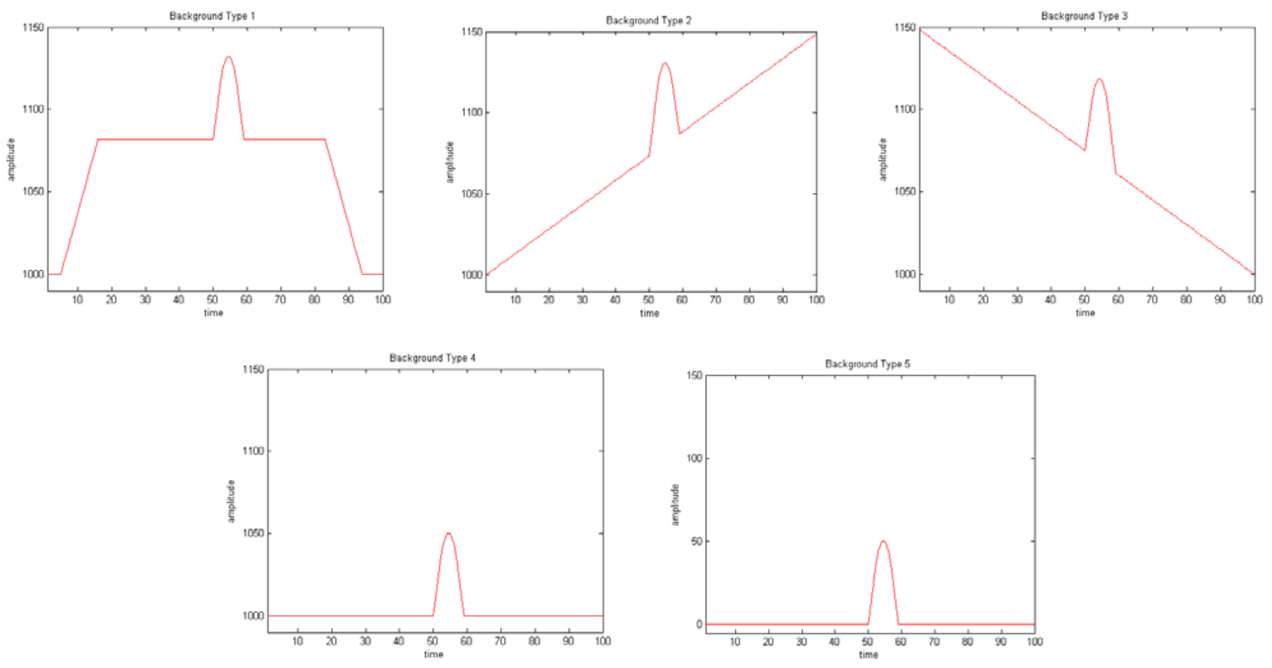


Figure 16: Synthetic background signals.

The type 1 signal simulated relatively fast and small clutter formation passing through a pixel. The type 2 signal simulates slowly entering clutter; Type 3 is the symmetrical slowly existing clutter. Type 4 simulates a noise dominated scene in which the base time line is constant; the type 5 signal serves as reference to the best-case scenario which comprises a constant zero-mean base line.

Target temporal profiles are characterized by a rapid rise and fall pattern. This behavior might be modeled by a half sine or triangular shape as shown in Figure 17.

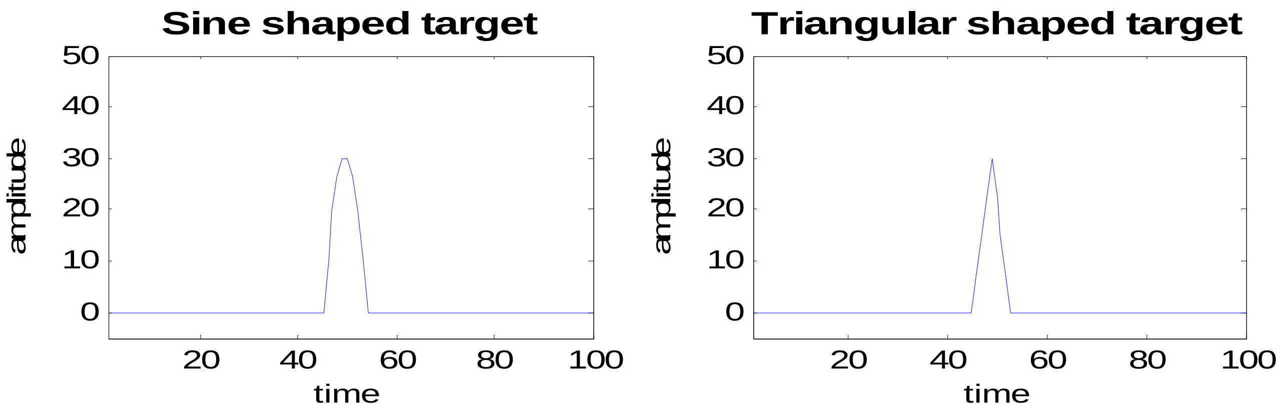


Figure 17: <u>Synthetic target examples</u>.

The base width corresponds to target velocity. Simulations show that there are no significant performance differences between the sine and triangular shaped targets [1] .

Figure 18 on the next page shows the various background models with the sine shape implanted at an SNR of 4.

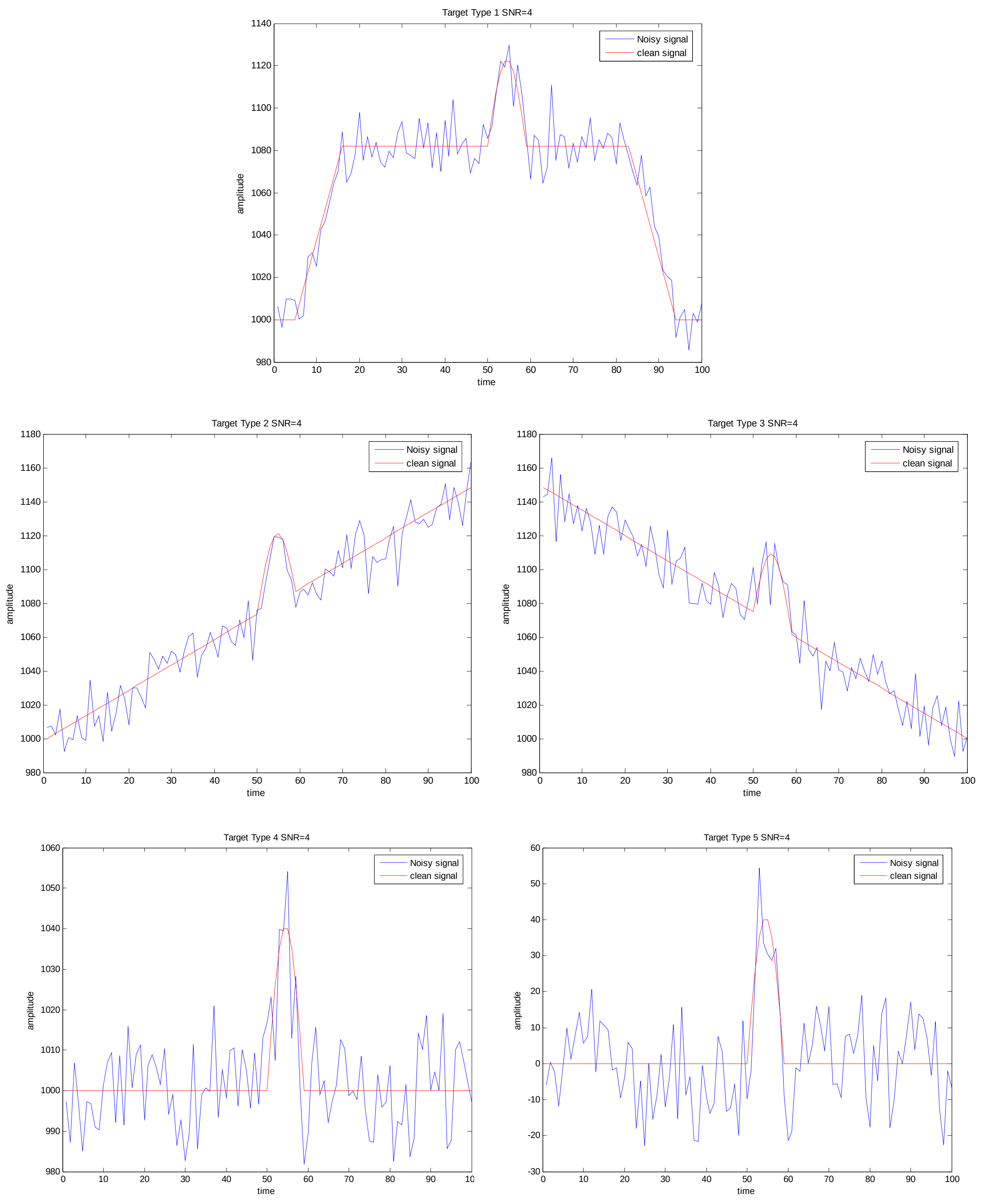


Figure 18: <u>Example of synthetic signals with the implanted sine target, SNR=4</u>.

5.1.2 Examination of the temporal algorithm on synthetic data

This section demonstrates the algorithm's operation on the synthetic data described in the previous section. The following parameters affect the algorithm's performance:

1. The background type.
2. The *SNR*, which is based on noise variance and the target's width and amplitude.
3. Parameters of the windowing procedure: the window sizes to estimate the background baseline DC, the grouping spatial window size to convert sub-pixel target velocity to the pixel target velocity at the frame (as an input to the *DPA*), size of short term variance at each sample and at each grouping and the step size of each (Overlapping).

The following subsection describes the dependence of the performance of the algorithm on these factors.

5.1.2.1 The background type

The background type affects mostly the *DC* estimation capabilities of the algorithm. It is expected that signals having constant *DC* level (signals of Type 4 and 5) and signals having slowly changing *DC* (signal of Type 2 and 3) would be the easiest in terms of *DC* estimation, since the linear regression is capable of estimating the two parameters of the linear model. Signals of Type 1 are the most problematic since their *DC* is not overall fitting a linear model, but depends on piecewise *DC* matching windows sizes as will be explained later.

Figure 18 until Figure 28 illustrate the algorithm's operation on the various signal types. The figure shows the results of signals with and without implemented target. The *DC* signal is plotted as well as the estimated variance values which were calculated after subtracting the estimated *DC* from the signal. The simulation was run for *DC* window of 20 samples, DC overlap of 50%, sub temporal profile of 15 samples, overlap between sub profiles of 5 samples and SNR of 4. The target width is 10 samples.

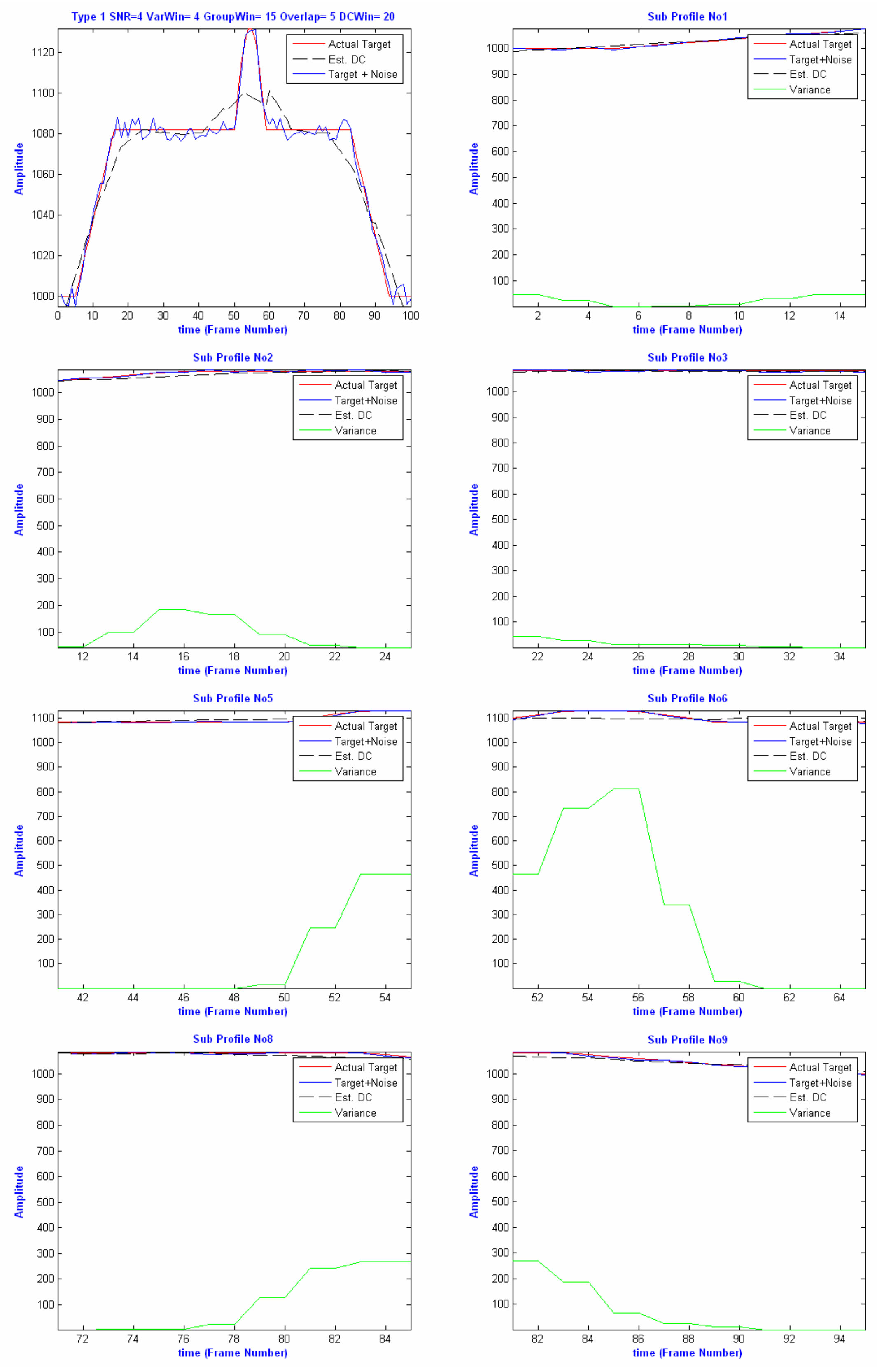


Figure 19: <u>Example of the temporal algrithm operation on the Type1 signal with target</u>.

As can been seen in Figure 19 the increase in the variance of sub-profiles 2, 8 and 9 occurs due to the imprecise *DC* estimation of the background. This case simulates a cloud entering and exiting.

Nevertheless, the variance score of the target (sub-profile 5, 6) is still much higher than sub-profiles 2, 8 and 9. The variance of the other sub-profiles (1, 3, 4, 7) is close to zero.

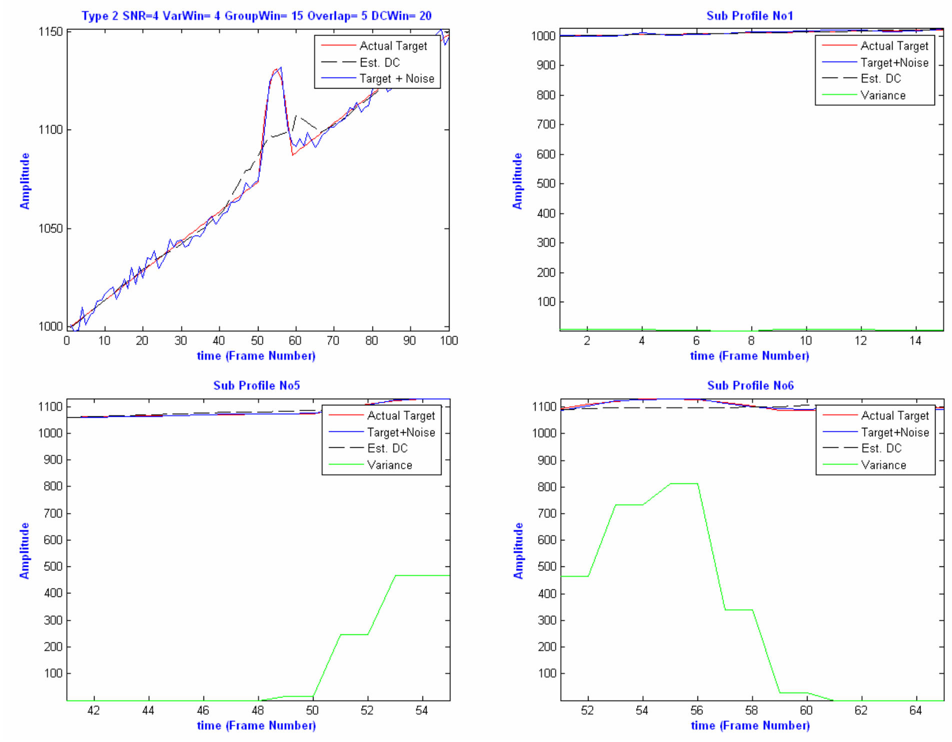


Figure 20: <u>Example of the temporal algrithm operation on the Type2 signal with target</u>.

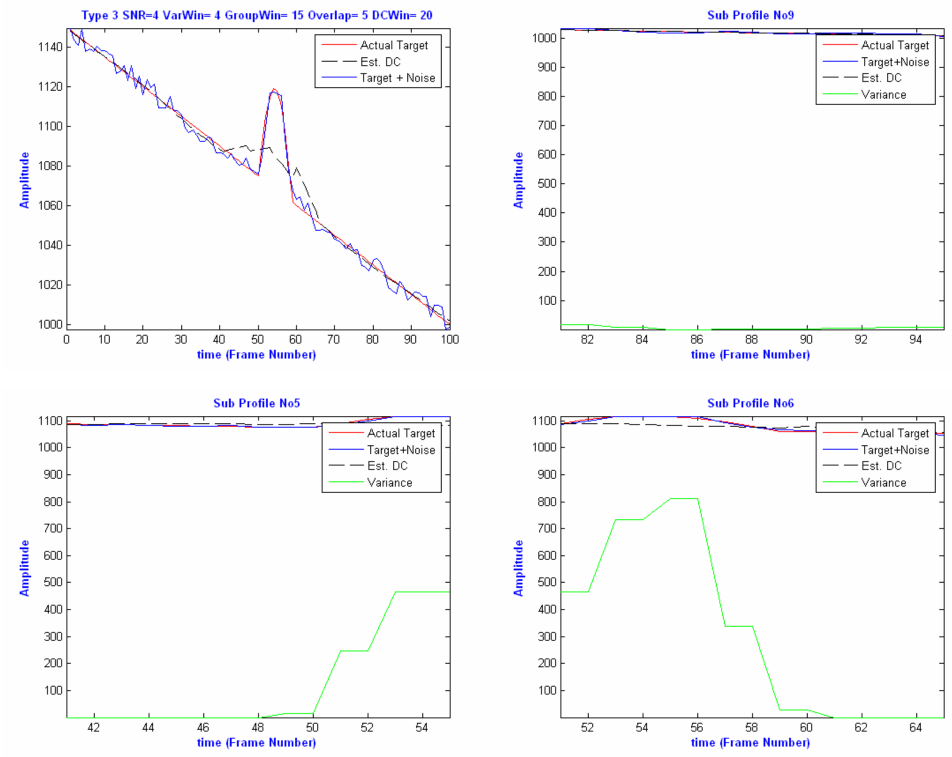


Figure 21: <u>Example of the temporal algrithm operation on the Type3 signal with target</u>.

The *DC* estimation for signals Type 2 and 3 is precise since the signal has a linear model. The variance increases significantly when the target passes thru the pixel and is close to zero at other times.

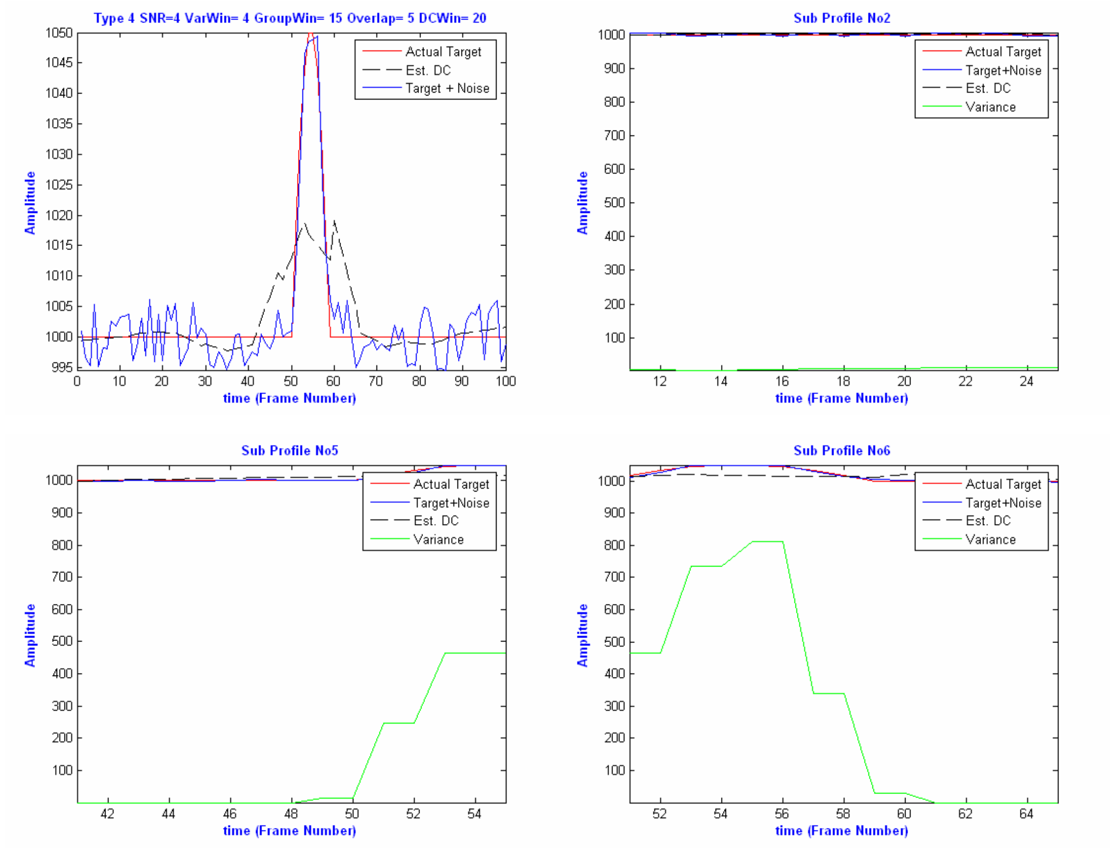


Figure 22: <u>Example of the temporal algrithm operation on the Type4 signal with target</u>.

Figure 22 and Figure 23 show a similar behavior of signal for different *DC* levels. As expected, the variance of each of signal (Type 4 and Type 5) is the same.

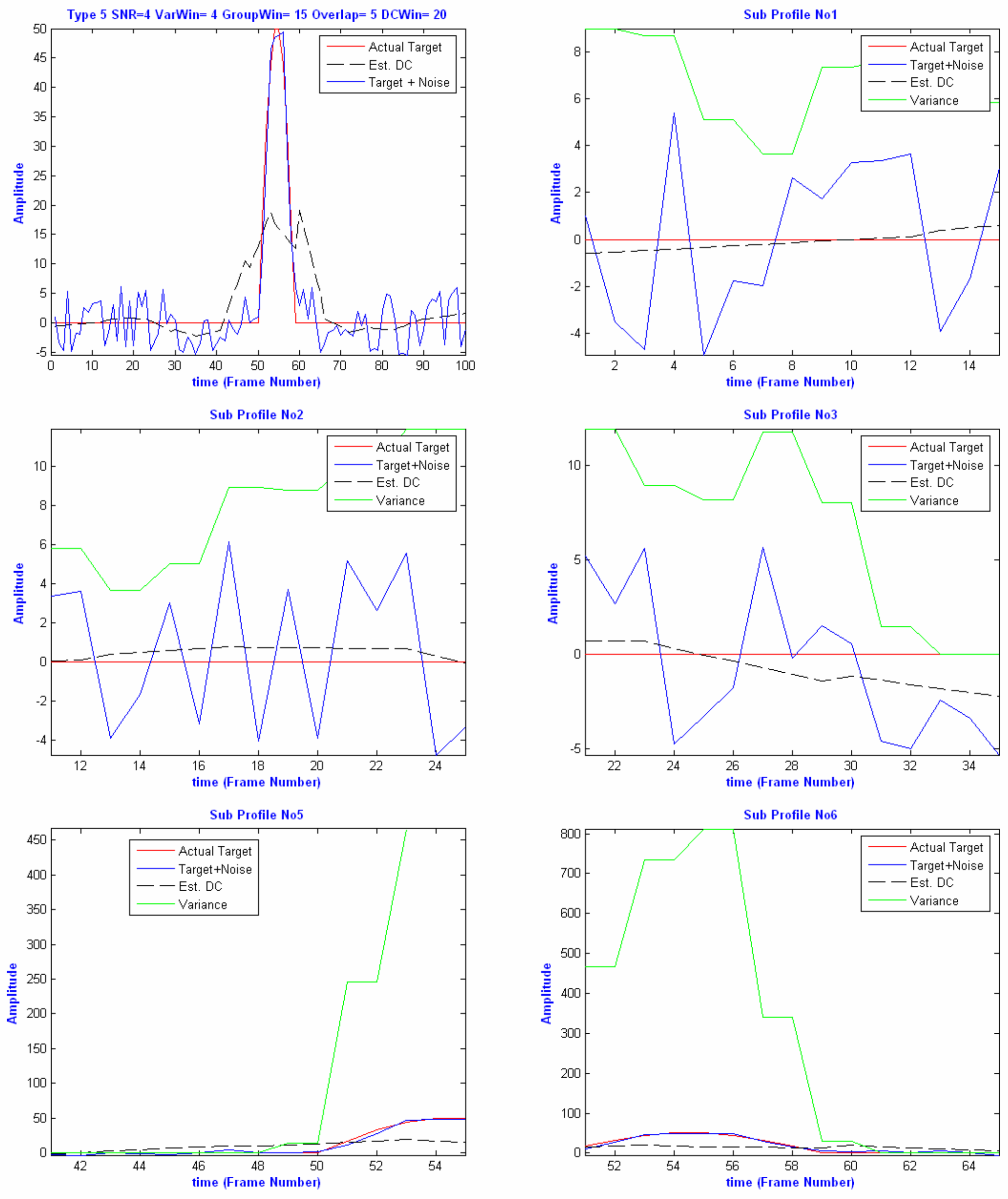


Figure 23: <u>Example of the temporal algrithm operation on the Type5 signal with target</u>.

The next figures show the results of temporal processing (variance estimation) for the different signal types, in the case where no target is present. Not all of sub-profiles are shown, since the exhibited variance values are around zero.

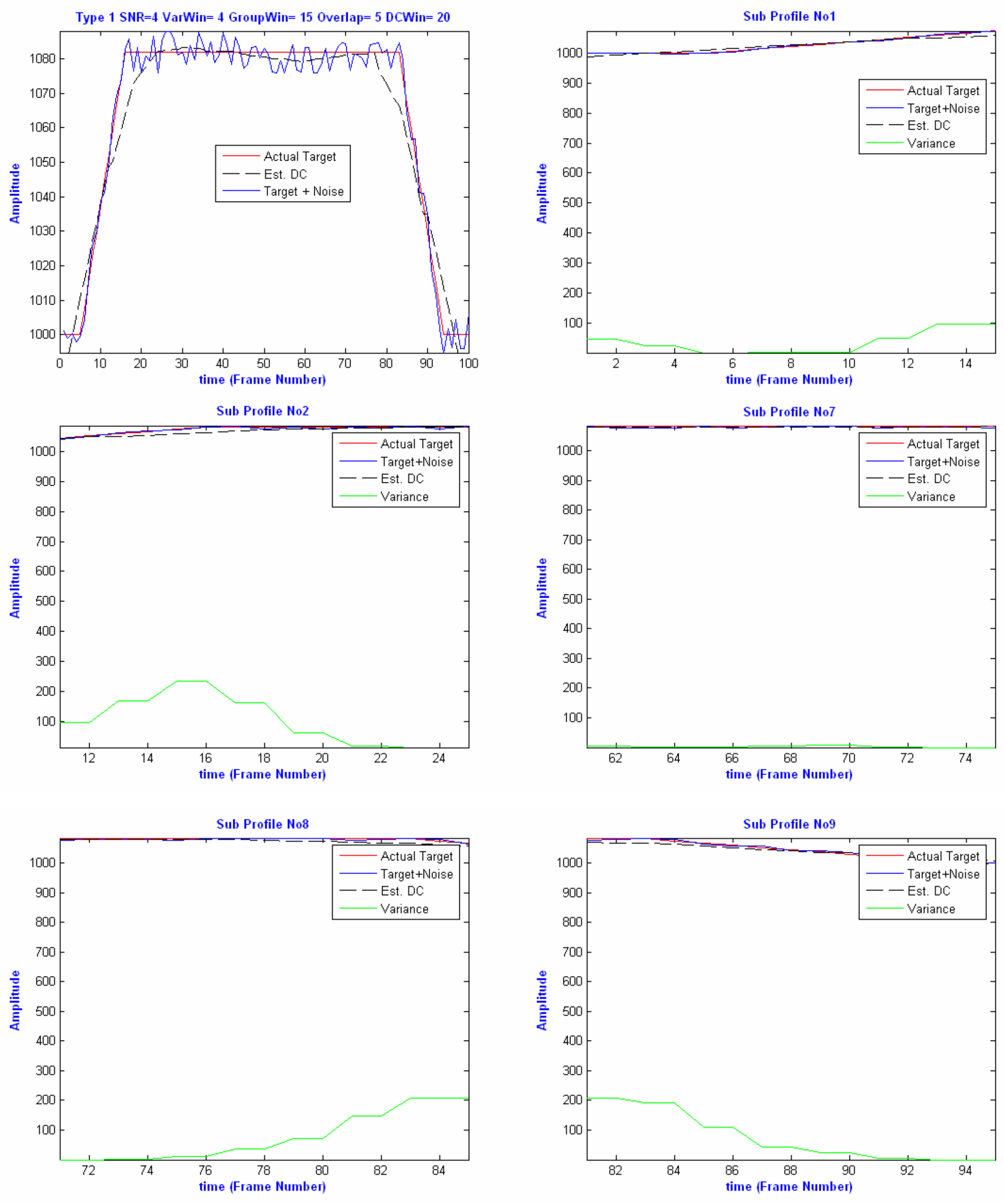


Figure 24: <u>Example of the temporal algrithm operation on the Type1 signal without target</u>.

By analogy to Figure 19, the variance increases at sub-profiles where the cloud enters and exits. Such cases (without target) may cause false alarms (*FA*).

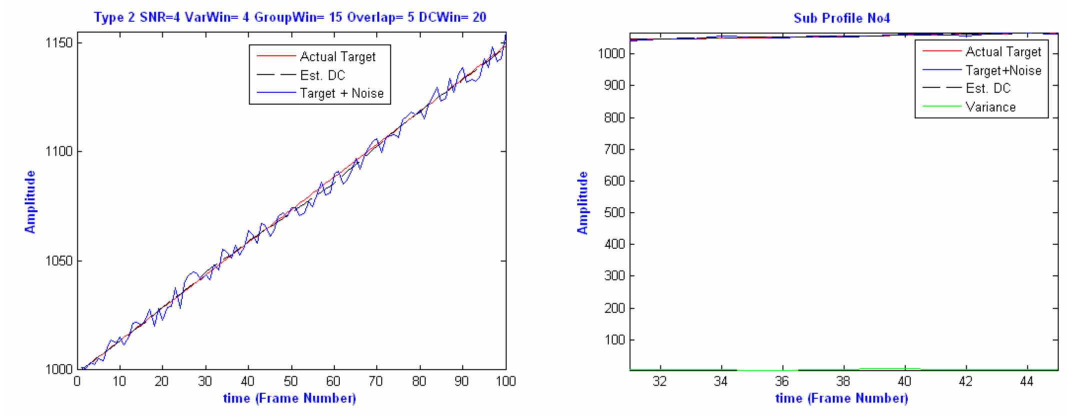


Figure 25: <u>Example of the temporal algrithm operation on the Type2 signal without target</u>.

