DIM TARGET DETECTION IN INFRARED IMAGERY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BARIŞ ÇİFÇİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Prof. Dr. Mete SEVERCAN Co-Supervisor	Assoc. Prof. Dr. A. Aydın ALATAN Supervisor

Examining Committee Members

Prof. Dr. Mübeccel DEMİREKLER	(METU, EE)	_____
Assoc. Prof. Dr. A. Aydın ALATAN	(METU, EE)	_____
Prof. Dr. Mete SEVERCAN	(METU, EE)	_____
Prof. Dr. Kemal LEBLEBİCİOĞLU	(METU, EE)	_____
Sevda ERDOĞDU (MSc.)	(ASELSAN)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Barış ÇİFÇİ

Signature :

ABSTRACT

DIM TARGET DETECTION IN INFRARED IMAGERY

Çifçi, Barış
M.Sc., Department of Electrical Electronics Engineering
Supervisor : Assoc. Prof. Dr. A. Aydın ALATAN
Co-Supervisor : Prof. Dr. Mete SEVERCAN

September 2006, 115 pages

This thesis examines the performance of some dim target detection algorithms in low-SNR imaging scenarios. In the past research, there have been numerous attempts for detection and tracking barely visible targets for military surveillance applications with infrared sensors. In this work, two of these algorithms are analyzed via extensive simulations. In one of these approaches, dynamic programming is exploited to coherently integrate the visible energy of dim targets over possible relative directions, whereas the other method is a Bayesian formulation for which the target likelihood is updated along time to be able to detect a target moving in any direction. Extensive experiments are conducted for these methods by using synthetic image sequences, as well as some real test data. The simulation results indicate that it is possible to detect dim targets in quite low-SNR conditions. Moreover, the performance might further increase, in case of incorporating any a priori information about the target trajectory.

Keywords: dim target, detection, tracking, infrared imagery, dynamic programming

ÖZ

INFRARED GÖRÜNTÜLERDE
SOLUK HEDEF TESPİTİ

Çifçi, Barış
Yüksek Lisans, Elektrik-Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. A. Aydın ALATAN
Ortak Tez Yöneticisi : Prof. Dr. Mete SEVERCAN

Eylül 2006, 115 sayfa

Bu çalışmada, düşük *işaret-gürültü-oranına* (SNR) sahip görüntüleme senaryolarında bazı soluk hedef tespit algoritmalarının performansı incelenmiştir. Geçmiş araştırmalarda, kızılötesi algılayıcı kullanan askeri izleme uygulamaları için zorlukla görülebilen hedeflerin tespit ve takibine yönelik çok sayıda deneme vardır. Bu çalışmada, kapsamlı simülasyonlarla bu algoritmalardan ikisi incelenmiştir. Bu yaklaşımlardan bir tanesi olan dinamik programlama soluk hedefin görülür enerjisini var olan bağıl yönler üzerinde uygun şekilde biriktirirken, diğer yöntem olan Bayes formülasyonu herhangi bir yönde hareket eden hedefi tespit edebilmek için zamanda hedef olabilirliğini günceller. Hem yapay görüntü dizileri hem de gerçek test verileri kullanılarak deneyler yapılmıştır. Deneyler oldukça düşük *işaret-gürültü-oranı* (SNR) seviyelerinde soluk hedeflerin tespit edilmesinin mümkün olduğunu göstermiştir. Ayrıca, hedef rotası hakkında herhangi bir ön bilgi kullanarak, tespit performansı daha da artabilir.

Anahtar Kelimeler: soluk hedef, tespit, takip, kızılötesi görüntüleme, dinamik programlama

To My Parents

ACKNOWLEDGMENTS

I would like to thank Assoc. Prof. Dr. A. Aydın Alatan for his valuable supervision, guidance, advice, criticism, encouragements, and insight throughout the development and improvement of this thesis.

I would also like to express my deepest gratitude to Mr. İlker Gürel for his interest, advice and support. I am also grateful to ASELSAN Inc. for facilities provided for the completion of this thesis.

I would like to extend my special appreciation to my family for their encouragement; they have given me not only throughout my thesis but also throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Detection and tracking are important issues in military projects, especially at radar applications and video tracking tasks. The problem of detection dim, weak, small, barely discernible, low-observable objects may appear not only in military projects, but also in different fields in research projects or in daily life, such as investigating meteors, satellites, or other small moving objects against night-sky background in astronomy; or in climatology detecting whirlpools in the oceans using the image sequences of earth, which are obtained from down-staring and orbiting satellites [14].

In this aforementioned problem, the main drawback is the missing information about both the location and the velocity of the target in such sequences. Moreover, the clutter in the environment, in which the desired target is roaming and the sensor noise is another challenging part of the task.

In visible, or bright, target cases, video target tracking has two main consecutive phases: detection and tracking [1]. In these approaches, typically, detection are achieved by an optimal linear filter [2][3], followed by an optimal threshold and the outcomes of the detection process are fed to a conventional tracker in order to estimate the target trajectories. Although, the performances of these traditional techniques are sufficient for the bright targets, i.e. targets with high signal-to-noise ratio (SNR), unfortunately, their performance degrades quickly for their

dim counterparts. This degradation is mainly due to the information loss, introduced during the application of the detection threshold.

There is a trade-off for utilizing such a detection threshold. If the detection threshold is not selected sufficiently low to detect targets with low amplitude, a large number of false alarms should be inevitable. Hence, such large number of false alarms is much more than that of the association part of a traditional tracker could handle. Therefore, it is apparent that the classical detect-before-track scheme should be modified, in order to achieve reliable detection in low SNR conditions.

As an alternative to the traditional detect-before-track approaches, the idea of processing a sequence of raw images in order to achieve target detection in very low-SNR conditions has been proposed [4][11]. The motivation behind the idea of such a batch processing approach is to coherently integrate the target energy present in each frame in order to obtain a relatively high-SNR image. However, since the location of the target in each frame is not known in advance, the batch processor should estimate, or hypothesize, the target location in each frame to make the integration along the trajectory of the target. Due to this inherent requirement for tracking possible targets in the scene, before making decisions about the presence of these targets, these techniques are commonly referred to as *track-before-detect* techniques in literature [5],[8],[11],[14].

It is theoretically possible to search all target trajectories in a given sequence of frames to integrate target energy. However, since the number of potential target trajectories grows exponentially with the length of the sequence, such an exhaustive search is computationally unachievable. It is crucial to constrain the search space, as well as to develop efficient search algorithms, in order convert the track-before-detect concept into a practical approach.

There are some proposed techniques to reduce the inevitable computational complexity for solving track-before-detect problem. In Chapter 2, some proposed techniques from the literature will be examined briefly.

1.2 Scope of Thesis

In this thesis, it is intended to examine some methods from the literature with promising solutions for the dim target detection problem. Past researches efforts about this topic are examined and it is observed that computational complexity is one of the important issues in dim target detection. Hence, a method, which is based on the well-known Dynamic Programming approach [9], is selected to overcome this issue. Dynamic Programming is a well-known simple batch processing algorithm, capable of decreasing any search time from exponential to linear order. In this aspect, a relatively novel algorithm is proposed and tested that utilizes dynamic programming to determine the best path between states, which consist of possible relative directions of a target. The state transition metrics are defined based on coherency of target signature between frames, as well as target trajectory consistency.

On the other hand, another algorithm from the dim target detection literature [17], which has a Bayesian formulation, is also fully implemented and tested via simulations. This algorithm is a sequential one, which iteratively determines the likelihood of having a target at a location in time and detects such a target, whenever the likelihood is above some threshold.

Extensive simulations are conducted on these two approaches by using synthetic and reel image sequences in order to asses their performance.

1.3 Outline of Thesis

The outline of this thesis can be briefly summarized as follows:

Chapter 2 consists of a brief summary of the noteworthy former research efforts on dim target detection problem.

Chapter 3 presents a revised version of a dynamic programming-based dim target detection algorithm. In this chapter, some details of this algorithm, such as state selection criteria, some design parameters as well as the velocity model, are stated. The simulation results for this algorithm for both synthetic and real data are also presented towards the end of this chapter.

Chapter 4 is devoted to a recent Bayesian approach from the literature. A detailed mathematical formulation of the algorithm, as well as, the simulation results for the algorithm, are both presented in this chapter.

Finally, Chapter 5 concludes this thesis by performing some comparisons between the implemented algorithms.

CHAPTER 2

PAST RESEARCH
ON DIM TARGET DETECTION

The detection and tracking of low-contrast moving targets in optical image processing have been under intensive investigation for over the past two or more decades. There have been many interesting and promising methods in the literature for dim target detection. Among these techniques, there are 5 fundamental approaches, which cover most of the proposed methods. These methods are 3-D Matched Filters (MF) [5][6], Recursive Moving Target Indicator (RMTI) [8], Dynamic Programming-based dim target detection (DP) [11][12], Sequential Hypothesis Testing (SHT) [13][14] and Bayesian approaches (BA) [15][16][17].

The aforementioned techniques can be classified into two classes, based on their decision mechanism, as *batch* and *sequential* processing. Another taxonomy can also be proposed, according to their processing domain, as *spatial* and *frequency* domains. The classification of these methods based on these classes is tabulated in Table 2.1.

Table 2.1 Classification of the dim target detection algorithms

		Decision Mechanism	
		Batch Processing	**Sequential Processing**
Processing Domain	**Spatial**	- DP [11]	- SHT [14] - BA [17]
	Frequency	- MF [5]	- RMTI [8]

The basic procedure for the detection of a dim target in background clutter is integrating the target energy along the sequence of frames. If the target is stationary, then this procedure is expected to work in an acceptable performance and with such a summation, relatively high SNR values could be obtained. However, if the target is non-stationary with its location and velocity both unknown, brute-force summing might not give the desired result. In order to overcome this problem, one should track the target with known or hypothesized features in advance for integration, hence, detection of the target. Having this nature, mostly these algorithms are denoted, as "track-before-detect" algorithms.

Track-before-detect algorithms are processing strategies designed to track a target with known characteristics through a sequence of optical images. The tracking process increases the effective target energy, while simultaneously reducing the received noise. The major "track-before-detect" methods from the literature are summarized in the following sections.

2.1 3-D Matched Filters (MF)

This technique involves moving target signature matched-filtering in the frequency domain. Specifically, the algorithm provides proper signal phasing to 3-D spatio-temporally transformed image sequence. In [5], the solution for the detection of a moving dim target, having a fixed velocity, is proposed by casting the problem into a general framework of three-dimensional filter theory. From this point of view, the target detection problem reduces to the problem of finding optimal 3-D filters in 3-D transform domain and processing the observed scene via this filtering.

3-D filter theory is a straightforward extension of 1-D filters, which derives optimal filters in a single domain (usually in time domain) for the maximization of a temporal SNR and 2-D filter theory, which derives optimal filters in two

dimensions (mostly in spatial coordinates) for a spatial SNR maximization. Since the application is related with an observation of a spatial area over a fixed time period, these three dimensions should be associated with area and time.

3-D filtering is a form of transform signal processing applied to an observable (imaged scene) in order to detect a hypothesized element (moving target intensity) from among additive background interference (scene clutter).
In one of the proposed MF techniques [5], the problem is reduced to derive optimum 3-D filters, which should be designed in harmony with the known properties of the target, clutter and observing optics. These filters are matched with the target motion and clutter models. Since the filtering is accomplished in the Fourier transform domain, all targets, moving with the same velocity, are detected automatically.

Let $\boldsymbol{r} = (x, y, t)$ be a vector coordinate in 3-D space of time and the Cartesian coordinates (x, y). The observed signal $v(\boldsymbol{r})$ is composed of target and noise components, as

$$v(\boldsymbol{r}) = s(\boldsymbol{r}) + n(\boldsymbol{r}) \qquad \boldsymbol{r} \in \Gamma_3 \tag{2.1}$$

where $s(\boldsymbol{r})$ denotes a target signal function, Γ_3 defines the 3-D region, over which $v(\boldsymbol{r})$ is observed, and $n(\boldsymbol{r})$ is an additive noise, which is stationary, over **r**, with known homogeneous mutual coherence, as

$$R(\boldsymbol{r}, \boldsymbol{r}_1) = E\,[n\,(\boldsymbol{r}_1)\,n\,(\boldsymbol{r}_1 + \boldsymbol{r})] \tag{2.2}$$

A linear filter, $h(\boldsymbol{r})$, is defined that operates on $v(\boldsymbol{r})$ to produce an output

$$y(\boldsymbol{r}) = \int_{\Gamma_3} v(\boldsymbol{u})\, h(\boldsymbol{r} - \boldsymbol{u})\, d\boldsymbol{u} \tag{2.3}$$

It is desired to find the particular linear filter that produces an output $y(\boldsymbol{r})$ at some point $\boldsymbol{r} = \boldsymbol{r}_0$ such that

$$\text{SNR} = \frac{[\text{mean of } y(\boldsymbol{r}_0)]^2}{\text{variance of } y(\boldsymbol{r}_0)} \tag{2.4}$$

is maximized. It is known [7] that SNR is maximized, if $h(\boldsymbol{r})$ is such that its 3-D transform

$$H(\boldsymbol{k}) = \int_{\Gamma_3} h(\boldsymbol{r})\, e^{\,j\boldsymbol{k}\cdot\boldsymbol{r}}\, d\boldsymbol{r} \tag{2.5}$$

Satisfies the following relation

$$H(\boldsymbol{k}) = \frac{S^*(\boldsymbol{k})}{B(\boldsymbol{k})}\, e^{\,-j\boldsymbol{k}\cdot\boldsymbol{r}_0} \tag{2.6}$$

where $S(\boldsymbol{k})$ and $B(\boldsymbol{k})$ are the corresponding 3-D transforms of $s(\boldsymbol{r})$ in (2.1) and $R(\boldsymbol{r})$ in (2.2), respectively, (* denotes complex conjugation). The resulting maximum SNR in (2.4) can be obtained as

$$\text{SNR} = \int_{\Gamma_3} \frac{|\, S(\boldsymbol{k})\,|^2}{B(\boldsymbol{k})}\, d\boldsymbol{k} \quad . \tag{2.7}$$

(2.6) defines the optimal filtering that must be applied in (2.3) for achieving the maximum SNR in (2.7). A detection system, looking for the peak values of $y(\boldsymbol{r})$, will be able to detect $s(\boldsymbol{r})$, if the SNR is maximal at some point. The derivation of the optimal three-dimensional filter at (2.8) is given in [5] and this relation shows that the filter is dependent on the characteristics of the target, clutter, optics and the detector and these parameters are effective on the SNR performance.

$$H(\mathbf{k},\omega) = \frac{G^*(\mathbf{k})A^*(\omega+\mathbf{k}\cdot v_T)D^*(\mathbf{k})H_d^*(\omega)e^{\mathbf{j}k\cdot\mathbf{u}_0}}{B_{sn}(\mathbf{k},\omega)+N_T+B_b(\mathbf{k},\omega+\mathbf{k}\cdot v_b)} \tag{2.8}$$

Following figure presents the block diagram for 3-D matched filtering technique.

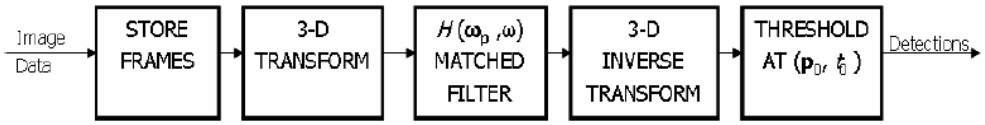


Figure 2-1 General block diagram of 3-D matched filtering solution [6]

The design of the 3-D matched filter requires the prior knowledge of the signal and the statistical characteristics of the additive noise. The primary problem for this technique is the requirement on the filter be matched to a specific velocity. In order to cover the desired ranges of speed, use of filter banks are introduced [5]. However, this might cause velocity mismatch problem, when the observed speed of the target does not exactly equal to those of filter banks.

2.2 Recursive Moving Target Indication (RMTI)

As another dim target detection approach, *Recursive Moving Target Indicator* (RMTI) algorithm [8] creates matched filter peaks, corresponding to specific target movement at enhanced SNRs, as in 3-D matched filtering algorithm. However, this specific technique results with a basic relation, in the form of a linear, constant-coefficient difference equation with a unity magnitude damping factor [8]. Similar to the 3-D matched filtering, RMTI algorithm also aims to integrate the signal energy to achieve better detection.

Let the received intensity, $s(\mathbf{r}, t)$, for a linearly moving target of constant intensity to be written as

$$s(\mathbf{r}, nt_0) = s_0(\mathbf{r} - \mathbf{v}nt_0) \tag{2.9}$$

where $s_0(\mathbf{r})$ is the spatial intensity distribution of imaged target and $\mathbf{v}$ is its vector velocity, n is the discrete time index and t_0 is the time-sampling interval. The observed signal is then,

$$y(\mathbf{r}, nt_0) = s_0(\mathbf{r} - \mathbf{v}nt_0) + n(\mathbf{r}, nt_0) \tag{2.10}$$

where $n(\mathbf{r}, nt_0)$ is the additive noise component. The 2-D spatial Fourier transform of (2.10) is obtained in the form,

$$Y(\mathbf{k}, nt_0) = S_0(\mathbf{k}) \exp\{-i\, \mathbf{k}\cdot \mathrm{v}nt_0\} + N(\mathbf{k}, nt_0) \tag{2.11}$$

where $\mathbf{k}$ denotes two-dimensional spatial wave-number vector. Except for the noise term, (2.11) states that the Fourier transform of each image in the sequence differs from its neighbors by a periodic function. Rewriting (2.11), one obtains

$$Y_n(\mathbf{k}) = S_0(\mathbf{k})\, \alpha^n + N_{\mathrm{n}}(\mathbf{k}) \tag{2.12}$$

where

$$\alpha = \exp\{-\, i\, \mathbf{k} \cdot \mathbf{v}t_0\} \tag{2.13}$$

For obtaining processed result, one should multiply this image by the appropriate matched filter and inverse Fourier transforms. Hence, (2.11) can be written in a recursive manner, as

$$\boldsymbol{X}_n(\mathbf{k}) = \boldsymbol{Y}_n\,(\mathbf{k}) + \alpha X_{n-1}(\mathbf{k}) \tag{2.14}$$

where $\boldsymbol{Y}_n\,(\mathbf{k})$ is the transform of the current observation and $X_{n-1}(\mathbf{k})$ is processed result of the previous observation.

It should be noted that (2.14) is a linear, constant-coefficient difference equation with damping factor equal unity. A spatial matched filter operates on this composite image and a processed picture is obtained after inverse Fourier transformation. The output of the filter is used for threshold detection. If the

predefined threshold is not exceeded, then the processed composite picture is stored for the next observation step, so that the decision can be deferred until a clear detection.

RMTI has similar drawbacks, as in MF; there is the strict requirement for a priori information about the velocity of the target. Obviously, such a constraint limits the applicability of this approach in practical scenarios.

2.3 Dynamic Programming (DP)

Application of Dynamic Programming (DP) for the detection and tracking of low-observable targets was first proposed by Barniv [11]. DP algorithm accumulates the target energy along the sequence of image frames by performing the equivalent of an exhaustive search of all possible target state sequences, representing physically realizable target paths. Hence, DP reduces the complexity of optimum state sequence search from exponential to linear. For a sequence of K frames of images with M x N pixels, the number of unconstrained possible target trajectories is $(M \text{ x } N)^K$, utilizing dynamic programming the number of trajectories are reduced to the number equal to the pixel number of the images, i.e. M x N trajectories. With constant velocity within a specified range assumption, the search space can be narrowed and the computational complexity can be further reduced.

Although, the absolute target locations are selected as DP states in [11], it is also possible to select relative target locations, as proposed in Chapter 3. In [11], each pixel is divided into 4 cells and computations are conducted after such a cell definition. However, selection of absolute locations as states should cause high numbers of states to cope with. Figure 2-2 illustrates the states of the approach in [11] and Figure 2-3 presents an example for the corresponding transitions. Due to defined nominal target speed bounds given in (2.15), there is an annulus where the consecutive states might locate.

$$V_{min} = 1 \ \frac{\text{pixel}}{\text{frame}} \ ; \qquad V_{max} = (G + 0.5) / G \ \frac{\text{pixels}}{\text{frame}} \tag{2.15}$$

where G is the number of frames in a stage.

It should be noted that in Figure 2-2, one quarter of the annulus is sketched; only 41 states are shown, which resides inside the speed bound annulus. However, there are a total of 164 states for a focal cell. Consequently, for a 64x64 size image, the number of all states turns out to be 671.744, which is a quite large number for any practical implementation.

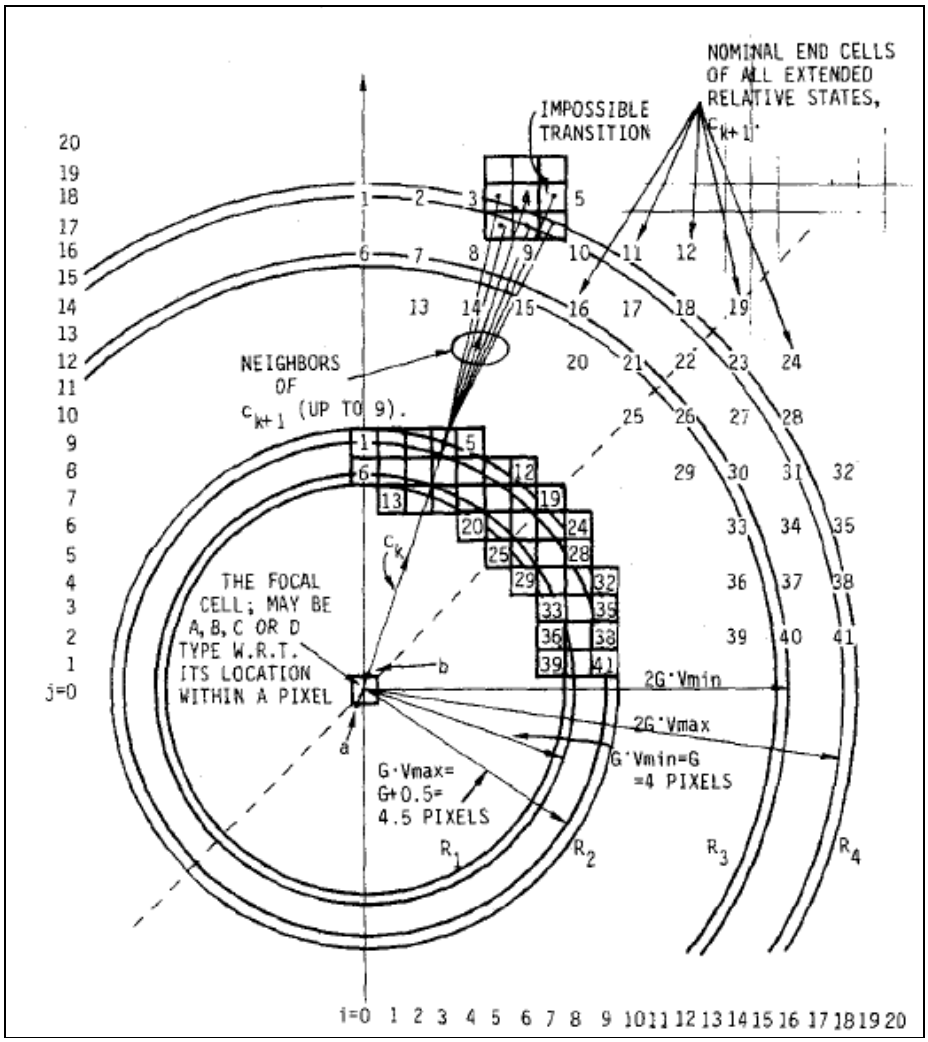


Figure 2-2 Absolute states in [11]

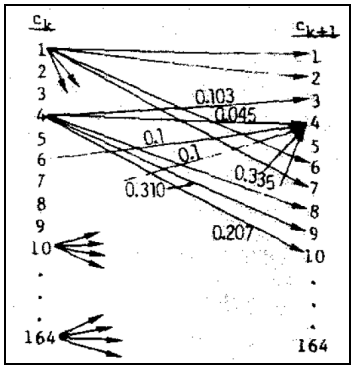


Figure 2-3 State transitions in [11]

In any DP algorithm, the states should be defined according to the requirements of the particular problem. Moreover, the state transition merit functions and state merits should also be defined to construct a trellis, which is the key feature of this algorithm.

A typical DP algorithm can be summarized with the following steps of operation:

1) *Initialization:* For frame $k = 1$; for all states ø

   Initiate the merit function: $I(ø) = z_{ij}(1)$, where $z_{ij}(1)$ is the measurement recorded at resolution cell (i,j) at time instance $k = 1$.

2) *Recursion:* For frame $k = 2, 3, \ldots, K$; for all states ø

   Register the most likely previous state: $\Psi_k(ø) = \arg\max\{I(ø)\}$.

   Update the merit function: $I(ø) \leftarrow \max\{I(ø)\} + z(k)$.

3) *Termination*: Find $\{x_K : I(x_K) > V_T\}$

   Final states with merit function exceeding threshold.

4) *Back-tracing*: For each x_K, construct the estimated track

   $\hat{\mathbf{X}}_K = \{\mathbf{x}(1), \mathbf{x}(2), \mathbf{x}(3), \ldots, \mathbf{x}(K), \}$ such that

   $\mathbf{x}(K) = x_K$, $\mathbf{x}(K-1) = \Psi_K(\mathbf{x}(K))$, $\mathbf{x}(K-2) = \Psi_{K-1}(\mathbf{x}(K-1))$, …,

   $\mathbf{x}(1) = \Psi_2(\mathbf{x}(2))$

   where $\mathbf{x}(k)$ is the state of the target for a specific time instant.

For each frame of the sequence, the states are traced and linked to the most appropriate state in the previous frame. With this approach, a set of linked lists are constructed, where each list ends at a different state in the last frame. The constructed lists are the optimum state sequences ending at a particular state.

A relatively novel version of a DP algorithm is further analyzed, implemented and tested in Chapter 3.

2.4 Sequential Hypothesis Testing (SHT)

Sequential Hypothesis Testing (SHT) is proposed by Blostein [14] for the detection of small, moving objects in image sequences. In SHT, large number of candidate trajectories is organized into a tree structure. In this manner, they are

hypothesized at each pixel in the sequence and tested sequentially for a shift in the mean intensity. A truncated sequential probability test is also used to prune the tree structured list of candidate trajectory segments at each pixel: Summed pixel values are sequentially compared against two thresholds until either the hypothesis that the trajectory contains an object, is rejected or accepted. When the test statistic falls between the two thresholds, this decision is deferred and the trajectory state is stored in a list.

In [14], the concepts of *Fixed Sample Size* (FSS) Test and *Sequential Probability Ratio Test* (SPRT) and a new introduced concept *truncated SPRT* are compared with each other in complexity of test design parameters aspects.

FSS is simply a binary hypothesis testing computed over K samples of observed realization of the random variable of collection of image pixels against a threshold. The number of test sample (K) and the threshold are given as functions of probability of false alarm and probability of detection. Since the object position and velocity are both unknown, the binary hypothesis testing should be used on a subset of the possible finite collections of image pixels. Each subset corresponds to a segment of a hypothesized object trajectory. In this brute-force search, the number of necessary tests at each image pixel will be quite large in order to consider performing exhaustive tests.

For overcoming this mentioned problem of FSS, a revised form, SPRT is introduced, in which at each stage K, a test statistic that is formed from the observations, is sequentially compared to an upper and a lower threshold. SPRT is an alternative to the FSS test. However, as opposed to FSS test, the sample size of SPRT is a random variable. By using SPRT, it is possible that the error probabilities can be achieved with an appropriate selection of thresholds, for which the test will eventually terminate. On the other hand, a disadvantage of SPRT is that occasionally long tests may occur. Moreover, the number of free parameters in SPRT is too large to be optimized. For a K stage test, there must be $2K + 1$ number of parameters to be specified. In [14], it has been shown that the

number of design parameters can be reduced to 4 via a truncated SPRT with boundaries a and b, and truncation point τ, at stage K, where a and b are the lower and upper thresholds until K^{th} observation and τ is the threshold used at K^{th} observation.

In order to detect small, low-contrast objects moving in the image plane, candidate trajectory segments, originating from each pixel in each image are hypothesized and each one is tested by using truncated SPRT. Various restrictions are put on these candidate trajectory segments to control the search space. The search is usually limited to straight paths, typically spanning less than 10 frames. In [14], tree-structured lookup tables are constructed, containing the information of the possible target trajectories with a confined speed of [0,1] pixels/frame range in all directions and test statistics prior to computation of the proposed algorithm. The real trajectories, with real-valued coordinates, are turned out to be discrete trajectory segments for computational purposes and relative locations are stored in these off-lien constructed lookup tables. The construction of candidate trajectories is depicted in Figure 2-4.

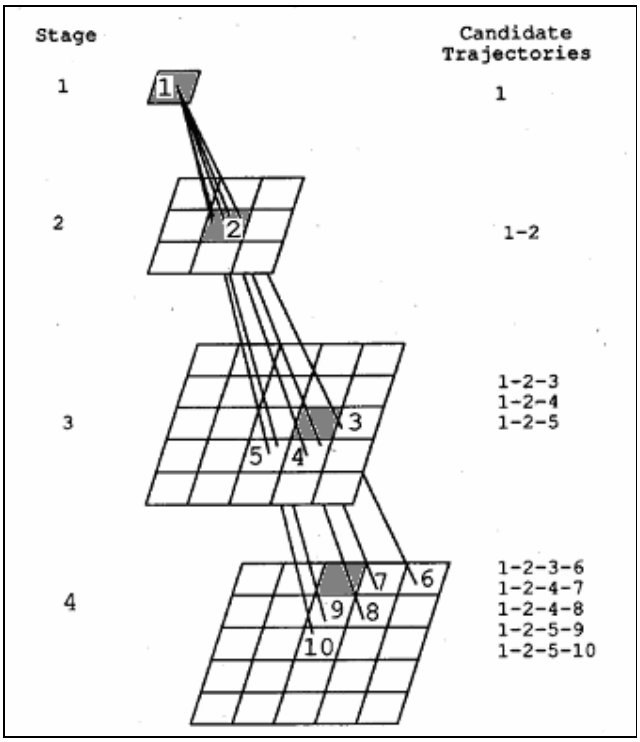


Figure 2-4 Candidate trajectories for truncated SPRT [14].

In the example in Figure 2-4, each plane represents an image in a sequence, whereas the lines represent 5 discrete trajectories. For clarity, the origin of the path is marked in gray color. For every pixel occurring in each frame, the truncated SPRT-based track-before-detect algorithm initiates D K-stages tests, where D is the number of candidate discrete trajectories, and the partial sums along the hypothesized trajectories are compared to upper and lower thresholds.

Although truncated SPRT seems to be confined the number of design parameters, the need for constructing lookup tables in advance, according to some constraints of target motion might reduce the applicability of the algorithm.

2.5 Bayesian Approaches (BA)

When considering the estimation problem of time-invariant parameters of a system, there are two widely used models for those parameters [2]:

- Non-random: The parameter is an unknown true value. (non-Bayesian approach)
- Random: The parameter is a random variable with a (prior) probability density function $p(x)$ (Bayesian approach)

In non-Bayesian approaches, there is no prior probability density function (PDF) and a posterior PDF cannot be defined. In this case, for estimating the parameters, the *maximum likelihood estimation* is used, which is defined as follows:

$$\hat{x}^{\mathbf{ML}}(k) \triangleq \arg\max_{x} p(\mathrm{Z}^{\mathrm{k}} \mid x) \tag{2.16}$$

where Z^k indicates the set of observed samples and x is the unknown parameter. On the other hand, in Bayesian approaches, given the prior PDF of the parameter, one can obtain its posterior PDF by using the well-known *Bayes' rule* :

$$p(x \mid \mathrm{Z}^{\mathrm{k}}) = \frac{p(\mathrm{Z}^{\mathrm{k}} \mid x)\, p(x)}{p(\mathrm{Z}^{\mathrm{k}})} \tag{2.17}$$

The corresponding estimate for a random parameter is the *maximum a posteriori* (*MAP*) estimate, which aims to maximize the numerator of (2.17), since the denominator is independent of x and therefore, irrelevant for this maximization:

$$\hat{x}^{\mathbf{MAP}}(k) \triangleq \arg\max_{x} p(x \mid \mathrm{Z}^{\mathrm{k}}) \triangleq \arg\max_{x} \left[p(\mathrm{Z}^{\mathrm{k}} \mid x)\, \mathrm{p}(x)\right] \tag{2.18}$$

Bayesian solution for *track-before-detect* problem has been investigated in [15] and [16]. In these approaches, the authors independently developed a recursive scheme for the calculation of the likelihood ratio. The tracker described in [15]

operates recursively on a succession of measurements fields over the state space of the target, and it generates as output a Bayesian posterior distribution for the target state. However, the method relies on an assumed "appearance probability". The study in [16] introduces a tracker, called "*Non-Associative Sensor Fusion*" (NASF). NASF is a track-before-detect Bayesian field tracker, which maintains an estimate over some bounded region of target state space, for which the probability density $p(\mathbf{x})$ that a target is present at $\mathbf{X}$. This method also relies on an assumed "appearance probability".

Recently, a novel Bayesian recursion scheme is proposed in [17]. This method estimates the posterior likelihood ratio and posterior state density, based on all past observations. The algorithm detects the presence of a possible target and estimates its position simultaneously. The recursive nature of the proposed solution eliminates the need to store all previous measurements; each received measurement frame is used for updates and discarded. As an important advantage, this technique neither assumes constant velocity motion nor requires a filter-bank implementation structure. The algorithm is also computationally quite efficient and suitable for any real-time operation. BA algorithm is further analyzed, implemented and tested in Chapter 4.

2.6 Discussion

First of all, dim target detection algorithms can be compared in terms of their properties. In general, these algorithms can be classified, as sequential or batch, according to their processing domain. In sequential processing algorithms, such as SHT, BA and RMTI, the decision can be make at any time, whenever the detection threshold is exceed, whereas in batch processing algorithms, e.g. DP and 3-D MF, the decision is stated at the end of the sequence of the frames. Hence, sequential processing algorithms have an advantage over the batch algorithms, since slightly brighter targets might be detected in a relatively faster time without facing any delays due to waiting the whole set of frames in a batch.

MF is the optimal solution for detecting the targets in a cluttered environment. However, this technique requires some priori information about the target, since the velocity and the initial position should be known. However, this information is not available in any practical target detection scenario. Hence, the target trajectories should be hypothesized before using any 3-D matched filtering. Since it is not practical to hypothesize all the target trajectories, filter banks are utilized to overcome this problem with some velocity mismatch effects. Therefore, it can be concluded that MF could only be useful for the scenarios with some reliable a priori information.

On the other hand, RMTI has similar detection properties with that of MF, but less complexity during execution due to its recursive formulation. However, RMTI also requires some a-priori information about the target motion, similar to MF. Unlike 3-D matched filtering, RMTI does not require all the data collected before processing due its recursive nature; rather RMTI processes each frame and generates a final processed picture or stores the result for further processing. However, it has to be matched to a specific velocity vector, as in the case of 3-D matched filtering, which could be overcome partially by using a bank of filters, each of which matched to a specific velocity.

Dynamic Programming does not strictly require some a priori information of the target, such as initial position, velocity. However, any information should be incorporated with the system, in order to decrease the computational complexity. By the help of some intuitive assumptions, DP can be made computationally efficient, if it is formulated to use reasonable number of states and stages.

Similar to MF and RMTI, SHT also assumes target motion to be constant. Unlike DP, SHT handles the exponential increase for the possible number of target trajectories by utilizing some special data structures. Nevertheless, the computation complexity of the overall SHT system could be quite drastic, especially for the low-SNR scenarios.

Finally, Bayesian approaches are simply formulated based on the recursive calculation of likelihood ratio. Obviously, there should be initial random models for target and background before having such a formulation. Nevertheless, the formulation itself does not explicitly require strict a priori information of the target motion. Moreover, the recursive method prevents to store all the past observation and computationally less complex compared to other algorithms considered in this work.

In conclusion, among different approaches for the dim target literature, dynamic programming-based and Bayesian approaches are considered for being more promising for any practical application due to their minimum number of assumptions about target and its trajectory, as well their low computational complexity. The next chapters, two particular methods will be explained and tested by simulations.

CHAPTER 3

DIM TARGET DETECTION VIA DYNAMIC PROGRAMMING

3.1 Fundamentals of Dynamic Programming

Dynamic Programming is a mature approach that has been developed to solve sequential, or multi-stage, decision problems. Dynamic programming, also known as Viterbi algorithm in information theory discipline, reduces the complexity of optimum state sequence search from exponential to linear. This approach is equally applicable for decision problems, where sequential property is induced solely for computational convenience.

