國防科技學術合作協調小組研究計畫成果報告

點防禦系統對高速超低高目標物之偵追與導引之研究（III）

計畫編號：NSC-91-2623-7-009-003

執行期間：91 年 1 月 1 日至 91 年 12 月 31 日

計畫主持人：林昇甫

共同主持人：林進燈

執行單位：國立交通大學電機與控制工程學系

中華民國 91 年 12 月 31 日

Detection and Tracking of High-Speed-Low-Height Moving Target and Its Guidance Law Study in Point wise Air Defense System (III)

成果報告

點防禦系統對高速超低高目標物之偵追與導引之研究(III)

Professor：Sheng Fuu Lin 林昇甫

NSC-91-2623-7-009-003

Institute of Electrical and Control Engineering

National Chiao Tung University

Hsinchu, Taiwan, Republic of China

Dec 31. 2002

參與計畫人員

相對主持人：金傳文

研究人員　　：林昇甫

林進燈

潘慶原

蒲鶴章

摘 要

預防未來數十年，來自空中的威脅是會避開雷達的敵方目標物，因此，解除這項威脅應是防空系統的發展重點之一。想偵測出企圖躲避雷達而低飛的敵方目標物，除了雷達影像偵測是個可考慮的方法，由於天候的變化不定，因此利用紅外線影像來作影像分析，進而分離出可能的目標。除了估測該目標目前的所在位置外，還記錄到目前所掌握到的整個運動軌跡，並對其前進路徑作預估。取得估測軌跡後，我們還將進行其引導邏輯之研究，並達到引導控制之目的。在去年度中我們完成了，決定差值影像之臨界值，計算多目標物的位置，紀錄目前所掌握目標物的位置，為了使目標物追蹤的結果更為完善，我們嘗試利用影像中物體移動速度不同的特性，來分割出影像中所需目標物而達到追蹤的目的。在今年度我們嘗試了另一種光流估測法，在運算時間和正確率方面得到更好的效果。而在導引方面，我們對於 DOA 的計算和 EID 在 EW 的應用層面提供了一個效用力強且有效率的方法，而最後模擬的結果也顯示出此方法優於其他的方法。

ABSTRACT

The major threat of air attract are thought as missiles in the next decades. Therefore, anti-missile systems are one of key points in the air defense task.

To detect the enemy targets with low height, image detection is an another approach expected using detection radar. Due to weather variousness, infrared images can be used to detect the enemy targets. First, the infrared images are analyzes in this project such that the enemy targets can be separated from the background. It is necessary to estimate the current location of the enemy target. We focus on the automatic target recognition of infrared image and basic guidance law. In the last two year, we calculate the position of multi-target, estimate and tracking of multi-target, estimate the current location of the enemy target, predict the trajectories of moving targets. In order to improve the result of the target detection, we try to use the property of the difference between objects in the image to segment the target that we need. In this year, we proposed another optical flow estimation method and then the results are more efficiency in computation time and rate of success..

From engineering point of view, this project aims to provide a powerful and effective methodology for direction of arrival (DOA) estimation and emitter identification (EID) in electronic warfare (EW) applications, respectively. Capabilities and performances of the proposed scheme have been verified and evaluated with other methods by various examples. Simulation results show that the proposed networks with associated algorithms are superior to other methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Survey

In the recent years, tracking moving objects using an image sequence has been very popular. It can be used for capturing and recognizing moving targets as well as for analyzing object motions, so as to be applied to various applications such as weapon systems, transportation systems, security systems and factory automation. Digital image processing encompasses a broad range of hardware, software, and theoretical underpinnings. The first step in the process is image acquisition-that is, to acquire a digital image. To do so requires an imaging sensor and the capability to digitize the signal produced by the sensor. After a digital image has been obtained, the next step deals with preprocessing that image. The key function of preprocessing is to improve the image in ways that increase the chances for success of the other processed.

The next stage deals with segmentation. Broadly defined, segmentation partitions an input image into its constituent parts or objects. In general, preprocessing deals with techniques for enhancing contrast, removing noise. For example, elementary contours can be derived by using a gradient operator and Laplacian operator [1], and the Hough transform [1] is a well-known method about global processing method, and the image thresholding method, such as simple global thresholding [1] [2], optimal thresholding based on boundary characteristics is often used.

The last stage involves recognition and interpretation. Recognition is the process that assigns a label to an object based on the information provided by its descriptors. When the targets are extracted, some noise will accompany the image. In order to decrease the noise disturbance, the techniques for canceling noise will be used. There are many researches about canceling noise, for example, lowpass filtering [1], median filtering [1], [3], high-boost filtering, derivative filtering and others. These filtering methods are discussed in spatial domain. Beside, the issue of canceling noise is also discussed in frequency domain. The Fourier transform is extracted from the intensity function of pixels in the time domain to generate the phase and the spectrum, which are analyzed to cancel the noise. However, heavy computing tasks to handle complex multiplication and additions are required. Then the method in spatial domain is chosen in this system.

To detect and track the high-speed-low-height moving target, image detection is another approach except using detection radar. And the weather is changing all the time, so the light is an important key. Infrared images can be used to detect the enemy targets. The infrared images can be separated from the background because of the target's temperature.

In the surveillance and reconnaissance communities, the direction of arrival (DOA, sometimes referred to as angle of arrival, AOA) information and emitter identification (EID) capability are extremely important problems, especially in the electronic warfare (EW) field. To cope with the drawbacks encountered in the RBFN, while still keeping its advantages, a new DOA estimation algorithm with a neural fuzzy network (NFN) is developed.

1.2 Organization of the Report

The following is a brief description of the organization of this report. In Chapter 2, some image processing techniques including the image thresholding, median filter, region growing and image difference. Furthermore, the optical flow method is introduced. In order to improve the efficiency of the detection, we use another method based on the velocity field in Chapter 3. Experiments and discussions are given in Chapter 4. In Chapter 5 we discuss the direction of arrival estimation based on phase differences using neural fuzzy network. Finally, the conclusions are given in Chapter 6.

Chapter 2

Image Processing Techniques

In this chapter, image processing techniques are introduced to obtain the feature points of targets, such as image thresholding method [1] [2], and median filter [1], [4], [3]. In this report, the image thresholding method is applied to extract the feature points of the targets. In Section 2.1, image processing methods are introduced to find the feature point of the target. Section 2.2 shows previous targets detection techniques. Section 2.3 describes optical flow methods.

2.1 Image Processing Techniques

To derive these feature points, several image processing methods are employed in this section. Image thresholding is introduced in Section 2.1.1. Section 2.1.2 then expresses these concepts of median filter. Low pass filter is illustrated in Section 2.1.3. Section 2.1.4 presents region growing by pixel aggregation. Finally, Section 2.1.5 introduces image difference.

2.1.1 Image Thresholding

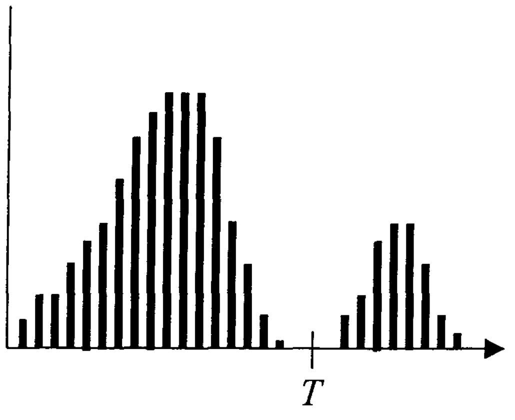
Image thresholding is one of the most important approaches to image segmentation. Suppose that the gray-level histogram shown in Fig. 2.1. (a) corresponds to an image, $f(x, y)$, composed of light objects on a darkling background, in such a way that objects and background pixels have gray levels grouped into two dominant modes. One obvious way to extract the objects from the background is to select a threshold T that separates these modes. Then, any point (x, y) for which $f(x, y)>T$ is called an object point; otherwise, the point is called a background point. Fig. 2.1. (b) shows a slightly more general case of this approach. Here, three dominant modes characterize the image histogram (for example, two types of light objects on a dark background). The same basic approach classifies a point (x, y) as belonging to one object class $T_1< f(x, y)\leqq T_2$, to the other object class if $f(x, y)> T_2$, and to the background if $f(x, y)\leqq T_1$.

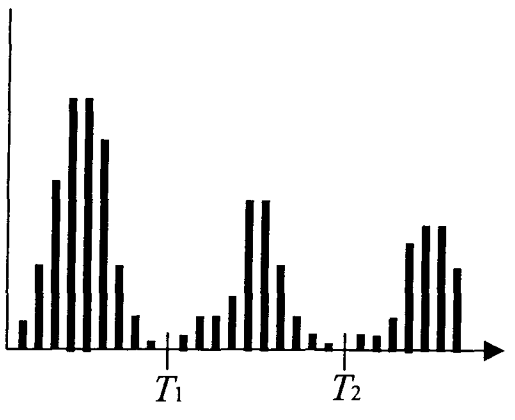
Thresholding may be viewed as an operation that involves tests against a function T of the form

$$T=T(x, y, p(x, y), f(x, y)), \tag{2.1}$$

where $f(x, y)$ is the gray level of point (x, y), and $p(x, y)$ denotes some local property of this point. A thresholded image $g(x, y)$ is defined as

$$\begin{aligned} g(x, y)&=1, && \text{if } f(x, y)>T \\ g(x, y)&=0, && \text{if } f(x, y)\leqq T \end{aligned}$$



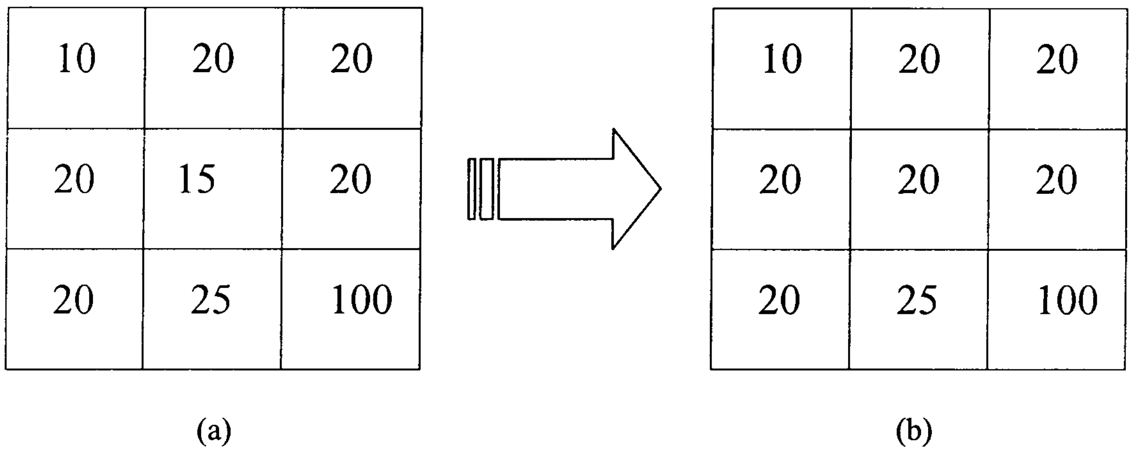


(a) (b)

Fig. 2.1. Gray-level histograms that can be partitioned. (a) A single threshold. (b) Multiple thresholds.

2.1.2 Median Filter

As indicated in Section 2.1.1, the image threshold method in a dynamic imaging problem has the tendency to cancel all background regions, leaving only image elements that correspond to noise and to the moving object. The noise problem can be handled by a filtering approach [1]. As our objective is to achieve noise reduction rather than blurring, **median filters** [1] are adopted. That is the gray level of each pixel is replaced by the median of the gray levels in a neighborhood of that pixel, instead of by the average. This method is particularly effective when the noise pattern consists of strong, spikelike components and the characteristic to be preserved is edge sharpness. The median m of a set of values is such that half the values in the set are less than *m* and half are greater than *m*. In order to perform median filtering in neighborhood of a pixel, we first sort the values of the pixels and its neighbors, determine the median, and assign this value to the pixel. For example, in a 3×3 neighborhood the median is the 5th largest value, in a 5×5 neighborhood the 13th largest value, and so on. When several values in a neighborhood are the same, all equal values have to be grouped. For example, suppose that a 3×3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus the principal function of median filtering is to force points with distinct intensities to be more like their neighbors, actually eliminating intensity spikes that appear isolated in the area of the filter mask. Fig. 2.2. (a) shows a original 3×3 neighborhood and Fig. 2.2. (b) shows the results of median filtering.



10	20	20
20	15	20
20	25	100

(a)

10	20	20
20	20	20
20	25	100

(b)

Fig. 2.2. Illustration of median filter method. (a) Original 3×3 neighborhood. (b) Resulting 3×3 neighborhood.

2.1.3 Low Pass Filter

Another method to delete the noise is low pass filter, and to realize the shape of the need of low pass filter, Fig. 2.3. shows all the parameter must be positive, and the final response is the sum of the 3×3 neighborhood. But it will make the value is too large to be continue next steps, so we can let the value divide nine, like a mask, and we will get the mean value, Fig. 2.3. (b) shows the result.

1	1	1
1	1	1
1	1	1

(a)

1/9 ×

1	1	1
1	1	1
1	1	1

(b)

Fig. 2.3. The mask is used to illustrate low pass filter method. (a) The sum of 3×3 neighborhood. (b) The result mean value of the 3×3 neighborhood sum.

2.1.4 Region Growing by Pixel Aggregation

Region growing, as named, is a procedure that groups pixels or subregions into larger regions. The simplest of these approaches is pixel aggregation, which starts with a set of "seed" points and from these grow regions by appending to each seed point those neighboring pixels that have similar properties such as gray level, texture, color. To illustrate this procedure let us consider Fig. 2.4. (a) where the numbers inside the cells represent gray level values. Let the points with coordinates (3,2) and (3,4) be used as seeds [7].

Using two starting points results in a segmentation consisting of, at most, two regions: R1 associated with seed (3,2) and R2 associated with seed (3,4). The property to be used include a pixel in either region is that the absolute difference between the gray level of that pixel and the gray level of the seed be less than a threshold *T*. Any pixel that satisfies this property simultaneously for both seeds is (arbitrarily) assigned to region R1. Fig. 2.4. (b) shows the result obtained using *T*=3. In this case, the segmentation consists of two regions, where the points in R1 are denoted *a*'s and the points in R2 by *b*'s. Note that any starting point in either of these two resulting regions would yield the same result. However, choosing *T*=8, would result in a single region, as Fig. 2.4. (c) shows.

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 0 & 5 & 6 & 7 \\ 2 & 1 & 1 & 5 & 8 & 7 \\ 3 & 0 & 1 & 6 & 7 & 7 \\ 4 & 2 & 0 & 7 & 6 & 6 \\ 5 & 0 & 1 & 5 & 6 & 5 \end{array}$$

(a)

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & a & a & b & b & b \\ 2 & a & a & b & b & b \\ 3 & a & a & b & b & b \\ 4 & a & a & b & b & b \\ 5 & a & a & b & b & b \end{array}$$

(b)

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & a & a & a & a & a \\ 2 & a & a & a & a & a \\ 3 & a & a & a & a & a \\ 4 & a & a & a & a & a \\ 5 & a & a & a & a & a \end{array}$$

(c)

Fig. 2.4. Example of region growing using known starting points. (a) Original image array. (b) Segmentation result using an absolute difference of less than 3 between intensity levels. R1 are denoted *a*'s and R2 are denoted *b*'s. (c) Result using an absolute difference of less 8.

2.1.5 Image Difference

In this subsection, we assume that the camera is fixed, because our infrared image is obtained from an immovable camera, the object must move fast then background. It has more variation. So we can define the difference image $Dif(x,y)$ as,

$$Dif(x,y) = image2(x,y) - image1(x,y),$$

where $image1(x,y)$ means the gray value of the former image, and $image2(x,y)$ means the gray value of the next image, then the image thresholding method can be used to get a binary image. We can take that binary image to another processing to get the what we need. Because we can observe the feature points in the result image easily, image difference is the basic step in the process of the image sequence. If the camera is fixed, we can use the image difference method to find the target. Because the background is fixed and the target is the only moving object, after the image difference method processing, we could find the target. Then we let this $image2(x,y)$ be a binary image. The methodology is :

$$\text{If } Dif(x,y) < 20, \text{ then } Dif(x,y) = 0, \text{ else } Dif(x,y) = 255.$$

In other words, if the difference is too small, we could delete it and remain others. After this, we can take it to multiply $image2(x,y)$. The methodology is as the follows :

$$Result(x,y) = image2(x,y) \times Dif(x,y),$$

$$\text{if } Result(x,y) \neq 0, \text{ then } Result(x,y) = 255.$$

In fact, the moving target can be detected from image *Result* (x, y). If the camera is not fixed, it is necessary to improve this method or use other methods.

2.2 Previous Targets Detection Techniques

To detect the enemy targets with low height, image detection is another approach except using detection radar. Due to weather constantly changing, infrared images can be used to detect the enemy targets. The infrared images are analyzed in this section such that the enemy targets can be separated from the background. Because the camera is movable, the change of background may be more than the change of the object. The method described before is not suit to be used in this situation. Unfortunately, the camera is movable in the great part of the situation. So we propose another method to solve this problem. It can be explained as the following steps：

1. Find the lightest point in the figure.
2. Region growing.
3. Reduce the searching range.
4. Decide the video section.
5. Target recognition.

2.2.1 Find the lightest point in the figure

Because of the images we used are infrared images, the object almost be the lightest point. So we can search each pixel of the image in turn to find the largest gray value be the center of the object. But because of the image convert, the radiant heat from the ground, the camera lens, the other noise, we forsake the figure edge. Our searching range is defined,

$$20 \le SRW \le image \quad width - 20,$$
$$25 \le SRH \le image \quad height - 30,$$

where SRW is the width of the searching range, and SRH is the height of the searching range.

2.2.2 Region growing

As its name implies, region growing is a procedure that groups pixels or subregions into larger regions. The simplest of these approaches is pixel aggregation, which starts with a set of "seed" points and from these grows regions by appending to each seed point those neighboring pixels that have similar properties (such as gray level, texture, color). After finding the lightest point in the image, we must decide the size of the object frame. Now we let the lightest point be the seed searching its neighboring pixels. When we find the pixels which gray value is less 10 than the gray value of the lightest point, they are defined as object frame edges. Because of sometimes the object is not obviously and may have noise interference, the object frame may be very large. We limit the range of the object frame not larger than 1600 pixels.

2.2.3 Reduce the searching range

In order to reduce the compute time and mistake, we can reduce the searching range. So when we get the position of the object by searching the lightest point in the whole image, spread 20 pixels in all direction. Let the range be the searching range next time. It is,

$$p\_x - 20 \le SSRW \le p\_x + 20,$$
$$p\_y - 20 \le SSRH \le p\_y + 20,$$

where (p_x, p_y) is the position of the object, $SSRW$ is the width of the small searching range and $SSRH$ is the height of the small searching range.