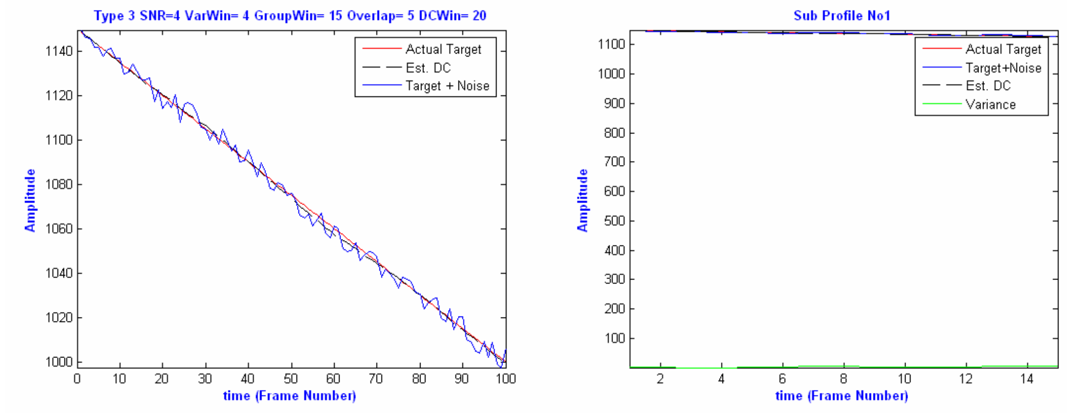


Figure 26: <u>Example of the temporal algrithm operation on the Type3 signal without target</u>.

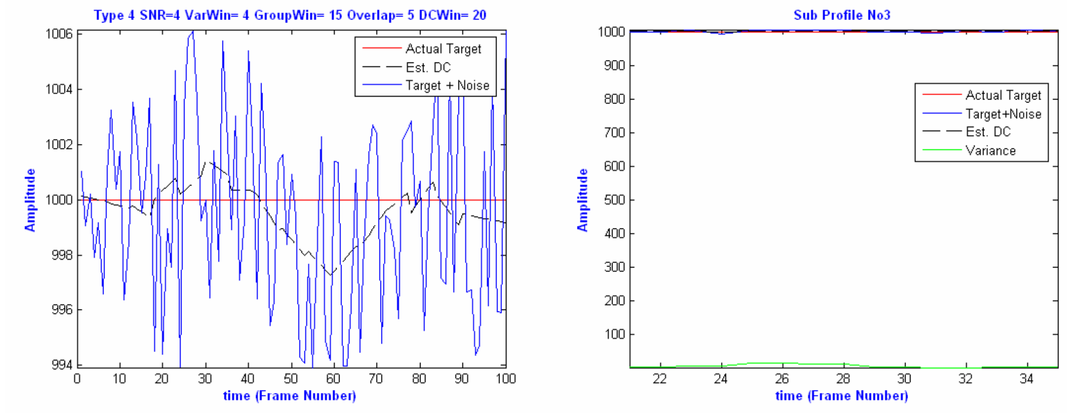


Figure 27: <u>Example of the temporal algrithm operation on the Type4 signal without target</u>.

Figure 25 to Figure 28 show that the temporal processing score is close to zero for signals without a target.

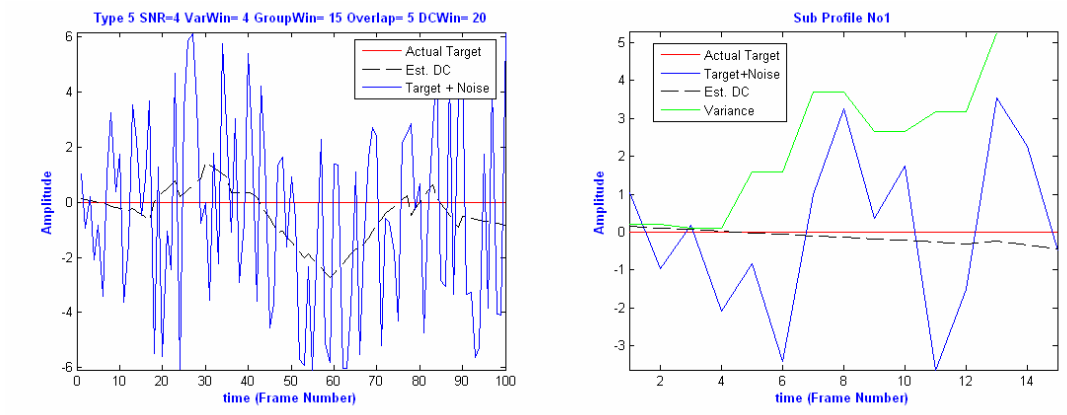


Figure 28: <u>Example of the temporal algrithm operation on the Type5 signal without target</u>.

At this point a simple means of performance evaluation is the ratio of the score obtained from the synthetic signal which contains the target and the same signal, but without a target (Target/Noise or T/N ratio). Table 1 shows the T/N ratio of the different signal types obtained by averaging 500 rehearses.

It is important to state that the T/N ratio is not the metric used to evaluate the algorithm as will be shown later.

Signal Type	T/N Ratio	T/N Ratio [14]
1	1.8098	1.72
2	10.9558	4.34
3	10.9658	4.41
4	10.9659	4.29
5	10.9659	4.41

Table 1: <u>Target/Noise ratio for varios signal types</u>.

As expected, the performance for signal Type 1 is the worst among the different signals. Signals 2-5 all have similar performance. If we compare between our sub-temporal processing algorithm and between a temporal processing algorithm as described at [14] it can be clearly seen (Table 1, 2nd column) that the performance has increased by at least a factor of two for signals 2-5, and has an insignificant increase for signal of Type 1.

5.1.2.2 The *SNR*

As in many other applications, the *SNR* is an important factor which affects the temporal algorithm's performance. The higher the *SNR*, the better the algorithm's performance is expected to be since both the *DC* and variance estimation will be more accurate. The results are shown in Figure 29.

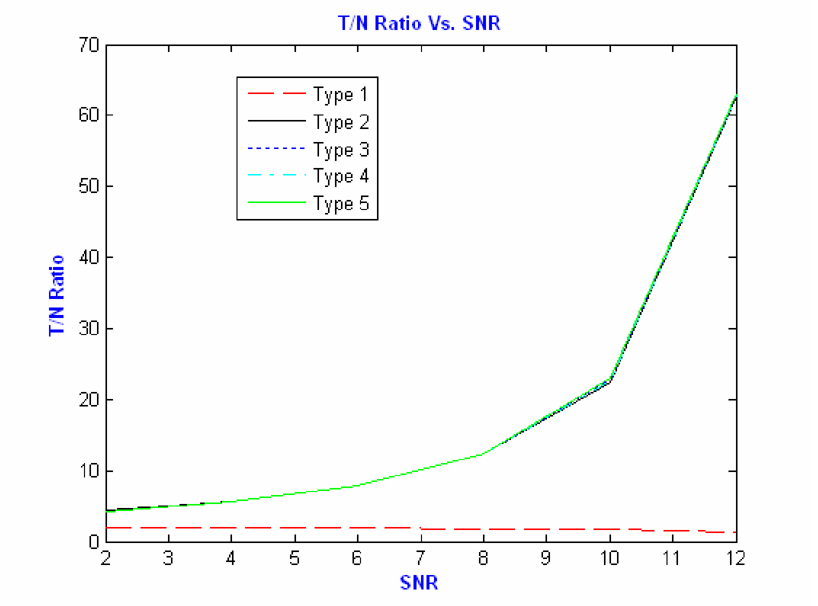


Figure 29: <u>T/N ratio Vs. SNR for different signal types</u>.

The algorithm has similar response for signals of Type 2-5, in agreement with the expectations – the performance increases as the *SNR* increases. The performance for signal Type 1 behaves differently – it increases with the *SNR* but at a slower rate than signals 2-5 and decreases later on. Roughly, the curve of signal Type 1 is a scaled version of the curves of signals of Type 2-5; the scaling factor originates from the inaccurate *DC* estimation, as will be detailed later. Since the *DC* of this signal doesn't fit to a linear model, and the estimation is performed in piecewise fashion, the size and the position of the windows used to perform the estimation act as limiting factor on the performance.

5.1.2.3 The window sizes

The window size for the baseline *DC* estimation, as well as the window size for the short term variance at the sub-profile has a great impact on the algorithm performance.

Larger window sizes for baseline *DC* estimation should improve the *DC* estimation in cases where the background changes monotonically (like signals of types 2-4). A too large a *DC* estimation window size might, in some cases, lead to inaccurate tracking of the clutter form and cause high false alarm rates (e.g., like in Type 1 signals). Thus, for background profiles the optimal window size is determined by the background type – for noise dominated background or backgrounds containing monotonically changing clutter, larger window sizes are preferred; for background containing rapidly changing clutter shorter window is preferred. For target temporal profiles, the larger the DC window, the higher the profiles score since the presence of the target peak will less influence the DC estimation, obviously estimated *DC* which tracks the target form is highly undesirable since it leads to target suppression. Thus the optimal *DC* window regarding the overall algorithm performance is the one which closely tracks background changes on one hand, but is large enough not to track the target peak.

The sub-temporal profile length should be matched to the target sub-pixel velocity which expressed as the base width of the peak of target profile and the sub-pixel velocity, although there is no acute need for an exact match. The short term variance window size at the sub-profile should be matched to the target rise time, although there is no acute need for an exact match as at the sub-temporal profile length. A sub-profile length which is larger than the target width will disable the ability to track/detect target with a sub-pixel velocity. Alternatively, a sub-profile length which is smaller than the target width will allow too few samples at the sub-profile.

A short time variance window which is larger than the target rise time will result in lower score for the target profile, since the variance calculated on each window is normalized by the window's length. Thus for target profile the optimal variance window size is expected to be less than or equal to the sub-profile length, but not larger. In fact, the shorter the window, the higher the score of target profile. On the other hand, a short variance window is more sensitive to random noise spikes in temporal profile dominated by noise. Therefore the optimal variance window size for noise dominated profiles will tend to be as large as possible in order to diminish the effect of noise spikes. The optimal variance window for the overall algorithm's performance is the one which best compromises the need to enhance the target profile score (as short as possible) and the need to suppress the noise short-term fluctuations (as long as possible).

Another impact is the overlap window between sub-profiles. The overlap window should allow for the compensation of low sub-pixel velocity that derives a small sub-profile length. Another aspect of the overlap window is the need to create more sub-profiles, as defined at equation (4.4), since more sub-profiles aids to achieve a more accurate tracking estimation.

5.2 Evaluation of the DPA on real data

As stated in section 3.2.3.2, the performance metric is defined as follows:

The block SNR is given as:

(5.2)
$$Block\_SNR(i,j)=\frac{E\left[v_{i,j}\in M\right]-E\left[v_{i,j}\notin M\right]}{\sigma_{v_{i,j}}}$$

where $v_{i,j}$ is the set of pixels belonging to the *(i,j)*th block, M is a set containing the five pixels with the highest gray level in that block, and σ is the standard deviation of the block pixels. The formula performs a subtraction between the expectation value of the highest pixels (target) and the expectation value of the rest of the pixels (background), divided by the standard deviation of block pixels. Since the probability matrices introduce the influence of target pixels on adjacent pixels, these influenced pixels might accumulate higher values than unaffected pixels (background), and can be regarded as target pixels. This might lower the expectation value of the target, but will lower the standard deviation of the background, since these high pixels are higher than the statistics of the background, to a more accurate one.

The algorithm score is given as:

(5.3)
$$Score=\frac{Block\_SNR(i,j)_{(i,j)=TB}-E\left[\left\{Block\_SNR(i,j)\right\}_{(i,j)=NTB}\right]}{\sigma_{\left\{Block\_SNR(i,j)\right\}_{(i,j)=NTB}}}$$

Using this metric for the algorithm score, the *EMC* value for optimized algorithm work can be found, in both cases of *g=0* (the max part) and *g=1* (the sum part) in the accumulated score equation.

The *DPA* algorithm has been applied to several IR sequences containing different scenes and clutter degree. Each sequence contains 95 to 100 frames, and the algorithm ran up to 25 frames. The results are shown for two IR sequences – *na23a* and *npa*. A single frame from the IR sequences showing the targets locations are shown in the figure below (in white rectangles). The *na23a* sequence has a single target moving in clear sky from right to left at $v\approx 0.3$ [*ppf*]. The *npa* sequence has two targets moving at $v\approx 0.2$ [*ppf*], the left target is engulfed in clouds moving from bottom to top and the right target is moving in scarce clouds. The algorithm will be applied on the surrounding of the left target, since it is of more interest due to its severe clutter condition.

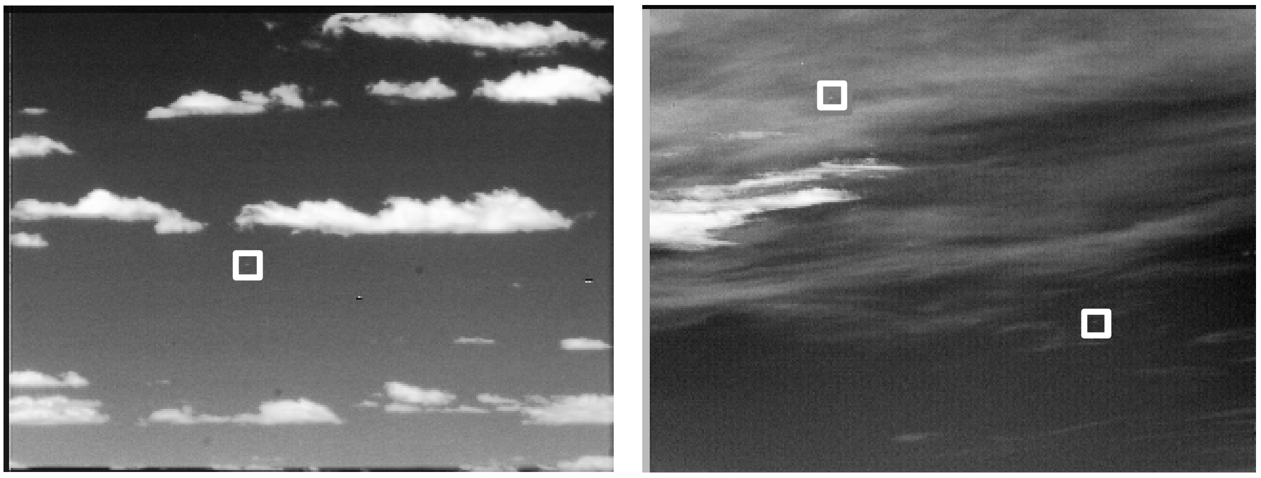
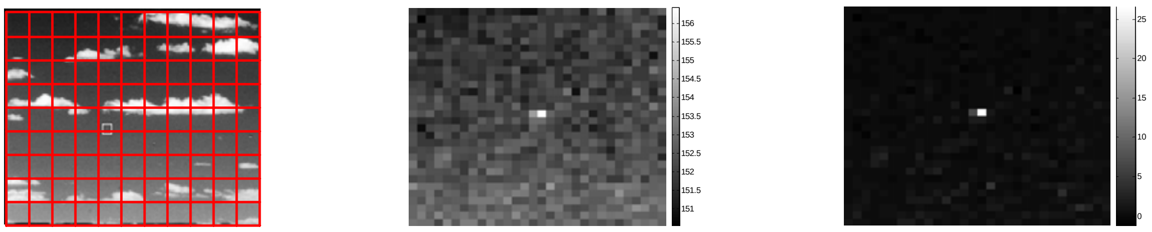


Figure 30: A single frame from (a) *na23a* (b) *npa* sequence showing the target's locations (white rectangles).

The *DPA* is applied for two different cases: g = 0 and g = 1, for different values of b, to check the effect of each part of the accumulated score formula on the algorithm results. The metric applied afterwards is used to find optimal values of b's. Afterwards, the effect of *Max_Numbers* (the number of pixels with the highest scores considered to be 'target' in the *SNR* formula) on the algorithm score is checked, to make sure the metric gives reliable score performance for the different *EMC*'s (b) .

5.2.1 The Preprocessing Stage (anti-mean)

The algorithm starts by whitening the input IR sequence. According to the introduced score metric, the sequence is divided into blocks. The target block and the eight surrounding blocks are then whitened (anti-mean) to reduce the clutter and emphasize the target, as can be seen in the figure below from the 1st frame of the *na23a* sequence.



(a) (b) (c)

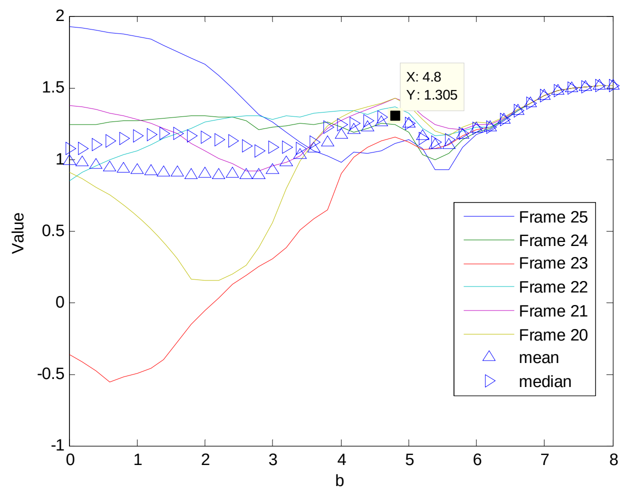
Figure 31: **<u>(a) 1st frame of the original *na23a* sequence divided into blocks, (b) 1st frame of target block (target at center of frame), (c) 1st frame of target block after the preprocessing (ANTI-MEAN) Stage</u>.**

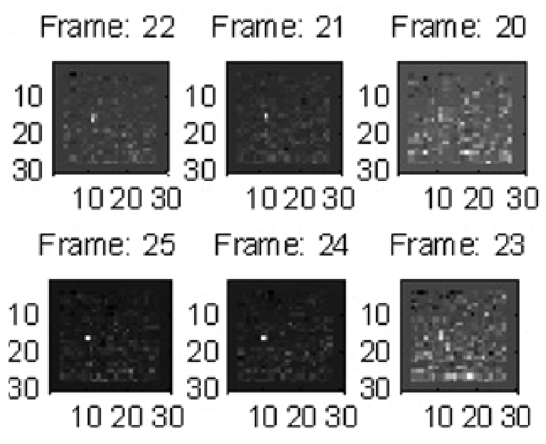
The process shown in the Figure 31 is repeated for the entire sequence before proceeding to the next stage of *DPA* (The frames can also be processed individually for a real-time implementation).

5.2.2 *na23a* sequence

5.2.2.1 Preliminary results (*sampling* = 1)

We start by showing the result for the *na23a* sequence, $g = 0$. Figure 32 shows the algorithm score for the range of $b = [0...8]$ at jumps of 0.2. The graph shows the scores for the last six frames, 20-25 in this case, for two reasons: 1. some frames might be noisier than the others; 2. we wish to show the accumulated score effect. The first reason might cause the target to be dimmer compared to the clutter or disappear. In that case, the memory persistence and the accumulation of the score from frame to frame helps, leading to the second reason, the accumulated score helps overcome noisy frames, and 'accumulates' *SNR* for the dim target. Looking at the last frames helps to see whether the effect of accumulation is enough for the number of frames processed, or whether more frames need to be processed. In previous research, [20] [21]), the algorithm was run over 10 to 20 frames, and 20 was found to be better, hence 25 frames were taken in this case, to be on the safe side. The graph shows a peak at around $b = 4.8$ for all frames. The rise at about $b = 6$ is irrelevant since the algorithm was unable to track at *EMC* of about 6 or above. It can be seen that above that *EMC*, all the frames scores converge, meaning that the *EMC* is too high and the weight of the current frame is insignificant to the accumulated score. It seems that the relevant range for *EMC* can be narrowed to *b* = [0…6], for the *g* = 0 case.





(a) ***(b)***

Figure 32: <u>(a) The algorithm score for the *na23a* sequence, *g* = 0, *Max_Numbers* = 5, *b* = [0…8], and six last frames (b) Images of the six last frames after the *DPA* algorithm</u>.

A rise in the algorithm score is expected as the frames advance. It can be seen the graph Figure 32 (a), that the score of the 20, 23 frames is lowest for b < 3.5. That is due to the fact that these frames are noisy, as can be seen in Figure 32 (b). Increasing the *EMC* improves the score in them – more memory persistence and less weight to current noisy frame. Since the target moves at v ≈ 0.3 [*ppf*] (stays in the same pixel between pixels most of the time) the expected peak was at around *b* = 4.65 in agreement with the results.

Figure 33 shows the graphs of the standard deviation of the *NTB*, the expectation value of the *NTB SNR*, the *TB's SNR* (all vs. *b*), and the last frame (25) for various values of *b*. For *b* = 5.4 the last frame shows high values for the trajectory of the target, meaning that the *EMC* is very high and every pixel traversed by the target retained it's high value (Figure 33 (d)). Thus, *EMC's* higher than that will prevent the algorithm from tracking the target, since the trail pixels and the pixel of origin will gain higher accumulated score, as will be seen later on (Figure 51).

We continue by showing the results for the *g* = 1 case in Figure 34. As can be seen from the algorithm score, the relevant range of *b* is [0…1], after which the scores of the various frames converge regardless of the *EMC*. The *TB's SNR* can be seen to fall at around *b* = 1. The algorithm score starts at its maximum (mean value) for no memory and drops as the *EMC* rises, until *b* = 1. This means that in this case, the algorithm is not needed for the detection.

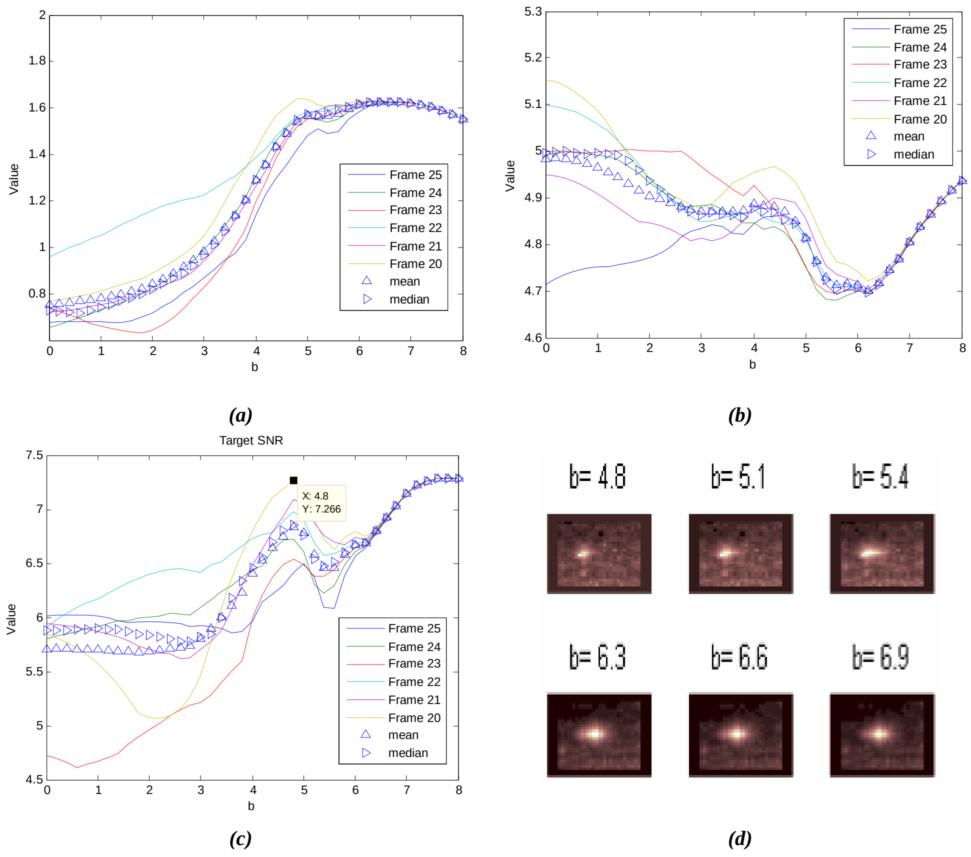
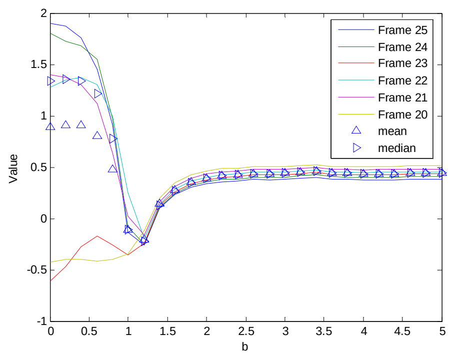
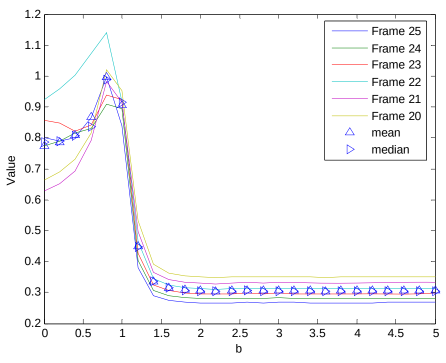


Figure 33: <u>(a) Standard deviation of *NTB*'s *SNR*, (b) Expectation value of *NTB*'s *SNR*, (c) *TB*'s *SNR*, (d) Last processed frame (25th) for various *b*'s</u>.

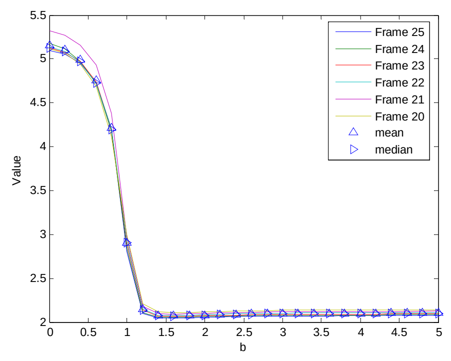
Figure 35 shows the influence of the *EMC* on the pixels of the last frame. In the *g* = 1 case, when starting to raise the *EMC* to too high values (above *b* = 1), no trail appears as in the *g* = 0, but a '*snow ball*' effect starts to build. The '*snow ball*' grows until the target is engulfed in it, and no tracking is possible. Due to the probability matrix and the summation, pixels close to where the target passes grow in their value. If the *EMC* is high, these adjacent pixels will also receive high accumulated scores, and so the ball grows from frame to frame. The *ANTI-MEAN* component weakens, and its weight decreases. Since the current frame influence is negligible, the main influence comes from the previous frames. This causes the graph to have constant spaces between the frames. Since the expectation values of the blocks grow as the frames progress, the *SNR* decreases (The target expectation value grows, but the 'background' expectation value also grows, so that the numerator decrease in the SNR formula). There is a convergence to an asymptotic value, where frame 20 gets the highest score, whilst frame 25 gets the lowest score (due to the lower *SNR* as frames progress).



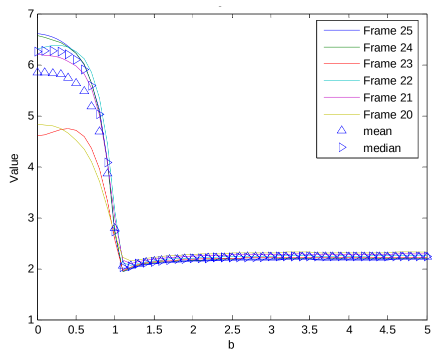
(a)



(b)



(c)



(d)

Figure 34: <u>(a) Algorithm score for *q* = 1, (b) Standard deviation of *NTB*'s *SNR*, (c) Expectation value of *NTB*'s *SNR*, (d) *TB*'s *SNR*</u>.

To conclude, a valid range of *EMC*, *b* = [0…0.8], has been found in which the algorithm is able to track and detect the target correctly; this concurs with the theoretical range.

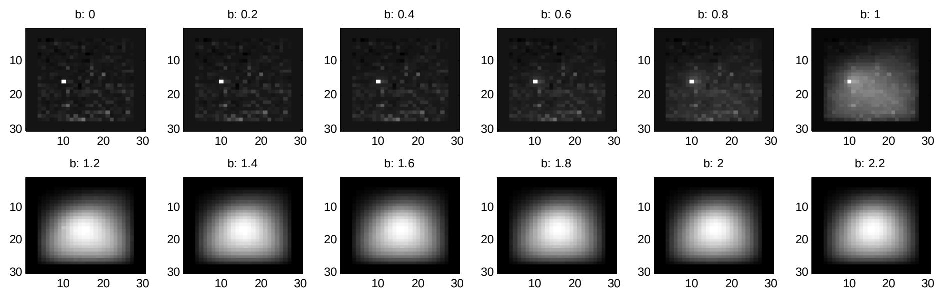
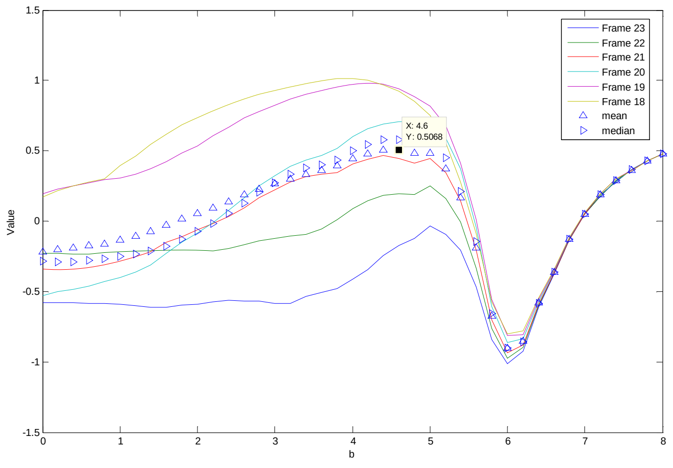


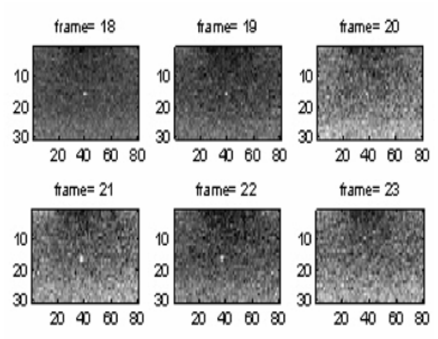
Figure 35: <u>Last processed frame (25th) for *b* = [0...1]</u>.

5.2.2.2 The *sampling* variable

Since the core of the algorithm is the usage of the punishment matrices, the target has to move at a velocity of at **<u>least around 1 [*ppf*]</u>**. If the target in the sequence moves at lower velocities, $v \approx 0.3$ [*ppf*] in our case, the punishment matrices are used only every three frames roughly. Since that is the case, sampling of the sequence has been suggested, so that the target moves at higher speed. The first simulation was done for *sampling* = 4 giving the target a velocity of $v \approx 1.2$ [*ppf*]. To keep the target in the '*target block*', a larger block size of 30*x*80 was used instead of 30*x*30. The algorithm score and the last six frames are shown in Figure 36, for the *g* = 0 case.

The score achieved is lower in this case due to the noisy frames compared to the last six frames in the *sampling* = 1 case. It should be noted that by sampling the sequence the algorithm, different frames were processed that might differ in their noise degree. Nevertheless, the score shows the effect of accumulation as the frames progress - the peak is distinguishable (above frame 20), and is around *b* = 5.0 for the last three frames. An increase in the *EMC* is expected since the target now moves faster. It seems that the algorithm needs around 20 frames for the dim point target to accumulate enough *SNR* to be distinguishable from the clutter.