Dynamic programming is invented by Richard Bellman, an American mathematician, in 1953. The essence of dynamic programming is Richard Bellman's *"Principle of Optimality"*. This principle, even without rigorously defining the terms, is intuitive:
"An optimal policy has the property that whatever the initial state and the initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." [9]

Dynamic Programming is a method for reducing the runtime of algorithms exhibiting the properties of overlapping sub-problems and optimal substructure. Optimal substructure means that optimal solutions of sub-problems can be used to find the optimal solutions of the overall problem. For example, the shortest path to a goal from a vertex in an acyclic graph can be determined by first computing the shortest path to the goal from all adjacent vertices, and then using this to pick the best overall path, as shown in Figure 3-1.

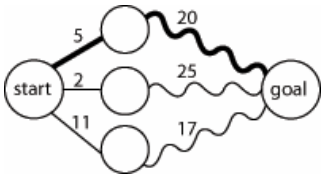


Figure 3-1 Finding the shortest path

In general, one can solve a problem with optimal substructure using a three-step process [10]:

1. Break the problem into smaller sub-problems.
2. Solve these problems optimally using this three-step process recursively.
3. Use these optimal solutions to construct an optimal solution for the original problem.

The sub-problems are, themselves, solved by dividing them into sub-sub-problems, and so on, until one reaches some simple case that is easy to solve.

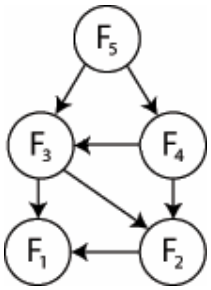


Figure 3-2 The sub-problem graph for the Fibonacci sequence

Figure 3-2 shows the sub-problem graph for the Fibonacci sequence. In other words, it is not a tree, but a directed acyclic graph indicates overlapping sub-problems. In order to state that a problem has overlapping sub-problems, the same sub-problems should be used to solve many different larger problems. For example, in the Fibonacci sequence, F3 = F2 + F1 and F4 = F3 + F2 computing

each number involves computing F2. Since both F3 and F4 are needed to compute F5, a brute-force approach to computing F5 may end up computing F2 twice or more. This applies, whenever overlapping sub-problems are present: a brute-froce approach might waste time by re-computing optimal solutions to the sub-problems that it has already solved.

In order to avoid this situation, one should save the solutions to problems that have been already solved, instead. Then, if the same problem is required to be solved later, one can retrieve and reuse the already-computed solution. This approach is called "memoization" (not memorization, although this term also fits) [10]. In some cases, one can even compute the solutions to sub-problems, that it is known to be required in advance.

In summary, dynamic programming makes use of:

- Overlapping sub-problems
- Optimal substructure
- "Memoization"

Dynamic programming usually takes one of two approaches:

- **Top-down approach**: The problem is broken into sub-problems, and these sub-problems are solved and the solutions remembered, in case they need to be solved again. This is the case in which recursion and "memorization" are combined together.
- **Bottom-up approach**: All sub-problems that might be needed are solved in advance and then used to build up solutions to larger problems. This approach is slightly better in stack space and number of function calls, but it is sometimes not intuitive to figure out all the sub-problems needed for solving given problem.

A top-down example is given in Figure 3-3.

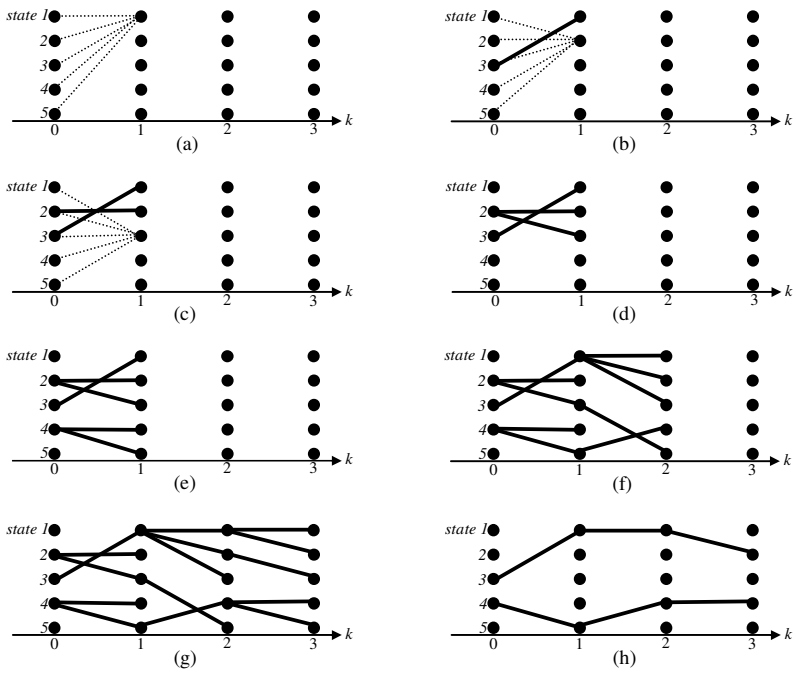


Figure 3-3 Various stages of the DP algorithm for a system of 5 states [17]

Figure 3-3 shows a system with 5 states and the decision is finalized at stage-3. Figure 3-3 (a) represents the search for the optimal previous state for state-1 at k=1. It is observed that state-3 is chosen, as the optimal one at Figure 3-3 (b). The complete selection of optimal previous states for k=1 is shown by Figure 3-3 (e) and for k=2 and k=3 by Figure 3-3 (f) and Figure 3-3 (g), respectively. The remaining state sequences (tracks) after thresholding are shown in Figure 3-3 (h).

3.2 Algorithm for Dim Target Detection

In the following algorithm, the fundamental concepts of the dynamic programming technique are adapted for the dim target detection problem. These basic definitions are given in Section 3.2.1.

3.2.1 Definitions

It is assumed that the image sequence is captured by a stationary sensor, i.e. the only moving object is the dim target, if any. In order to simplify the problem, the dim target is assumed to be 1-pixel in size and moves 1 pixel/frame to one of the 9 possible directions (including staying at the same location) at each frame. These 9 directions are as follows: North (N), North East (NE), East (E), South East (SE), South (S), South West (SW), West (W), North West (NW) and Stay at Center (C). Since the target in real world will be in a far distance and far targets motion is observed in low in speed, C is also considered as an option.

In terms of dynamic programming formulation, each of these 8+1 directions is taken as a "state". Every time instant (frame) is assumed as one stage to be processed. The possible dim target (I_x) and its 8 neighbors are given in Figure 3-4,

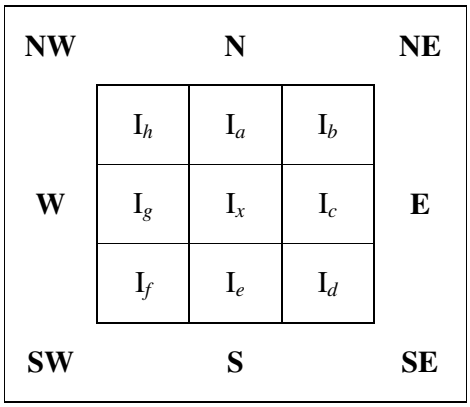


Figure 3-4 States of Dynamic Programming implementation

where *I* stands for the gray-level (illumination) of the pixels.

State benefit and transition probability are key concepts of dynamic programming technique. State benefit is a metric which is assigned to a state. In this problem,

state benefit is assumed to equal, since there is no a priori information about the directions of the target. Transition probability is the probability of switching from one state at a stage to a new state at the following stage and calculated by using a suitable metric, which matches to the problem that is solved by using dynamic programming. The appropriately selected metrics for dim target detection problem are given in Section 3.2.4.2.

3.2.2 Constructing Trellis

For implementing dynamic programming in dim target detection problem, a state transition diagram, called as *trellis*, as shown in figure below, is constructed.

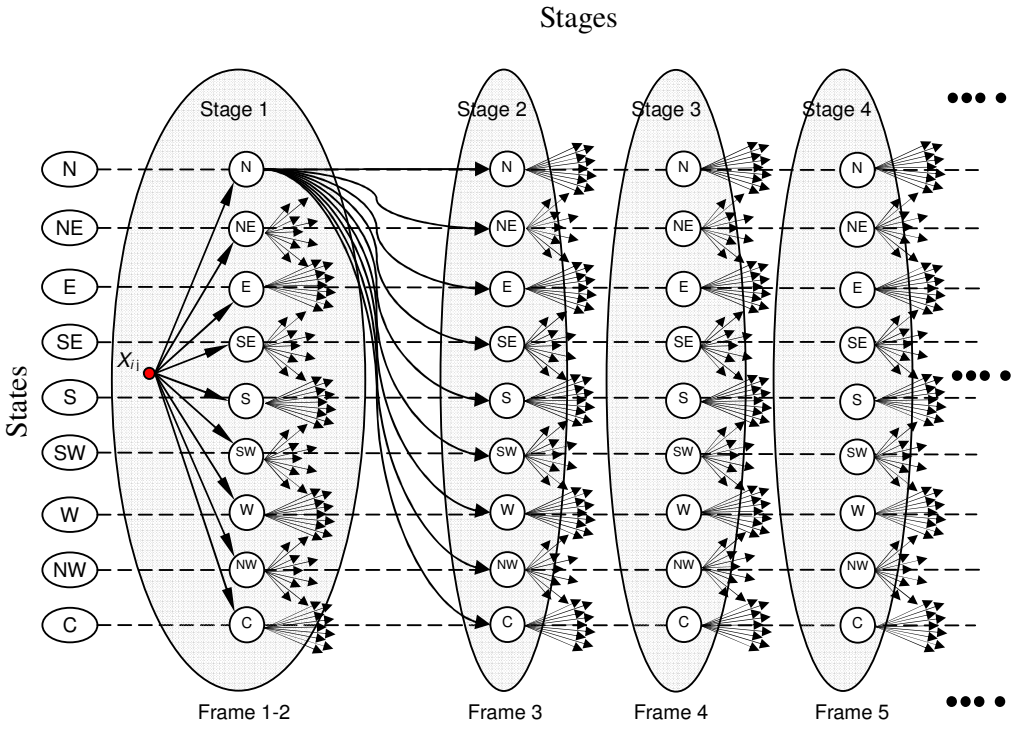


Figure 3-5 Trellis structure

In order to form a trellis matrix accompanying to the transitions of Figure 3-5, its members can be calculated by using different transition metrics. For N-stage dim target computation, the size of the trellis matrix will be 9xN. Trellis matrix is constructed by using a state transition matrix of size 9x9. The size of the state transition matrix is independent from either the number of the stages or the size of the frames processed. The elements of state transition matrix store the sum of the state benefit of source state, the transition probability between source and target and the target state benefit. Each column of the state transition matrix stores to the total scores of the paths from one state to 9 different target states, so that each row stores the path scores form 9 different source states to one target state. Elements of the state transition matrix are refreshed at every state transition. For a particular state transition the maximum value of each row of the state transition matrix is transferred to the corresponding trellis matrix row.

Figure 3-6 illustrates an example for state transition and trellis matrices, evaluation for a 5-stage execution. At the initialization phase, the elements of the matrices are all initialized to zero.

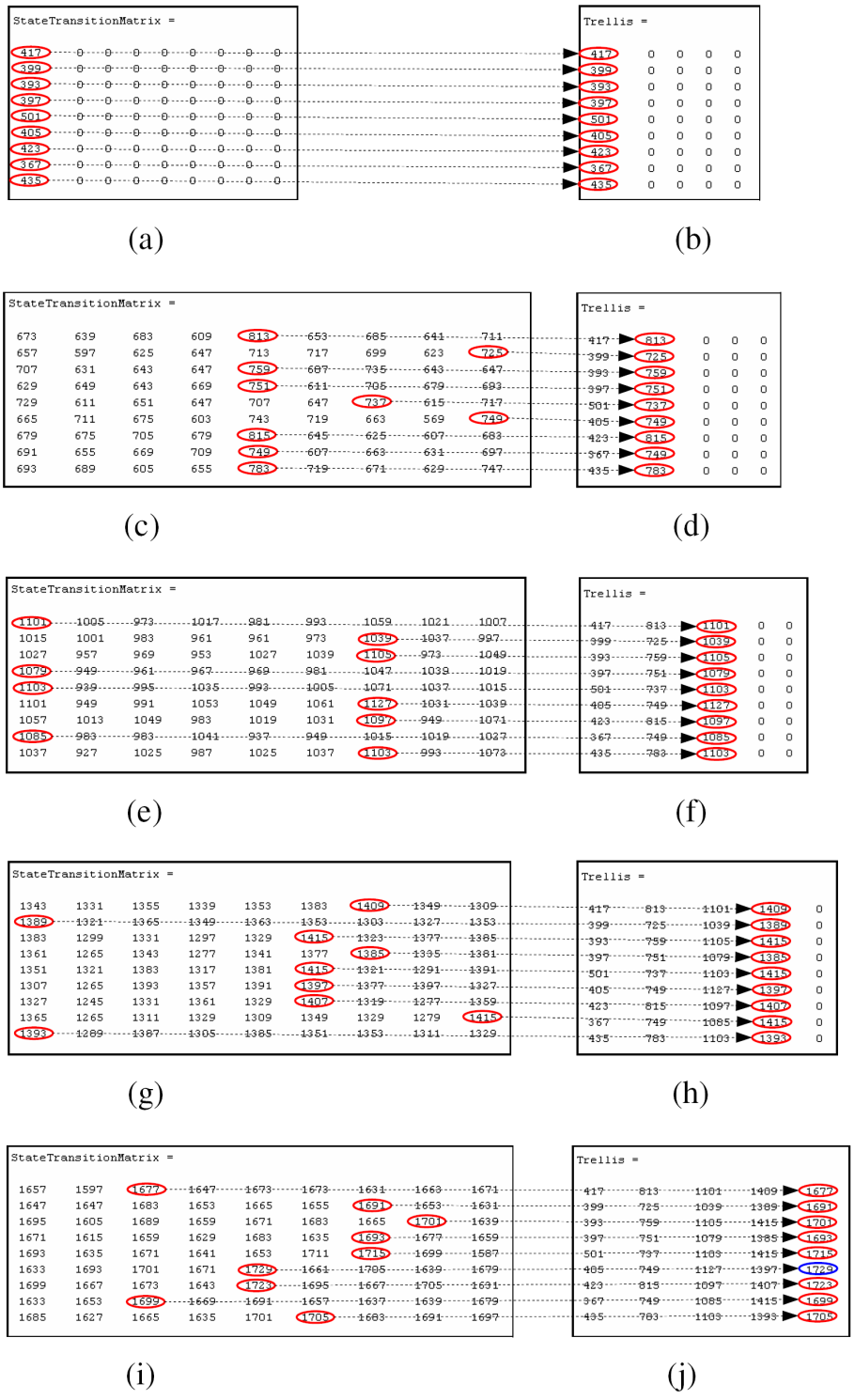


Figure 3-6 An example construction of a trellis matrix for a 5-stage execution

At every stage, the transition probabilities for 81 transitions are calculated and stored in a volatile state transition matrix, instead of storing the scores of all these 81 transitions, only highest scores for 9 state transitions are stored in trellis matrix for a given stage. In other words, for any stage with 9 states only 9 scores are stored instead of a total of 81 transition scores.

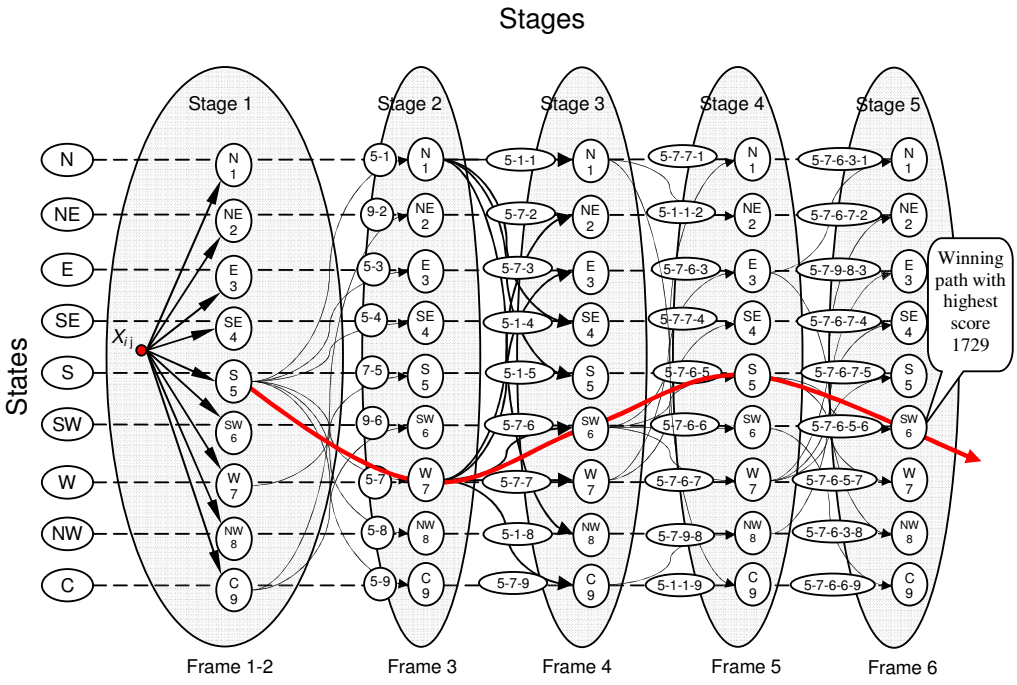


Figure 3-7 Illustration of the trellis matrix

Figure 3-7 shows the methodology of score assignment for the trellis in Figure 3-6 and paths formation.

3.2.3 Implementation

The operations mentioned below will be applied on *every* pixel on the image with size MxN.

Initialization Stage:

The first image of the sequence is captured and the pixel at position i,j is taken as starting point of the target. All elements of trellis matrix and state transition matrix are initialized to zero. Next, a second frame is captured from the sequence of frames and the elements of the state transition matrix are calculated, according to the selected state transition metric (details of state transition metrics can be found in Section 3.2.4.2.). At the end of processing, there will MxN paths with different scores. The path score is the sum of all transition probabilities on a particular path.

Recursive stages:

The next frame is captured from the frame sequence and the elements of the state transition matrix are calculated. The elements of the column of the trellis matrix corresponding to the current stage obtained from the state transition matrix with the highest score of each row. While storing the highest score of each row of the state transition matrix, the absolute location of the target state and the direction is also stored in different matrices. This iterative operation is repeated until the last frame of the sequence. When last frame is reached the values in last column of the trellis matrix which are also the scores of each path can be used, after thresholding in order to declare detection. The threshold value should be determined as a result of some experiments.

3.2.4 Algorithm Parameters

3.2.4.1 State Selection

As it has been mentioned before, the states are selected as the relative locations of the pixels, i.e. towards the direction of the target motion. With the 1 pixel/frame speed assumption, these states are selected as 8-neighbors of the pixel of interest (N, NE, E, SE, S, SW, W, NW). In order to cover the stationary targets, staying at the same location, the state CENTER (C) is also considered as the 9^{th} state.

In case of considering target speed up to 2 pixels/frame, not only the 1^{st} order 8 neighbors, but also the 2^{nd} order 16 neighbors should also be included in the state diagram to obtain a total of 25 states.

Unlike in Barniv's work, the selection of relative locations as states, instead of using absolute locations, dramatically reduces the size of the resulting trellis and computationally more effective.

3.2.4.2 Selection of Transition Metrics

While implementing the proposed algorithm, different transition probability models are introduced and examined. It should be noted that it is preferable to denote these values as *transition metrics*, since these values do not meet the probability axioms. These metrics should be chosen carefully, so that they should be useful, while solving the dim target detection problem by using dynamic programming.

In this thesis, the following 3 metrics are introduced:

- Metric-1 : Absolute intensity metric $m1 = I_b$,
- Metric-2 : Contrast-based metric $m2 = I_b - \text{avg}(I_b)$,
- Metric-3 : Normalized contrast $m3 = (I_b - \text{avg}(I_b)) / \sigma_N$,

where I_b is the intensity of the next target position to be inspected. The index b stands for the observed pixel.

Absolute intensity metric, $m1$, uses the intensity of the target pixel, since the intensity information reflects the absolute temperature in an infrared image. By defining such a metric, then it would be possible to accumulate the absolute temperature information for the dim target along frames.

On the other hand, contrast-based metric, $m2$, is the difference between the intensity of the target pixel and the average neighborhood intensity of the same pixel. Average neighborhood intensity is calculated by adding intensity values of 8 neighbors and dividing by 8. Such a metric is useful for the cases in which the neighboring background and target has enough temperature difference that is sufficient for detection of the target. For example, although the absolute temperature of the target is not "hot" enough, it should be still possible to detect this target, if it causes sufficient temperature difference between its neighbors.

Finally, Normalized contrast metric, $m3$, is the normalization of $m2$ by the standard deviation of the background noise (σ_N), which can be estimated reliably in the dim target detection problem, since the effect of target is negligible for any statistical parameter estimation for the background. Such a metric is expected to be useful, in order to determine a single detection threshold in any background noise level.

All these metrics are tested via simulations for different image noise levels and target trajectories and the results are presented in Section 3.3.

3.2.4.3 Velocity Model

Most of the track-before-detects methods in the literature make use of some a priori information about the absolute velocity of the target. However, the

proposed dynamic programming-based approach does not assume any velocity for the target, except for the assumption of small displacements. Although, it is impractical to assume the value of any velocity vector, it should be a neat idea to support the continuity of its motion along its trajectory. In other words, without making any assumptions on the velocity vector, it is intuitively quite acceptable to punish abrupt motion vector changes along its present trajectory.

A velocity model can be incorporated into the proposed dynamic programming approach by utilizing an additive parameter, δ, which is incorporated into the transition metrics. This parameter might increase or decrease the transition value and at the same time, the overall path score, according to the motion model of the target. The following figures show the various proposed alternatives to determine the velocity model parameter, δ, which is based on the assumption of a smooth target trajectory. As it is observed from these figures, any target following the same directions as its previous direction (e.g. from N to N) will be assigned 2δ, whereas sharp directions changes, such as from N to S will get -2δ, during the determination of the "best" path with maximum transition values. The velocity values in between these two extremes are relatively ad-hoc, but they can be derived based on a priori information about typical target motions. An example for the effect of velocity model on the path score for the same example is depicted in Figure 3-8.

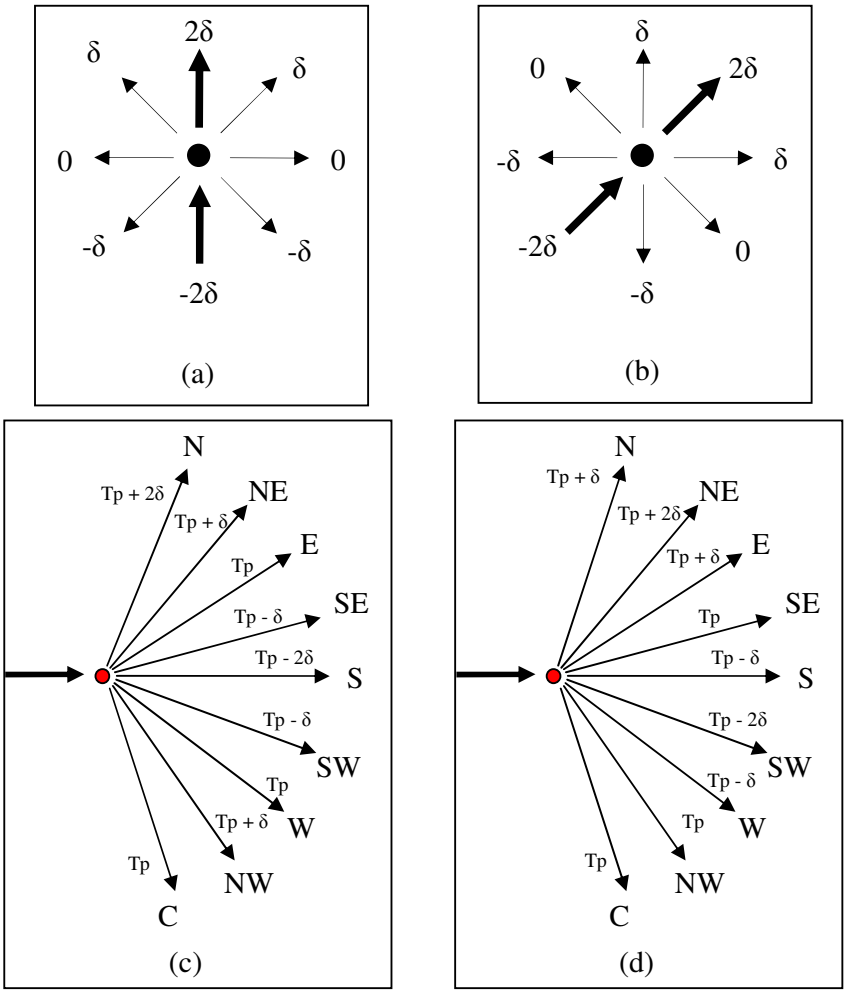


Figure 3-8 Illustrative example for velocity model

3.3 Simulations

In this section, the proposed DP-based dim target detection algorithm is tested by using artificial and real data sets. The artificial simulations are conducted for different noise levels, velocity models, δ, and target trajectories.

3.3.1 Artificial Data Set Tests

3.3.1.1 Test Setup

A number of sequences are generated in order to model the infrared images. In these sequences, the background model of an image is assumed to be Gaussian, with a mean gray level of 128, while the target has a constant intensity of value 141. In this thesis, the aim is to detect a dim target with 5% contrast, and hence, the target pixel gray-level is calculated as 141, based on the formula given in (3.1)

$$I_{(target)} = \left[255 \text{ x } \frac{5}{100} \right] + I_{(background)} \qquad (3.1)$$

For this target-background pair, different sequences are generated with

- various background noise levels and
- a variety of target trajectories.

The image sequences are 30-200 frames in length and their frame size is equal to 64x64 pixels.

6 video sequences are generated having different target trajectories,

- Video-1: The target is one-pixel in size with gray-level of 141 and its initial location is at (10,10). The target is moving with a speed of 1 pixel/frame to the direction East (E). All the other pixels of the images in this Video-1 sequence have gray-level 128 for the noise-free case. The corresponding target trajectory is illustrated in Figure 3-9.

- Video-2: Video-2 differs from Video-1 only in terms of the speed of the target. The average speed of target in Video-2 is 0.5 pixel/frame on the average with the same direction to that of Video-1. The initial location of the target is (10, 10) and the target moves one pixel to E and stops at that

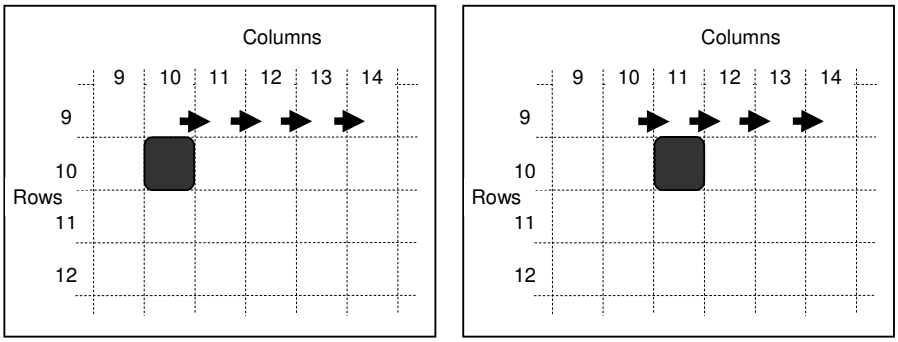
location for the next frame, and continues in this manner, periodically. The target trajectory is given in Figure 3-10.

- Video-3: This sequence has a target which has a circular trajectory, instead of a linear trajectory. The initial location of this maneuvering target is (10, 45). The speed of the target is 1 pixel/frame. The target trajectory is presented in Figure 3-11.

- Video-4: In Video-4, the target trajectory is similar to that of Video-2. The initial location of the target is (10, 10). The target moves one pixel to East at every fourth frame of the sequence; it stops at that location for the next 3 frames. The target trajectory is given in Figure 3-12.

- Video-5: The target resides always at the same location, (32, 32), along the sequence of frame in Video-5. The target trajectory is given in Figure 3-13.

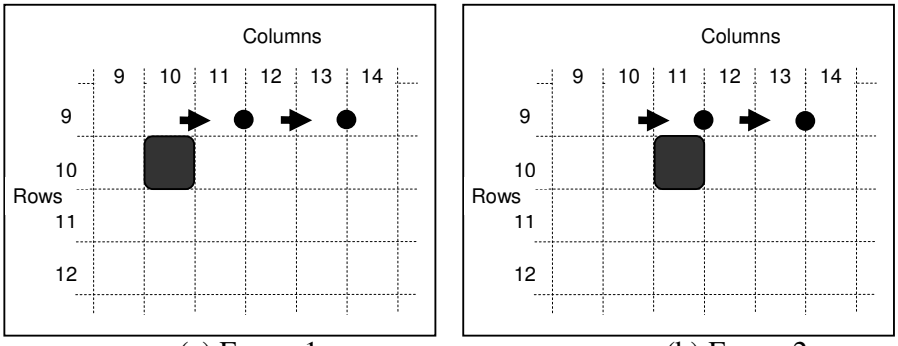
- Video-6: The target in Video-6 is not one pixel in size. The target has a size of 3x3 and moves one pixel to East at every fourth frame of the sequence; it waits at that location for the next 3 frames, similar to Video-4. The initial location of the target is (10, 10). The target trajectory is shown in Figure 3-14.



(a) Frame 1 (b) Frame 2

(c) Frame 3 (d) Frame 4

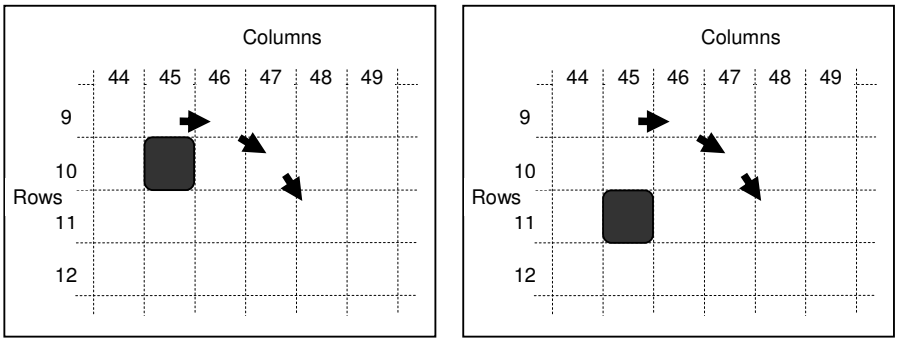
Figure 3-9 A sample view of Video-1



(a) Frame 1 (b) Frame 2

(c) Frame 3 (d) Frame 4

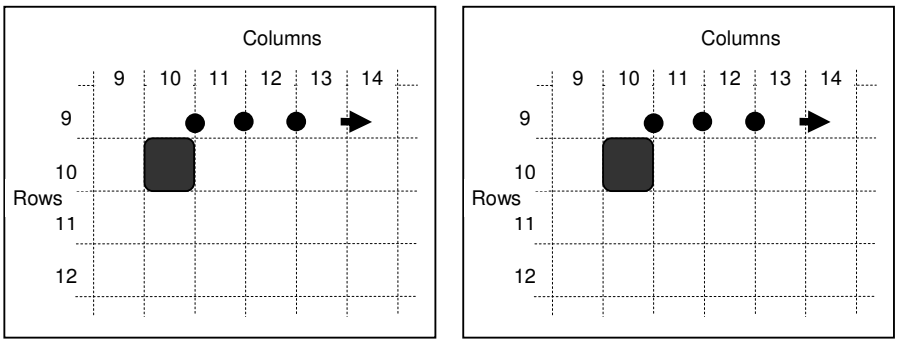
Figure 3-10 A sample view of Video-2



(a) Frame 1 (b) Frame 2

(c) Frame 3 (d) Frame 4

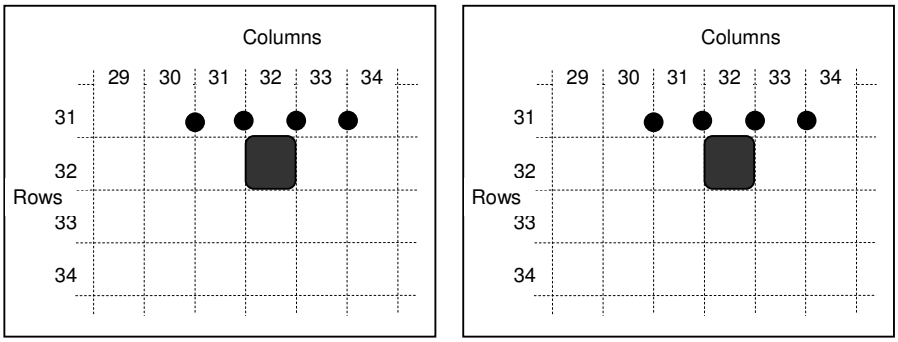
Figure 3-11 A sample view of Video-3



(a) Frame 1 (b) Frame 2

(c) Frame 3 (d) Frame 4

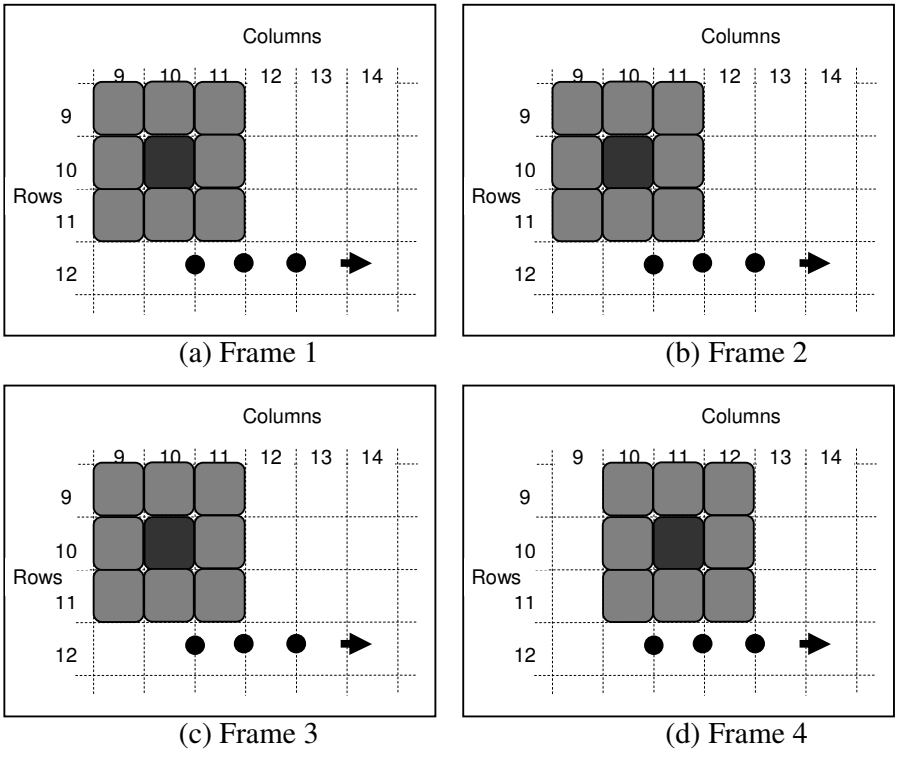
Figure 3-12 A sample view of Video-4



(a) Frame 1 (b) Frame 2

(c) Frame 3 (d) Frame 4

Figure 3-13 A sample view of Video-5



(a) Frame 1 (b) Frame 2

(c) Frame 3 (d) Frame 4

Figure 3-14 A sample view of Video-6

3.3.1.2 Results for Various Background Noise Levels

In order to simulate infrared imagery due to different sensors, Gaussian distributed noise, whose standard deviations are equal to 1, 5, 7, 10 and 13, are added onto these sequences. The first frames of Video-1 for different noise levels are presented in Figure 3-15.

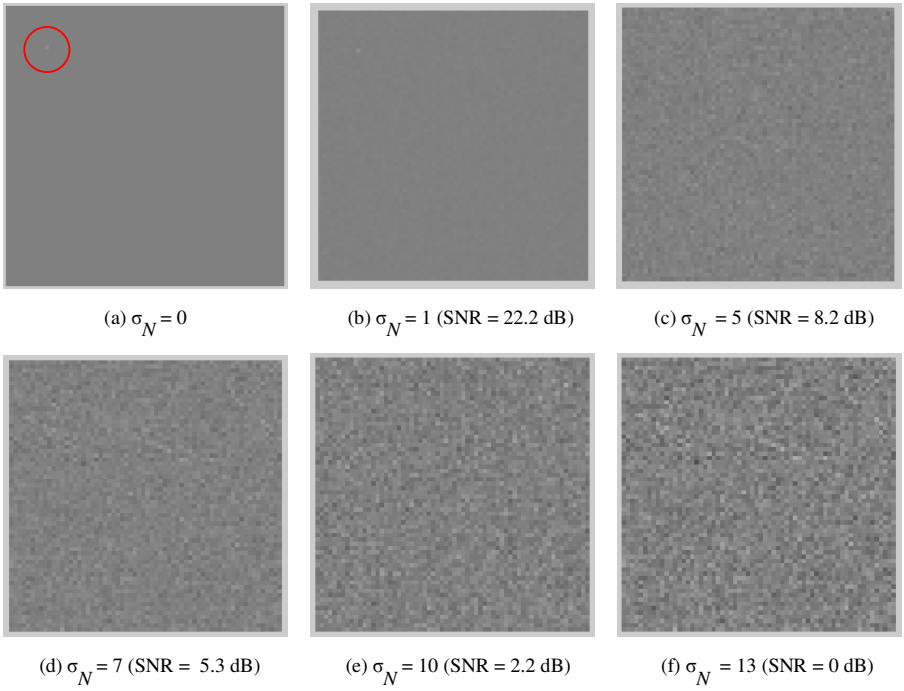


Figure 3-15 Video-1 with different noise levels.