2.2.4 Decide the video section

When reducing the searching range, we have two problems must be solved. First, we have to know where is the video section. If the next image is not the same section, the object may be not in the small range. Second, if the position is not the correct pcsition of the object, it will be wrong in the next time. When these two problems are occurred, we have to change to use the whole searching. For the first problem, we can use mean value to decide whether it is a broken point of the video section. It means:

$$mean = \frac{1}{height \times width} \sum_{i=0}^{height} \sum_{j=0}^{width} image(i, j),$$

if the mean of the image is differ from the mean of the next image larger than 5, we decide it is the broken point of the video section. Fig. 2.5. shows that we do not know it is a broken point of the video section, and Fig. 2.6. shows that we know it is a broken point of the video section then change to use the whole image range.

2.2.5 Target Recognition

Sometimes object is not clearly and the noise may interference our algorithm. So sometimes it is not the correct position of the object. We have to judge whether it is the object, otherwise it will be wrong in the next image. We obtain the series of images from the original video sequence per 1/30 sec. we obtain two test videos, respectively 10144 images and 1105 images. Fig. 2.7. and Fig.2.8. are shown some images of the results of the experiment. By this method, the rate of the success is higher than 98%. Because the target is not always the lightest point, this method is easy to be influenced by brightness. So we will propose another method to improve the rate of success in the next chapter.

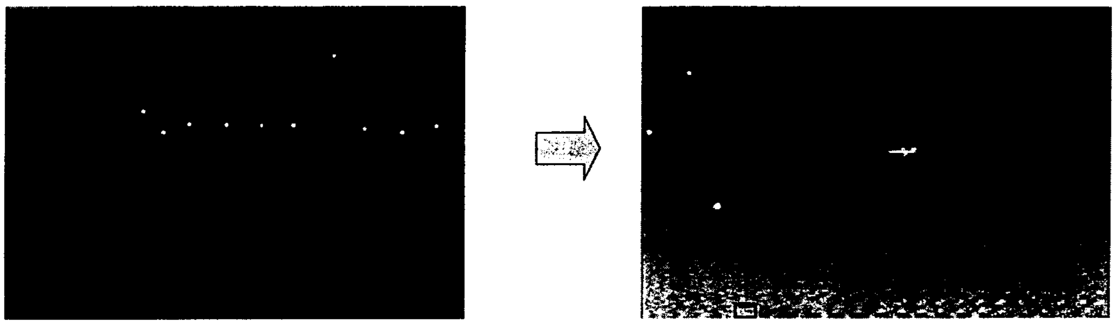


Fig. 2.5. Only using the small range searching.

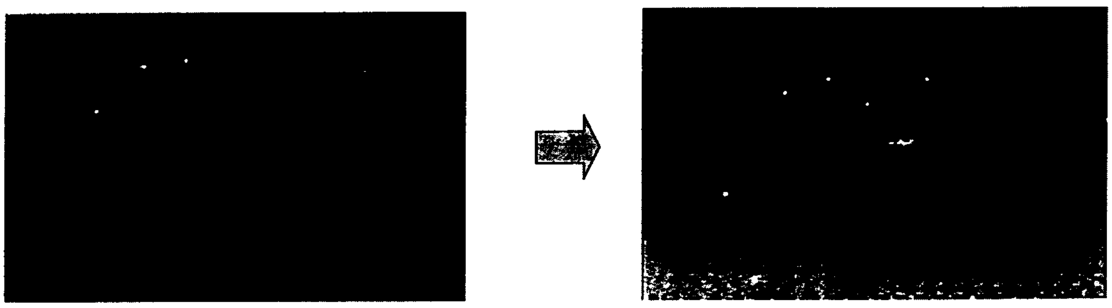
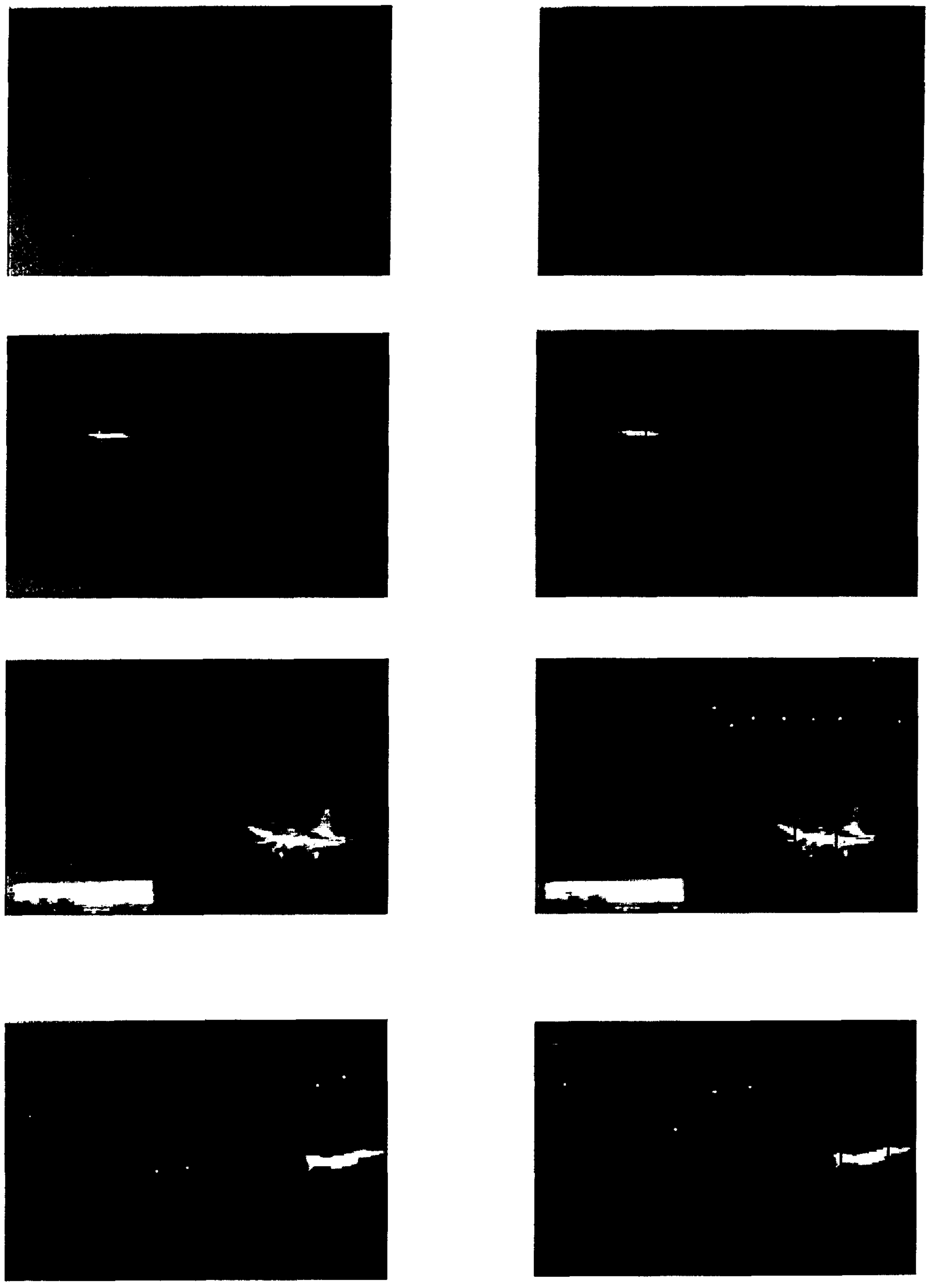


Fig. 2.6. Decide it is a broken point of the video section then change to use the whole image range searching.



(a) (b)

Fig. 2.7. Some final image results. (a) Original images. (b) Results using lightest point method.

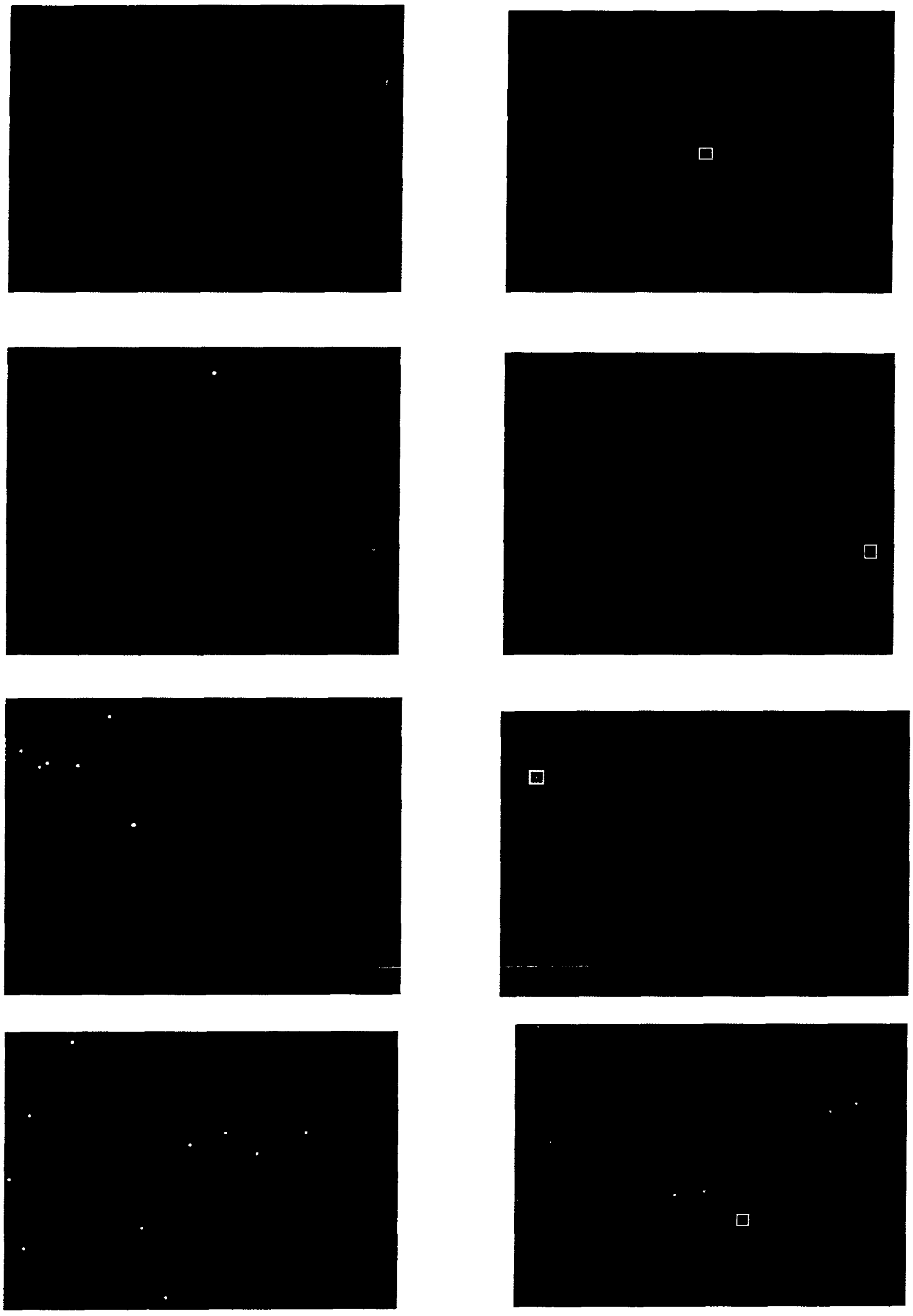

(a) (b)

Fig. 2.8. Some final image results. (a) Original images. (b) Results using lightest point method.

2.3 Optical Flow Methods

Optical flow implies floating fluid. The vector field formed by these motions is caused by the orbit that all the pixels of images moving in the space-time. Since the dynamic object that this report will discuss is not limited in some particular objects, optical flow should be calculated to estimate the speed of images plane movement. Section 2.3.1 introduces the gradient based constraint for optical estimation. Section 2.3.2 illustrates smoothness constraints. Section 2.3.3 describes gradient estimation. Section 2.3.4 introduces minimization. Section 2.3.5 deals with choice of iterative scheme.

2.3.1 The gradient based constraint for optical estimation

Optical flow is a velocity field associated with brightness changes in the image. This suggests an assumption often made in methods for optical flow estimation, the brightness conservation assumption, which states that brightness of an image of any point on the object is invariant under motion. Optical flow suited in rigid body motion analysis and nonrigid body motion analysis.

Let $E(x, y, t)$ be the image brightness at the point $\boldsymbol{x} = (x, y)$ in the image plane at time t and let $\boldsymbol{u} = (u, v)$ be a projection of velocity vector of this point. After δt, point $\boldsymbol{x}$ will move to a new position $\boldsymbol{x} + \boldsymbol{u}\,\delta t$. Since the brightness of a particular point in the pattern is constant, so that

$$E(x,y,t) \cong E(x+\delta x, y+\delta y, t+\delta t). \tag{2.2}$$

By Taylor series expansion, we get

$$E(x+\delta x, y+\delta y, t+\delta t) = E(x,y,t) + \delta x\frac{\partial E}{\partial x} + \delta y\frac{\partial E}{\partial y} + \delta t\frac{\partial E}{\partial t}. \tag{2.3}$$

Assuming an infinitesimal time interval, we end up with the following equation

$$\frac{\partial E}{\partial x}u + \frac{\partial E}{\partial y}v + \frac{\partial E}{\partial t} = 0, \tag{2.4}$$

which is the major optical flow constraint. It involves two unknowns at each point of the image plane: the optical flow components u and v. if we rewrite this equation using the gradient notation

$$(\nabla \dot{A})^T \boldsymbol{u} + E_t = 0. \tag{2.5}$$

It is obvious that the optical flow field cannot be found from this equation only, but some additional assumptions have to be made, or determine both optical flow components.

Gradient-based methods are generally relatively simple to implement, efficient to compute, and can produce surprisingly accurate optical fields. We will discuss two additional assumptions to determine both components of the optical flow field in the other sections.

2.3.2 Smoothness constraints

In this case neighboring points on the object have similar velocities, their velocities differing only slight almost everywhere. Horn and Schunck [41] were first to make this assumption and exploit it for determining an optical flow. If every points of the brightness pattern can move independently, there is little hope of recovering the velocities.

One way to express the additional constraint is to limit the different between the flow velocity at a point and the average velocity over a small neighborhood containing the point. Equivalently can minimize the sum of the squares of the Laplacians of the x-component and y-component of the flow. The Laplacians of u and v are defined as

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad \text{and} \quad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}. \tag{2.6}$$

They transformed the optical flow estimation into an optimization problem involving combination of the two criteria: the error in the image brightness changes measurement

$$\dot{a}_b = E_x u + E_y v + E_t \tag{2.7}$$

and the quantity reflecting a non-smoothness of the velocity field

$$\dot{a}_c^2 = (\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2. \tag{2.8}$$

2.3.3 Gradient estimation

It is important that the estimates of E_x, E_y and E_t be consistent. They should all refer to the same point in the image at the same time. We will use a set that gives us estimate of E_x, E_y and E_t at a point in the center of a cube formed by eight measurements. The relationship in space and time between these measurements

$$E_x \approx 1/4\{E_{i,j+1,t} - E_{i,j,t} + E_{i+1,j+1,t} - E_{i+1,j,t} + E_{i,j+1,t+1} - E_{i,j,t+1} + E_{i+1,j+1,t+1} - E_{i+1,j,t+1}\},$$

$$E_y \approx 1/4\{E_{i+1,j,t} - E_{i,j,t} + E_{i+1,j+1,t} - E_{i,j+1,t} + E_{i+1,j,t+1} - E_{i,j,t+1} + E_{i,+1j+1,t+1} - E_{i,j+1,t+1}\},$$

and

$$E_t \approx 1/4\{E_{i,j,t+1} - E_{i,j,t} + E_{i+1,j+1,t} - E_{i+1,j,t} + E_{i,j+1,t+1} - E_{i,j+1,t} + E_{i+1,j+1,t+1} - E_{i+1,j+1,t}\}$$

(2.9)

is shown in Fig. 2.8. The unit of length is the grid spacing interval in each image frame and the unit of time is the image frame sampling period.

It is difficult to solve the Laplacians of u and v. Horn and Schunck [41] used the approximation defined by the 3x3 mask as shown in Fig. 2.9. yielding

$$\nabla^2 u \approx \bar{u} - u \quad \text{and} \quad \nabla^2 v \approx \bar{v} - v, \tag{2.10}$$

where the average values $\hat{u}$ and $\bar{v}$ are calculated using the following formulate:

$$\bar{u}_{i,j,k} = \frac{1}{6}\{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\}$$
$$+\frac{1}{12}\{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\}$$

and

$$\bar{v}_{i,j,k} = \frac{1}{6}\{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\}$$
$$+\frac{1}{12}\{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\}. \tag{2.11}$$

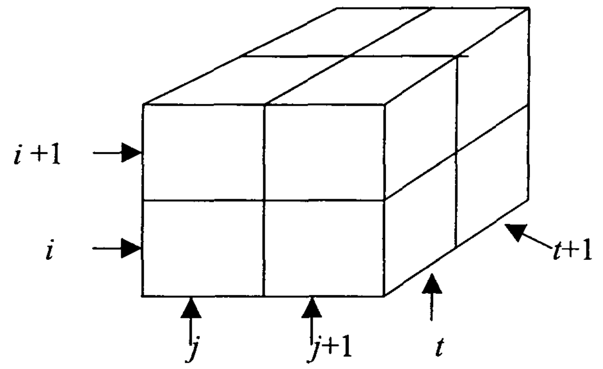


Fig. 2.8. The three derivatives of image brightness at the center of the cube are each estimated from the average of first differences along four parallel edges of the cube.

1/12	1/6	1/12
1/6	-1	1/6
1/12	1/6	1/12

Fig. 2.9. The Laplacian estimation mask

2.3.4 Minimization

Since the input image is corrupted by noise and quantization error, we expect ε_b to be indentically zero. This quantity would have a magnitude proportional to the noise in the measurement, therefore the weighting factor α^2 in the sum should be chosen equal to the estimate of the noise variance in the image.

Let the total error to be minimized be

$$\varepsilon^2 = \alpha^2 \varepsilon_c^2 + \varepsilon_b^2 . \tag{2.12}$$

The minimization is to be accomplished by finding suitable values for the optical flow velocity (u, v). This yields two equations for each point of the image,

$$u = \bar{u} - E_x\left[E_x \bar{u} + E_y \bar{v} + E_t\right] \Big/ \left(a'^2 + E_x^2 + E_y^2\right),$$
$$v = \bar{v} - E_y\left[E_x \bar{u} + E_y \bar{v} + E_t\right] \Big/ \left(a'^2 + E_x^2 + E_y^2\right). \tag{2.13}$$

When the brightness gradient is zero, the velocity estimates will be simply averaged of the neighboring velocity estimates as shown in Eq. (2.13).