(a) *(b)*

Figure 36: (a) *sampling* = 4: The algorithm score for *g* = 0, *Max_Numbers* = 5, *b* = [0…8], and six last frames, (b) Images of the six last frames after the *DPA* algorithm.

Figure 37 shows the graphs of the standard deviation of the *NTB*, the expectation value of the *NTB SNR*, the *TB*'s *SNR* (all vs. *b*), and the last found target track for *sampling* = 4 (the pink portion of the track is the target track for *sampling* = 1).

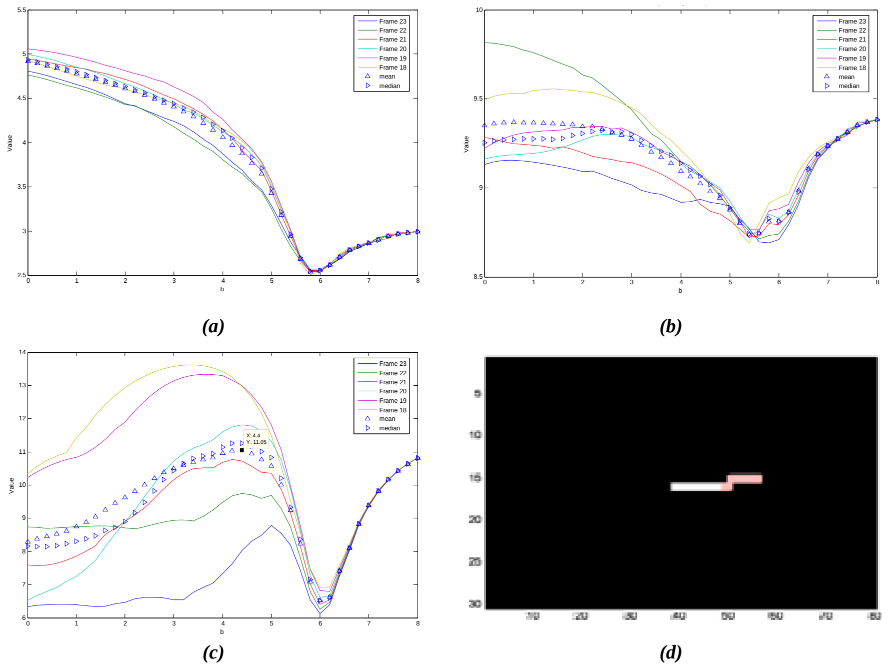


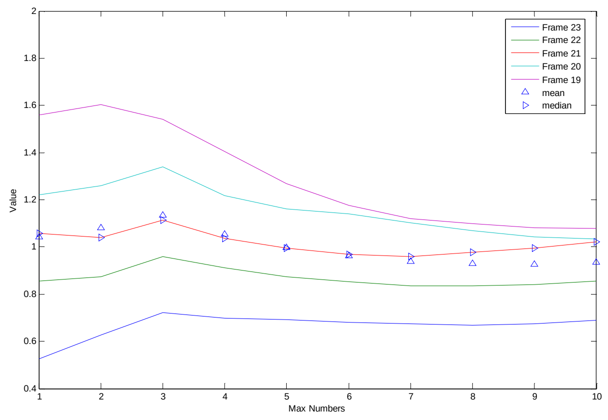
Figure 37 : <u>*sampling* = 4 : (a) Standard deviation of *NTB*'s *SNR*, (b) Expectation value of *NTB*'s *SNR*, (c) *TB*'s *SNR*, (d) the target's track (the pink track is the target track for *sampling* = 1)</u>

Comparing Figure 33 to Figure 37 shows that the *TB* & *NTB SNR*'s behave likely for *sampling* = 1 and only *TB SNR* has a peak for *sampling* = 4. This shows that the penalty matrices help distinguish between target and clutter, and that the algorithm needs targets at around *v* = 1 [*ppf*] to work effectively. In the case of *g* = 1, the algorithm score is lower compared to the one achieved for *sampling* = 1, due the noisy last frames, as in the *g* = 0 case.

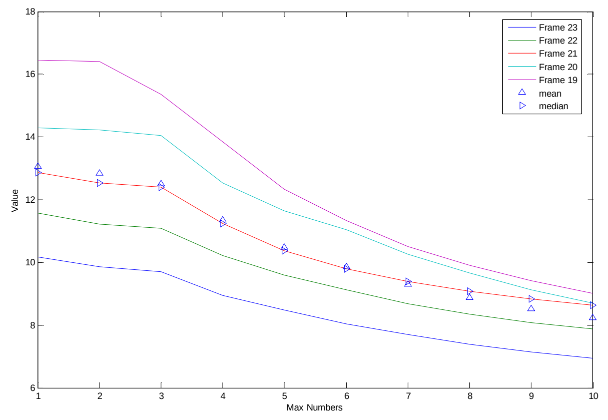
The next sub-sections deals with the issue of using values of the Max_Numbers parameter that will correctly take only target pixels as the highest pixels.

5.2.2.3 Finding Max_Numbers

As specified before, the parameter *Max_Numbers* controls the number of pixels considered as '*target*' pixels, and thus affects the result of the *SNR* of the blocks and consequently the algorithm score. So far *Max_Numbers* = 5 has been used. Instead of using an arbitrary number of highest pixels, a value for *Max_Numbers* can be found via optimization of the algorithm score, for the two cases of *g* = 0 and *g* = 1. A value of *b* = 4.8 and *sampling* = 4 was taken. The optimization was first run for the *g* = 0 case. Figure 38 shows a peak of mean algorithm score for *Max_Numbers* = 3 (there is a peak for all frames but the 19th frame where there is not enough accumulation as discussed before). A decrease in the TB's SNR can also be seen from the figure for *Max_Numbers* > 3.



(a)



(b)

Figure 38: <u>(a) The algorithm score vs. *Max_Numbers*, (b) The *TB*'s *SNR* vs. *Max_Numbers*</u>

Figure 50 shows the *TB*'s *SNR* vs. *b*. The graph is separated to three sections according to the resulting highest pixels in the last frame – (1) Target pixels, (2) Trail & Target pixels, (3) Pixel of origin and Trail pixels. The figure in the next page elaborates on the section separation. Note that the resulting separation concurs with the relevant range of *b* = [0…6]. The graph helps to distinguish a good range in which the highest pixels taken as 'target pixels' indeed belong to the target.

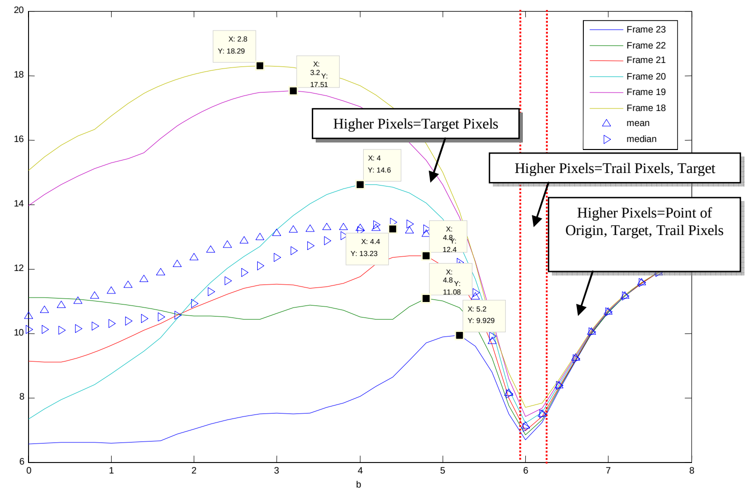


Figure 39: <u>The *TB*'s *SNR* vs. *b*</u>

In the case of the 5 highest pixels, the score takes only the target pixels up to *b* = 5.6. Above that value the trail pixels start to accumulate higher scores than the target pixels, until the pixel of origin also accumulates a higher score (*b* = 6.0). In the case of 3 highest pixels, the score takes only the target pixels up to *b* = 5.8. Higher *EMC*'s causes non-target pixels to be higher than the target pixels as before. In this case of *g* = 0 and *sampling* = 4, the algorithm was able to track and locate the target up to *b* = 6.0. Using that and the figure below leads to the conclusion that a good range of *EMC*'s would be *b* = [4.4 … 6]. This range contains the theoretical *EMC* for non-moving targets.

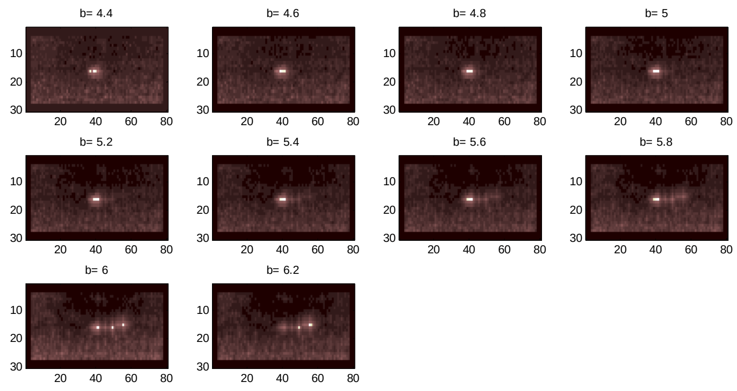


Figure 40: **The last frame of the algorithm, highest 3 pixels emphasized**

The discussion continues now to the *g* = 1 case. The last frames of the algorithm and the target's track vs. *b* are shown in Figure 52. It can be clearly seen that for *b* = [0…0.8] there is only one high pixel belonging to the target (surrounded by white circle). The algorithm tracks the target in the range of *b* = [0.2…0.8]. These results suggest that *Max_Numbers* = 1 and *b* = [0.2 … 0.8] should be used in the case of *g* = 1.

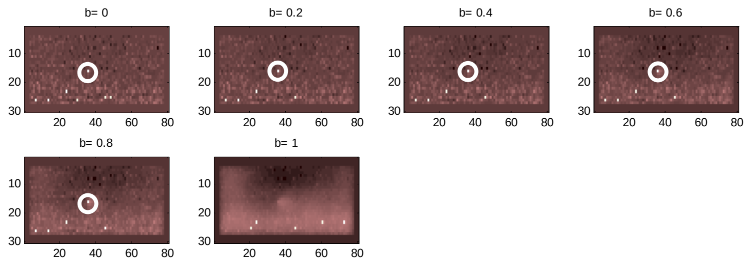


Figure 41: **The last frame of the algorithm, highest 5 pixels emphasized (target pixel surrounded by a white circle) vs. *b***

The algorithm score using Max_Numbers = 1 is shown in Figure 53 Below. A peak in the mean algorithm score can be clearly seen for *b* = 0.6 and the best tracking is achieved for *b* = 0.8 (not shown) in accordance with the theoretical value. The algorithm score is higher in this case than the *g* = 0 case for *sampling* = 4, and higher than both cases for *sampling* = 1. This result suggests that the algorithm performs better for faster targets, given correct value of *Max_Numbers* according to the case under investigation.

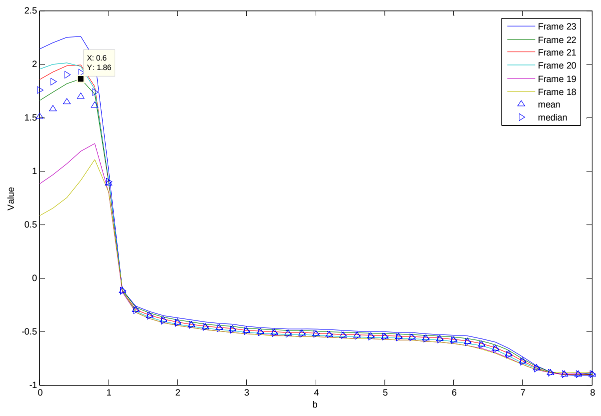


Figure 42: Algorithm score for g = 1 case, Max_Numbers = 1, sampling = 4

5.2.2.4 Summary

Results of the *na23a* sequence have been shown, followed by discussion and further results of the *Max_Numbers* and the *sampling* parameters. Preferable values for *Max_Numbers* were found, 3 for the *g* = 0 case, and 1 for *g* = 1 case. Relevant ranges of *b*'s were also found for each case, *b* = [4.4…6] for the *g* = 0 case, *b* = [0.2...0.8] for the *g* = 1 case. The core of the algorithm is the punishment matrices. If the target is moving at a sub-pixel velocity these matrices are not used often and the algorithm will not perform at its peak. In order to find the velocity for which the algorithm works best, the algorithm has been run over the sequence for *sampling* = [1…4] for both cases. For each sampling the highest score in the relevant ranges of *b*'s was chosen. It should be noted that for the algorithm to accumulate effectively, at least 20 frames have to be processed, thus limiting the amount of sampling that can be done, depending on the number of available sequence frames. Also, different frames are processed for the different samplings, so a comparison might not be precise. The results are shown in Figure 43.

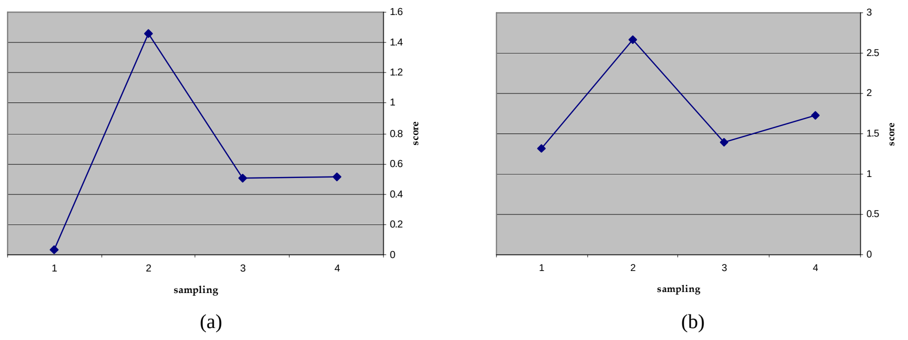


Figure 43 : Algorithm score vs. *sampling* (a) *g* = 0 case, *Max_Numbers* = 3 (b) *g* = 1 case, *Max_Numbers* = 1

The algorithm has a preferable target velocity at around *v*=0.6 [*ppf*].

5.2.3 *npa* sequence

The sequence contains a target moving at $v \approx 0.2$ [*ppf*] in the proximity of clouds. In this case, the cloud's edges in the target block pose a challenge since they behave like targets, and receive a high score from the whitening preprocessing stage. In this sequence, tracking and detection was achieved for *sampling* ≥ 3. Hence, detailed results for lower sampling values will be skipped and only detailed results for *sampling* = 3 will be shown. The summary sub-section will show scores for the various *sampling*.

5.2.3.1 Results (*sampling* = 3)

Figure 44 shows the following graphs for the $g = 0$ case : the algorithm score, the standard deviation of the *NTB*, the expectation value of the *NTB SNR*, the *TB*'s *SNR* (all vs. *b*), and the last frame (25) for various values of *b*.

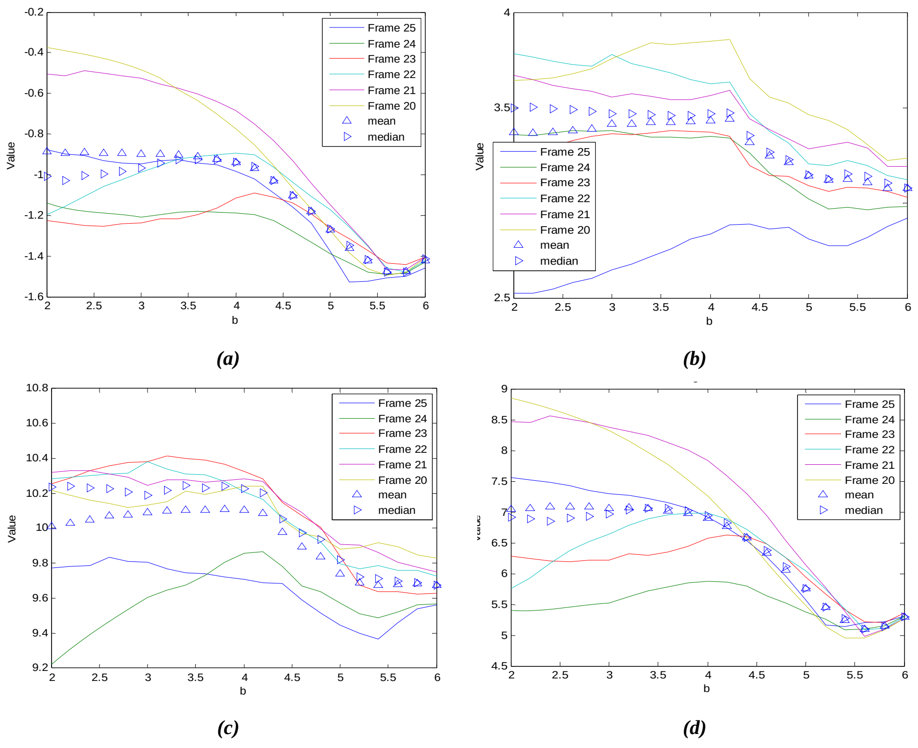
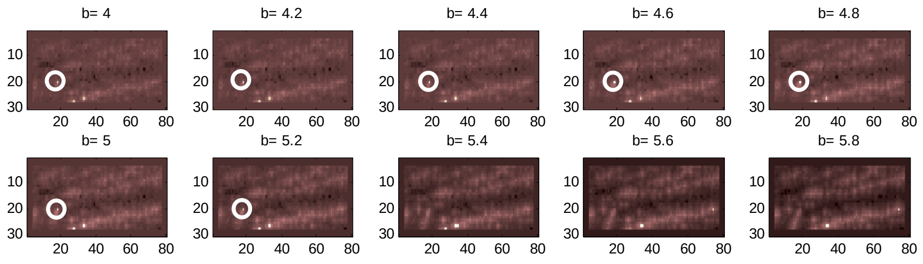
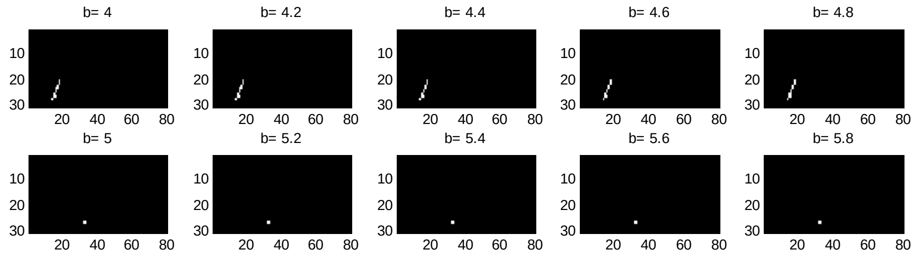


Figure 44 : *sampling* = 3 : (a) Algorithm score for the *npa* sequence, *g* = 0, *Max_Numbers* = 3 (b) Standard deviation of *NTB*'s *SNR* (c) Expectation value of *NTB*'s *SNR* (d) *TB*'s *SNR*

The resulting scores are negative for all the *b*'s in the range. This is due to the cloud edges at the bottom of the target block that get high accumulated scores (Figure 45 (a)). This causes the *TB*'s *SNR* to be low. Cloud edges in the *NTB* give rise to their *SNR*, leading to a negative algorithm score. Nevertheless the algorithm is able to track and detect the target for $b = [2\ldots4.8]$. Higher *EMC*'s lead to cloud's edge having higher score than the target itself in the *TB*, and no tracking.



(a)



(b)

Figure 45 : <u>(a) The last frame of the algorithm, highest 5 pixels emphasized vs. *b* (target pixel surrounded by a white circle), (b) The target track vs. *b*</u>

The results for the g = 1 have also been negative as the g = 0 case. Nevertheless, the algorithm has been able to track and detect the target for b = [0…0.8].

5.2.3.2 Summary

Results of the *npa* sequence have been shown. Preferable values for *Max_Numbers* were found, 3 for the g = 0 case, and 1 for g = 1 case. Relevant ranges of b's were also found for each case, b = [2…4.8] for the g = 0 case, b = [0...0.8] for the g = 1 case. The algorithm performed best for sampling = 3, where the effective target velocity was at around v=0.6 [*ppf*] as in the *na23a* sequence.

5.2.4 DPA Results Summary

Relevant ranges of b's for which the algorithm was able to track and detect the target, and the optimal *Max_Numbers* values, are shown in Table 2.

Accumulated Score Part	Relevant range of b's where tracking occurred		Max_Numbers
	npa sequence[2]	*na23a* sequence	
sum part (g = 0)	[2..4.8]	[4.4..6]	3
max part (g = 1)	[0..0.8]	[0.2..0.8]	1

Table 2: <u>EMC's for which tracking occurred in the IR sequences</u>.

[2] Tracking occurred for *sampling* $\geq$ 3

5.3 Evaluating of the temporal processing algorithm on real data

5.3.1 Real IR sequences

The real world IR image sequences from Reference [16] are used as a means for evaluating the temporal algorithm. The movies are comprised of 95 or 100 12 bit infrared frames. The sequences contain raw data of unresolved targets flying around Hanscom AFB. There are five scenes in the available dataset containing various types of clutters and sky as well as targets moving at different velocities.

Figure 46 shows a single frame (frame 50) of each of the IR sequences examined in this work.

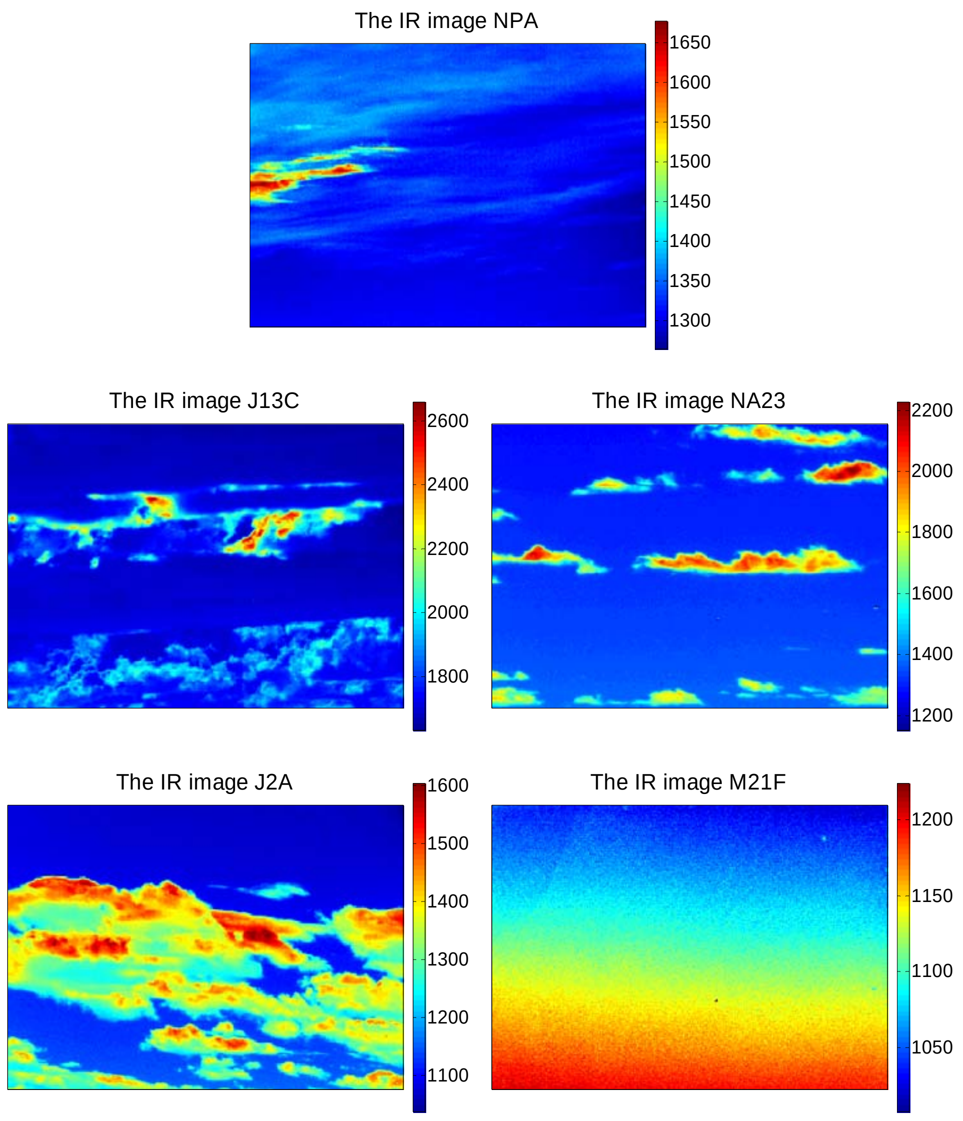


Figure 46: <u>Frame 50 of each real data IR sequence</u>.

Table 2 summarizes the number and nature of targets for each IR sequence, as well as the background type of scene.

IR sequence name	Scene description
NPA	two targets in wispy clouds
J13C	one slow target in clear of cloudy scene
NA23	one fast target in bright clouds
J2A	two targets in fluffy clouds
M21F	one weak target in hot hazy night sky

Table 3: Description of the IR sequences.

5.3.2 Evaluation metrics

In order to evaluate the algorithm a new metric has been defined, as describe at section 3.2.3.2. Each frame in the sequence is divided into blocks, the size of *HxN* (30*x*30 were used).The algorithm is run over 9 blocks – Target block and eight adjacent blocks. The *SNR* of the target block (*TB*) and its eight adjacent non-target blocks (*NTB*) are calculated. Afterwards, an algorithm score is calculated based on the resulting *SNR*'s.

The block SNR is given as:

(5.4)
$$Block\_SNR(i,j)=\frac{E\left[v_{i,j}\in M\right]-E\left[v_{i,j}\notin M\right]}{\sigma_{v_{i,j}}}$$

where $v_{i,j}$ is the set of pixels belonging to the *(i,j)*th block, *M* is a set containing the five pixels with the highest gray level in that block, and σ is the standard deviation of the block pixels. The formula performs a subtraction between the expectation value of the highest pixels (target) and the expectation value of the rest of the pixels (background), divided by the standard deviation of block pixels. Since the probability matrices introduce the influence of target pixels on adjacent pixels, these influenced pixels might accumulate higher values than unaffected pixels (background), and can be regarded as target pixels. This might lower the expectation value of the target, but will lower the standard deviation of the background, since these high pixels are higher than the statistics of the background, to a more accurate one.

The algorithm score is given as:

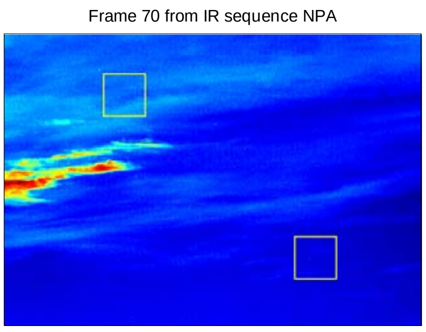
$$Score = \frac{Block\_SNR(i,j)_{(i,j)=TB} - E\left[\left\{Block\_SNR(i,j)\right\}_{(i,j)=NTB}\right]}{\sigma_{\left\{Block\_SNR(i,j)\right\}_{(i,j)=NTB}}} \tag{5.5}$$

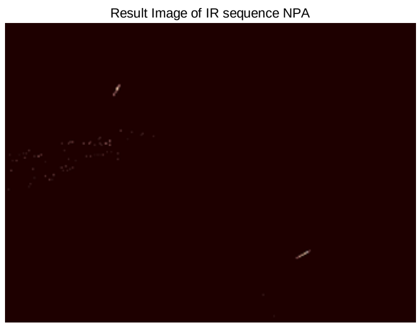
The score is calculated by subtracting between the target block *SNR* and the expectation value of the *SNR* of the non-target blocks, divided by the standard deviation of the non-target blocks *SNR*.

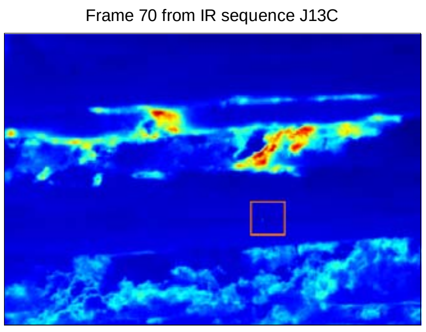
The final grade of the algorithm serves as a tool for comparison between the suggested temporal processing algorithm and other temporal processing algorithms which deal with the same problems. The grade evaluates the difference between the score of the block containing the target and the expected values of the rest of the blocks in the image, normalized by their standard deviation.

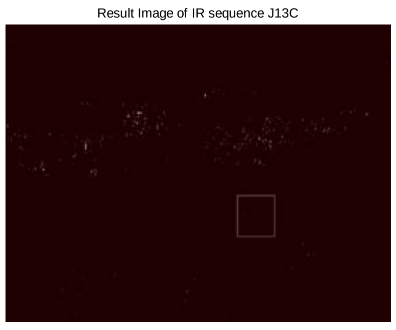
5.3.3 Real data results

The real-world IR image sequences described in the previous section were used to evaluate the temporal tracking algorithm. The algorithm output images are given in Figure 47 together with a representative frame from each sequence. The sequences were chosen to comprise both clutter and noise dominated scenes. The parameters of each simulation were chosen to be the optimal set, as will be explained in the following subsection. The target tracks are seen as brighter short stripes; in some cases, clutter leakage is also evident.