In all the simulations, SNR is calculated, by the following formula

$$\mathrm{SNR} = 20 \log \frac{I_{(\mathrm{target})} - I_{(\mathrm{background})}}{\sigma_N} \tag{3.2}$$

3.3.1.3 Simulation Results

The simulations are conducted for a number of target trajectories, different state transition metrics and various values of velocity model parameter, δ. The target trajectories in sequences, Video-1, Video-2 and Video-3 are utilized during these simulations. The proposed state transition metrics are compared during these simulations. All the results are tabulated at Tables 3.1–3.6. At these tables, the different noise levels of the input image sequences can be observed in columns. In order to observe the effects of different initial locations, some cases, corresponding to the locations other than the true points, are also included. These starting locations are selected such that one of them is the true location of the target trajectory, whereas the other two locations are selected from neighborhood of the start point, and finally, completely different location is also chosen, which is far away from the true target trajectory. Since the true target trajectory is also known in advance, as a ground-truth, a distance measure between true target path and the path determined by the corresponding simulation is also presented. This measure is defined as the sum of the pixel distance for every frame in the sequence. In all these tables, the symbol, "✓", stands for the reaching to the correct final position of the target trajectory with highest path, whereas the symbol, "👎", denotes that dynamic programming method has failed to locate the target.

In Table 3.1, the simulation results of the proposed DP-based dim target detection algorithm for Video-1 are presented. For all different noise levels, up to noise-levels with the standard deviation 5, the algorithm marks successfully the trajectory, for the case starting at the true initial point (10,10). Since the trajectories, which initiates from the locations (20,20), have lower path scores, this result gives the ability to define a threshold for detection of dim targets. However, for the higher noise levels, above $\sigma_N = 7$, the detector fails to determine the trajectory for all 3 metrics. It should be noted that for different noise levels, the detection threshold between target and no-target path scores should be adapted appropriately. Hence, Metric-3, which normalizes the path score

according to the noise level, is designed to overcome this problem, to give the detector the ability to detect the target trajectory with a single threshold.

Table 3.1 Simulation results for Video-1 with $\delta = 0$

Path Starting Location	Background Noise Level	Noise-free	$\sigma_N = 1$ $\sigma_N^2 = 1$	$\sigma_N = 5$ $\sigma_N^2 = 25$	$\sigma_N = 7$ $\sigma_N^2 = 49$	$\sigma_N = 10$ $\sigma_N^2 = 100$	$\sigma_N = 13$ $\sigma_N^2 = 169$
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(10,10)	*Best path score*	**705**	**703**	**704**	727	721	752
	Displacement error	0	0	0	1	37	14
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(11,15) (✓)	(14,7) (✓)	(9,11) (✓)
(10,13)	*Best path score*	692	691	689	717	730	747
	Displacement error	5	5	6	7	19	7
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(11,15) (✓)	(9,9) (✓)	(11,15) (✓)
(9,9)	*Best path score*	640	649	685	695	741	742
	Displacement error	42	19	18	17	26	17
		(4,9) (✓)	(10,12) (✓)	(10,11) (✓)	(9,11) (✓)	(8,8) (✓)	(9,11) (✓)
(20,20)	*Best path score*	640	648	668	704	715	759
	Displacement error	90	113	93	103	92	103
		(15,20) (✓)	(18,24) (✓)	(19,18) (✓)	(19,19) (✓)	(17,18) (✓)	(19,19) (✓)
	State Transition	**Metric - 2**					
(10,10)	*Best path score*	**65**	**63.375**	**61**	90.125	98.250	122.625
	Displacement error	0	0	0	1	23	14
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(11,15) (✓)	(8,8) (✓)	(9,11) (✓)
(10,13)	*Best path score*	52	50.625	50.250	80.375	98	114.500
	Displacement error	6	6	6	7	5	7
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(11,15) (✓)	(10,15) (✓)	(11,15) (✓)
(9,9)	*Best path score*	0	10	41.750	61.500	103.375	117.125
	Displacement error	42	32	39	17	26	17
		(4,9) (✓)	(12,8) (✓)	(7,9) (✓)	(9,11) (✓)	(8,8) (✓)	(9,11) (✓)
(20,20)	*Best path score*	0	9.250	32.125	68.250	75.875	127
	Displacement error	90	98	102	111	99	111
		(15,20) (✓)	(22,16) (✓)	(18,22) (✓)	(23,20) (✓)	(21,17) (✓)	(23,20) (✓)
	State Transition	**Metric - 3**					
(10,10)	*Best path score*	**65**	**63.375**	**9.540**	10.640	9.258	7.729
	Displacement error	0	0	0	1	24	14
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(11,15) (✓)	(8,8) (✓)	(9,11) (✓)
(10,13)	*Best path score*	52	50.625	9.110	10.441	8.329	7.594
	Displacement error	6	6	6	7	15	16
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(11,15) (✓)	(10,15) (✓)	(9,11) (✓)
(9,9)	*Best path score*	0	10	8.335	7.699	9.258	7.696
	Displacement error	42	32	17	17	26	17
		(4,9) (✓)	(12,8) (✓)	(10,11) (✓)	(9,11) (✓)	(8,8) (✓)	(9,11) (✓)
(20,20)	*Best path score*	0	9.250	6.085	7.929	6.725	7.817
	Displacement error	90	98	102	111	93	111
		(15,20) (✓)	(22,16) (✓)	(18,22) (✓)	(23,20) (✓)	(17,18) (✓)	(23,20) (✓)

Table 3.2 Simulation results for Video-1 with $\delta = 15$

Path Starting Location		**Noise-free**	**$\sigma_N = 1$ $\sigma_N^2 = 1$**	**$\sigma_N = 5$ $\sigma_N^2 = 25$**	**$\sigma_N = 7$ $\sigma_N^2 = 49$**	**$\sigma_N = 10$ $\sigma_N^2 = 100$**	**$\sigma_N = 13$ $\sigma_N^2 = 169$**
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(10,10)	*Best path score*	**825**	**823**	**824**	**845**	**833**	**861**
	Displacement error	0	0	0	0	0	0
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(10,15) (✓)
(10,13)	*Best path score*	773	772	781	777	793	804
	Displacement error	9	9	20	27	32	24
		(10,13) (↰)	(10,13) (↰)	(10,8) (↰)	(5,11) (↰)	(14,18) (↰)	(5,16) (↰)
(9,9)	*Best path score*	760	765	781	795	830	836
	Displacement error	42	47	42	18	27	18
		(4,9) (↰)	(14,4) (↰)	(9,4) (↰)	(6,14) (↰)	(9,9) (↰)	(6,14) (↰)
(20,20)	*Best path score*	760	763	775	783	807	825
	Displacement error	90	75	105	105	108	103
		(15,20) (↰)	(15,15) (↰)	(15,25) (↰)	(20,20) (↰)	(25,17) (↰)	(19,19) (↰)
	State Transition	**Metric - 2**					
(10,10)	*Best path score*	**185**	**183**	**181**	**207**	**196**	227
	Displacement error	0	0	0	0	0	1
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(11,15) (↰)
(10,13)	*Best path score*	129.750	129.875	137.500	138	156.750	166.500
	Displacement error	9	9	20	27	32	28
		(10,13) (↰)	(10,13) (↰)	(10,8) (↰)	(5,11) (↰)	(14,18) (↰)	(14,18) (↰)
(9,9)	*Best path score*	120	125.500	138.125	164.375	190.375	203
	Displacement error	42	47	42	47	27	47
		(4,9) (↰)	(14,4) (↰)	(9,4) (↰)	(14,4) (↰)	(9,9) (↰)	(14,4) (↰)
(20,20)	*Best path score*	120	124.500	134.750	141.625	175.375	185.375
	Displacement error	90	75	108	105	108	108
		(15,20) (↰)	(15,15) (↰)	(17,25) (↰)	(20,20) (↰)	(25,17) (↰)	(23,18) (↰)
	State Transition	**Metric - 3**					
(10,10)	*Best path score*	**185**	**183.375**	**129.540**	**130.194**	**127.117**	**126.373**
	Displacement error	0	0	0	0	0	0
		(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(10,15) (✓)	(10,15) (✓)
(10,13)	*Best path score*	129.750	129.875	124.120	122.092	122.974	122.018
	Displacement error	9	9	20	24	33	24
		(10,13) (↰)	(10,13) (↰)	(10,8) (↰)	(15,13) (↰)	(15,18) (↰)	(15,13) (↰)
(9,9)	*Best path score*	120	125.500	123.075	124.242	126.318	124.104
	Displacement error	42	47	47	47	27	47
		(4,9) (↰)	(14,4) (↰)	(14,4) (↰)	(14,4) (↰)	(9,9) (↰)	(14,4) (↰)
(20,20)	*Best path score*	120	124.500	122.430	121.638	123.936	121.773
	Displacement error	90	75	105	120	105	120
		(15,20) (↰)	(15,15) (↰)	(15,25) (↰)	(20,25) (↰)	(25,15) (↰)	(20,25) (↰)

In Table 3.2, the simulation results for the velocity model parameter, $\delta = 15$, are presented. Compared to Table 3.1 the paths starting with location (10,10) have the highest path scores as desired at all noise levels, which means that utilizing

the velocity parameter for the targets, increases the success of the algorithm, considerably.

Table 3.3 Simulation results for Video-2 with $\delta = 0$

Path Starting Location		**Noise-free**	**$\sigma_N = 1$ $\sigma_N^2 = 1$**	**$\sigma_N = 5$ $\sigma_N^2 = 25$**	**$\sigma_N = 7$ $\sigma_N^2 = 49$**	**$\sigma_N = 10$ $\sigma_N^2 = 100$**	**$\sigma_N = 13$ $\sigma_N^2 = 169$**
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(10,10)	*Best path score*	**705**	**705**	**701**	**704**	725	753
	Displacement error	0	0	0	1	10	12
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(8,11) (✓)
(10,13)	*Best path score*	692	693	696	698	727	767
	Displacement error	5	5	5	5	14	13
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(11,14) (✓)
(9,9)	*Best path score*	692	694	700	704	732	795
	Displacement error	3	3	3	3	16	34
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(4,10) (✓)
(20,20)	*Best path score*	640	648	678	691	734	736
	Displacement error	96	121	118	118	111	121
		(15,20) (✓)	(23,19) (✓)	(21,19) (✓)	(21,19) (✓)	(21,18) (✓)	(22,20) (✓)
	State Transition	**Metric - 2**					
(10,10)	*Best path score*	**65**	**65.125**	**61.750**	**67.625**	87.375	129.250
	Displacement error	0	0	0	1	10	16
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(8,11) (✓)
(10,13)	*Best path score*	50.375	51.875	55.375	58.375	94.625	124
	Displacement error	5	5	5	5	14	13
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(10,15) (✓)
(9,9)	*Best path score*	50.375	52.875	61.500	67.625	88.500	150.750
	Displacement error	3	3	3	3	30	34
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(7,7) (✓)	(4,10) (✓)
(20,20)	*Best path score*	0	8.250	40.125	55.500	103.375	105
	Displacement error	96	125	118	114	110	112
		(15,20) (✓)	(23,22) (✓)	(21,19) (✓)	(21,19) (✓)	(21,17) (✓)	(19,20) (✓)
	State Transition	**Metric - 3**					
(10,10)	*Best path score*	**65**	**65.125**	**11.130**	**8.253**	8.332	8.336
	Displacement error	0	0	0	1	10	16
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(8,11) (✓)
(10,13)	*Best path score*	50.375	51.875	10.875	8.064	8.405	8.451
	Displacement error	5	5	5	5	14	12
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(18,15) (✓)
(9,9)	*Best path score*	50.375	52.875	11.120	8.156	8.332	11.330
	Displacement error	3	3	3	30	12	34
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,7) (✓)	(9,9) (✓)	(4,10) (✓)
(20,20)	*Best path score*	0	8.250	7.405	7.036	8.357	6.506
	Displacement error	96	125	118	118	112	112
		(15,20) (✓)	(23,22) (✓)	(21,19) (✓)	(21,19) (✓)	(22,18) (✓)	(19,20) (✓)

For the simulation results for Video-2, the algorithm succeded marking the target trajectory at higher noise levels, compared to Video-1. At noise level $\sigma_N = 7$, the algorithm is able to mark the target trajectory, correctly for all the paths, starting from the locations around the to real trajectory of the target.

Table 3.4 Simulation results for Video-2 with $\delta = 4$

Path Starting Location		**Noise-free**	**$\sigma_N = 1$ $\sigma_N^2 = 1$**	**$\sigma_N = 5$ $\sigma_N^2 = 25$**	**$\sigma_N = 7$ $\sigma_N^2 = 49$**	**$\sigma_N = 10$ $\sigma_N^2 = 100$**	**$\sigma_N = 13$ $\sigma_N^2 = 169$**
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(10,10)	*Best path score*	**708**	**709**	**711**	**712**	723	753
	Displacement error	1	1	1	1	13	14
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(10,7) (✓)
(10,13)	*Best path score*	692	693	693	703	733	771
	Displacement error	4	4	30	30	16	13
		(10,13) (✓)	(10,13) (✓)	(7,18) (✓)	(7,18) (✓)	(9,9) (✓)	(11,14) (✓)
(9,9)	*Best path score*	704	706	**712**	**716**	740	807
	Displacement error	3	3	3	3	16	34
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(9,9) (✓)	(4,10) (✓)
(20,20)	*Best path score*	672	674	691	700	744	741
	Displacement error	96	96	117	114	110	106
		(15,20) (✓)	(15,20) (✓)	(21,19) (✓)	(21,19) (✓)	(21,17) (✓)	(21,16) (✓)
	State Transition	**Metric - 2**					
(10,10)	*Best path score*	**66.375**	**67.875**	**72.125**	78.125	84.500	137.250
	Displacement error	1	1	1	18	5	16
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(14,11) (✓)	(9,14) (✓)	(8,11) (✓)
(10,13)	*Best path score*	50.375	51.750	51.375	59.250	101.500	132
	Displacement error	4	4	5	30	19	13
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(7,18) (✓)	(9,16) (✓)	(10,15) (✓)
(9,9)	*Best path score*	62.375	64.875	**73.500**	82.125	99.375	162.750
	Displacement error	3	3	3	20	16	34
		(10,13) (✓)	(10,13) (✓)	(10,13) (✓)	(14,11) (✓)	(9,9) (✓)	(4,10) (✓)
(20,20)	*Best path score*	32	34.625	54.625	63.500	115.375	112
	Displacement error	96	117	117	114	110	119
		(15,20) (✓)	(25,18) (✓)	(21,19) (✓)	(21,19) (✓)	(21,17) (✓)	(24,18) (✓)
	State Transition	**Metric - 3**					
(10,10)	*Best path score*	**66.375**	**67.875**	32.725	32.574	34.095	34.217
	Displacement error	1	1	21	21	9	21
		(10,13) (✓)	(10,13) (✓)	(15,15) (✓)	(15,15) (✓)	(10,10) (✓)	(15,15) (✓)
(10,13)	*Best path score*	50.375	51.750	34.770	34.270	34.609	33.152
	Displacement error	4	4	14	14	39	9
		(10,13) (✓)	(10,13) (✓)	(10,8) (✓)	(10,8) (✓)	(5,18) (✓)	(10,13) (✓)
(9,9)	*Best path score*	62.375	64.875	37.450	37.207	35.440	37.657
	Displacement error	3	3	36	36	36	35
		(10,13) (✓)	(10,13) (✓)	(9,4) (✓)	(9,4) (✓)	(4,9) (✓)	(4,10) (✓)
(20,20)	*Best path score*	32	34.625	33.475	33.474	36.540	35.150
	Displacement error	96	117	111	111	96	126
		(15,20) (✓)	(25,18) (✓)	(15,25) (✓)	(15,25) (✓)	(20,15) (✓)	(25,20) (✓)

For Video-2, different values for the velocity model parameter is tested. However, even for the modest levels of this parameter, the performance of the algorithm does not yield satisfactory results. In Table 3.4, the results of the simulations of Video-2 with velocity model parameter $\delta = 4$, are presented. For the transition metric m1, the algorithm is able to mark the trajectory of target up to the noise level $\sigma_N = 7$, whereas m2 succeeds to detect target for the noise level $\sigma_N = 5$. However, m3 is only successful for only the noise-free and for $\sigma_N = 1$ cases.

Table 3.5 Simulation results for Video-3 with $\delta = 0$

Path Starting Location		Noise-free	$\sigma_N = 1$ $\sigma_N^2 = 1$	$\sigma_N = 5$ $\sigma_N^2 = 25$	$\sigma_N = 7$ $\sigma_N^2 = 49$	$\sigma_N = 10$ $\sigma_N^2 = 100$	$\sigma_N = 13$ $\sigma_N^2 = 169$
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(10,45)	*Best path score*	**705**	**702**	**707**	717	**741**	750
	Displacement error	0	0	3	6	3	8
		(12,46) (✓)	(12,46) (✓)	(10,47) (✓)	(13,48) (✓)	(12,46) (✓)	(10,48) (✓)
(11,45)	*Best path score*	705	702	707	717	**741**	750
	Displacement error	1	1	4	7	4	8
		(12,46) (✓)	(12,46) (✓)	(10,47) (✓)	(13,48) (✓)	(12,46) (✓)	(10,48) (✓)
(10,43)	*Best path score*	692	691	692	708	725	737
	Displacement error	3	3	6	9	35	12
		(12,46) (✓)	(12,46) (✓)	(10,47) (✓)	(13,48) (✓)	(8,40) (✓)	(10,48) (✓)
(20,20)	*Best path score*	640	647	684	703	704	759
	Displacement error	192	199	214	217	182	207
		(15,20) (✓)	(21,23) (✓)	(21,20) (✓)	(21,17) (✓)	(17,25) (✓)	(19,19) (✓)
	State Transition	**Metric - 2**					
(10,45)	*Best path score*	**65**	**61.125**	**70.125**	75.125	**102**	110.750
	Displacement error	0	0	0	6	4	8
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(12,46) (✓)	(10,48) (✓)
(11,45)	*Best path score*	65	61.125	70.125	75.125	**102**	110.750
	Displacement error	1	1	1	7	5	9
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(12,46) (✓)	(10,48) (✓)
(10,43)	*Best path score*	50.375	49.125	53.500	70	91.625	116.500
	Displacement error	3	3	4	9	35	23
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(8,40) (✓)	(13,42) (✓)
(20,20)	*Best path score*	0	8.500	49.500	64.500	63	127
	Displacement error	192	200	215	220	194	213
		(15,20) (✓)	(21,23) (✓)	(21,20) (✓)	(21,17) (✓)	(18,22) (✓)	(23,20) (✓)
	State Transition	**Metric - 3**					
(10,45)	*Best path score*	**65**	**61.125**	**11.225**	10.028	**8.344**	7.980
	Displacement error	0	0	0	6	4	20
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(12,46) (✓)	(13,42) (✓)
(11,45)	*Best path score*	65	61.125	11.225	10.028	**8.344**	7.980
	Displacement error	1	1	1	7	5	21
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(12,46) (✓)	(13,42) (✓)
(10,43)	*Best path score*	50.375	49.125	10.560	9.923	7.126	8.030
	Displacement error	3	3	4	9	35	23
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(8,40) (✓)	(13,42) (✓)
(20,20)	*Best path score*	0	8.500	8.860	7.426	6.878	7.817
	Displacement error	192	200	215	219	183	213
		(15,20) (✓)	(21,23) (✓)	(21,20) (✓)	(21,17) (✓)	(17,25) (✓)	(23,20) (✓)

In contrast to Video-1 and Video-2, the target trajectory in Video-3 is non-linear (maneuvering). Table 3.5 shows the performance of the proposed algorithm for this maneuvering dim target. In general, the algorithm could mark the true

trajectory up to σ_N = 5 noise level. There are also some cases, where the algorithm determines the final position of the target for the noise level σ_N = 10, correctly, although the displacement error is non-zero.

Table 3.6 Simulation results for Video-3 with δ = 1

Path Starting Location		**Noise-free**	**σ_N = 1 σ_N^2 = 1**	**σ_N = 5 σ_N^2 = 25**	**σ_N = 7 σ_N^2 = 49**	**σ_N = 10 σ_N^2 = 100**	**σ_N = 13 σ_N^2 = 169**
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(10,45)	*Best path score*	**705**	**702**	**707**	717	**738**	752
	Displacement error	0	0	3	6	3	8
		(12,46) (✓)	(12,46) (✓)	(10,47) (✓)	(13,48) (✓)	(12,46) (✓)	(10,48) (✓)
(11,45)	*Best path score*	707	704	709	719	**739**	751
	Displacement error	1	1	4	6	4	9
		(12,46) (✓)	(12,46) (✓)	(10,47) (✓)	(13,48) (✓)	(12,46) (✓)	(10,48) (✓)
(10,43)	*Best path score*	695	694	695	713	727	740
	Displacement error	3	3	6	9	35	12
		(12,46) (✓)	(12,46) (✓)	(10,47) (✓)	(13,48) (✓)	(8,40) (✓)	(10,48) (✓)
(20,20)	*Best path score*	648	651	687	703	709	758
	Displacement error	192	207	215	217	182	207
		(15,20) (✓)	(25,25) (✓)	(21,20) (✓)	(21,17) (✓)	(17,25) (✓)	(19,19) (✓)
	State Transition	**Metric - 2**					
(10,45)	*Best path score*	**65**	**61.125**	**70.125**	77.375	**98.500**	112.750
	Displacement error	0	0	0	5	3	8
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(12,46) (✓)	(10,48) (✓)
(11,45)	*Best path score*	67	63.125	72.125	**79.375**	**99.500**	111.750
	Displacement error	1	1	1	6	4	9
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(12,46) (✓)	(10,48) (✓)
(10,43)	*Best path score*	53.375	52.125	55.875	75	93.625	120.500
	Displacement error	3	3	3	9	35	23
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(8,40) (✓)	(13,42) (✓)
(20,20)	*Best path score*	8	12.500	53.500	64.500	75	126
	Displacement error	192	222	215	220	183	213
		(15,20) (✓)	(25,20) (✓)	(21,20) (✓)	(21,17) (✓)	(17,25) (✓)	(23,20) (✓)
	State Transition	**Metric - 3**					
(10,45)	*Best path score*	**65**	**61.125**	**13.375**	14.679	**12.316**	12.126
	Displacement error	0	0	4	7	4	29
		(12,46) (✓)	(12,46) (✓)	(10,47) (✓)	(13,48) (✓)	(12,46) (✓)	(6,50) (✓)
(11,45)	*Best path score*	67	63.125	15.375	15.013	12.570	14.012
	Displacement error	1	1	5	6	19	29
		(12,46) (✓)	(12,46) (✓)	(10,47) (✓)	(13,48) (✓)	(10,43) (✓)	(6,50) (✓)
(10,43)	*Best path score*	53.375	52.125	13.535	15.638	11.989	12.749
	Displacement error	3	3	3	10	33	12
		(12,46) (✓)	(12,46) (✓)	(12,46) (✓)	(13,48) (✓)	(13,39) (✓)	(10,48) (✓)
(20,20)	*Best path score*	8	12.500	12.860	10.232	12.878	11.911
	Displacement error	192	222	215	183	183	216
		(15,20) (✓)	(25,20) (✓)	(21,20) (✓)	(16,23) (✓)	(17,25) (✓)	(23,18) (✓)

Based on the results in Table 3.6, as in the case of Video-2, where the target is not moving consistently towards a single direction with a constant velocity, utilizing velocity model parameter has no significant contribution for the performance of the proposed algorithm for this maneuvering target by the proposed velocity model in this thesis.

Table 3.7 Simulation results for Video-4 with $\delta = 0$

Path Starting Location		**Noise-free**	**$\sigma_N = 1$ $\sigma_N^2 = 1$**	**$\sigma_N = 5$ $\sigma_N^2 = 25$**	**$\sigma_N = 7$ $\sigma_N^2 = 49$**	**$\sigma_N = 10$ $\sigma_N^2 = 100$**	**$\sigma_N = 13$ $\sigma_N^2 = 169$**
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(10,10)	*Best path score*	**705**	**705**	**709**	702	**724**	761
	Displacement error	0	0	1	2	6	6
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(10,10) (✓)	(8,13) (✓)	(12,11) (✓)
(10,13)	*Best path score*	679	**680**	698	707	**718**	755
	Displacement error	10	8	10	26	18	21
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(8,15) (✓)	(8,13) (✓)	(7,11) (✓)
(9,9)	*Best path score*	705	705	709	702	724	761
	Displacement error	2	2	3	4	8	8
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(10,10) (✓)	(8,13) (✓)	(12,11) (✓)
(20,20)	*Best path score*	640	648	682	710	716	742
	Displacement error	102	104	120	123	109	108
		(15,20) (✓)	(16,21) (✓)	(20,21) (✓)	(21,22) (✓)	(19,18) (✓)	(19,20) (✓)
	State Transition	**Metric - 2**					
(10,10)	*Best path score*	**65**	**66.375**	**63.250**	65.875	**77.875**	125.750
	Displacement error	0	0	1	3	9	6
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(10,10) (✓)	(9,10) (✓)	(12,11) (✓)
(10,13)	*Best path score*	39	41.750	48.875	71.625	**78**	128.125
	Displacement error	10	7	9	26	18	13
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(8,15) (✓)	(8,13) (✓)	(12,11) (✓)
(9,9)	*Best path score*	65	66.375	63.250	65.875	78.125	125.750
	Displacement error	2	2	3	5	5	8
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(10,10) (✓)	(9,10) (✓)	(12,11) (✓)
(20,20)	*Best path score*	0	9.250	44.750	65.375	90.500	109.375
	Displacement error	102	104	132	120	108	105
		(15,20) (✓)	(16,21) (✓)	(22,23) (✓)	(20,21) (✓)	(19,18) (✓)	(17,19) (✓)
	State Transition	**Metric - 3**					
(10,10)	*Best path score*	**65**	**66.375**	**10.230**	7.747	**6.460**	8.062
	Displacement error	0	0	1	7	9	9
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(10,10) (✓)	(9,10) (✓)	(12,11) (✓)
(10,13)	*Best path score*	39	41.750	8.835	8.717	**6.860**	8.223
	Displacement error	10	7	9	26	20	13
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(8,15) (✓)	(8,13) (✓)	(12,11) (✓)
(9,9)	*Best path score*	65	66.375	10.230	8.140	6.114	6.653
	Displacement error	2	2	3	8	5	19
		(10,11) (✓)	(10,11) (✓)	(10,11) (✓)	(10,10) (✓)	(9,10) (✓)	(7,11) (✓)
(20,20)	*Best path score*	0	9.250	8.010	8.176	7.970	7.180
	Displacement error	102	104	132	120	108	128
		(15,20) (✓)	(16,21) (✓)	(22,23) (✓)	(21,22) (✓)	(19,18) (✓)	(20,22) (✓)

In Table 3.7, simulation results for Video-4 are presented. Although the motion of the target in Video-4 is similar to that of in Video-2, in Video-4 the target is

slower than the target in Video-2. The algorithm could only barely mark the true trajectory up noise levels $\sigma_N = 5$, in contrast to the case of Video-2.

Table 3.8 Simulation results for Video-5 with $\delta = 0$

Path Starting Location		**Noise-free**	**$\sigma_N = 1$ $\sigma_N^2 = 1$**	**$\sigma_N = 5$ $\sigma_N^2 = 25$**	**$\sigma_N = 7$ $\sigma_N^2 = 49$**	**$\sigma_N = 10$ $\sigma_N^2 = 100$**	**$\sigma_N = 13$ $\sigma_N^2 = 169$**
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(32,32)	*Best path score*	**705**	**705**	**692**	721	**717**	770
	Displacement error	0	0	3	2	9	11
		(32,32) (✓)	(32,32) (✓)	(33,32) (✓)	(32,32) (✓)	(33,34) (✓)	(30,30) (✓)
(32,33)	*Best path score*	705	705	693	721	**723**	760
	Displacement error	1	1	6	3	11	12
		(32,32) (✓)	(32,32) (✓)	(33,32) (✓)	(32,32) (✓)	(33,34) (✓)	(30,30) (✓)
(20,20)	*Best path score*	640	649	682	702	711	751
	Displacement error	159	136	148	148	138	155
		(15,20) (✓)	(19,22) (✓)	(21,18) (✓)	(21,18) (✓)	(21,19) (✓)	(18,20) (✓)
	State Transition	**Metric - 2**					
(32,32)	*Best path score*	65	64.750	52.750	83	**90.750**	128
	Displacement error	0	0	3	2	9	11
		(32,32) (✓)	(32,32) (✓)	(33,32) (✓)	(32,32) (✓)	(33,34) (✓)	(30,30) (✓)
(32,33)	*Best path score*	65	64.750	53.125	83	93.500	121.125
	Displacement error	1	1	6	3	11	24
		(32,32) (✓)	(32,32) (✓)	(33,32) (✓)	(32,32) (✓)	(33,34) (✓)	(35,36) (✓)
(20,20)	*Best path score*	0	9.125	46.250	70.875	96.875	115.875
	Displacement error	159	126	133	148	138	155
		(15,20) (✓)	(22,23) (✓)	(23,19) (✓)	(21,18) (✓)	(22,18) (✓)	(18,20) (✓)
	State Transition	**Metric - 3**					
(32,32)	*Best path score*	65	64.750	9.170	9.270	**8.134**	7.556
	Displacement error	0	0	3	2	9	11
		(32,32) (✓)	(32,32) (✓)	(33,32) (✓)	(32,32) (✓)	(33,34) (✓)	(30,30) (✓)
(32,33)	*Best path score*	65	64.750	9.205	9.270	8.214	7.826
	Displacement error	1	1	6	3	11	24
		(32,32) (✓)	(32,32) (✓)	(33,32) (✓)	(32,32) (✓)	(33,34) (✓)	(35,36) (✓)
(20,20)	*Best path score*	0	9.125	8.355	8.839	8.034	7.866
	Displacement error	159	126	134	147	138	155
		(15,20) (✓)	(22,23) (✓)	(23,19) (✓)	(21,18) (✓)	(22,18) (✓)	(18,20) (✓)

The simulation results for Video-5 are presented in Table 3.8. It is observed that the performance of the algorithm degraded for the stationary target motion model. The algorithm is only succesful to mark the target trajectory up to noise level $\sigma_N = 5$, although exceptional successful results occurred for the noise level of $\sigma_N = 7$.

Table 3.9 Simulation results for Video-6 with $\delta = 0$

Path Starting Location		Noise-free	$\sigma_N = 1$ $\sigma_N^2 = 1$	$\sigma_N = 5$ $\sigma_N^2 = 25$	$\sigma_N = 7$ $\sigma_N^2 = 49$	$\sigma_N = 10$ $\sigma_N^2 = 100$	$\sigma_N = 13$ $\sigma_N^2 = 169$
	SNR		**22.2 dB**	**8.2 dB**	**5.3 dB**	**2.2 dB**	**0 dB**
	State Transition	**Metric - 1**					
(10,10)	*Best path score*	**705**	**705**	**719**	743	**745**	789
	Displacement error	0	0	5	6	6	4
		(10,11) (✓)	(10,11) (✓)	(9,11) (✗)	(10,10) (✗)	(9,10) (✗)	(10,11) (✓)
(10,8)	*Best path score*	701	702	718	733	**737**	770
	Displacement error	4	4	8	8	19	7
		(10,11) (✓)	(10,11) (✓)	(9,11) (✗)	(10,10) (✗)	(9,10) (✗)	(10,11) (✓)
(10, 13)	*Best path score*	688	690	705	732	735	754
	Displacement error	7	7	11	10	11	17
		(10,11) (✓)	(10,11) (✓)	(9,11) (✗)	(10,10) (✗)	(9,10) (✗)	(10,11) (✓)
(20,20)	*Best path score*	640	648	682	708	739	758
	Displacement error	102	128	110	139	102	135
		(15,20) (✗)	(23,20) (✗)	(21,18) (✗)	(21,24) (✗)	(18,18) (✗)	(23,21) (✗)
	State Transition	**Metric - 2**					
(10,10)	*Best path score*	**25.625**	**27.125**	**48.500**	72.875	**80.625**	130.500
	Displacement error	10	6	9	6	10	4
		(9,12) (✗)	(10,11) (✓)	(10,9) (✗)	(10,10) (✗)	(9,10) (✗)	(10,11) (✓)
(10,8)	*Best path score*	25.625	27.125	48.500	65.500	**82.375**	102.125
	Displacement error	12	8	11	32	13	7
		(9,12) (✗)	(10,11) (✓)	(10,9) (✗)	(9,6) (✗)	(9,10) (✗)	(10,11) (✓)
(10, 13)	*Best path score*	18.250	19.750	35.625	61.750	85.750	105.375
	Displacement error	14	12	18	12	23	28
		(11,12) (✗)	(10,11) (✓)	(11,12) (✗)	(10,10) (✗)	(10,15) (✗)	(7,15) (✗)
(20,20)	*Best path score*	0	8.625	42.125	63.375	106	113.250
	Displacement error	102	134	110	139	102	112
		(15,20) (✗)	(23,20) (✗)	(21,18) (✗)	(21,24) (✗)	(18,18) (✗)	(19,20) (✗)
	State Transition	**Metric - 3**					
(10,10)	*Best path score*	**25.625**	**27.125**	**8.645**	9.477	**6.878**	7.305
	Displacement error	10	6	8	6	16	4
		(9,12) (✗)	(10,11) (✓)	(10,9) (✗)	(10,10) (✗)	(14,12) (✗)	(10,11) (✓)
(10,8)	*Best path score*	25.625	27.125	8.540	8.296	**6.951**	7.137
	Displacement error	12	8	10	8	30	7
		(9,12) (✗)	(10,11) (✓)	(10,9) (✗)	(10,10) (✗)	(9,6) (✗)	(10,11) (✓)
(10, 13)	*Best path score*	18.250	19.750	7.810	9.199	7.861	6.348
	Displacement error	14	12	13	10	24	28
		(11,12) (✗)	(10,11) (✓)	(10,9) (✗)	(10,10) (✗)	(10,15) (✗)	(7,15) (✗)
(20,20)	*Best path score*	0	8.625	7.105	7.431	8.474	9.172
	Displacement error	102	134	110	139	102	115
		(15,20) (✗)	(23,20) (✗)	(21,18) (✗)	(21,24) (✗)	(18,18) (✗)	(19,20) (✗)

The simulation results for Video-6 in Table 3.9 present that the DP-based algorithm exhibit poor performance for the targets, which are slow in speed. For the noise levels above $\sigma_N = 1$, the algorithm could not mark the target trajectory correctly, except for the unreliable results in very low SNR case such as 0dB at the last column of the Table 3.9. Although, the algorithm points the ending location of the target trajectory correctly, the trajectory does not fully matches the true trajectory of the target even for the (10,10) starting point case.

At the table below, the path scores for the path initializing pixels of all the 64x64 pixels of the image, are presented. The scores are calculated according to transition metric m1 and velocity model parameter $\delta = 0$. The algorithm computed on Video-1 with noise level $\sigma_N = 5$.