The minimization is accomplished by finding suitable values for the optical flow velocity (u, v). Using the calculus of variation we obtain

$$\frac{\partial \ddot{a}^2}{\partial u} = -2a'^2(\bar{u} - u) + 2(E_x u + E_y v + E_t)E_x,$$
$$\frac{\partial \ddot{a}^2}{\partial v} = -2a'^2(\bar{v} - v) + 2(E_x u + E_y v + E_t)E_y.$$

Using the approximation of Laplacian, we obtain

$$(a'^2 + E_x^2)u + E_x E_y v = (a'^2 \bar{u} - E_x E_t), \tag{2.14}$$

$$E_x E_y u + (a'^2 + E_y^2)v = (a'^2 \bar{v} - E_y E_t). \tag{2.15}$$

Eqs. (2.14) and (2.15) can be rewritten as the following form,

$$(a'^2 + E_x^2 + E_y^2)(u - \bar{u}) = -E_x[E_x \bar{u} + E_y \bar{v} + E_t], \tag{2.16}$$

$$(a'^2 + E_x^2 + E_y^2)(v - \bar{v}) = -E_y[E_x \bar{u} + E_y \bar{v} + E_t]. \tag{2.17}$$

Dividing both sides of Eqs. (2.16) and (2.17) by $(a'^2 + E_x^2 + E_y^2)$, we obtain

$$u = \bar{u} - E_x[E_x \bar{u} + E_y \bar{v} + E_t]/(a'^2 + E_x^2 + E_y^2), \tag{2.18}$$

$$v = \bar{v} - E_y[E_x \bar{u} + E_y \bar{v} + E_t]/(a'^2 + E_x^2 + E_y^2). \tag{2.19}$$

2.3.5 Choice of iterative scheme

How the iterations are to be interlaced with the time steps. On the other hand, it could iterate until the solution has stabilized before advancing to the next image frame. On the other hand, if the optical flow found for the previous frame was used as the initial estimate, a sufficiently precise optical flow estimate for the next frame was obtained may need only little iterations per time-step. Number of iterations depending on close the initial estimate was to the correct optical flow.

Chapter 3

Target Detection

From Chapter 2, we can find the target that we need from infrared image sequence, and from this chapter, another algorithm will be proposed to detect the target based on the velocity estimation. This algorithm would be more efficient and practical in target detection. In this chapter, we present gradient based constraint and two principle methods for optical flow estimation and the thus describe the multiresolution optical. Section 3.1 deals with basic definitions of optical flow. Section 3.2 covers optical flow estimation. Section 3.3 shows long image sequence analysis. Section 3.4 introduces mutiresolution optical flow estimation. Section 3.5 illustrates detection of the target. Finally, Section 3.6 gives a summarization.

3.1 Basic Definitions of Optical Flow

Motion field is a 3D field of object velocities at each point of space. The 3D motion of objects in a time varying scene is completely defined by the motion field.

One of the aims of the motion recovery process is to reconstruct the motion field of the scene.

Image flow is the visible portion of the 2D projection of the motion field on an image plane. We maybe obtain an image flow as an intermediate result in the 3D-motion estimation process, and then try to recover the 3D-motion field from this 2D projection. However, this seems to be impossible without a priori knowledge on the motion field. Instead, we extract an optical flow, which is a 2D field of velocities associated with the variation of brightness pattern of the image.

The following two examples help us to understand the difference between an image flow and an optical flow. The first one is a uniformly painted ball rotating around its center in some way. In this case the image flow is nonzero for every point of every point of the ball projection on the image plane, while the optical flow is zero, since the image brightness does not change at all. The second example is a stationary scene with moving light source. Here the situation is exactly opposite; the optical flow is non-zero due to intensity changes in the images, whereas absence of motion causes zero image flow. For motion recovery, it is usually assumed that the optical flow and the image flow are close enough to be used interchangeably. In most situations this approximation is quite reasonable, but one should remember the optical flow are significantly different to the image flow [56].

3.2 Optical Flow Estimation

Optical flow is the apparent velocities of movement of brightness patterns in an image. Optical flow can arise from relative motion of objects and the viewer.

Discontinuities in the optical flow are important cues for image segmentation to obtain motion objects with different velocities.

The gradient-based approach we introduced in section 2.3 assumes brightness conservation under motion, with the formula

$$E(x,y,t) \cong E(x+\delta x, y+\delta y, t+\delta t), \tag{3.1}$$

where $E(x, y, t)$ be the image brightness at the point $x = (x, y)$ in the image plane at time t, after δt ,point x will move to a new position $(x+\delta x, y+\delta y)$.

This motion equation provides one constraint for the unknown local components of the velocity vector. The velocity field at each image pixel has x-coordinate and y-coordinate components while the change in image brightness at a point in the image plane due to motion. The optical flow cannot be computed without introducing additional constraints. The additional assumptions are made in order to obtain a well-determined system of equations to yield both components of the velocity vector at each location on the image. The general assumptions for estimating optical flow include one of the following:

a. Optical flow is smooth almost everywhere.

b. Optical flow is constant over an entire segment in the image.

c. Optical flow is the result of restricted motion; for example, planar motion.

Horn and Schunck [41] assumed that the apparent velocity of the brightness pattern varies smoothly almost everywhere in the image. An iterative method for solving the resulting equation was developed as

$$u^{n+1} = \bar{u}^n - E_x[E_x\bar{u}^n + E_y\bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2)$$

and

$$v^{n+1} = \bar{v}^n - E_y[E_x\bar{u}^n + E_y\bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2), \qquad (3.2)$$

,,,

where n denotes the iteration number, u^0 and v^0 denote initial velocity estimates, and $\bar{u}^n$ and $\bar{v}^n$ denote neighborhood average of u^n and v^n, E_x, E_y and E_t are three partial derivatives of image brightness, α is a weighting factor.

To avoid variations in brightness due to shading effects, they initially assumed that the surface is flat. They further assumed that the incident illumination is uniformly across the surface, assumed initially the reflectance varies smoothly, and has no spatial discontinuities. A needle diagram is then provided visual confirmation of confirmation of the solution.

3.3 Long Image Sequence Analysis

Using more than two Frames to estimate optical flow may improves the accuracy of the computed optical flow. Methods for estimating local image velocities with large temporal regions have been proposed. These methods assume that motion should remain uniform in the analysis sequence. The major difficulty in increasing the size of the spatial region for analysis is the possibility that larger regions will include more than one single motion. Thus the method almost assumes motion constancy over several successive frames in the analyzed sequence.

In generally, the sources of error generated in most flow methods are:

a. Noise sensitivity, which will generate false matches.

b. Quantization, which contributes significantly to the errors, generated as analysis of the flow information proceeds.

Kenner and Pong [61] discussed much evidence to support the validity of the use of optical flow as a basis for many computational vision tasks. However, optical flow estimation methods are problematic due to inherent errors in the computational methods involved in the processing. Many of the problems associated with the analysis of image flow can be alleviated if information extracted from a long sequence of images is utilized as a basis for the derivation of the flow information, rather then simple between frame processing. Combine features from several sequential images, with each pair of successive separated by a short time interval. Features are then associated according to the object feature that generated hem, and the resulting point lists are then analyzed to determine the long sequence flow information. Since the long sequence flow overcomes many of the effects of noise and quantization errors.

3.4 Multiresolution Optical Flow Estimation

Due to the different nature of optical flow, standard modeling does not hold for large displacements. To circumvent the problem, we consider an increment estimation of the flow field captured by the optical flow multiresolution setup. The multiresolution framework involves a pyramidal decomposition of the image data. Pyramid is built by using multiple copies of image. Each level in the pyramid is 1/4 size of previous level. We shall assume to work at a given resolution structure.

However, one has to keep in mind that the expression and computations are meant to reproduce at each resolution level according to a coarse to fine strategy shown in Fig. 3.1. Embedded into a multiresolution coarse-to-fine strategy, this incremental approach allows estimating large velocities. This method includes three steps:

Step 1. Coarsen the original image as shown in Fig. 3.2.

Step 2. Optical flow estimation started in the low-resolution level image.

Step 3. The optical flow found for the previous level was used as the initial estimate shown in Fig. 3.3.

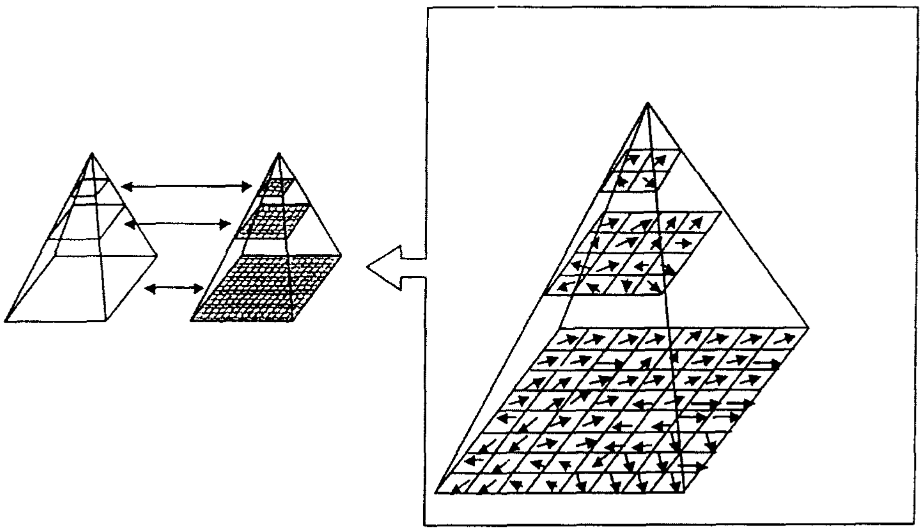


Fig. 3.1. Multiresolution structure.

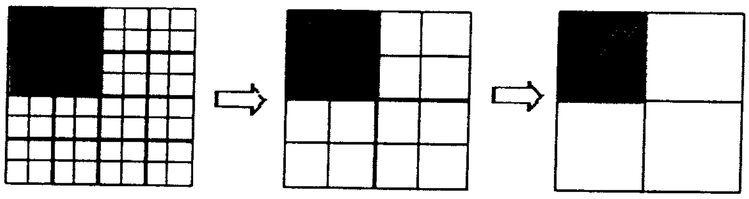


Fig. 3.2. Four pixels are averaged into one pixel.

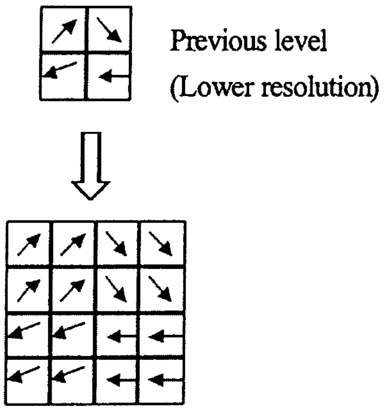


Fig. 3.3. Optical flow found in the previous level.

3.4.1 Performance comparison from theory

Compare Horn-Schunck methods used image pyramid with original methods from the theory. From Eq. (3.2), we can know it has $13 \times n^2$ multiplications and $38 \times n^2$ additions with image size $n \times n$. After using image multiresolution, it have $3.5 \times n^2$ multiplications and $41/4 \times n^2$ additions. Form Table 3.1, it is obvious that used image multiresolution spent less time.

From previous SSD correlation-based approach, we assume search window w_s size 5x5, 4x4 and 3x3. A window w_p of size 3x3. Table 3.2 shows the number of multiplications and additions with SSD method. Hence, the proposed method has better performance in computation time. Table 3.3 shows the number of pixels processed in proposed method and SSD method.

Table 3.1 The number of multiplication and addition.

	Horn-Schunck method	Horn-Schunck method with multiresolution
The number of multiplications	$13 \times n^2$	$3.5 \times n^2$
The number of additions	$38 \times n^2$	$41/4 \times n^2$

Table 3.2 The number of multiplications and additions with SSD method.

	w_s (with size 3x3)	w_s (with size 4x4)	w_s (with size 5x5)
The number of multiplications	$81 \times n^2$	$144 \times n^2$	$225 \times n^2$
The number of additions	$162 \times n^2$	$288 \times n^2$	$450 \times n^2$

Table 3.3 The number of pixels processed.

	The SSD method	The proposed method
The number of pixels	n^2	$0.25 \times n^2$

3.4.2 Performance comparison from experiment

This section test the relative error and time of Horn-Schunck method for single-resolution and multiresolution strategies with image size 640x480. Fig. 3.4. is an artificial image in which the whole image translates with $(u, v)=(1, 1)$.

The process time and relative error of Horn-Schunck method with single resolution strategies are compared in Table 3.4. The differences are more obvious with more iterations. It is apparent that the multiresolution has better performance than the single resolution.

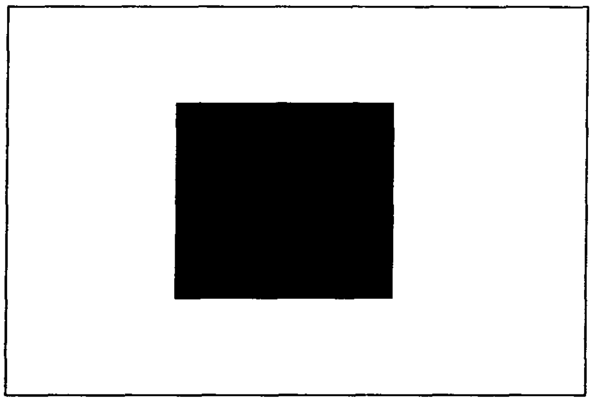


Fig. 3.4. An artificial image(640x480).

3.5 Detection Of The Target

Fig. 3.5. is shown some images of the series video. From above statements, we can calculate velocity field of the image by gradient-based approach method. And we can get lots of information from the velocity field. So we propose this method to solve

Table 3.4 Process time and relative error of Horn-Schunck method for single-resolution and multiresolution strategies.

The number of iterations	Single resolution		Multiresolution	
	Relative error	Time (sec)	Relative error	Time (sec)
10	25.1500	4.72	17	4.01
20	5.80	8.3	4.45	6.57
40	0.8966	16.21	0.69	10.67
60	0.2480	29.48	0.21	19.53
80	0.1482	40.60	0.132	31.50
100	4.476E-02	51.65	4.24E-02	38.37

this problem, it can be explained as the following steps：

1. Get the image intensity.
2. Preprocessing.
3. Compute velocity field.
4. Find the target from velocity field.
5. Check the computing range.
6. Decide the target is real or not.

Section 3.5.1 introduces image intensity. Section 3.5.2 illustrates the preprocessing techniques pyramidal image analysis. Section 3.5.3 deals with velocity field. Section 3.5.4 shows computing range. Finally, Section 3.5.5 covers target recognition, according one decision rule to recognize the target.

3.5.1 Image intensity

The images from this infrared sequence are RGB level, and we must get the pixel value to find the correlation between the first image and the second image, so the first step is to get the image intensity. Then in order to make this method wok with real-time, it is very important to decide the correlation area to make this method more efficiently. The intensity can be expressed as follows：

$$Intensity(x,y) = \tfrac{1}{3}[image(x,y) + image(x+1,y) + image(x+2,y)]. \qquad (3.3)$$

3.5.2 Image pyramid

The preprocessing techniques include pyramidal image analysis and the median filter operation. An image pyramid is one kind of multiple resolution which can observe the same image with different resolutions. Therefore, Multiple resolution processing has been proved very efficient in the application of image analysis such as region segmentation and edge detection. The multiple resolution analysis transforms an image into a sequence of images of pyramidal structure, denoted by $F_k, k = 0, 1, \cdots, n$. F_{k+1} is four times the size of F_k. $f_k(i,j)$ is the gray value of the point (i, j) of F_k and is produced by four near points $f_{k+1}(i,j)$ around F_{k+1}, that is

$$\begin{aligned} f_k(i,j) = [&f_{k+1}(2i,2j) + f_{k+1}(2i,2j+1) + \\ &f_{k+1}(2i+1,2j) + f_{k+1}(2i+1,2j+1)] \times 0.25. \end{aligned} \qquad (3.4)$$

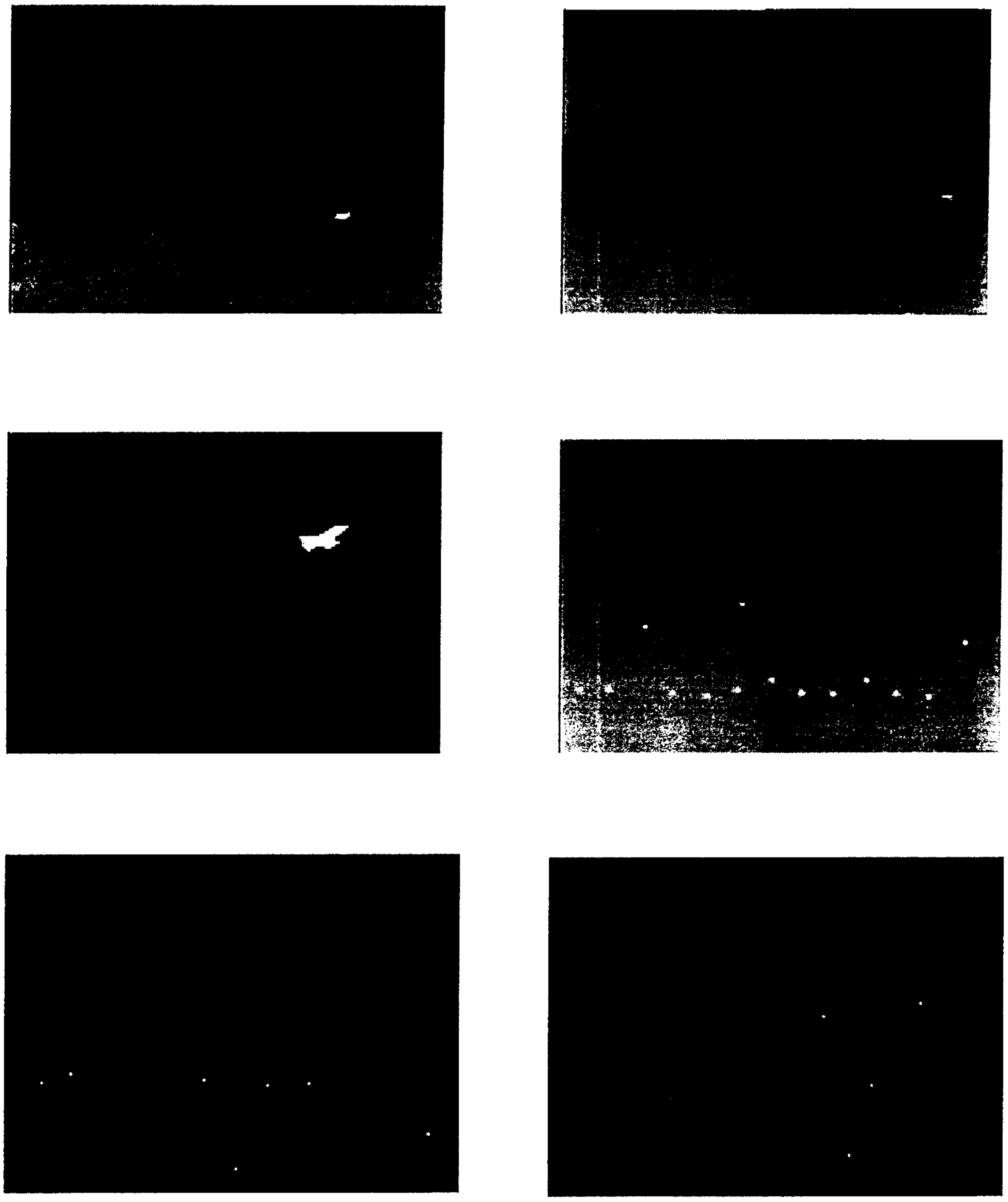


Fig. 3.5. Some images in video.

