Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 146925, 19 pages
doi:10.1155/2008/146925

Research Article

Parameter Adjustment for a Dynamic Programming Track-before-Detect-Based Target Detection Algorithm

O. Nichtern and S. R. Rotman

Department of Electro-Optics, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

Correspondence should be addressed to O. Nichtern, nichtern@ee.bgu.ac.il

Received 28 March 2007; Revised 13 November 2007; Accepted 29 January 2008

Recommended by Roy Streit

This paper addresses the problem of tracking a dim moving point target in a sequence of IR images. The proposed tracking system, based on the track-before-detect (TBD) approach, is designed to track and detect dim maneuvering targets from an image sequence under low SNR conditions; a dynamic programming algorithm (DPA) is used to process the frames in the sequence. This paper deals with the practical issues of setting the parameters of our algorithm to maximize tracking capability. In our flexible approach, we are able to accommodate different levels of noise and different target velocities and mobilities. Our algorithm is tested on real data to determine its efficacy.

1. INTRODUCTION

The detection and tracking of point targets is necessary in many military applications. One particular use is in IRST (infrared search and track) systems where the data from the infrared spectra is used to detect targets which can be moving at subpixel velocities and subtend areas less than a pixel. Various researchers have developed track-before-detect (TBD) methods based on the dynamic programming algorithms (DPA) [1–5].

In the DPA approach, one continually incorporates new data (consisting of new infrared images or frames) into an estimate of where the targets may be. Each pixel is considered a potential target; a score is assigned to how target-like the pixel is. No attempt is made to continually decide definitely where the target is; that decision is made only when called upon or when the data ceases to flow. Hence this methodology is called track-before-detect.

In this paper, we will examine the practical problems involved in implementing a DPA on real data involving clouds in the sky (as clutter) and jet airplanes (as targets). We will note how practical examination of the data allows us to determine what the ratio between the reliability given to the new data compared to that associated with the accumulated score should be. Results will be shown on real data.

2. POINT TARGET DETECTION AND TRACKING IN AN IR SEQUENCE

2.1. *General architecture*

The system performs target detection and tracking in three steps. The first step is whitening preprocess, which suppresses the clutter and emphasizes the target [6]. In this paper, we will assume that it is done by the simple antimedian filter where we subtract from each pixel an estimate based on the mean of the surrounding neighboring pixels. The second step is target detection and tracking based on the TBD approach and implemented using the DPA. The third step is the decision of the target presence and position after the last frame and retrieval of its trajectory. Currently, the pixel with the highest accumulated score in the last processed frame is selected as the target. Future research can involve the usage of a CFAR (constant false alarm rate) algorithm using known null hypothesis probabilities. The general system architecture is given in Figure 1; we concentrate in this paper on the DPA stage.

2.1.1. *The target model*

The long range detection to which IRST systems are dedicated implies a point-like target shape. For this reason, the

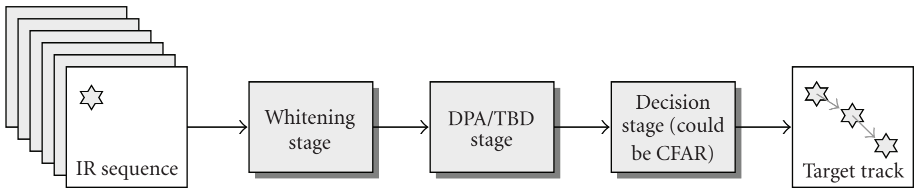


FIGURE 1: Tracking system architecture.

spatial target signature, as projected onto the sensors array focal plane, is dominated by the optics point spread function (PSF). Being dependent on the range to the target and the aspects angle between the target and detector array, the target PSF is not constant. Still, the sampled target signature is often modeled as Gaussian:

$$S(x,y) = \gamma\cdot e^{-1/2}\left(\left(\frac{x-x_0}{\sigma_x}\right)^2 + \left(\frac{y-y_0}{\sigma_y}\right)^2\right), \qquad (1)$$

where x_0 and y_0 are the possibly fractional targets' coordinates (pixel coordinates of the signature in the focal plane), $\gamma\psi$ is the peak intensity at $(x_0, \psi y_0)$, and $\sigma_x^2\psi$ and $\sigma_y^2\psi$ are the targets' horizontal and vertical radiance variances, respectively [7, 8]. Both the detected target radiation intensity and its PSF depend on the wavelength to which the detectors are sensitive, the target specifications, for example, as hot body tracking or plume tracking and the atmospheric transmission window (e.g., the 3–5 [μm] MWIR is often used for image target tracking systems).

2.1.2. IR background scene and atmospheric effects

The background of interest is composed of skies, either clear or cloudy. The sky primary source of radiation in the IR band is of course the sun. The amount of sun radiation that reaches the sensor array depends on the number of atmospheric layers the radiation pass through and the angle between the sun and the sensing system. For low sun angles or many atmospheric layers, less radiation will be collected by the sensors. Another major source of illumination is the sky light, which is scattered sun radiation but has different spectral distribution (i.e., its peak is in the visible blue spectrum). Absorption is the most serious problem faced by IR systems and it actually defines the wavelength "window" at which IR systems work. Within these windows the received radiation is affected by clouds and haze, fog and atmospheric gases such as ozone and carbon dioxide (although the later influence on the radiation within the near atmosphere is minimal). It is scattered away from the line of sight by aerosols suspended in the atmosphere and temperature changes along the radiation path can cause scintillation, which is a severe problem that IRST systems have to deal with, this phenomena occurs mostly near sea level. Fog and clouds are strong scatters and can be opaque to distant IR targets radiation. The presence and characteristics of fog, haze, and clouds depend on the geographic region, as well as on the season and altitude; hence fog is more probable in regions with high humidity, whereas dust can be expected in dry areas. There exist several computer models such as LOWTRAN (low spectral resolution) or MODTRAN (moderate spectral resolution) that serve to model and forecast atmospheric conditions.

Although there exists no unique model for the IR sky background scene, many models assume a first-order Markov process and describe elevation and azimuth correlation to model cloud clutter analytically.

2.2. The DPA stage

An IR sequence containing background noise and clutter Section 2.1.2, and a subpixel maneuveringtarget are the bases of our data model (first-order hidden Markov model), where the target track is the hidden sequence of events, and the IR sequence frames are the observed data. Barniv [9, 10] has suggested a method for using the DPA in target tracking, which was further developed by Arnold [11], to allow tracking where nonwhite Gaussian noise (WGN) is present. The algorithm presented in this paper further develops the ideas presented in [2–4].

The DPA aids in the tracking of a dim point target maneuvering in a noisy background. The noise is assumed to be temporally and spatially uncorrelated (after the whitening stage); the target on the other hand moves in a smooth manner with only gradual changes in direction and speed. Deviations from either of these two assumptions will degrade the eventual system performance.

The algorithm processes a batch of frames rather than a single frame. In every frame, each pixel is given an accumulated score composed of its score in the present frame (after the whitening stage), and an accumulated score of the previous frames. The score is calculated using an accumulated score function (ASF) which will be introduced in Section 2.2.1. These accumulated scores of the pixels in the processed batch, as well as the calculated direction Section 2.2.3 of the pixels are stored in the *accumulated score matrix* (ASM) for every processed frame.

During the processing of the batch frames, a calculation is made to find the origin of each pixel within the previous frame thus forming a *tracking matrix* Section 2.2.4, and the estimated direction of the target Section 2.2.3. At the last processed batch frame, a pixel with the highest accumulated score is assumed to be the target, and using the *tracking matrix,* its trajectory is found by backward calculation Section 2.2.4.

2.2.1. Accumulated score matrix

As stated, the accumulated score matrix (ASM) holds the accumulated score of each pixel calculated using the accumulated score function (ASF), and the direction from which the pixel has originated.

The ASF is composed of three components: (1) pixel score in the current whitened frame (WF), and (2) and (3) are two different functions of the pixel's accumulated score up to the previous frame, weighted by a parameter g. The two last components are both multiplied by a memory coefficient b, determining the past information influence on the ASF of the current frame. The two last components take a valid search area, or matrix, from which the target may have arrived from in the previous frame. The first method, called the max method, or $\alpha(x, y)$, takes the maximum of the ASF elements within the valid search area. The second method, called sum method, or $\beta(x, y)$, takes the summation of the ASF elements within the valid search area.

For clear tracks, the maximum value (1st method, maximum) will allow an accurate evaluation of the track; when the target is very weak, tracking can be done through the tacking of the overall cloud being produced (2nd method, summation) rather than the maximum value from a single target. By combining and weighting these two approaches, coverage of all the possibilities might be achieved. This is an extension to the classic DPA where the evolution of the system is done using optimization alone, that is, always choosing the maximum value. The ASF is calculated as follows:

$$\mathrm{ASF}_n(x, y) = \mathrm{WF}_n(x, y) + b \cdot [g \cdot \alpha(x, y) + (1 - g) \cdot \beta(x, y)]. \tag{2}$$

We will assume that the target velocity is within the range of 0–2 [ppf] (pixels per frame) and a possible jitter of 0.5 [ppf]; thus, the pixel can move up to 2.5 [ppf] in the horizontal and vertical directions; hence a valid search area γ, from which a pixel might originate from in the previous frame is a 7×7 matrix. Given the target model above, $\alpha(x, y)$ and $\beta(x, y)$ are defined as

$$\begin{aligned} \alpha(x, y) &\equiv \sum_{i,j=-3}^{3} \gamma(x, y)_{i,j}, \\ \beta(x, y) &\equiv \max \left\{\gamma(x, y)_{i,j}\right\}_{i,j=-3,\dots,3}, \end{aligned} \tag{3}$$

where γ holds the valid search area elements ASF of the previous frame:

$$\gamma(x, y)_{i,j} = \mathrm{ASF}_{n-1}(i + x, j + y). \tag{4}$$

2.2.2. The probability matrix W

In (4), $\gamma(x, y)_{i,j}$ the ASF values of the previous frame were taken. Since the pixels in the valid search area have different probabilities of being the target in the previous frame, the ASF values of the previous frame can be multiplied by a weighting matrix that will take into account unreasonable changes in direction and velocity, and the sensor jitter. The following paragraphs will elaborate on the matter.

Since the target radiation spreads over several adjacent states (pixels), we are actually looking for the progress of this group of pixels. As mentioned in Section 2.2.1, the radiation spread is defined by the target's signature that is related to its shape and movement parameters as velocity and trajectory, by the sensor parameters as its sensitivity and integration time and by the platform jitter and movement. Since we are looking for the progress of a group of pixels, it is natural to combine their total energy into the score function, thus tracking the pixel to which the total contribution was maximal. Since not all of the contributing pixels have the same probability to be the main source of energy to the examined pixel, their position is assigned with a weighted factor according to the velocity/jitter/sensor PSF assumptions.

This point can be clarified by the following 1D example: consider a subpixel size target that captures a quarter of the pixel FOV moves horizontally at 0.25 pixels per frame and is captured by a platform whose jitter distribution is estimated to be horizontal with U $[-1, +1]$ (uniform PDF over the range of -1 to $+1$ pixels). Because of uncertainty in the targets' origin, it can be captured by either the same pixel, with probability of 3/4; or by the next adjacent pixel, with probability of 1/4. Because of the sensor jitter, each pixel can capture either its former FOV or, at most, one of its horizontally adjacent FOVs. Therefore, the origin of the current target can be any one of four different pixels in the former frame, each with a different probability. Now consider a 1 by 3 Gaussian target. This adds to the number of possible origins two more positions, one to each horizontal side, resulting in total of 6 possible origins. An introduction to the subpixel detection problem is presented in [12].

Since the probability of the jitter and the probability of position due to velocity are independent, the overall position probability is the product of these two probabilities. Marking the position probability due to jitter (p_j) of the pixel in the θ location, and in the kth frame by $p_j(\theta_k)$, and the position probability due to target motion by $p_v(\theta_k)$ the weight is provided $wv(\theta_k) = p_v(\theta_k) \cdot p_j(\theta_k)$. For the $(k + 1)$th frame, the group of weights for former possible pixels from which a pixel might have originated is denoted by $\{wv(\theta_k)\}, i \in \Omega$; where Ω is the valid search area in the kth frame. Ω is a moving window, and its size is determined by the maximum allowed velocity plus the maximum shift due to the jitter. The jitter is a remnant of the imperfectly registered raw data at the input, which we assume to be given.