Table 3.10 The top 100 path scores for Video-1 ($\sigma_N = 5$, $\delta = 0$)

Path Number	Path Score	Path Starting Location	Path End Location	Path
1	706	28 27	27 29	↘ ↑ ↖ → →
2	706	28 28	27 29	↓ ↑ ↖ → →
3	706	28 29	27 29	↙ ↑ ↖ → →
4	706	29 27	27 29	→ ↑ ↖ → →
5	706	29 28	27 29	· ↑ ↖ → →
6	706	29 29	27 29	← ↑ ↖ → →
7	706	30 27	27 29	↗ ↑ ↖ → →
8	706	30 28	27 29	↑ ↑ ↖ → →
9	706	30 29	27 29	↖ ↑ ↖ → →
10	704	9 10	10 15	↘ → → → →
11	704	9 11	10 15	↓ → → → →
12	704	9 12	10 15	↙ → → → →
13	704	10 10	10 15	→ → → → →
14	704	10 11	10 15	· → → → →
15	704	10 12	10 15	← → → → →
16	704	11 10	10 15	↗ → → → →
17	704	11 11	10 15	↑ → → → →
18	704	11 12	10 15	↖ → → → →
19	699	28 26	27 29	↘ ↗ ↖ → →
20	699	29 26	27 29	→ ↗ ↖ → →
21	699	30 26	27 29	↗ ↗ ↖ → →
22	698	27 27	27 29	↘ · ↖ → →
23	698	27 28	27 29	↓ · ↖ → →
24	698	27 29	27 29	↙ · ↖ → →
25	697	47 44	50 43	↓ ↙ ↓ ← →
26	697	47 45	50 43	↙ ↓ ↙ ← →
27	697	48 44	50 43	· ↙ ↓ ← →
28	697	48 45	50 43	← ↙ ↓ ← →
29	697	49 43	50 43	↗ ↙ ↓ ← →
30	697	49 44	50 43	↑ ↙ ↓ ← →
31	697	49 45	50 43	↖ ↙ ↓ ← →
32	696	47 43	50 43	↘ ↙ ↓ ↙ ↗
33	696	48 43	50 43	→ ↙ ↓ ↙ ↗
34	696	49 42	50 43	→ · ↓ ← →
35	696	50 42	50 43	↗ · ↓ ← →
36	696	50 43	50 43	↑ · ↓ ← →
37	696	50 44	50 43	↖ · ↓ ← →
38	695	13 15	15 17	↘ → → ↓ ←
39	695	13 16	15 17	↓ → → ↓ ←
40	695	13 17	15 17	↙ → → ↓ ←
41	695	14 15	15 17	→ → → ↓ ←
42	695	14 16	15 17	· → → ↓ ←
43	695	14 17	15 17	← → → ↓ ←
44	695	15 15	15 17	↗ → → ↓ ←
45	695	15 16	15 17	↑ → → ↓ ←
46	695	15 17	15 17	↖ → → ↓ ←
47	695	48 42	50 43	↘ → ↙ ↙ ↗
48	695	49 46	50 43	← ← ↙ ← →
49	695	50 45	50 43	↑ ← ↙ ← →
50	695	50 46	50 43	↖ ← ↙ ← →
51	694	24 27	27 29	↘ ↙ ↓ → →
52	694	24 28	27 29	↓ ↙ ↓ → →
53	694	24 29	27 29	↙ ↙ ↓ → →
54	694	25 25	27 29	↘ → ↓ → →
55	694	25 26	27 29	↓ → ↓ → →
56	694	25 27	27 29	↙ → ↓ → →
57	694	25 28	27 29	· ↙ ↓ → →
58	694	25 29	27 29	← ↙ ↓ → →
59	694	26 25	27 29	→ → ↓ → →
60	694	26 26	27 29	· → ↓ → →
61	694	26 27	27 29	← → ↓ → →
62	694	26 28	27 29	↑ ↙ ↓ → →
63	694	26 29	27 29	↖ ↙ ↓ → →
64	694	27 25	27 29	↗ → ↓ → →
65	694	27 26	27 29	↑ → ↓ → →
66	694	30 22	29 24	↘ ← ↗ · ↗
67	694	30 23	29 24	↓ ← ↗ · ↗
68	694	30 24	29 24	↙ ← ↗ · ↗
69	694	31 22	29 24	→ ← ↗ · ↗
70	694	31 23	29 24	· ← ↗ · ↗
71	694	31 24	29 24	← ← ↗ · ↗
72	694	32 22	29 24	↗ ← ↗ · ↗
73	694	32 23	29 24	↑ ← ↗ · ↗
74	694	32 24	29 24	↖ ← ↗ · ↗
75	694	46 27	46 28	↘ ↗ · ↙ ↑
76	694	46 28	46 28	↓ ↗ · ↙ ↑
77	694	46 29	46 28	↙ ↗ · ↙ ↑
78	694	47 27	46 28	→ ↗ · ↙ ↑
79	694	47 28	46 28	· ↗ · ↙ ↑
80	694	47 29	46 28	← ↗ · ↙ ↑
81	694	48 27	46 28	↗ ↗ · ↙ ↑
82	694	48 28	46 28	↑ ↗ · ↙ ↑
83	694	48 29	46 28	↖ ↗ · ↙ ↑
84	694	50 26	52 31	↘ → ↘ → →
85	694	50 27	52 31	↓ → ↘ → →
86	694	50 28	52 31	↙ → ↘ → →
87	694	51 26	52 31	→ → ↘ → →
88	694	51 27	52 31	· → ↘ → →
89	694	51 28	52 31	← → ↘ → →
90	694	52 26	52 31	↗ → ↘ → →
91	694	52 27	52 31	↑ → ↘ → →
92	694	52 28	52 31	↖ → ↘ → →
93	693	8 25	7 25	↘ · ↖ ↗ ←
94	693	8 26	7 25	↓ · ↖ ↗ ←
95	693	8 27	7 25	↙ · ↖ ↗ ←
96	693	9 25	7 25	→ · ↖ ↗ ←
97	693	9 26	7 25	· · ↖ ↗ ←
98	693	9 27	7 25	← · ↖ ↗ ←
99	693	10 25	7 25	↗ · ↖ ↗ ←
100	693	10 26	7 25	↑ · ↖ ↗ ←

Table 3.10 represents the highest 100 path scores among 64x64 paths, sorted according to their path scores, which are calculated over Video-1 by using the proposed DP-based algorithm.

The top 9 paths have the maximum path scores; unfortunately the paths, which they indicate, do not match the true target trajectory of the target. The paths between 10 and 18 have true ending locations. The path with number 13 fully matches to the true target trajectory. In order to asses the contribution of velocity model parameter, δ, for Video-1 with noise level σ_N = 5, the experiment is repeated for δ = 15. At the table below, the path scores for the path initializing pixels of all the 64x64 pixels of the Video-1, are presented. The scores are calculated according to transition metric m1 and velocity model parameter δ = 15.

Table 3.11 The top 100 path scores for Video-1 (σ_N = 5, δ = 15)

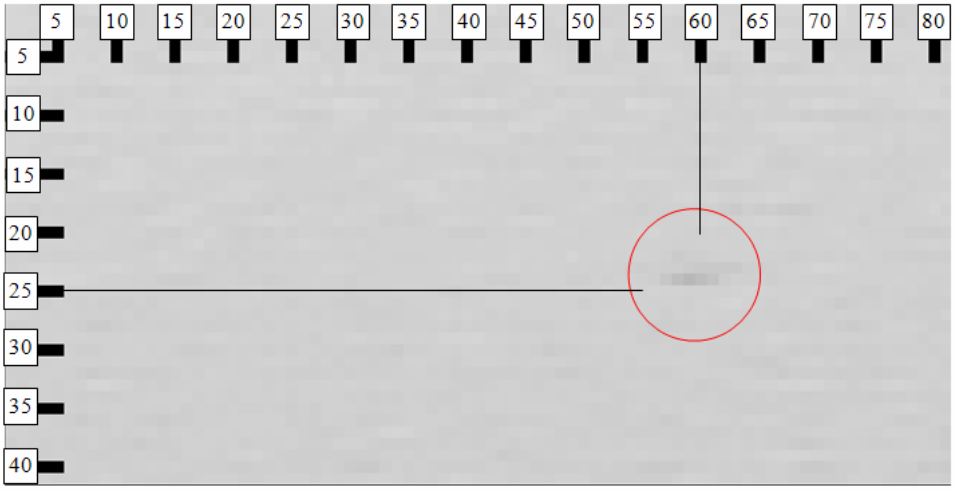
Path Number	Path Score	Path Starting Location	Path End Location	Path
1	824	10 10	10 15	→ → → → →
2	809	9 10	10 15	↘ → → → →
3	809	11 10	10 15	↗ → → → →
4	807	17 15	12 15	↑ ↑ ↑ ↑ ↑
5	806	14 15	14 20	→ → → → →
6	803	10 11	10 11	• • • • •
7	799	27 24	27 29	→ → → → →
8	798	12 30	12 25	← ← ← ← ←
9	798	30 27	30 22	← ← ← ← ←
10	797	31 26	26 26	↑ ↑ ↑ ↑ ↑
11	796	8 10	10 15	↘ ↘ → → →
12	796	14 17	14 12	← ← ← ← ←
13	796	17 9	12 4	↖ ↖ ↖ ↖ ↖
14	795	6 13	6 13	• • • • •
15	794	8 20	13 20	↓ ↓ ↓ ↓ ↓
16	794	9 11	10 15	↓ → → → →
17	794	11 11	6 11	↑ ↑ ↑ ↑ ↑
18	794	30 28	25 25	↑ ↑ ↖ ↖ ↖
19	793	10 9	5 4	↖ ↖ ↖ ↖ ↖
20	793	17 20	17 25	→ → → → →
21	793	30 30	25 25	↖ ↖ ↖ ↖ ↖
22	792	6 17	11 22	↘ ↘ ↘ ↘ ↘
23	792	13 25	13 20	← ← ← ← ←
24	792	14 10	9 15	↗ ↗ ↗ ↗ ↗
25	792	14 26	19 21	↙ ↙ ↙ ↙ ↙
26	792	17 14	12 15	↗ ↑ ↑ ↑ ↑
27	792	19 18	14 18	↑ ↑ ↑ ↑ ↑
28	791	8 25	13 20	↙ ↙ ↙ ↙ ↙
29	791	11 22	16 17	↙ ↙ ↙ ↙ ↙
30	791	13 15	14 20	↘ → → → →
31	791	15 15	14 20	↗ → → → →
32	791	16 13	21 8	↙ ↙ ↙ ↙ ↙
33	791	30 20	25 20	↑ ↑ ↑ ↑ ↑
34	790	10 4	10 4	• • • • •
35	790	14 12	14 12	• • • • •
36	790	17 8	17 8	• • • • •
37	790	17 29	12 29	↑ ↑ ↑ ↑ ↑
38	789	6 14	11 19	↘ ↘ ↘ ↘ ↘
39	789	7 30	7 25	← ← ← ← ←
40	789	11 18	16 13	↙ ↙ ↙ ↙ ↙
41	789	13 17	18 12	↙ ↙ ↙ ↙ ↙
42	789	19 21	14 21	↑ ↑ ↑ ↑ ↑
43	789	22 6	22 11	→ → → → →
44	789	30 24	27 29	↗ ↗ ↗ → →
45	788	9 13	4 18	↗ ↗ ↗ ↗ ↗
46	788	12 10	12 15	→ → → → →
47	788	14 31	14 26	← ← ← ← ←
48	788	27 9	22 9	↑ ↑ ↑ ↑ ↑
49	788	29 21	29 16	← ← ← ← ←
50	788	31 6	26 6	↑ ↑ ↑ ↑ ↑
51	788	32 7	27 7	↑ ↑ ↑ ↑ ↑
52	787	7 26	7 26	• • • • •
53	787	13 20	13 20	• • • • •
54	787	17 5	22 10	↘ ↘ ↘ ↘ ↘
55	787	17 26	22 26	↓ ↓ ↓ ↓ ↓
56	787	18 8	13 8	↑ ↑ ↑ ↑ ↑
57	787	19 24	14 20	↖ ↖ ↖ ↖ ↑
58	787	21 4	16 9	↗ ↗ ↗ ↗ ↗
59	787	22 20	27 20	↓ ↓ ↓ ↓ ↓
60	787	27 23	27 23	• • • • •
61	787	30 19	30 24	→ → → → →
62	787	32 20	27 20	↑ ↑ ↑ ↑ ↑
63	786	5 17	5 17	• • • • •
64	786	6 16	11 16	↓ ↓ ↓ ↓ ↓
65	786	7 9	12 4	↙ ↙ ↙ ↙ ↙
66	786	8 5	5 10	→ → ↗ ↗ ↗
67	786	13 10	10 15	↗ ↗ ↗ → →
68	786	13 19	13 24	→ → → → →
69	786	13 30	13 30	• • • • •
70	786	14 27	9 27	↑ ↑ ↑ ↑ ↑
71	786	17 17	12 15	↖ ↖ ↑ ↑ ↑
72	786	21 13	21 8	← ← ← ← ←
73	785	4 10	4 5	← ← ← ← ←
74	785	5 19	5 19	• • • • •
75	785	5 20	10 15	↙ ↙ ↙ ↙ ↙
76	785	7 17	7 12	← ← ← ← ←
77	785	11 16	16 21	↘ ↘ ↘ ↘ ↘
78	785	17 12	12 12	↑ ↑ ↑ ↑ ↑
79	785	18 22	18 22	• • • • •
80	785	19 26	19 26	• • • • •
81	785	29 29	29 24	← ← ← ← ←
82	784	6 18	11 13	↙ ↙ ↙ ↙ ↙
83	784	6 19	11 19	↓ ↓ ↓ ↓ ↓
84	784	8 4	13 4	↓ ↓ ↓ ↓ ↓
85	784	8 22	13 20	↙ ↙ ↓ ↓ ↓
86	784	8 30	13 25	↙ ↙ ↙ ↙ ↙
87	784	10 26	5 26	↑ ↑ ↑ ↑ ↑
88	784	11 19	11 24	→ → → → →
89	784	11 21	16 21	↓ ↓ ↓ ↓ ↓
90	784	11 26	6 26	↑ ↑ ↑ ↑ ↑
91	784	12 16	7 16	↑ ↑ ↑ ↑ ↑
92	784	14 13	14 8	← ← ← ← ←
93	784	19 31	14 26	↖ ↖ ↖ ↖ ↖
94	784	20 25	20 25	• • • • •
95	784	25 29	25 24	← ← ← ← ←
96	784	26 10	26 5	← ← ← ← ←
97	784	26 25	26 30	→ → → → →
98	784	26 26	26 26	• • • • •
99	784	27 30	27 25	← ← ← ← ←
100	784	28 6	23 6	↑ ↑ ↑ ↑ ↑

Table 3.11 similarly represents the highest 100 path scores among 64x64 paths, which are sorted according to their path scores for Video-1 ($\sigma_N = 5$, $\delta = 15$). The path with highest score at the top of the table fully matches to the true target trajectory. In this experiment, it can be stated that utilization of the velocity model parameter increases the detection performance of the algorithm.

3.3.2 Real Data Set Tests

For the real test set, a helicopter in far distance is recorded by using a day-light sensor. A typical frame from the recorded sequence is shown below.

(a)



(b)

Figure 3-16 (a) Day-light sensor real test data (b) Zoomed target location.

In this video, the target departs from the pixel location (23,58) and reaches to (24,56) at the end of a 5-frame sequence. The mean intensity of the images in the sequence is the gray-level of 200 and the highest intensity of the target pixel is about 185 gray-level, which gives the contrast, as % 5.8. The sensor noise variance is measured as 5.07. A region of interest, whose size is equal to 30x75 pixels, is selected around the target, and dynamic programming method is computed over this region of interest.

At the table below, the path scores for the path initializing pixels, which are located around the neighborhood of the target location, are presented. The scores are calculated according to transition metric m1 and velocity model parameter, $\delta = 0$.

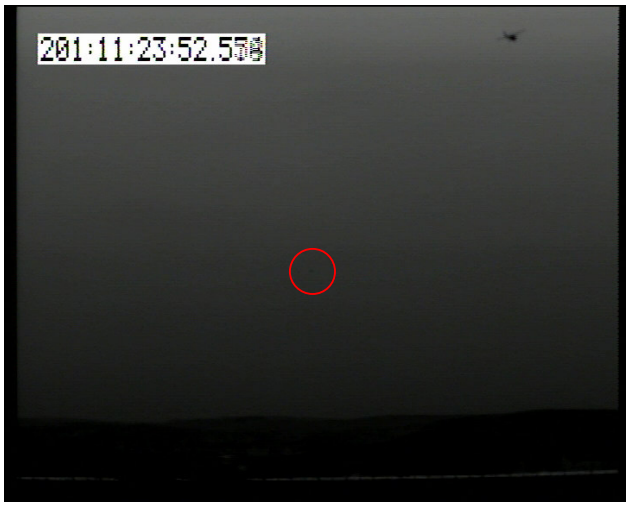
Table 3.12 The path scores for the pixels around the correct target location

Path Number	Path Score	Path Starting Location		Path End Location		Path				
1	354	23	58	24	56	↘	←	←	←	•
2	354	23	59	24	56	↓	←	←	←	•
3	354	23	60	24	56	↙	←	←	←	•
4	354	24	58	24	56	→	←	←	←	•
5	354	24	59	24	56	•	←	←	←	•
6	354	24	60	24	56	←	←	←	←	•
7	354	25	58	24	56	↑	←	←	←	•
8	354	25	59	24	56	↑	←	←	←	•
9	354	25	60	24	56	↖	←	←	←	•
10	352	23	57	24	56	↘	•	←	←	•
11	352	24	57	24	56	→	•	←	←	•
12	352	25	57	24	56	↑	•	←	←	•
13	350	23	56	24	56	↘	→	←	←	•
14	350	24	56	24	56	→	→	←	←	•
15	350	25	56	24	56	↑	→	←	←	•
16	344	22	56	24	56	↘	↘	←	←	•
17	344	22	57	24	56	↓	↘	←	←	•
18	344	22	58	24	56	↙	↘	←	←	•
19	344	22	59	24	56	↙	↓	←	←	•
20	344	22	60	24	56	↙	↙	←	←	•
21	342	23	61	24	56	↙	←	←	←	←
22	342	24	61	24	56	←	←	←	←	←
23	342	25	61	24	56	↖	←	←	←	←
24	337	23	55	24	56	↘	→	•	←	•
25	337	24	55	24	56	→	→	•	←	•
26	337	25	55	24	56	↑	→	•	←	•
27	336	22	55	24	56	↘	↘	•	←	•
28	335	26	56	24	56	↑	↑	←	←	•
29	335	26	57	24	56	↑	↑	←	←	•
30	335	26	58	24	56	↑	↑	←	←	•
31	335	26	59	24	56	↖	↑	←	←	•
32	334	22	61	24	56	↙	↙	←	←	←
33	334	26	60	24	56	↖	↖	←	←	•
34	332	26	55	24	56	↑	↑	•	←	•
35	330	22	54	24	56	↘	↘	→	←	•
36	330	23	54	24	56	↘	→	→	←	•
37	330	24	54	24	56	→	→	→	←	•
38	330	25	54	24	56	↑	→	→	←	•
39	328	26	54	24	56	↑	↑	→	←	•
40	327	26	61	24	56	↖	↖	←	←	←
41	317	21	55	24	56	↘	↘	↓	←	•
42	317	21	56	24	56	↘	↘	↙	←	•
43	315	21	57	24	56	↘	↘	↙	←	←
44	315	21	58	24	56	↓	↘	↙	←	←
45	315	21	59	24	56	↙	↘	↙	←	←
46	315	21	60	24	56	↓	↙	↙	←	←
47	315	21	61	24	56	↙	↙	↙	←	←
48	315	22	53	24	56	↘	↘	→	•	•
49	315	23	53	24	56	→	↘	→	•	•
50	315	24	53	24	56	↑	↘	→	•	•
51	313	25	53	24	56	→	↑	→	•	•
52	313	26	53	24	56	↑	↑	→	•	•
53	312	21	54	24	56	↘	↘	↘	←	•
54	309	27	60	24	56	↑	↖	↖	←	←
55	306	23	62	24	57	↙	←	←	←	←
56	306	24	62	24	57	←	←	←	←	←
57	306	25	62	24	57	↖	←	←	←	←
58	305	21	53	24	56	↘	↘	↘	•	•
59	305	22	52	24	56	↘	↘	→	→	•
60	305	23	52	24	56	↘	→	→	→	•
61	305	24	52	24	56	→	→	→	→	•
62	305	25	52	24	56	↑	→	→	→	•
63	303	22	62	24	57	↙	↙	←	←	←
64	303	26	52	24	56	↑	↑	→	→	•
65	298	21	52	24	56	↘	↘	↘	→	•
66	297	26	62	24	57	↖	↖	←	←	←
67	294	24	51	24	56	↘	↑	→	→	→
68	294	25	51	24	56	→	↑	→	→	→
69	294	26	51	24	56	↑	↑	→	→	→
70	293	22	51	24	56	↘	↘	→	→	→
71	293	23	51	24	56	→	↘	→	→	→
72	292	21	62	24	57	↙	↙	↙	←	←
73	290	20	53	24	56	↘	↙	↘	↘	→
74	290	20	54	24	56	↓	↙	↘	↘	→
75	290	20	55	24	56	↙	↙	↘	↘	→
76	282	23	50	24	55	↘	↑	→	↘	→
77	282	24	50	24	55	→	↑	→	↘	→
78	281	22	50	24	55	↘	→	→	↘	→
79	281	25	50	24	55	↑	↘	↑	→	→
80	272	23	63	24	58	↙	←	↖	↙	←
81	272	24	63	24	58	←	←	↖	↙	←
82	272	25	63	24	58	↖	←	↖	↙	←
83	271	22	63	24	58	↙	↙	↖	↙	←
84	269	21	63	24	58	↙	↙	←	↙	←
85	267	26	63	24	58	↖	↖	↖	↙	←
86	265	27	51	24	54	↑	↑	↑	→	→
87	265	27	52	24	54	↖	↑	↑	→	→
88	263	20	60	24	58	↘	↙	↓	↙	←
89	260	27	73	28	75	↘	•	→	↑	↓
90	260	27	74	28	75	↓	•	→	↑	↓
91	260	27	75	28	75	↙	•	→	↑	↓
92	260	28	73	28	75	→	•	→	↑	↓
93	260	28	74	28	75	•	•	→	↑	↓
94	260	28	75	28	75	←	•	→	↑	↓
95	260	29	73	28	75	↑	•	→	↑	↓
96	260	29	74	28	75	↑	•	→	↑	↓
97	260	29	75	28	75	↖	•	→	↑	↓
98	259	27	72	28	75	↘	→	→	↑	↓
99	259	28	72	28	75	→	→	→	↑	↓
100	259	29	72	28	75	↑	→	→	↑	↓

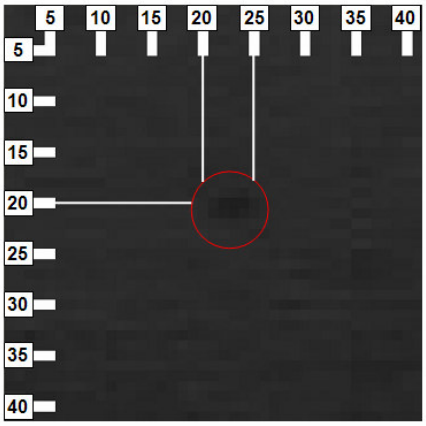
Table 3.12 represents the highest “100” path scores among 30x75 paths, sorted according to their path scores, which are calculated over the given real image sequence by using the proposed DP-based algorithm.

In this table, the first 9 paths have the path score of 354, and all of them are matching paths, which match to the locations in the vicinity of the real target trajectory. If the table is examined in detail, it can be observed that the top 88 paths have the true ending location. The number of candidate trajectories can be eliminated by proper thresholding of these path scores, or implementation of an association algorithm, or by applying dead zones around the most probable trajectory starting point.

For a second real test set, a helicopter in far distance is recorded by using an IR sensor. A typical frame from the recorded sequence is shown below.



(a)



(b)

Figure 3-17 (a) IR sensor real test data (b) Zoomed target location.

In this video, the target is stationary at pixel location (20,23) along the 5-frame sequence. The mean intensity of the images in the sequence is the gray-level of 40 and the lowest intensity of the target pixel is about 31 gray-level, which gives the contrast, as % 3.5. The sensor noise variance is measured as 3.84. A region of interest, whose size is equal to 30x30 pixels, is selected around the target, and dynamic programming method is computed over this region of interest.

Table 3.13 The path scores for the pixels around the correct target location

Path Number	Path Score	Path Starting Location	Path End Location	Path
1	1127	19 22	20 23	↘ ↓ • ↑ •
2	1127	19 23	20 23	↓ ↓ • ↑ •
3	1127	19 24	20 23	↙ ↓ • ↑ •
4	1127	20 22	20 23	→ ↓ • ↑ •
5	1127	20 23	20 23	• ↓ • ↑ •
6	1127	20 24	20 23	← ↓ • ↑ •
7	1127	21 22	20 23	↗ ↓ • ↑ •
8	1127	21 23	20 23	↑ ↓ • ↑ •
9	1127	21 24	20 23	↖ ↓ • ↑ •
10	1126	19 21	20 23	↘ ↘ • ↑ •
11	1126	20 21	20 23	↘ → • ↑ •
12	1126	21 21	20 23	→ → • ↑ •
13	1126	22 21	20 23	↗ → • ↑ •
14	1126	22 22	20 23	↑ → • ↑ •
15	1126	22 23	20 23	↖ → • ↑ •
16	1126	22 24	20 23	↖ • • ↑ •
17	1125	19 25	20 23	↙ ↙ • ↑ •
18	1125	20 25	20 23	← ↙ • ↑ •
19	1125	21 25	20 23	↖ ↙ • ↑ •
20	1125	22 25	20 23	↖ ← • ↑ •
21	1123	23 22	20 23	↗ ↑ • ↑ •
22	1123	23 23	20 23	↑ ↑ • ↑ •
23	1123	23 24	20 23	↖ ↑ • ↑ •
24	1122	23 21	20 23	↗ ↗ • ↑ •
25	1122	23 25	20 23	↖ ↖ • ↑ •
26	1121	19 20	20 23	↘ ↘ → ↑ •
27	1121	20 20	20 23	↘ → → ↑ •
28	1121	21 20	20 23	→ → → ↑ •
29	1121	22 20	20 23	↗ → → ↑ •
30	1120	19 26	20 23	↙ ↙ ← ↑ •
31	1120	20 26	20 23	← ↙ ← ↑ •
32	1120	21 26	20 23	↖ ↙ ← ↑ •
33	1119	22 26	20 23	↖ ← ← ↑ •
34	1118	23 20	20 23	↗ ↗ → ↑ •
35	1117	23 26	20 23	↖ ↖ ← ↑ •
36	1113	20 19	20 23	↘ → → ↗ •
37	1113	21 19	20 23	→ → → ↗ •
38	1113	22 19	20 23	↗ → → ↗ •
39	1111	18 20	20 23	↘ ↘ ↘ ↑ •
40	1111	18 21	20 23	↘ ↓ ↘ ↑ •
41	1111	18 22	20 23	↓ ↓ ↘ ↑ •
42	1111	18 23	20 23	↙ ↓ ↘ ↑ •
43	1111	19 19	20 23	↘ ↘ → ↗ •
44	1110	18 24	20 23	↙ ↓ ↓ ↑ •
45	1110	18 25	20 23	↙ ↙ ↓ ↑ •
46	1109	17 21	20 23	↘ ↘ ↓ • •
47	1109	17 22	20 23	↓ ↘ ↓ • •
48	1109	17 23	20 23	↙ ↘ ↓ • •
49	1108	17 24	20 23	↙ ↓ ↓ • •
50	1108	17 25	20 23	↙ ↙ ↓ • •
51	1108	18 19	20 23	↘ ↘ ↘ ↗ •
52	1108	23 19	20 23	↗ ↗ → ↗ •
53	1107	17 26	20 23	↙ ↙ ↙ • •
54	1107	18 26	20 23	← ↙ ↙ • •
55	1104	20 27	20 23	← ↙ ↖ ← •
56	1104	21 27	20 23	↖ ↙ ↖ ← •
57	1104	22 27	20 23	↖ ← ↖ ← •
58	1103	23 27	20 23	↖ ↖ ↖ ← •
59	1098	16 21	20 23	↙ ↘ ↘ ↘ •
60	1097	20 18	20 23	↘ ↗ ↘ ↗ →
61	1097	21 18	20 23	→ ↗ ↘ ↗ →
62	1095	20 28	20 23	← ↙ ↖ ← ←
63	1095	21 28	20 23	↖ ↙ ↖ ← ←
64	1095	24 21	20 23	↖ → ↗ ↗ ↑
65	1094	17 18	20 23	↘ → ↘ ↘ →
66	1094	17 19	20 23	↓ → ↘ ↘ →
67	1094	17 20	20 23	↙ → ↘ ↘ →
68	1094	18 18	20 23	→ → ↘ ↘ →
69	1094	19 18	20 23	↗ → ↘ ↘ →
70	1094	22 28	20 23	↖ ← ↖ ← ←
71	1093	16 18	20 23	↘ ↘ ↘ ↘ →
72	1093	16 19	20 23	↓ ↘ ↘ ↘ →
73	1093	16 20	20 23	↙ ↘ ↘ ↘ →
74	1093	24 23	20 23	• ↑ ↑ ↑ ↑
75	1093	25 23	20 23	↑ ↑ ↑ ↑ ↑
76	1090	22 18	20 23	↘ → ↗ ↗ ↗
77	1090	23 18	20 23	→ → ↗ ↗ ↗
78	1090	24 18	20 23	↗ → ↗ ↗ ↗
79	1090	24 19	20 23	↑ → ↗ ↗ ↗
80	1090	24 20	20 23	↖ → ↗ ↗ ↗
81	1085	26 18	28 21	↘ ↘ ↑ • ↘
82	1085	26 19	28 21	↓ ↘ ↑ • ↘
83	1085	26 20	28 21	↙ ↘ ↑ • ↘
84	1085	27 18	28 21	→ ↘ ↑ • ↘
85	1085	27 19	28 21	• ↘ ↑ • ↘
86	1085	27 20	28 21	← ↘ ↑ • ↘
87	1085	28 18	28 21	↗ ↘ ↑ • ↘
88	1085	28 19	28 21	↑ ↘ ↑ • ↘
89	1085	28 20	28 21	↖ ↘ ↑ • ↘
90	1084	24 22	28 21	↙ ↙ ↓ • ↘
91	1084	25 20	28 21	→ ↙ ↓ • ↘
92	1084	25 21	28 21	• ↙ ↓ • ↘
93	1084	25 22	28 21	← ↙ ↓ • ↘
94	1084	26 21	28 21	↙ ↓ ↑ • ↘
95	1084	26 22	28 21	↙ ↙ ↑ • ↘
96	1084	27 21	28 21	← ↓ ↑ • ↘
97	1084	27 22	28 21	← ↙ ↑ • ↘
98	1084	28 21	28 21	↖ ↓ ↑ • ↘
99	1084	28 22	28 21	↖ ↙ ↑ • ↘
100	1084	29 18	28 21	↗ → ↑ • ↘

Table 3.13 represents the highest 100 path scores among 30x30 paths, sorted according to their path scores, which are calculated over the given real image sequence by using the proposed DP-based algorithm.

In this table, the first 9 paths have the path score of 1127, and all of them are matching paths, which match to the locations in the vicinity of the real target trajectory. If the table is examined in detail, it can be observed that the top 80 paths have the true ending location. The number of candidate trajectories can also be eliminated by proper thresholding of these path scores, or implementation of an association algorithm, or by applying dead zones around the most probable trajectory starting point.

3.4 Discussion

The simulations in this chapter are grouped according to utilized the parameters

- Target trajectory and speed,
    - Video-1: Linear trajectory to East with a speed of 1 pixel/frame.
    - Video-2: Linear trajectory to East with average speed of 0.5 pixel/fr.
    - Video-3 : Circular and continuous trajectory
    - Video-4 : Linear trajectory to East with average speed of 0.2 pixel/fr.
    - Video-5 : Stationary target trajectory at location (32,32)
    - Video-6 : Linear trajectory of 3x3 size target to East with a speed of 1 pixel/frame
- State Transition Metrics,
    - Metric-1 : Absolute intensity metric : $m1 = I_b$,
    - Metric-2 : Contrast-based metric : $m2 = I_b - avg(I_b)$,

      Metric-3 : Normalized contrast : $m3 = (I_b - avg(I_b)) / \sigma_N$
- δ (Velocity Model).
    - 0
    - 1
    - 4
    - 15.

The results over the artificial sequences showed that the DP-based algorithm is successful up to noise levels of $\sigma_N = 5$ for the examined trajectories, without utilizing any velocity model parameter. Proper utilization of velocity model parameter increases the performance of the algorithm as in the case of Video-1. Any significant performance difference has not been declared between 3 different state transition metrics *m*1, *m*2, *m*3. However, *m*3 is preferable due its normalization property.

On the other hand, two different real data sets are tested by utilizing the proposed algorithm. This algorithm has marked the true target trajectory at both day-light and IR sensor cases.

The algorithm is successfully performed on real data sets, since the observed noise level of these sequences are within the noise limits of the proposed algorithm.

CHAPTER 4

DIM TARGET DETECTION VIA BAYESIAN FORMULATION

In any type of detection problem, Bayesian formulation simply aims modeling the class-conditional probability density functions of the "target" and "no-target" classes, by incorporating the measurements and the available a priori information for the target and background models into these functions. In dim target detection problem, the likelihood of these class-conditional densities should yield, not only the existence of a target, but also give an estimate for the trajectory of the target.

The algorithm in [17] detects the presence of a target and estimates its position, simultaneously. There is no need to store all the history of measurements, since the algorithm uses a recursive method to detect a target and its positional estimate. The technique neither assumes constant velocity motion nor requires a filter-bank implementation structure. The algorithm is computationally quite efficient and suitable for any real-time operation. In this chapter, the effectiveness of the proposed algorithm is tested over sequences of synthetic and real data, and it will be presented to be capable of detecting targets in quite low-SNR conditions.

4.1 Algorithm

This algorithm can be divided into two stages as detection and tracking.
The detection stage is based on a hypothesis testing problem, which can be defined as,

$$H_0 : \mathbf{z}(k) = \mathbf{n}(k)$$
$$H_1 : \mathbf{z}(k) = A(\mathbf{x}(k)) + \mathbf{n}(k) \qquad (4.1)$$

where $\mathbf{z}(k)$ is the matrix consisting of the image frame at time k, $\mathbf{x}(k)$ is the state vector at time k, $\mathbf{n}(k)$ is the measurement noise and $A(.)$ is the target signal component inside noisy measurement. In this formulation, H_0 is the hypothesis that target does not exit, whereas H_1 denotes the existence of a target in the observed scenery. Assuming that the sensor noise is Gaussian distributed with variance σ^2, the class-conditional probability density functions for those two hypotheses can be written as

$$p(\mathbf{z}(k) \mid H_0) = \prod_{i,j} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{z_{i,j}^2(k)}{2\sigma^2}} \qquad (4.2)$$

$$p(\mathbf{z}(k) \mid \mathbf{x}(k), H_1) = \prod_{i,j} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(z_{i,j}(k) - A_{i,j}(\mathbf{x}(k)))^2}{2\sigma^2}} \qquad (4.3)$$

In the above relation, $A_{i,j}$ is the intensity of the target at location (i,j). The ratio of these two densities gives a likelihood value, indicating the existence of a target in the observed scenery. For every measurement, the following likelihood should be computed, in order to be compared to a threshold to give a decision of detection. It should be noted that Z^k denotes the set of all observations up to time k in (4.4).

$$\Lambda(k) = \frac{p(Z^k \mid H_1)}{p(Z^k \mid H_0)} \qquad (4.4)$$

If the likelihood is larger than a pre-defined threshold than one should declare detection of a target.

$$\Lambda(k) \underset{H_0}{\overset{H_1}{\gtrless}} \delta \tag{4.5}$$

In order to obtain some computational simplification of (4.4), the recursive computation of this likelihood is introduced [17], after a number of simplifications

$$\Lambda(k) = K \ \Lambda(k-1) \sum_{\mathbf{x}(k)\in X} \prod_{i,j} e^{\frac{z_{i,j} A_{i,j}(\mathbf{x}(k))}{\sigma^2}} p(\mathbf{x}(k) \mid Z^{k-1}, H_1) \tag{4.6}$$

where the normalization constant K is equal to

$$K = \prod_{i,j} e^{-\frac{A_{i,j}^2(\mathbf{x})}{2\sigma^2}} \tag{4.7}$$

In tracking stage, the posterior state probabilities $p(\mathbf{x}(k) \mid Z^k, H_1)$ are calculated, recursively and the expected value of the target state estimate is determined as,

$$\hat{\mathbf{X}}(k) = \sum_{\mathbf{x}(k)\in X} \mathbf{x}(k)\, p\left(\mathbf{x}(k) \mid \mathbf{Z}^k\right) \tag{4.8}$$

In the equation above, the posterior state density can be calculated by using the following equation, given in (4.9)[17].

$$p\left(\mathbf{x}(k) \mid \mathbf{Z}^{k}\right) = \frac{p\left(\mathbf{x}(k) \mid \mathbf{Z}^{k-1}\right) e^{\frac{1}{\sigma^2}\sum_{i,j} z_{i,j}(k)\, A_{i,j}(\mathbf{x}(k))}}{\sum_{\mathbf{x}(k)} p\left(\mathbf{x}(k) \mid \mathbf{Z}^{k-1}\right) e^{\frac{1}{\sigma^2}\sum_{i,j} z_{i,j}(k)\, A_{i,j}(\mathbf{x}(k))}} \tag{4.9}$$

The prediction equation is given below:

$$p\left(\mathbf{x}(k+1) \mid \mathbf{Z}^{k}\right) = \sum_{\mathbf{x}(k-1) \in X} p(\mathbf{x}(k+1) \mid \mathbf{x}(k))\, p\left(\mathbf{x}(k) \mid \mathbf{Z}^{k}\right) \tag{4.10}$$

For incorporating the a priori target model, since the target motion is unknown, the random-walk state transition model should be selected. According to this model state transition is defined as

$$p(\mathbf{x}(k) \mid \mathbf{x}(k-1)) = \begin{cases} 1/9 & \| \mathbf{x}(k) - \mathbf{x}(k-1) \| \leq 1 \\ 0 & otherwise \end{cases} \tag{4.11}$$

However, in this model, it is assumed that the target can not move more than 1 pixel/frame. With point target signature assumption, likelihood computation is reduces to

$$\Lambda(k) = K \;\; \Lambda(k-1) \sum_{\mathbf{x}(k) \in X} e^{\frac{z_{x,y}}{\sigma^2}}\, p(\mathbf{x}(k) \mid Z^{k-1}, H_1) \tag{4.12}$$

This algorithm can be summarized with the following pseudo-code;

```
Initialization:
      Λ(0) = 1, p(x(1)| Z^0) = 1/M for all x,y
For k = 1..K
   1. Retrieve z(k)
   2. Calculate Λ(k)
   3. Calculate p(x(k)| Z^k) (estimation matrix)
   4. Calculate p(x(k+1)| Z^k) (prediction matrix)
   5. If Λ(k) > δ declare detection and calculate x_ML(k)
```

In this algorithm, M is the number of pixels in each frame and K is the total number of frames in sequence and steps between 1 and 5 are repeated in a recursive manner.