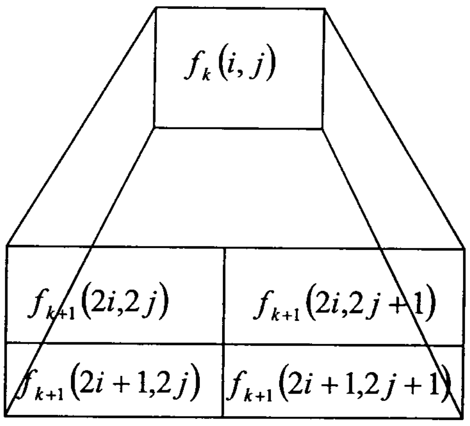


Fig. 3.6. Mapping from one pyramid level to the next level.

3.5.3 Velocity field

From the above Eq. (3.2), we can find the velocity estimation after several times iteration, but we still can't know where the target we need. Therefore we still have to calculate the velocity field to segment what we need. Based on the property that the velocities of the target and background are different, we can define the target by its maximum velocity difference with others in the image.

After finishing the calculation of the velocity field, we can find the mean velocity of the area we set. Finally we are able to find the maximum difference with the mean velocity, and that is the target we need.

$$diff = \sum_{i=-1}^{1}\sum_{j=-1}^{1}[U(x+i,y+j)-mean\_x]^2+[V(x+i,y+j)-mean\_y]^2, \tag{3.5}$$

where $U(x, y)$ is the velocity in X direction of $I_1(x,y)$; $V(x,y)$ is the velocity in Y direction, and $mean\_x$ is the mean velocity in X direction; $mean\_y$ is the mean velocity in *Y* direction. Fig. 3.7. shows the velocity fields of some images.

3.5.4 Computing range

In order to let this method can realize with real-time, not only downsampling the information but also checking the computing range of the image are necessary steps. About downsaping the information, we select $M=3$, it means the downsampling factor is 3, to decrease the computing quantity. And about checking the computing wind range this way, we set that the center is $I_1(x,y)$, it is the target position, and one 5×5 region to find the velocity field in the next image. So the iteration numbers of the next image will be decreased to 25 times, i.e., And we can decide the search range from Eq. (3.6). Some velocity fields shows in Fig. 3.7.

$$\begin{cases} u2 = x+5 \\ u1 = x-5, \end{cases}$$

$$\begin{cases} v2 = y+5 \\ v1 = y-5. \end{cases} \tag{3.6}$$

3.5.5 Target recognition

Although we constrain the loop times 25 times in every image, that is our assumption that the target motion is continuous in the image sequence. For this problem, building one decision rule is necessary. And we discover that the difference of the target and the mean velocities, it denotes ***diff***, will be greater than 1, in the situation that the target is real. To the contrary, if the ***diff*** value is smaller than 1, we couldn't decrease the iteration numbers to 25, and we must restart the whole image again to make sure the result correctness.

3.6 Summarization

In this chapter, we proposed an algorithm that detect the target based on the velocity estimation. After using image pyramid mutiresolution illustrated in Section 3.4 and reduce the search region introduced in Section 3.5.4, this algorithm is more efficient in computation time and practical in target detection. In next chapter, we will use the algorithm introduced in Section 3.5.1 ~ 3.5.5 to get a better performance.

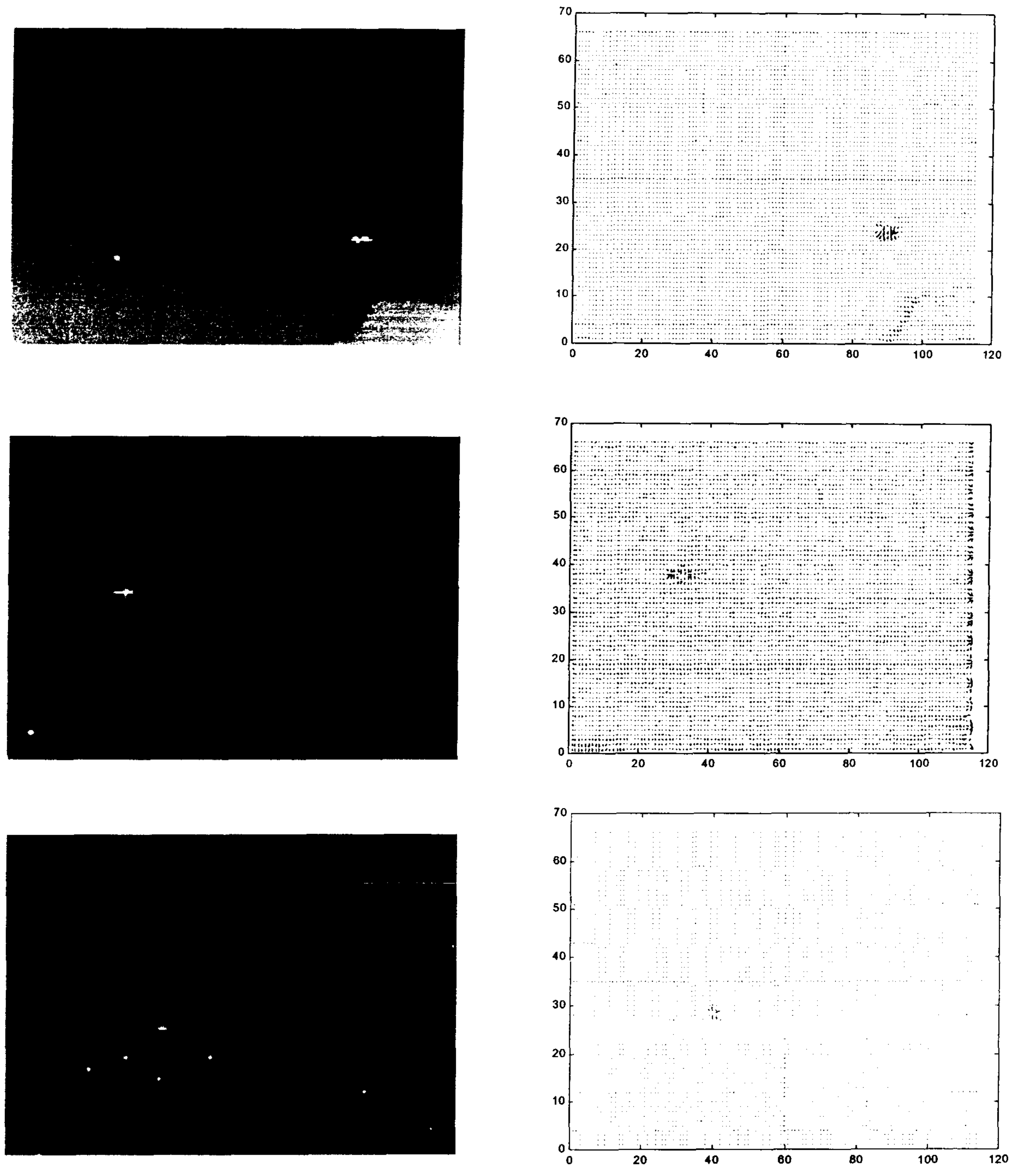


Fig. 3.7. The velocity fields of some test images in video.

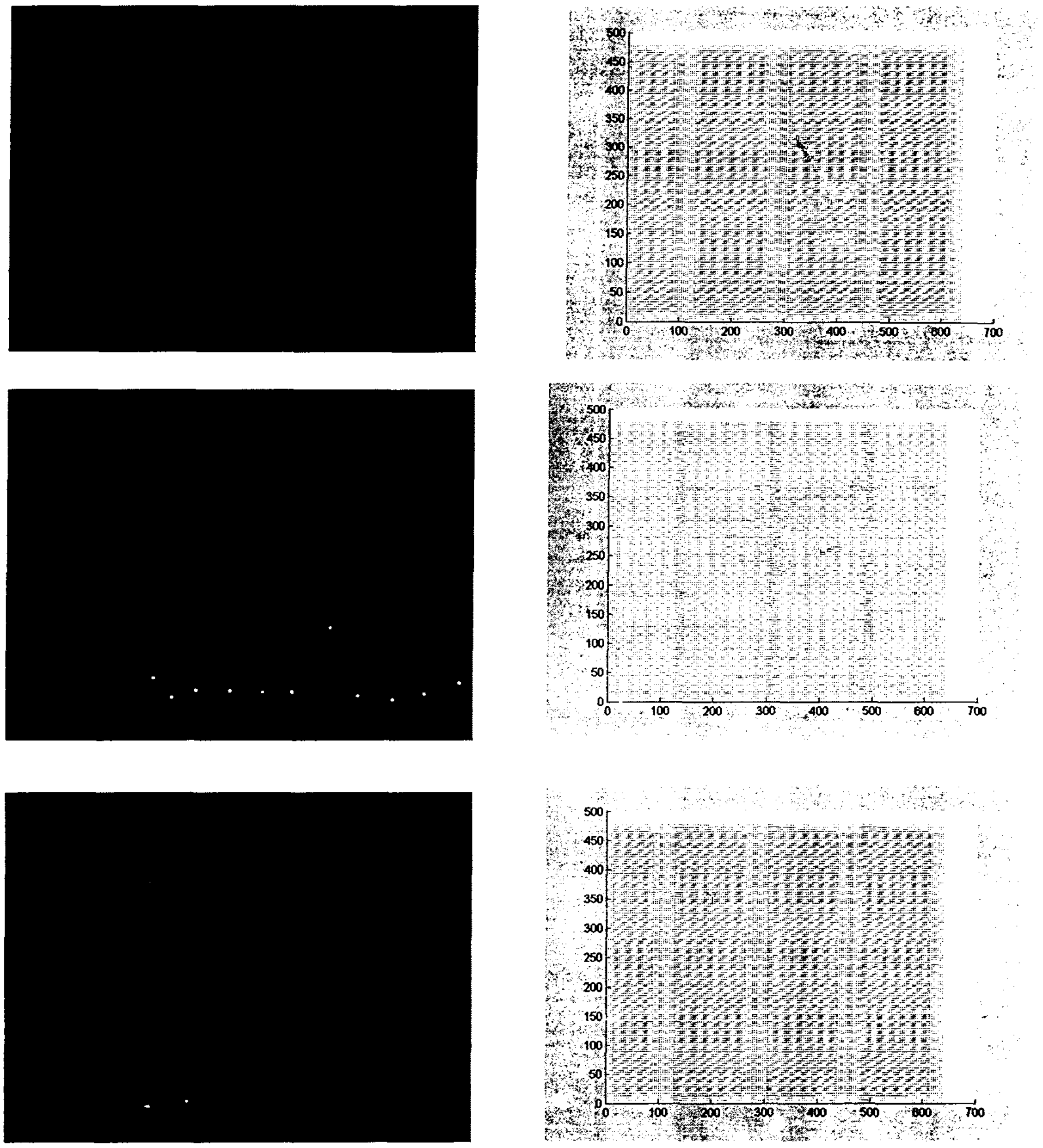


Fig. 3.7. (Continued)

Chapter 4

Experiments and Discussions

In this chapter, several experiments of optical flow estimation are performed on real sequences. The multi-target tracking experiment is evaluated. We also bring several discussions finally. Section 4.1 introduces experimental environments. Section 4.2 illustrates the experimental results. Finally, Section 4.3 deals with discussions.

4.1 Experimental Environments

Our experimental platform is an Intel-based PC with Pentium IV 2.4G CPU, 128 RAMs and run in Windows 2000 operating system. The computed optical flow is represented by a needle diagram. The estimated flow velocities are depicted as short line, showing the apparent displacement during one time step. In this section, we give experimental results of the proposed method this year and last year for optical flow estimation. We obtain these images per 1/30 second one image. In all of this sequence, we obtain 9 single target images and 2 multi-target images. The image sequences we used are real sequences.

Two real test image sequences will be used to target detection. One is the single target sequences, we obtain these images per 1/30 second one image. In all of this sequence, we obtain 11249 images. There are 9 section of test image sequences shown in Fig. 4.1. to Fig. 4.9 respectively. The other one is multi-target sequences, shown in Fig. 4.10. and Fig. 4.11. In this Hamburg taxi video, there were three moving objects: 1. the taxi turning the corner; 2. a car in the lower left, driving from left to right; 3. a van in the lower right driving right to left. In this Hall sequences, there were two moving people.

4.2 The Experimental Results

The experiment is divided into two parts. The results of estimating the single moving target is discussed in Section 4.2.1. The results of tracking multi-targets are discussed in Section 4.2.2.

4.2.1 The results of estimating the single moving target

In this section, we use the method proposed in chapter 3 to detect the target, and Fig. 4.1. to Fig. 4.9. have 9 section of single target test images respectively. We obtain these images per 1/30 second one image. The first section of test images shown as Fig. 4.1 has 1755 images. The second section of test images shown as Fig. 4.2 has 874 images. The third section of test images shown as Fig. 4.3 has 943 images. The 4th section of test images shown as Fig. 4.4 has 1256 images. The 5th section of test images shown as Fig. 4.5 has 1406 images. The 6th section of test images shown as Fig. 4.6 has 623 images. The 7th section of test images shown as Fig. 4.7 has 986 images. The 8th section of test images shown as Fig. 4.8 has 1520 images. The 9th section of test images shown as Fig. 4.9 has 1886 images. All of these image sequences are single moving target. Some of these image sequences have simple background and some of these image sequences have complex background.

Both situations, the some results of experiment are shown in Fig. 4.12. It shows some results by proposed method, because it is based on the velocity field, if the target velocity is different from the background velocity, even the target is not the lightest pixel in the image, we still can find out the target in the most case. Because of we use image pyramid that introduced in Section 3.4 and reduce search region that introduced in Section 3.5.4, the computation time and rate of success have better performance. Even if the iteration number reaches the upper bound, we still can detect the target in one second. Table 4.1 to Table 4.9 show the rate of success and the computation time. So the detection of the target is workable by proposed method.

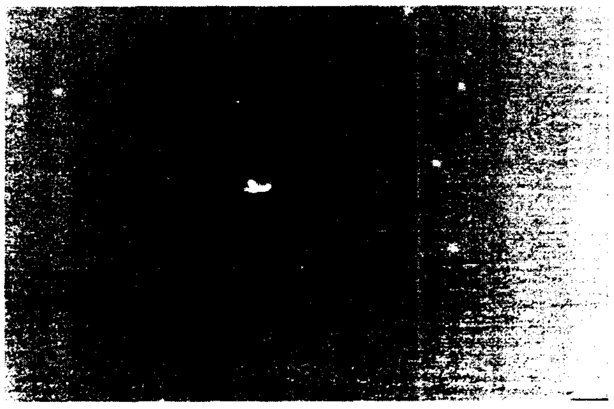
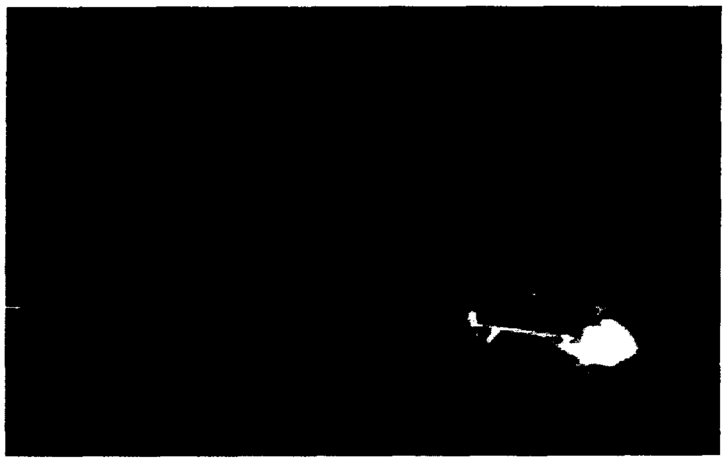


Fig. 4.1. The flight is moving from left to right with simple background.

Fig. 4.2. The flight is moving from left to right with complex background.



(a)

Fig. 4.3. (a) The helicopter is moving from right to left with simple background. (b) The helicopter is moving from top to ground with complex background.

(b)

Fig. 4.3. (Continued)

Fig. 4.4. The helicopter is moving from left to right with simple background.

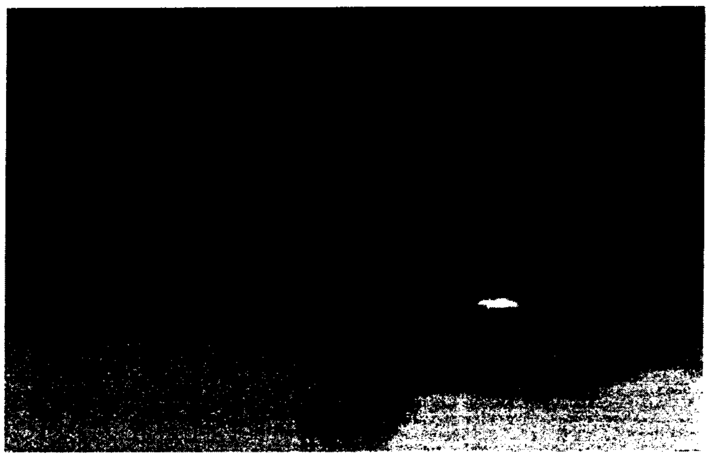


Fig. 4.5. The flight is moving from right to left.