As mentioned in Section 1, the requirement to detect distant targets moving at subpixel velocities rationalizes the straight motion assumption. For 1D movement the set of probabilities $\{wv, i, k\}$ is a vector; the expansion to a 2D motion is straightforward, either by using the notation of Arnold and Barniv and dividing each observation into numerous cells, each representing data of a particular axis; or by computing a 2D set of probabilities $\{wv, i, k\}$; that is, let $\{wv, i, k\}$ be a matrix.

In our implementation, the valid search area will define a probability matrix W which contains the probabilities of pixels in the previous frames being the origin of the pixel in the current frame. The probability matrix is a superposition of two matrices—(1) a basic matrix W_{basic} and (2) a penalty matrix W_{Penalty}. The superposition is weighted by a parameter p.

W_{basic} includes the effect of Gaussian target spread and the sensor jitter. This matrix represents a target standing

9	8	7	6	5	4	3
10	9	7.5	6	4.5	3	2
11	10	9	6	3	1.5	1
12	12	12	12	0	0	0
11	10	9	6	3	1.5	1
10	9	7.5	6	4.5	3	2
9	8	7	6	5	4	3

Figure 2: 7×7 penalty matrix $W_{\text{Penalty}}(\text{direction} = 1)$ in the estimated direction $\rightarrow(0^\circ)$.

in place; the second matrix $W_{\text{Penalty}}(\text{direction})$ gives high probability to pixels in an assumed direction from which a target might have originated, and lower probabilities as the pixels vary from the assumed direction of arrival. Estimation of the movement in the current frame is discussed later in direction calculation Section 2.2.3.

We will assume a sensor jitter of U $[-0.5, +0.5]$ (uniform PDF over the range of -0.5 to $+0.5$ pixels), and a Gaussian target spread of 3×3. In our case W_{basic} elements, $w_{\text{basic},i,j}$, hold the probabilities $p_j(\theta_k)$ discussed above.

Instead of using the suggested $p_j(\theta_k)$ elements in W_{Penalty}, we use different implementations. $W_{\text{Penalty}}(\text{direction})$ gives small penalties for small turns and severely decrease the score for sharp turns. Succary et al. [2] suggested using a penalty parameter that increases as the turn angle increases. Using that method, a turn of 45 degrees leads to penalty of $-x$, 90 degrees to $-2x$, and so on. The parameter value used was found by empirical. The angles are symmetrical with respect to the axis of assumed direction. Figure 2 shows the 7×7 penalty matrix $W_{\text{Penalty}}(\rightarrow)(0^\circ)$. We have taken $x = 3$. The estimated region of arrival gets the highest values, 12 in this case. The values decrease until reaching the cells opposite to the estimated direction. There is a tradeoff between the need for penalty on change of direction and the need for adaptation to maneuvering targets.

The element values of the 7×7 $W_{\text{Penalty}}(\text{direction})$ matrix are normalized to probabilities using the denominator in (5). The matrix is then multiplied by the W_{frame} matrix which is used to adapt the penalty matrix to the assumed range of velocities by zeroing out all penalty pixels not contained in Ω (the valid search area in the kth frame) and keeping pixels within Ω. Hence W_{frame} is 1 within Ω and 0 otherwise.

There are nine different probability matrices for each of the possible assumed directions: 0°–315° at steps of 45° (directions 1–8, resp.), and remaining in place (direction 9). The probability matrix is given as

$$W(\text{direction}) = p \cdot W_{\text{basic}} + (1-p) \cdot \frac{W_{\text{Penalty}}(\text{direction}) \cdot W_{\text{frame}}}{\sum_{i,j} w_{\text{Penalty}}(\text{direction})_{i,j} \cdot w_{\text{frame};i,j}}. \tag{5}$$

Figure 3 shows the basic probability matrix (direction 9), the eight penaltymatrices for the different directions (direction 1–8), and the resulting nine probability matrices. The basic probability matrix incorporates the allowed jitter and represents a standing target. In Figure 3, the target is assumed to be at a velocity range of 0-1 [ppf]; thus only a 5×5 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 $W_{\text{Penalty}}(\text{direction})$. All matrices are normalized to values between 0-1.

Equation (4) can be rewritten to contain the probability matrix:

$$y(x,y)_{i,j} \equiv w(\text{direction})_{i,j} \cdot \text{ASF}_{n-1}(i+x, j+y), \tag{6}$$

where $w(\text{direction})_{i,j}$ is the (i,j) element in the probability matrix, W.

2.2.3. *Direction calculation*

To use $W(\text{direction})$, the direction from which a pixel originated has to be found. We must now decide where it is most likely that the target previously was when moving in this particular direction. Estimation of the direction is based on (1) ASF, (2) the basic probability matrix (W_{basic}), and (3) direction consistency.

The calculation of the direction starts by creating a temporary 7×7 matrix (given our assumption of target velocity and sensor jitter). The previous frame ASFs as well as the directions of pixels in the area of 7×7 are 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, Figure 4 shows the division of the 7×7 matrix into the eight different directions, direction 1–8). The parameter gives maximal value for identical directions (direction consistency) and minimal value for opposite directions (direction inconsistency). No direction difference sets Var $= 6$, and every increase of 45 degrees difference lowers the value of Var by 1 (down to a value of 2 for 180 degree difference). The final score of each pixel in the temporary matrix is defined as

$$\begin{aligned} &\text{Source\_Pixel\_Score}(x+i, y+j) \\ &\quad = \text{ASF}_{n-1}(x+i, y+j) \cdot w_{\text{basic};i,j} \cdot \text{Var}(i,j), \end{aligned} \tag{7}$$

where $\text{ASF}_{n-1}(x+i, y+j)$ is the pixel's accumulated score from the previous frame, $w_{\text{basic},i,j}$ is the value of W_{basic} at the (i,j) coordinate, and $\text{Var}(i,j)$ is the direction difference parameter for the (i,j) coordinate.

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 in the current frame is now known and is saved in the ASM. In conclusion, to calculate the pixel's direction:

(i) create a 7×7 temporary matrix;
(ii) find the direction of the pixels in the previous frame;
(iii) calculate Var according to the direction consistency;
(iv) calculate the temporary matrix score, source_pixel_score $(x+i, y+j)$;
(v) find the pixel with the highest score, and deduce the direction.

2.2.4. *The tracking matrix*

During each frame a calculation is made to find the origin of each pixel within the previous frame as described in the

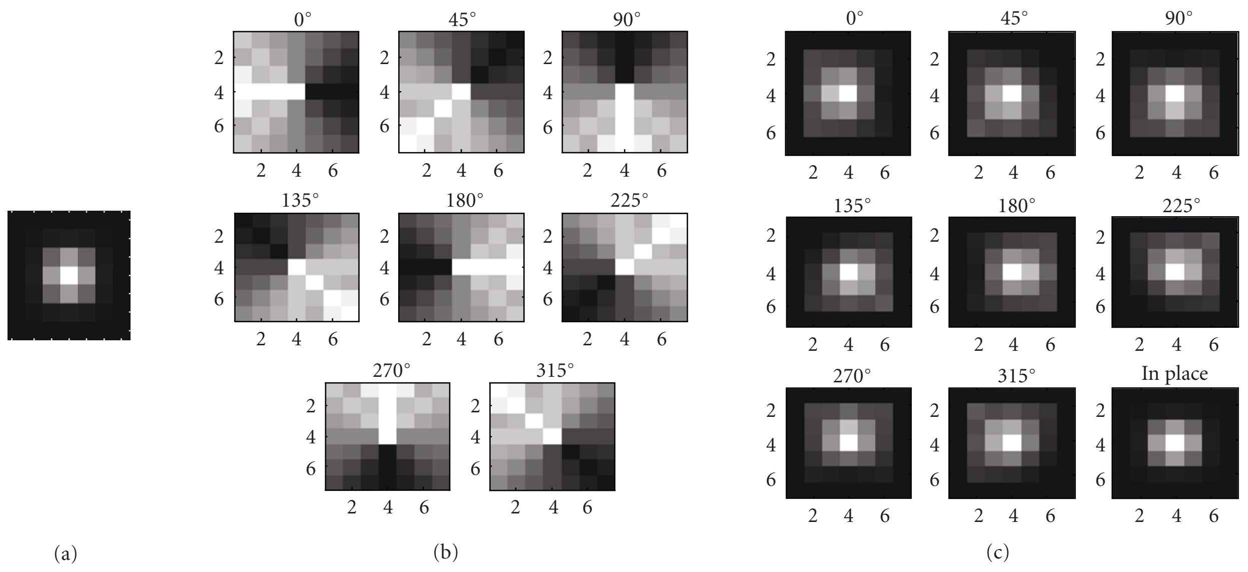


Figure 3: Probability matrices for the various directions. (a) The 7 × 7 W_{basic} matrix, (b) the eight 7 × 7 W_{Penalty} matrices for the different directions, after using W_{frame}, and (c) the nine W(direction) matrices for p set to 0.5, and velocity range [0-1] [ppf].

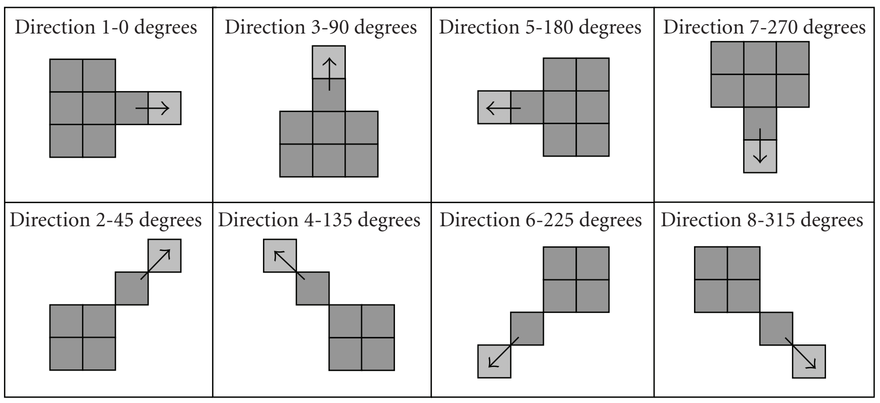


Figure 4: The pixel groups belonging to each of the possible eight directions. The arrow points to the central pixel of the 7 × 7 matrix, and shows the direction of the pixels from the previous frame to the current frame.

previous Sections 2.2.2, 2.2.3. The origin is calculated by taking the element having the maximum value in (7):

$$(\hat{i}, \hat{j})_{x,y} = \max\{\text{Source\_Pixel\_Score}(x+i, y+j)\}. \quad (8)$$

The index of that pixel of origin is saved in the tracking matrix, in the [index, frame] element, where index represents the index of the pixel under investigation in the current frame, and frame is the value of the current frame. Given an image frame or a block the size of N columns and M rows, index $= (N-1)\cdot M + M$.

In that way, the DPA builds the tracks which are most probable of being the target's track. At the last frame processed, the pixel with the highest score is picked, and the index of its pixel of origin from the previous frame is taken. At the previous frame, the found index is used to retrieve the origin of that index. That procedure is iterated down to frame 2. Figure 5 shows a "target" found at index = 578 in frame 15. Tracking it back leads to index = 470 in frame 14 and so on.

2.2.5. *Parameter summary*

In the previous sections, several parameters that control the algorithm functionality were introduced. These parameters are of high significance since they allow flexibility in the algorithm to eventually track and detect targets in an optimal fashion. A summary of the parameters is used in Table 2.

We will concentrate our paper on finding and evaluating the influence of b, and g on the algorithm. We assume a probability matrix weighting of $p = 0.5$ to allow for a certain amount of target maneuvering.

Table 3 deals with the issue of the memory coefficient and discusses the other two parameters b and g. The parameter b will be assigned the name effective memory coefficient (EMC) in Section 2.2.6.