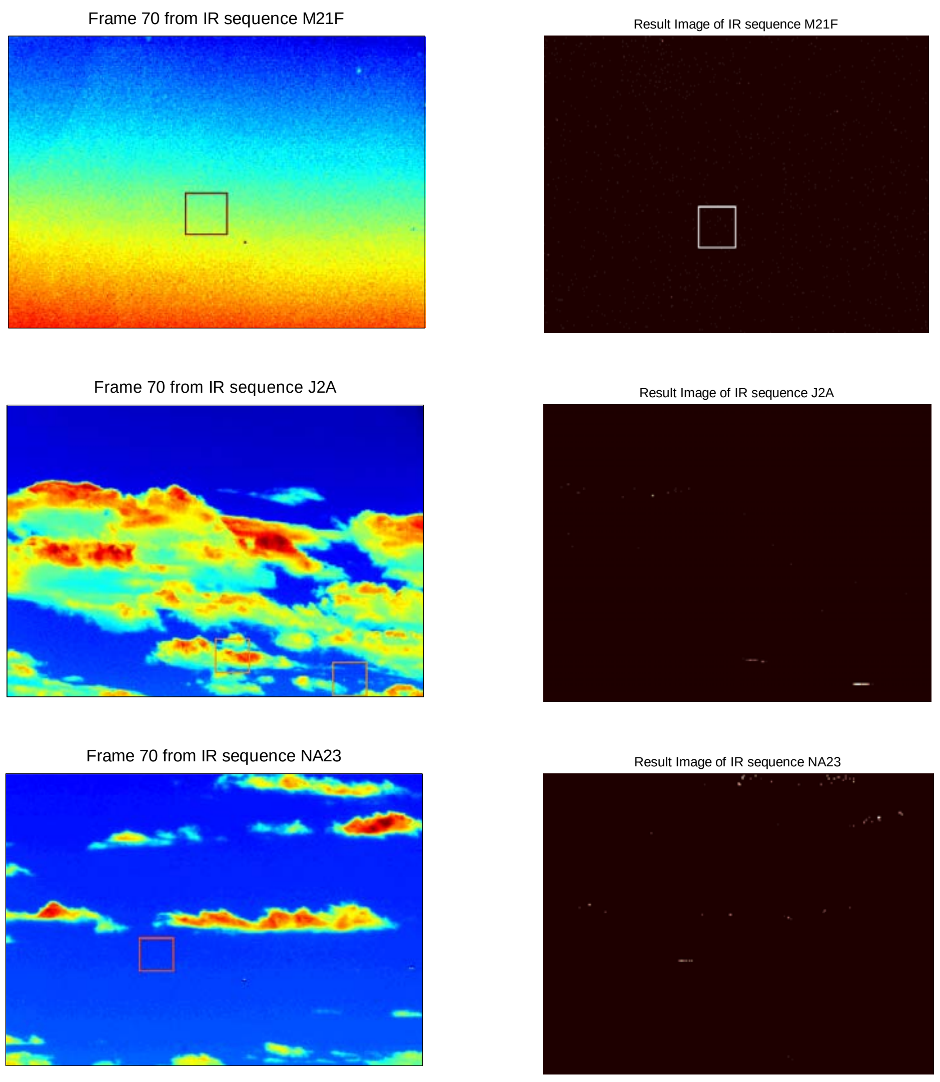
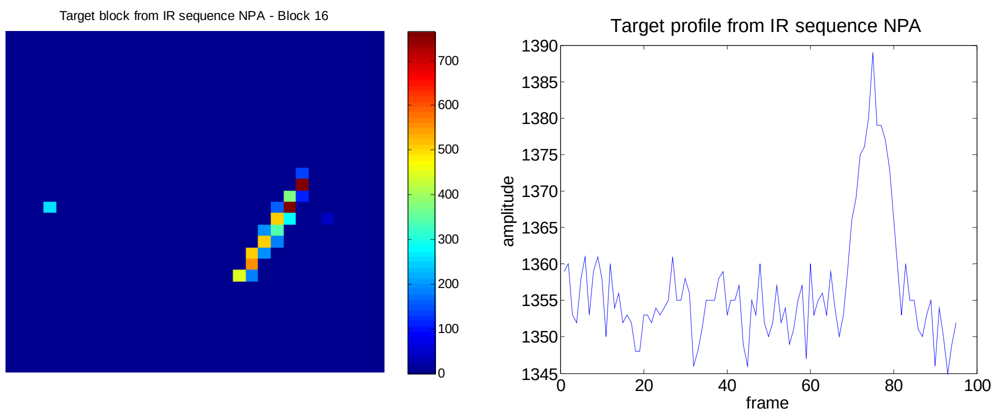
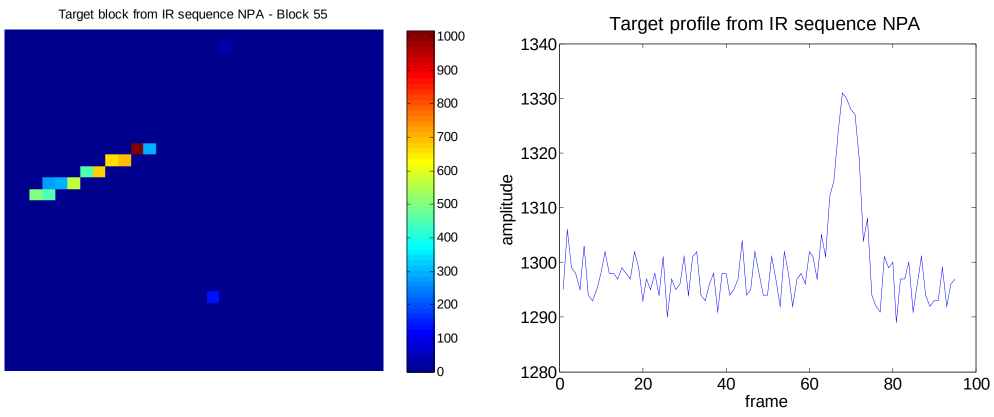


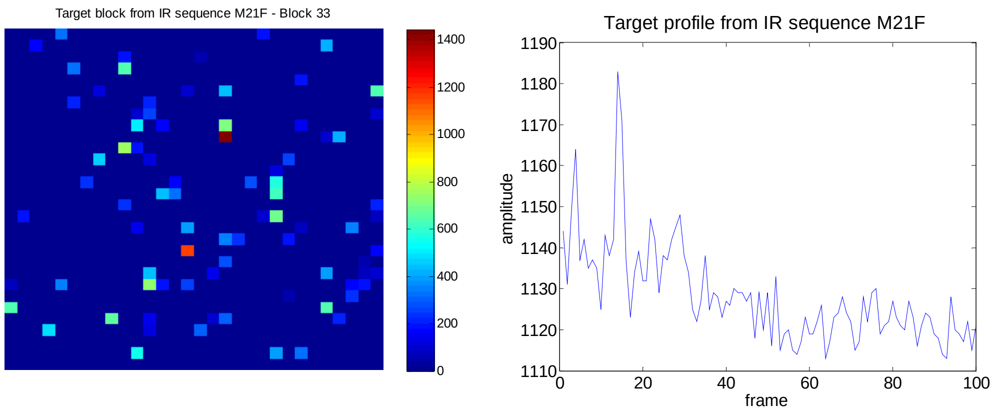
Figure 47 : <u>Output images of the temporal tracking algorithm</u>.

Figure 48 provides a closer view of the block containing the target of each output image together with a representative target temporal profile.

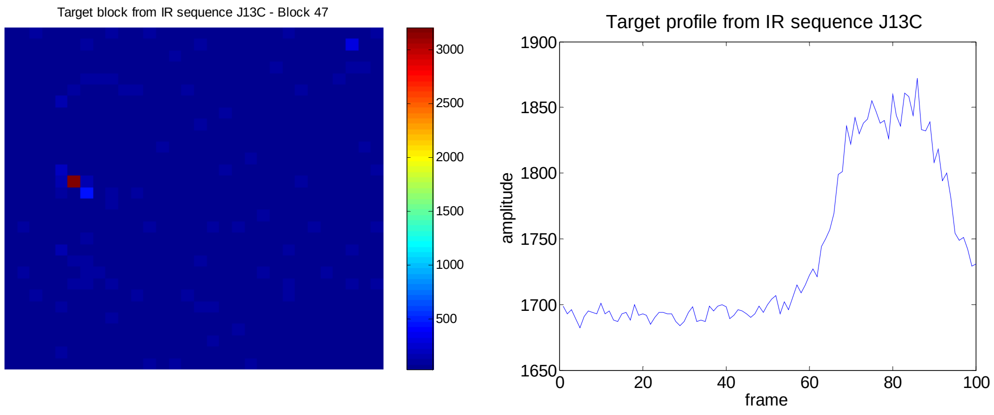




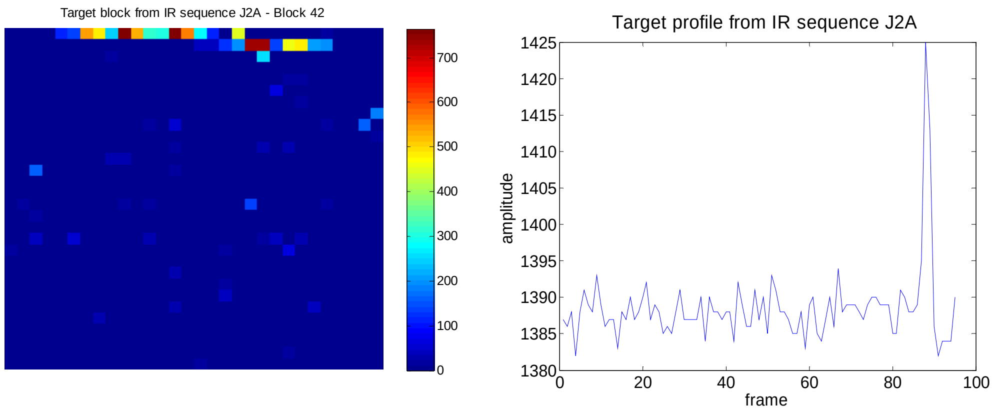
a. Target blocks and profiles from the IR sequence NPA

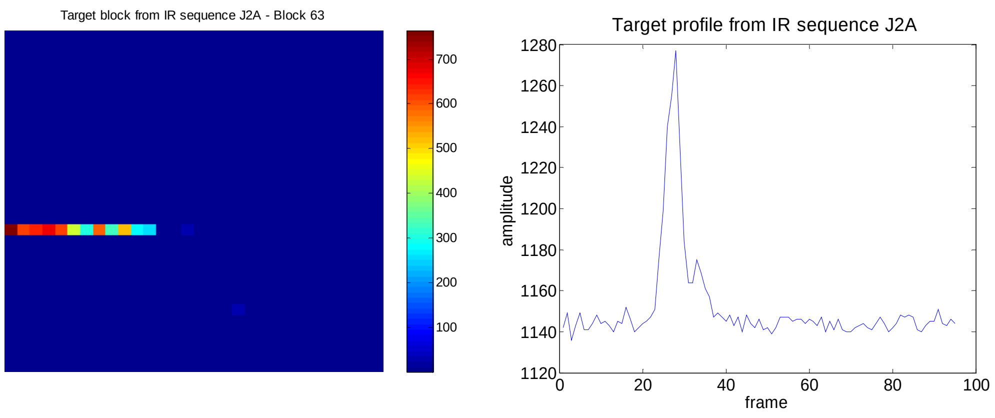


b. Target block and profile from the IR sequence M21F

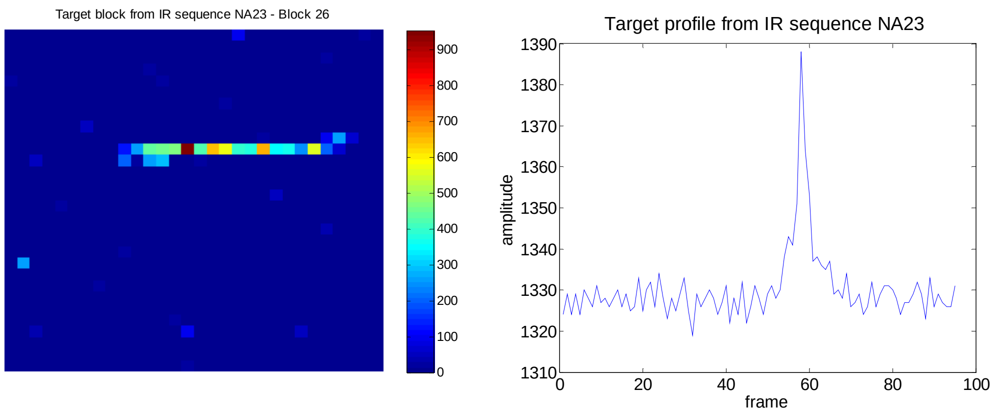


c. Target block and profile from the IR sequence J13C





d. Target blocks and profiles from the IR sequence J2A



e. Target block and profile from the IR sequence NA23

Figure 48: <u>Real data results</u>.

Visual evaluation of the results images presented in Figure 48 suggests that in terms of target pixels enhancement, the best results were obtained for the sequences J2A and NPA since the target trace there is stronger relative to the background. The M21F sequence for example has more noise in the background, although the target pixels are clearly visible. The metric defined for assessing the overall performance of the algorithm, which was given in equation (5.4) takes into consideration not only the target enhancement but the suppression of the background abilities of the algorithm as well. This is achieved by grading each block by a score which evaluates the difference between the maximal 5 values of the block and the block average values, normalized by the standard deviation. The goal of coarse is to have a target blocks with high scores and background blocks with low scores.

The purpose of the final grade of the algorithm, defined in equation (5.5) is to evaluate the separability between the target block/s and the background blocks. It evaluates the difference between the target block score (if there are more than one target, the mean of the target blocks) and the mean of the background blocks, normalized by the standard deviation of the background. The final grade obtained for each sequence is given in Table 4. The table also shows the grade of the two temporal processing algorithms presented in section 3.2.3 – the variance filter (VF) and the noise filter (NF) and previous temporal processing algorithm from reference [14] .

Image sequence name	*Grade*	*Ref. [14] Grade*	*VF Grade*	*NF Grade*
NPA	78.417	52.68	2.70	2.85
M21F	8.65	10.82	1.20	13.89
NA23	13.056	35.70	0.54	0.47

Table 4: Algorithm performance grade for each sequence.

The variety of the target scores for the different sequences can be understood by examining the amplitude of the maximal target peak, relative to the profiles average values.

Sequence name	Target Peak
NPA	~ 30
J2A	~ 140
M21F	~ 18
NA23	~ 60
J13C	~ 150

Table 5: Target maximal peak for the different IR sequences.

The sequences NPA and NA23 are inconsistent with each other – although the target in NPA seems weaker, its block score is higher than the one of NA23. This follows from the fact that in the NA23 scene strong clutter is present. As stated in the following subsection the optimal window size parameters are not optimal for the target block, but are fitted to the background as well. Choosing inappropriate window set lowers the target block score.

Therefore, although the target in NA23 is stronger than the one in NPA, the target block score received after the temporal processing is lower.

Finally the lowest target score is that of the sequence M21F. The target here is quite weak and the scene is noise dominated. Hence the low target amplitude and low target block score. The final algorithm is the lowest one as well.

5.3.4 The optimal window size

Choosing an appropriate set of parameters for the temporal algorithm is crucial for the detection capabilities of the system. In this section the dependence of the algorithm on the window sizes is evaluated on real data. The optimal set of parameters is obtained for each IR sequence.

The expected optimal window sizes depend on both the target temporal profile shape, mainly on the target's peak width (inverse proportional to the target's velocity) and on the background scene – the presence of clouds, their size and velocity, as stated in section 5.1.2.3.

In order to determine the optimal set of window sizes on a real data sequence, the algorithm was run on the sequence with various sets of parameters, the set which yielded the highest algorithm grade, defined in equation (5.3), was chosen.

Figure 49 shows the results of the simulation on the IR sequence NPA.

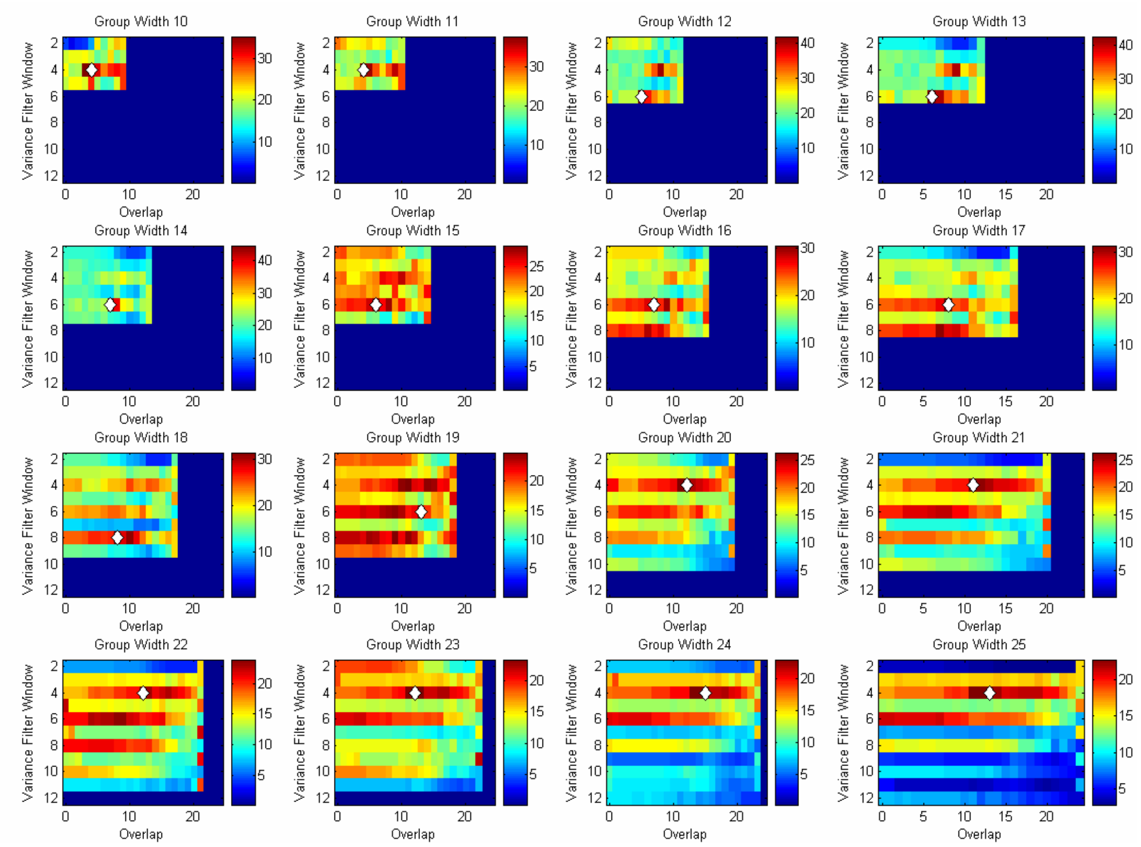


Figure 49: Algorithm score vs. window size sets for the IR sequence NPA.

The results show that the highest algorithm score was obtained for group width if size 14 samples, overlap of size 7 samples and variance window of size 6 samples at DC window of size 50 samples.

The final algorithm grade evaluates the difference between the target block score (if there are more than one targets, the relevant scores are averaged) and the mean background score, normalized by the standard deviation of the background blocks. Thus, the optimal window set will tend to maximize the target while minimizing the background and standard deviation scores.

Table 6 summarizes the optimal window sets for each IR sequence. The results of the algorithm performance for different window sizes for each IR sequence are given in the appendix of this work.

Sequence name	*Group Width*	*Overlap*	*Var. window*
NPA	14	7	6
M21F	19	18	8
NA23	16	5	4

Table 6: <u>Optimal window sets for each IR sequence</u>.

5.4 *Synthetic hypercube creation*

In order to properly evaluate the system, a real hyperspectral movie was needed. Since such data were not available, a simple procedure for creating a hyperspectral movie was developed. In order to make the movie as realistic as possible, each hypercube from the sequence is created from a real IR frame. This efficiently simulates the temporal characteristics of the evolving background.

The hypercube creation is performed in two steps. The first step is to create a "background" hypercube which contains cloud and sky mixed pixels. The second step is to implant into the "background" cube a synthetic target.

5.4.1 The background hypercube

The hypercube is created using the consecutive frame IR sequence presented in the previous section as a reference for obtaining the percent of sky and cloud signature of each pixel. The IR frame sequence containing F frames is denoted by $IR=\{IR(f)\}_{f=1,2,\ldots,F}$.

First arbitrary signatures representing the target, cloud and sky spectra are chosen; these signatures are denoted by **t**, **c** and **b** respectively. For S available spectral bands, the signatures are vectors of size Sx1. Examples of the signatures used in this work are given in Figure 50.

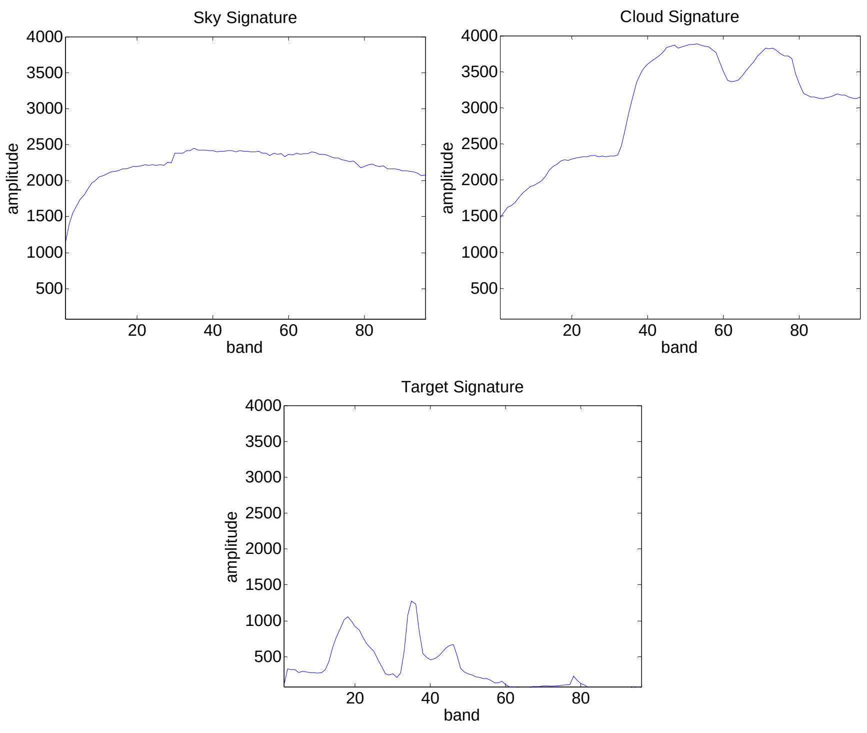


Figure 50: Spectral signatures used to represent the sky, cloud and target signatures.

The hypercube at each time index f is constructed by using the real infrared frame at time f, $IR(f)$, as the reference for the percentage of sky and cloud component for each pixel. The hypercube created at time index f is denoted by $D(f)$. $D_{ij}(f)$ is a vector of size Sx1 representing the spectral signature of the pixel coordinated at (i,j).

$$D_{ij}(f)=\left(\frac{IR_{ij}(f)-V_{min}}{V_{max}-V_{min}}\right)\cdot\mathbf{c}+\left(1-\frac{IR_{ij}(f)-V_{min}}{V_{max}-V_{min}}\right)\cdot\mathbf{b} \tag{5.6}$$

where $IR_{ij}(f)$ refers to the pixel value of the real infrared image at time f, V_{min} is the minimum pixel value of $IR(f)$($V_{min}=min_{ij}[IR(f)]$), V_{max} is the maximum pixel value of $IR(f)$ ($V_{max}=max_{ij}[IR(f)]$). For example, the pixel having the maximum value of $IR(f)$ will contain 100% cloud signature; the pixel with the minimum value of $IR(f)$ will contain 100% sky signature. Pixels with intermediate values will contain mixtures of sky and cloud signatures in proportion matching the pixel's value and normalized to a fractional value between 0-1.

The hypercube movie is comprised of F hypercubes; it is denoted by $M=\{D(f)\}_{f=1,2,\ldots,F}$.

5.4.2 Target implantation

The synthetic target is characterized by the following parameters:

1. Spatial shape, denoted by $s(x,y)$, is dependent on the camera's point spread function and the actual target shape. A possible choice might be a two-dimensional gauss function. Figure 51 shows an example of a two dimensional target having a half-period sine shape in both x and y directions.

2. Temporal behavior, where the velocity is measured in pixels/frame, denoted by v_x and v_y.

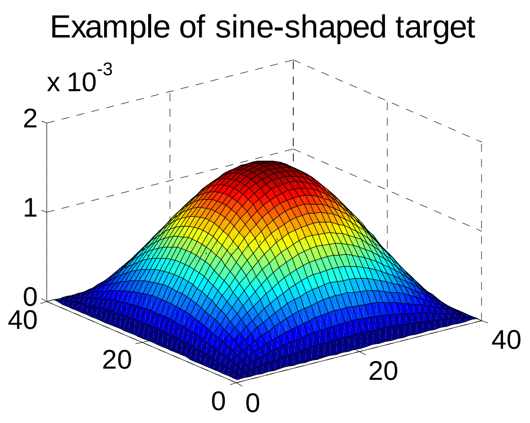


Figure 51: A two dimensional sine shaped target.

The fraction of the target in a pixel affected by it is given by:

(5.7) $$p(f) = \int_{x_0}^{x_0+\Delta} \int_{y_0}^{y_0+\Delta} s(x,y)dxdy$$

where $(\{x_0,x_0+\Delta\},\{y_0,y_0+\Delta\})$ denotes the pixel's spatial boundaries.

There are two methods for target implantation [15] .

(5.8) $$y = pt + x$$

(5.9) $$y = pt + (1-p)x$$

After creating the sequence *$\{D(f)\}_{f=1,2,\ldots,F}$* the target is implanted into the relevant pixels; equations **(5.8)** and **(5.9)** are modified to:

(5.10) $$D_{ij}(f) = p(f)Gt + (1-p(f))D^1{}_{ij}(f)$$

(5.11) $$D_{ij}(f) = p(f)Gt + D^1{}_{ij}(f)$$

where $D^1{}_{ij}(f)$ represents the appropriate mixed pixel from the background cube created at the first step and G is a normalizing constant factor which acts as the target's base gain.

The indices of the pixels affected by the moving target change according to the target's spatial shape, velocity and direction. Assuming the target's start position is at coordinates (x_0,y_0), at frame f the target's position will be $(x_0 + f\cdot v_x, y_0 + f\cdot v_y)$. Figure 52 illustrates the movement of the target having a half-period sine spatial shape, a width of 2x2 pixels, a horizontal velocity of 0.5 pixel/frame, and a vertical velocity of 0.25 pixel/frame.

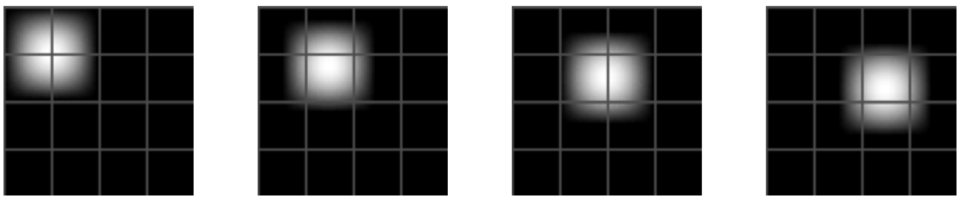


Figure 52: Illustration of 2x2 pixels target moving at v_x=0.5 pixel/frame, v_y=0.25 pixel/frame.

5.4.3 Noise addition

The noise added to each synthetic hypercube is White Gaussian Noise. In order to keep the noise proportional to the relevant spectral signature magnitude, the noise variance is dynamically determined using the following general description:

(5.12)

$$\sigma^2 = NoiseFactor * \max(signature)^2$$

where *NoiseFactor* is a constant with value between 0-1. This is consistent with IR imagery where the noise power is proportional to the signal power (brightness of the object).

In addition, noise is added to the synthetic target temporal movement – at each new frame, the effective step size of the synthetic target is calculated using the following:

(5.13)

$$\begin{aligned} &StepSize = \max(TheoreticalStepSize + n, 1) \\ &n \sim N(0, (NoiseFactor * TheoreticalStepSize)^2) \end{aligned}$$

where *NoiseFactor* is a constant between 0-1, *TheoreticalStepSize* is a constant, calculated from the target velocity, and *n* is the noise added (or subtracted) which has zero mean Gaussian Distribution with standard deviation proportional to the theoretical step size. In order to keep the motion model physically consistent, the step size must be a positive number.

5.4.4 Examples of hyperspectral movie

Figure 54 shows examples of several bands of the hypercube with a synthetic target implanted at the upper left region of the cube. This hypercube was created without the addition of noise to the spectral signatures. The IR image used for this hypercube is shown in Figure 53.

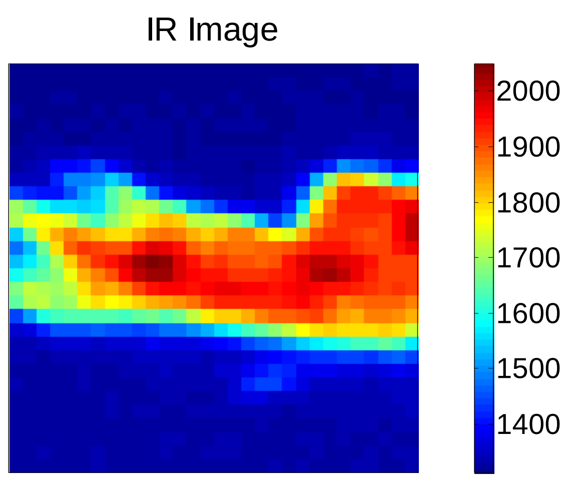


Figure 53: Example of IR block used for creation of hypercube.

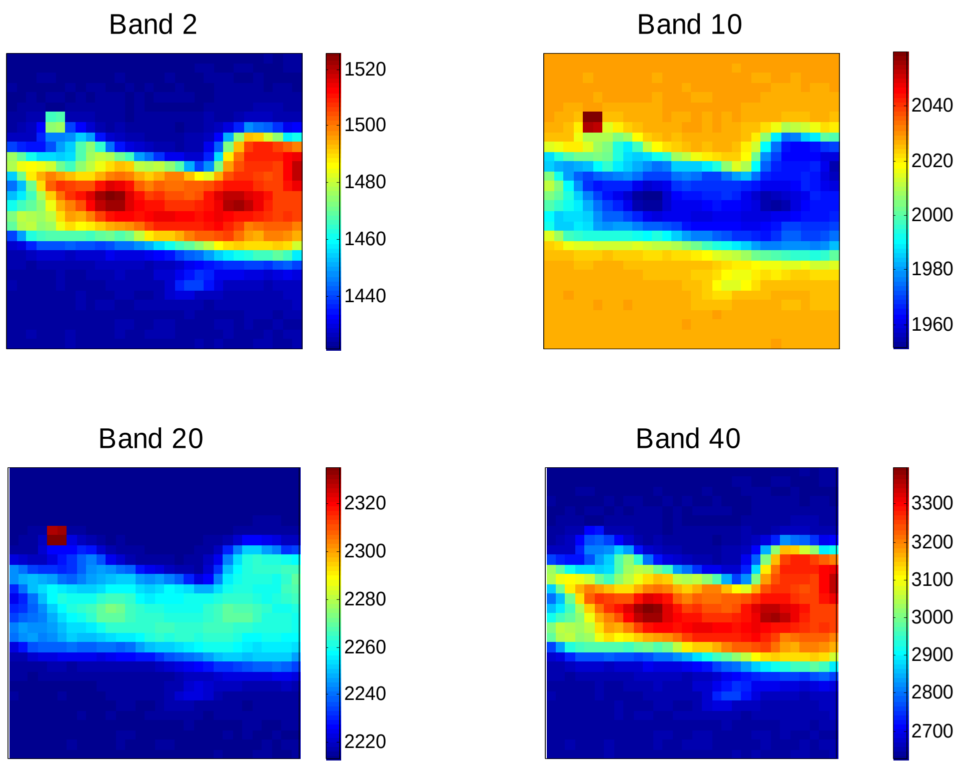


Figure 54: Bands 2, 10, 20 and 40 of synthetic hypercube.

Figure 55 shows the same hypercube's bands, but with the addition of white Gaussian noise to each spectral signature. The noise variance is set to be $[0.1 \cdot max(signature)]^2$ (noise factor of 0.1).

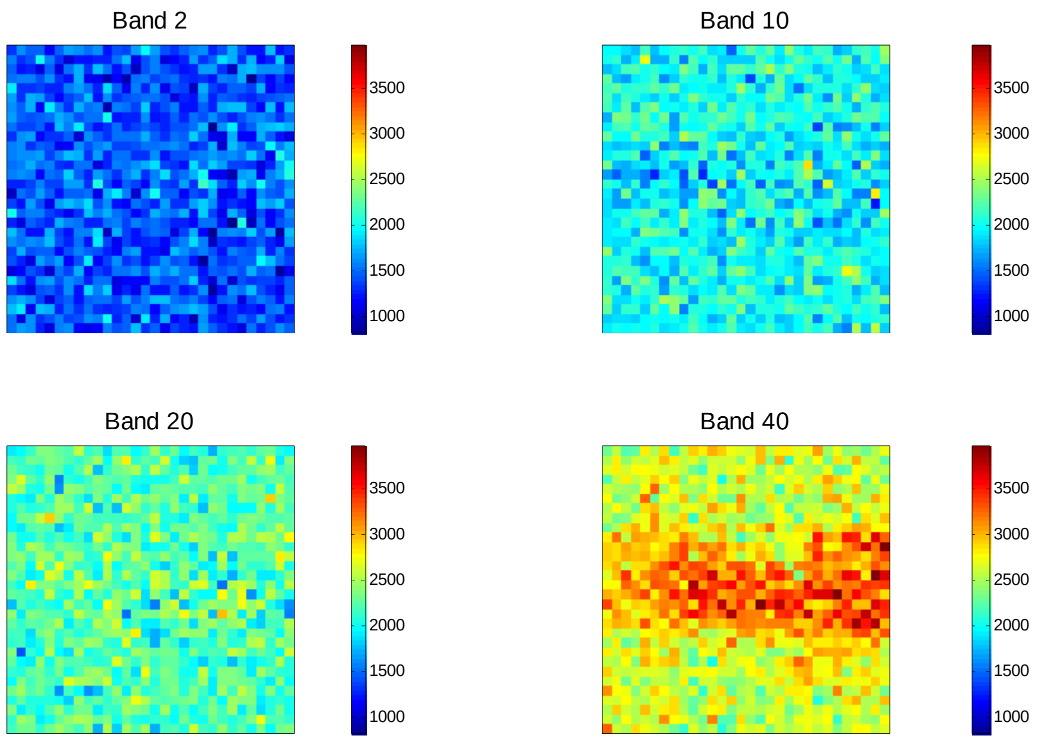


Figure 55: <u>Bands 2, 10, 20 and 40 of synthetic hypercube with addition of spectral noise</u>.