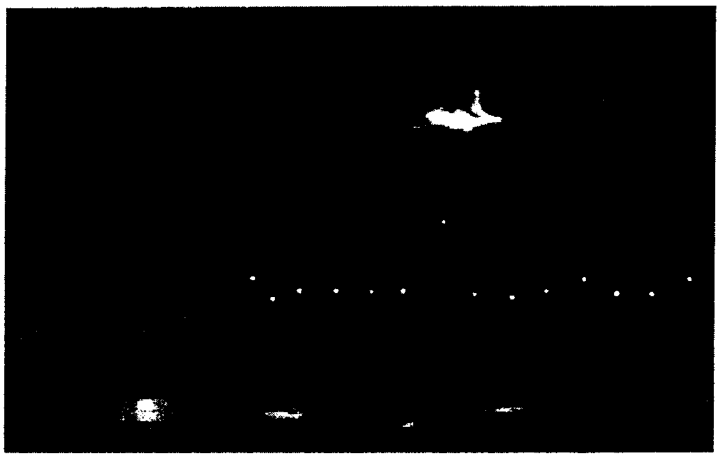


Fig. 4.6. The flight is moving right to left with high speed.

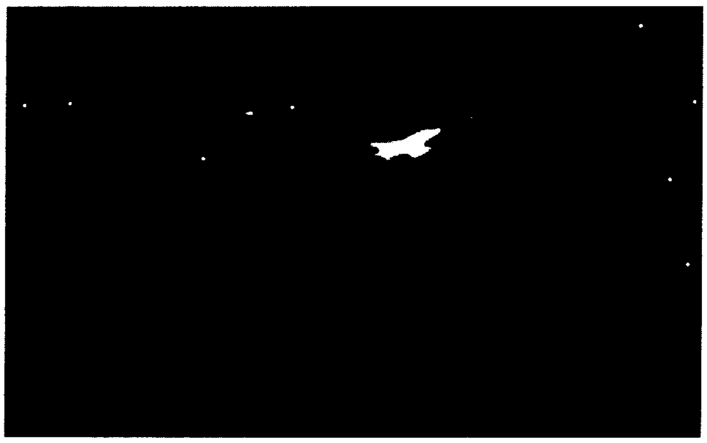


Fig. 4.7. The flight is moving from ground to sky.

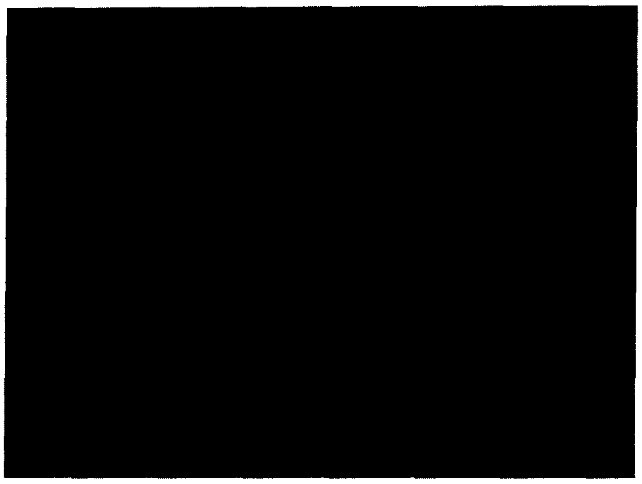


Fig.4.8. The flight is moving from left to right .

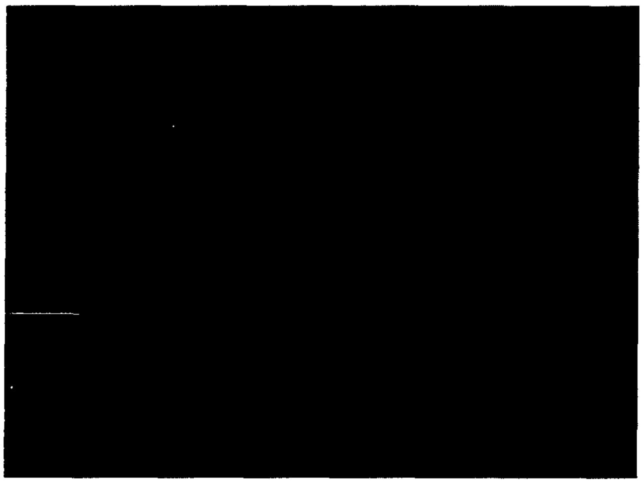


Fig. 4.9. The flight is moving from left to right.

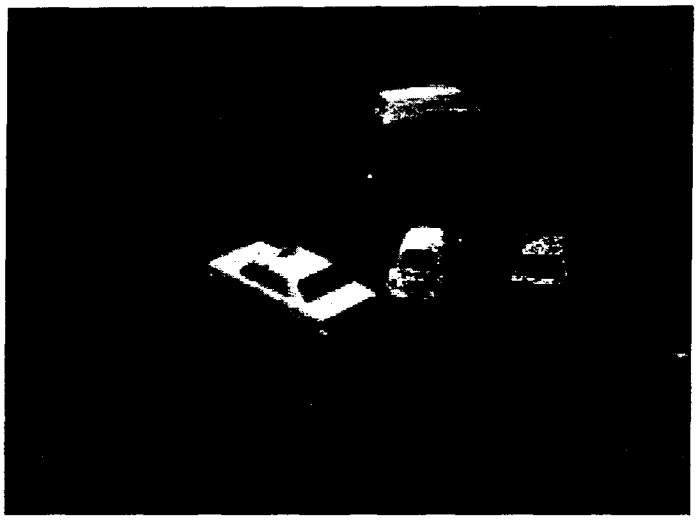


Fig. 4.10. Hamburg taxi.

Fig. 4.11 The hall sequences.

Table 4.1 Experiment result of target tracking, test images shown as Fig. 4.1.

	Proposed method	SSD method
The number of frames	1755	1755
The rate of success	100%	100%
The computation time (one frame)	0.031 sec	0.033 sec

Table 4.2 Experiment result of target tracking, test images shown as Fig. 4.2.

	Proposed method	SSD method
The number of frames	874	874
The rate of success	99.5%	99.2%
The computation time (one frame)	0.032 sec	0.034 sec

Table 4.3 Experiment result of target tracking, test images shown as Fig. 4.3.

	Proposed method	SSD method
The number of frames	943	943
The rate of success	95.3%	92.8%
The computation time (one frame)	0.034 sec	0.037 sec

Table 4.4 Experiment result of target tracking, test images shown as Fig. 4.4.

	Proposed method	SSD method
The number of frames	1256	1256
The rate of success	98.4%	98.2%
The computation time (one frame)	0.032 sec	0.036 sec

Table 4.5 Experiment result of target tracking, test images shown as Fig. 4.5.

	Proposed method	SSD method
The number of frames	1406	1406
The rate of success	99.5%	99.1%
The computation time (one frame)	0.031 sec	0.033 sec

Table 4.6 Experiment result of target tracking, test images shown as Fig. 4.6.

	Proposed method	SSD method
The number of frames	623	623
The rate of success	98.2%	98%
The computation time (one frame)	0.033 sec	0.035 sec

Table 4.7 Experiment result of target tracking, test images shown as Fig. 4.7.

	Proposed method	SSD method
The number of frames	986	986
The rate of success	99%	99.3%
The computation time (one frame)	0.034 sec	0.036 sec

Table 4.8 Experiment result of target tracking, test images shown as Fig. 4.8.

	Proposed method	SSD method
The number of frames	1520	1520
The rate of success	97.8%	96.5%
The computation time (one frame)	0.032 sec	0.036 sec

Table 4.9 Experiment result of target tracking, test images shown as Fig. 4.9.

	Proposed method	SSD method
The number of frames	1886	1886
The rate of success	95.9%	93.6%
The computation time (one frame)	0.031 sec	0.034 sec

4.2.2 The results of tracking multi-targets

Because multi-targets tracking in real world occurs frequently, the proposed method must have good performance in multi-targets tracking. To evaluate the performance and efficacy of multi-target tracking method, the experiments are established. These test images are shown in Fig. 4.10. and Fig. 4.11. We obtain these images per $\frac{1}{30}$ second one image. Fig. 4.14. is shown the velocity field of the Hamburg taxi image and the Hall image. Fig. 4.15. is shown some results of multi-target tracking. The results of experiment are shown in Table 4.10 and Table 4.11, the proposed method can deal with multi-target detection and use less computation time than SSD method. To find lightest point method just can deal with single target, it is shown in Table 4.10 and Table 4.11.

Table 4.10 Experiment result of Hamburg taxi sequences.

	Proposed method	SSD method	Lightest point method
Number of frames	300	300	300
Number of targets	3	3	1
The computation time (one frame)	0.037 sec	0.063 sec	0.028 sec

Table 4.11 Experiment result of the hall sequences.

	Proposed method	SSD method	Lightest point method
Number of frames	240	240	240
Number of targets	2	2	1
The computation time (one frame)	0.035 sec	0.058 sec	0.029 sec

4.3 Discussions

For single target tracking, the rate of success and computation time of proposed method and previous method are compared in Table 4.1 to Table 4.9 respectively. From the Table 4.1 to Table 4.9, we find that the method we proposed has better performance than the previous method. The test images are shown as Fig. 4.1, the results have best performance, because the target is all the lightest point in image and have a singular background. Test images are shown as In Fig. 4.3, the results are worst, because the frames have much noise, complex background and the target is not the lightest point in image. The results in Table 4.9 are not good enough, because the target is too small. It is sensitive to noise, so the target is hard to recognize from background. As a result, the target size and complexity of background have much influence to the method we proposed. When the target is consist of 20 ~ 50 pixels and target is the lightest point in image, the results of tracking are best.

One of the main problems we find with the SSD-based matching techniques is their ability to estimate sub-pixel displacements, as shown in Fig. 4.13. With image translation and higher speeds they appear to perform well, but when the motion field involves small velocities with a significant dilational component the estimated displacements are often poor. In these cases it appears that SSD-based estimates of displacements are more accurate with integer displacements than subpixel velocities.

An image pyramid is one kind of multiple resolution manifestation which can observe the same image with different resolution. Therefore, multiple resolutions can easily abstract some important features from images. We used the image pyramid to reduce the amount of data used in the optical flow calculation. In the optical flow estimation, the image pyramid strategy is utilized to speed up the process at the same time we reduce the searching range that can reduce the computation time and mistake. Hence make our system being suitable for per $^1/_{30}$ second one image. So we have better performance in process time than previous methods.

From Table 4.9 and Table 4.10, the results illustrate that proposed method can deal with multi-target. The lightest point method assumes the target is the lightest point in image, so it just can deal with single target. It failed when the target is not the lightest point in image. The correlation-based SSD approach needs much computation time. The method we proposed has less computation time and can deal with multi-target. In the future, the algorithm should be more robust in complex background and increases the number of targets.

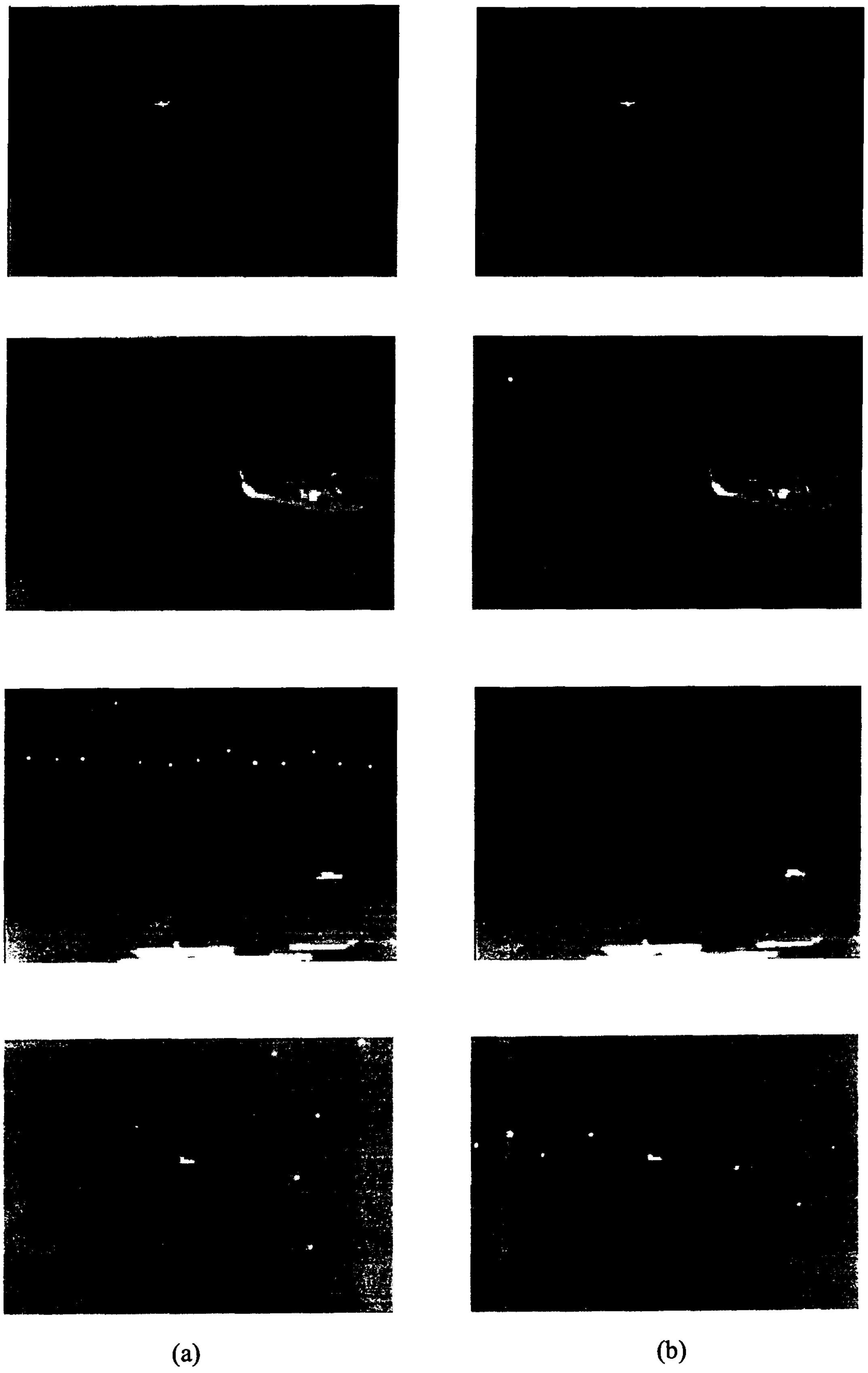
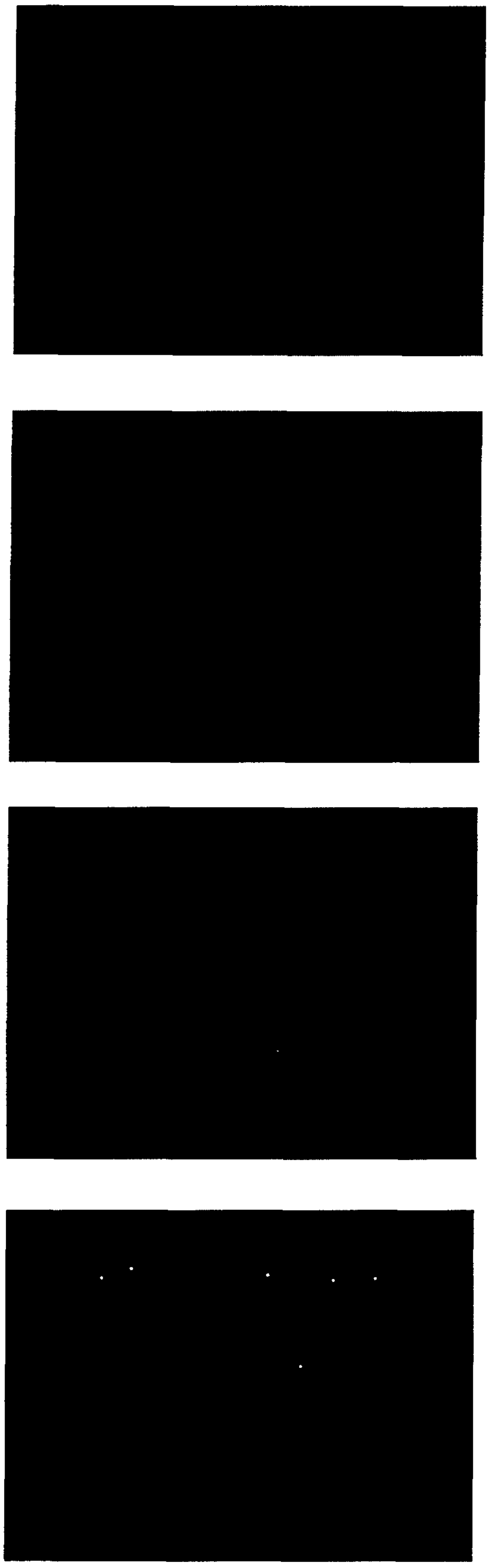


Fig. 4.12. Some final results of single target. (a) Original images. (b) Results using proposed method.