In the next section, simulations are conducted in order to asses the performance of the algorithm, explained in this section.

4.2 Simulations

In this section, the proposed Bayesian formulated detection algorithm is tested by using artificial and real data sets. The artificial simulations are conducted for different noise levels and frame length. For real data sets, the image sequences of a dim target observed by a day-light sensor and another sequence of an IR sensor are used, whose sample screen shots were given in Section 3.3.2.

4.2.1 Artificial Test Data Set

In order to test this algorithm, artificial image sequences, given in Section 3.3.1.1 are utilized in order to be able perform fair comparisons between these two algorithms.

The simulation results are presented by the help of some figures (Figure 4-1 - Figure 4-60). These figures are organized in such a way that, the plot of the likelihood (in log-scale) for the case, in which the target exists in the image sequence (subfigures (a)), the resulting posterior state density of image sequence at the end of iterations (subfigures (b)), the expected value of target state (trajectory) estimate ("o"), and the true target trajectory ("Δ"), as groundtruth (subfigures (c)), a typical likelihood figure (in log-scale) for the case, in which there is no target in the image sequence for the same noise level (subfigures (d)) and, the resulting posterior state density at the end of iterations for no-target case (subfigures (e)) and, finally, the most probable target state ("x"), and the true target trajectory ("Δ"), as groundtruth (subfigures (f)) are presented, respectively.

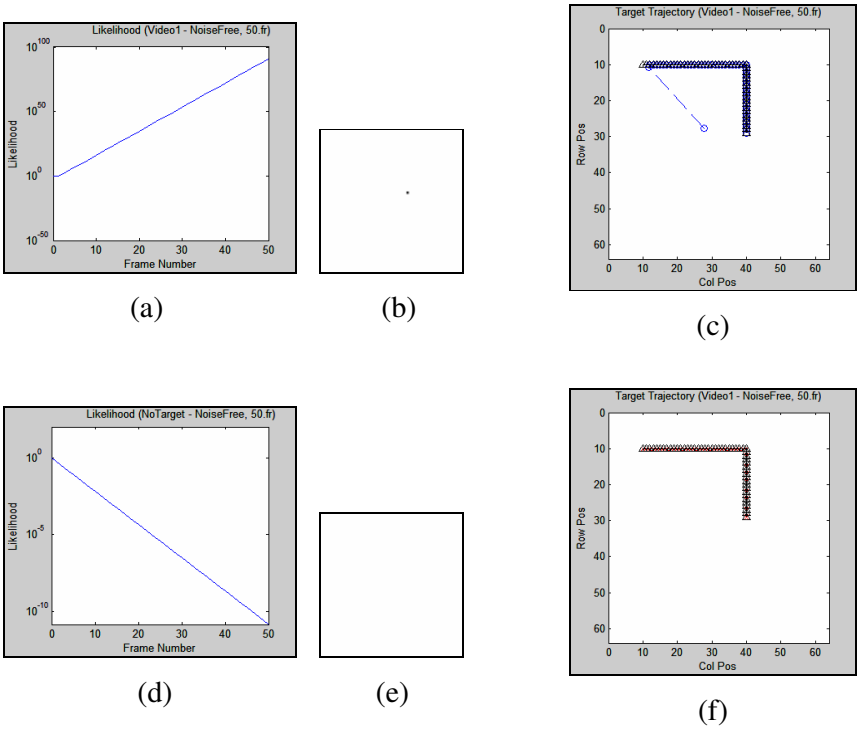


Figure 4-1 Simulation Result for Video-1 (Noise-free, 50 frames)

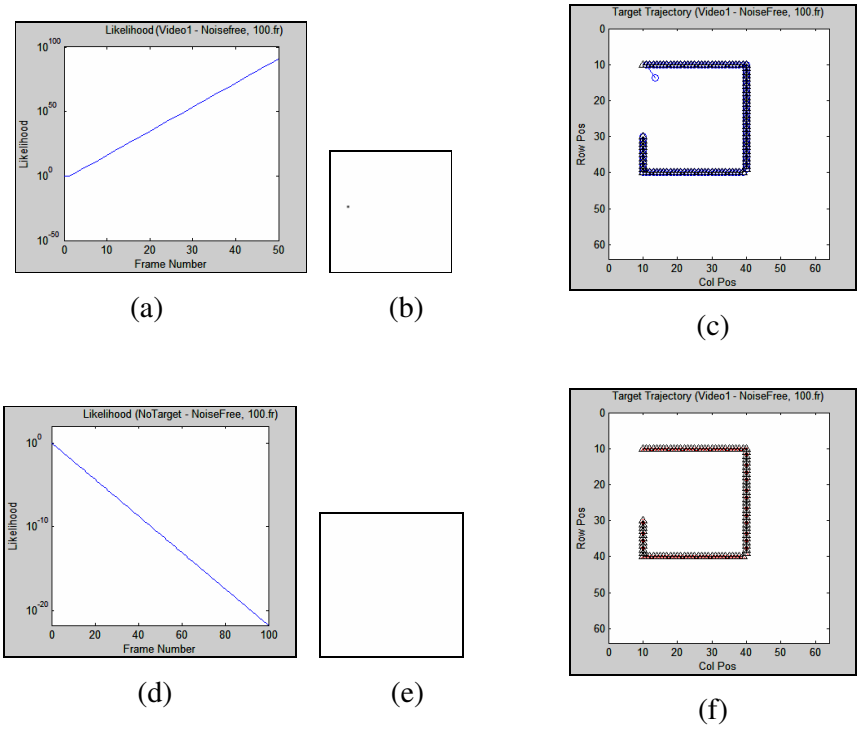
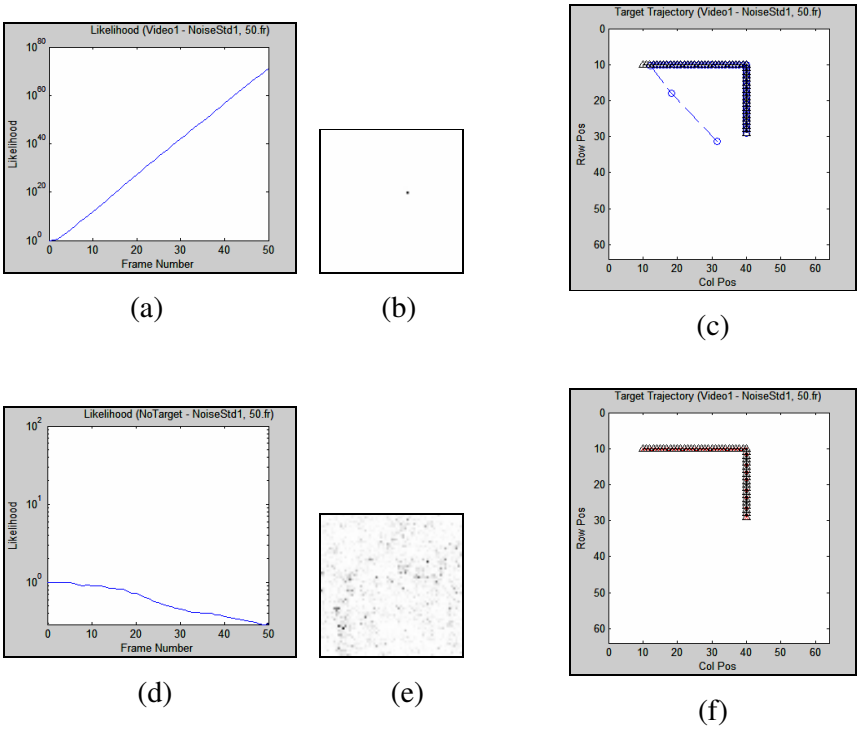


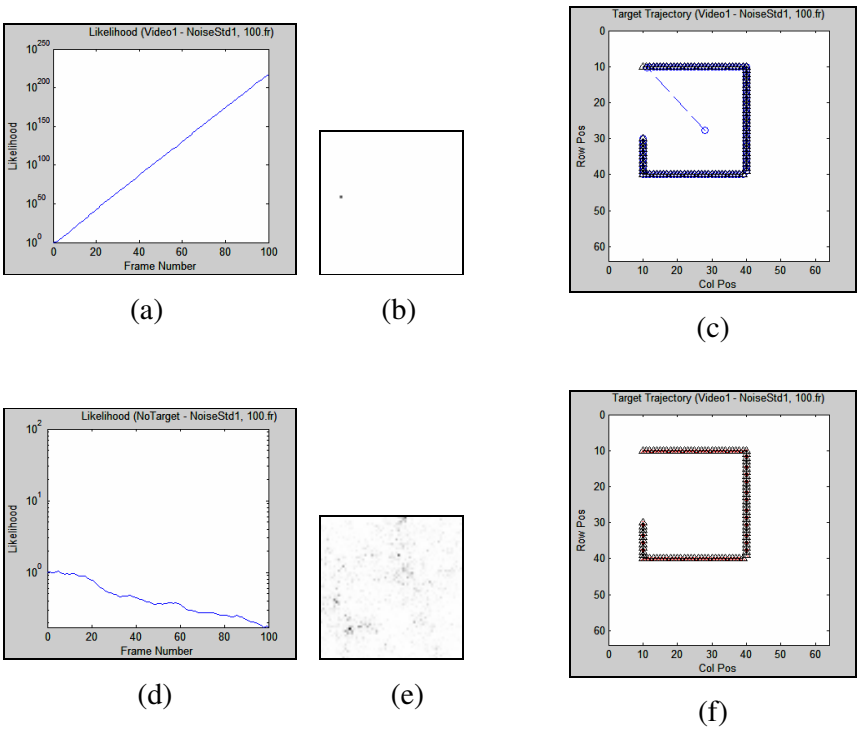
Figure 4-2 Simulation Result for Video-1 (Noise-free, 100 frames)



(a) (b) (c)

(d) (e) (f)

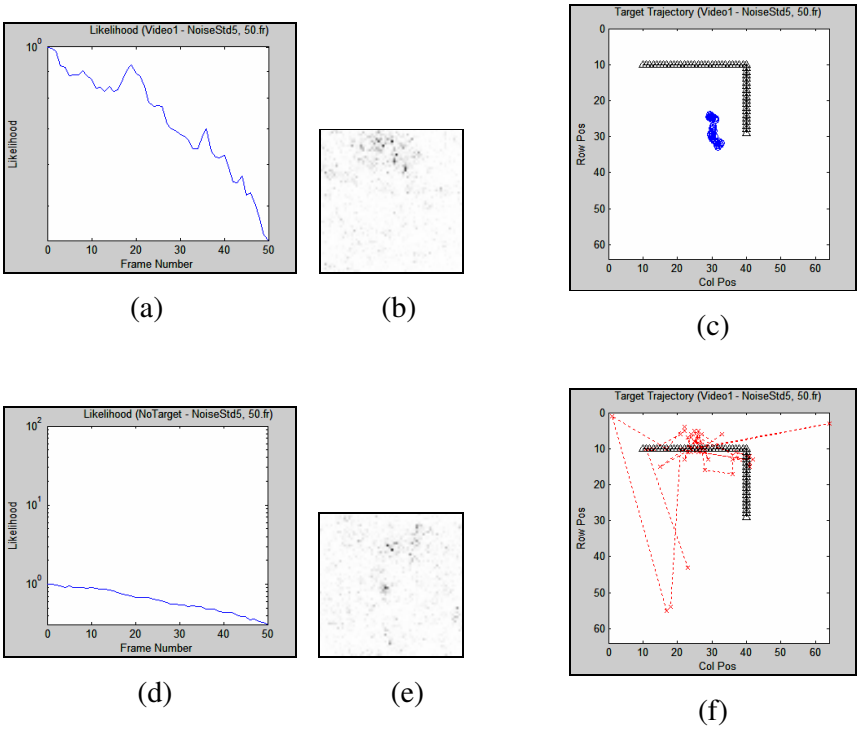
Figure 4-3 Simulation Result for Video-1 (σ_N = 1, 50 frames)



(a) (b) (c)

(d) (e) (f)

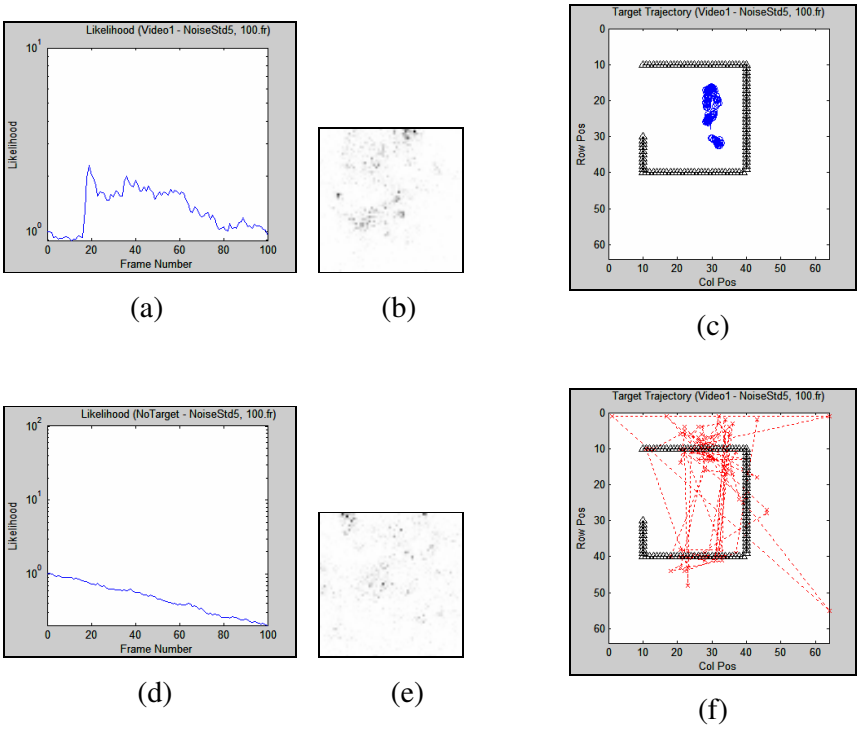
Figure 4-4 Simulation Result for Video-1 (σ_N = 1, 100 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-5 Simulation Result for Video-1 (σ_N = 5, 50 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-6 Simulation Result for Video-1 (σ_N = 5, 100 frames)

The figures between Figure 4-1 and Figure 4-6 presents the simulation results of proposed algorithm for Video-1 at different noise levels and for 50- and 100-frame length sequences.

The algorithm is found out to be successful only for the noise-free case and for noise level $\sigma_N = 1$. The reason for this failure is due to fact that the target velocity is faster than the algorithm could handle. Moreover, the target model is assumed to be Gaussian distributed and this assumption is violated by using a point-target in the simulations.

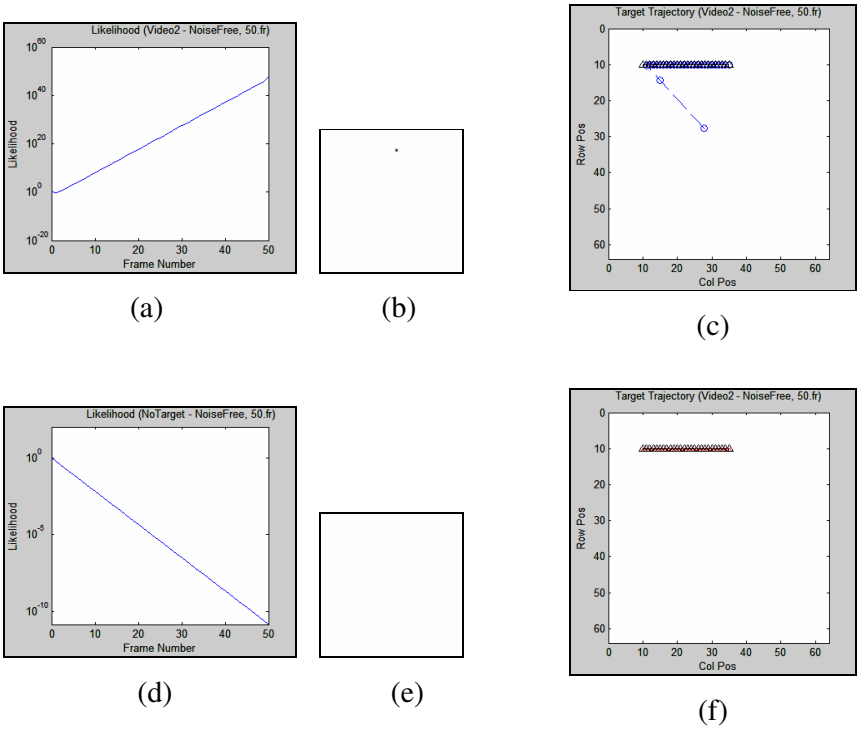


Figure 4-7 Simulation Result for Video-2 (Noise-free, 50 frames)

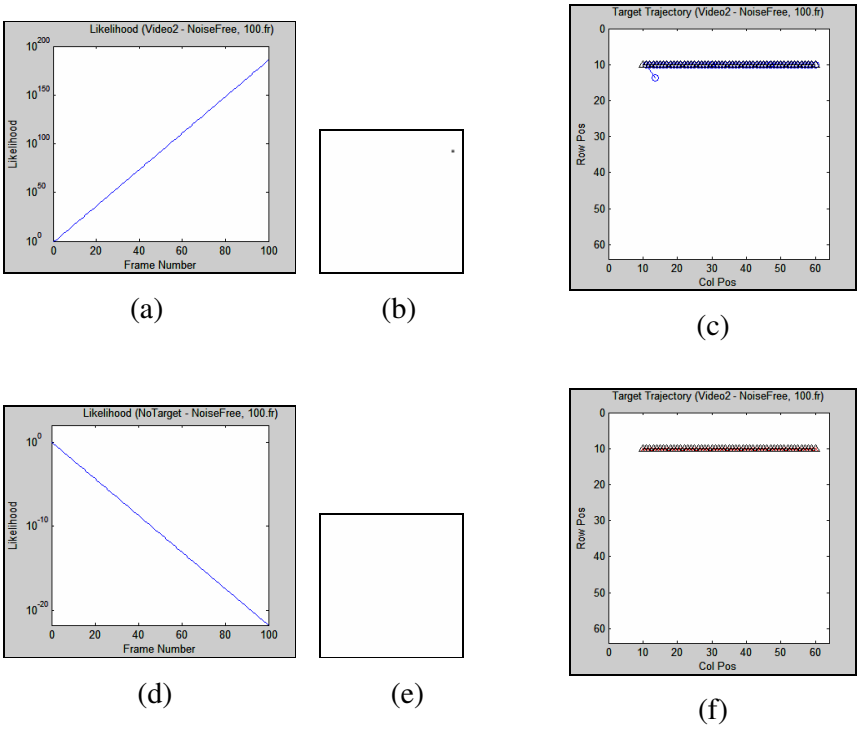


Figure 4-8 Simulation Result for Video-2 (Noise-free,100 frames)

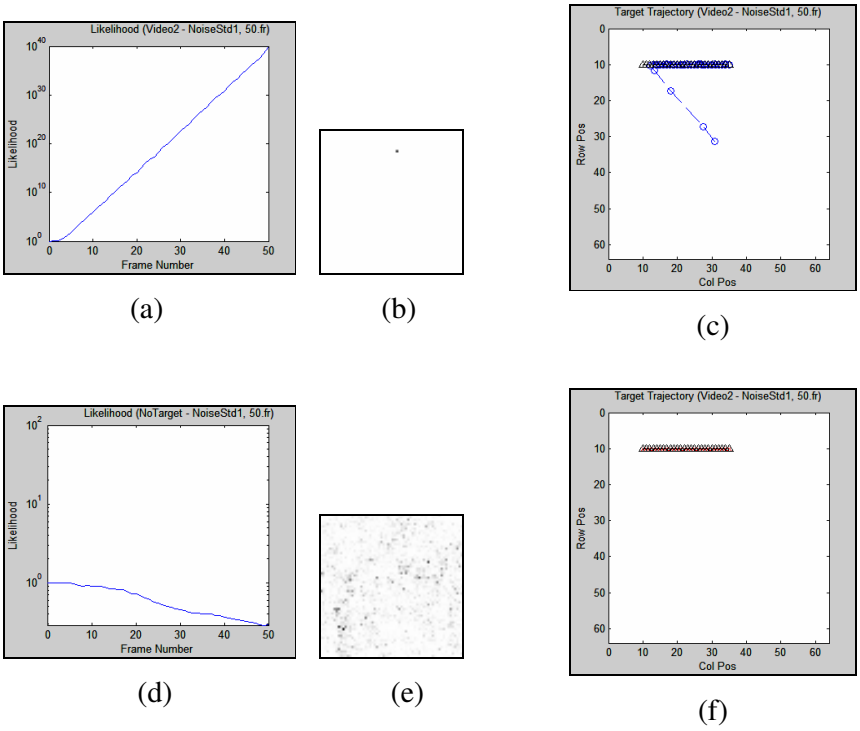


Figure 4-9 Simulation Result for Video-2 (σ_N = 1, 50 frames)

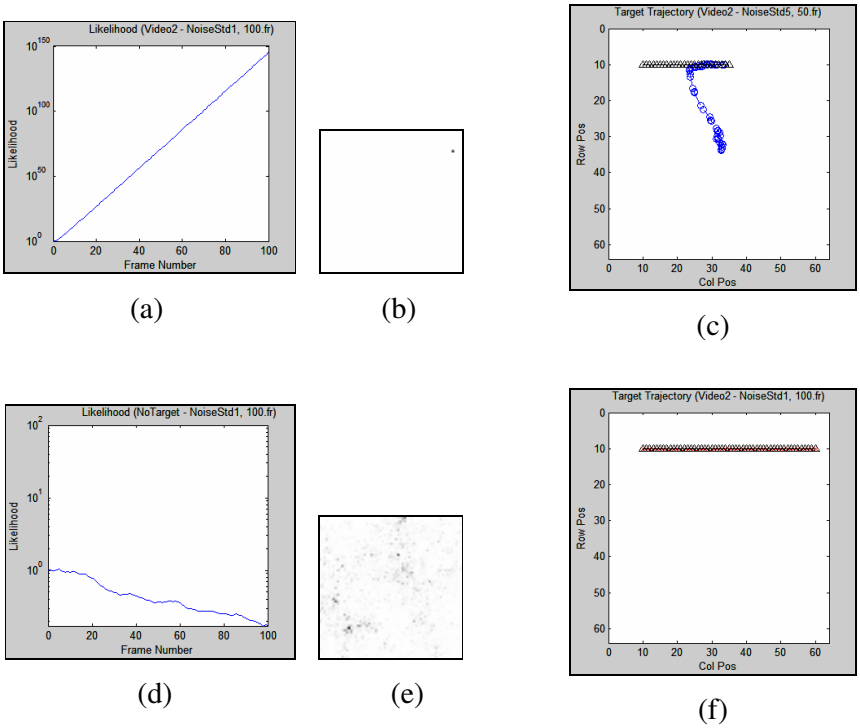


Figure 4-10 Simulation Result for Video-2 (σ_N = 1, 100 frames)

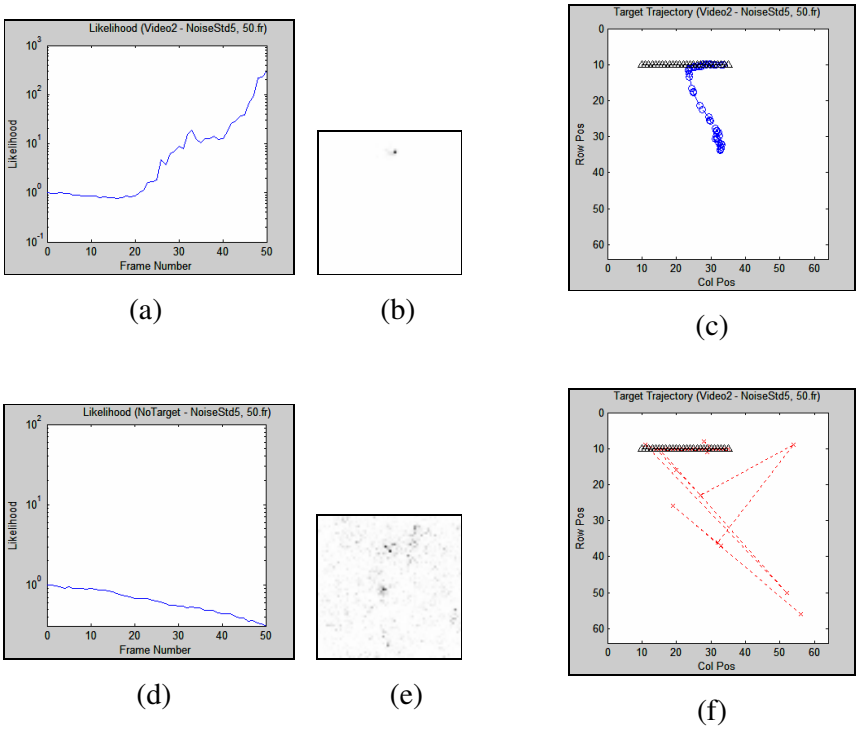


Figure 4-11 Simulation Result for Video-2 (σ_N = 5, 50 frames)

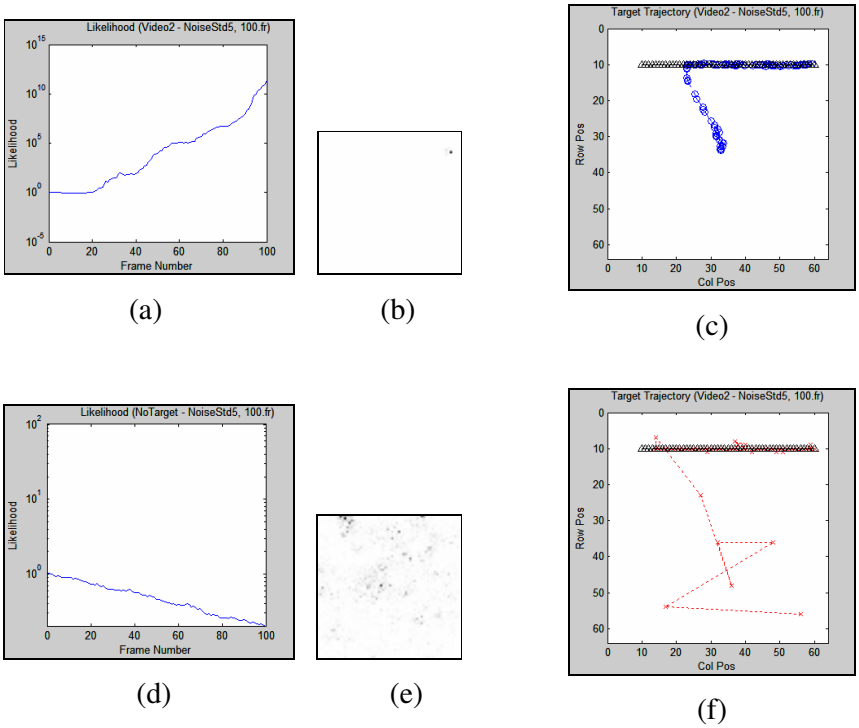


Figure 4-12 Simulation Result for Video-2 (σ_N = 5, 100 frames)

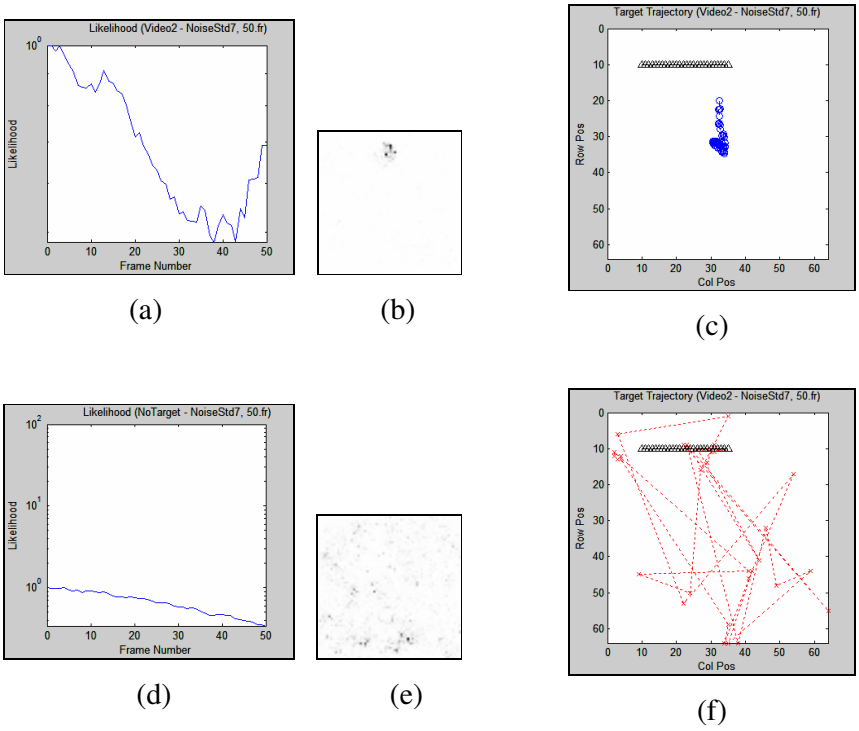


Figure 4-13 Simulation Result for Video-2 (σ_N = 7, 50 frames)

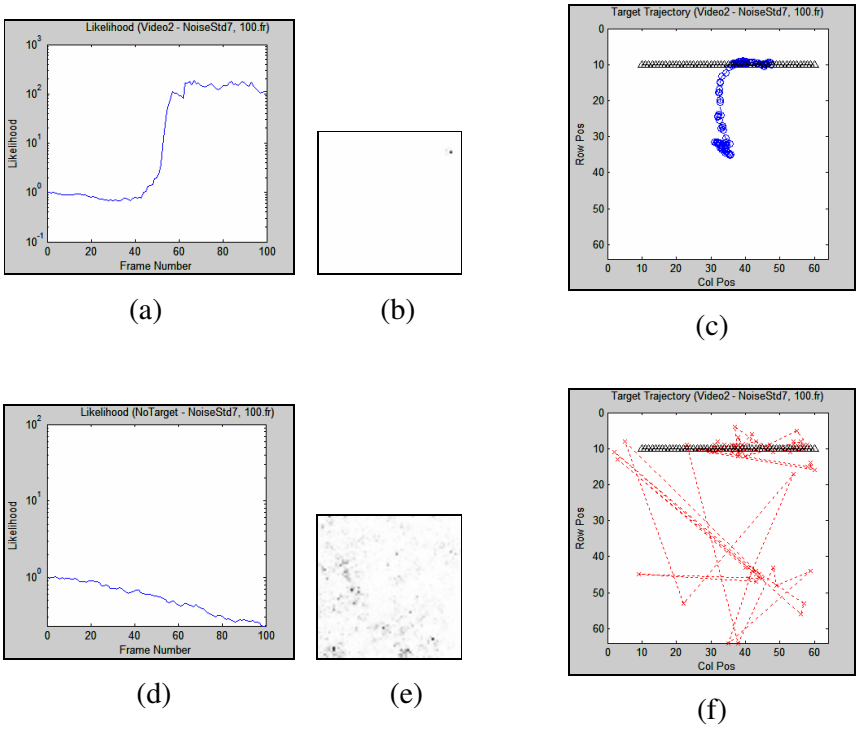
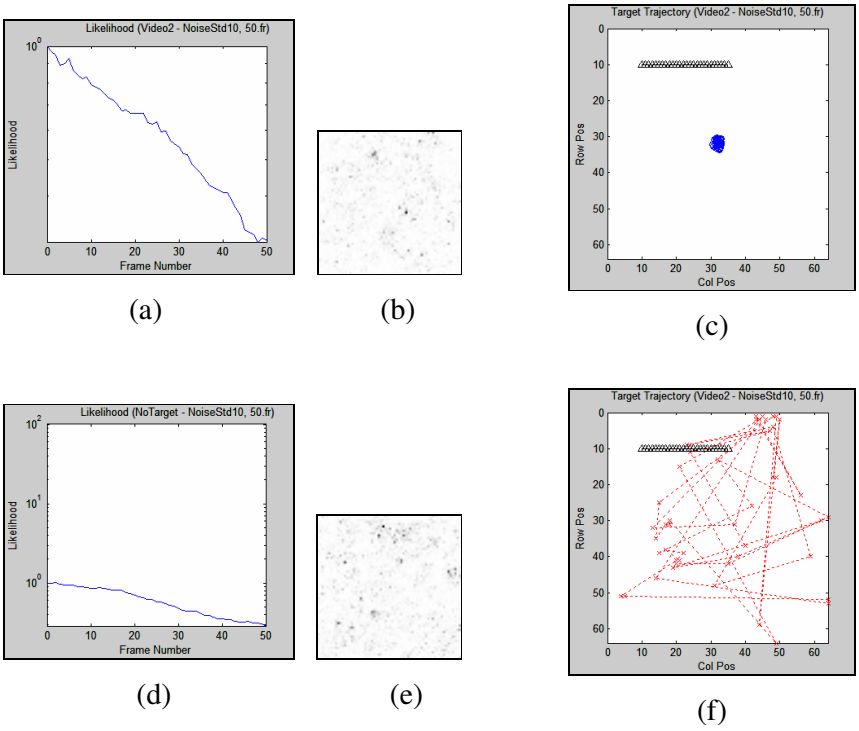


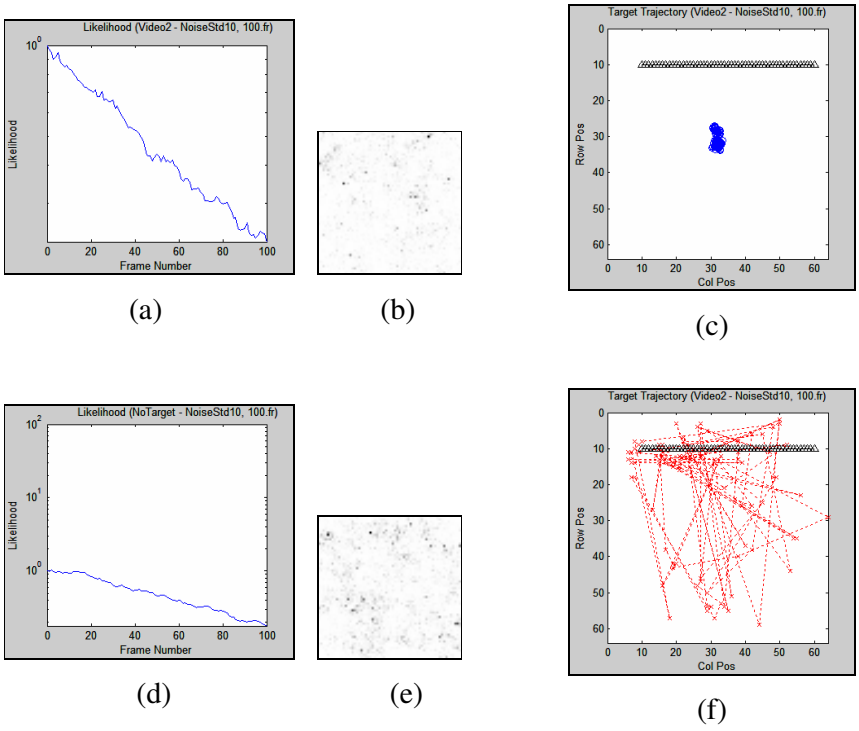
Figure 4-14 Simulation Result for Video-2 (σ_N = 7, 100 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-15 Simulation Result for Video-2 (σ_N = 10, 50 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-16 Simulation Result for Video-2 (σ_N = 10, 100 frames)

The figures between Figure 4-7 and Figure 4-16 present the simulation results of proposed algorithm for Video-2 for different noise levels.

The algorithm gives successful performance up to the noise level $\sigma_N = 7$. The increase of the likelihood ratio is apparent up to the noise level $\sigma_N = 7$ and the algorithm marked the true target location at the last stages of execution. Another important point to note is as follows: as noise levels increasing, higher number of iterations are required for a successful detection. In other words, the algorithm should be executed longer, as the noise energy increases.