Figure 54 and Figure 55 show several spectral bands of a hypercube at a single time instance. A spectrogram of single spatial location (a pixel) of the hypercube can be plotted in which the spectral variations as a function of time can be observed for the given pixel. Figure 56 shows a spectrogram from a pixel taken from the hypercube based on the IR image given in Figure 53. The IR image obviously contains drifting clutter. The clutter impact on the temporal spectral changes of the pixel is evident from the spectrogram. The signatures used for this hypercube were given in Figure 50. The first spectrogram doesn’t contain target. The spectrogram shows higher amplitude values for spectral bands 35-80 with low values at bands 0-10 at time indices 0-40. This is consistent with the spectral signature representing the cloud, which has peaks at spectral bands 40-80 besides local minima around band 60. Thus, the pixel presented was traversed by a clutter approximately at times 0-60. The sky spectral signature appears to be constant with the time at time indices 65-95. Its features fro bands 1-35 are similar to the sky signature, the major difference is evident for bands 35-95 where the sky signature has lower and more unified levels of amplitude.

The lower spectrogram shows the same pixel but with target implanted into it. The target influence can be seen at time indices 5-15, causing amplitude increase around bands 20, 35 and 45. Compared to the sky signature, the target signature has higher amplitude at the lower spectral bands. Since the method of target implantation is additive (and not by replacing the relevant portion of the signature), the spectrum of the pixel affected by both clutter and target is actually the sum of the two signatures.

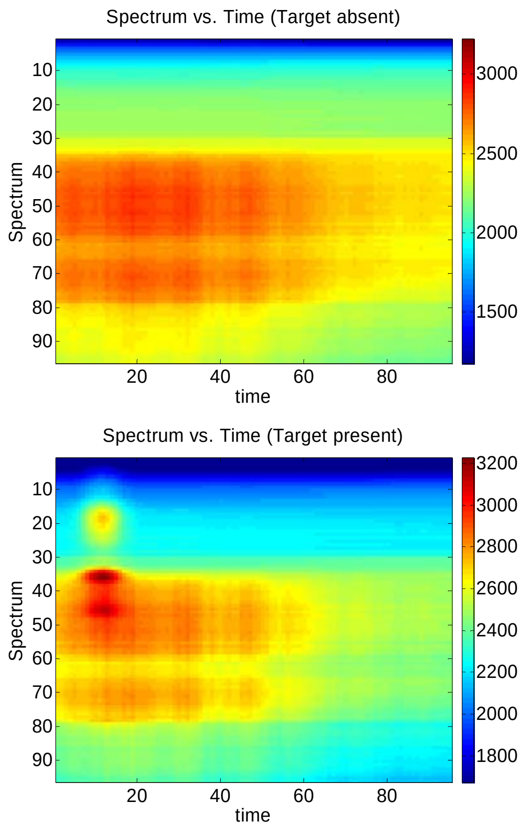


Figure 56: <u>Spectrogram of pixel affected by clutter with and without target implanted in it</u>.

Figure 57 shows the same spectrogram of the pixel with target implanted into it with addition of white Gaussian noise to each spectral signature. The noise variance is set to be [*0.1*max(signature)*]^2 (noise factor of 0.1).

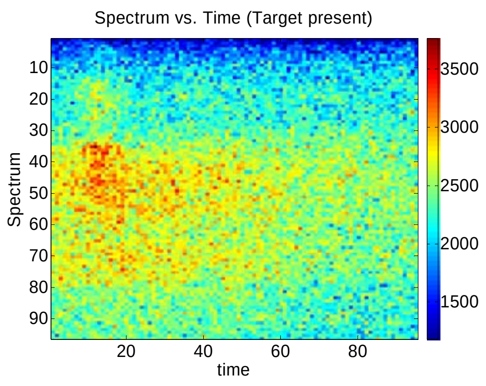


Figure 57: <u>Spectrogram of pixel affected by clutter and target with addition of spectral noise</u>.

5.5 Evaluation of the complete system on real data

The hyperspectral movie is created as described in the previous section. The movie consists of a sequence of 30x30x96 cubes (width x height x bands). A synthetic target is implanted into the sequence. The target is sine shaped 2x2 pixels wide and has a horizontal and vertical velocity of 0.1[*ppf*]. White Gaussian noise is added to each spectral signature; the noise variance is set to be $[0.1*Max(signature)]^2$.

The movie is then input into our system. The third reduction test given in section 4.2 (using the match filter with the estimated covariance matrix) is applied and used for the first stage processing of each hypercube, and the temporal processing algorithm described in section 4.3 for the target detection at the second stage. The output of that stage is input into the DPA. At the end the last processed frame is taken and the highest pixel is declared as 'target' and its track is found. In this section, we will compare Test 3, and two new tests, Test 4 and Test 5, defined in the next sub-section, will be used.

5.5.1 Metrics definitions

The metric at 3.2.3.2 defines the score of any two-dimensional image:

The block SNR is given as:

$$Block\_SNR(i,j) = \frac{E\left[v_{i,j} \in M\right] - E\left[v_{i,j} \notin M\right]}{\sigma_{v_{i,j}}} \tag{5.14}$$

where $v_{i,j}$ is the set of pixels belonging to the *(i,j)*th block, *M* is a set containing the five pixels with the highest gray level in that block, and σ is the standard deviation of the block pixels. The formula performs a subtraction between the expectation value of the highest pixels (target) and the expectation value of the rest of the pixels (background), divided by the standard deviation of block pixels. Since the probability matrices introduce the influence of target pixels on adjacent pixels, these influenced pixels might accumulate higher values than unaffected pixels (background); we will regard them as target pixels. This might lower the expectation value of the target, but will lower the standard deviation of the background, since these high pixels are higher than the statistics of the background, to a more accurate one.

The algorithm score is given as:

(5.15)
$$Score = \frac{Block\_SNR(i,j)_{(i,j)=TB} - E\left[\left\{Block\_SNR(i,j)\right\}_{(i,j)=NTB}\right]}{\sigma_{\left\{Block\_SNR(i,j)\right\}_{(i,j)=NTB}}}$$

Three tests based on this metric are further defined. Test 3 uses a *MF* for the cube collapsing. Test 4 uses a *MF* detector as the input of the temporal processing. Test 5 adds to Test 4 the Dynamic Programming algorithm. These tests were created to evaluate the effect of the *IR* tracking algorithms on the overall score of Hyperspectral tracking system.

The *MF* and the temporal processing create images with pixel scores according to their likelihood of being a target, whereas the *DPA* accumulates the scores of pixels according to the probability of the path going thru them to be the target's path.

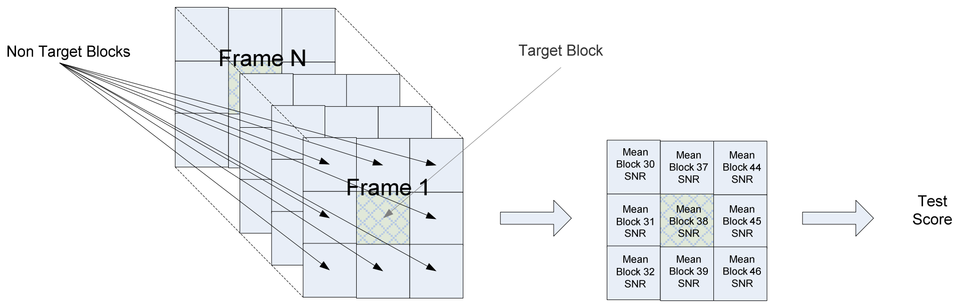


Figure 58: Metric definition.

5.6 Discussion of results obtain on real data

This section presents the results of applying the complete system algorithm on hyperspectral movie based on blocks from the real *IR* sequence *NA23*. The algorithm was run on target block and the eight surrounding background blocks. The blocks were chosen to represent different scenes, which might be roughly categorized as 'clear sky' scene – contains only sky, 'weak clutter' – contains partial weak clutter (with low to medium *IR* amplitude) and 'strong clutter' – contains partial clutter with high IR amplitudes. The parameters of the simulation are summarized in Table 7.

Table 8 - Table 13 presents the results for three different background hyper-cubes, based on blocks from the real *IR* sequence *NA23*, Figure 59 shows a single frame from the sequence, divided into labeled blocks.

Parameter	***Value***
Hyperspectral movie parameters	
Movie length	95 frames
Spectral signatures	Identical to the ones presented in Figure 50
Block size	30x30 pixels
Number of spectral bands	100 bands
IR source sequence	NA23
Noise added	WGN, noise factor of 0.05 (std = noise factor * 0.05)
Synthetic target properties	
Spatial shape	half sine, 2x2 pixels, integral of the spatial distribution normalized to 0.5
Horizontal velocity	1/8 pixels/frame
Vertical velocity	1/8 pixels/frame
Velocity error	as described in section 5.4.3, noise factor of 0.25
Hyperspectral cube reduction	
Reduction Filter	Test 1, Test 2, Test 3
Target Block	37 (scarce clutter), 38 (sky only), 39 (strong clutter)
Target factor	10, 20, 40, 60, 80, 100, 500, 1000
Temporal processing parameters	
Sub profile length	15 samples
Overlap	10 samples
DC window	50 samples
DC step size	15 samples
Variance window	4 samples
DPA	
EMC (*b*)	[0…6] for *g* = 0, [0…1] for *g* = 1
a	1
p	0.5
y	24

Table 7: Complete system simulation parameters.

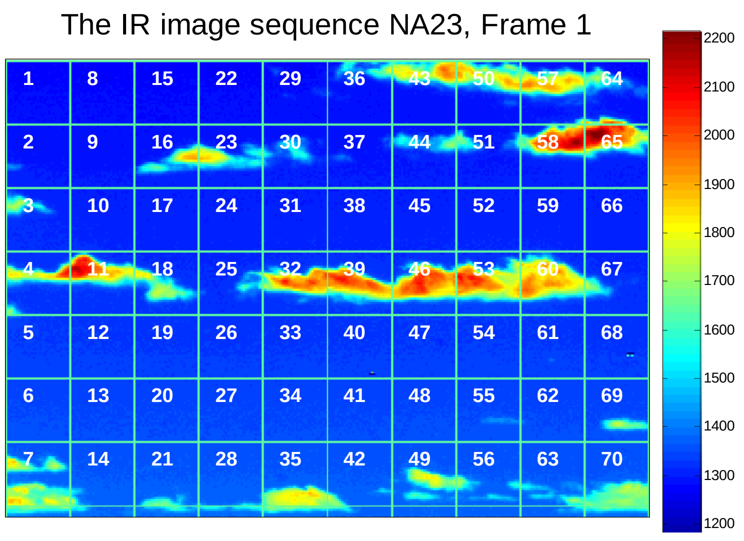


Figure 59 : Single frame of IR sequence NA23 with labeled blocks division.

Block	Scene Description	Test 1 – Spectral Average			
	Target factor	20	40	60	80
37	Partial weak clutter	1.7622	1.7542	1.7414	1.7204
38	Mainly sky	-1.3432	-1.0566	-0.5864	-0.0326
39	Mainly strong clutter	-0.0362	-0.0371	-0.0354	-0.0392
mean		0.1276	0.2201	0.3732	0.5495

Table 8: <u>Test 1 system evaluation results</u>.

Block	Scene Description	Test 2 – Scalar Product			
	Target factor	20	40	60	80
37	Partial weak clutter	1.3957	1.3572	1.3213	1.3103
38	Mainly sky	-1.2325	-0.8918	-0.3695	0.2114
39	Mainly strong clutter	-1.2276	-1.2379	-1.1656	-1.0806
mean		-0.3548	-0.2575	-0.0712	0.1470

Table 9: <u>Test 2 system evaluation results</u>.

Block	Scene Description	Test 3 - MF			
	Target factor	20	40	60	80
37	Partial weak clutter	2.3023	2.2752	2.2497	2.2419
38	Mainly sky	-0.2196	0.2049	0.7912	1.3592
39	Mainly strong clutter	-0.2132	-0.2266	-0.1337	-0.0263
mean		0.62317	0.7512	0.9690	1.1916

Table 10: <u>Test 3 system evaluation results</u>.

Block	Scene Description	Test 4 - MF&TP			
	Target factor	20	40	60	80
37	Partial weak clutter	2.9422	3.6824	3.7198	3.7072
38	Mainly sky	3.2252	3.6421	3.7222	3.7188
39	Mainly strong clutter	0.5539	1.6206	3.1649	3.4353
mean		2.2404	2.9817	3.5356	3.6204

Table 11: <u>Test 4 system evaluation results</u>.

Block	Scene Description	Test 5 - MF, TP & DPA			
	Target factor	20	40	60	80
37	Partial weak clutter	3.6473 (1)	3.8205 (2)	3.8248 (3)	3.8448 (4)
38	Mainly sky	3.7568 (5)	3.8255 (2)	3.8318 (2)	3.8365 (2)
39	Mainly strong clutter	2.0955 (6)	2.3476 (6)	3.7458 (7)	3.7171 (8)
mean		3.1665	3.3312	3.8008	3.7994

Table 12: Test 5 system evaluaetion results for *g*=0, *Max_Numbers*=3.

Block	Scene Description	Test 5 - MF, TP & DPA			
	Target factor	20	40	60	80
37	Partial weak clutter	3.4213 (9)	3.7624 (2)	3.8384 (2)	3.8063 (2)
38	Mainly sky	3.6276 (10)	3.8049 (11)	3.8492 (2)	3.8350 (2)
39	Mainly strong clutter	0.8168 (6)	0.2714 (6)	3.4032 (12)	3.2192 (2)
mean		2.6219	2.6128	3.6969	3.6201

Table 13: Test 5 system evaluaetion results for *g*=1, *Max_Numbers*=1.

(1) - best tracking occurs for b≥7

(2) - tracking occurs for all *b*'s

(3) - best tracking occurs for b≥3.4

(4) - best tracks occur for 4≤*b*≤5.4

(5) - target is tracked for *b*≥1.4.

(6) - no tracking occurs.

(7) - target is tracked for 0≤*b*≤1

(8) - tracking occurs for: (1) 0≤*b*≤4.2, (2) 5.4≤*b*≤5.6, (3) 6≤*b*

(9) - Tracking occurs for 0≤*b*≤0.3, *b*=0.6

(10) - Tracking occurs for 0.6≤*b*≤1

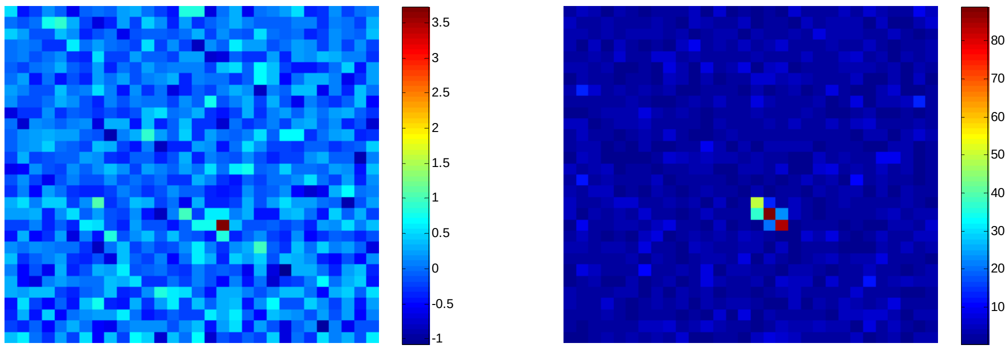
(11) - Tracking occurs for 0≤*b*≤0.9

(12) - Tracking occurs for 0≤*b*≤0.5

Previous research, [14] , has shown that Test 1 allows a rough assessment of the target pixels relative amplitude compared to their background. The low values of the results indicated that the implantation of the target and taking the maximal score without any processing is not sufficient for detection; in other words, the implantation method does not allow “easy” detection. The highest values of Test 1 were in the weak clutter scenes which is reasonable since the implantation method is additive, and, in weak clutter surroundings, the amplitude levels are obviously higher than clear sky scenes. Comparison between Test 1 and Test 2 allowed us to estimate the improvement of using a primitive hyperspectral processing – simply taking the average of all the bands. Although this brought improvement in sky or weak clutter scenes, it had negative impact on strong clutter scenes which proved that simply averaging the bands is disastrous for certain sets of spectral signatures and cannot be used as a detection method by itself. Thus the focus of the discussion should be the use of “smart” hyperspectral processing only (Test 3), the use of hyperspectral processing and temporal processing (Test 4) the use of hyperspectral processing, temporal processing and DPA (Test 5).The results in [14] have shown that in all of the cases, there is an obvious advantage to using both hyperspectral detection (Matched Filter) and temporal processing (Test 4 vs. Tests 1-3).

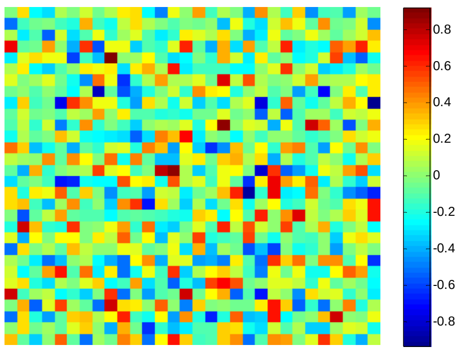
When the target is implanted in clear sky scenes, the use of temporal processing significantly improves the performance compared to use of hyperspectral detection only. In most cases the use of the Matched Filter compared to simple averaging was clearly advantageous; the exception being block 31 for which the performance was similar for both of the techniques. This can be attributed to the relative "easiness" of detection in this kind of scene and the fact that the high level of noise might cause a disadvantage to the Matched Filter and an advantage to the averaging filter. When weak clutter was present, the temporal processing combined with Matched Filter detector was always better than temporal processing only which in turn is better to hyperspectral detection only.

Preliminary runs have been done for target factor 1000, 500, 100 and 10. A valid range of target factor was needed to be found in order to define the boundaries of the full system. The results have shown that target factor of 100 or above are 'too easy' to use a detection algorithm, whereas target factor of 10 is 'too hard' for the full system to detect and track a target. Since that is the case, tryouts of the following values of target factor have been applied: 20 – 80 at steps of 20.

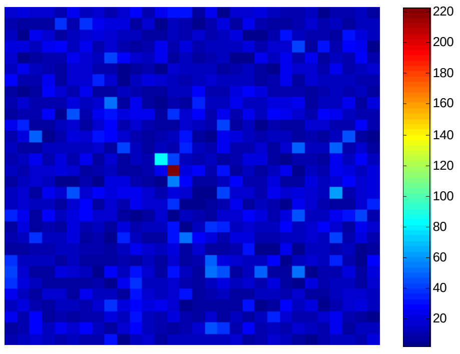


(a) (b)

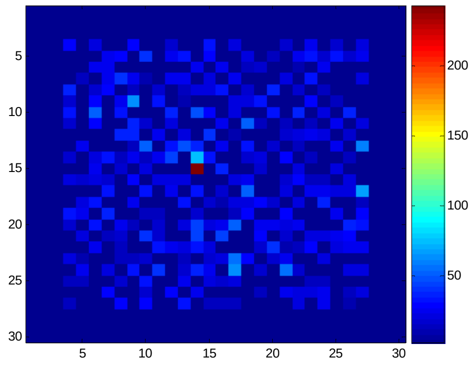
Figure 60: <u>Block 38, Target Factor 100: (a) Single Frame after MF (Test 3), (b) Single frame after MF and TP (Test 4)</u>.



(a)



(b)



(c)

Figure 61: <u>Block 38, Target Factor 20: (a) Single Frame after MF (Test 3 - Frame 64[3]), (b) Single frame after MF and TP (Test 4 - Frame 12), (c) Single frame after MF, TP and DPA (Test 5 - Frame 12)</u>.

[3] As described at the Table 7, "sub profile length is equal to 15, overlap is equal to 10" have been chosen: taking frame 12 after the *TF* is equivalent to taking a frame between 55-70 in the sequence after MF.

6 Summary and Conclusions

This report consists of the culmination of three years of work. It is interesting to consider the steps which have been taken to produce these results. If we look at the System Diagram in Figure 6, we find that the following work has been performed:

1. The construction of a database of hyperspectral movies. The fundamental broadband moview were provided by the Air Force Research Laboratory; we designed a way to convert them to hyperspectral movies.

2. The matched filter stage was taken from the standard literature. Future work will involve using somewhat more sophisticated algorithms to be used to process the spectral domain.

3. The temporal stage was designed and tested on the AFRL movies. This work has been documented extensively in Ref. [14] .

4. The Dynamic Programming Algorithm has been built and tested.

5. An alternative approach for the temporal and DPA stages has been considered; this involved the Kalman Filter and will be documented in a separate report (see appendix).

6. Metrics were developed to test the effect of each stage on our target acquisition capability.

We overall conclude that each stage enhanced our target acquisition capability.

One important note concerning this work is that it is modular. The modules for spectral processing, temporal processing, DPA processing and even our method of evaluating algorithms can be individually replaced and tested. Our work can thus easily be extended to new developments.

It has been a pleasure to work on this project for the last three years; we thank AFOSR for the opportunity.

7 Bibliography

[1] G. Shaw, D. Manolakis, "Signal Processing for Hyperspectral Image Exploitation", *IEEE Signal Processing Magazine*, Vol. 19, Jan 2002.

[2] D. Landgrebe, “Hyperspectral Image Data Analysis”, *IEEE Signal Processing Magazine,* Vol. 19, Jan 2002.

[3] G. Shaw, H. Burke, “Spectral Imaging for Remote Sensing”, *Lincoln Laboratory Journal*, Vol. 14, Num. 1, 2003.

[4] C.E. Caefer, S.R. Rotman, J. Silverman, P.W. Yip “Algorithms for point target detection in hyperspectral imagery” *Proc. SPIE,* Vol. 4816, 2002.

[5] D. Manolakis, G. Shaw, “Detection Algorithms for Hyperspectral Imaging Applications”, *IEEE Signal Processing Magazine*, Vol. 19, Jan 2002.

[6] N. Keshava, J. F. Mustard, “Spectral Unmixing”, *IEEE Signal Processing Magazine*, Vol. 19, Jan 2002.

[7] D. W. J. Stein, S. G. Beaven, L. E. Hoff, E. M. Winter, A. P. Schaum, A. D. Stocker, “Anomaly Detection from Hyperspectral Imagery”, *IEEE Signal Processing Magazine*, Vol. 19, Jan 2002.

[8] C.E. Caefer, J.M. Mooney, J. Silverman, “Point target detection in consecutive frame staring IR imagery with evolving cloud clutter”, *Proc. SPIE*, Vol. 2561, 1995.

[9] C.E. Caefer, J. Silverman, J.M. Mooney, S. DiSalvo, R.W. Taylor, “Temporal filtering for point target detection in staring IR imagery: I. damped sinusoid filters”, *Proc. SPIE*, Vol. 3373, 1998.

[10] C.E. Caefer, J. Silverman, J.M. Mooney, “Otimization of Point Target Tracking Filters”, *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 36, No. 1, 2000.

[11] J. Silverman, C.E. Caefer, S. DiSalvo, V.E. Vickers, “Temporal filtering for point target detection in staring IR imagery: II. Recursive variance filter”, *Proc. SPIE,* Vol. 3373, 1998.

[12] J. Silverman, C.E. Caefer, J.M. Mooney, “Performance metrics for point target detection in consecutive frame IR imagery”, *Proc. SPIE,* Vol. 2561, 1995.

[13] S. Chatterjee, A. S. Hadi, “Influential Observations, High Leverage Points, and Outliers in Linear Regression”, *Statistical Science*, Vol. 1, No 3, 1986.

[14] L. Varsano, I. Yatskaer, S.R. Rotman, “Temporal target tracking in hyperspectral images”, accepted for publication in Optical Engineering.

[15] M. Cohen, Y. Simson, and S.R. Rotman, “Point target detection in segmented hyperspectral images”, *Proc. SPIE*, Vol. 5546, Imaging Spectrometry X; Sylvia S. Shen, Ed., 2004.

[16] http://www.sn.afrl.af.mil/pages/SNH/ir_sensor_branch/sequences.html.

[17] Y. Barniv, “Dynamic Programming Solution for Detecting Dim Moving Targets”, *Aerospace and Electronic Systems*, Proc. IEEE 29, No.1, pp. 44-56, 1985.

[18] Y. Barniv, O. Kella, “Dynamic Programming Solution for Detecting Dim Moving Targets Part II: Analysis”, *Aerospace and Electronic Systems*, Proc. IEEE 23, No.6, pp. 776-788, 1987.

[19] J. Arnold, H. Pasternack, “Detection and tracking of low observable targets through dynamic programming”, *Proc. SPIE*, Vol. 1305, Signal and Data Processing of Small Targets 1990/207.

[20] R. Succary, H. Kalmanovitch, Y. Shurnik, Y. Cohen, E. Cohen, S. R. Rotman, “Point Target Detection”, *Infrared Technology and Applications XXVII*, Proc. SPIE 3820, pp. 671-675, 2003.

[21] R. Succary, A. Cohen, P. Yaractzi, S. R. Rotman, “A Dynamic Programming Algorithm for Point Target Detection: Practical Parameters for DPA” *Signal and Data Processing of Small Targets*, Proc. SPIE 4473, pp. 96-100, 2001.

[22] O. Raviv and S.R. Rotman “An improved filter for point target detection in multi-dimensional imagery”, *Imaging Spectrometry IX*, eds. S.S. Shen and P.E. Lewis, Vol. 5159, pp. 32-40, 2003.

Appendix

A Practical Method of Tracking a Single Target in Cluttered Hyperspectral Data

Y. Simson and S.R. Rotman

A Practical Method of Tracking a Single Target in Cluttered Hyperspectral Data

Y. Simson and S.R. Rotman

August 22, 2006

Contents

List of Tables

1 Introduction

Tracking a dim target in clutter is a well known problem in the scientific literature. There are two main approaches. The first is to initially threshold every image and then make a decision as to which sequence of measurements is the most likely track. Among the various methods in this category are the IMM (Interacting Multiple Models), GPBF (Generalized Pseudo Bayesian Filter) and MHT (Multiple Hypothesis Testing). The second approach is to delay the thresholding until the end of the tracking process. This is usually is carried out by using the DPA (Dynamic Programming Algorithm). Both approaches have advantages and disadvantages. The first approach (IMM,MHT) is better suited to real time applications whereas the second approach (DPA) is considered better at tracking targets with a low SNR. In this paper the former approach is taken; specifically we implement the IMM algorithm.

When using the IMM algorithm there always is a chance of false alarms. This makes the problem a lot more complex. Not only is there a challenge of estimating the real position and speed but in addition one has to decide: out of a number of possible targets, which is the real target or whether a target has even been detected at all. There are two different types of clutter. The first is *residual clutter*. The second is termed *persistent clutter*. The two basic assumptions about *residual clutter* are that the clutter or false measurements are independent across time and uniformly spatially distributed. With these assumptions there are ways to model the *residual clutter*. The *persistent clutter* can either be canceled by pre-processing or by modeling it and tracking it. For example false tracks generated by edges between different regions in an image. If the speed of the background is known, then these false tracks will have the same speed and direction, therefore can be filtered out. The approach adopted here for dealing with the *residual clutter* is that of Y. Bar Shalom in [7] and [5] namely the PDAF (Probabilistic Data Association Filter) algorithm. When combined with the IMM the resulting algorithm is termed IMM-PDAF. This problem and the problem of initiating the track sequence will be dealt with after treating the simpler problem of a single target in residual clutter.

2 Tracking Overview

Both tracking methods (DPA/IMM-PDAF) have the some main steps in common. However the steps are carried out in a different order. For the first approach the IMM-PDAF algorithm was chosen for its relative simplicity yet good results in comparison to the GPBF and the MHT. The principle steps are:

- Pre-processing: Usually done by linear filtering, ordered statistics filters.

- Thresholding the image.
- IMM-PDAF: The algorithm will be explained later in detail.
- Thresholding the surviving tracks.

For the DPA the basic steps are:

- Pre-processing: Usually done by linear filtering or ordered statistics filters.
- DPA: The Viterbi algorithm.
- Thresholding the results.

2.1 Pre-processing

In this research two methods for pre-processing are considered; linear and ordered statistics filters. The assumption behind both approaches is that the target is added in the following manner to the original picture:

$$u(m,n)' = u(m,n) + t \tag{1}$$

where $u(m,n)$ is the original pixel at coordinates (m,n). The modified pixel is $u(m,n)'$, which is due to the added target t. The purpose of the pre-processing stage is to extract pixels with the added target t. Ideally if the intensity of the original pixel is known then there is no problem. Simply subtract $u(m,n)$ from $u(m,n)'$: then one gets t if there is a target and 0 if there is no target. In real problems there is only an estimate of the original pixel $\widehat{u}(m,n)$ which can be modeled as the original pixel with some noise

$$\widehat{u}(m,n) = u(m,n) + \eta(m,n) \tag{2}$$

One usually assumes that the added noise is spatially uncorrelated. When a target is present one gets a whitened image with the target in the relevant pixels

$$u(m,n)' - \hat{u}(m,n) = t + \eta(m,n) \tag{3}$$

The name of the game is to find an estimate of the background $\hat{u}(m,n)$ such that the variance of the estimation noise $\eta(m,n)$ is minimized.

2.1.1 Linear filtering

Estimating the background pixel with linear filtering is the classic approach and is the most efficient. The estimate is the result of the average of the 8 surrounding pixels from a 3×3 frame or the 16 outer pixels from a 5×5

frame. The resulting estimation noise $\eta(m, n)$ is gaussian distributed and usually has a zero mean. The disadvantage to this method is that there are long tails that result from areas in the picture with a high local variance. The tails are the main cause of false alarms.

2.1.2 Ordered Statistics Filtering

One of the best ways of eliminating the false alarms on the edges is to estimate the background by taking the maximum value out of the 8 or 16 surrounding pixels. A second option is to use the median; however, the maximum is better at reducing the false alarms. The results are shown for a section of the first frame of the 'npa' picture. The distribution for median filtering is normal while the distribution for the Maximum filter is Extreme Value distributed. The Extreme Value Distribution typically occurs when choosing the maximum value out of a set of Gaussian distributed random variables. The probability density function for the Extreme Value distribution is:

$$f\left(x|\mu,\sigma\right) = \frac{1}{\sigma}\exp\left\{\frac{x-\mu}{\sigma}\right\}\exp\left\{-\exp\left(\frac{x-\mu}{\sigma}\right)\right\} \tag{4}$$

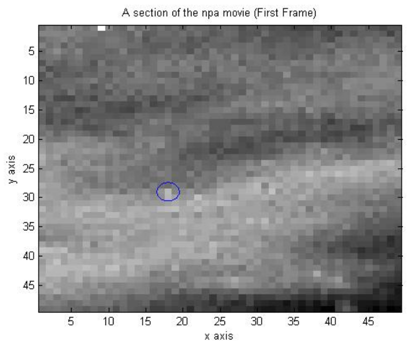
2.2 Normalizing the Noise

After choosing between a variety of basic methods for whitening the noise $\eta(m, n)$,we must deal with the fact that the remaining noise isn't really white. The main problem being the image is non-stationary and is comprised of different types of background. The typical example in our data sets is a target with both clouds and sky in the background. The sky and cloud backgrounds have different variances. Since, the cloud has the highest variance the false alarms and tracks usually are in the cloud. This can be addressed by normalizing the pixel by their local standard deviation. The problem with this approach is that the estimate of the local deviation can be zero. Division by zero or a small number will invariably be the cause of undesirable false alarms. A solution to this problem has been suggested by Raviv and Rotman in [1]. The idea is to divide the pixel by the standard deviation and at the same time avoid division by zero. This can happen when the local standard deviation is zero. One simply divides the pixel by the standard deviation with an added constant in the following manner

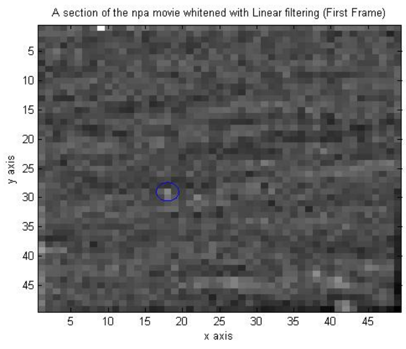
$$I(m, n) = \frac{u(m, n)' - \hat{u}(m, n)}{sdv(m, n) + c} \tag{5}$$

where $sdv(m, n)$ is the local standard deviation at pixel $u(m, n)$ calculated from the 16 or 8 closest pixels. The easiest way to choose the constant c is to take the pixel with the maximum probability.