(a)

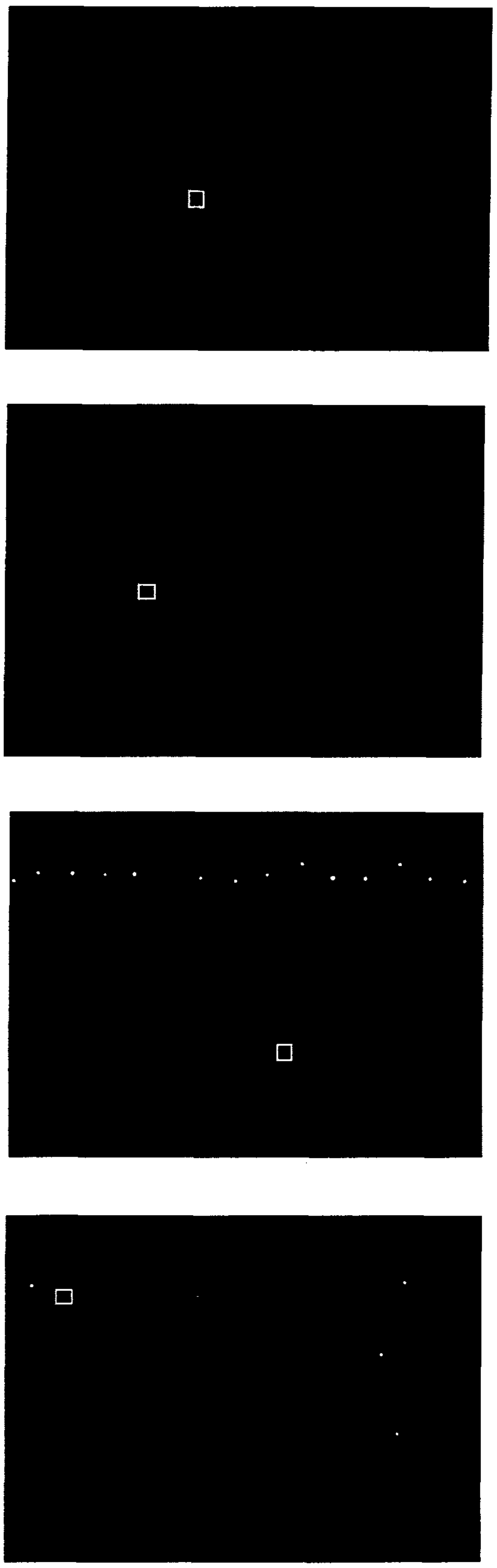
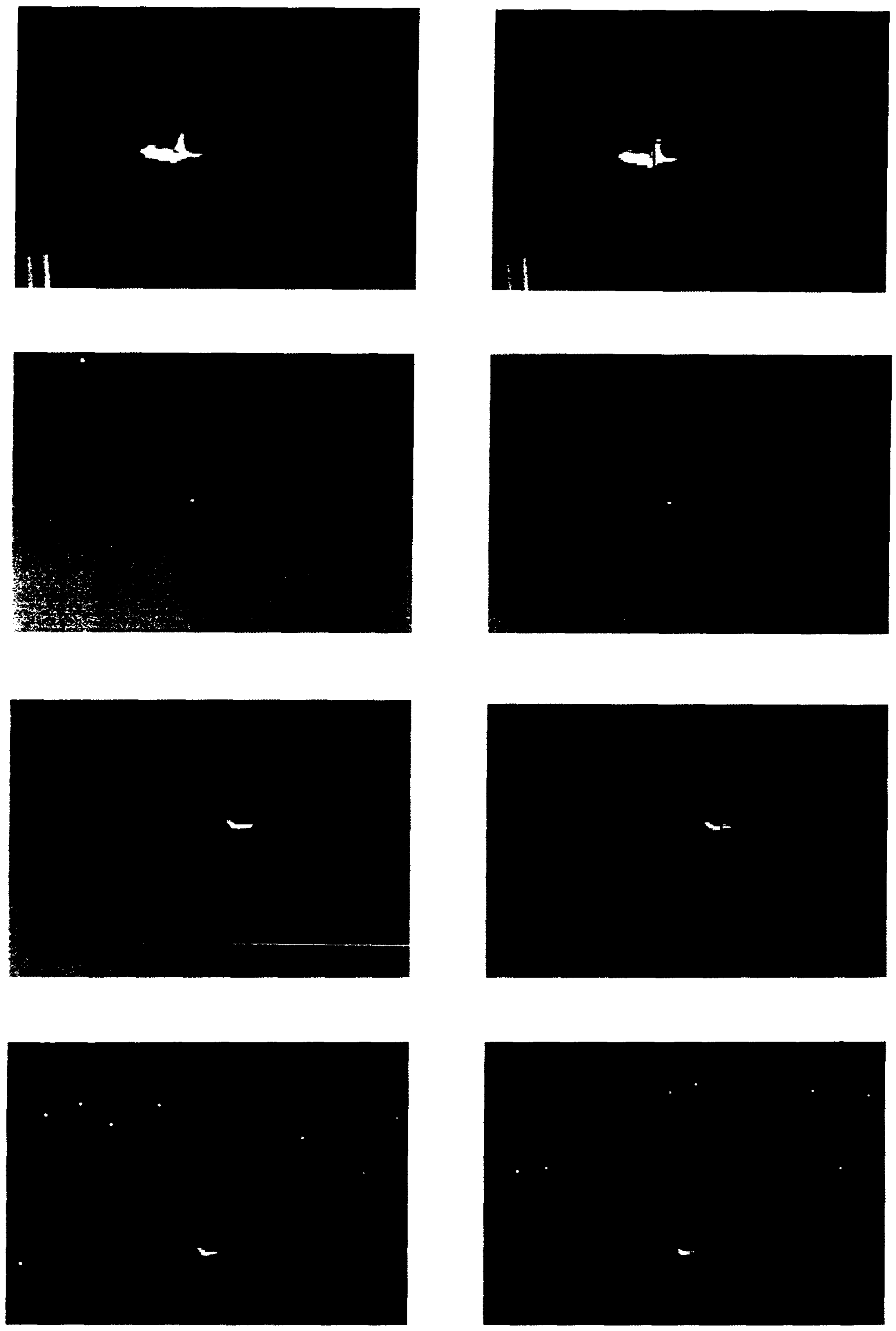

(b)

Fig. 4.12. (Continued)



(a) (b)

Fig. 4.12. (Continued)

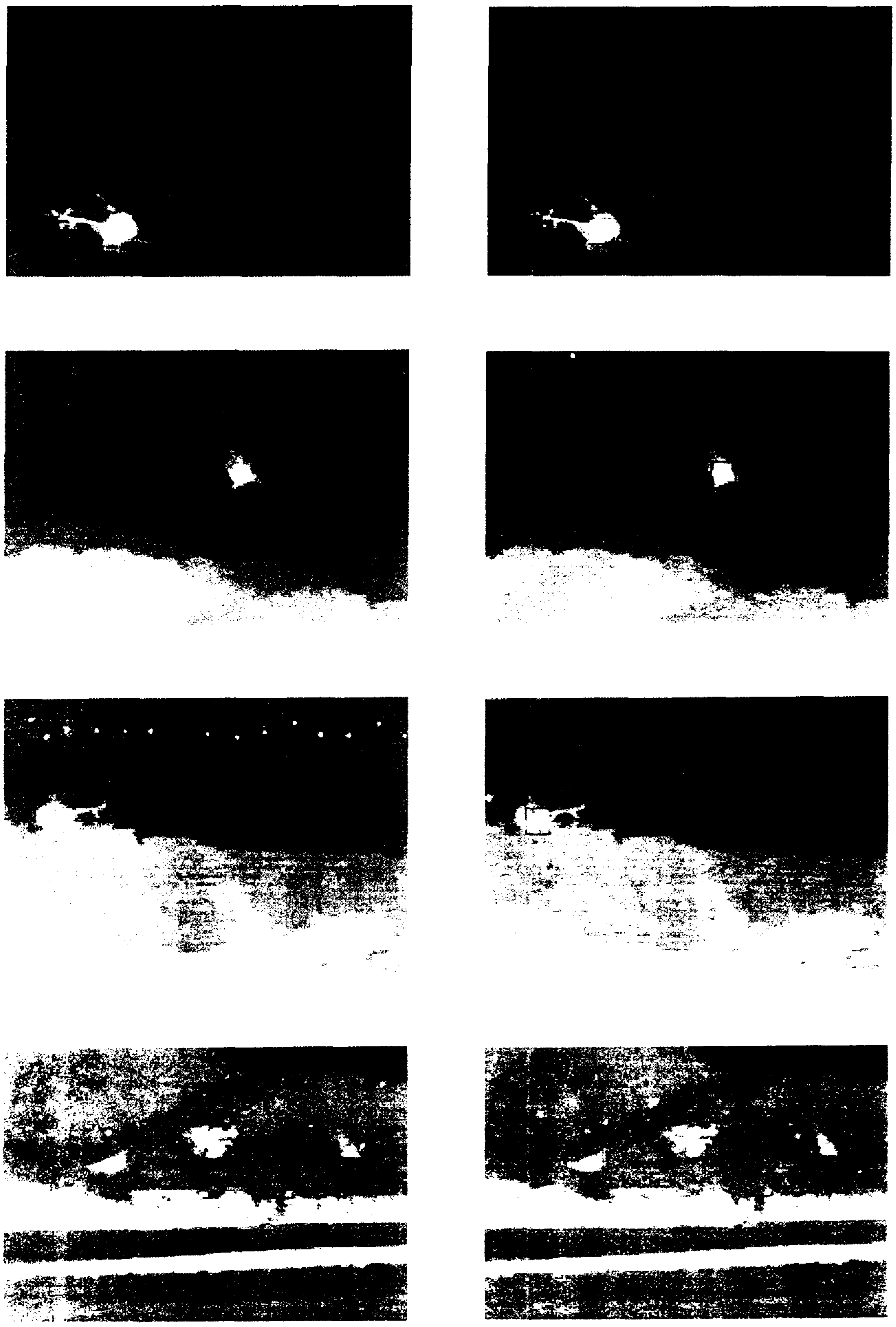
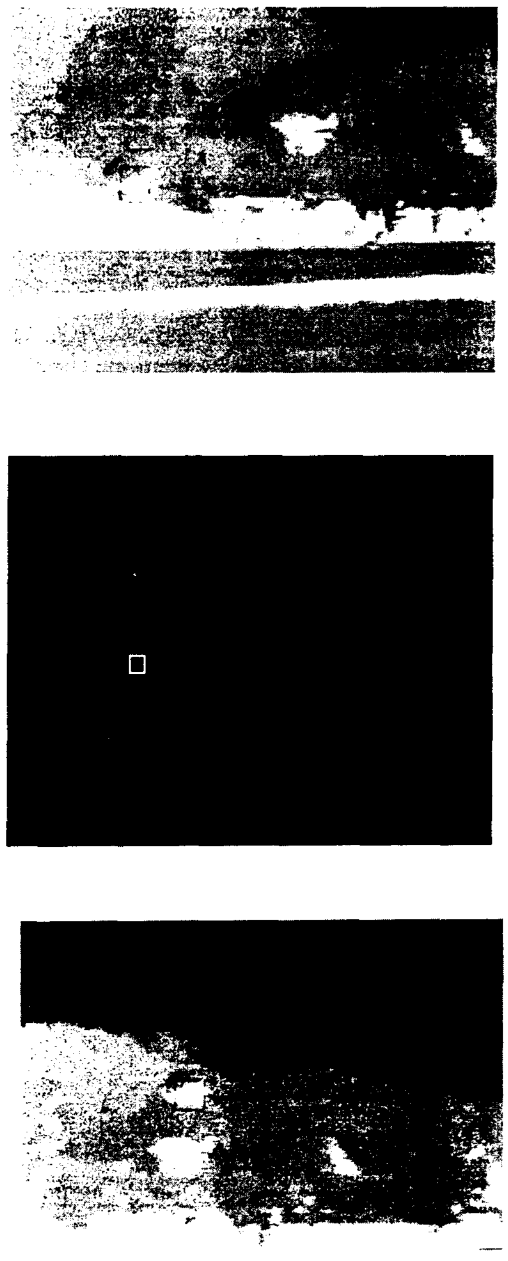


Fig. 4.12. (Continued)


(a)

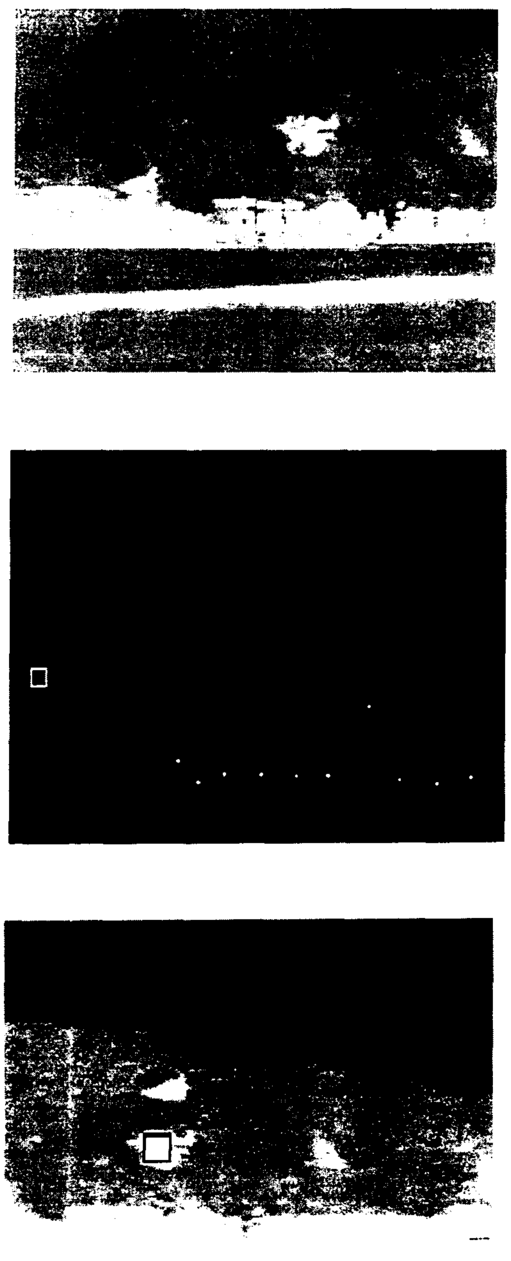
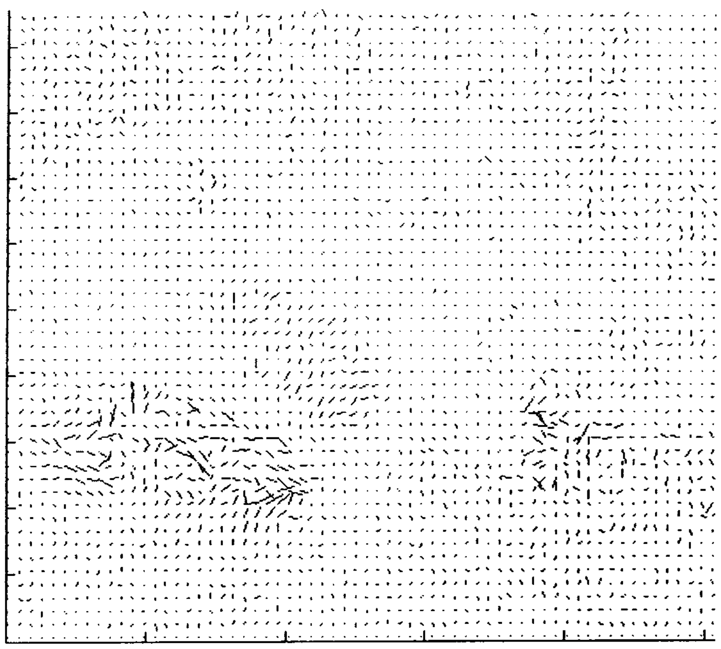
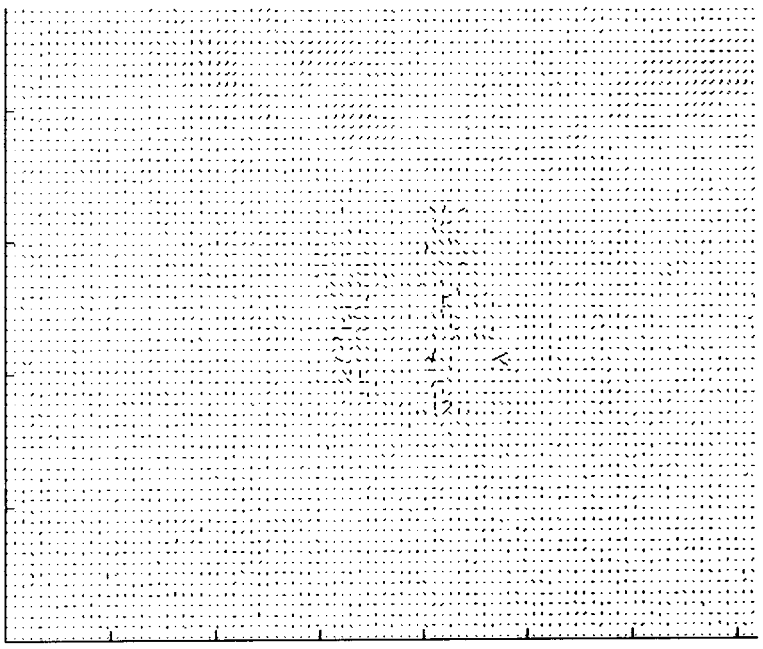

(b)

Fig. 4.13. The experimental results. (a) The method we proposed. (b) The previous method.

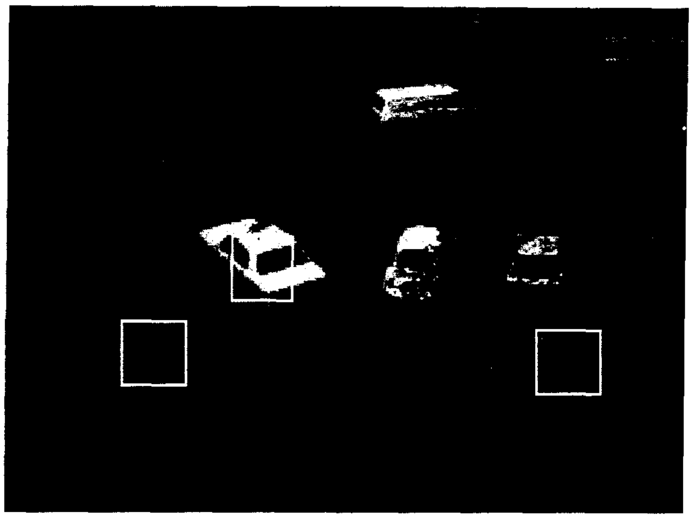


(a)



(b)

Fig. 4.14. The velocity field. (a) Hamburg taxi. (b) The Hall sequences.



(a)

(b)

Fig. 4.15. The final results of multi-target. (a) Hamburg taxi. (b) The Hall sequences.

Chapter 5

Direction of Arrival Estimation Based on Phase Differences Using Neural Fuzzy Network

This report aims to provide a powerful and effective methodology for direction of arrival (DOA) estimation and emitter identification (EID) in electronic warfare (EW) applications, respectively. At first, we propose a six-layered neural fuzzy network (NFN) with supervised learning algorithm to perform direction of arrival (DOA) estimation instead of the conventional DOA estimation methods, such as MUSIC and MLE, which are computationally intensive and difficult to implement in real time. The input feature for DOA estimation is phase difference (PD). The trained NFN can always find itself an economical network size in fast learning process. Second, we also propose an a three-layer vector neural network (VNN) with a supervised learning algorithm suitable for emitter identification (EID). The VNN can accept interval-value input data as well as scalar input data. The input features of the EID problems include the radio frequency (RF), pulse width (PW), and pulse repetition interval (PRI) of a received emitter signal. A suitable new vector-type backpropagation (NVTBP) learning algorithm is also derived. The algorithm can tune the weights of VNN optimally to approximate the nonlinear mapping between a given training set of feature intervals and the corresponding set of desired emitter types. Above constructing networks including the NFN and VNN are verified by emulating examples with/without noise conditions through computer simulations. The proposed networks are also compared to other methods by the same examples. In summary, each one of the aforementioned networks (either NFN or VNN) has high accuracy/high correction rate in the DOA estimation and in the EID, respectively.

5.1 The Introduction of Direction of Arrival Estimation Based on Phase Differences

In the surveillance and reconnaissance communities, the direction of arrival (DOA, sometimes referred to as angle of arrival, AOA) information and emitter identification (EID) capability are extremely important problems, especially in the electronic warfare (EW) field [62]. The relationship DOA information with radar is illustrated as shown in Fig. 5.1. Many conventional DOA estimation methods have been proposed, including the multiple signal classification (MUSIC) method of Schmidt [63], and the maximum likelihood (ML) technique [64, 65]. However, these methods are computationally intensive and difficult to implement in real time [66].