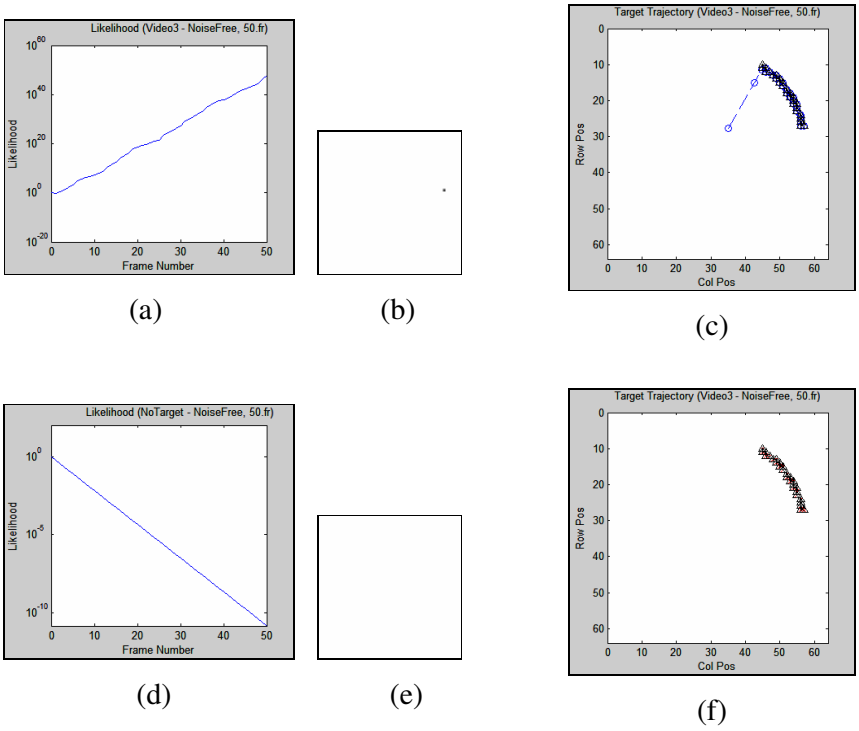


Figure 4-17 Simulation Result for Video-3 (Noise-free, 50 frames)

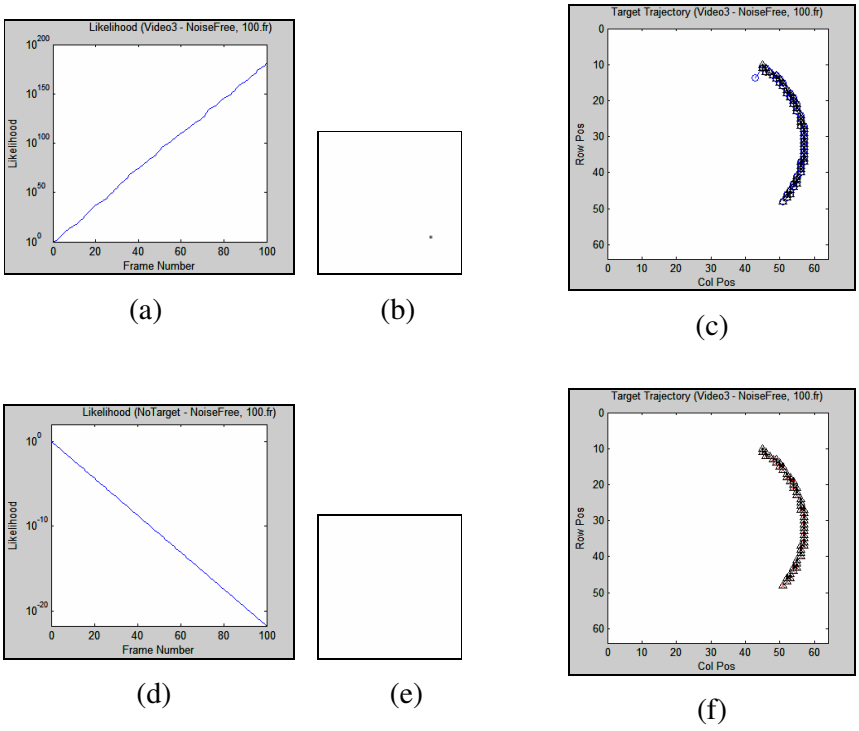
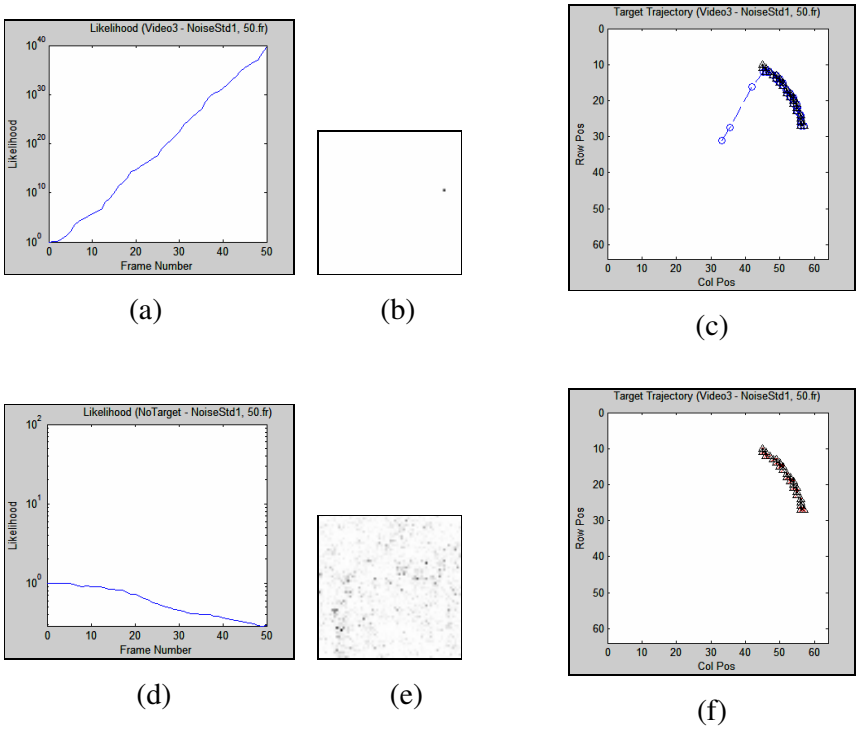


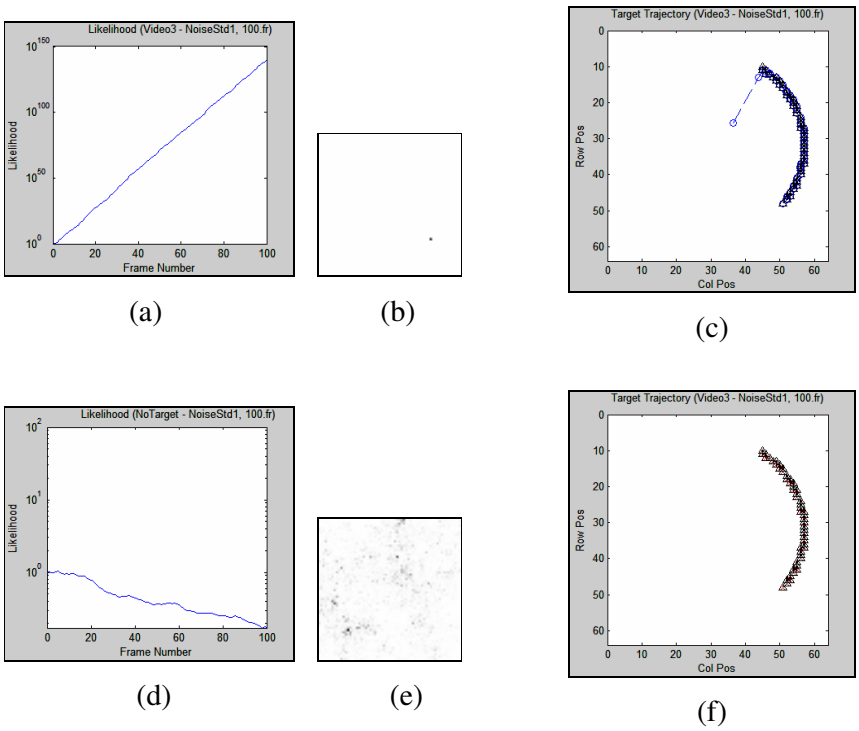
Figure 4-18 Simulation Result for Video-3 (Noise-free,100 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-19 Simulation Result for Video-3 (σ_N = 1, 50 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-20 Simulation Result for Video-3 (σ_N = 1, 100 frames)

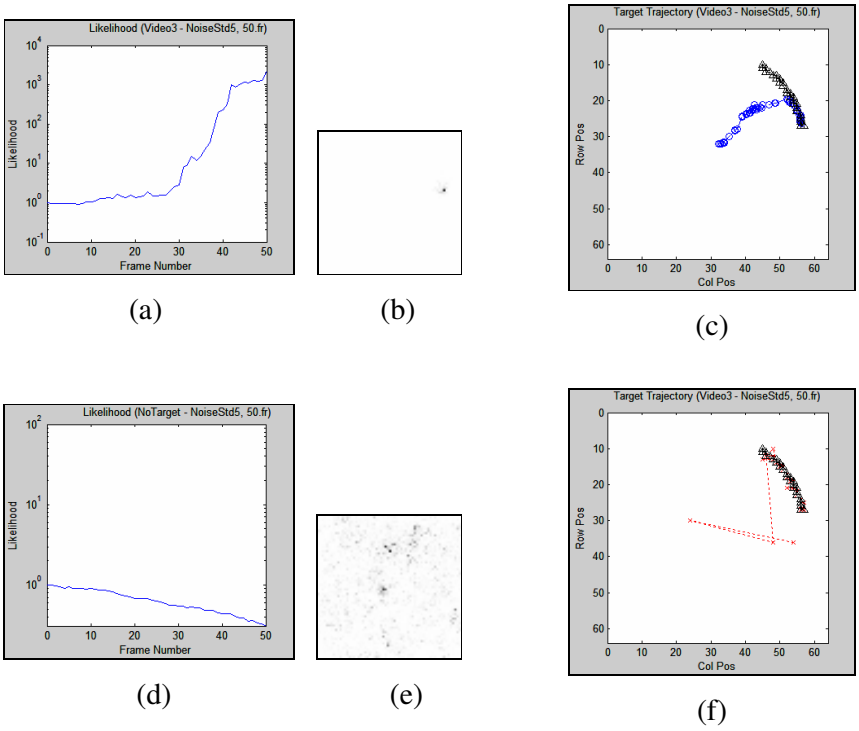


Figure 4-21 Simulation Result for Video-3 (σ_N = 5, 50 frames)

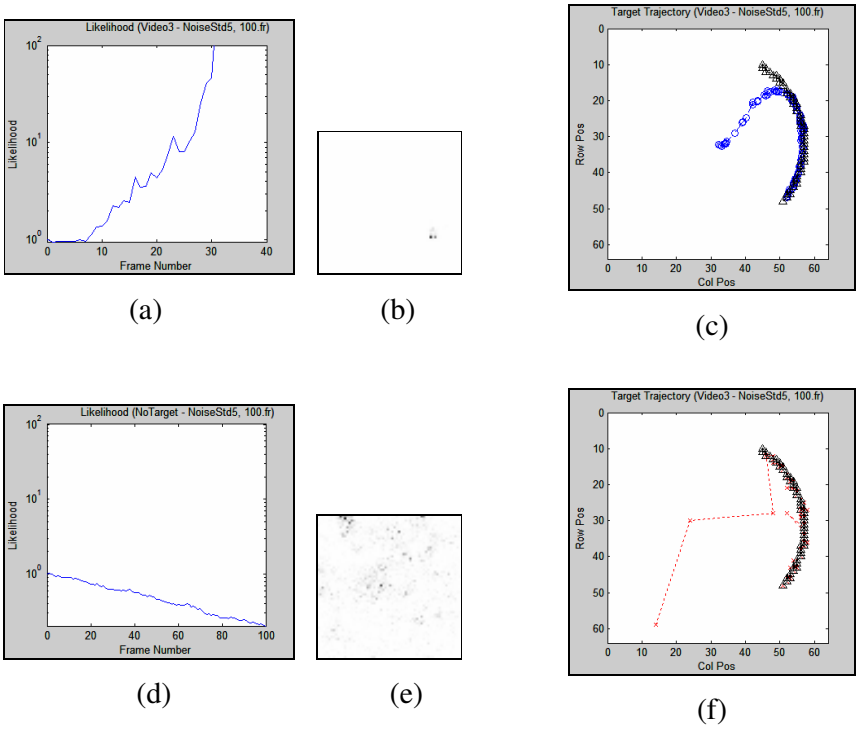


Figure 4-22 Simulation Result for Video-3 (σ_N = 5, 100 frames)

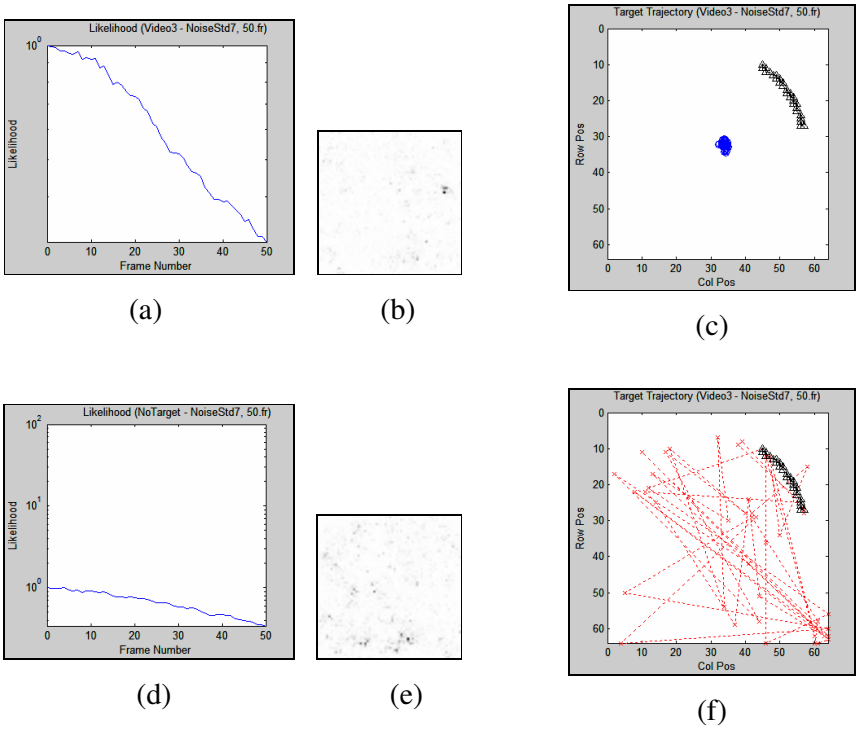


Figure 4-23 Simulation Result for Video-3 (σ_N = 7, 50 frames)

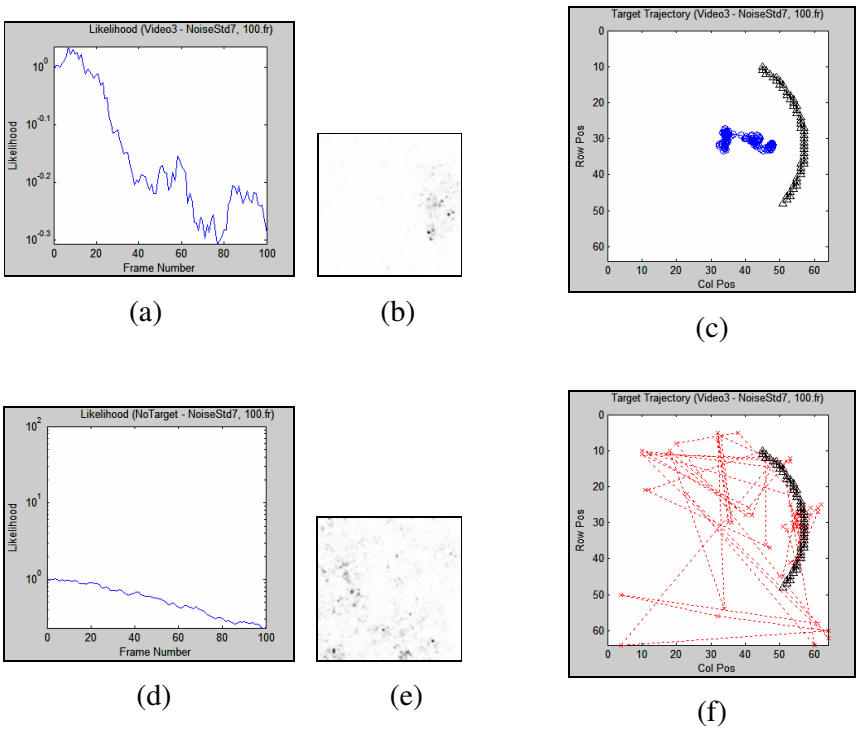


Figure 4-24 Simulation Result for Video-3 (σ_N = 7, 100 frames)

The figures between Figure 4-17 and Figure 4-24 gives the simulation results of the proposed algorithm for Video-3, where the target has a circular trajectory, at different noise levels and for 50 and 100 frame executions of the algorithm.

Since the speed of the target is quite fast for the algorithm could handle, the results were successful only up to noise level $\sigma_N = 5$. In this case, the same conclusion, as in the previous experiments, could be stated, as increasing the number of processed frames, improves the detection rate of the algorithm.

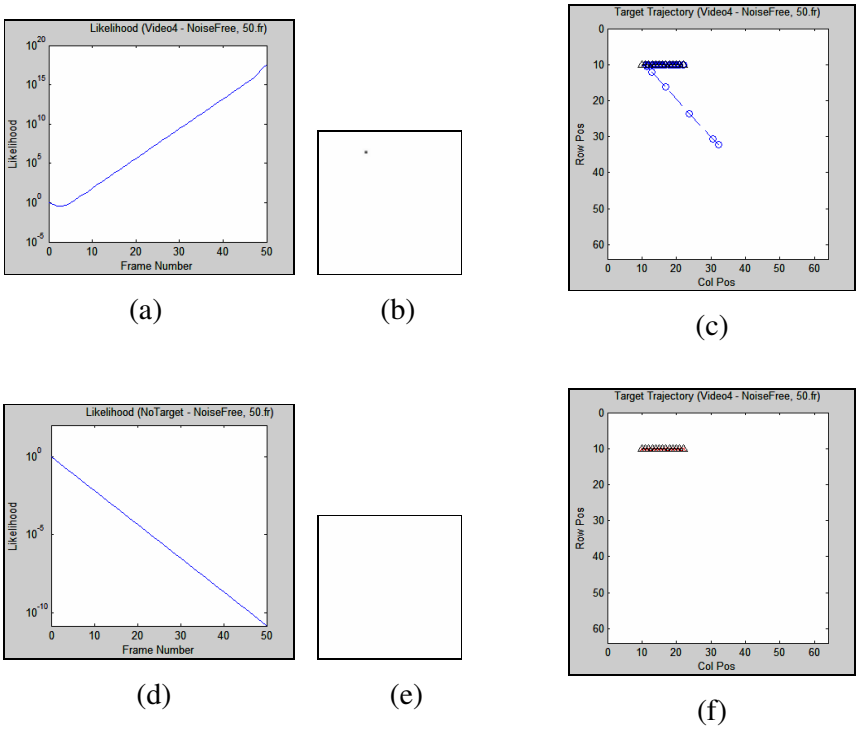


Figure 4-25 Simulation Result for Video-4 (Noise-free, 50 frames)

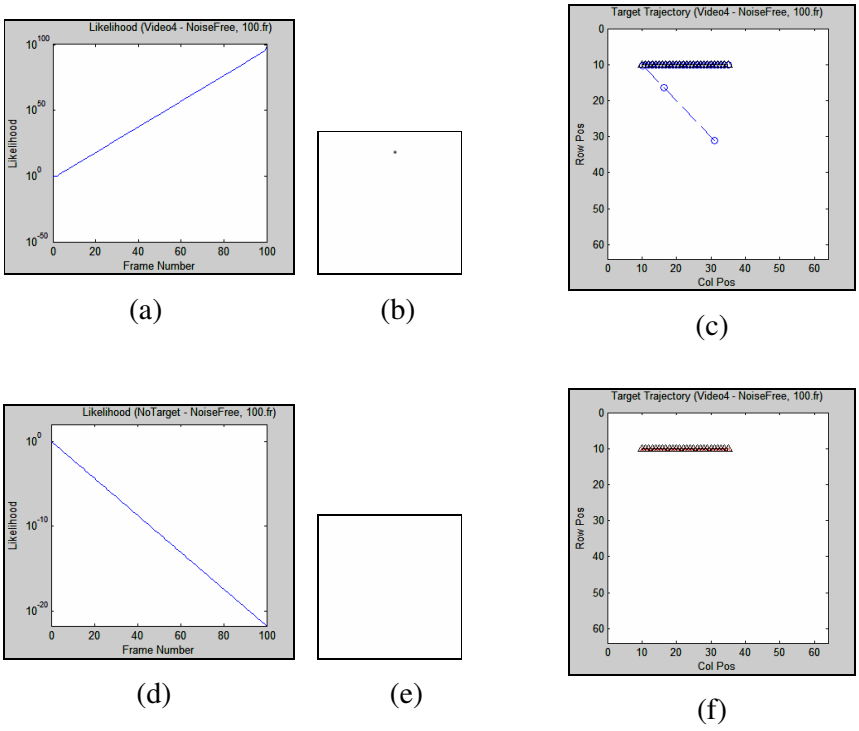


Figure 4-26 Simulation Result for Video-4 (Noise-free, 100 frames)

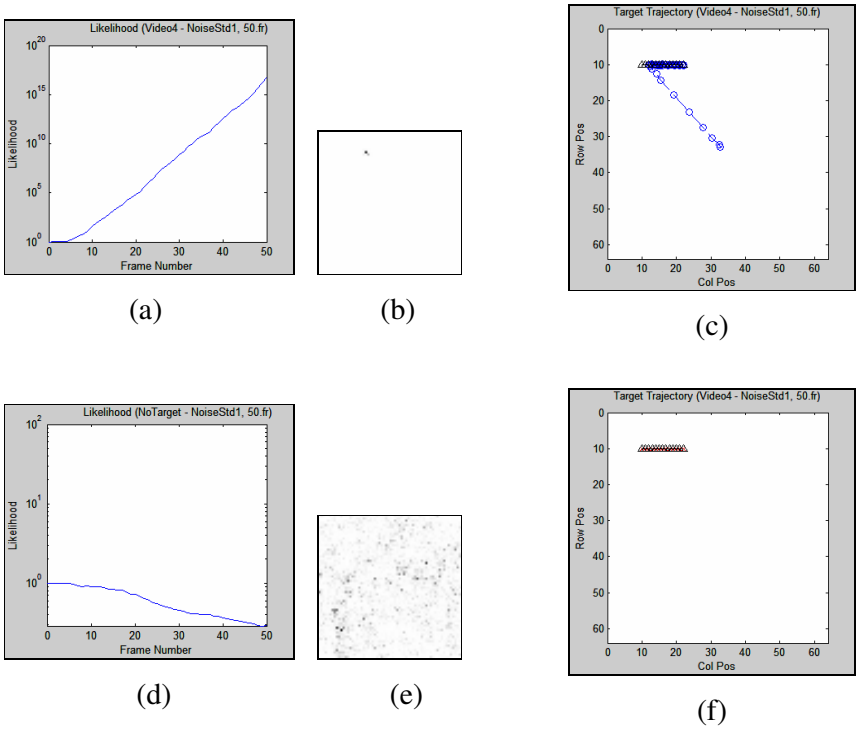


Figure 4-27 Simulation Result for Video-4 (σ_N = 1, 50 frames)

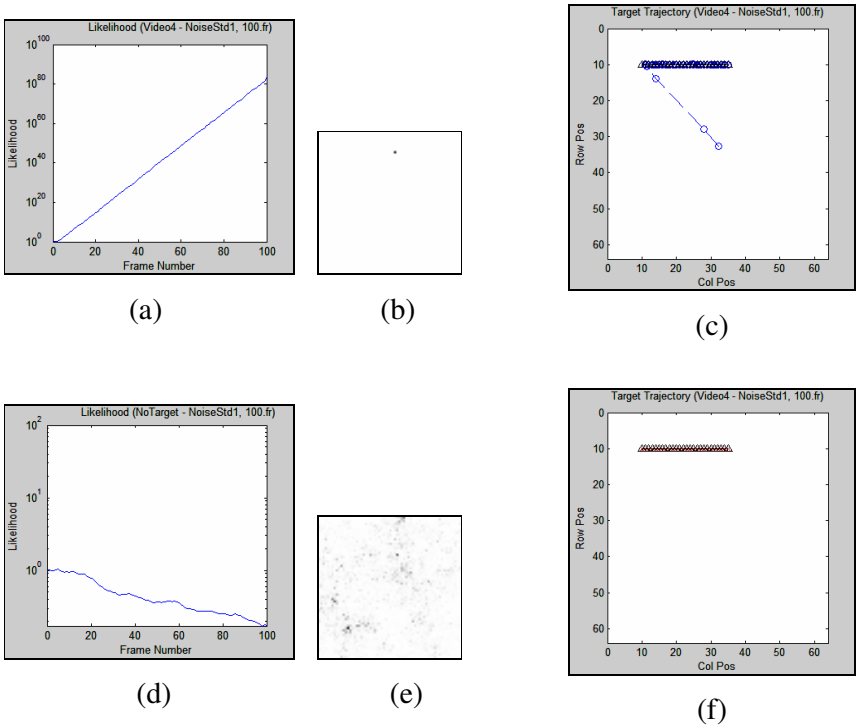


Figure 4-28 Simulation Result for Video-4 (σ_N = 1, 100 frames)

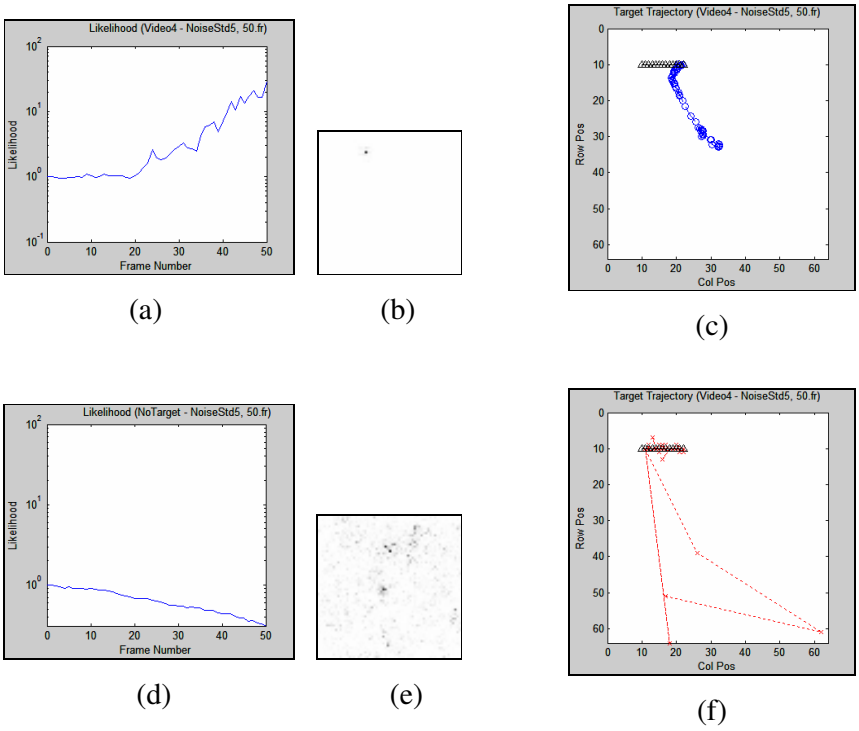


Figure 4-29 Simulation Result for Video-4 (σ_N = 5, 50 frames)

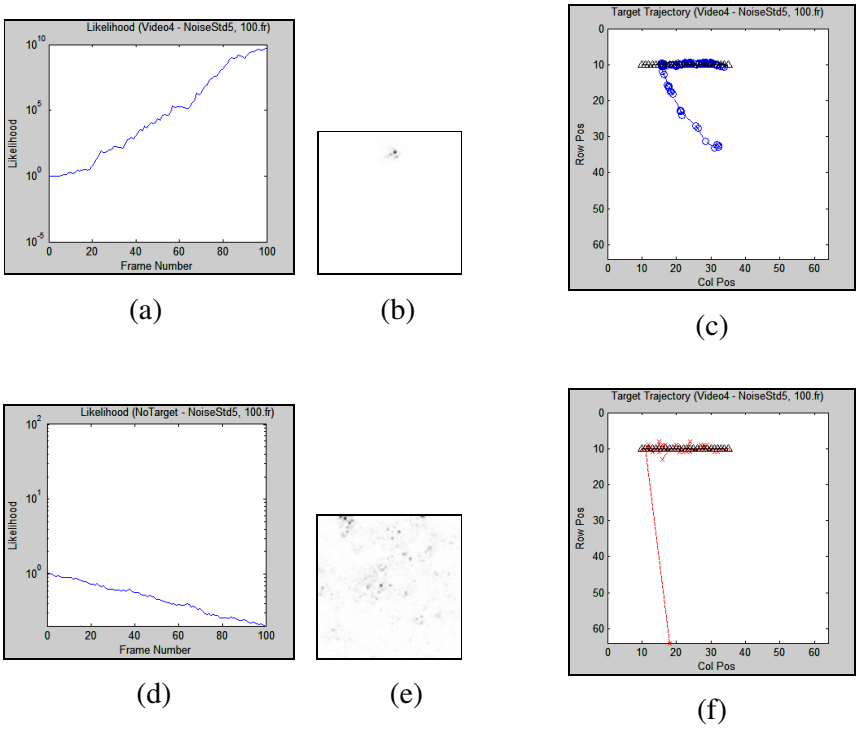


Figure 4-30 Simulation Result for Video-4 (σ_N = 5, 100 frames)

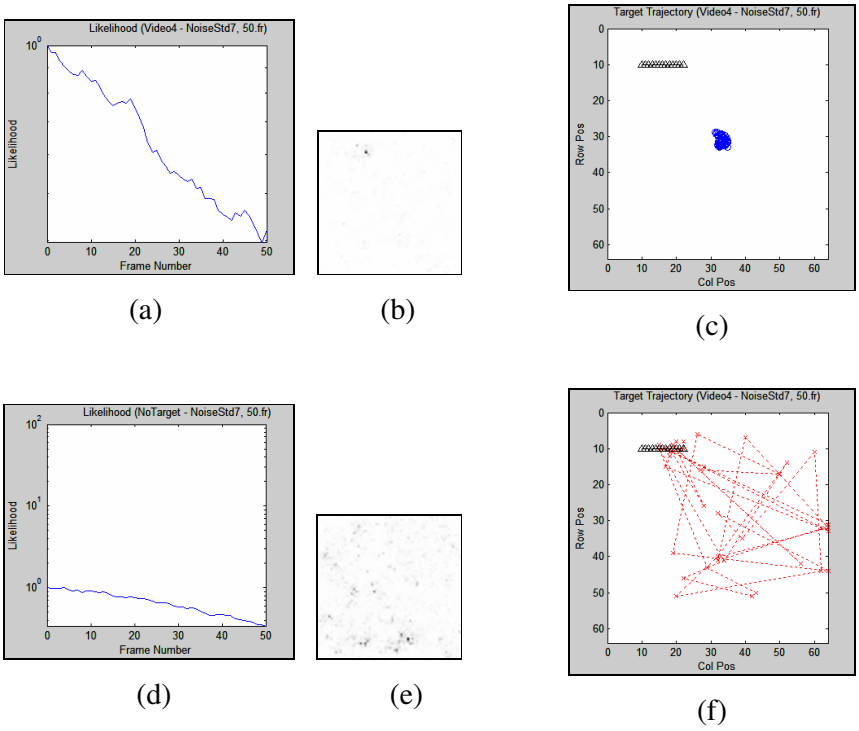


Figure 4-31 Simulation Result for Video-4 (σ_N = 7, 50 frames)

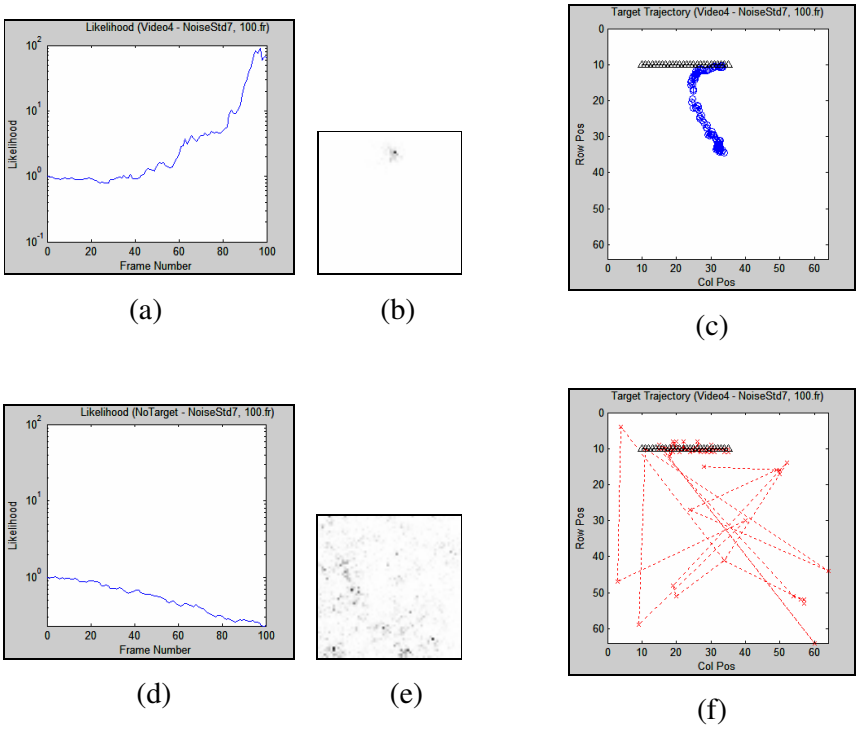
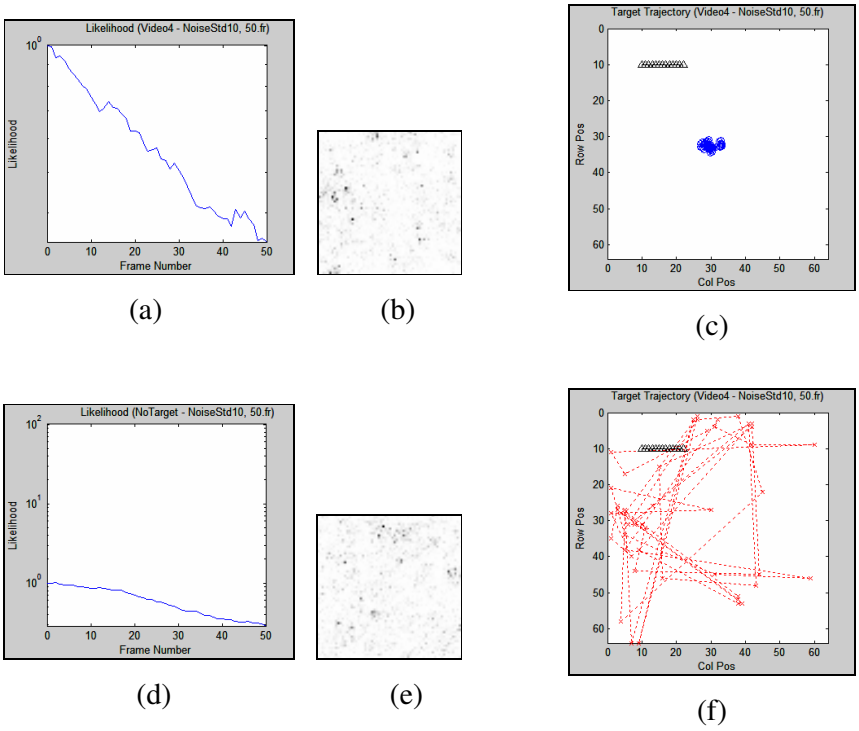


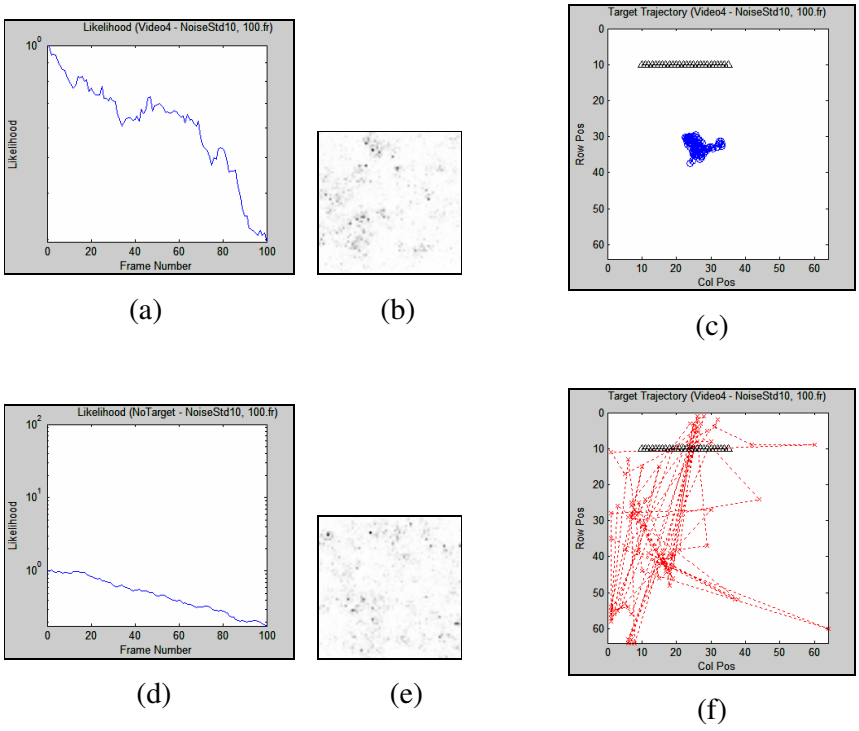
Figure 4-32 Simulation Result for Video-4 (σ_N = 7, 100 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-33 Simulation Result for Video-4 (σ_N = 10, 50 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-34 Simulation Result for Video-4 (σ_N = 10, 100 frames)

The figures between Figure 4-25 and Figure 4-34 show the results for the proposed algorithm for Video-4, where the target has a similar trajectory as in Video-2, at different noise levels for different sequence lengths.