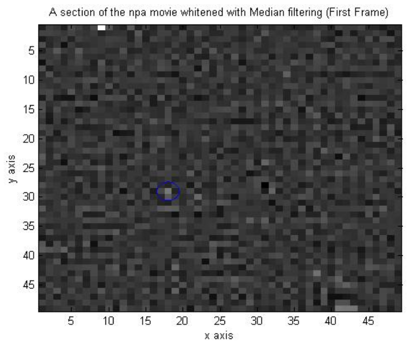
$$c = \max\{P_{sdv}(x)\} \tag{6}$$



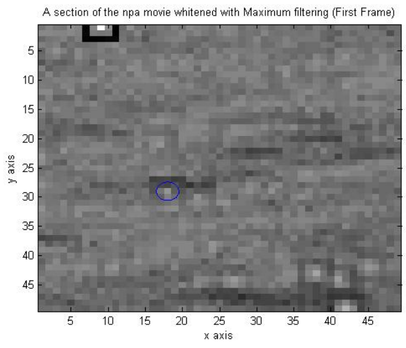
(a)


A section of the npa movie whitened with Linear filtering (First Frame)
y axis
x axis

(b)

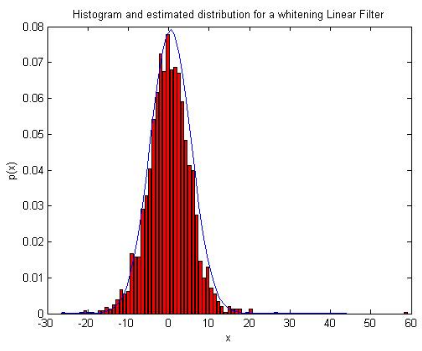

A section of the npa movie whitened with Median filtering (First Frame)
y axis
x axis

(c)

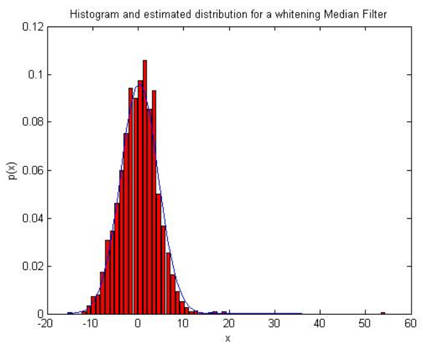

A section of the npa movie whitened with Maximum filtering (First Frame)
y axis
x axis

(d)

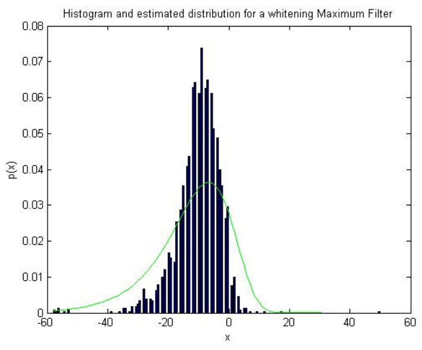
Figure 1: Pre-Processed images before thresholding: (a) The original image, (b) The image whitened with a Linear filter, (c) The image whitened with a Median filter, (d) The image whitened with a Maximum filter. The encircled pixel is the target.



(a)



(b)

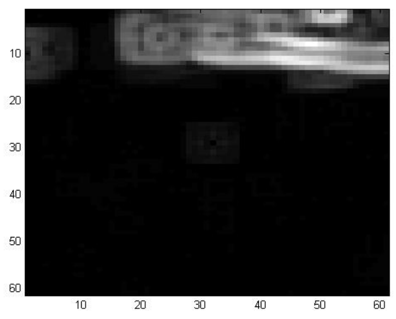


(c)

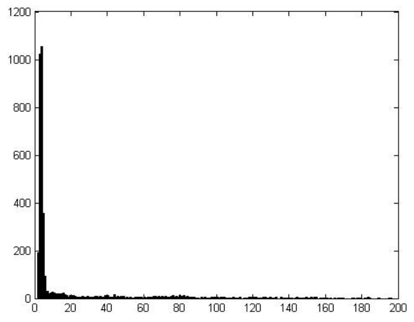
Figure 2: The resulting distributions of the whitened images: (a) The image whitened with a Linear filter, (b) The image whitened with a Median filter, (c) The image whitened with a Maximum filter.

where $P_{sdv}(x)$ is the PDF of the standard deviations. The constant c is calculated individually for each frame.

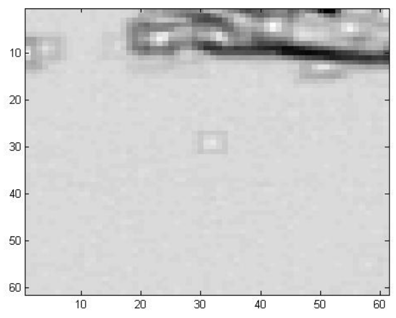
The local standard deviation of the first frame of a movie can be seen in fig. 3(a). One can see from fig. 3(b) that it is a simple matter to choose the constant c. From figures 3(c) and 3(d) we can see that the normalization reduces the variance in the area of the clouds, while at the same time it emphasizes the target (the white pixel close to the center of the picture). The alternative to normalization with the standard deviation is to use a spatially adaptive threshold as explained in the following section.



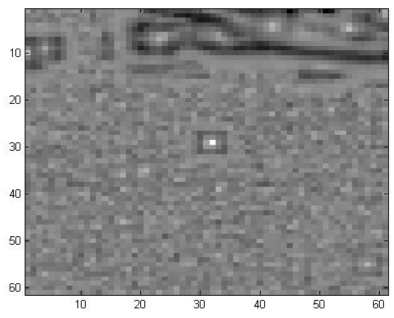
(a)



(b)



(c)



(d)

Figure 3: The image of the local standard deviations: (b) The original image, (b) The histogram of image (a),(c) The first frame after having the mean removed, (d) The first frame after having the mean removed and normalized by the local standard deviation.

2.3 Thresholding the Image

The most basic way of trying to separate between the real targets and the false alarms is to set a global threshold. Ordinarily one chooses a threshold such that it results in an acceptable level of false alarms. Thus the desired threshold th is extracted from the following equation

$$P_{FA} = \int_{th}^{\infty} p_0\left(a\right) da \qquad (7)$$

where P_{FA} is the level of false alarms and $p_0\left(a\right)$ is the PDF (Probability Density Function) for $\eta(m,n)$ when there is no target present.

There are ways of improving on this.

- A Spatially Dynamic Threshold
- A time Dynamic Threshold

2.3.1 A Spatially dynamic threshold

In a non-stationary image, a global threshold is not a very good idea. It would be better to segment the image and set a different threshold for each segment. Another possibility would be to use a threshold that is a function of the local variance calculated from the surrounding 8 or 16 neighboring pixels. Thus

$$th = \alpha + \beta \cdot \sigma(m,n) + \delta \cdot \sigma^2(m,n) \qquad (8)$$

The challenge in this case is to find out whether one can make a significant reduction in the ratio of false alarms to detections with this approach. Another even greater challenge is to globally estimate the set of parameters (α, β, δ). A successful way to estimate these parameters online has yet to found and should be a topic for further research.

2.3.2 A Time Dynamic Threshold

Setting the threshold in a correct manner is vital for the success of the tracking algorithm. This goes both for the initial stage and during the tracking process. The first issue is how to determine the right threshold in the beginning. Currently the threshold is set by setting P_{FA} the acceptable false alarm rate, and determining a threshold that will satisfies equation (7). If the resulting signal lands below the average signal amplitude then the threshold is set slightly below the signal average. This comes at the cost of more false alarms but it is necessary because otherwise the targets will all be below the threshold. Some ad hoc methods for changing the threshold during the tracking process are discussed in section 4.5.4.

2.4 The Tracking Stage

After thresholding comes the tracking stage. We chose the IMM-PDAF method for it's numerical efficiency on one hand and good performance on the other hand. The PDAF (Probability Data Association Filter) is designed to take care of random clutter. The idea is to associate a set of measurements to a given track with prior knowledge of the targets behavior. When one doesn't have exact prior knowledge of the targets behavior the situation calls for the multiple model approach. If the target switches between a few different models, then the IMM (Interactive Multiple Model) algorithm has been widely shown to provide good results by Blackman. S in [2] and Bar-Shalom [7].

In a problem with a high SNR or a high ratio of clutter to the real target the single target approach might not be good enough. It is necessary to maintain a Track Before Detect Algorithm and keep tracking a number of high probability tracks throughout the tracking process. This means that ones simultaneously tracks a number of possible targets at the same time. When tracking multiple targets an additional complication arises, i.e. tracks cross and the validation regions overlap. Then, it is possible to have observations that could belong to either track. When that happens, one has to make an optimal decision regarding the assignment of the observations to the tracks. I use a simplified method in which the track with the highest probability takes the measurements in its validation region. The methods to assign a probability to the track and set up the validation region are discussed in the following sections. Better but more complex approaches are the JPDAF (Joint PDAF) and MHT (Multiple Hypothesis Tracking) as described in [2] and [7].

I will give a brief outline of the tracking algorithm which is explained in detail in the following sections. There are 3 different stages:

Track Initiation At first pairs are created from the first two scans. Then one sets a gate around the next predicted detection. Either there are detections in the following gate and the track survives or it's probability drops due to no detections and it is killed. The IMM-PDAF algorithm is run for each and every pair. Measurements that are not associated with any tracks are used to set up new tracks at every scan.

Track maintenance In the case of high SNR and a single target, the best track is chosen from the track initiation stage and the rest of the other tracks are dropped. If the target's SNR is low then it will be difficult to differentiate between the clutter and the target. Hence it will be necessary to track many possible tracks before making a decision to keep or drop them. In the low SNR case, it may be necessary to use the multi-target approach to help cope with this problem even in the case of a single target.

Track Termination and Re-Initialization When tracking either a single target or even multi-targets, sometimes the target can be lost and the track probability drops below a pre-determined acceptable value. Then the low probability track is dropped. In the single target case it will be necessary to re-initialize the whole process. In the multi-target case this won't be necessary as associated measurements can be used to start new tracks. In the multi-target case, some times, all existing tracks are killed, and, in this case, it is necessary to drop the threshold and create new tracks.

The IMM-PDAF algorithm can be broken down into the following steps which are carried out in an iterative manner.

- The mixing stage of the IMM
- The filtering stage of the IMM- application of the PDAFAI:
  - Use the prediction equations of the kalman filter for each mode. Calculate the gate size
  - Scan Image
  - Pre-process the image
  - Threshold the image
  - Validate the measurement
  - Calculate association probabilities $\beta_k^{i,j}$ for measurement i and for mode j at time k.
  - Update the measurement
- The Combination stage of the IMM

2.5 The Post Tracking Stage

A probability value is assigned to each track. The track with the highest score is the real track. Another important criterion is the length of the track life. If the algorithm locks to the target for less than 50% of the time then, we suggest that the track should be ignored. True tracks tend to have longer lifetimes than tracks that are generated by clutter.

3 Tracking a Single Target with Background Clutter

3.1 Validation of Measurements

In practice we do not consider every possible target in the image or from the sensor. In order to reduce the complexity of the problem we focus only on areas where it is highly probable to find the target. The validation procedure suggested by [5] is to only consider observations that fall in the vicinity of the prior estimate of the next measurement $\hat{z}_{k|k-1} = E[z_k|Z]$. The cumulative set of the real measurements up to time k is

$$Z_k = \{z_1, z_2, \ldots, z_k\} \tag{9}$$

The next observation z_{k+1} conditioned on Z_k is assumed to be gaussian distributed

$$z_{k+1|k} \sim \mathcal{N}\left(\hat{z}_{k+1|k}, S_{k+1}\right) \tag{10}$$

where the associated covariance matrix S_{k+1} is defined as

$$\begin{aligned} S_{k+1} &= E\left\{\left[z_{k+1} - \hat{z}_{k+1|k}\right]\left[z_{k+1} - \hat{z}_{k+1|k}\right]^T \middle| Z_k\right\} \\ &= H_{k+1}\Sigma_{k+1|k}H_{k+1}^T + R_{k+1} \end{aligned} \tag{11}$$

and the elliptical validation region at time $k+1$ is defined as

$$\tilde{V}_{k+1}(\gamma) = \left\{z : \left[z_{k+1} - \hat{z}_{k+1|k}\right]^T S_{k+1}^{-1}\left[z_{k+1} - \hat{z}_{k+1|k}\right] < \gamma\right\} \tag{12}$$

The quadratic expression from (12) is chi-square distributed with n_z degrees of freedom. Alternatively expression (12) can be written as

$$\tilde{V}_{k+1}(\gamma) = \left\{z : \nu_{k+1}^T\left(\gamma S_{k+1}\right)^{-1}\nu_{k+1} < 1\right\} \tag{13}$$

where the innovation ν_k is defined as

$$\nu_k = z_k - \hat{z}_{k|k-1}$$

This defines a hyper-ellipsoid region of the dimension n_z. As an illustration, let's take a look at the 2-Dimensional case. The area of a unit circle is π, a and b are the lengths of the axes. This leads to the next important fact: the volume of the hyper-ellipsoid region. This is defined by the size of unit-hypersphere multiplied by the product of the lengths of the semi-axis of the hyper-ellipsoid. In the example of the 2-Dimensional ellipsoid the lengths of the semi-axis are a and b. The "volume" of a 2-Dim sphere is πa^2. The area of an ellipsoid is πab.

The volume of a hyper-sphere of dimension n with a radius r is

$$c_n = \frac{\pi^{\frac{n}{2}} r^n}{\Gamma\left(\frac{n}{2}+1\right)}$$

where $\Gamma(n)$ is the gamma function defined as

$$\Gamma(n) = \int_0^\infty t^{n-1} e^{-t} dt \qquad ; \quad n > 0$$

The product of the lengths of the semi-axis is obtained by the product of the square root of the eigenvalues of the matrix. The matrix in this case is γS_{k+1} and the squared root of the determinant is equivalent to the product of the lengths of the semi-axis. This is because the determinant is equal to the product of the eigenvalues.

To summarize, the volume of the *Validation Region* is

$$\begin{aligned} V_{k+1} &= c_{n_z} \left|\gamma \cdot S_{k+1}\right|^{1/2} \\ &= c_{n_z} \left|\gamma I_{n_z} \cdot S_{k+1}\right|^{1/2} \\ &= c_{n_z} \left|\gamma I_{n_z}\right|^{1/2} \cdot \left|S_{k+1}\right|^{1/2} \\ &= c_{n_z} \gamma^{n_z/2} \cdot \left|S_{k+1}\right|^{1/2} \\ &= c_{n_z} g^{n_z} \cdot \left|S_{k+1}\right|^{1/2} \end{aligned} \tag{14}$$

where $g \overset{\Delta}{=} \gamma^{1/2}$. The probability of the real target falling in the gate or the validation region is defined as

$$P_G = P\left\{z_{k+1} \in \tilde{V}_{k+1}\right\} \tag{15}$$

Calculating this value as a function of γ is difficult enough but a simple manipulation can make things easier. First we define a linear transform of z_k

$$\tilde{z}_k = \Lambda_k^{-1/2} U_k^T \left(z_k - \hat{z}_{k|k-1}\right) \tag{16}$$

where Λ_k is the eigenvalue matrix of S_k and U_k is the right eigenvector of S_k. Then we get $\tilde{z}_k$ which is gaussian distributed as

$$\tilde{z}_k \sim \mathcal{N}\left(0, I_{n_z}\right) \tag{17}$$

This enables us to redefine (12) as

$$\begin{aligned} \tilde{V}_{k+1}(\gamma) &= \left\{\tilde{z} : \tilde{z}_{k+1}^T \tilde{z}_{k+1} < \gamma\right\} \\ &= \left\{\tilde{z} : \|\tilde{z}_{k+1}\|_2^2 < \gamma\right\} \end{aligned} \tag{18}$$

$\gamma = g^2$	1	4	9	16	25	6.6	9.2	11.4
g:	1	2	3	4	5	2.57	3.03	3.38
$n_z = 1$	.683	.954	.997	.99994	1.0	.99		
$n_z = 2$	.393	.865	.989	.9997	1.0		.99	
$n_z = 3$	.199	.739	.971	.9989	.99998			.99

Table 1: Gating thresholds and values of probability mass in gate

Instead of having to calculate

$$P_G(\gamma) = \int_{\{z_{k+1} \in \tilde{V}_{k+1}\}} f_{z_{k+1}|Z_k}(\alpha_{k+1}) d\alpha_{k+1}$$

one calculates

$$P_G(\gamma) = \int_{\{\tilde{z}_{k+1} \in \tilde{V}_{k+1}\}} \tilde{f}_{\tilde{z}_{k+1}|Z_k}(\alpha_{k+1}) d\alpha_{k+1}$$

where $f_{z_{k+1}|Z_k}(\alpha_{k+1})$ is the PDF from (10) and $\tilde{f}_{\tilde{z}_{k+1}|Z_k}(\alpha_{k+1})$ the PDF from (17).

We show how to explicitly calculate these values in Appendix A. The above Table 1 of gating thresholds can be found in references [7] and [5].

The set of validated measurements at time k is defined as

$$Z(k) = \left\{{z_k}^i\right\}_{i=1}^{m_k} \tag{19}$$

where m_k is the number of valid measurements at time k. The cumulative set of measurements is up to time k is Z_k. The size of the gate or validation region is determined by the parameter γ. This should vary in time according to the amount of clutter and various other criteria.

3.2 The Nearest-Neighbor Standard Filter

In the *nearest-neighbor standard filter* the closest measurement to the prior estimated measurement $\hat{z}_{k|k-1}$ is chosen. The distance that we use in order to decide which measurement is closest in probability to the prior estimated measurement is

$$d_i^2(k) = \left[{z_k}^i - \hat{z}_{k|k-1}\right]^T S_k^{-1} \left[{z_k}^i - \hat{z}_{k|k-1}\right] \tag{20}$$

This is also referred to as the *normalized innovation squared (NIS)*. The measurement that we choose is the one that brings expression (20) to a minimum. The disadvantage to this method is that we don't take into account previous possible measurements. Another disadvantage is that it doesn't take into account the possibility that none of the measurements in the given set of measurements is a target.

3.3 Clutter Model

3.3.1 The Parametric model

In the case of tracking a target in an image, let's say that there are N pixels. In addition we shall assume that

- The detection events are independent of each other.
- The probability of such a detection being a false alarm is P_{FA} in each cell.

Then the probability of the *number of false alarms* is Bernoulli distributed and given by

$$P\left\{n_{FA} = n\right\} = \left(\begin{array}{c} N \\ m \end{array}\right) p^m \left(1-p\right)^{N-m} \tag{21}$$

Assign the volume of the N cells under inspection the value V and call the spatial density of the false alarms

$$\lambda = \frac{E[n_{FA}]}{V} = \frac{Np}{V} \tag{22}$$

If $p \leq 1$ and N is large enough then the Bernoulli distribution can be approximated by a Poisson distribution and thereby we obtain

$$\mu_F(m) = e^{-Np}\frac{(Np)^m}{m!} \tag{23}$$

By using the spatial density ratio from (22), (23) now becomes

$$\mu_F(m) = e^{-\lambda V}\frac{(\lambda V)^m}{m!} \tag{24}$$

This method requires knowledge of the parameter λ. In the next subsection a non-parametric model will be discussed that doesn't require knowledge of the parameter.

3.3.2 The non-parametric Model

In the non-parametric model we assume that the PMF is uniformly distributed.

$$\mu_F(m) = \frac{1}{M}, \qquad m_k = 0, 1, \ldots, N-1 \tag{25}$$

where M is the number of false alarms. We will see in section 3.4.2 that the value of M is not important.

3.4 The Probabilistic Data Association Filter

3.4.1 The Assumptions of the PDAF

With this approach, we take into account the possibility that none of the valid measurements are due to the target. We note that the previous valid measurement are not taken into account. (An approach that takes these measurements into account is referred to as the *The Optimal Bayesian Approach*). This approach will be discussed in a future section. The disadvantage with this approach is that the computational burden grows with time and can saturate even the most powerful computer systems. This method however might be useful for the initialization sequence.

The Probabilistic Data Association Filter (PDAF) is based on three assumptions:

- The track has been initialized
- There is only one target
- The prior state estimate is Gaussian distributed

$$x_{k|k-1} \sim \mathcal{N}\left(\hat{x}_{k|k-1}, \Sigma_{k|k-1}\right) \tag{26}$$

- At each stage before the new set of measurements arrives, a validation region is set up, as in (12)
- Out of the valid measurements there is either one valid measurement or none at all
- The false measurements are uniformly distributed

Under these assumptions the events

$$\theta_k{}^i = \begin{cases} \left\{z_k{}^i \text{ is the target originated measurement}\right\}, & i = 1, \dots, m_k \\ \{\text{none of the measurements is target originated}\}, & i = 0 \end{cases} \tag{27}$$

are mutually exclusive and exhaustive. Using the theorem of total probability the conditional mean of the state can be written as

$$\begin{aligned} \hat{x}_{k|k} &= E[x_k|Z_k] \\ &= \sum_{i=0}^{m_k} E[x_k|\theta_k{}^i, Z_k] p\{\theta_k{}^i|Z_k\} \\ &= \sum_{i=0}^{m_k} \hat{x}_{k|k}^i \beta_k{}^i \end{aligned} \tag{28}$$

where $\hat{x}_{k|k}^{i}$ is the updated state conditioned on the event that the i^{th} validated measurement is correct and

$$\beta_k{}^i \triangleq p\{\theta_k{}^i | Z_k\} \tag{29}$$

is the probability that the i^{th} measurement is the target for $i = 1, \ldots, m_k$. For $i = 0$ Equation (29) represents the probability that there is no valid target in the validation region. Equation (29) is referred to as the *association probability.*

3.4.2 Probabilistic Data Association

The most difficult part of this method is deriving the *association probabilities.* For this reason we will deal with them first. Equation (29) can be written alternatively as

$$\beta_k{}^i = p\{\theta_k{}^i | Z_k\} = p\{\theta_k{}^i | Z(k), m_k, Z_{k-1}\} \tag{30}$$

Using Bayes' rule, this can be rewritten as

$$\beta_k{}^i = \frac{1}{c} p\left[Z(k) | \theta_k{}^i, m_k, Z_{k-1}\right] p\{\theta_k{}^i | m_k, Z_{k-1}\}, \qquad i = 0, 1, \ldots, m_k \tag{31}$$

Now we will derive the first PDF from (31). The PDF of a false measurement is

$$p\left[z_k{}^i | \theta_k{}^j, m_k, Z_{k-1}\right] = \frac{1}{V_k} \tag{32}$$

where $i \neq j$[1]. The PDF of the correct measurement is

$$\begin{aligned} p\left[z_k{}^i | \theta_k{}^i, m_k, Z_{k-1}\right] &= P_G^{-1} \mathcal{N}\left(z_k{}^i; \hat{z}_{k|k-1}, S_k\right) \\ &= P_G^{-1} \mathcal{N}\left(\nu_k{}^i; 0, S_k\right) \\ &= P_G^{-1} \frac{1}{|2\pi S_k|^{\frac{1}{2}}} \exp\left\{-0.5 \nu_k^i{}^T S_k^{-1} \nu_k{}^i\right\} \end{aligned} \tag{33}$$

where $\mathcal{N}\left(\nu_k{}^i; 0, S_k\right)$ is the normal PDF with the argument $\nu_k{}^i$, a mean of zero and a covariance matrix of S_k. The gate probability P_G is given in (15) and V_k is given in (14). After putting the last two expressions together[2] we get

$$p\left[Z(k) | \theta_k{}^i, m_k, Z_{k-1}\right] = \begin{cases} V_k^{-m_k+1} P_G^{-1} \mathcal{N}\left(\nu_k{}^i; 0, S_k\right), & i = 1, \ldots, m_k \\ V_k^{-m_k}, & i = 0 \end{cases} \tag{34}$$

[1]This is based on the assumption that the clutter is uniformly distributed

[2]and by assuming that the entire set of measurements is conditionally independent

The second PDF from (31) little more complex. The second PDF can be shown to be

$$\begin{aligned}\gamma^i(m_k) &\triangleq P\left\{{\theta_k}^i|m_k, Z_{k-1}\right\}\\ &= P\left\{{\theta_k}^i|m_k\right\}\\ &= \begin{cases}\frac{1}{m_k}P_DP_G\left[P_DP_G+(1-P_DP_G)\frac{\mu_F(m_k)}{\mu_F(m_k-1)}\right]^{-1}, & i=1,\ldots,m_k\\ (1-P_DP_G)\frac{\mu_F(m_k)}{\mu_F(m_k-1)}\left[P_DP_G+(1-P_DP_G)\frac{\mu_F(m_k)}{\mu_F(m_k-1)}\right]^{-1}, & \text{i=0}\end{cases}\end{aligned} \tag{35}$$

where P_D is the probability of detecting the target and $\mu_F(m_k)$ is the PMF of the number of false measurements described in section 3.3. The derivation of (35) is a little bit technical. Based on the assumption that there can be at most one valid measurement, let's denote the number of measurements as m_k and it's random variable as m^T. If the random number of false measurements is denoted as m^F, then its value is $m_k - 1$. Now we can show how to calculate (35).

$$\begin{aligned}\gamma^i(m_k) &= P\left\{{\theta_k}^i|m^T=m_k\right\}\\ &= P\left\{{\theta_k}^i|m^F=m_k-1, m^T=m_k\right\}P\left\{m^F=m_k-1|m^T=m_k\right\}\\ &\quad +P\left\{{\theta_k}^i|m^F=m_k, m^T=m_k\right\}P\left\{m^F=m_k|m^T=m_k\right\}\\ &= \begin{cases}\frac{1}{m_k}P\left\{m^F=m_k-1|m^T=m_k\right\}+0\cdot P\left\{m^F=m_k|m^T=m_k\right\}, & i=1,\ldots,m_k\\ 0\cdot P\left\{m^F=m_k-1|m^T=m_k\right\}+1\cdot P\left\{m^F=m_k|m^T=m_k\right\}, & i=0\end{cases}\end{aligned} \tag{36}$$

In the case where there is only one valid measurement by using Bayes' formula we get

$$\begin{aligned}P\left\{m^F=m_k-1|m^T=m_k\right\} &= \frac{P\left\{m^T=m_k|m^F=m_k-1\right\}P\left\{m^F=m_k-1\right\}}{P\left\{m^T=m_k\right\}}\\ &= \frac{P_DP_G\mu_F(m_k-1)}{P\left\{m^T=m_k\right\}}\end{aligned} \tag{37}$$

P_GP_D represents the possibility that the target has been detected and falls within the gate.

In the complementary case where all of the measurements are false measurements we get

$$\begin{aligned}P\left\{m^F=m_k|m^T=m_k\right\} &= \frac{P\left\{m^T=m_k|m^F=m_k\right\}P\left\{m^F=m_k\right\}}{P\left\{m^T=m_k\right\}}\\ &= \frac{(1-P_DP_G)\,\mu_F(m_k)}{P\left\{m^T=m_k\right\}}\end{aligned} \tag{38}$$

where the denominator for both (37) and (38) is given by

$$\begin{aligned} P\left\{m^T = m_k\right\} &= P\left\{m^T = m_k | m^F = m_k - 1\right\} P\left\{m^F = m_k - 1\right\} \\ &\quad + P\left\{m^T = m_k | m^F = m_k\right\} P\left\{m^F = m_k\right\} \\ &= P_D P_G \mu_F(m_k - 1) + \left(1 - P_D P_G\right) \mu_F(m_k) \qquad (39) \end{aligned}$$

By combining equations (37)-(39) and inserting them into (36) we get (35).

The next stage is to insert the expressions for μ_F into the PMF. In section 3.3 we had two different models. The first one was the parametric model and the second the non-parametric model. The non-parametric is simpler and more robust and has better results in practice according to [7]. So by inserting (25) into (35) we get the following result

$$\gamma^i(m_k) = \begin{cases} \frac{1}{m_k} P_D P_G, & i = 1, \dots, m_k \\ 1 - P_D P_G, & i = 0 \end{cases} \qquad (40)$$

We went through all of the mathematics above in order to calculate ${\beta_k}^i$. By inserting (40) and (34) into (31), we get β up to a normalizing constant

$$\begin{aligned} {\beta_k}^i &\propto \begin{cases} V_k^{-m_k+1} P_G^{-1} \frac{1}{|2\pi S_k|^{\frac{1}{2}}} e_i \cdot \frac{1}{m_k} P_D P_G, & i = 1, \dots, m_k \\ V_k^{-m_k} \cdot \left(1 - P_D P_G\right), & i = 0 \end{cases} \\ &\propto \begin{cases} V_k^{-m_k+1} \frac{P_D}{m_k} \mathcal{N}\left(\nu_k^i; 0, S_k\right), & i = 1, \dots, m_k \\ V_k^{-m_k} \cdot \left(1 - P_D P_G\right), & i = 0 \end{cases} \qquad (41) \end{aligned}$$

where

$$e_i \triangleq \exp\left\{-0.5 \nu_k^{i\ T} S_k^{-1} {\nu_k}^i\right\}$$

Since equation (41) is correct up to a normalizing constant it can be written as

$$\begin{aligned} {\beta_k}^i &\propto \begin{cases} e_i, & i = 1, \dots, m_k \\ V_k^{-1} \left|2\pi S_k\right|^{\frac{1}{2}} \cdot m_k \frac{1 - P_D P_G}{P_D}, & i = 0 \end{cases} \\ &\propto \begin{cases} e_i, & i = 1, \dots, m_k \\ \left(\frac{2\pi}{\gamma}\right)^{n_z/2} c_{n_z}^{-1} m_k \frac{1 - P_D P_G}{P_D}, & i = 0 \end{cases} \qquad (42) \end{aligned}$$

To summarize, β_k is calculated by the following expression

$${\beta_k}^i = \begin{cases} \frac{e_i}{b + \sum_{j=1}^{m_k} e_j}, & i = 1, \dots, m_k \\ \frac{b}{b + \sum_{j=1}^{m_k} e_j}, & i = 0 \end{cases} \qquad (43)$$

where

$$b \triangleq \left(\frac{2\pi}{\gamma}\right)^{n_z/2} c_{n_z}^{-1} m_k \frac{1 - P_D P_G}{P_D} \qquad (44)$$

3.4.3 The state estimation

Returning to equation (28)

$$\hat{x}_{k|k} = \sum_{i=0}^{m_k} \hat{x}^{\,i}_{k|k} {\beta_k}^i \tag{45}$$

we now know how to calculate β_k^i and now it's time to see how to calculate the updated state $\hat{x}^{\,i}_{k|k}$. The current state conditioned on the past measurements and on the i^{th} measurement being correct is

$$\hat{x}^{\,i}_{k|k} = \hat{x}_{k|k-1} + M_k {\nu_k}^i \tag{46}$$

where the innovation is defined as

$${\nu_k}^i = {z_k}^i - \hat{z}_{k|k-1} \tag{47}$$

The gain M_k is the same as in the standard filter

$$M_k = \Sigma_{k|k-1} H_k^T S_k^{-1} \tag{48}$$

for $i = 0$ or when $m_k = 0$ we have

$$x^{\,0}_{k|k} = \hat{x}_{k|k-1} \tag{49}$$

By inserting

$$\hat{x}^{\,i}_{k|k} = \hat{x}^{\,i}_{k|k-1} + M_k {\nu_k}^i \tag{50}$$

where the combined innovation is

$$\nu_k = \sum_{i=1}^{m_k} {\beta_k}^i {\nu_k}^i \tag{51}$$

The updated covariance matrix is

$$\Sigma_{k|k} = \Sigma_{k|k-1} - \left[1 - {\beta_k}^0\right] M_k S_k M_k^T + \widetilde{\Sigma}_k \tag{52}$$

where the extra term in

$$\widetilde{\Sigma}_k \triangleq M_k \left[\sum_{i=1}^{m_k} {\beta_k}^i {\nu_k}^i {\nu_k}^{i\ T} - \nu_k \nu_k^T \right] M_k^T \tag{53}$$

accounts for the extra uncertainty added by not knowing which measurement is the correct measurement. The proof for this can be found in [7]. The prediction equations are the same as in the standard Kalman filter.

$$\begin{aligned}
\hat{x}_{k+1|k} &= F_k \hat{x}_{k|k} && (54)\\
\hat{z}_{k+1|k} &= H_{k+1} \hat{x}_{k+1|k} && (55)\\
\Sigma_{k+1|k} &= F_k \Sigma_{k|k} F_k^T + Q_k && (56)\\
S_{k+1} &= H_{k+1} \Sigma_{k+1|k} H_{k+1}^T + R_{k+1} && (57)
\end{aligned}$$

The likelihood of the current set of measurements filter given the previous measurements is defined as

$$\begin{aligned}\Lambda(k) &\triangleq P\left\{Z(k)|m_k, Z_{k-1}\right\} \\ &= P\left\{z_1(k),\ldots,z_{m_k}(k)|m_k, Z_{k-1}\right\} \\ &= P\left\{\nu_1(k),\ldots,\nu_{m_k}(k)|m_k, Z_{k-1}\right\} \\ &= V_k^{-m_k}\gamma^0(m_k) + V_k^{-m_k+1}\sum_{j=1}^{m_k} P_G^{-1}\mathcal{N}\left(\nu_k{}^j; 0, S_k\right)\gamma^j(m_k) \quad (58)\end{aligned}$$

For the convenience of the reader a summary of the PDAF algorithm is provided in Table 2.