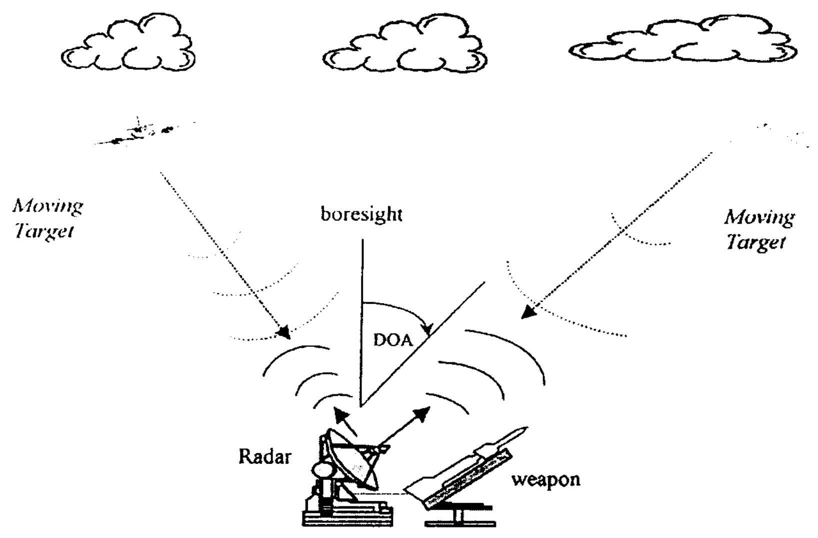


Figure. 5.1. The relationship DOA information with radar.

Neural networks have recently drawn a great deal of attention in many practical signal processing problems [67, 68] for the sake of their massive parallelism and global connectivity. The DOA estimation problem can be considered as a mapping problem. Park and Sandberg proved the radial basis function network (RBFN) with one hidden layer was capable of universal approximation [69]. Thus, a RBFN was proposed to approximate the unknown mapping function such that whenever the available phase differences were fed into the network, the estimated DOA could be obtained from the output of the network [70]. Although the RBFN can overcome the large computation and high cost problems in the conventional DOA estimation methods, a large number of parameters (network weights) need to be tuned in order to reach good performance of estimation because all of input variables are fully connected to its hidden nodes. To cope with the drawbacks encountered in the RBFN, while still keeping its advantages, a new DOA estimation algorithm with a neural fuzzy network (NFN) is developed.

While, in military operations, an electronic support measures (ESM) system such as radar warning receiver (RWR) is needed to intercept, identify, analyze, and locate the existence of emitter signals. The primary function of the RWR is emitter identification (EID), which is used to warn the crew of an immediate threat with enough information to take evasive action. Many conventional signal recognition techniques are computationally intensive and require a key man to validate and verify the analysis [71].They often fail to identify signals under high signal density environment, especially, in near real time.

In addition, the EID problem is also considered as a nonlinear mapping problem. The input features, including RF, PW, and PRI, are extracted from pulse descriptor words (PDWs). Since the values of these features vary in interval ranges in accordance with a specific radar emitter, a vector neural network (VNN) is proposed to process interval-value input data. To train the VNN, a suitable learning algorithm should be developed. Most existing learning methods in neural networks are designed for processing numerical data [71]-[77]. Ishibuchi and his colleagues extended a normal (scalar-type) backpropagation (BP) learning algorithm to the one which can train a feedforward neural network with fuzzy input and fuzzy output [78]. Similar to their approach, we derive the conventional vector-type BP (CVTBP) and new vector-type BP (NVTBP) algorithms for training the proposed VNN. The effectiveness of the NVTBP learning algorithm are demonstrated on several EID problems, including the two-emitter and three-emitter identification problems with/without additive noise. The rest of this report is organized as follows. Section 5.2 gives the problem statements. In Section 5.3, the basic structure and function of the NFN is briefly introduced. Some simulation results are also presented. In Section 5.4, the basic architecture of a vector neural network is presented. The associated parameter learning algorithm with vector feature is also derived based on different error function. In addition, some concluding remarks are given in Section 5.5. Also the extension of the report and future research are described.

5.2 Problem Statements

For DOA estimation problem, We assume that a plane wave is coming in at incident angle θ from the boresight. Then the phase differences between a signal in the reference sensor and signals in the other sensors with additive noise can be well-expressed by

$$\psi_{1j} = 2\pi\rho_{1j}\sin\theta + \delta\psi_{1j}. \tag{5.1}$$

$$\rho_{1j} = \frac{1}{2}\sum_{i=1}^{j-1}\gamma_i,\ j = 2,3,4, \tag{5.2}$$

where

- ψ_{1j} and $\delta\psi_{1j}$ are the phase difference and phase error (noise) between the first sensor and the jth sensor, respectively;

- ρ_{1j} is the normalized physical spacing between the first sensor and the jth sensor, where the normalization is made with respect to the wavelength at the operating frequency;
- The adjacent sensor spacings (normalized to half-wavelengths at the operating frequency) are $\gamma 1$ between sensors 1 and 2, $\gamma 2$ between sensors 2 and 3, and $\gamma 3$ between sensors 3 and 4;
- θ is angle of arrival.

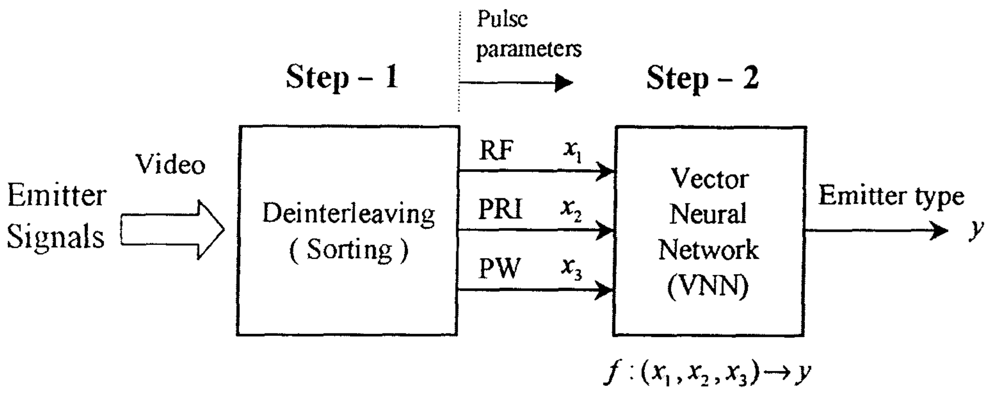


Figure. 5.2. The flow chart of emitter signal classification.

The phase differences are measured from Eq. (5.1), then the DOA can be determined through the complex hardware circuits with DOA processing algorithm. From a different point of view, the DOA estimation problem can be considered as a mapping from the space of DOA (θ) to the space of phase difference (x) as x $= f(\theta)$. Then the DOA can be obtained via the inverse of this mapping directly, i.e., $\theta = g(\mathrm{x})$

$= f^{-1}$(x). The objective of learning is to minimize the error function

$$E(w) = \frac{1}{2}\sum_{p=1}^{Nt}(d_p - y_p)^2, \tag{5.3}$$

where the subscript p indicates the pth training pair, Nt is the total number of training pairs, d_p is the desired output, and y_p is the actual output.

On the other hand, the problem of emitter signal classification is performed in a two-step process as illustrated in Fig. 5.2. In this report, the EID problem is considered as a nonlinear mapping problem, the mapping from the space of feature vectors of emitter signals to the space of emitter types. The three parameters, RF(x1), PRI(x2), and PW(x3), are used to form the feature vector [x1, x2, x3] in this problem. Such a nonlinear mapping function can be approximated by a suitable neural network [71, 72]. To endow a neural network with the interval-value processing ability, we propose a vector neural network (VNN) which can accept either interval-value or scalar-value input and produce scalar output. In the training phase, the VNN is trained to form a functional mapping from the space of interval-value features to the space of emitter types based on Nt samples of training pairs $(\tilde{x}_p ; d_p)$ for the EID problem, where p = 1, . . . ,Nt indicating the pth training pair, $\tilde{x}_p$= $[\tilde{x}_{p1}, \tilde{x}_{p2}, \tilde{x}_{p3}]$. The input-output relationship of the VNN is denoted by

$$y_p = \hat{f}(\tilde{X}_p),, \tag{5.4}$$

where y_p is a m-dimensional vector indicating the actual output of the VNN, and $\hat{f}$ represents the approximated function formed by the VNN. More clearly, the VNN is trained to represent the EID mapping problem in the following if-then form:

$$\text{IF } x_{p1} \in [x_{p1}^L, x_{p1}^U] \text{ and } \cdots \text{ and } x_{pn} \in [x_{pn}^L, x_{pn}^U]$$
$$\text{THEN } \mathbf{x}_p = [x_{p1}, \cdots, x_{pn}] \text{ belongs to } C_k. \tag{5.5}$$

where c_k denotes the *k*th emitter type. The objective of learning is to obtain an approximated model $\hat{f}(\cdot)$ for the mapping in Eq. (5.4) and Eq. (5.5) such that the error function indicating the difference between d_p and y_p, p = 1, 2, . . . ,*Nt*, is minimized. Two di.erent error functions are used in this paper, one is the common root-mean-square error function, and the other is

$$E(w) = -\sum_{p=1}^{N_t} \{\mathbf{d}_p \ln \mathbf{y}_p + (1 - \mathbf{d}_p)\ln(1 - \mathbf{y}_p)\}. \tag{5.6}$$

5.3 DOA Estimation Using a Neural Fuzzy Network

The structure of the NFN is shown in Fig. 5.3. This 6-layered network realizes a fuzzy model of the following form

$$\text{Rule } i: \text{IF } x_1 \text{ is } A_1^i \text{ and } \cdots \text{ and } x_n \text{ is } A_n^i$$
$$\text{THEN } y \text{ is } m_0^i + a_j^i x_j + \cdots.$$

where A_j^i is the fuzzy set of the *i*th linguistic term of input variable *xj* , m_0^i is the center of a symmetric membership function on *y*, and a_j^i is a consequent parameter. We shall next describe the functions of the nodes in each of the six layers of the NFN.

Layer 1 : No computation is done in this layer. Each node in this layer, only transmits input values to the next layer directly. That is,

$$f = u_i^{(1)} \text{ and } a^{(1)} = f \tag{5.7}$$

From the above equation, the link weight in layer one ($w_i^{(1)}$) is unity.

Layer 2 : Each node in this layer corresponds to one linguistic label (low, high, etc.) of one of the input variables in Layer 1. The membership value specifies the degree to which an input value belongs to a fuzzy set is calculated in Layer 2. With the choice of Gaussian membership function, the operation performed in this layer is

$$f_j(u_i^{(2)}) = -\frac{(u_i^{(2)} - m_{ij})^2}{\sigma_{ij}^2} \text{ and } a^{(2)}(f) = e^f, \tag{5.8}$$

where m_{ij} and σ_{ij} are, respectively, the center (or mean) and the width (or variance) of the Gaussian membership function of the *j*th partition for the *i*th input variable *ui*. Hence, the link weight in this layer can be interpreted as m_{ij}.

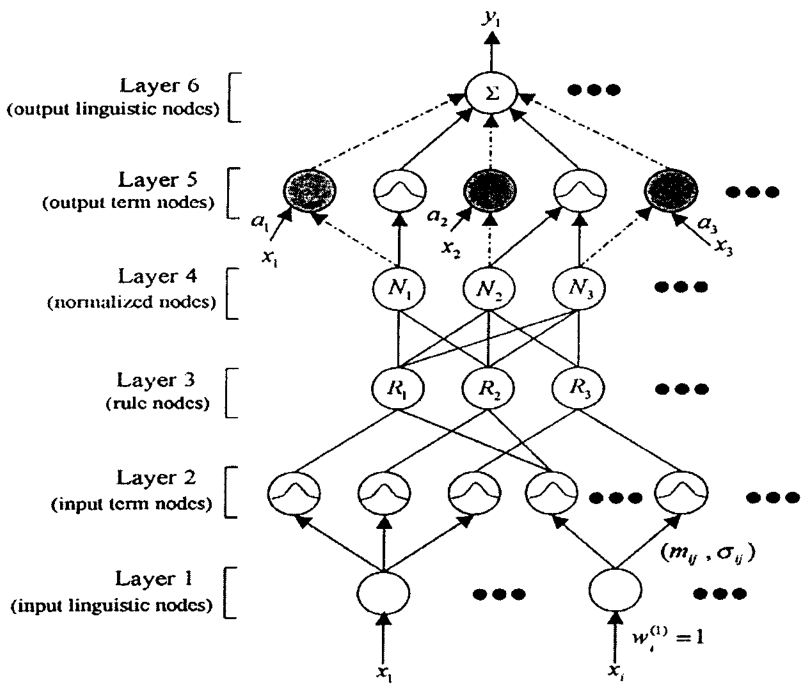


Figure. 5.3. Structure of the NFN.

Layer 3 : A node in this layer represents one fuzzy logic rule and performs precondition matching of a rule. Here, we use the following AND operation for each Layer-3 node,

$$f(u_i^{(3)}) = \prod_{i=1}^{q} u_i^{(3)} = e^{-[D_i(\mathrm{x}-\mathrm{m}_i)]^T[D_i(\mathrm{x}-\mathrm{m}_i)]}$$

$$a^{(3)}(f)=f, \qquad (5.9)$$

where q is the number of Layer-2 nodes participating in the IF $Di = diag(1/\sigma i1,\ 1/\sigma i2,\ \cdot\ \cdot\ \cdot,\ 1/\sigma in)$, and m$i$ = $(mi1, mi2,\ \cdot\ \cdot\ \cdot, min)T$. The connection weights in Layer 3 ($w_i^{(3)}$) have the value of one. The output f of a Layer-3 node represents the .ring strength of the corresponding fuzzy rule.

Layer 4 : The number of nodes in this layer is equal to that in Layer 3, and the .ring strength calculated in Layer 3 is normalized in this layer by

$$f(u_i^{(4)}) = \sum_{i=1}^{r} u_i^{(4)} \quad \text{and} \quad a^{(4)}(f) = u_i^{(4)}/f. \tag{5.10}$$

where r is the number of rule nodes in Layer 3. Like Layer 3, the link weight ($w_i^{(4)}$) in this layer is unity, too.

Layer 5 : This layer is called the consequent layer. Two types of nodes are used in this layer, and they are denoted as blank and shaded circles in Fig. 5.3, respectively. The node denoted by a blank circle (blank node) is the essential node representing a fuzzy set (described by a Gaussian membership function) of the output variable. Different nodes in Layer 4 may be connected to a same blank node in Layer 5, meaning that the same consequent fuzzy set is specified for different rules. The function of the blank node is

$$f = \sum_{i=1}^{s} u_i^{(5)} \quad \text{and} \quad a^{(5)}(f) = f \cdot a_0^i. \tag{5.11}$$

where s is the number of nodes in Layer 4, and $a_0^i = m_0^i$ is the center of a Gaussian membership function. As to the shaded node, it is generated only when necessary. Each node in Layer 4 has its own corresponding shaded node in Layer 5. One of the inputs to a shaded node is the output delivered from Layer 4, and the other possible inputs (terms) are the input variables from Layer 1. The shaded node function is

$$f = \sum_{j=1}^{n} a_j^i x_j \quad \text{and} \quad a^{(5)}(f) = f \cdot u_i^{(5)}. \tag{5.12}$$

where the summation is over the significant terms only connected to the shaded node, and a_j^i is the corresponding parameter. Combining these two types of nodes in Layer 5, we obtain the whole function performed by this layer as

$$a^{(5)}(f) = (\sum_{j=1}^{n} a_j^i x_j + a_0^i) u_i^{(5)}. \tag{5.13}$$

Layer 6 : Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by Layer 5 and acts as a defuzzifier with

$$f(u_i^{(6)}) = \sum_{i=1}^{t} u_i^{(6)} \quad \text{and} \quad a^{(6)}(f) = f. \tag{5.14}$$

where t is the number of nodes in Layer 5. Two types of learning, structure and parameter learning, are used concurrently for constructing the NFN. A detailed description of the overall learning algorithms can be found in [79].

5.3.1 Simulation Results

The procedure of DOA estimation using NFN based on phase differences obtained from Interferometer is shown in Fig. 5.4. The system performance is verified for input signal with frequency 2.702 GHz.

(1) *Performance of DOA Estimation without Noise*

The phase differences between a signal in the reference sensor and signals in the other sensors without additive noise can be well-expressed by

$$\varphi_{1j} = 2\pi\rho_{1j}\sin\theta, \tag{5.15}$$

$$\rho_{1j} = \frac{1}{2}\sum_{i=1}^{j-1}\gamma_i,\ j = 2,3,4, \tag{5.16}$$

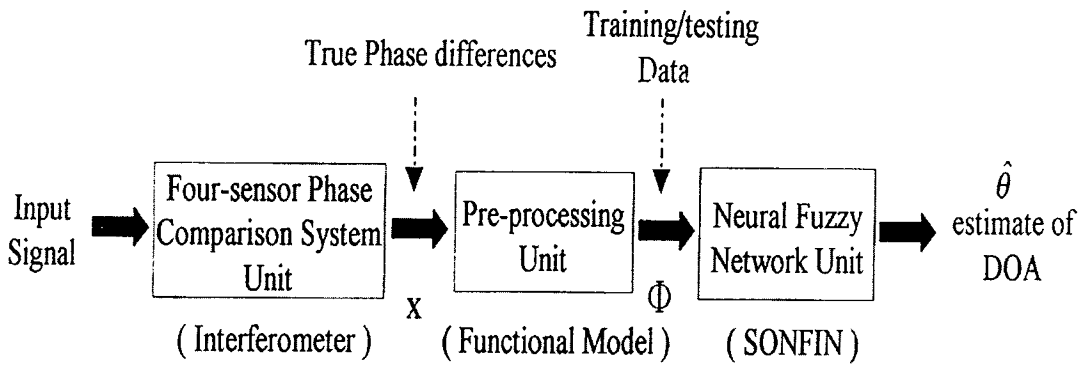


Figure. 5.4. Flow chart of DOA estimation using the NFN.

First, the training data are generated as follows. We divide the range of DOA values, -45 to +45, equally into 180 intervals, with each interval being 0.5 degree. As a result, we have 181 DOA values in the set {-45.0,-44.5,..., +44.5, +45}. With the same procedure, we can obtain 301 testing patterns by dividing the range of DOA values,

-45 to +45, into 0.3-degree intervals. Simulation results show that the convergence rate of the NFN is much higher than that of the RBFN in the training phase. Also, a detailed performance comparisons are listed in Table 5.1.

(2) *Performance of DOA Estimation with Noise*

In this simulation, the training was performed with 181 data sets derived from Eqs. (5.15) and (5.16) (assuming the absence of noise), whereas the testing was performed

Table 5.1 Performance comparison of the NFN and RBFN estimators on the DOA estimation problem.