Since the speed of the target is slower than that of in Video-2, the algorithm marked the true location of the target up to the noise level $\sigma_N = 7$, at the end of 100 frame execution.

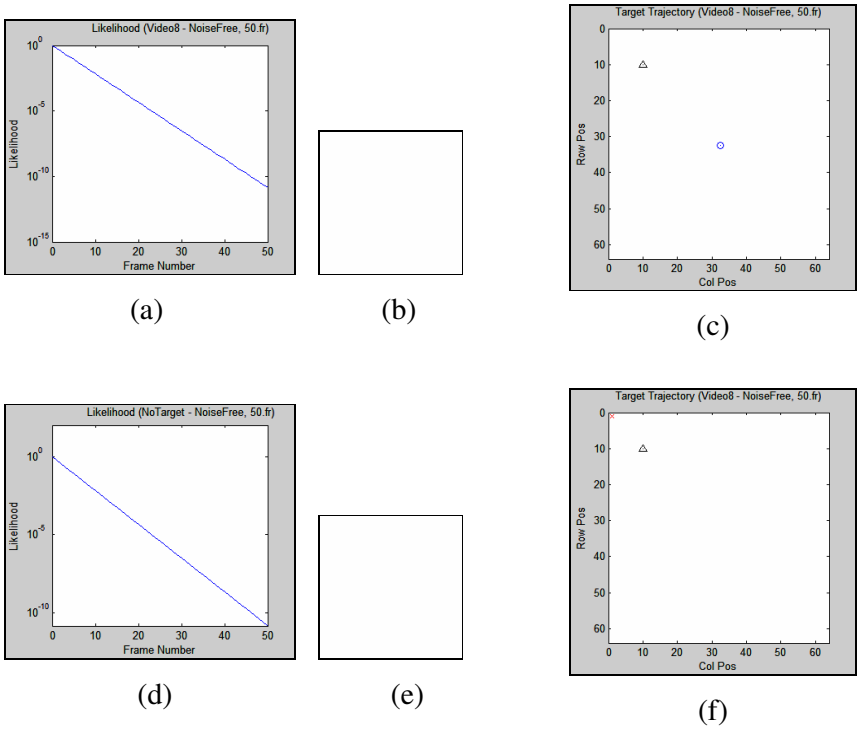


Figure 4-35 Simulation Result for Video-5 (Noise-free, 50 frames)

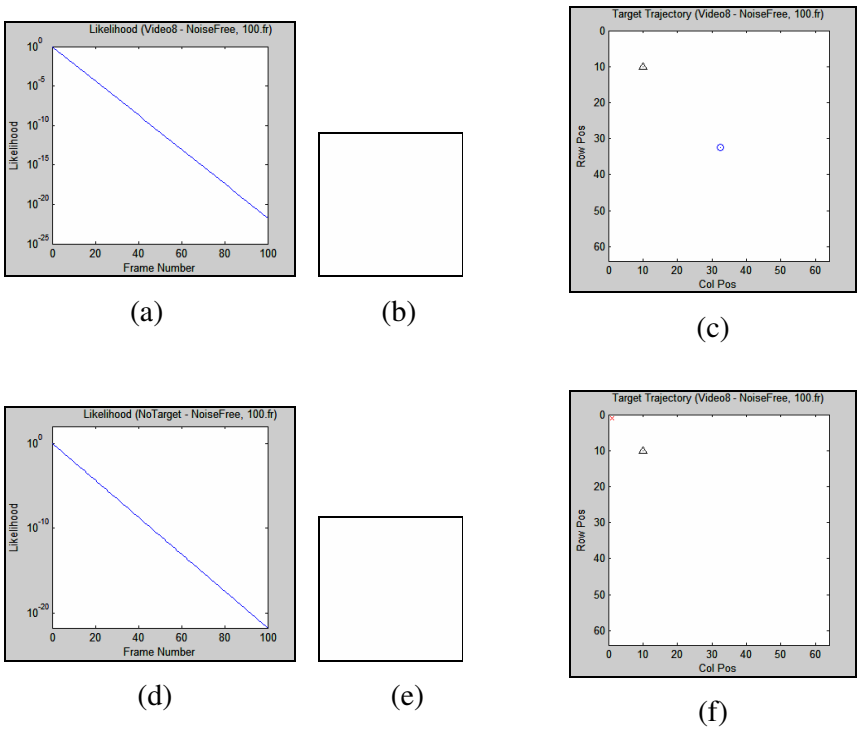


Figure 4-36 Simulation Result for Video-5 (Noise-free,100 frames)

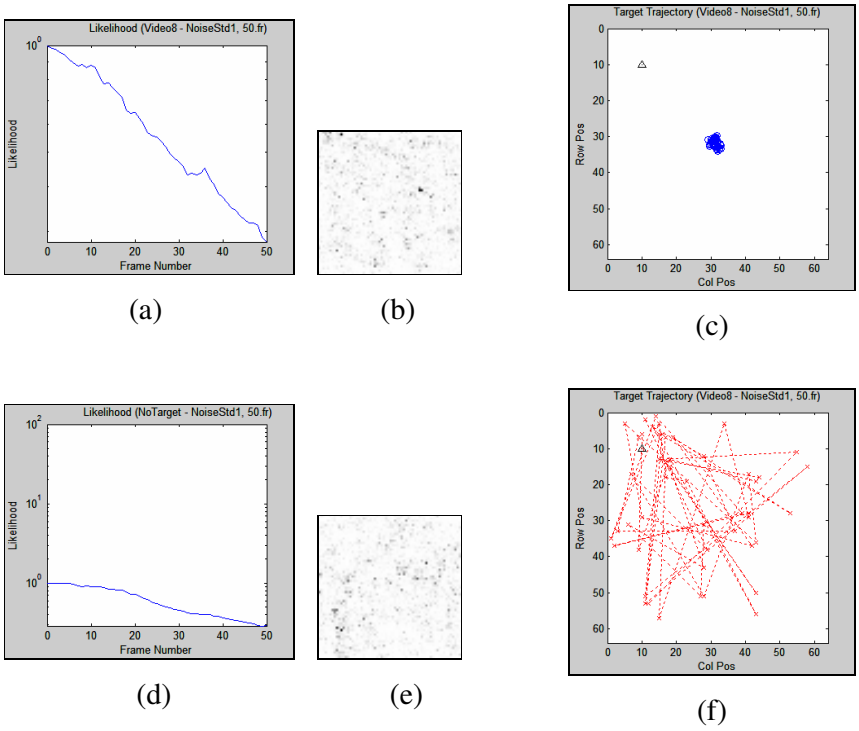


Figure 4-37 Simulation Result for Video-5 (σ_N = 1, 50 frames)

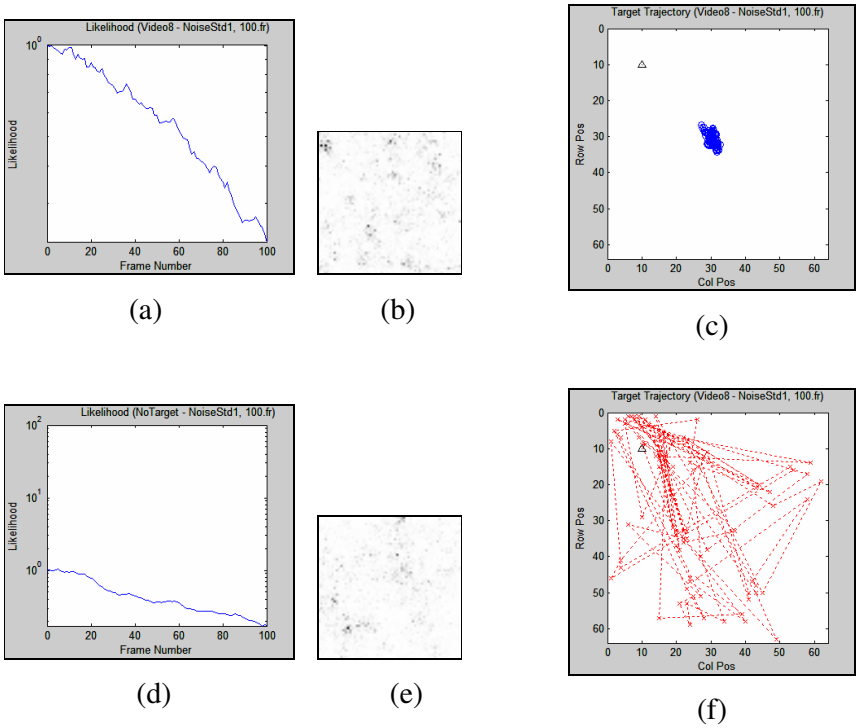


Figure 4-38 Simulation Result for Video-5 (σ_N = 1, 100 frames)

The figures between Figure 4-35 and Figure 4-38 present the simulations for the proposed algorithm for Video-5, where the target stays stationary along the sequence of frames, at different noise levels and for 50 and 100 frame executions of the algorithm.

These unexpected results show that the algorithm is not successful even for the noise free case, since the size of the target was one-pixel and it does not move along the sequence.

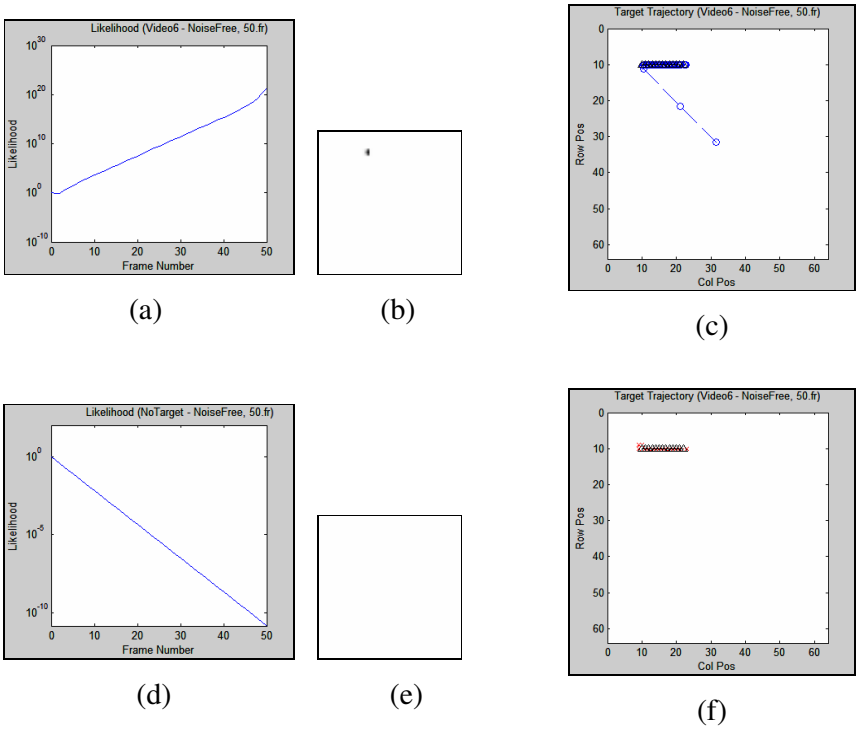


Figure 4-39 Simulation Result for Video-6 (Noise-free, 50 frames)

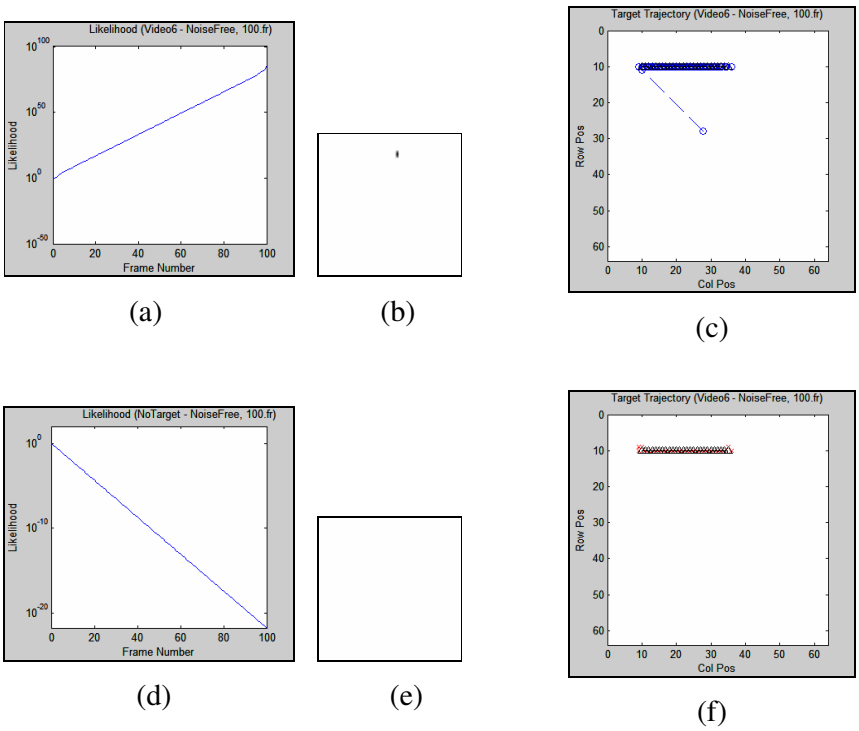
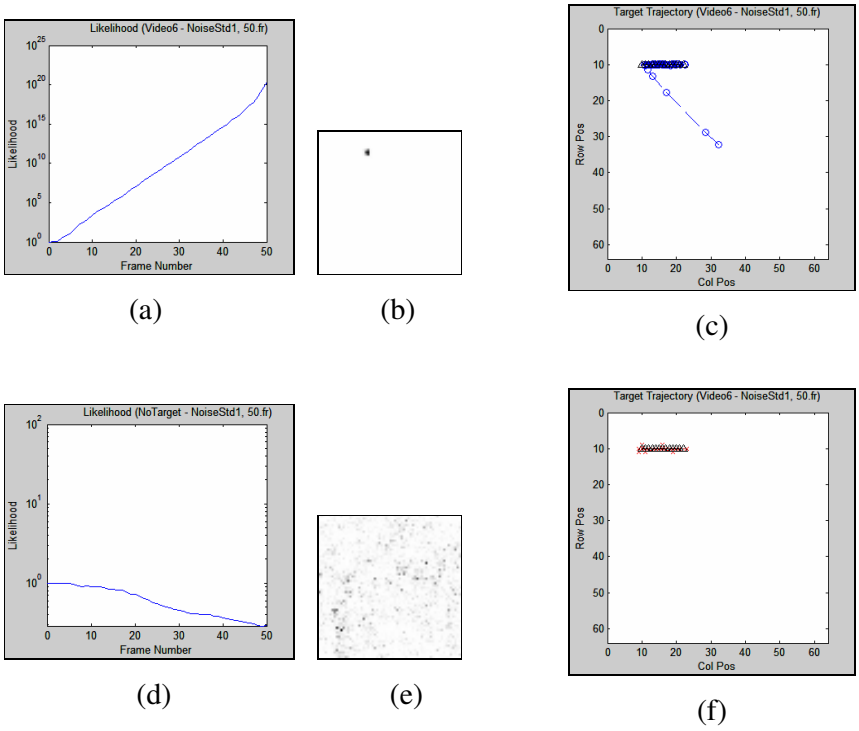


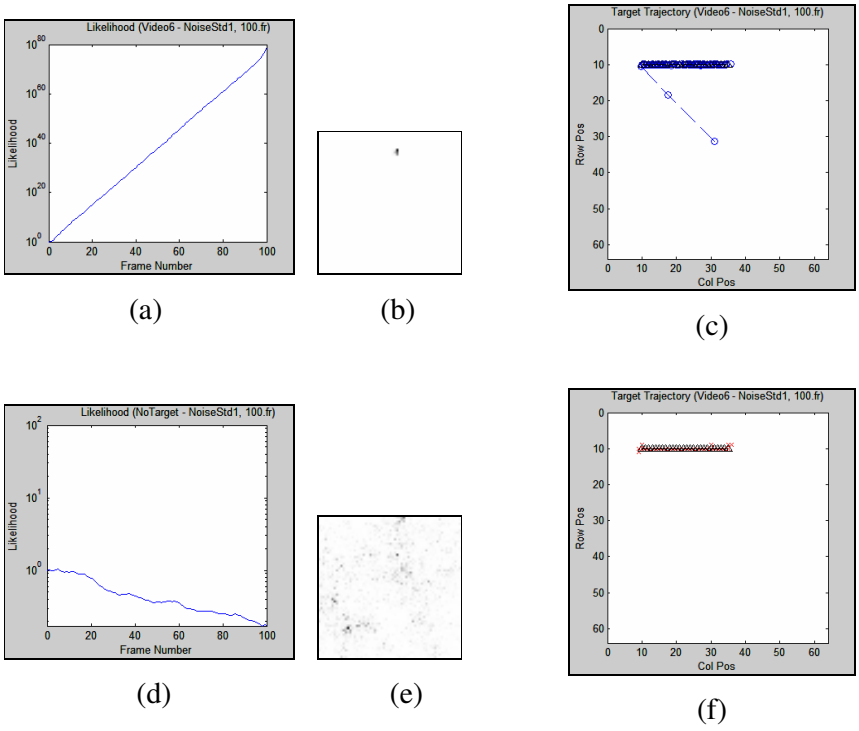
Figure 4-40 Simulation Result for Video-6 (Noise-free,100 frames)



(a) (b) (c)

(d) (e) (f)

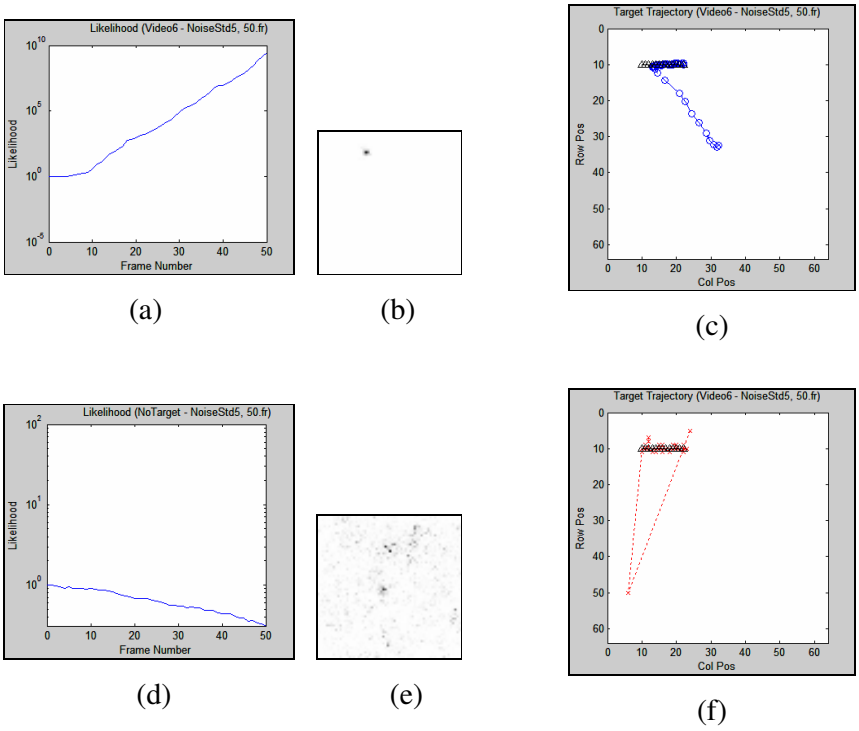
Figure 4-41 Simulation Result for Video-6 (σ_N = 1, 50 frames)



(a) (b) (c)

(d) (e) (f)

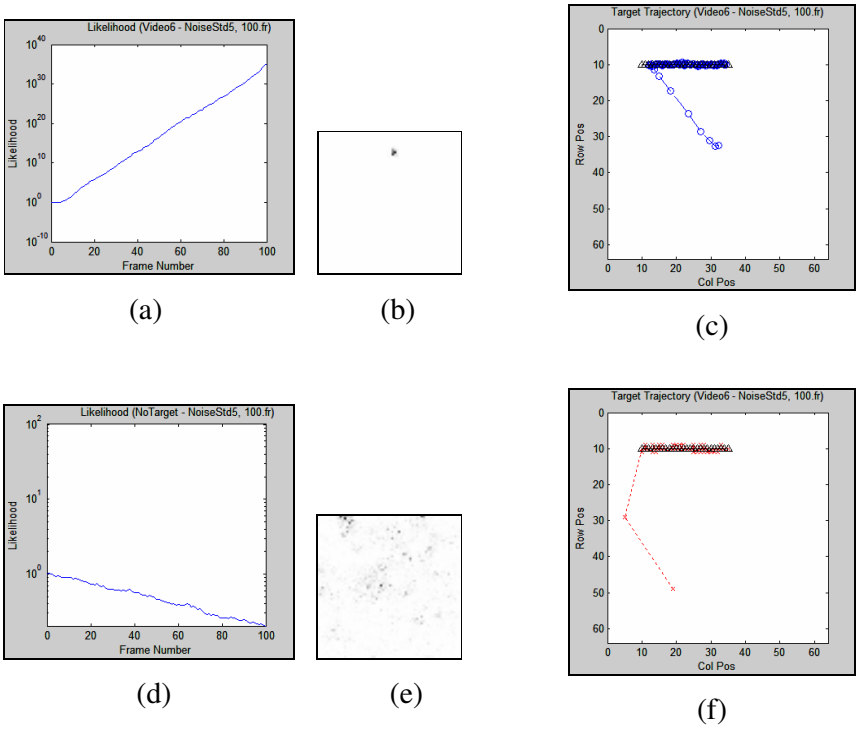
Figure 4-42 Simulation Result for Video-6 (σ_N = 1, 100 frames)



(a) (b) (c)

(d) (e) (f)

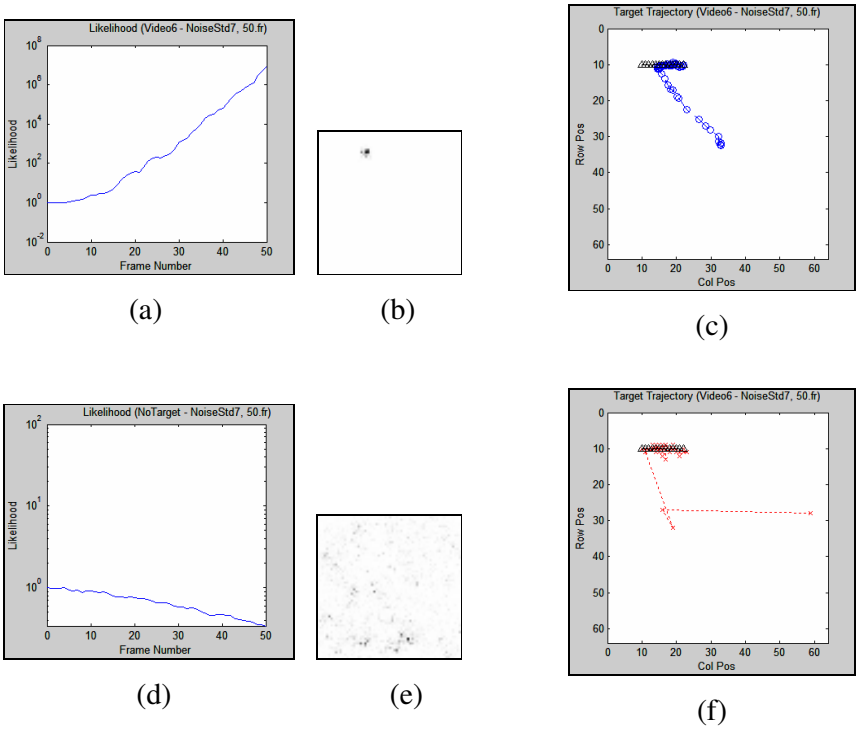
Figure 4-43 Simulation Result for Video-6 (σ_N = 5, 50 frames)



(a) (b) (c)

(d) (e) (f)

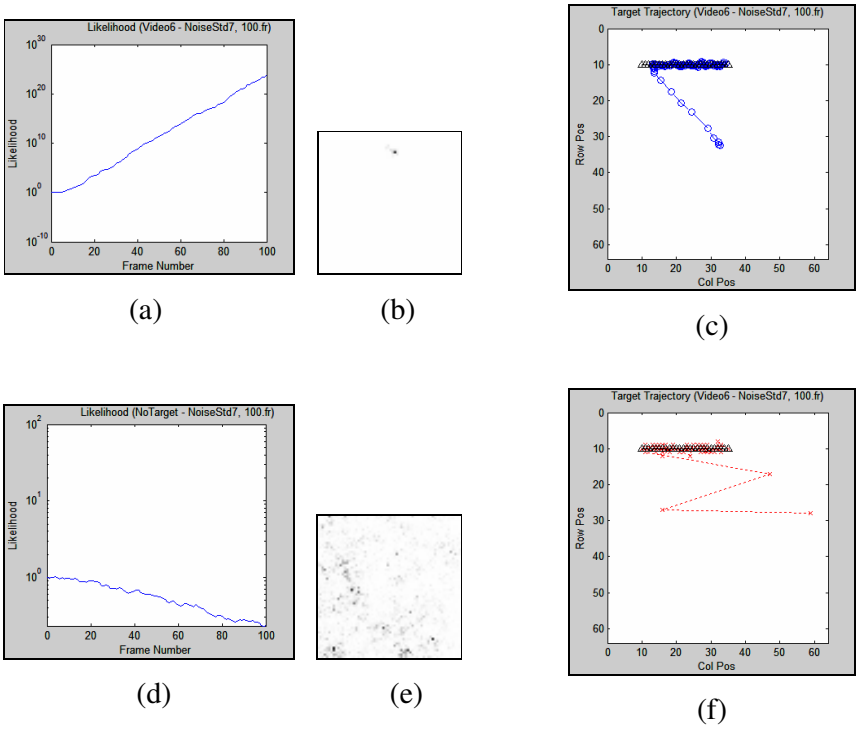
Figure 4-44 Simulation Result for Video-6 (σ_N = 5, 100 frames)



(a) (b) (c)

(d) (e) (f)

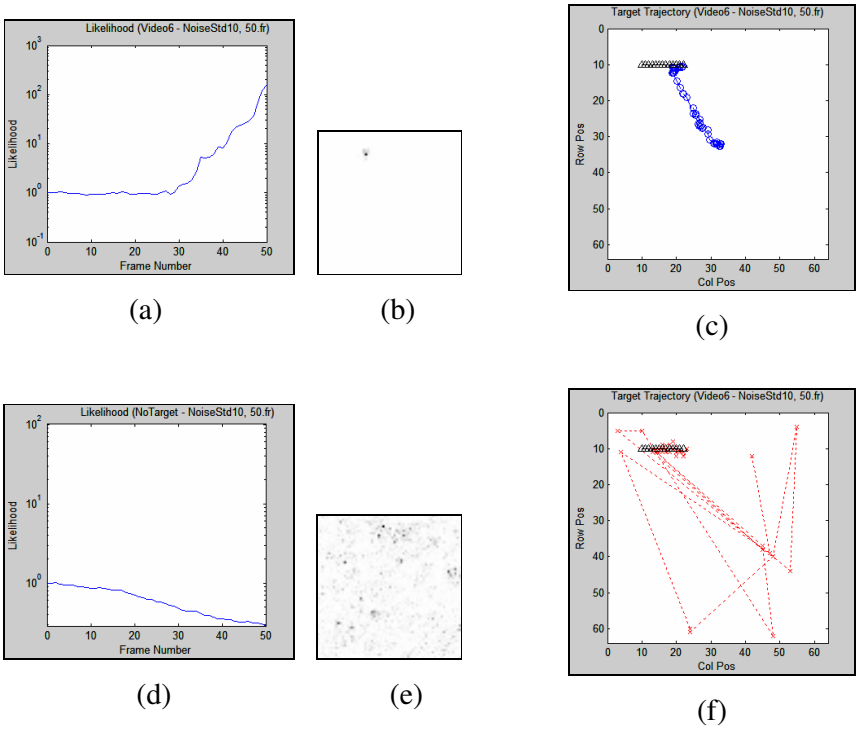
Figure 4-45 Simulation Result for Video-6 (σ_N = 7, 50 frames)



(a) (b) (c)

(d) (e) (f)

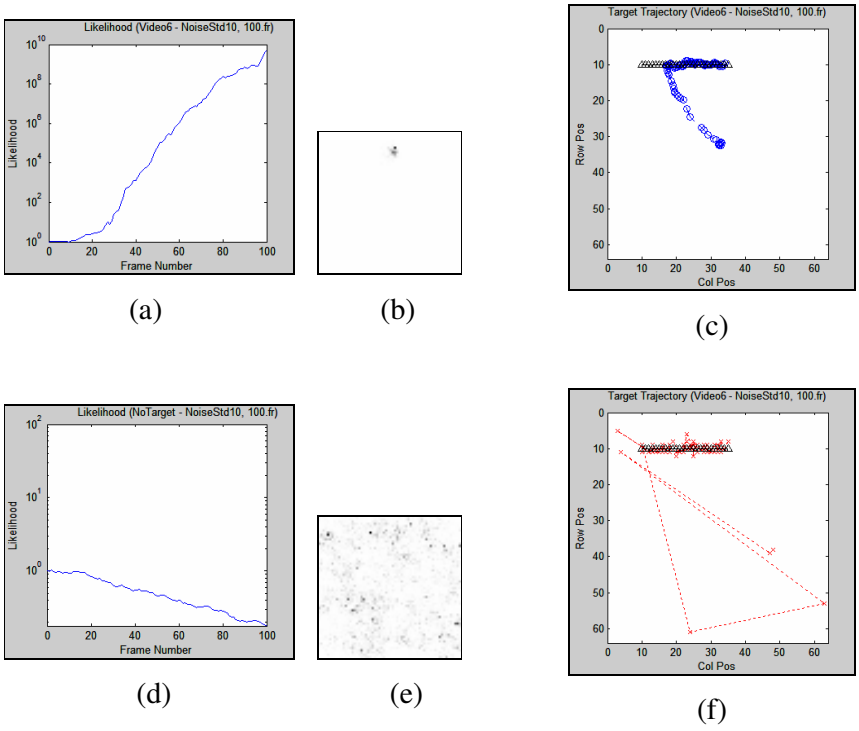
Figure 4-46 Simulation Result for Video-6 (σ_N = 7, 100 frames)



(a) (b) (c)

(d) (e) (f)

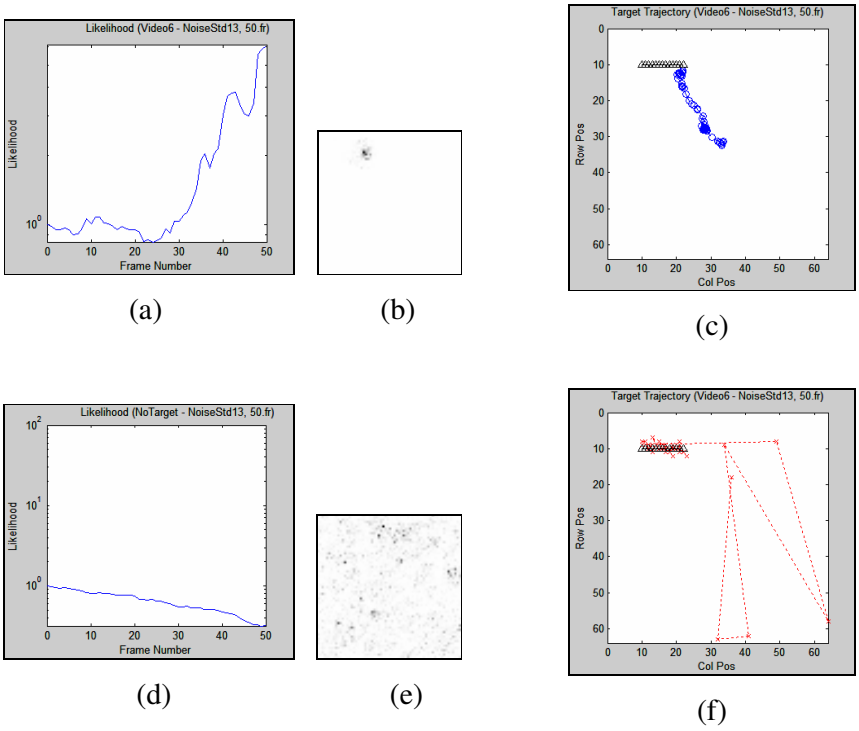
Figure 4-47 Simulation Result for Video-6 (σ_N = 10, 50 frames)



(a) (b) (c)

(d) (e) (f)

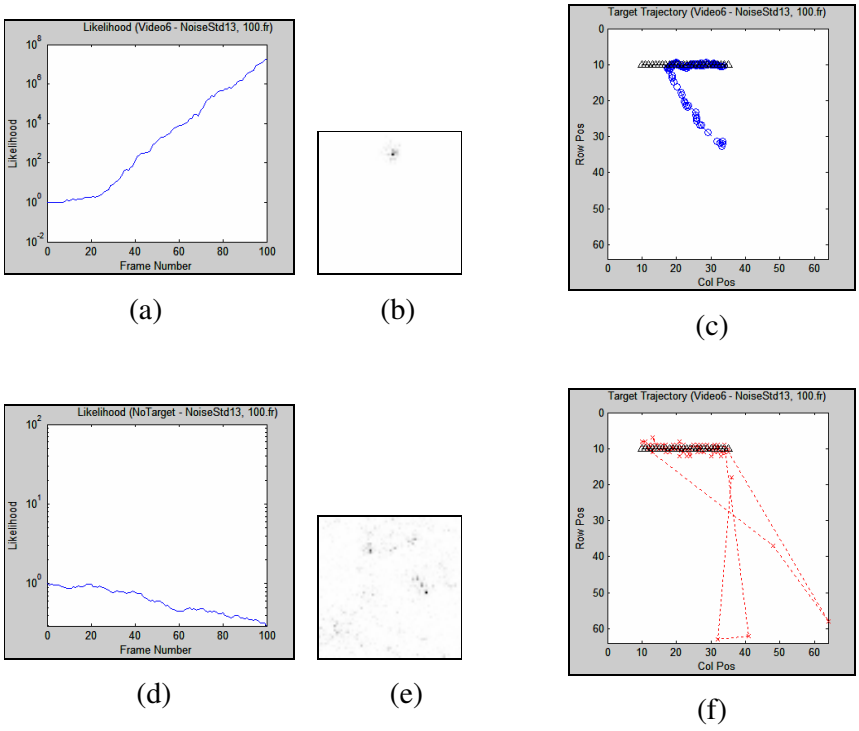
Figure 4-48 Simulation Result for Video-6 (σ_N = 10, 100 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-49 Simulation Result for Video-6 (σ_N = 13, 50 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-50 Simulation Result for Video-6 (σ_N = 13, 100 frames)