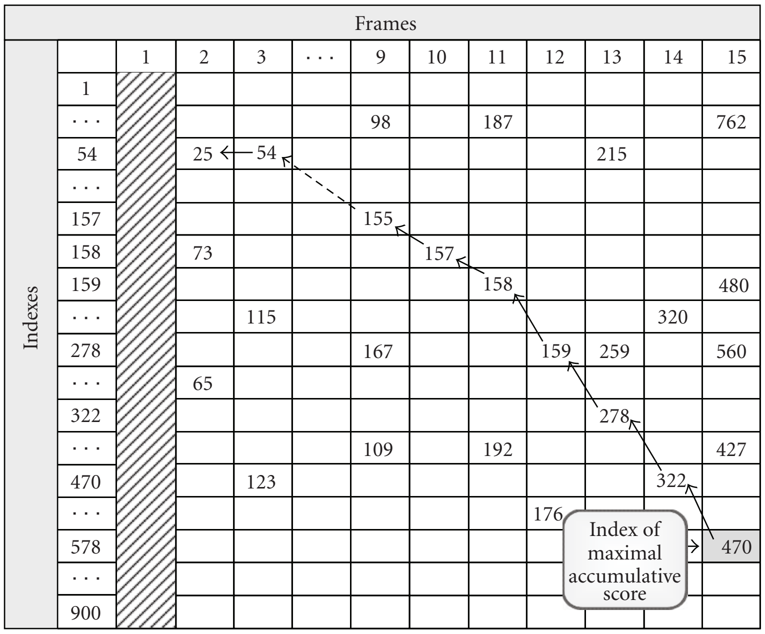


Figure 5: Example of a tracking matrix, and the "target" track.

Table 1: The various real IR sequences used for the testing.

IR sequence name	IR sequence scene description	Target(s) starting location $[y, x]$	Target(s) velocity [ppf]
NPA	Two targets in wispy clouds	[49, 99], [160, 240]	≈0.2
NA23	Single fast target in bright clouds	[136, 128]	≈0.3

Table 2: Summary of EMCs for which tracking occurred in the IR sequences.

IR sequence name	Maximal target peak ([13, page 74])	Relevant range of b's where tracking occurred		IR sequence name		Relevant range of s_n where tracking occurred	
		max part ($g = 0$)	sum part ($g = 1$)	sum part ($g = 1$)	max part ($g = 0$)	sum part ($g = 1$)	max part ($g = 0$)
NPA	~30	2–4.8	0–0.8	3	1	3–5	3–5
NA23	~60	4.4–4.6	0.2–0.8	3	1	1–5	1–5

Table 3

b	—	The memory persistence coefficient, determining the influence of the accumulated score.
p (and $(1-p)$)	—	The $W_{\text{basic}}(W_{\text{Penalty}}(\text{direction}))$ coefficient determining the influence of the 7×7 $W_{\text{basic}}(W_{\text{Penalty}}(\text{direction}))$ matrix on $W(\text{direction})$ probability matrix.
$g(1-g)$	—	The sum coefficient determining the amount of influence of the summation (maximum) on the accumulated score.

2.2.6. *The effective memory coefficient (EMC)*

As can be seen from the (2), b determines the memory persistency or the decaying influence of the scores from the previously processed frames. To prevent the algorithm from "exploding" (i.e., exponentially increasing rather than decreasing), 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 lower the overall score will be (assuming that the influence is mainly on the target compared to the false alarms). 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. The 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, thus providing faster detection capabilities of well maneuvering targets. In previous research [2], where low velocity targets of 0.1-0.2 [ppf] were tested in both straight lines and in maneuvering, a preferred memory

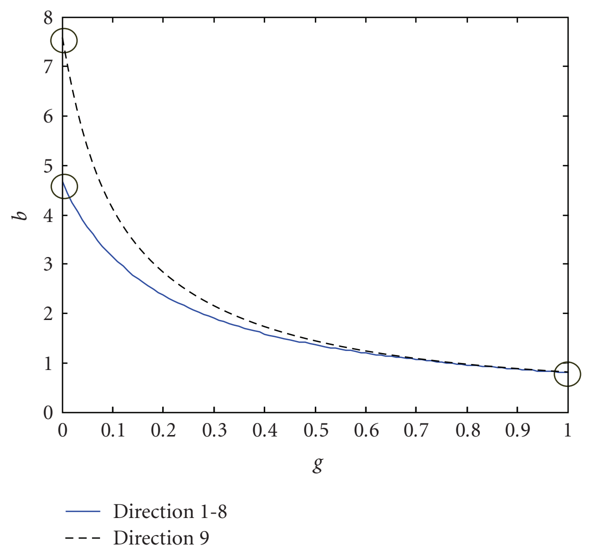


FIGURE 6: b versus g for the direction 1–8 (solid blue line) and direction 9 (dashed line) the black circles represent the $g = 0/1$ cases and the equivalent b values.

coefficient was obtained, $b = 0.7$; on the basis of more recent research, we recommend $b = 0.8$. The formula used for the accumulated score was

$$\mathrm{ASF}_n = \mathrm{WF}_n + b \cdot \max\{w \cdot \mathrm{ASF}_{n-1}\} \tag{9}$$

and the normalization,

$$b = \frac{0.8}{\max\{w\}} \tag{10}$$

was used in order to limit b to $0 \le b \le 1$, and to maximize it for best performance.

The score formula in the current DPA version is

$$\mathrm{ASF}_n = \mathrm{WF}_n + b \cdot [g \cdot \mathrm{sum}\{w \cdot \mathrm{ASF}_{n-1}\} + (1-g) \cdot \max\{w \cdot \mathrm{ASF}_{n-1}\}]. \tag{11}$$

Similarly, the requirement here is

$$b = \frac{0.8}{[g \cdot \mathrm{sum}\{w\} + (1-g) \cdot \max\{w\}]}. \tag{12}$$

By the substitution of $g = 0$, $g = 1$ case into the requirement, the following is obtained:

$$b_{g=0} = \frac{0.8}{\max\{w\}}, \qquad b_{g=1} = \frac{0.8}{\mathrm{sum}\{w\}}. \tag{13}$$

It should be noted that $b_{g=0}$ and $b_{g=1}$ are constant and direction dependent since they are derived from the probability matrices mentioned in Section 2.2.2.

After rearrangement, the following formula for EMC is derived:

$$\begin{aligned} b &= \left[\frac{g \cdot \mathrm{sum}\{w\} + (1-g) \cdot \max\{w\}}{0.8}\right]^{-1} \\ &= [g \cdot b_{g=1}^{-1} + (1-g) \cdot b_{g=0}^{-1}]^{-1} \end{aligned} \tag{14}$$

Using the probability matrices achieved empirically Section 2.2.2, an appropriate range for b can be found so that $0 \le \mathrm{EMC} \le 1$. Putting that into b leads to

$$\begin{aligned} b &= [1.118 \cdot g + 0.131]^{-1} \quad \text{for direction 1 to 8}, \\ b &= [1.035 \cdot g + 0.214]^{-1} \quad \text{for direction 9}. \end{aligned} \tag{15}$$

The derived formula suggests the values of b that should be used for a given value of g. Since $0 \le g \le 1$, a graph of b versus g can be plotted, as in Figure 6, and a valid range of values for b can be found.

As can be seen from the graph, a valid range of values for b is about $0 \le b \le 8$ for $g = 0$ and $0 \le b \le 1$ for $g = 1$. Since $b_{g=1} = 0.8$ 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 subpixel velocities, mainly 0.2-0.3 [ppf], most of the time they are in direction 9 (standing in place); thus the algorithm is expected to be most effective for $b_{g=1} = 0.8$ and $b_{g=0}$ close to 4.65.*

The graph above might also aid if a superposition of the sum and the max parts of the ASF are needed. Adjusting the superposition is done by simply assigning g a value and deriving the appropriate value of b from the graph. Finding the optimal bs (or range of bs) for the different scenes will be done by optimizing an algorithm score, given by a metric that will be defined in Section 2.3.

2.3. Performance metric

In order to evaluate the algorithm, the metric below has been defined. Each frame in the processed batch of frames is divided into blocks of size $M \times N$ (30×30 were used for slow targets, and 30×80 for fast targets). The algorithm is run over nine blocks, that is, the target block and eight adjacent blocks. The SNR of the target block (TB) and its eight adjacent nontarget blocks (NTBs) are calculated. Afterwards, an algorithm score is calculated based on the resulting SNRs (signal to noise ratios).

The block SNR is given as

$$\mathrm{Block\_SNR}(i,j) = \frac{E[v_{i,j} \in M] - E[v_{i,j} \notin M]}{\sigma_{v_{i,j}}}, \tag{16}$$

where $v_{i,j}$ is the set of pixels belonging to the (i, j) block, M is a set containing the five pixels with the highest scores 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 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; it will also 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 (A_S) is given as

$$\text{A\_S} = \frac{\text{Block\_SNR}(i,j)_{\{(i,j)\}\in\text{TB}} - E\left[\{\text{Block\_SNR}(i,j)\}_{\{(i,j)\}\in\text{NTB}}\right]}{\sigma_{\{\text{Block\_SNR}(i,j)\}_{\{(i,j)\}\in\text{NTB}}}}. \tag{17}$$

The A_S is calculated by subtracting between the TB's SNR and the expectation value of the SNR of the NTB's, divided by the standard deviation of the NTB's 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 NTBs than the true detection in the TB, resulting in a positive A_S. It should be noted that the division to blocks is only for the purpose of algorithm evaluation (A_S), whereas in real scenarios, the frames will not be divided. Also, only nine blocks were used due to memory and time complexity.

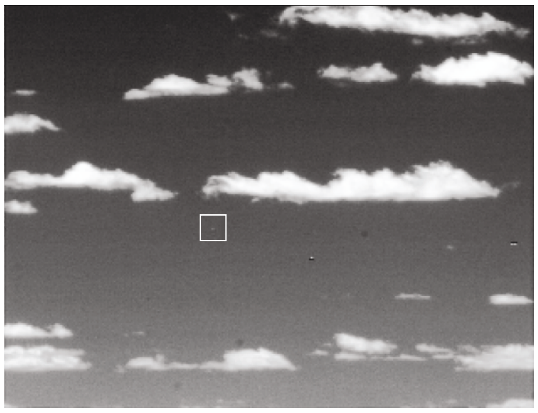
In the tests performed, other numbers of highest pixels, defined as Max_Numbers, were taken to check the influence of the number of high pixels regarded as "*target pixel*" on the *Block_SNR*. Using the metric above for the A_S, the EMC value for optimized algorithm work can be found, in both cases of $g = 0$ (the max part) and $g = 1$ (the sum part) in the accumulated score of (2).

3. RESULTS

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 on a batch of 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 (in white rectangles) is shown in Figure 7. The NA23A sequence has a single target moving in clear sky from right to left at $\nu \approx 0.3$ [ppf]. The NPA sequence has two targets moving at $\nu \approx 0.2$ [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 surroundings of the left target, since it is of more interest due to its more difficult condition. A target range velocity of 0-1 [ppf] has been assumed; hence 5×5 probability matrices were used. Table 1 depicts the different IR sequences tested, the target(s) location, and the target(s) velocity.

The DPA was applied for two different cases: $g = 0$ (the maximum part in the ASF) and $g = 1$ (the summation part in the ASF) over a range of values of b to check the effect of each part of the accumulated score formula on the algorithm results. The introduced metric Section 2.3 was used to find the optimal values of bs, for which tracking has occurred.