Comments:

- This method can be further improved by generalizing the method to incorporate the *feature* or intensity of the target in the calculation of the probabilities.
- The method relies on the assumption that the track has been initialized. In the next section, a method for track formation with clutter will be discussed.

The PDAF Algorithm

Prediction Equations

$$\begin{aligned}
\hat{x}_{k|k-1} &= F_{k-1}\hat{x}_{k-1|k-1} \\
\hat{z}_{k|k-1} &= H_k\hat{x}_{k|k-1} \\
\Sigma_{k|k-1} &= F_{k-1}\Sigma_{k-1|k-1}F_{k-1}^T + G_{k-1}Q_{k-1}G_{k-1}^T \\
S_k &= H_k\Sigma_{k|k-1}H_k^T + R_k
\end{aligned}$$

Validation of measurements

$$Z(k) = \left\{{z_k}^i\right\}_{i=1}^{m_k}$$

is the set of measurements that fall in the region defined by

$$\tilde{V}_k(\gamma) = \left\{z : \nu_k^T \left(S_k\right)^{-1} \nu_k < \gamma\right\}$$

where

$${\nu_k}^i = {z_k}^i - \hat{z}_{k|k-1} \qquad i = 1, \ldots, m_k$$

and m_k is the number of validated measurements.

Calculate ${\beta_k}^i \qquad i = 1, \ldots, m_k$

$$b \triangleq \left(\frac{2\pi}{\gamma}\right)^{n_z/2} c_{n_z}^{-1} m_k \frac{1 - P_D P_G}{P_D}$$

$$e_i \triangleq \exp\left\{-0.5\nu_k^{i\ T} S_k^{-1} {\nu_k}^i\right\} \qquad i = 1, \ldots, m_k$$

$${\beta_k}^i = \begin{cases} \frac{e_i}{b + \sum_{j=1}^{m_k} e_j}, & i = 1, \ldots, m_k \\ \frac{b}{b + \sum_{j=1}^{m_k} e_j}, & i = 0 \end{cases}$$

Measurement update

If $m_k = 0$ **then:**

State update:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1}$$

Covariance update:

$$\Sigma_{k|k} = \Sigma_{k|k-1}$$

Else:

Combined Innovation:

$$\nu_k = \sum_{i=1}^{m_k} {\beta_k}^i {\nu_k}^i$$

Kalman Gain:

$$M_k = \Sigma_{k|k-1} H_k^T S_k^{-1}$$

State update:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + M_k \nu_k$$

Covariance update:

$$\widetilde{\Sigma}_k \triangleq M_k \left[\sum_{i=1}^{m_k} {\beta_k}^i {\nu_k}^i {\nu_k}^{i\ T} - \nu_k \nu_k^T\right] M_k^T$$

$$\Sigma_{k|k} = \Sigma_{k|k-1} - \left[1 - {\beta_k}^0\right] M_k S_k M_k^T + \widetilde{\Sigma}_k$$

■

Table 2: Summary Of The PDAF Algorithm (One Cycle)

4 Tracking a low SNR target with background clutter

4.1 Introduction

Algorithms for track initiation can be divided into two different categories:

- Non-Bayesian/Logic based methods
- Bayesian methods

The first approach is described in [7] in sections 2.6 and 7.4. The second approach can be done either by using the Optimal Bayesian approach as described in [7] section 3.5 or by using IMM-PDAF (Interactive Multiple Model) filter from section 4.4. We describe in the second approach using the IMM-PDAF filter for the track formation. Before that we give a brief outline of the different models that are used to describe the behavior of manoeuvring targets. After that, we give a brief outline of the IMM algorithm and then describe how to form a track with this method.

4.2 Modeling the behavior of a manoeuvring target

I will give a brief outline on some of the more popular models used for tracking manoeuvring targets. This is in no way exhaustive and in some cases more exotic models are required.

The most basic and simple model is that of a stationary target. For the sake of simplicity, we analyze the problem in two dimensions. This does not limit the generality of the model as one can easily extend the model to higher dimensions.

Our proposal for the stationary model is the following:

$$
\begin{aligned}
x_{k+1} &= Fx_k + Gw_k && (59)\\
z_k &= Hz_k + v_k && (60)
\end{aligned}
$$

where

$$w_k \sim \mathcal{N}\left(0, q_k I_{n_x}\right) \tag{61}$$

where n_x is the dimension of the state and

$$\dot{r}_k \sim \mathcal{N}\left(0, r_k I_{n_z}\right) \tag{62}$$

where n_z is the dimension of the measurement. The SS (State Space) matrices are given by

$$
\begin{aligned}
F &= I_2 && (63)\\
H &= I_2 && (64)\\
G &= 1 && (65)
\end{aligned}
$$

and the state includes only the position of the target

$$x_k = \begin{bmatrix} r_x(k) \\ r_y(k) \end{bmatrix} \tag{66}$$

where r_x corresponds to the position on the x-axis and r_y corresponds to the position on the y-axis. Usually, one will assume that q_k is very low and constant in time. This describes a target that wobbles around itself and mostly stays in the same place[3].

The next model is called the *Uniform Motion Model* or *White Noise Acceleration Model* and has been described before:

$$\dot{r}_x(k+1) = \dot{r}_x(k) + Tw_k^{(x)} \tag{67}$$

where $\dot{r}_x$ denotes the speed on the x-axis. The equations for the position and speed on the x-axis are

$$\begin{aligned} r_x(k+1) &= r_x(k) + T \cdot \text{Av. Speed} && (68)\\ &= r_x(k) + \frac{T}{2} \cdot (\dot{r}_x(k+1) + \dot{r}_x(k)) && (69)\\ &= r_x(k) + \frac{T}{2} \cdot \left(2\dot{r}_x(k) + Tw(k)^{(x)}\right) && (70)\\ &= r_x(k) + T \cdot \dot{r}_x(k) + \frac{T^2}{2} w(k)^{(x)} && (71) \end{aligned}$$

The equations for position and speed on the y-axis are similar. To summarize the matrices for the SS equations (59) and (60) are given by

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ T & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & T & 1 \end{bmatrix} \tag{72}$$

$$G = \begin{bmatrix} T & 0 \\ T^2/2 & 0 \\ 0 & T \\ 0 & T^2/2 \end{bmatrix} \tag{73}$$

$$H = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{74}$$

where the state vector is

$$x_k = \begin{bmatrix} \dot{r}_x(k) \\ r_x(k) \\ \dot{r}_y(k) \\ r_y(k) \end{bmatrix} \tag{75}$$

[3]Note that one is assuming that the white noise input is piece-wise constant if this is not the case, one has use the method described in [5] section 2.3 and section 2.7 for approximating a continuous SS (state space) model by a discrete SS model

For an accelerating target, the most common model used is referred to as the *Wiener Process Acceleration Model.* By augmenting the state vector with a variable for the acceleration, we get here a third order model with the following matrices:

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ T & 1 & 0 & 0 & 0 & 0 \\ T^2/2 & T & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & T & 1 & 0 \\ 0 & 0 & 0 & T^2/2 & T & 1 \end{bmatrix} \tag{76}$$

$$G = \begin{bmatrix} 1 & 0 \\ T & 0 \\ T^2/2 & 0 \\ 0 & 1 \\ 0 & T \\ 0 & T^2/2 \end{bmatrix} \tag{77}$$

$$H = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{78}$$

where the state vector is

$$x_k = \begin{bmatrix} a_x(k) \\ \dot{r}_x(k) \\ r_x(k) \\ a_y(k) \\ \dot{r}_y(k) \\ r_y(k) \end{bmatrix} \tag{79}$$

A description of these models can be found [5] section 2.7. More sophisticated models such as the *Singer Model* for maneuvering targets with a colored noise input [8] or James Helferty's Turn-Rate Distribution Model [9] can be used if necessary, but, for this particular problem dealt with in this project, the first 3 models should suffice.

Another model worthy of mentioning is the *Coordinated Turn Model.* This model is especially useful for Air Traffic Control where civilian aircraft perform turns of a constant radius before landing. A description of it's application is given by Bar-Shalom and Li in [10]. A brief survey of some of the models mentioned here[4] is given by Lerro and Bar-Shalom in [13].

4.3 The Interacting Multiple Model

When there is uncertainty regarding the type of model and the parameters of the model, it is necessary to use multiple models. For example, take a

[4]Except for the *Singer Model* and *Helferty Model*

target that is driving along a road and then leaves the road and starts driving across rough terrain. One would require two models in this case. The IMM (Interacting Multiple Model) is a hybrid approach that has an effect of soft switching between different models. Its performance is comparable to other algorithms such as the GPB (Generalized-Pseudo-Bayesian) approach and it is shown by Bar-Shalom to be more efficient [7].

The hybrid system is not surprisingly formulated by a set of linear SS equations corresponding to each individual mode or model.

$$\begin{aligned} x(k+1) &= F\left(k, \mathcal{M}_k\right) x(k) + G\left(k, \mathcal{M}_k\right) w\left(k, \mathcal{M}_k\right) & (80) \\ z(k) &= H\left(k, \mathcal{M}_k\right) + v\left(k, \mathcal{M}_k\right) & (81) \end{aligned}$$

where $\mathcal{M}_k$ is the true mode at time k. The state noise $w\left(k, \mathcal{M}_k\right)$ and the measurement noise are uncorrelated with the Gaussian initial state $x(0)$; they, are mutually uncorrelated white Gaussian noise vectors with the covariance $Q\left(k, \mathcal{M}_j\right)$ and $R\left(k, \mathcal{M}_j\right)$ respectively.

The idea behind the algorithm is to use homogeneous Markovian transition system where the probability of moving from one mode to another is

$$P\left\{\mathcal{M}(k+1) = \mathcal{M}_j | \mathcal{M}(k) = \mathcal{M}_i\right\} = \pi_{ij} \qquad \forall i, j \in M \tag{82}$$

where M denotes the set of all modes assumed to be in the MM (Multiple Model) scheme and $\mathcal{M}(k)$ represents the mode at time k.

Now I will describe the conventions used in this model

$\hat{x}_j(k\|k),\ P_j(k\|k)$	state estimate of the mode-matched filter j at k and its covariance
$\hat{x}_{oj}(k\|k),\ P_{oj}(k\|k)$	mixed condition for the mode matched filter j at k
$\hat{x}(k\|k),\ P(k\|k)$	combined state estimate and its covariance
$\mu_j(k)$	mode probabilities for filter j at time k
$\mu_{i\|j}(k)$	the probability of going from mode i to j at time k,
$\Lambda_j(k)$	the likelihood function of mode matched filter j

The idea behind the IMM algorithm is simple. Say we have two different models. At every different stage, the number of possible options is 2. Either the target is in model 1 or model 2. Then for the next stage the target can be in either model 1 or model 2. As the process continues the number of possibilities grows exponentially. In this specific case of two models, the number of possibilities grows at a rate of 2^k like a binary tree. In order to avoid this, one has to make use of sub-optimal algorithms. The IMM does this by a mixing stage before individually applying the Kalman filter to each stage. This is demonstrated in Figure 4.

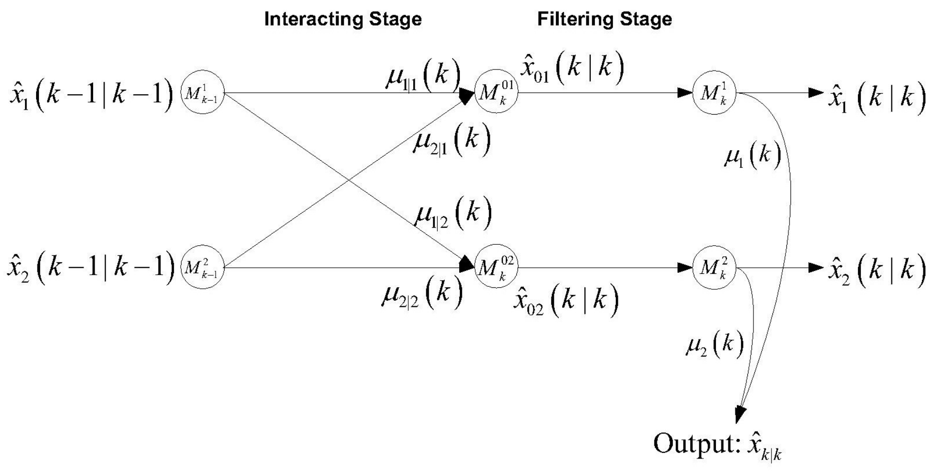


Figure 4: IMM diagram

More details and background on this algorithm are discussed by Blom in [11] and by Bar-Shalom and Li in [12] and section 1.5.4 of [7]. The algorithm is described in Table 3.

When plugging the PDAF algorithm into the IMM, one has to replace Equation (92) with the expression for the combined innovation for the PDAF from (51). The likelihood function from (95) is replaced with the expression from (58). The association event probabilities β are calculated by using equation (41) instead of (42). This is necessary in the event of $P_D = 0$.

Interaction ($\forall j \in M$)**:**

- predicted mode probability:

$$\overline{\mu}_j \triangleq P\left\{\mathcal{M}_j(k)|Z_{k-1}\right\} = \sum_i \pi_{ij}\mu_i(k-1) \tag{83}$$

- mixing probability:

$$\mu_{i|j} \triangleq P\left\{\mathcal{M}_i(k-1)|\mathcal{M}_j(k), Z_{k-1}\right\} = \pi_{ij}\mu_i(k-1)/\bar{\mu}_j \tag{84}$$

- $$\hat{x}_{oj}(k-1|k-1) \triangleq E\left[x(k-1)|\mathcal{M}_j(k), Z_{k-1}\right] = \sum_i \hat{x}_i(k-1|k-1)\mu_{i|j} \tag{85}$$

- $$P_{oj}(k-1|k-1) = \sum_i P_i(k-1|k-1)\mu_{i|j} + X_j \tag{86}$$

- where the "spread-of-the-means" term in the mixing is

$$X_j \triangleq \sum_i \left[\hat{x}_i(k-1|k-1) - \hat{x}_{oj}(k-1|k-1)\right]\left[\hat{x}_i(k-1|k-1) - \hat{x}_{oj}(k-1|k-1)\right]^T \mu_{i|j} \tag{87}$$

Filtering ($\forall j \in M$)**:**

- state prediction:

$$\hat{x}_j(k|k-1) = F_j(k-1)\hat{x}_{oj}(k-1|k-1) \tag{88}$$

- covariance prediction:

$$P_j(k|k-1) = F_j(k-1)P_{oj}(k-1|k-1)F_j(k-1)^T + G_j(k-1)Q_j(k-1)G_j(k-1)^T \tag{89}$$

- residual covariance:

$$S_j = H_j P_j(k|k-1)H_j^T + R_j \tag{90}$$

- filter gain:

$$W_j = P_j(k|k-1)H_j^T S_j^{-1} \tag{91}$$

- residual:

$$\nu_j \triangleq z(k) - H_j\hat{x}_j(k|k-1) \tag{92}$$

- state correction:

$$\hat{x}_j(k|k) = \hat{x}_j(k|k-1) + W_j\nu_j \tag{93}$$

- covariance correction

$$P_j(k|k) = P_j(k|k-1) - W_j S_j W_j^T \tag{94}$$

- likelihood function:

$$\Lambda_j = \mathcal{N}\left(\nu_j; 0, S_j\right) = |2\pi S_j|^{-1/2} \exp\left(-\frac{1}{2}\nu_j^T S_j^{-1}\nu_j\right) \tag{95}$$

- mode probability:

$$\mu_j = \frac{\bar{\mu}_j\Lambda_j}{\sum_i \bar{\mu}_i\Lambda_i} \tag{96}$$

Combination:

- $$\hat{x} \triangleq E\left[x(k)|Z_k\right] = \sum_i \hat{x}_j(k|k)\mu_j \tag{97}$$

- $$P(k|k) \triangleq E\left[\left[x(k) - \hat{x}(k|k)\right]\left[x(k) - \hat{x}(k|k)\right]^T |Z_k\right] = \sum_i P_j(k|k)\mu_j + X \tag{98}$$

- where the "spread-of-the-means" term in combination is

$$X \triangleq \sum_i \left[\hat{x}_i(k|k) - \hat{x}(k|k)\right]\left[\hat{x}_i(k|k) - \hat{x}(k|k)\right]^T \mu_i \tag{99}$$

■

Table 3: Summary Of The IMM Algorithm (One Cycle)

4.4 Combining The IMM with the PDAF

In order to apply the IMM-PDAF algorithm to the problem of track initiation, two models are used: "observable target" (true target),$\mathcal{M}_1$, an "unobservable target" (no target) model, $\mathcal{M}_0$. Both filters are *Uniform Motion Models.* The difference between them is that in the "unobservable target" model $P_D = 0$. The TTP (True Target Probability) is used to discriminate between false tracks and the real track. The TTP is defined as

$$TTP(k) \triangleq \mu_1(k) \tag{100}$$

where $\mu_1(k)$ is the mode probability for the "observable target" model. There are two different ways to implement this. The first is the fixed window implementation. The second is the sliding window implementation, when unvalidated measurements are used to initiate new tracks. This is better, since when the algorithm locks on to the wrong track there is a chance of switching back to the right track. Another way to improve the results is to include the *Amplitude Information.* This will be discussed in later sections.

The process of track formation in fixed window implementation consists of

1. **Track Pair Initiation:** Initial pairs of measurement are created by using the first two scans or images. The pairs are further pruned by using the following criterion

$$\begin{array}{rll} & \left|z_1^i(1) - z_1^j(1)\right| < v_{1\max}T + 2\sqrt{R_{11}} & \\ \text{and} & \left|z_1^i(2) - z_1^j(2)\right| < v_{2\max}T + 2\sqrt{R_{22}} & \forall i \neq j \end{array} \tag{101}$$

   where $z_k^i(1)$ is the i^{th} measurement for the x-axis (1) at time k and $z_k^i(2)$ is the i^{th} measurement for the y-axis (1) at time k.

2. **Tracking Maintenance:** For $k = 3, \ldots, N_W$ the IMM-PDAF algorithm is implemented with two models as previously explained. In this method the *Track before Declare* or *Track before Detect* procedure is used. The *markov transition matrix* used to describe the transition between models is

$$\pi = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix} \tag{102}$$

   The track with the highest TTP is given preference when assigning measurements in order to avoid giving the same measurement to different tracks. The gating region has to be the same for both models.

Otherwise one cannot perform the calculation in (96). Both models also have to have the same number of measurements (by union of all the gates for each individual model). In practice the volume of the gating region is taken from the model with the largest volume. When the track stops getting measurements in its gate region the covariance matrix grows and with it the gating region. This is because the gating region grows so large it eventually gets flooded with measurements. To prevent this from happening, tracks with too many measurements are dropped.
Tracks with a TTP below a certain threshold specified by the user are dropped. In the sliding window implementation, unassociated measurements are used to initiate new tracks. This is especially important in a problem with a low P_D.

The IMM-PDAF algorithm can also be useful for tracking a manoeuvering target once the real track has been confirmed. By adding on additional models, one can dramatically increase the robustness of the algorithm and maintain the track. Another advantage is that by monitoring the TTP, one has a fairly reliable criterion for terminating a track.

4.5 Using Amplitude Information in the IMM-PDAF

It is possible to improve the reliability of the tracking by adding the information about the amplitude onto the measurement vector. The set of measurements is obtained by thresholding an image; thereby one obtains a set of measurements with varying amplitudes. If one takes into account only the position and not the amplitude one loses valuable information. By using both one should expect a drastic improvement in the performance. First a description of the PDAFAI (PDAF Amplitude Information) will be given as described in [7] and [13]. Secondly I will explain in detail how to combine this algorithm with the IMM algorithm.

4.5.1 PDAFAI

The statistical information about the amplitude can be incorporated into the PDAF algorithm in the following manner. One assumes that the following information is available:

- $p_0(a)$ = PDF of the amplitude it is due to noise only
- $p_1(a)$ = PDF of the amplitude if it originated from the target

The simplest model for the signal amplitude is

$$\begin{aligned} H_0 : &\qquad a_k = n_k \\ H_1 : &\qquad a_k = s + n_k \end{aligned} \tag{103}$$

where H_0 is the hypothesis for no target, H_1 is the hypothesis for a target, s is the real amplitude and n_k is assumed to be i.i.d. additive white gaussian noise with a variance of σ_n^2. Then, we use the maximum likelihood to estimate the variance of the pixels in the first frame. As each frame in the movie is pre-processed and normalized the background noise does not vary much over time. If the signal strength is not known (and this usually is the case) then one can estimate it by taking the maximum pixel in each frame defined as

$$s_k = \max_{m,n}\{u_k(m,n)\} \tag{104}$$

where $u_k(m,n)$ is a pixel at index (m,n) at time k. Then the estimated signal strength is

$$\bar{s} = \frac{1}{K}\sum_{l=1}^{K} s_l \tag{105}$$

This is a weak assumption but it works very well for real data. When there is no target then one gets a signal at the height of the noise. If the noise is white then the tracks that are initiated eventually get killed off during the tracking process. Then the probability functions are assumed to be

- $p_0(a) = \mathcal{N}\left(a; 0, \sigma_n^2\right)$
- $p_1(a) = \mathcal{N}\left(a; \bar{s}, \sigma_n^2\right)$

It is assumed that the strength of the signal/amplitude s is known or more or less constant; therefore, the estimate is fairly good. In the detection problem where one is tracking a point target with a fluctuating amplitude strength, a slight modification is required. In Fig. 5 there is an example of the signal strength in an Irena movie. The method for producing these movies is explained in [14]. It is in fact the maximum pixel in each frame. One can see that signal is a combination of a periodic signal that has a period of 8 frames and a gradually increasing average that changes over time. The periodic signal can be explained by the nature of the signal. It is spread over 3 or 4 neighboring pixels. When the center of target moves from one pixel to the next, its intensity is spread evenly between a few pixels and it is at its lowest intensity. When the center of the target is in the center of the pixel, its intensity is the highest. In the synthetic data here, the target is traveling at a speed of 1/8; the results fit our theory. The changing average can be explained by clouds entering and leaving the picture. A better model for this case would be

$$\begin{array}{ll} H_0: & a_k = n_k \\ H_1: & a_k = s + \xi_k + n_k \end{array} \tag{106}$$

where ξ_k is a i.i.d. white gaussian noise and is uncorrelated with n_k. The variance of ξ_k is estimated by taking the sample variance over $\{s_k\}_{l=1}^{K}$. The

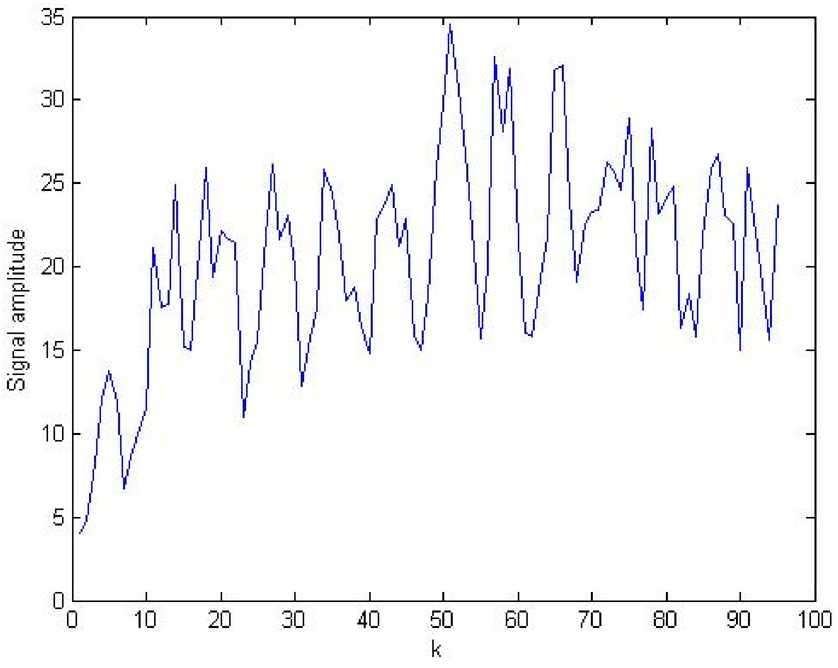


Figure 5: The signal amplitude taken from an Irena movie

variance is calculated by

$$\widehat{\sigma}_{\xi}^{2} = \frac{1}{K-1}\sum_{l=1}^{K}\left(s_l - \bar{s}\right)^2 \tag{107}$$

Finally the PDF's for both the target present hypothesis H_1 and no target hypothesis H_0 is assumed to be

- $p_0(a) = \mathcal{N}\left(a; 0, \sigma_n^2\right)$
- $p_1(a) = \mathcal{N}\left(a; \bar{s}, \sigma_n^2 + \sigma_{\xi}^2\right)$

In practice the best results are obtained by using the moving average of the estimated signal amplitude. Take for example the following estimated signal amplitude and its moving average for 10 samples in Fig. 6

Another reason why this is better is because one can implement this method online while scanning. The previous model with the added noise ξ has to be carried out offline. The estimation of the signal variance σ_{ξ}^2 can only be carried out after scanning the whole video. In this case the PDF's for both the target present hypothesis H_1 and no target hypothesis H_0 is assumed to be

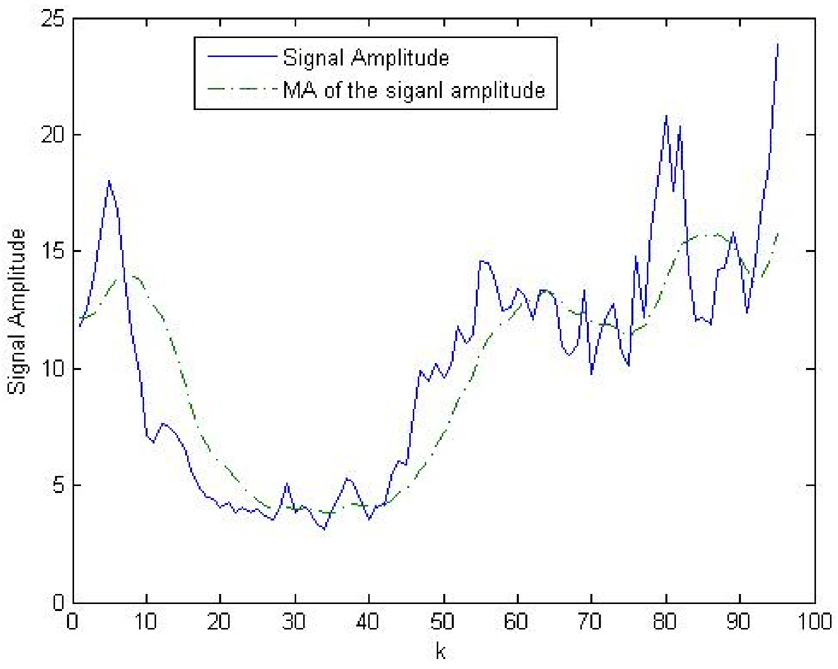


Figure 6: The solid blue line is the estimated signal amplitude and the dashed green line is it's moving average

- $p_0(a) = \mathcal{N}\left(a; 0, \sigma_n^2\right)$
- $p_1(a) = \mathcal{N}\left(a; s_{MA}(k), \sigma_n^2\right)$

where the moving average is calculated by

$$s_{MA}(k) = \frac{1}{10}\sum_{l=-10}^{0} s(l) \tag{108}$$

After finding an appropriate model for the problem, the probability of detection P_D and the probability of a false alarm P_{FA} are calculated with the following equation:

$$P_D = \int_\tau^\infty p_1(a)da, \qquad a > \tau \tag{109}$$

$$P_{FA} = \int_\tau^\infty p_0(a)da, \qquad a > \tau \tag{110}$$

After applying the threshold, the resulting distributions for "target" and "no-target" measurements are

$$p_1^\tau(a) = \frac{1}{P_D} p_1(a) \tag{111}$$

$$p_0^\tau(a) = \frac{1}{P_{FA}} p_0(a) \tag{112}$$

The additional information about the amplitude will be included in the algorithm by rewriting equations (32) and (33).

$$p\left[{z_k}^i | {\theta_k}^j, m_k, Z_{k-1}\right] = \frac{1}{V_k} p_0^\tau({a_k}^i) \tag{113}$$

where $i \neq j$. In order to simplify the calculations define the ratio

$$\lambda_i \triangleq \frac{p_1^\tau({a_k}^i)}{p_0^\tau({a_k}^i)} \qquad i = 1, \ldots, m_k \tag{114}$$

The PDF of the correct measurement is

$$\begin{aligned} p\left[{z_k}^i | {\theta_k}^i, m_k, Z_{k-1}\right] &= P_G^{-1} \mathcal{N}\left({z_k}^i; \hat{z}_{k|k-1}, S_k\right) p_1^\tau({a_k}^i) \\ &= P_G^{-1} \mathcal{N}\left({\nu_k}^i; 0, S_k\right) p_1^\tau({a_k}^i) \\ &= P_G^{-1} \frac{1}{|2\pi S_k|^{\frac{1}{2}}} \exp\left\{-0.5 \nu_k^i{}^T S_k^{-1} {\nu_k}^i\right\} p_1^\tau({a_k}^i) \end{aligned} \tag{115}$$

and by inserting equations (113) and (115) into (34) we get the following

$$\begin{aligned} p\left[Z(k) | {\theta_k}^i, m_k, Z_{k-1}\right] &= \begin{cases} V_k^{-m_k+1} P_D \frac{1}{m_k} \mathcal{N}\left({\nu_k}^i; 0, S_k\right) p_1^\tau({a_k}^i) \prod_{\substack{j=1 \\ i \neq j}}^{m_k} p_0^\tau({a_k}^j), & i = 1, \ldots, m_k \\ V_k^{-m_k} \prod_{j=1}^{m_k} p_0^\tau({a_k}^j), & i = 0 \end{cases} \\ &= \begin{cases} V_k^{-m_k+1} P_D \frac{1}{m_k} \mathcal{N}\left({\nu_k}^i; 0, S_k\right) \lambda_i \prod_{j=1}^{m_k} p_0^\tau({a_k}^j), & i = 1, \ldots, m_k \\ V_k^{-m_k} \prod_{j=1}^{m_k} p_0^\tau({a_k}^j), & i = 0 \end{cases} \end{aligned} \tag{116}$$

After inserting this into equation (31) and after a bit of normalizing one gets

$${\beta_k}^i = \begin{cases} \frac{e_i \lambda_i}{b + \sum_{j=1}^{m_k} e_j \lambda_j}, & i = 1, \ldots, m_k \\ \frac{b}{b + \sum_{j=1}^{m_k} e_j \lambda_j}, & i = 0 \end{cases} \tag{117}$$

where

$$e_i \triangleq P_G^{-1} \mathcal{N}\left(\nu_k^i; 0, S_k\right) \tag{118}$$

and

$$b \triangleq m_k \frac{1 - P_D P_G}{P_D P_G V_k} \tag{119}$$

The likelihood of the filter is given by

$$\begin{aligned}
\Lambda(k) &\triangleq P\left\{Z(k)|m_k, Z_{k-1}\right\} \\
&= P\left\{z_1(k), \ldots, z_{m_k}(k)|m_k, Z_{k-1}\right\} \\
&= P\left\{\nu_1(k), \ldots, \nu_{m_k}(k)|m_k, Z_{k-1}\right\} \\
&= V_k^{-m_k}\gamma^0(m_k)\prod_{j=1}^{m_k} p_0(a_k{}^j) \\
&\qquad + V_k^{-m_k+1}\prod_{j=1}^{m_k} p_0(a_k{}^j)\sum_{j=1}^{m_k} P_G^{-1}\mathcal{N}\left(\nu_k{}^j; 0, S_k\right)\lambda_j\gamma^j(m_k) \\
&= V_k^{-m_k}\gamma^0(m_k)\prod_{j=1}^{m_k} p_0(a_k{}^j) \\
&\qquad + V_k^{-m_k+1}\prod_{j=1}^{k} p_0(a_k{}^j)\sum_{j=1}^{m_k} P_G^{-1} e_j \lambda_j \gamma^j(m_k)
\end{aligned} \tag{120}$$

where $\gamma_j(m_k)$ is given by the nonparametric clutter model from (40) and hereby the final result is

$$\Lambda(k) = \left(b + \sum_{j=1}^{m_k} \lambda_j e_j\right) \frac{P_D P_G V^{-m_k+1}}{m_k} \prod_{j=1}^{m_k} p_0(a_k{}^j) \tag{121}$$

4.5.2 The IMM-PDAFAI Algorithm

In the implementation of the IMM-PDAFAI, a few modifications are necessary. The first is that the same number of measurements is required for all models. This is done by using a common gating region or a union of all the gates. In practice this is done by taking the gate with the largest region. Ideally, one would apply the PDAFAI separately to each model; however the likelihood functions for each model have to have the same number of measurements in order to apply the equations. The likelihood function is calculated for $\mathcal{M}_j(k)$, model j, at time k

$$\begin{aligned}\Lambda_j(k) &\triangleq P\left\{Z(k)|\mathcal{M}_j(k), m_k, Z_{k-1}\right\} \\ &= V_k^{-m_k}\gamma^0(m_k)\prod_{l=1}^{m_k} p_0({a_k}^l) \\ &\qquad +V_k^{-m_k+1}\prod_{l=1}^{m_k} p_0({a_k}^l)\sum_{l=1}^{m_k} P_G^{-1}\mathcal{N}\left({\nu_k}^{j,l}; 0, {S_k}^j\right)\lambda_l\gamma^l(m_k)\end{aligned} \tag{122}$$

where

$$\nu_k^{j,l} = z_k^l - \hat{z}_{k|k-1}^j \tag{123}$$

where z_k^l is the l^{th} measurement at time k and the measurements are the same for all of the models. The prior estimated measurement for the j^{th} model is $\hat{z}_{k|k-1}^j$. The measurement covariance matrix for the j^{th} model is given by ${S_k}^j$.