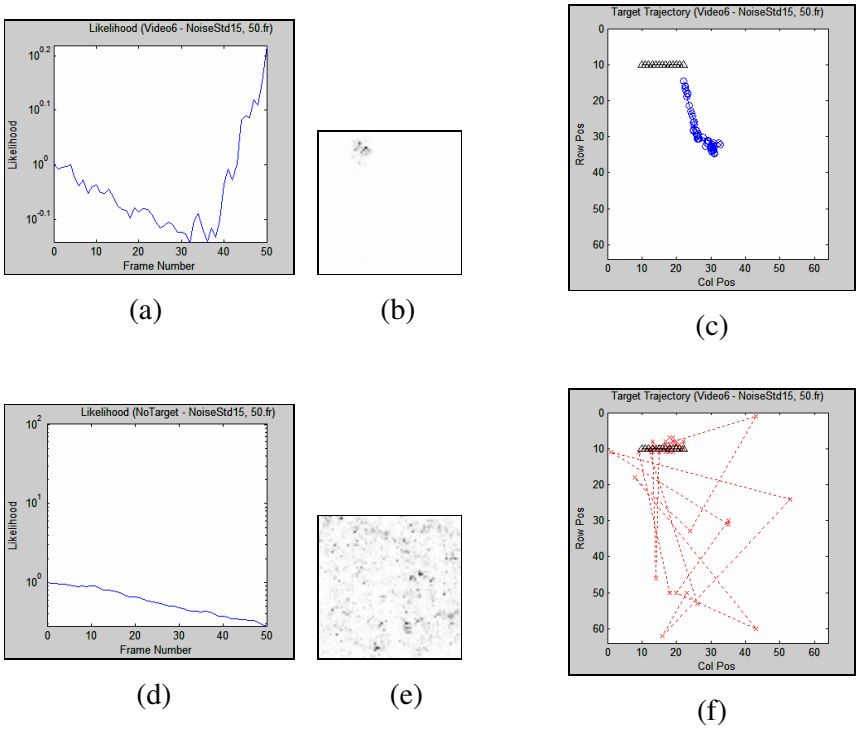


Figure 4-51 Simulation Result for Video-6 (σ_N = 15, 50 frames)

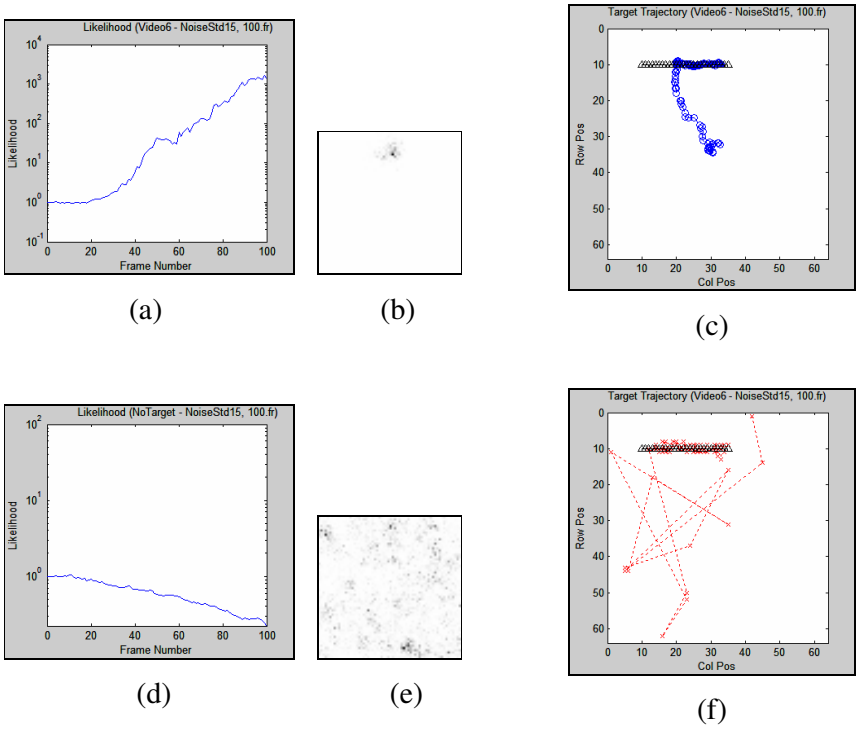


Figure 4-52 Simulation Result for Video-6 (σ_N = 15, 100 frames)

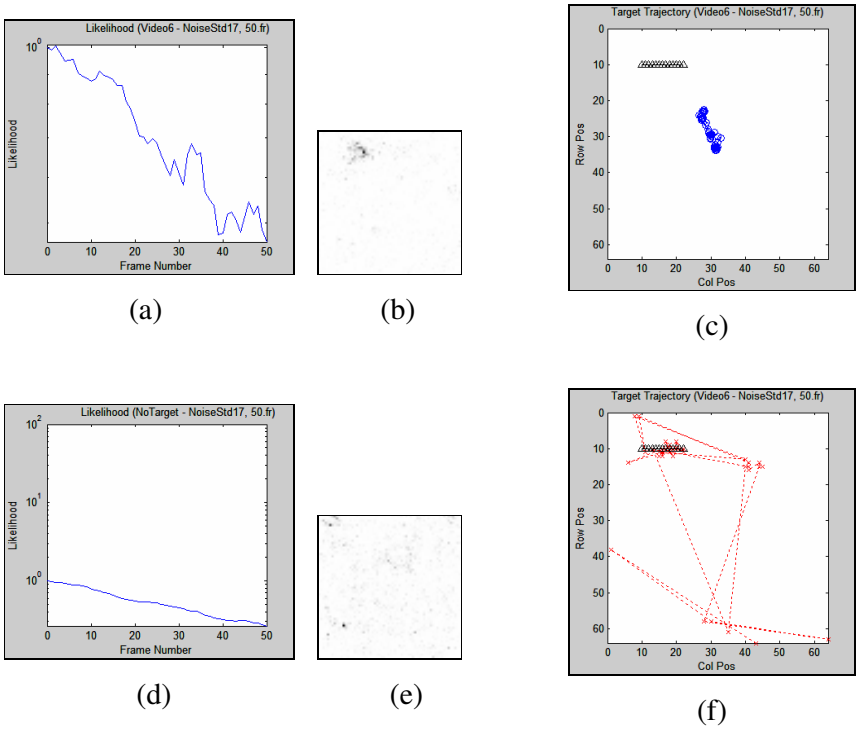


Figure 4-53 Simulation Result for Video-6 (σ_N = 17, 50 frames)

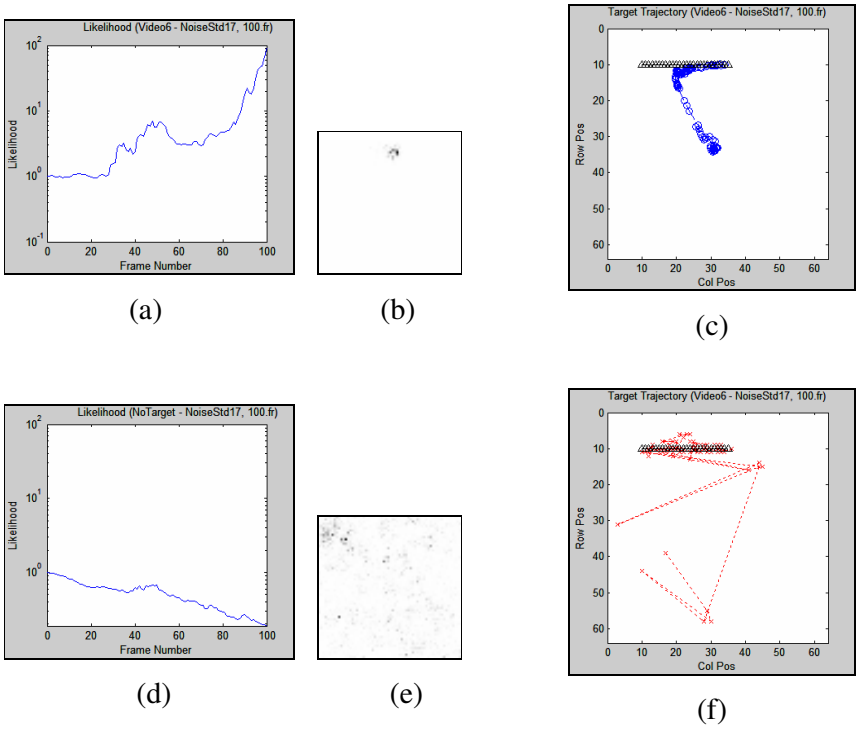
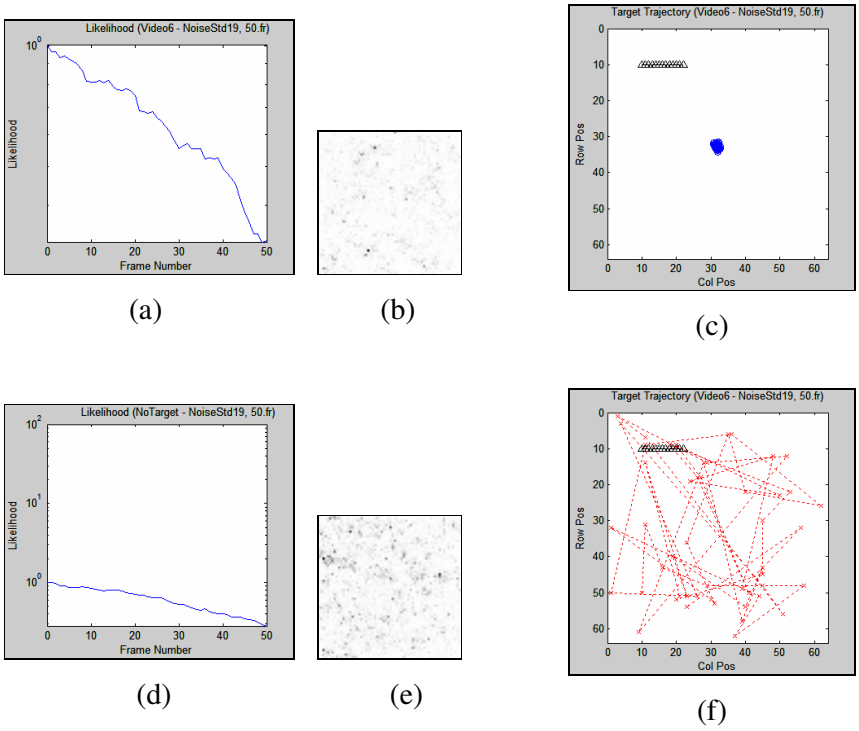


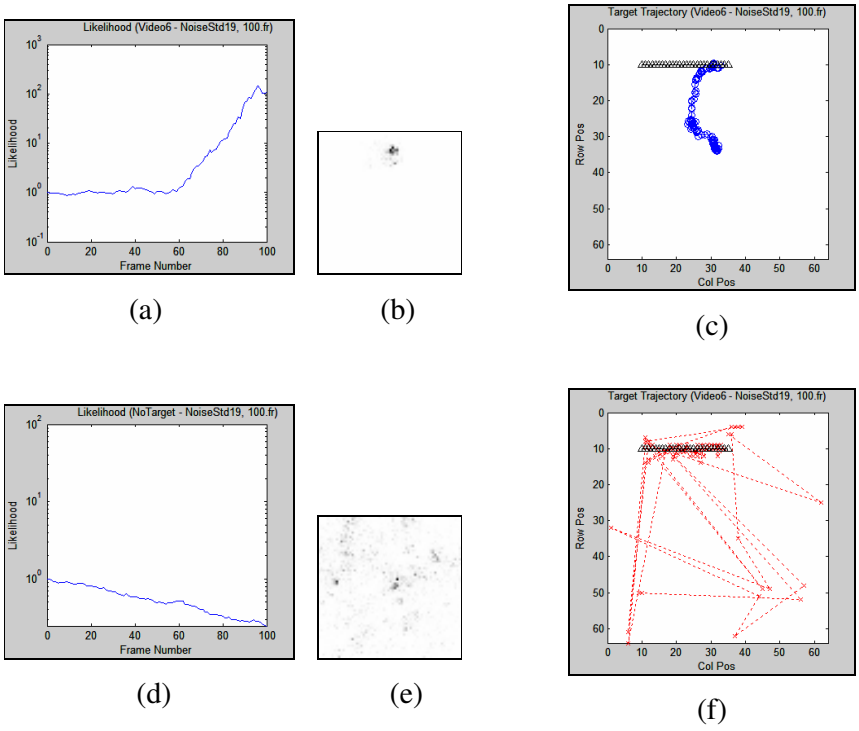
Figure 4-54 Simulation Result for Video-6 (σ_N = 17, 100 frames)



(a) (b) (c)

(d) (e) (f)

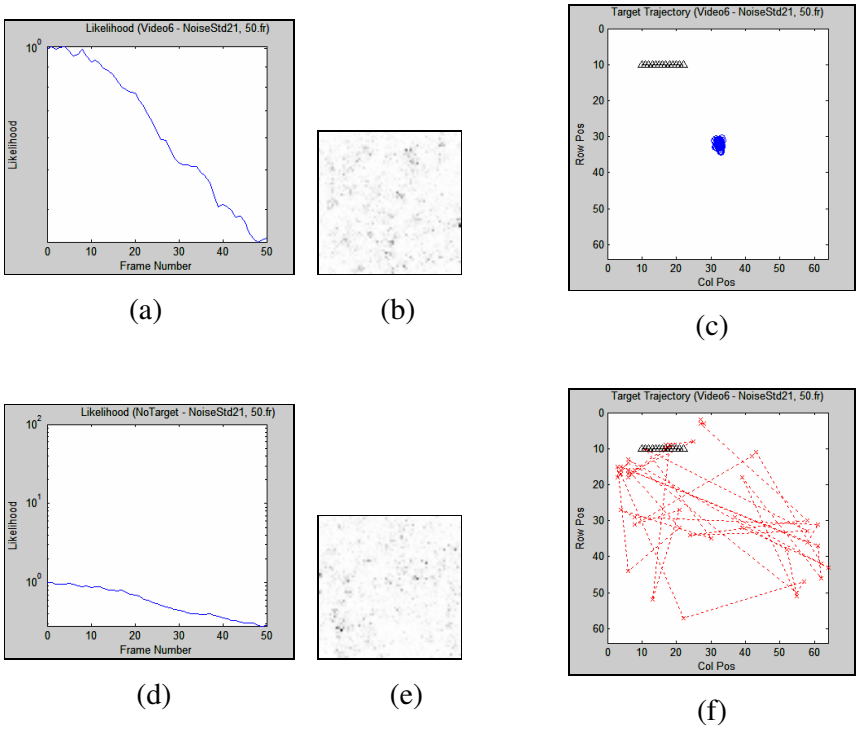
Figure 4-55 Simulation Result for Video-6 (σ_N = 19, 50 frames)



(a) (b) (c)

(d) (e) (f)

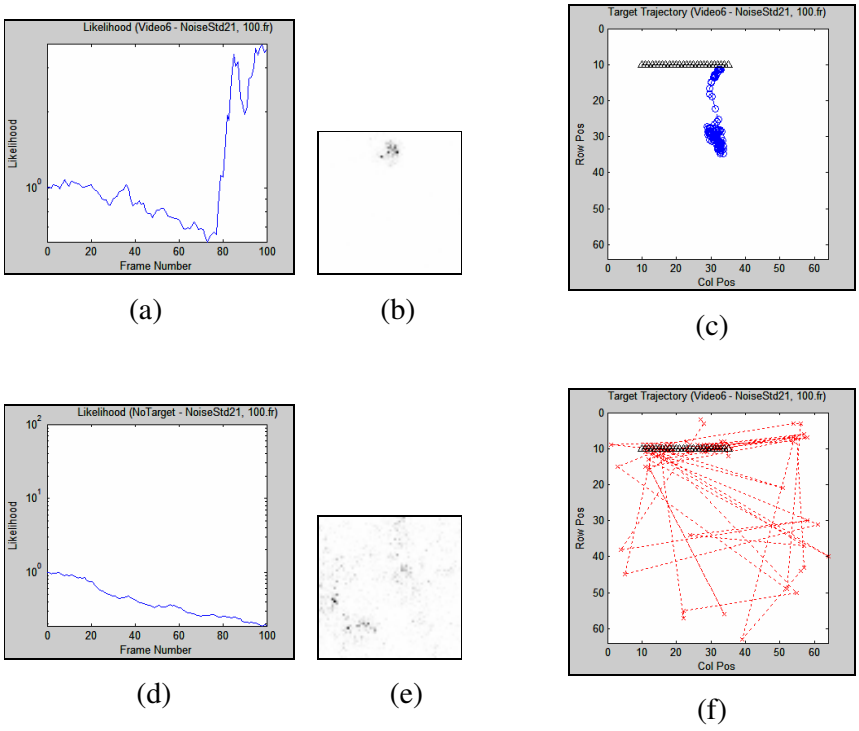
Figure 4-56 Simulation Result for Video-6 (σ_N = 19, 100 frames)



(a) (b) (c)

(d) (e) (f)

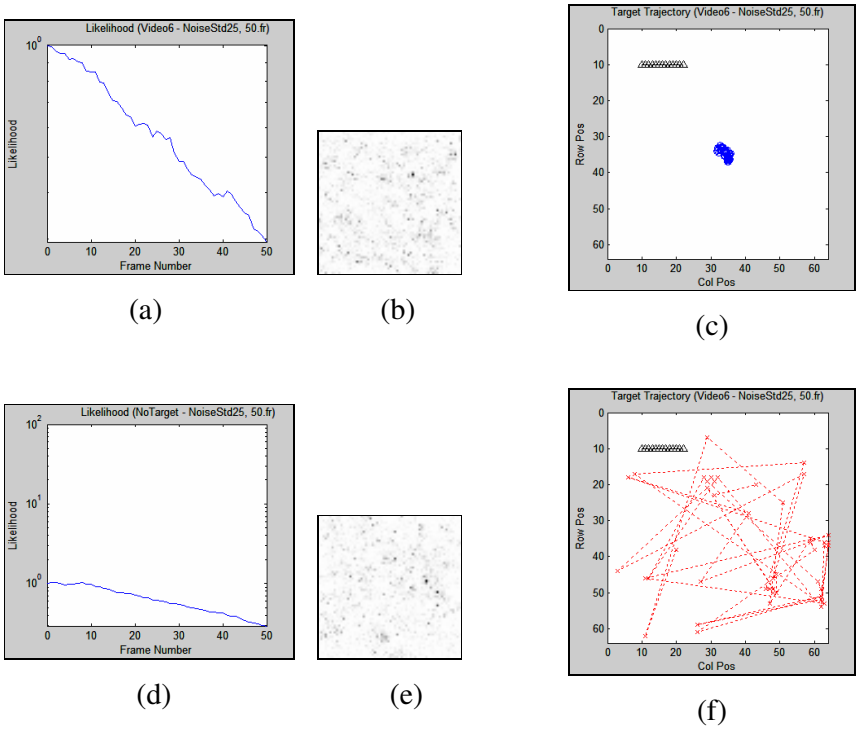
Figure 4-57 Simulation Result for Video-6 (σ_N = 21, 50 frames)



(a) (b) (c)

(d) (e) (f)

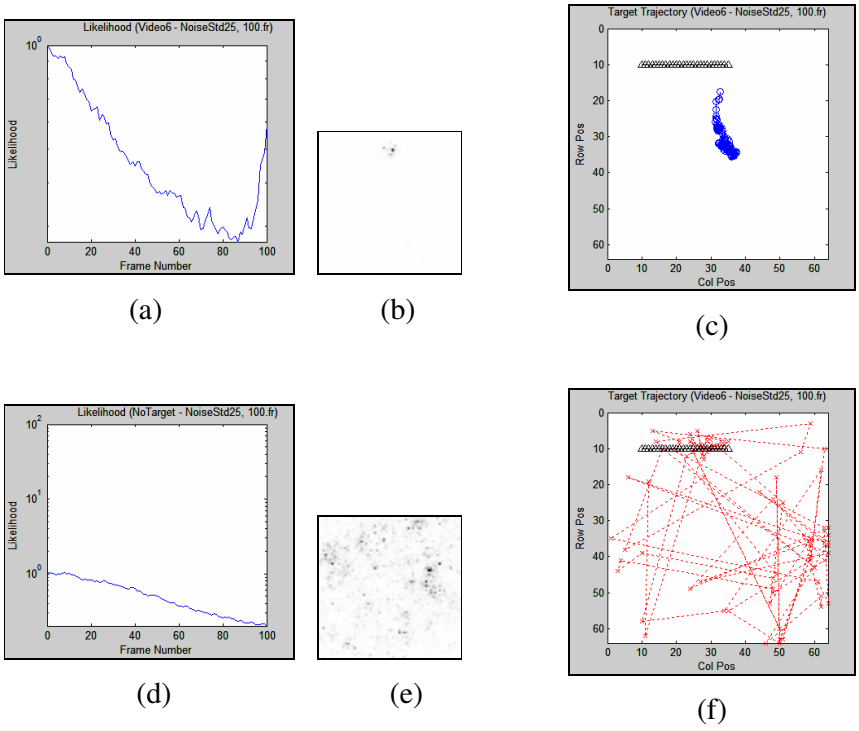
Figure 4-58 Simulation Result for Video-6 (σ_N = 21, 100 frames)



(a) (b) (c)

(d) (e) (f)

Figure 4-59 Simulation Result for Video-6 (σ_N = 25, 50 frames)



(a) (b) (c)

(d) (e) (f)

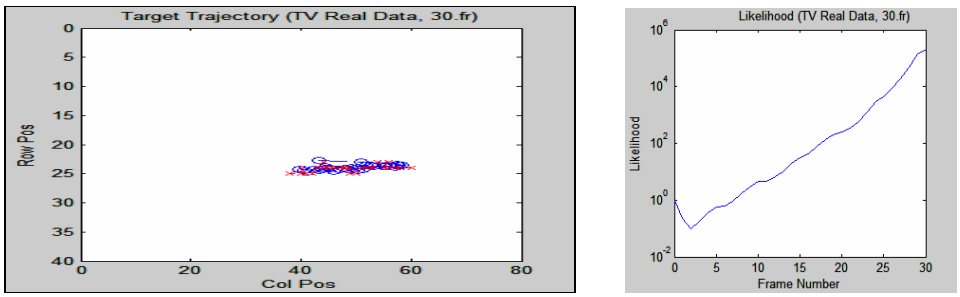
Figure 4-60 Simulation Result for Video-6 (σ_N = 25, 100 frames)

The figures between Figure 4-39 and Figure 4-60 give the simulation results for the proposed algorithm for Video-6, where the target is modeled, as 3x3 pixels in size and being the slowest target among all of the previous experiments, at different noise levels and for 50 and 100 frame executions of the algorithm. This target is clearly a better fit for the Gaussian assumption.

The performance of the proposed algorithm up to the noise levels of $\sigma_N = 21$ (corresponds to a SNR level of -4.16 dB) is remarkable. Since the speed of the target is quite slow (0.2 pixels/frame) and the target has a Gaussian form in shape, the algorithm exhibits this superb performance for the Video-6 case.

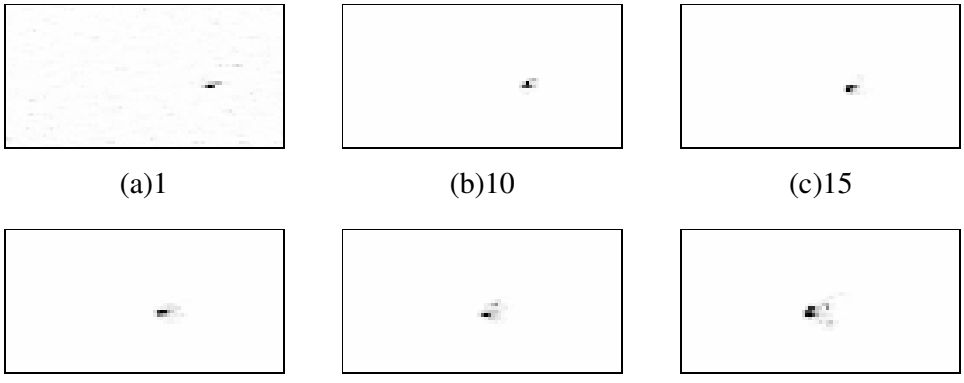
4.2.2 Real Test Data Set

For the real test set, an air-vehicle in far distance is recorded by using a day-light sensor. A typical frame from the recorded sequence is given in Figure 3-16.



(a) (b)

Figure 4-61 Simulation Result of Bayesian Algorithm
for the day-light sensor sequence
(a) Target state estimates (estimated vs. groundtruth)
(b) Likelihood Ratio (in log-scale)



(a)1 (b)10 (c)15

(d)20 (e)25 (f)30

Figure 4-62 The evolution of the target posterior density along the sequence
(a) frame no : 1, (b) frame no : 10, (c) frame no : 15,
(d) frame no : 20, (e) frame no : 25, (f) frame no : 30

For the second test set, another dim air-vehicle is recorded by using a IR sensor. A typical frame from the recorded sequence is given in Figure 3-17.

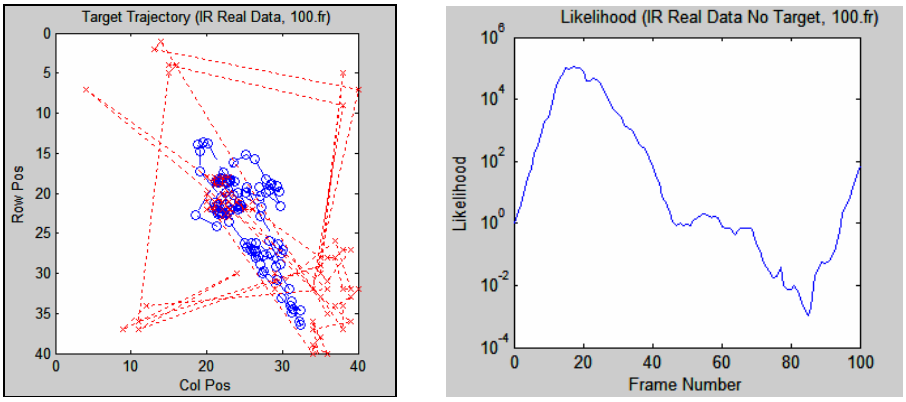


Figure 4-63 Simulation Result of Bayesian Algorithm for IR sensor sequence (a) Target state estimates (b) Evolution of the likelihood Ratio (in log-sclae)

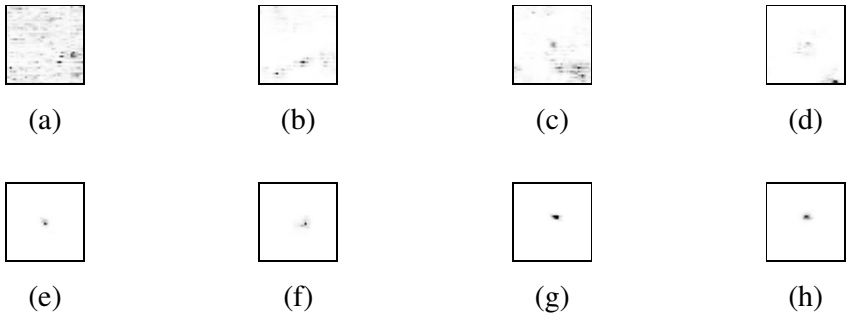


Figure 4-64 The evolution of the target posterior density along the sequence (a) frame no : 1, (b) frame no : 20, (c) frame no : 40, (d) frame no : 60, (e) frame no : 70, (f) frame no : 80, (f) frame no : 90, (g) frame no : 100

In both cases, with day-light sensor sequences and IR sequences, since the noise level of the frames is quite low and the target moves slowly (less than 1 pixel/frame), the algorithm gives a promising performance on the real test data, as well.

4.3 Discussion

The proposed technique for detection of dim targets in image sequences is a quite promising approach which has the potential to be applied to the real-life scenarios. Unlike the existing TBD algorithms, this technique is capable of dealing with very low SNR situations, does not require filter banks and computationally inexpensive. The impressive performance of the Bayesian approach can be obtained for the cases in which the target intensity distribution is Gaussian. For such cases, the Bayesian TBD algorithm is capable of detecting dim target in extremely low-SNR conditions. Another factor improves the performance of Bayesian algorithm is the length of the image sequence. In case of longer sequences, the algorithm detection performance improves, as expected.

CHAPTER 5

COMPARISONS AND CONCLUSIONS

5.1 Summary

In this thesis, dim target detection problem is examined via two promising algorithms. Dynamic programming-based (DP) approach and a Bayesian-formulated method are implemented and tested to compare their performances for detecting dim targets.

DP-based algorithm is a batch processing algorithm, whereas the Bayesian-formulated algorithm is a sequential approach. DP-based algorithm is slightly revised compared to similar methods in the literature by selecting relative target locations as states, novel state transition metrics and utilizing a velocity model parameter. On the other hand, there is some past research on Bayesian formulated dim target detection problem and a recent Bayesian algorithm [17] is selected to test due to promising performance and computational effective property. Following sections present conclusions for the comparison of these two algorithms.

5.2 Conclusions on Dynamic Programming Based Dim Target Detection

The proposed algorithm is tested on some artificial image sequences with different noise levels by using different state transition metrics ($m1$, $m2$, $m3$) and a design parameter, δ, as a target velocity model. Although there is no significant performance difference when utilizing these different metrics, $m3$ should be

preferred due to its normalization property. However, a precise normalization of the path sums for different noise levels could not be obtained, as desired.

There are also cases, where the velocity model parameter, δ, is useful, especially for the targets, moving along a single direction, consistently. By utilizing a proper velocity model parameter, the true target trajectory has been achieved to be marked, even in 0 dB SNR. However, the value for this parameter should be selected carefully, since a mismatch between the possible target speed and the velocity model parameter might cause a failure on detecting dim targets.

After the simulations, it is observed that the proposed DP-based algorithm has mostly marked the true target trajectory only after 5-stage (frame) processing. However, for the noise levels, above $\sigma_N = 7$, the DP detector has failed in 5-stage detection. By utilizing new metrics or longer executions might improve the performance.

The proposed algorithm is also tested on real data sets, which includes an image sequence captured by using a day-light sensor and another sequence via an IR sensor. In both cases, the algorithm has successfully marked the true target trajectory after 5-stage processing.

5.3 Conclusions on Bayesian Formulated Dim Target Detection

Extensive tests on artificial image sequences showed that the Bayesian formulated dim target detection algorithm is not capable of marking the weighted target estimate in early frames. This observation is due to the fact that the algorithm is not able to accumulate likelihood in typical neighborhoods for the fast moving targets. The algorithm usually shows a better performance for slow targets. In other words, it could be concluded that the targets which have sub-pixel motions can also be detected successfully with the aforementioned property of the algorithm.

The tests also showed that for the cases in which the target stays stationary longer periods of time for a detection (as seconds instead of milliseconds), the algorithm results with satisfactory detection results, even for the cases with 0dB SNR levels.

While testing the algorithm for one-target cases, it has been observed that the false alarm rate of the algorithm is relatively low, although the false alarm rate is not quantitatively measured. In other words, it is quite unlikely to observe any false declaration from the algorithm, when there is no target. This conclusion can also be observed by considering the likelihood curves, which do not have any increasing nature for any of the no-target cases.

5.4 Comparison between DTD Methods

Both of these methods give competitive performances on most of the tested data. Bayesian formulated algorithm is quite reluctant to declare any detection for no-target cases, whereas DP-based algorithm easily indicates a possible target trajectory, even there is no target in the image sequences. Improper thresholding of such path scores for DP might cause high false alarm rate.

Bayesian formulated algorithm performs quite promising for the Gaussian-shaped targets, as in the case of Video-6 experiment, whereas DP-based algorithm is usually more successful at one-pixel targets by the help of the velocity model parameter.

Bayesian formulated algorithm requires more frames compared to that of the DP-based algorithm, but still quite successful to detect the target in an acceptable execution time. However, for the case with many frames, DP algorithm requires quite long execution time before giving any results.

Target speed is another important issue to be considered. DP-based algorithm is usually more successful compared to the Bayesian algorithm, for the cases of fast moving targets, more than 1 pixel/frame. However, Bayesian formulated algorithm gives better performance for lower (sub-pixel) target speeds.

In order to summarize the performance of both algorithms for different noise levels, are tabulated in the following table by using the artificial data results.

Table 5.1 Comparison between DP-Based and Bayesian formulated algorithm

DP-based / Bayesian Form.	$\sigma_N = 0^*$	$\sigma_N = 1$	$\sigma_N = 5$	$\sigma_N = 7$	$\sigma_N = 10$	$\sigma_N = 13$
Video-1 ($\delta = 0$)	✓	✓	✓	👎	👎	👎
	✓	✓	👎	👎	👎	👎
Video-1 ($\delta = 15$)	✓	✓	✓	✓	✓	✓
	✓	✓	👎	👎	👎	👎
Video-2	✓	✓	✓	✓	👎	👎
	✓	✓	✓	✓	👎	👎
Video-3	✓	✓	✓	👎	👎	👎
	✓	✓	✓	👎	👎	👎
Video-4	✓	✓	✓	👎	👎	👎
	✓	✓	✓	✓	👎	👎
Video-5	✓	✓	✓	✓	👎	👎
	👎	👎	👎	👎	👎	👎
Video-6	✓	✓	👎	👎	👎	👎
	✓	✓	✓	✓	✓	✓

*Noise-free case

As it can be observed from this table, both algorithms have similar noise performance for Video-2 and Video-3. By the help of the velocity model parameter, DP-based algorithm gives superior results compared to Bayesian formulated algorithm. At Video-4, Bayesian formulated algorithm seems to result in slightly better performance than DP-based algorithm, since it is successful at the noise level $\sigma_N = 7$, where DP-based algorithm was not successful on detecting the dim target. At Video-5 Bayesian formulated algorithm exhibit a relatively poor performance, since the target was one-pixel in size and stationary, where DP-based algorithm could detect the target up $\sigma_N = 7$ noise levels.

The most interesting result is obtained for Video-6 case. Bayesian formulated algorithm is successful for quite high noise levels. In fact, as observed in Section 4.2.1, Bayesian formulated algorithm was successful up noise levels $\sigma_N = 21$, which is a surprising result. This result is due to the fact that Bayesian formulated algorithm is fully optimized for the Gaussian shaped, slow targets.

It can be concluded that the Bayesian algorithm is more preferable than DP-based algorithm, especially with more realistic target models, as in case of Video-6.

5.5 Future Directions

Image sequences consisting of one-pixel target is not realistic. When the targets in real data sets are examined, it has been observed that the targets are small, but not one-pixel size; i.e. they are mostly larger than one pixel with small tails in different directions in sub-pixel resolution. Another set of experiments might be conducted by using more realistic target models in image sequences, including considerations of different target speeds and maneuvering targets.

In order to asses the performance the algorithms in a more reliable way, Receiver Operating Characteristics (ROC) curves for different cases should be obtained. In these curves, the probability of detection and false alarm for these two

approaches could be observed for various algorithm parameters. However, the presented simulation results still give enough insight to understand the performance of these algorithms.

The experiments in this thesis concentrated on single sensor (camera) systems. The re-formulation of the algorithm to incorporate target energies from different sensors should also improve the performance, considerably for any target detection approach.

REFERENCES

[1] H.L. Van Trees, Detection, Estimation, and Modulation Theory, Wiley, 1968.

[2] Bar-Shalom, Y., and Fortmann, T. E., Tracking and Data Association, New York: Academic Press, 1988.

[3] Reid, D. B., "An algorithm for tracking multiple targets", IEEE Transactions on Automatic Control, 24, Dec. 1979, 843-854.

[4] Fernandez M. F., Aridgides A., Bray D., "Detecting and tracking low-observable targets using IR", Proceedings of SPIE, 1305, 1990, 193-206.

[5] Reed, I. S., Gagliardi, R. M., and Shao, H. M., "Application of three-dimensional filtering to moving target detection", IEEE Transactions on Aerospace and Electronic Systems, 19, 1983, 898-905.

[6] Reed, I. S., Gagliardi, R. M., and Stotts, L. B., "Optical moving target detection with 3-D matched filtering", IEEE Transactions on Aerospace and Electronic Systems, 24, July 1988, 327-336.

[7] Pratt, W., Digital Image Processing, New York: Wiley, 1978.

[8] Reed, I. S., Gagliardi R. M., and Stotts L. B., "A recursive moving-target indication algorithm for optical image sequences" IEEE Transactions on Aerospace and Electronic Systems, 26, May 1990, 434-440.

[9] Bellman, R., Dynamic Programming, Princeton, NJ: Princeton University Press,1957.

[10] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., Introduction to Algorithms, 2nd ed., The MIT Press, 2001

[11] Barniv, Y., "Dynamic programming solution for detecting dim moving targets", IEEE Transactions on Aerospace and Electronic Systems, 21, 1985, 144-156.

[12] Barniv, Y., and Kella, O., “Dynamic programming solution for detecting dim moving targets. Part II: Analysis”, IEEE Transactions on Aerospace and Electronic Systems, 23, 1987, 776-788.

[13] Tantaratana, S., and Poor H. V., “Asymptotic efficiencies of truncated sequential tests”, IEEE Transactions on Information Theory, 28, 1982, 911-923.

[14] Blostein, S. D., and Huang, T. S., “Detecting small moving objects in image sequences using sequential hypothesis testing”, IEEE Transactions on Signal Processing, 39, 1991, 1611-1629.

[15] Lindgren, R.G., Taylor, L.A., “Bayesian field tracking”, Proc. SPIE Vol. 1954, Signal and Data Processing of Small Targets 1993, 292-303.

[16] Barlow, C.A., Blackman, S.S. “New Bayesian track-before-detect design and performance study”, Proc. SPIE Vol. 3373, Signal and Data Processing of Small Targets 1998, 181-191.

[17] Tekinalp, S., Alatan A. A., “Efficient Bayesian Track-Before-Detect”, IEEE ICIP 2006, Atlanta.