Afterwards, the effect of Max_Numbers (the number of pixels with the highest scores considered to be "*targe*" in the *Block_SNR* formula (16)) on the A_S was checked. This action is taken to assure that the metric gives a reliable A_S performance for the different EMCs (b), namely, selecting target affected pixels only as "target" pixels. Another aspect examined was the effect of the target velocity on the


(a)

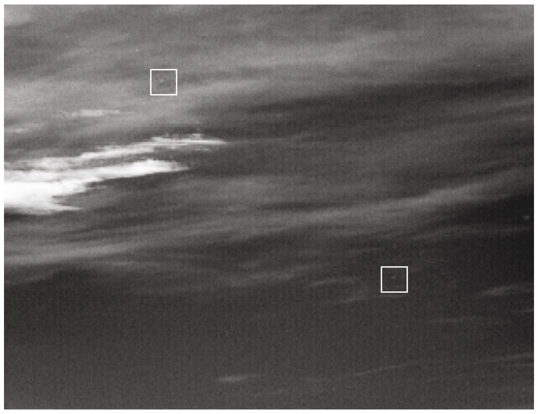

(b)

Figure 7: A single frame from IR sequences. (a) NA23A and (b) NPA sequence showing the target's locations (white rectangles).

algorithm performance by sampling the given IR sequences every s_n frames, where s_n is a value in the range of 1 to 5.

Section 3 is ordered as follows: (a) a preprocessing stage example is given for the NA23A sequence, (b) the NA23A and NPA sequence results are given, and (c) a brief summary of findings.

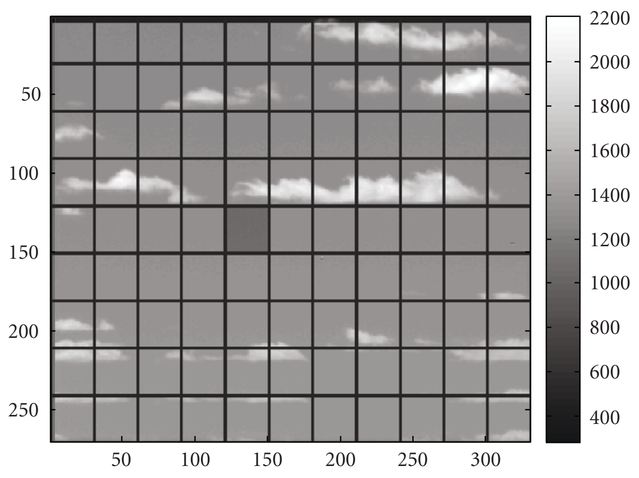
3.1. *The preprocessing whitening stage*

The algorithm starts by whitening the input IR sequence. According to the introduced A_S metric, the sequence is divided into blocks. The TB and the eight surrounding NTBs are then whitened to reduce the clutter and emphasize the target, as can be seen in Figure 8 of the 1st frame of the NA23A sequence.

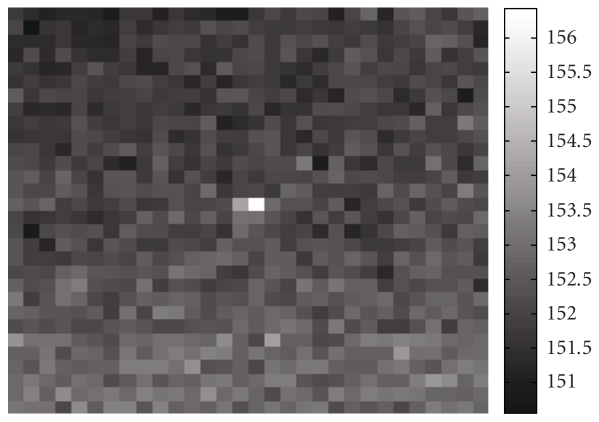
The process shown in Figure 8 is repeated for the required sequence frames before proceeding to the next stage of the DPA. (The frames can also be processed individually for a real-time implementation).

3.2. *The DPA stage*

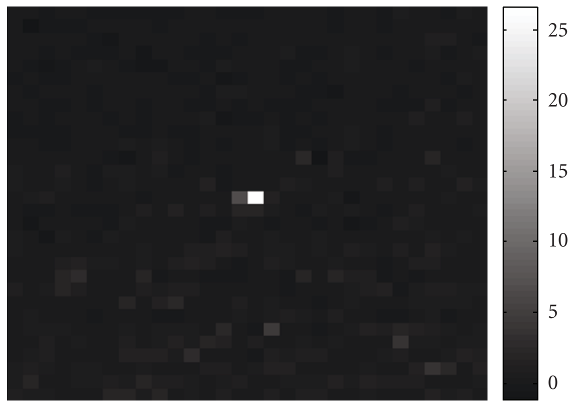
The key advantage of the algorithm is the penalty matrices; it gives an advantage to targets moving in a consistent manner



(a)

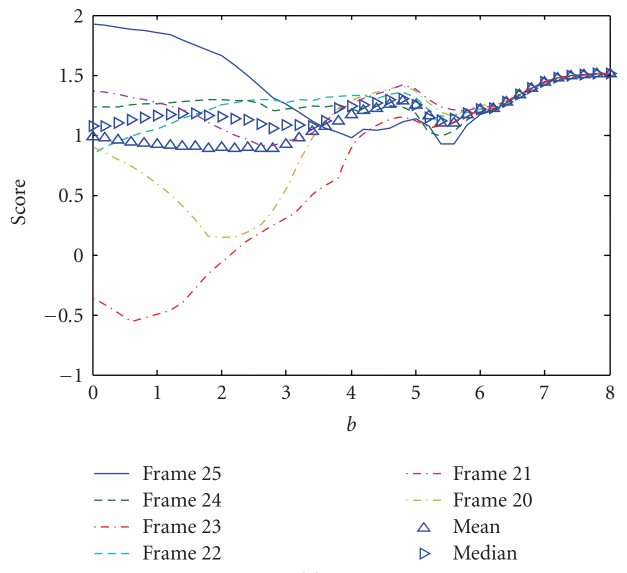


(b)

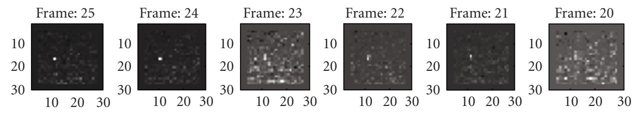


(c)

Figure 8: Algorithm stages. (a) 1st frame of the original NA23A sequence divided into blocks, (b) 1st frame of TB (target at center of frame), and (c) 1st frame of TB after the preprocessing whitening stage.



(a)



(b)

Figure 9: NA23A sequence preliminary results; $g = 0$, Max_Numbers $= 5$. (a) A_S for $b = [0 \cdots 8]$, and six last frames; (b) the last six processed frames.

from pixel-to-pixel. If the target is moving at a subpixel 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 worked best, the algorithm has been run over the sequence, in both cases of g, for s_n from 1 to 5, for example, given $s\_n = 5$, take only every 5th frame.

It should be noted that for the algorithm to accumulate effectively, approximately 20 frames at minimum have to be processed, thus limiting the amount of s_n that can be used depending on the number of available sequence frames.

To keep the target in the “target block,” a larger block size of 30×80 was used for fast targets instead of 30×30.

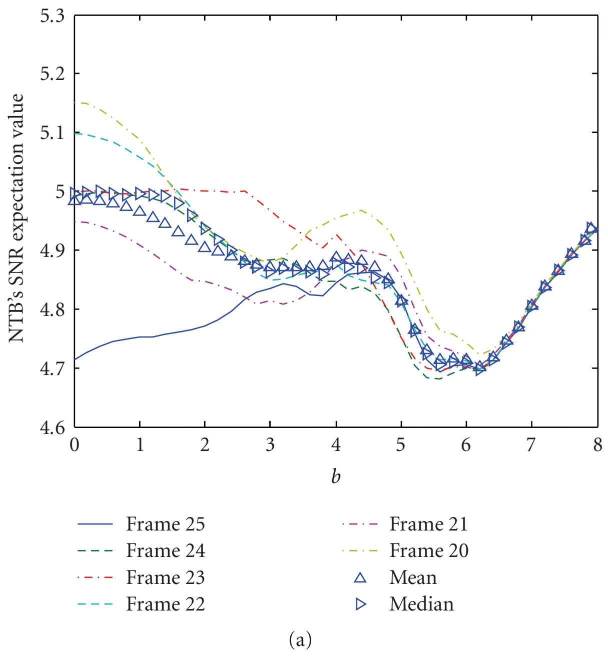
3.2.1. NA23A sequence

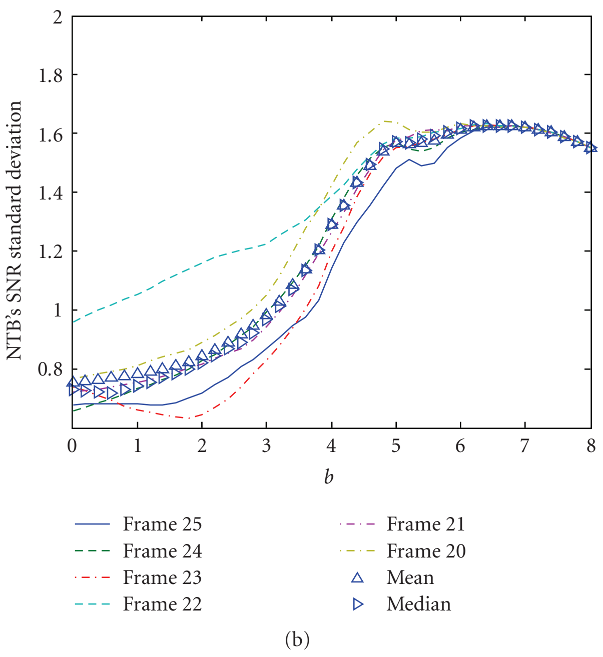
(1) Preliminary results $(s\_n = 1)$

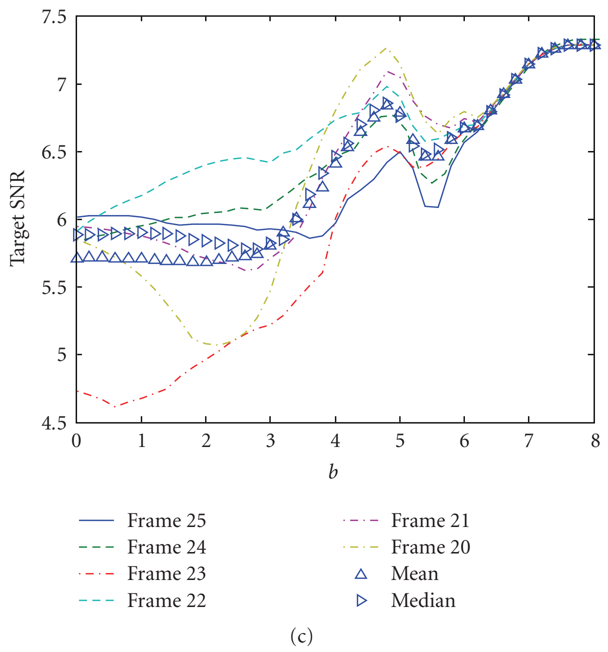
The results are first shown for the NA23A sequence, $g = 0$ case. Figure 9 shows the A_S for the range of $0 \le b \le 8$ at intervals of 0.2. The graph shows the A_Ss for the last six frames, 20–25 in this case, for two reasons: (1) some frames might be noisier than the others; (2) we wish to show the accumulated score effect.

The first cause will result in the target being dimmer compared to the clutter; the target may even disappear. In that case, the memory persistence and the accumulation of the score from frame to frame help maintain the track. The accumulated score helps overcome noisy frames, and “accumulates” the SNR for the dim target. Looking at the last frames helps us understand whether the effect of accumulation is sufficient for the number of frames processed, or whether more frames need to be processed.

In previous research [4, 6] the algorithm was run over 10 to 20 frames; 20 was found to be significantly better; hence 25 frames were taken in this case, to be confident that we have given the algorithm a fair test.