To put it in a slightly different and more comprehensible manner

$$\Lambda_j(k) = \begin{cases} \left(b_j + \sum_{l=1}^{m_k}\lambda_l e_{j,l}\right)\frac{P_D P_G V^{-m_k+1}}{m_k}\prod_{j=1}^{m_k} p_0({a_k}^j), & \text{“target model”} \\ V_k^{-m_k}\prod_{j=1}^{m_k} p_0({a_k}^j), & \text{“no-target model”}\end{cases} \tag{124}$$

where

$$e_{j,i} \triangleq P_G^{-1}\mathcal{N}\left(\nu_k^{j,i}; 0, {S_k}^j\right) \qquad i = 1, \ldots, m_k \tag{125}$$

and

$$b_j \triangleq m_k\frac{1 - P_D P_G}{P_G P_D V_k} \tag{126}$$

A summary of the IMM-PDAFAI for the filtering stage is given in Table 4. The rest of the algorithm's stages are the same as in Table 3.

4.5.3 Track Formation with the IMM-PDAFAI Algorithm

Lets assume that one is using the `sliding-window` version of the algorithm. This means that at time k and $k+1$ unclaimed measurements are used to initiate new tracks. The IMM-PDAFAI is employed on the resulting pairs from $k+2$. At stages k and $k+1$ where there is no original uncertainty

Filtering $(\forall j \in M)$**:**

Prediction Equations

- state prediction: $\hat{x}_j(k|k-1) = F_j(k-1)\hat{x}_{oj}(k-1|k-1)$
- covariance prediction:

$$P_j(k|k-1) = F_j(k-1)P_{oj}(k-1|k-1)F_j(k-1)^T + G_j(k-1)Q_j(k-1)G_j(k-1)^T$$

- residual covariance: $S_j = H_j P_j(k|k-1)H_j^T + R_j$
- filter gain: $W_j = P_j(k|k-1)H_j^T S_j^{-1}$
- Gate size $V_k^j = c_{n_z}\gamma^{n_z/2} \cdot \left|{S_k}^j\right|^{1/2}$

The largest gate region is defined as $V_k = \max_j \left\{{V_k}^j\right\}$

Validation of Measurements (only for "no-target" models)

$$Z(k) = \left\{{z_k}^i\right\}_{i=1}^{m_k}$$

is the set of measurements that fall in union the regions defined by

$$\tilde{V}_k^j = \left\{z : \nu_k^{j,i\ T}\left[{S_k}^j\right]^{-1}{\nu_k}^{j,i} < \gamma\right\}$$

where

$${\nu_k}^{j,i} = {z_k}^i - \hat{z}^j_{k|k-1} \qquad i = 1, \ldots, m_k$$

and m_k is the number of validated measurements.

Calculate $\beta_k^{j,i}$ $\qquad i = 1, \ldots, m_k$

$$\lambda_i \triangleq \frac{p_1^\tau({a_k}^i)}{p_0^\tau({a_k}^i)} \qquad i = 1, \ldots, m_k$$

$$b_j \triangleq m_k \frac{1 - P_D P_G}{V_k P_G P_D}$$

$$e_{j,i} \triangleq P_G^{-1}\mathcal{N}\left(\nu_k^{j,i}; 0, {S_k}^j\right) \qquad i = 1, \ldots, m_k$$

$$\beta_k^{j,i} = \begin{cases} \frac{e_{j,i}\lambda_i}{b_j + \sum_{l=1}^{m_k} e_{j,l}\lambda_l}, & i = 1, \ldots, m_k \\ \frac{b_j}{b_j + \sum_{l=1}^{m_k} e_{j,l}\lambda_l}, & i = 0 \end{cases}$$

Measurement update

If $m_k = 0$ **then:**

State update: $\hat{x}_j(k|k) = \hat{x}_j(k|k-1)$
Covariance update: $P_j(k|k) = P_j(k|k-1)$
Likelihood function: $\forall j \qquad \Lambda_j(k) = 1$
If $\mathcal{M}_0$="no target" and $\mathcal{M}_1$='target' then
$\Lambda_0(k) = 1 - P_{FA}$ and $\Lambda_1(k) = 1 - P_D$

Else:

Combined Innovation: $\nu_j(k) = \sum_{i=1}^{m_k} \beta_k^{j,i}\nu_k^{j,i}$
Kalman Gain: $W_j(k) = P_j(k|k-1)H_k^T[S_k^j]^{-1}$
State update: $\hat{x}_j(k|k) = \hat{x}_j(k|k-1) + W_j(k)\nu_j(k)$
Covariance update: $\widetilde{P}(k) \triangleq W_j(k)\left[\sum_{i=1}^{m_k} {\beta_k}^{j,i}{\nu_k}^{j,i}\nu_k^{j,i\ T} - \nu_j(k)\nu_j(k)^T\right]W_j(k)^T$
$P_j(k|k) = P_j(k|k-1) - \left[1 - \beta_k^{j,0}\right]W_j(k){S_k}^j W_j(k)^T + \widetilde{P}(k)$
Likelihood function: Equation (124)

- mode probability: $\mu_j = \frac{\bar{\mu}_j\Lambda_j}{\sum_i \bar{\mu}_i\Lambda_i}$

$\blacksquare$

Table 4: Summary Of The Filtering Stage for the IMM-PDAFAI Algorithm (One Cycle)

for an initiating pair, the likelihood function relies only on the amplitude information:

$$\Lambda_1(k) = p_1^{\tau}(a_k) \quad \text{for the “target model”} \tag{127}$$

$$\Lambda_0(k) = p_0^{\tau}(a_k) \quad \text{for the “no-target model”} \tag{128}$$

The initial mode probabilities, for an initial pair at time k, are assumed to be

$$\mu_1(k) = 0.5 \tag{129}$$

$$\mu_0(k) = 0.5 \tag{130}$$

where the mode probabilities are updated according to equations (83) and (96). This algorithm is not in fact a Multi-Target tracking algorithm, but some ad hoc changes can be made in order to track more than one target when it is known that there is only one target

4.5.4 Common Problems with the Algorithm and their Solutions

No initial pairs The threshold is lowered until there exist legitimate pairs to start the process.

Too many initial pairs The threshold is raised until the number of initial tracks drops below a predetermined threshold. Too many tracks are a problem as this algorithm isn't strictly a multiple model tracking algorithm. This algorithm can theoretically track more than one target as long as the tracks don't cross over each other.

All of the tracks get killed off during the tracking process The threshold is dropped until there exist initial pairs to carry on the tracking process.

Tracking gate gets flooded with measurements for a single track When the output covariance matrix S_k as defined in Equation. (11) grows, so does the gate size. Then, the gate becomes so large, it takes up all of the measurements including measurements that belong to other tracks. This happens because the more measurements that enter the gate the more spread, there is between the innovations, consider Equation. 53. Unfortunately, this ends up like a rolling snowball. Things can only get worse. The simplest solution is to not allow tracks with large output covariance matrices access to any measurements.

The number of total tracks crosses a reasonable number When there are too many tracks in a small area, they start stealing measurements from each other. That means that even legitimate tracks are sometimes denied access to their measurements and are killed off because of

this. Also, a large number of tracks is an unnecessary computational burden. A large number of tracks typically occurs when there is no target at all or when the target gets lost in the clutter. Our strategy is to stop the tracking process when this happens. In order to take into account the size of the image, the criteria for exiting the tracking algorithm is the following:
If $p_f > 0.03$ or $ImageSize * p_f > 100$, then exit.
This exit strategy is used most often when there is no target and the algorithm is tracking only clutter.

5 Hyperspectral Movies

The main focus of the thesis is to examine the effect of different detection algorithms and assumed target signatures on the tracking process when applied to Hyperspectral movies. Lets assume that the target was implanted in a manner described by L. Varsano, I. Yatskaer and S.R. Rotman, "Tracking Point Targets in Hyperspectral Data", accepted by Opt. Eng.

$$\mathbf{u}(m,n)' = \mathbf{u}(m,n) + \theta\mathbf{t} \tag{131}$$

where $\mathbf{t}$ is the target signal vector θ is the relative intensity. There are different types of detection. The first is when the signal is known; in that case, the matched filter is used. The equation for the matched filter is:

$$MF(m,n) = \mathbf{t}^T\mathbf{\Sigma}^{-1}\big(\mathbf{u}(m,n) - \widehat{\mathbf{u}}(m,n)\big) \tag{132}$$

The second is when the signal is unknown; then, we use the RX filter:

$$RX(m,n) = \big(\mathbf{u}(m,n) - \widehat{\mathbf{u}}(m,n)\big)^T\mathbf{\Sigma}^{-1}\big(\mathbf{u}(m,n) - \widehat{\mathbf{u}}(m,n)\big) \tag{133}$$

6 System Design and Analysis

Designing a system is a bit like tailoring. Anyone with a bit of cloth and a needle and thread can make a suit, but it won't necessarily be worth anything if it is not suited to the client. There are a bewildering number of possibilities to choose from. The only way to make heads and tails out of the designing problem is to have a trustworthy method for comparing and analyzing the systems performance.

I suggest two different score methods. The first is simple. If the real targets track is known, then the test is simple. Usually there are a few possible tracks at the end of the process. Only tracks that have a lifetime of more than 50% are taken into consideration. Then the track with the highest TTP 100 is the winning track. The next step is to compare the track with the real track. This is done by calculating the MSE between the estimated track and the real track position. We do this from the creation of the track denoted as t_{birth} until the end of the tracking process.

$$MSE = \frac{1}{N} \sum_{n=t_{\text{birth}}}^{N} \sqrt{\left(r_x(n) - \hat{r}_x(n)\right)^2 + \left(r_y(n) - \hat{r}_y(n)\right)^2} \tag{134}$$

where $r_x(n)$ is the real position on the x-axis and $\hat{r}_x(n)$ is the estimated position on the x-axis. The same applies for $r_y(n)$ and $\hat{r}_y(n)$. If $MSE < 1$, then the track is valid.

The second method is based on the TTP. First, using the real track, we find the real track if it exists. The TTP of the real track is termed TTP_{true}. If TTP_{true} doesn't exist, then it is set to zero. Then the maximum of the remaining false tracks is termed $\max\{TTP_{false}\}$. If there are no false tracks, then $\max\{TTP_{false}\}$ is set to zero. Then the criterion is the ratio between the two. However, in order to get a score between $0-1$ the following equation is used

$$\frac{2}{\pi} \arctan\left(\frac{TTP_{true}}{\max\{TTP_{false}\}}\right) \tag{135}$$

If the score is equal or larger than 0.5, then the tracking is successful; a failure occurs for scores less than 0.5.

Because the results are dependent on the scenario, we take ten different frames and repeat the process, then we average the scores from both tests. We refer to the first method as the success rate and to the second test as the score rate.

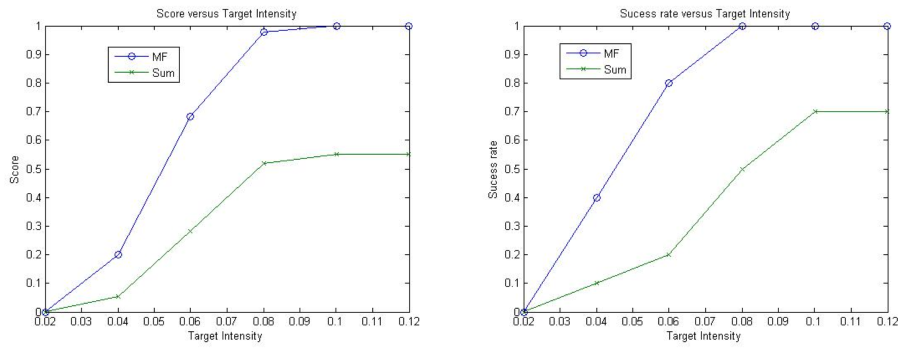
These are tests for when the target is known to be in the picture. When the tracking algorithm is applied to the same picture with no target implanted, then ideally no false tracks show up. For a no-target picture we expect a score or success rate of zero. This would mean that no high value TTP tracks have survived for over 50% of the simulation.

6.1 Tracking Curves

The main point of the this tracking algorithm is to track targets at a low SNR level. It is important to have a yardstick to compare different algorithms performance or even the same algorithmic performance but with different parameters, for different levels of SNR. This is easy with simulated data. The target is inserted at different levels of intensity. For a higher intensity one expects a higher average score or success rate. The difference between algorithms is, first, the minimum level of target intensity that results in a score or success rate above zero; second, the rate that the score or success rate rises. In fig. 7 two different methods of collapsing the hyperspectral cube are compared. The first is the dot product between the pixel vector and the target vector

$$\mathbf{t}^T \left(\mathbf{u}(m,n) - \widehat{\mathbf{u}}(m,n)\right) \tag{136}$$

and the second is the matched filter from Equation. (132).



(a) (b)

Figure 7: The TOC curves: (a) The average score rate versus target intensity, (b) The average success rate versus target intensity

It is clear that the matched filter is better then merely projecting the target onto the target signal.

7 Tracking results for Synthetic Data Produced by Irena and Louisa

The synthetic data here is taken from na23a. It was converted into a Hyperspectral movie by using a cloud signature and sky signature. Research on converting IR movies into hyperspectral movies has been done by Louisa Varsano, continued by Irena and is explained in [14]. A target signal was implanted diagonally in time in the movie. The target signal is spread over a few pixels. Three ways of collapsing the hyperspectral movie are:

Test 2

$$\mathbf{1}^T\left(\mathbf{u}(m,n)-\widehat{\mathbf{u}}(m,n)\right) \tag{137}$$

Test 2 half

$$\mathbf{t}^T\left(\mathbf{u}(m,n)-\widehat{\mathbf{u}}(m,n)\right) \tag{138}$$

Test 3

$$\mathbf{t}^T\boldsymbol{\Sigma}^{-1}\left(\mathbf{u}(m,n)-\widehat{\mathbf{u}}(m,n)\right) \tag{139}$$

The results for the Irena movies are shown in Table 5:

		Test 2	Test 2 half	Test 3
Irena Movie	Success Rate	1	1	1
	Score Rate	1	1	1
Stanley Movie	Maximum TTP	0.665	0.6653	0.1111

Table 5: The Table of results for the Irena Movie

Here is an example of a target path and tracking result from an Irena movie with Test 3. in Fig. 8.

Note that I compare the results to the Stanley movies. The Stanley movies are movies without targets implanted. The score for evaluating the Stanley Movies is the maximum TTP amongst the tracks that survive for over 50% of the tracking process. The Maximum TTP for the Stanley movies have to be very low, i.e. , at the very least, lower than 0.9. For example in Table 5 we get 0.665 for Test 2 on a Stanley movie. This means that 6 of the 9 movies have a maximum TTP of almost 1. In other words we have a false track rate of 6 out 9 when there is no target. For a real system this is not acceptable Otherwise this means that the algorithm will track clutter when there are no targets which isn't very useful.

The Louisa movies are divided into 4 different types according to the intensity of the target. The intensity goes from 1 to 4 with 1 being the most intense.The results are shown in the following tables:

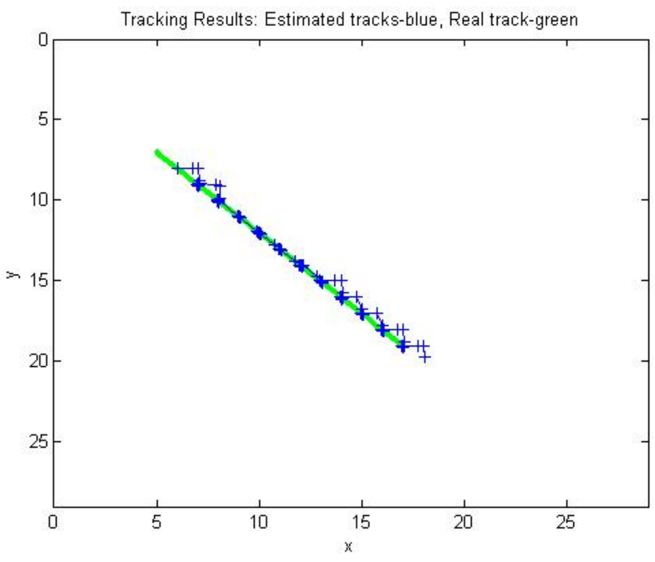


Figure 8: The target starts in left upper corner and proceeds to move diagonally over the frame. The zigzag motion is due to the pixel entering and leaving a pixel.The green line is the path of center the target and the blue line is the result of the tracking algorithm.

		Test 2	Test 2 half	Test 3
Louisa Movie	Success Rate	0.7778	1	1
	Score Rate	0.6881	0.9142	0.9142
Stanley Movie	Maximum TTP	0.665	0.6653	0.1111

Table 6: The Table of results for the Louisa Movies ,Intensity 1

		Test 2	Test 2 half	Test 3
Louisa Movie	Success Rate	0.5556	0.6667	0.6667
	Score Rate	0.4937	0.6115	0.6115
Stanley Movie	Maximum TTP	0.665	0.6653	0.1111

Table 7: The Table of results for the Louisa Movies ,Intensity 2

		Test 2	Test 2 half	Test 3
Louisa Movie	Success Rate	0	0.4444	0.4444
	Score Rate	0	0.3889	0.3889
Stanley Movie	Maximum TTP	0.665	0.6653	0.1111

Table 8: The Table of results for the Louisa Movies ,Intensity 3

The only test that provides acceptable results for both movies is Test 3. The reason is that the clutter isn't white for Test 2 and Test $2\frac{1}{2}$ therefore the

		Test 2	Test 2 half	Test 3
Louisa Movie	Success Rate	0	0	0
	Score Rate	0	0	0
Stanley Movie	Maximum TTP	0.665	0.6653	0.1111

Table 9: The Table of results for the Louisa Movies ,Intensity 4

algorithm locks on to clutter usually edges of clouds. The tracking results for Test $2\frac{1}{2}$ were good and usually equal to those of Test 3. However the results when no target was implanted are not good for Test $2\frac{1}{2}$. It might be possible to eliminate the problem by better pre-processing methods, then Test $2\frac{1}{2}$ might provide better results also when there is no target. In order to be worth the effort a pre-processing method would have to be more efficient than estimating the covariance matrix.

It can be seen from the Louisa movies that a higher intensity target has a higher chance of being tracked.

8 Conclusions

There are five principal issues which have been addressed in order to get the IMM approach to work in a high clutter environment.

- The pre-processing level is vital to successful tracking among the more common approaches we have applied linear filtering and ordered statistics filter. In the future, beside searching for the best method of feature extraction, we should consider combining different methods of feature extraction/pre-processing methods.
- Normalization of background noise and time dynamic thresholding;
  - Normalization of background noise-The normalization of the background noise is important for the tracking process. This was done by dividing each pixel by a constant and the local standard deviation around each pixel. A method for calculating the constant was described.
  - Time dynamic thresholding- Raising or lowering the threshold in the case of too many tracks or no tracks at all.
- When tracking in an environment with a low SNR it is necessary to continually track a number of high probability tracks simultaneously. That is a good way to keep tracking the target if its probability drops. This makes it possible to regain acquisition of the target, if it disappears and then reappears. In fact this is the same approach taken for the track initiation stage. In future work, the current algorithm, which is better suited to a single target, can be improved significantly by using the multiple target approach. The various ways to do this are the JPDAF or MHT.
- The conclusions for the Collapsed Hyperspectral movies are as expected. The higher the intensity of the implanted target, the better chance there is of tracking it. In order to reduce the probability of tracking clutter, the clutter must be whitened. The results show that the Matched Filter does a good job at this.
- The model used for the amplitude of the target is in fact crucial for successfully tracking the target. This point target is in general, difficult to model; however, a satisfactory solution was found for this problem.

9 Bibliography

References

[1] O. Raviv and S.R. Rotman, "Improved filter for point target detection in multidimensional imagery", *Signal and Data Processing of Small Targets*, Proceedings of SPIE Vol. 5159, 2003, pp. 32-40.

[2] S.Blackman and R.Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House Radar Library, 1999.

[3] J.B.Moore, and B.D.O.Anderson, *Optimal Filtering*, Prenctice-Hall, Inc., Englewood Cliffs, N.J., 1979.

[4] M.S.Grewal, and A.P. Andrews, *Kalman Filtering*, Prentice Hall, Englewood Cliffs, N.J., 1993.

[5] Y.Bar-Shalom, and T.E.Fortmann, *Tracking and Data Association*, Academic Press, Mathematics in Science and Engineering, 1988.

[6] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, 2nd Ed. 1994, Addison Wesly Longman.

[7] Y. Bar-Shalom, and Xiao-Rong Li, *Multitarget-Multisensor Tracking: Principles and Techniques* YBS publishing, 1995.

[8] R.A. Singer, "Estimating Optimal Tracking Filter Peformance for manned Maneuvering Targets", *IEEE Trans. Aerospace & Electronic Systems*, Vol AES-9 (2), (Feb 1970),pp 473-483.

[9] J.P. Helferty, "Improved Tracking of Maneuvering Targets: The Use of Turn-Rate Distributions for Acceleration Modeling", *IEEE Trans. Aerospace & Electronic Systems*, Vol AES-32 (4), (Oct 1996),pp 1355-1361.

[10] X.Rong. Li and Y. Bar-Shalom, "Design of an Interacting Multiple Model Algorithm for Air Traffic Control Tracking ", *IEEE Trans. Control Systems*, Vol CS-1 (3), (Sep 1993),pp 186-194.

[11] H.A.P Blom, and Y. Bar-Shalom, "The Interacting Multiple Model Algorithm for Systems with Markovian switching coefficients", *IEEE Trans. Automatic Control*, Vol AC-33 , (Aug 88),pp 780-783.

[12] X.Rong. Li and Y. Bar-Shalom, "Peformance Prediction of the Interacting Multiple Model Algorithm", *IEEE Trans. Aerospace & Electronic Systems*, Vol AES-29 (3), (Jul 1993),pp 755-771.

[13] D. Lerro and Y. Bar-Shalom, “Interacting Multiple Model Tracking with Target Amplitude Feature”, *IEEE Trans. Aerospace & Electronic Systems*, Vol AES-29 (2), (April 1993),pp 494-509.

[14] L. Varsano, I. Yatskaer and S.R. Rotman, ” Point target tracking in Hyperspectral Images”, accepted for publication in Opt. Eng.

Appendixes

A Calculating P_G explicitly

I will show how to explicitly calculate the gate probability P_G for $n_z = 1, 2, 3$. The first dimension is easy

$$\begin{aligned}
P_G &= \frac{1}{\sqrt{2\pi}} \int_{-g}^{g} \exp\left\{-\frac{1}{2}z^2\right\} dz \\
&= \frac{1}{\sqrt{2}} \frac{2}{\sqrt{\pi}} \int_{0}^{g} \exp\left\{-\frac{1}{2}z^2\right\} dz \\
&= \frac{2}{\sqrt{\pi}} \int_{0}^{g} \exp\left\{-\frac{1}{2}z^2\right\} \frac{dz}{\sqrt{2}} \\
&= \frac{2}{\sqrt{\pi}} \int_{0}^{g/\sqrt{2}} \exp\left\{-t^2\right\} dt \\
&= \operatorname{erf}\left(\sqrt{\frac{\gamma}{2}}\right) \qquad (140)
\end{aligned}$$

For the second dimension $n_z = 2$

$$\begin{aligned}
P_G(\gamma) &= \frac{1}{(2\pi)^{2/2}} \iint_{\{\tilde{z} \in \tilde{V}_{k+1}\}} \exp\left\{-0.5(z_1^2 + z_2^2)\right\} dz_1 dz_2 \\
&= \frac{1}{(2\pi)} \iint_{\{z_1^2 + z_2^2 \leq \gamma\}} \exp\left\{-0.5(z_1^2 + z_2^2)\right\} dz_1 dz_2 \\
&= \frac{1}{(2\pi)} \iint_{\{r \leq \sqrt{\gamma}\}} \exp\left\{-0.5r^2\right\} dr d\phi \\
&= \frac{1}{(2\pi)} \int_{0}^{\sqrt{\gamma}} r \exp\left\{-0.5r^2\right\} dr \int_{0}^{2\pi} d\phi \\
&= \int_{0}^{\sqrt{\gamma}} \exp\left\{-0.5r^2\right\} d\left(\frac{r^2}{2}\right) \\
&= 1 - \exp\left\{-\frac{\gamma}{2}\right\} \qquad (141)
\end{aligned}$$

This is done by using the following transform from cartesian coordinates to polar coordinates.

$$\begin{aligned}
z_1 &= r\cos\phi \qquad (142) \\
z_2 &= r\sin\phi \qquad (143)
\end{aligned}$$

For the third dimension

$$
\begin{aligned}
P_G(\gamma) &= \frac{1}{(2\pi)^{3/2}} \iiint_{\{\tilde{z}\in\tilde{V}_{k+1}\}} \exp\left\{-0.5(z_1^2+z_2^2+z_3^2)\right\} dz_1dz_2dz_3 \\
&= \frac{1}{(2\pi)^{3/2}} \iiint_{\{z_1^2+z_2^2+z_3^2\le\gamma\}} \exp\left\{-0.5(z_1^2+z_2^2+z_3^2)\right\} dz_1dz_2dz_3 \\
&= \frac{1}{(2\pi)^{3/2}} \iiint_{\{r\le\sqrt{\gamma}\}} \exp\left\{-0.5r^2\right\} r^2dr\sin\theta d\theta d\phi \\
&= \frac{1}{(2\pi)^{3/2}} \int_0^{\sqrt{\gamma}} r^2\exp\left\{-0.5r^2\right\} dr \int_0^{\pi}\sin\theta d\theta \int_0^{2\pi} d\phi \\
&= \frac{2}{(2\pi)^{1/2}} \int_0^{\sqrt{\gamma}} r^2\exp\left\{-0.5r^2\right\} dr \qquad (144)
\end{aligned}
$$

Note that once again I transformed the cartesian coordinates to spherical coordinates with the following transformation

$$
\begin{aligned}
z_1 &= r\cos\phi\sin\theta && (145)\\
z_2 &= r\sin\phi\sin\theta && (146)\\
z_3 &= r\cos\theta && (147)
\end{aligned}
$$

Expression (144) can be turned into an incomplete gamma function by a change of variables. The incomplete gamma function is defined as

$$
P(x,a) = \frac{1}{\Gamma(a)}\int_0^x t^{a-1}e^{-t}dt \qquad (148)
$$

A change of variables is carried out by defining $t=\frac{1}{2}r^2$ and then we get

$$
\begin{aligned}
P_G(\gamma) &= \frac{2}{(2\pi)^{1/2}} \int_0^{\sqrt{\gamma}} r\exp\left\{-0.5r^2\right\} d\left(\frac{1}{2}r^2\right) \\
&= \frac{2}{(2\pi)^{1/2}} \int_0^{\gamma/2} \sqrt{2t}\exp\left\{-t\right\} dt \\
&= \frac{2}{\sqrt{\pi}} \int_0^{\gamma/2} t^{1/2}\exp\left\{-t\right\} dt \\
&= \frac{2}{\sqrt{\pi}}\Gamma\left(\frac{3}{2}\right)P\left(\frac{\gamma}{2},\frac{3}{2}\right) \\
&= P\left(\frac{\gamma}{2},\frac{3}{2}\right) \qquad (149)
\end{aligned}
$$

where $\Gamma\left(\frac{3}{2}\right)=\frac{\sqrt{\pi}}{2}$

Inductively it can be shown that P_G as a function of γ and n_z is

$$P_G(\gamma, n_z) = P\left(\frac{\gamma}{2}, \frac{n_z}{2}\right) \tag{150}$$