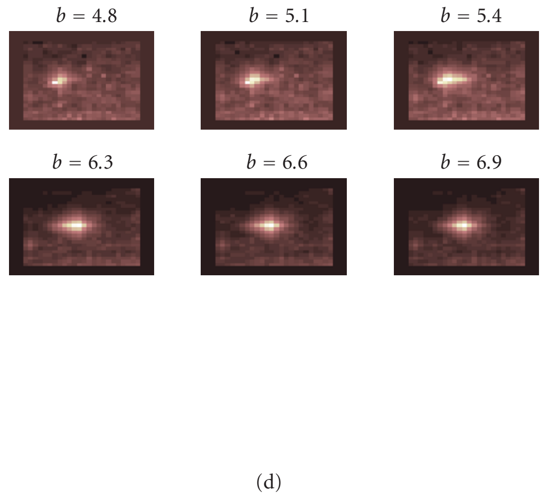


FIGURE 10: NA23A sequence preliminary results; $g = 0$, Max_Numbers $= 5$. (a) Standard deviation of NTB's SNR, (b) expectation value of NTB's SNR, (c) TB's SNR, and (d) last-processed frame (25th) for various values of b.

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 A_S converge, suggesting that the EMC is too high and that the weight of the current frame is insignificant compared to the accumulated score. It seems that the relevant range for EMC can be narrowed to $0 \le b \le 6$, for the $b = 0$ case, in the current IR sequence.

A rise A_S is expected as the frames advance. It can be seen in the graph in Figure 9(a), that the A_S 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 9(b). Increasing the EMC improves the A_S of these frames, that is, more memory persistence and less weight to current noisy frame. Since the target moves at $\nu \approx 0.3$ [ppf] (stays in the same pixel most of the time) the expected theoretical peak (EMC calculation in Section 2.2.6) was at around $b = 4.65$ in agreement with the results shown in Figure 9(a).

Figure 10 shows the graphs of the standard deviation of the NTB, the expectation value of the NTB's SNR, the TB's SNR (all versus b), and the last processed 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

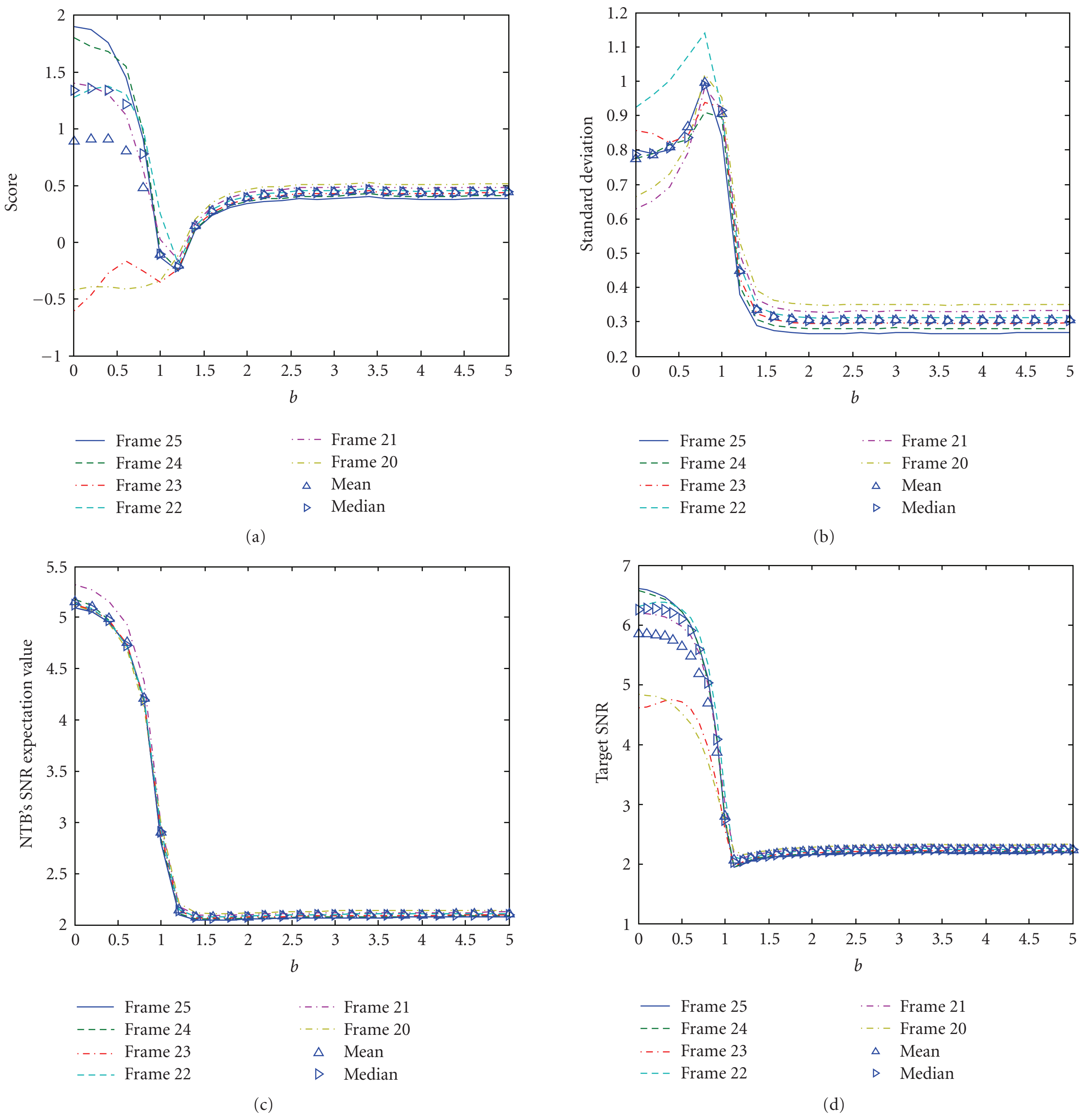


FIGURE 11: NA23A sequence preliminary results; $g = 1$, Max_Numbers $= 5$. (a) A_S, (b) standard deviation of NTB's SNR, (c) expectation value of NTB's SNR, and (d) TB's SNR.

retained its high value (Figure 10(d)). Thus values of EMC higher than that will prevent the algorithm from tracking the target, since the trail pixels and the pixel of origin will gain a higher accumulated score, as will be seen later in Section 3.2.1(3).

Results for the $g = 1$ case are shown in Figure 11. As can be seen from the A_S, the relevant range of b is $0 \leq b \leq 1$, after which A_S's of the various frames converge regardless of the EMC. The TB's SNR can be seen to fall at around $b = 1$. The A_S starts at its maximum (mean value) for no memory and drops as the EMC rises, until $b = 1$. This suggests that in this case and for this particular sequence, the algorithm is not needed for the detection.

Figure 12 shows the influence of the EMC on the pixels of the last frame. In the $g = 1$ case, if we raise the EMC to too high a value (above $b = 1$), no trail appears as in the $g = 0$, but a "*snow bal*" 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 increase in their value. If the EMC is high, these adjacent pixels will also receive high accumulated scores, and in the next frame, pixels adjacent to

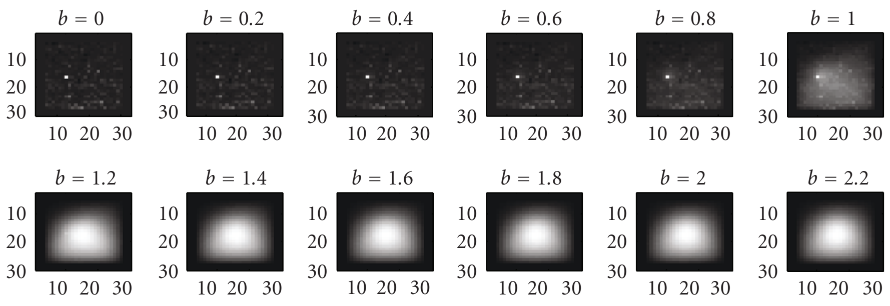


Figure 12: Last processed frame (25th) for $b = [0 \cdots 1]$.

them will also increase in their value, causing the ball to grow from frame to frame.

To conclude, a valid range of EMC, $0 \leq b \leq 8$, has been found in which the algorithm is able to track and detect the target correctly; this concurs with the theoretical range. Section 3.2.1(2) will deal with the issue of target velocity, and will demonstrate that a preferred EMC exists, $0.6 \leq b \leq 0.8$, close to the theoretical value.

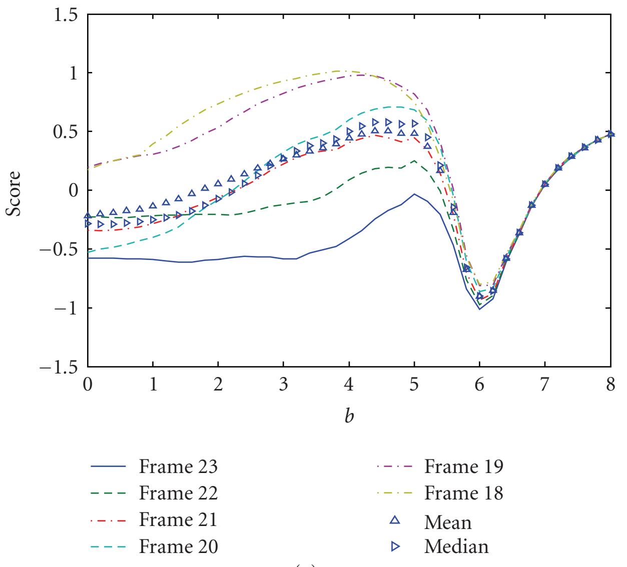
(2) The s_n variable

As stated in the beginning of Section 3.2, since the main advantage of the algorithm is the usage of the penalty matrices, the target has to move at a velocity of at least around 1 [ppf]. If the target in the sequence moves at lower velocities, $v \approx 0.3$ [ppf] in our case, the penalty 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 $s\_n = 4$ (only every 4th frame was taken) giving the target a velocity of $v \approx 1.2$ [ppf]. To keep the target in the "*target bloc*," a larger block size of 30×80 was used instead of 30×30.

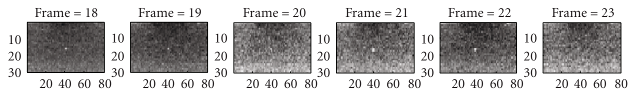
The A_S and the last six frames are shown in Figure 13, for the $g = 0$ case. The A_S achieved is lower in this case due to the noisy frames compared to the last six frames in the $s\_n = 1$ case. It should be noted that by sampling the sequence of the algorithm, different frames were processed that differ in their noise degree. Nevertheless, the A_S 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.

Figure 14 shows the graphs of the standard deviation of the NTB, the expectation value of the NTB SNR, the TB's SNR (all versus b), and the target track found for $s\_n = 4$ (the pink portion of the track is the target track for $s\_n = 1$).

Comparing Figure 10 to Figure 14 shows that the TB and NTB SNRs behave similarly for $s\_n = 1$ and for $s\_n = 4$; only the TB SNR has a distinguishable peak. This shows that the penalty matrices help distinguish between target and clutter, and that the algorithm needs targets at around $v \approx 1$ [ppf] to work effectively. In the case of $g = 1$, the A_S is lower compared to the one achieved for $s\_n = 1$, due to the noisy



(a)



(b)

Figure 13: NA23A sequence, $s\_n = 4$. (a) The A_S for $g = 0$, Max_Numbers $= 5$, $b = [0 \cdots 8]$, and the six last frames, (b) images of the six last frames after the DPA algorithm.

last frames, as in the $g = 0$ case. Section 3.2.1(3) deals with the issue of using values of the Max_Numbers parameter that will correctly take only target pixels as the highest pixels.

(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 A_S. 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 A_S, for the two cases of $g = 0$ and $g = 1$. A value of $b = 4.8$ and $s\_n = 4$ was taken. The optimization was first run for the $g = 0$ case. Figure 15 shows a peak of mean A_S 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

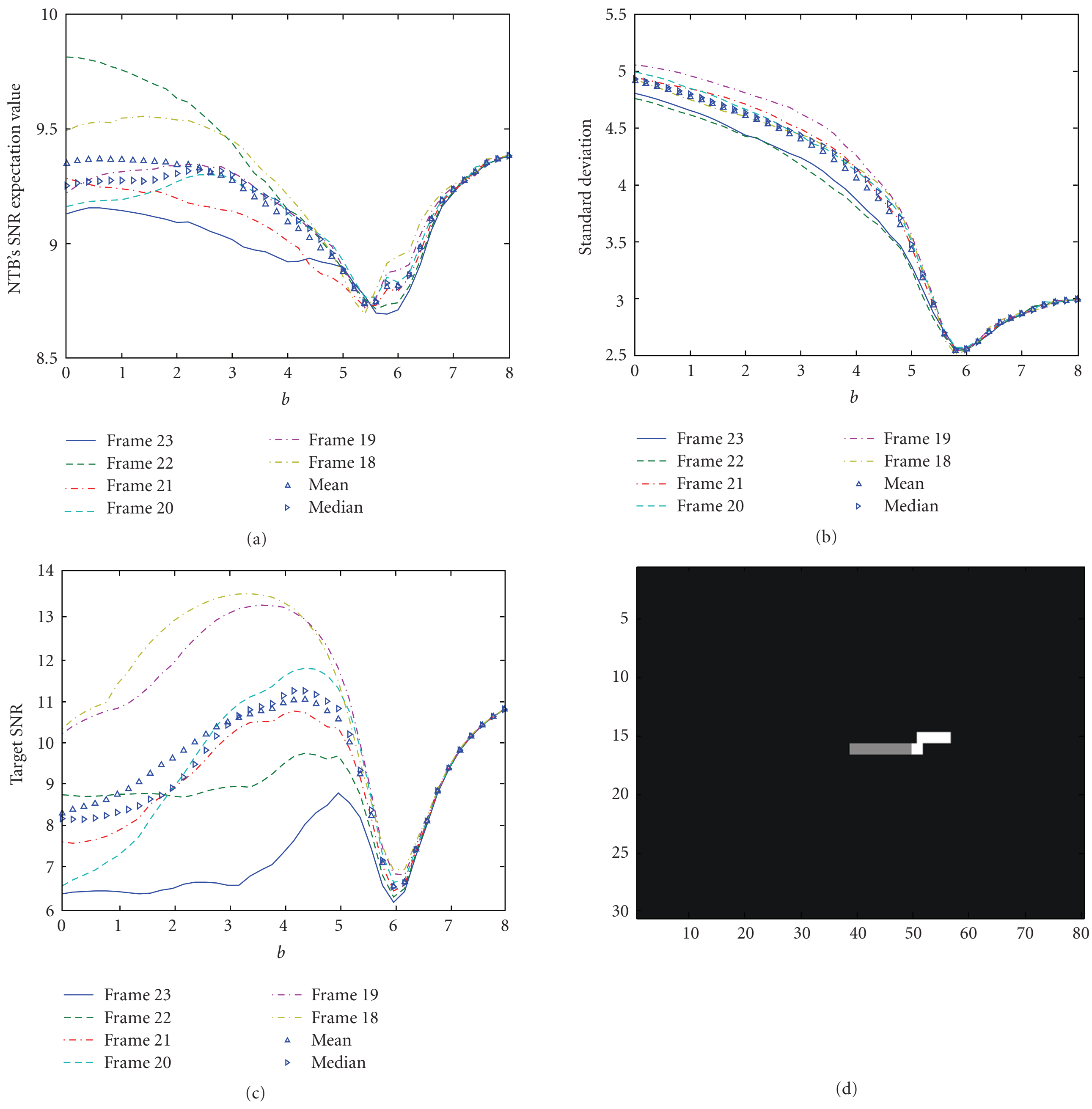
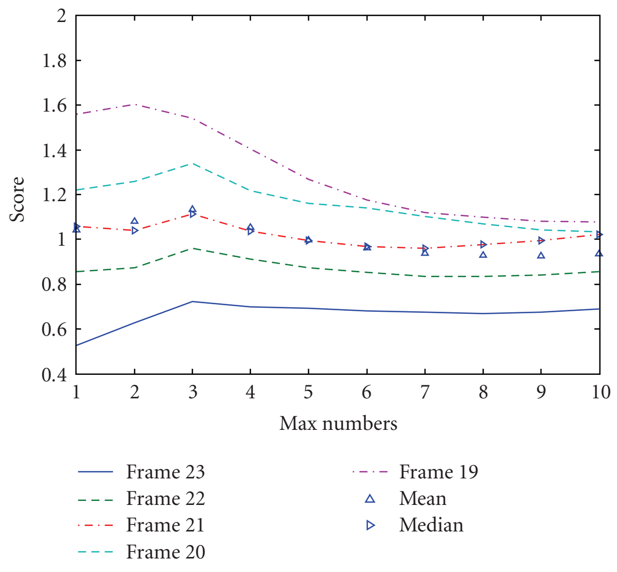


FIGURE 14: NA23A sequence, $s\_n = 4$. (a) Standard deviation of NTB's SNR, (b) expectation value of NTB's SNR, (c) TB's SNR, and (d) the target's track (the pink track is the target track for $s\_n = 1$).

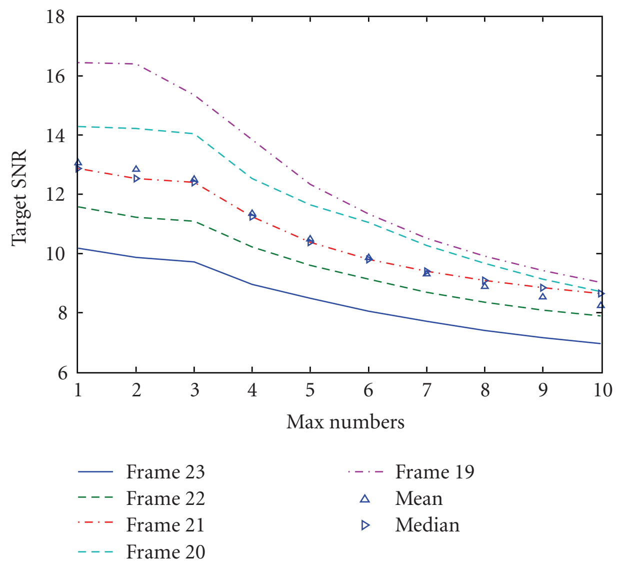
the figure for Max_Numbers > 3. The result suggests that Max_Numbers = 3 is preferable for the $g = 0$ case.

Figure 16 shows the A_S versus b of the NA23A sequence, for $s\_n = 4$. The graph is separated to three sections according to the resulting highest pixels in the last frame: (1) target pixels, (2) trail and target pixels and pixel of origin, and (3) pixel of origin and trail pixels. In the case of the 5 highest pixels, the A_S takes only the target pixels up to $b = 5.6$ (as can be seen in Figure 17(a)). 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 $b = 5.8$ (as can be seen in Figure 17(b)). Higher EMCs cause nontarget pixels to be higher than the target pixels as before. In this case of $g = 0$ and $s\_n = 4$, the algorithm was able to track and locate the target up to $b = 6.0$. Using that and Figure 16 leads to the conclusion that a good range of EMCs would be $4.4 \leq b \leq 6$. This range contains the theoretical EMC for nonmoving targets.

It should be noted that the resulting separation concurs with the relevant range of $0 \leq b \leq 6$ deduced from the preliminary results. The graph in Figure 15 helps to distinguish a good range in which the highest pixels taken as "target pixels" indeed belong to the target.



(a)



(b)

Figure 15: NA23A sequence, $s\_n = 4$. (a) A_S versus Max_Numbers, (b) TB SNR versus Max_Numbers.

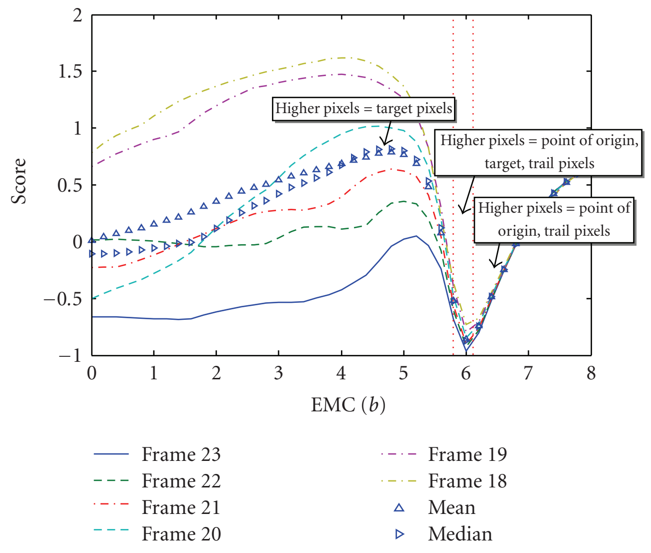
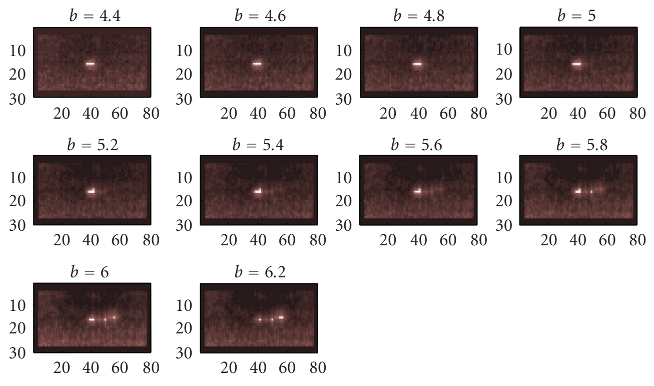


Figure 16: NA23A sequence, $s\_n = 4$, Max_Numbers $= 3$, A_S versus b, $g = 0$ case.

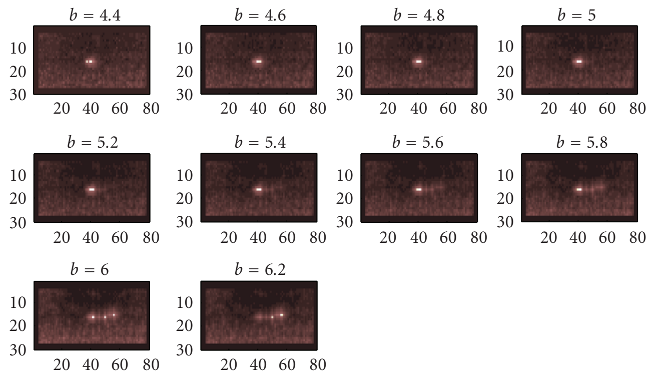
Figure 18 shows the detected target's track. It can be clearly seen that up to $b = 5.6$ the tracking is reliable. An increase in the EMC causes inaccurate target tracking, where further increase ($b > 6$) causes previously target traversed pixels to achieve the highest accumulated score, and hence no tracking occurs.

The discussion continues now to the $g = 1$ case. The last frames of the algorithm versus b are shown in Figure 19. It can be clearly seen that for $0 \leq b \leq 0.8$ there is only one high pixel belonging to the target (surrounded by white circle). The algorithm tracks the target in the range of $0.2 \leq b \leq 0.8$. These results suggest that Max_Numbers $= 1$ and $0.2 \leq b \leq 0.8$ should be used in the case of $g = 1$.

The A_S using Max_Numbers $= 1$ is shown in Figure 20. A peak in the mean A_S can be clearly seen for $b = 0.6$



(a)



(b)

Figure 17: NA23A sequence, $s\_n = 4$, last processed frame. (a) Highest 5 pixels emphasized and (b) highest 3 pixels emphasized.

and the best tracking is achieved for $b = 0.8$ (not shown) in accordance with the theoretical value. The A_S is higher in this case than the $g = 0$ case for $s\_n = 4$ (*s_n number*), and higher than both cases for $s\_n = 1$. This result suggests that the algorithm performs better for faster targets, given

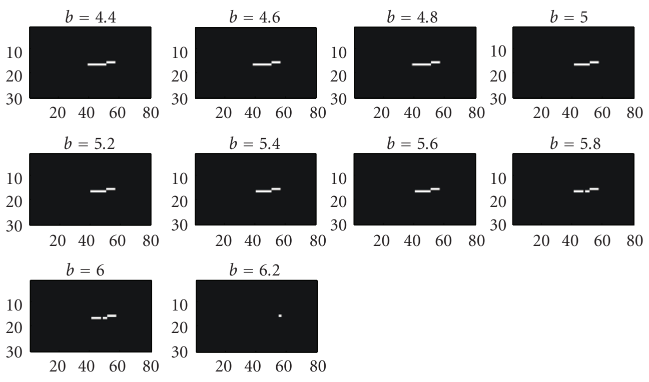


FIGURE 18: NA23A sequence, $s\_n = 4$, target's track versus b.

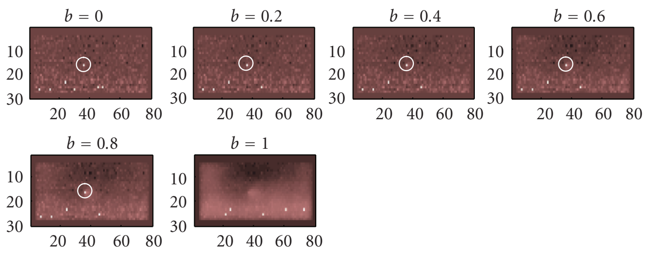


FIGURE 19: NA23A sequence, $s\_n = 4$, last processed frame, $g = 1$ case, highest 5 pixels emphasized (target pixel surrounded by a white circle) versus b.

correct value of Max_Numbers according to the case under investigation.

(4) Summary

Results of the NA23A sequence have been shown, with an emphasis on the effect of the Max_Numbers and the s_n parameters. Preferable values for Max_Numbers were found: Max_Numbers = 1 for the $g = 0$ case, and Max_Numbers = 1 for $g = 1$ case. Relevant ranges of bs were also found for each case: $4.4 \leq b \leq 6$ for the $g = 0$ case $0.2 \leq b \leq 0.8$ for the $g = 1$ case. The $g = 1$ case has performed better for the faster target scenario ($s\_n = 4$). Once again, for each s_n rate, the highest A_S in the relevant ranges of bs 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 s_ns, so a comparison might not be precise. The results are shown in Figure 21. The algorithm has a preferable target velocity at around $\nu \approx 0.6 \pm 0.15$ [ppf].

3.2.2. NPA Sequence

The sequence contains a target moving at $\nu \approx 0.2$ [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 A_S from the whitening preprocessing stage (source for clutter leakage). In this sequence, tracking and detection were achieved for $s\_n \geq 3$. Hence detailed results for lower s_n values will be skipped and only detailed

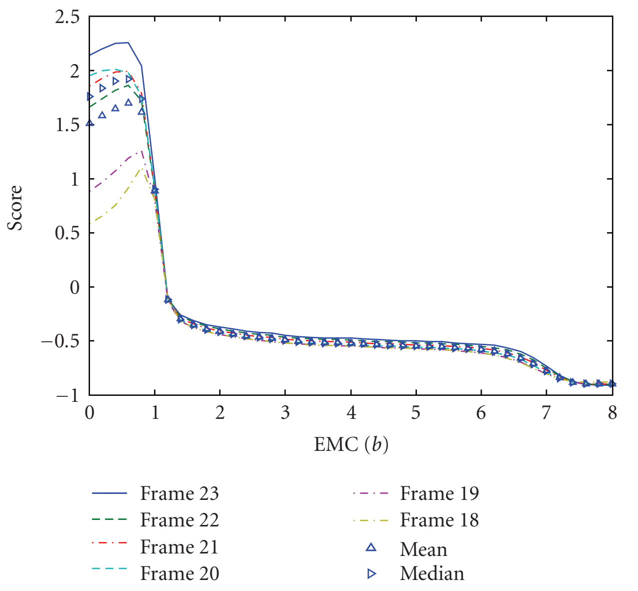


FIGURE 20: NA23A sequence, $s\_n = 4$ A_S versus b, $g = 1$ case and Max_Numbers = 1.

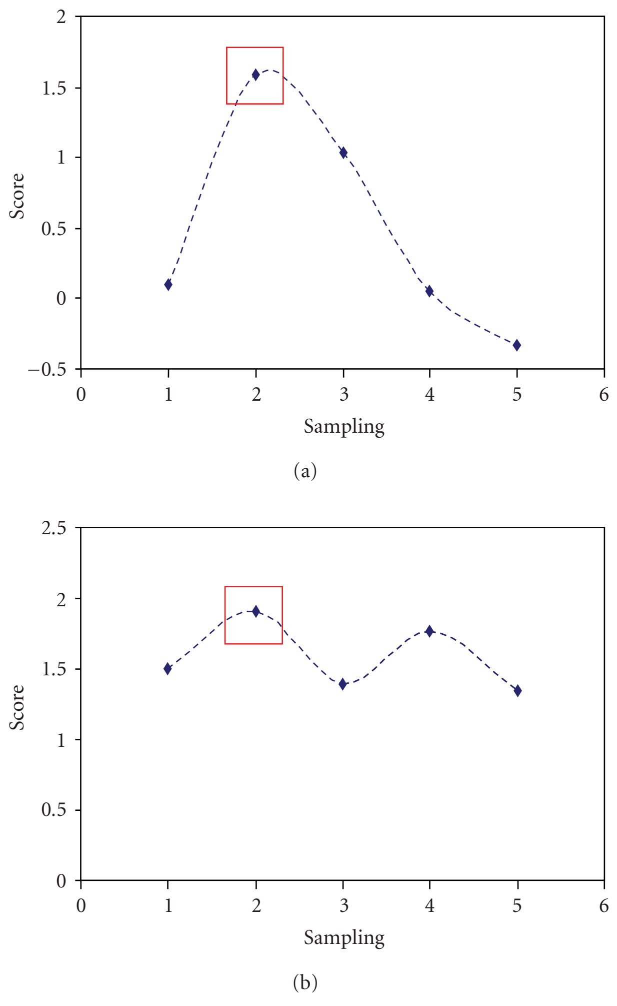
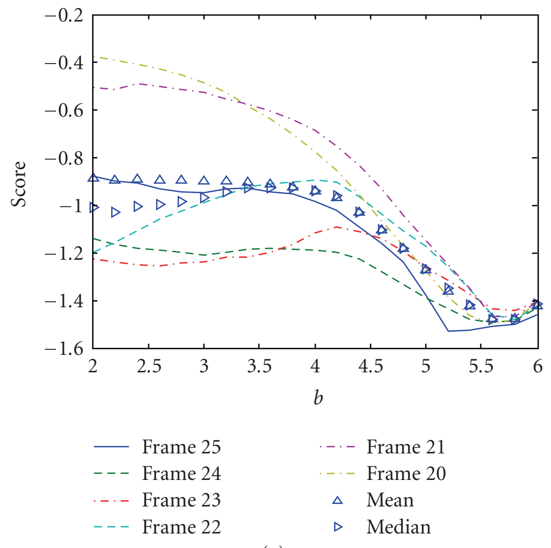
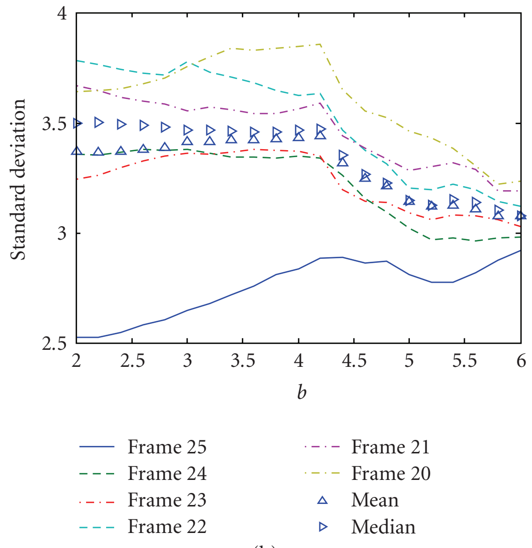


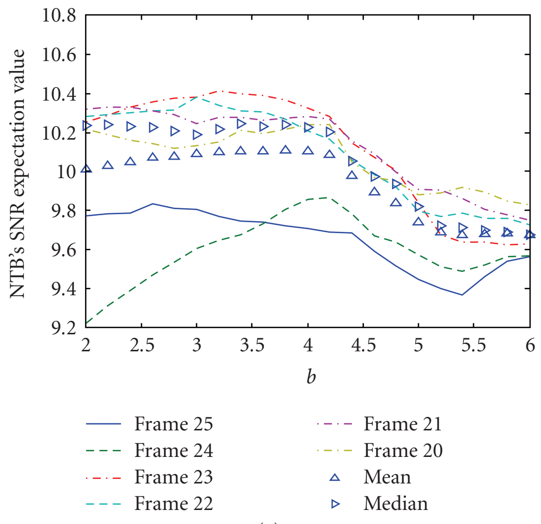
FIGURE 21: NA23A sequence, A_S versus s_n. (a) $g = 0$ case, Max_Numbers = 3 and (b) $g = 1$ case, Max_Numbers = 1.



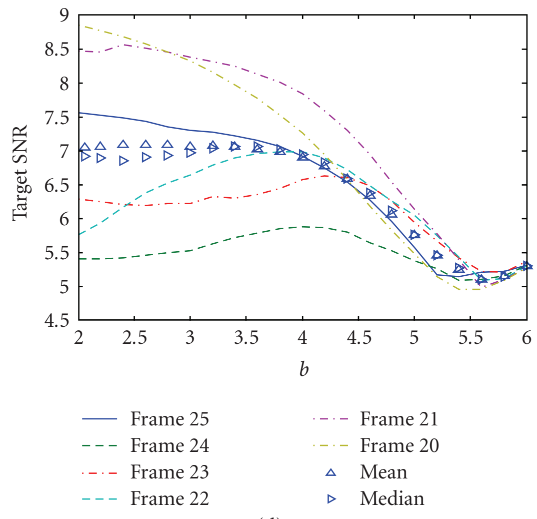
(a)



(b)

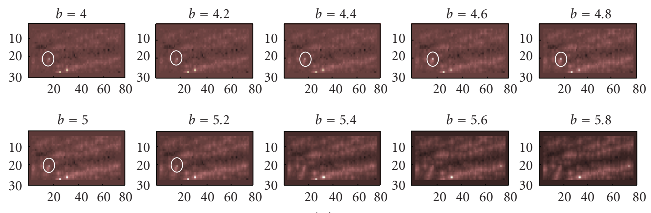


(c)

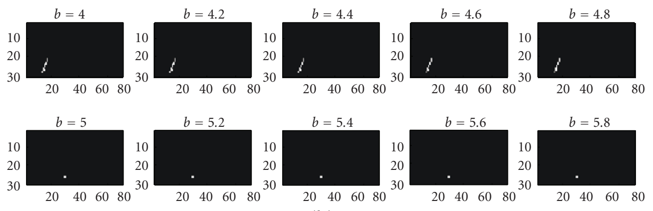


(d)

FIGURE 22: NPA sequence, $s\_n = 3$. (a) A_S, $g = 0$, Max_Numbers = 3, (b) STD of NTB's SNR, (c) expectation value of NTB's SNR, and (d) TB's SNR.



(a)



(b)

FIGURE 23: NPA sequence, $s\_n = 3$, $g = 1$ case. (a) The last frame of the algorithm, highest 5 pixels emphasized versus b (target pixel surrounded by a white circle), (b) the target track versus b.

results for $s\_n = 3$ will be shown. Section 3.2.2(2) will show A_Ss for the various s_ns.

(1) Results ($s\_n = 3$)

Figure 22 shows the following graphs for the $g = 0$ case: A_S, standard deviation of the NTB, expectation value of the NTB SNR, the TB's SNR (all versus b), and the last frame (25) for various values of b.

The resulting A_Ss are negative for all the bs in the range. This is due to the cloud edges at the bottom of the target block that get high accumulated scores (Figure 23(a)). This causes the TB's SNR to be low. Cloud edges in the NTB give rise to their SNR, leading to a negative A_S. Nevertheless,

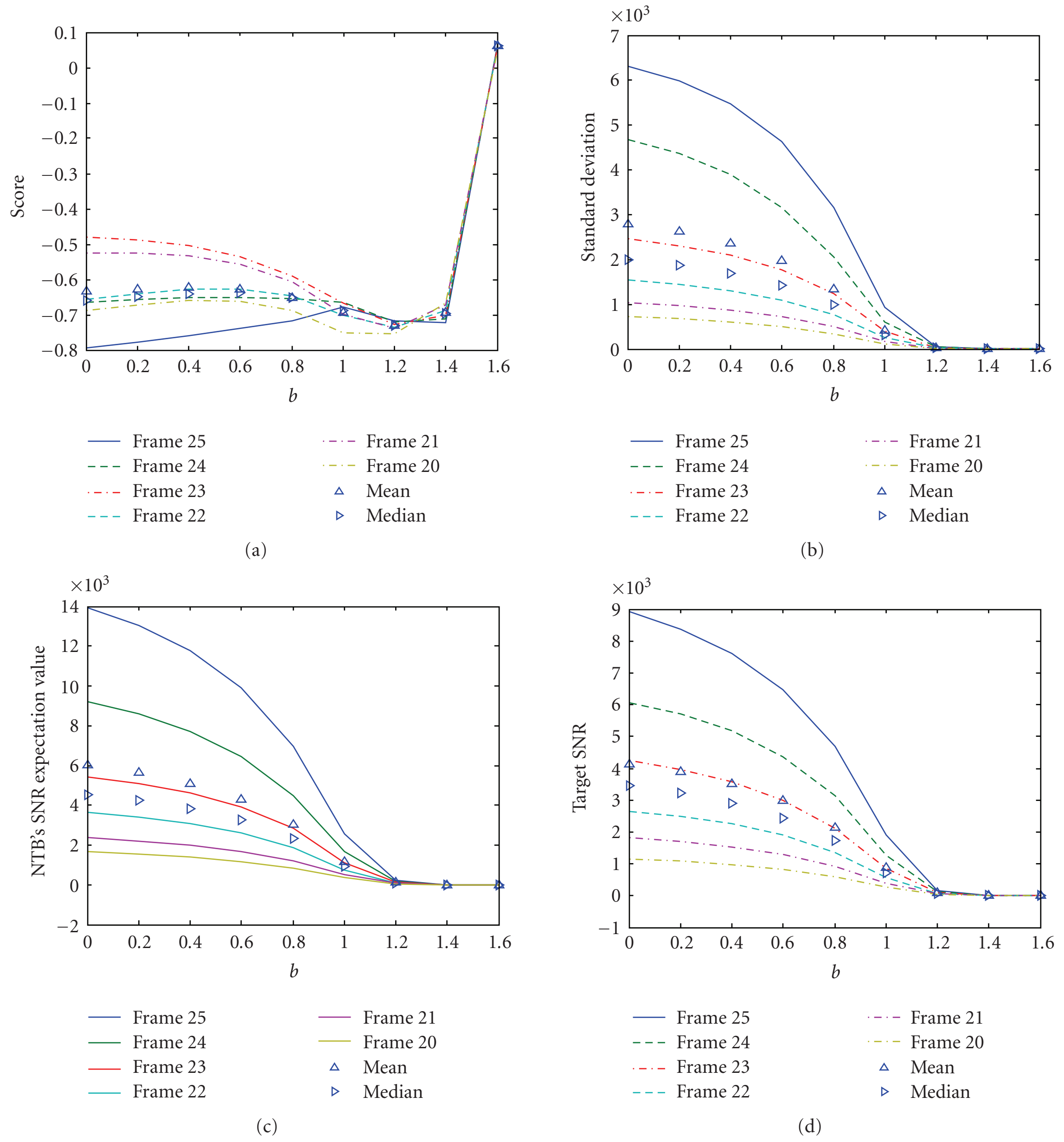


FIGURE 24: NPA sequence, $s\_n = 3$. (a) A_S, $g = 1$, Max_Numbers = 1, (b) standard deviation of NTB's SNR, (c) expectation value of NTB's SNR, and (d) TB's SNR.

the algorithm is able to track and detect the target for $2 \le b \le 4.8$. Higher EMCs lead to cloud's edge having higher score than the target itself in the TB, and hence no tracking.

The results for the $g = 1$ are shown in Figure 24. The results have also been negative as the $g = 0$ case. Nevertheless, the algorithm has been able to track and detect the target for $0 \le b \le 0.8$, as can be seen in Figure 25.

(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 bs were also found for each case: $2 \le b \le 4.8$ for the $g = 0$ case, $0 \le b \le 0.8$ for the $g = 1$ case. The algorithm performed best for $s\_n = 4$, as shown in Figure 26, where the effective target velocity was at around $v \approx 0.8 \pm 0.1$ [ppf].

3.2.3. *DPA results summary*

Relevant ranges of bs for which the algorithm was able to track and detect the target, and the optimal Max_Numbers values, are shown in Table 1 for the various IR sequences. The table also indicates the range of s_n in which the algorithm was able to track and detect the target, hence describing the range of target velocities where the algorithm proved effective.

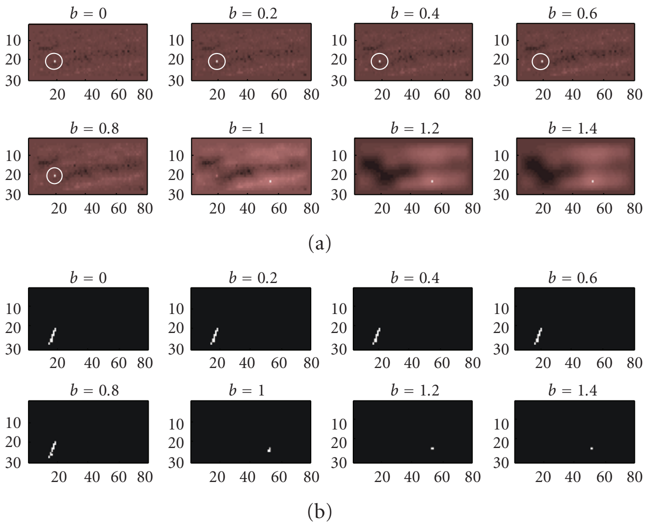


FIGURE 25: NPA sequence, $s\_n = 3$, $g = 1$ case. (a) The last frame of the algorithm, highest 5 pixels emphasized versus b (target pixel surrounded by a white circle), (b) the target track versus b.

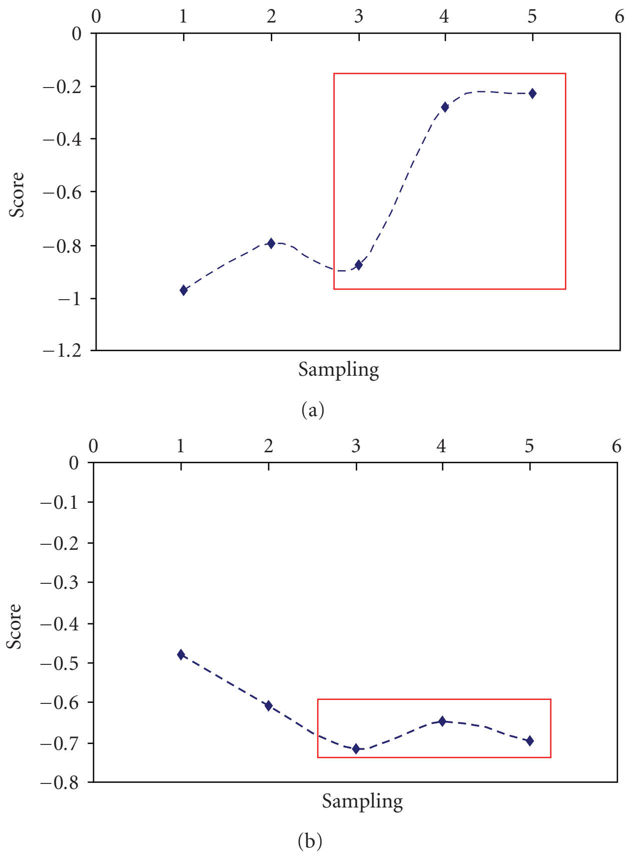


FIGURE 26: NPA sequence, A_S versus s_n. (a) $g = 0$ case, Max_Numbers = 3, (b) $g = 1$ case, Max_Numbers = 1.

Among the goals was finding optimal values of EMC for the various scenes, and various speeds. A table can be built from the data gathered from the results, so that a future tracking system might adapt to the scenery. Evaluation of the scenery might be done using the preprocessing whitening stage which tries to approximate the statistics of the background.

4. SUMMARY AND CONCLUSIONS

This paper has presented a system for the tracking of a dim point target in an IR sequence using the DPA approach. The developed DPA formulation and parameters have been discussed and a metric was introduced as means of optimizing the parameters for different scenes. The results have shown that in hard conditions where clouds are sparse, the cloud edges receive high scores by the preprocessing stage causing false alarms at the DPA stage. In this case, it was shown that the lower EMC range has to be used, so as to "get rid" of this old information of cloud edges. This usage of lower EMC will limit the lowest SNR of the target that the algorithm can track. Generally, 20 frames or more were preferable for the noticeable accumulation in the tracking range of EMC.

In the case of IR sequences containing "fast targets," the algorithm has been able to track target which is at a velocity of at least $\nu \approx 0.3$ [ppf] in low clutter scenes, and velocity of at least $\nu \approx 0.6$ [ppf] in high clutter scenes. Since that is the case, the penalty matrices-based DPA implemented here has to be used with another prestage algorithm that acts as a sampler, and outputs a faster target so that tracking of subpixel velocity point targets is possible. Using the sampling of the sequences, the algorithm was found to perform at its peak for target velocity of $0.6 \leq b \leq 0.8$.

Dynamic programming algorithms for track-before-detect are neither easy nor automatic. A careful study of the relevant parameters shows that the nature of the images, including the target speed and the presence of clutter, affects the optimal setting of the parameters. This paper attempts to contribute to these analyses.

REFERENCES

[1] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[2] R. Succary, A. Cohen, P. Yaractzi, and S. R. Rotman, "Dynamic programming algorithm for point target detection: practical parameters for DPA," in *Signal and Data Processing of Small Targets*, vol. 4473 of *Proceedings of SPIE*, pp. 96–100, San Diego, Calif, USA, July 2001.

[3] H. Madar, T. Avishai, R. Succary, and S. R. Rotman, "Developing a CFAR filter for detecting point targets using a dynamic programming algorithm," in *Signal and Data Processing of Small Targets*, vol. 5204 of *Proceedings of SPIE*, pp. 31–34, San Diego, Calif, USA, August 2003.

[4] R. Succary, H. Kalmanovitch, Y. Shurnik, Y. Cohen, E. Cohen, and S. R. Rotman, "Point target detection," in *Infrared Technology and Applications XXVIII*, vol. 4820 of *Proceedings of SPIE*, pp. 671–675, Seattle, Wash, USA, July 2003.

[5] M. Sniedovich, *Dynamic Programming*, Marcel Dekker, New York, NY, USA, 1992.

[6] M. Bar-Tal and S. R. Rotman, "Performance measurement in point source target detection," *Infrared Physics & Technology*, vol. 37, no. 2, pp. 231–238, 1996.

[7] S. K. C. David, D. A. Langan, and D. A. Staver, “Spatial processing techniques for the detection of small targets in IR clutter,” in *Signal and Data Processing of Small Targets*, vol. 1305 of *Proceedings of SPIE*, Orlando, Fla, USA, January 1990.

[8] S. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, London, UK, 1999.

[9] Y. Barniv, “Dynamic programming solution for detecting dim moving targets,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 21, no. 1, pp. 144–156, 1985.

[10] Y. Barniv and O. Kella, “Dynamic programming solution for detecting dim moving targets part II: analysis,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 23, no. 6, pp. 776–788, 1987.

[11] J. Arnold, S.W. Shaw, and H. Pasternack, “Efficient target tracking using dynamic programming,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 1, pp. 44–56, 1993.

[12] V. Samson, F. Champagnat, and J.-F. Giovannelli, “Point target detection and subpixel position estimation in optical imagery,” *Applied Optics*, vol. 43, no. 2, pp. 257–263, 2004.

[13] V. R. Louisa, “Point target tracking in hyperspectral images,” M.S. thesis, Ben-Gurion University, Beer-Sheva, Israel, June 2005.