Methods	SONFIN	RBFN		
		Nr=79	*Nr*=99	*Nr*=119
Estimation Accuracy (degree)	0.0801	2.7159	0.4559	0.0823
Number of nodes	57	86	106	126
Number of parameters	41	1027	1287	1547

with 301 data sets contaminated with uniformly distributed phase errors derived from Eqs. (5.1) and (5.2) to simulate real measurements. In Eq. (5.1), the additive noise term is given by

$$\delta\psi_{1i} = \frac{\sigma_\phi}{\sqrt{2}\pi\rho_{1i}}, j = 2, 3, 4, \tag{5.17}$$

$$\sigma_\phi = \frac{180^\circ}{\pi\sqrt{SNR}}. \tag{5.18}$$

where SNR is in terms of power. The above additive noise term is given by references [62 ,80]. For comparison, the DOA estimation errors obtained from the NFN and those from the RBFN with additive phase errors at different signal-to-noise ratio values are plotted in Fig. 5.5.

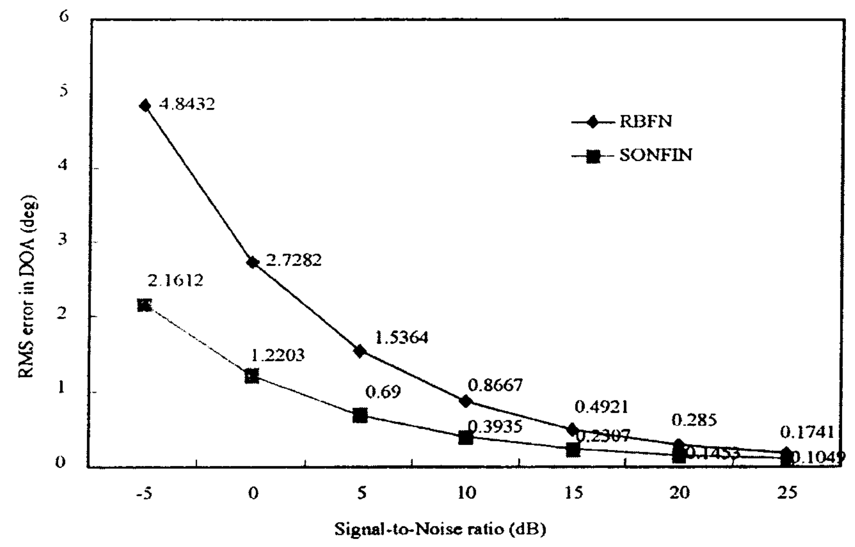


Figure. 5.5. RMS error in DOA of the NFN and RBFN (*Nr* = 119) under the additive phase error conditions with different signal-to-noise ratios.

5.4 Structure of Vector Neural Network (VNN)

We shall introduce the structure and function of the vector neural network (VNN) which can process interval-value as well as scalar-value data. The general structure of VNN is shown in Fig. 5.6, where the solid lines show the forward propagation of signals, and the dashed lines show the backward propagation of errors. The input-output relation of each node of VNN is explicitly calculated as follows, where $\tilde{x}_{pi} = [x^L_{pi}, x^U_{pi}]$.

Input nodes: Each input node just passes the external input, $\tilde{x}_{pi} = [x^L_{pi}, x^U_{pi}]$, $i = 1, \ldots, n$, forward to the hidden nodes.

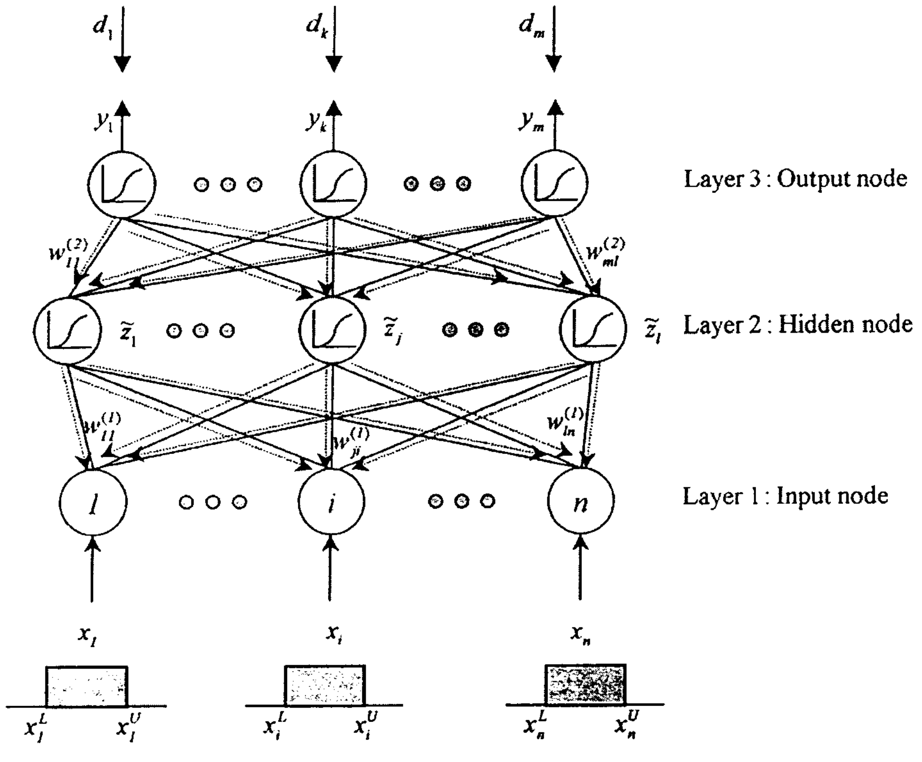


Figure. 5.6. The three-layer architecture of the proposed VNN.

Hidden nodes:

$$\tilde{z}_{pj} = [z^{L}_{pj}, z^{U}_{pj}] = [f(net^{L}_{pj}), f(net^{U}_{pj})], \quad j = 1, \ldots, l. \tag{5.19}$$

$$net^{L}_{pj} = \sum_{\substack{i=1 \\ w^{(1)}_{ji} \geq 0}}^{n} w^{(1)}_{ji} x^{L}_{pi} + \sum_{\substack{i=1 \\ w^{(1)}_{ji} < 0}}^{n} w^{(1)}_{ji} x^{U}_{pi} + \theta_{j}. \tag{5.20}$$

$$net^{U}_{pj} = \sum_{\substack{i=1 \\ w^{(1)}_{ji} \geq 0}}^{n} w^{(1)}_{ji} x^{U}_{pi} + \sum_{\substack{i=1 \\ w^{(1)}_{ji} < 0}}^{n} w^{(1)}_{ji} x^{L}_{pi} + \theta_{j}. \tag{5.21}$$

Output nodes:

$$\tilde{y}_{pk} = [f(net^{L}_{pk}), f(net^{U}_{pk})], \quad k = 1, \ldots, m. \tag{5.22}$$

$$net^{L}_{pk} = \sum_{\substack{j=1 \\ w^{(2)}_{kj} \geq 0}}^{l} w^{(2)}_{kj} z^{L}_{pj} + \sum_{\substack{j=1 \\ w^{(2)}_{kj} < 0}}^{l} w^{(2)}_{kj} z^{U}_{pj} + \theta_{k}. \tag{5.23}$$

$$net^{U}_{pk} = \sum_{\substack{j=1 \\ w^{(2)}_{kj} \geq 0}}^{l} w^{(2)}_{kj} z^{U}_{pj} + \sum_{\substack{j=1 \\ w^{(2)}_{kj} < 0}}^{l} w^{(2)}_{kj} z^{L}_{pj} + \theta_{k}. \tag{5.24}$$

where the weights $w^{(1)}_{ji}, w^{(2)}_{kj}$ and the biases θj , θk are real parameters and the outputs $\tilde{z}_{pj,}$ and $\tilde{y}_{pk}$ are intervals. It is noted that the VNN can also process scalar-value input data by setting $x^{L}_{pi} = x^{U}_{pi} = x_{pi}$, where is x_{pi} the scalar-value input. Correspondingly, the VNN can produce scalar output, $y^{L}_{pi} = y^{U}_{pi} = x_{pk}$.

5.4.1 Supervised Learning Algorithms for VNN

In this section, we shall derive a new vector-type backpropagation (NVTBP) learning algorithm for the proposed VNN with interval-value input data. A normal cost function in the conventional vector-type backpropagation (CVTBP) algorithm, E_{pk} , is first considered using the interval output. $\tilde{y}_{pk} = [y^L_{pk}, y^U_{pk}]$ and the corresponding desired output *dpk* for the *p*th input pattern, as

$$E_{pk} = \begin{cases} (d_{pk} - y^L_{pk})^2/2, & \text{if } d_{pk} = 1 \\ (d_{pk} - y^U_{pk})^2/2, & \text{if } d_{pk} = 0 \end{cases} \tag{5.25}$$

for the case of an interval-value input vector and a crisp desired output, where the subscript *pk* represents the *p*th input pattern and *k*th output node. However, to enhance the identification power of VNN for emitter identification, we propose a NVTBP algorithm, which uses a new error cost function instead of the squares of the differences between the actual interval output $\tilde{y}_{pk}$ and the corresponding desired output *dpk* as in Eq. (5.25).

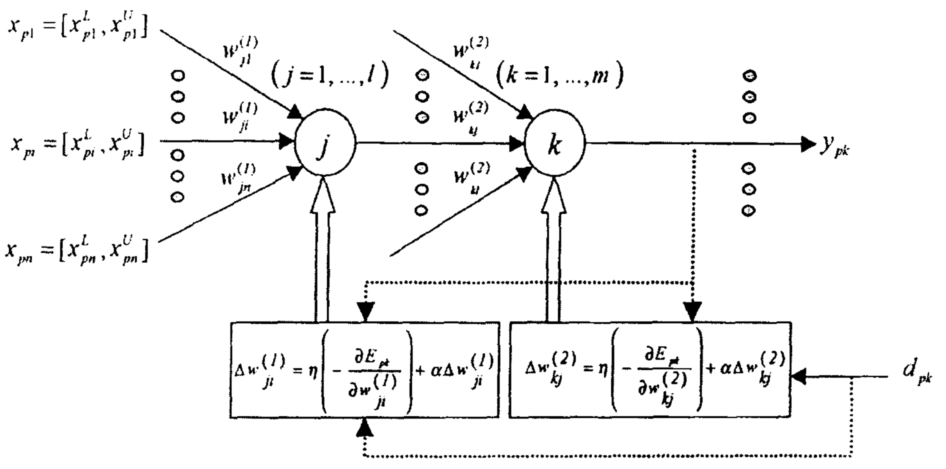

$x_{p1} = [x_{p1}^L, x_{p1}^U]$
$x_{pi} = [x_{pi}^L, x_{pi}^U]$
$x_{pn} = [x_{pn}^L, x_{pn}^U]$
$w_{j1}^{(1)}$
$w_{ji}^{(1)}$
$w_{jn}^{(1)}$
$(j = 1, \dots, l)$
j
$w_{k1}^{(2)}$
$w_{kj}^{(2)}$
$w_{kl}^{(2)}$
$(k = 1, \dots, m)$
k
y_{pk}
$\Delta w_{ji}^{(1)} = \eta \left(-\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} \right) + \alpha \Delta w_{ji}^{(1)}$
$\Delta w_{kj}^{(2)} = \eta \left(-\frac{\partial E_{pk}}{\partial w_{kj}^{(2)}} \right) + \alpha \Delta w_{kj}^{(2)}$
d_{pk}

Figure. 5.7. Illustration of backpropagation learning rule for the VNN.

The new error cost function is defined as

$$E_{pk} \quad \begin{cases} -d_{pk} \ln y_{pk}^L - 1 + d_{pk}) \ln(1 - y_{pk}^L).\, d_{pk} = 1 \\ -d_{pk} \ln y_{pk}^U - 1 + d_{pk}) \ln(1 - y_{pk}^U).\, d_{pk} = 0 \end{cases} \tag{5.26}$$

The learning objective is to minimize the error function in Eq. (5.26). The weight updating rules for the VNN are illustrated in Fig. 5.7. To show the learning rules, we shall calculate the computation of $\partial E_{pk} / \partial w$ layer by layer along the dashed lines in Fig. 5.6, and start the derivation from the output nodes.

Layer 3 : Using Eqs. (5.22)-(5.24) to calculate $\partial E_{pk} / \partial w_{kj}^{(2)}$ for various values of the weights and desired output.

Layer 2 : Using Eqs. (5.19)-(5.21) to calculate $\partial E_{pk} / \partial w_{ji}^{(1)}$ for different values of the weights and desired output. Notice that, the notations $\delta_{pk}^{\prime L}$ and $\delta_{pk}^{\prime U}$ are defined as follows:

$$\delta_{pk}^{\prime L} \triangleq \frac{\partial E_{pk}}{\partial net_{pk}^{L}} = (d_{pk} - y_{pk}^{L}) \ . \tag{5.27}$$

$$\delta_{pk}^{\prime U} \triangleq \frac{\partial E_{pk}}{\partial net_{pk}^{U}} = (d_{pk} - y_{pk}^{U}) \ . \tag{5.28}$$

A detailed description of quantitative analysis can be found in [81].

5.4.2 Simulation Results

All reference data of the simulated emitters are given by references [71, 72], and [82]. Two problems are examined to demonstrate the identification capability of the proposed VNN in this section.

(1) *Performance Evaluation without Measurement Error*

Experiment 1: For the two-emitter identification problem, we employ a VNN with 3 input nodes, 5 hidden nodes and 2 output nodes (denoted by 3-5-2 network). In the training phase, we use 10 input-output training pairs (5 pairs for each type) to train the VNN using the CVTBP and NVTBP algorithms, respectively, and find individually a set of optimal weights. In the testing phase, 80 testing patterns (40 patterns for each emitter type) are presented to the trained VNNs for performance testing. However, the VNN trained by the NVTBP algorithm performs better than the VNN trained by the CVTBP algorithm; the former achieves an average identification rate of 99.91% and the latter 96.26% as listed in the last row of Table 5.2.

Experiment 2: In this experiment, a three-emitter identification problem is solved by two 3-8-3 VNNs trained by the NVTBP and CVTBP algorithms, respectively. The 15 input-output training pairs (5 pairs for each type) are used to train the two VNNs. After training, 120 testing patterns (40 patterns for each emitter type) are presented to the trained VNNs for performance testing.

(2) *Performance Evaluation with Measurement Error*

Two experiments are performed to evaluate the robustness of VNN with measurement distortion. At first, we define the error deviation level (EDL) by

$$\mathrm{EDL}_i\ (\%) = \frac{\xi_{pi}}{x_{pi}} \times 100\%,\ i = 1, 2, 3, \tag{5.29}$$

for emitter type i, where ξpi is a small randomly generated deviation for the pth input pattern.

Experiment 3: First, we consider the two-emitter identification problem with the input data corrupted by additive noise. The noise testing patterns are obtained by adding random noise, ξpi (i = 1, 2, 3), to each testing pattern ($xp1$, $xp2$, $xp3$), p = 1, . . . , 80, used in Experiment 1 to form the noise testing database, ($xp1 \pm \xi p1$, $xp2 \pm \xi p2$, $xp3 \pm \xi p3$). The noisy testing patterns with different EDLs (from 1 % to 15 %) are presented to the trained 3-5-2 VNNs in Experiment 1 for performance testing. The testing results are shown in Table 5.2.

Experiment 4: In this experiment, we consider the three-emitter identification problem with the input data corrupted by additive noise. The noise testing patterns are obtained by adding random noise, ξpi (i = 1, 2, 3), to each testing pattern ($xp1$, $xp2$, $xp3$), p = 1, . . . , 120, used in Experiment 2 to form the noise testing database, ($xp1 \pm \xi p1$, $xp2 \pm \xi p2$, $xp3 \pm \xi p3$). The noisy testing patterns with different EDLs are presented to the trained 3-8-3 VNNs in Experiment 2 for performance testing. The testing results are shown in Table 5.3.

Table 5.2 Testing results of the 3-5-2 VNN on the two-emitter identification problem with/without noise.

Error Deviation Level (%)	3-5-2 VNN trained by the NVTBP algorithm			3-5-2 VNN trained by the CVTBP algorithm		
	Average Correction Rate (%)		Total Average Correction Rate (%)	Average Correction Rate (%)		Total Average Correction Rate (%)
	Type 1	Type 2		Type 1	Type 2	
15	99.54	99.87	99.71	88.08	94.01	91.04
13	99.92	99.88	99.90	93.08	94.42	93.75
11	99.94	99.88	99.91	95.07	94.63	94.85
9	99.94	99.88	99.91	96.19	94.80	95.49
7	99.94	99.88	99.91	96.74	94.93	95.83
5	99.94	99.88	99.91	97.03	95.03	96.03
3	99.94	99.88	99.91	97.19	95.11	96.15
1	99.94	99.88	99.91	97.29	95.17	96.23
0	99.94	99.88	99.91	97.31	95.21	96.26

5.5 Conclusions

A novel neural fuzzy network (NFN) is proposed to estimate the direction of arrival of moving targets based on the phase differences from an interferometer. The main contribution of the proposed NFN is that it always produces an economical networks size, and the learning speed and modeling ability are superior to ordinary neural networks. In this report, we use two networks, RBFN and NFN, to estimate DOA and compare their performance under either with noise or without noise conditions.

Simulation results show that the NFN always produces actual output very close to the desired DOA values, and the required number of parameters in the NFN is less than that in the RBFN under the same RMS error in DOA. In addition, we also construct a three-layered vector neural network (VNN) to perform emitter identification. Simulation results show that the proposed VNN can gives high identification capability. With these features, we believe that our proposed networks (NFN/VNN) may be applied for solving the problems of the signal detection, signal localization, signal tracking, and beamforming in the military applications (such as reconnaissance and surveillance).

Table 5.3 Testing results of the 3-8-3 VNN on the three-emitter identification problem with/without noise.

Error Deviation Level (%)	3-8-3 VNN trained by the NVTBP algorithm				3-8-3 VNN trained by the CVTBP algorithm			
	Average Correction Rate (%)			Total Average Correction Rate (%)	Average Correction Rate (%)			Total Average Correction Rate (%)
	Type 1	Type 2	Type 3		Type 1	Type 2	Type 3	
15	63.00	89.39	74.87	75.75	57.89	87.97	70.78	72.21
13	72.20	90.36	74.93	79.16	59.06	89.36	70.88	73.10
11	74.25	92.26	74.95	80.49	60.05	90.27	70.96	73.76
9	79.15	93.78	79.35	84.09	60.85	92.15	75.50	76.17
7	86.35	96.18	85.80	89.44	67.11	93.12	80.52	80.25
5	96.01	97.94	94.16	96.04	75.16	94.04	88.69	85.96
3	99.34	99.30	99.69	99.44	80.56	94.82	92.18	89.19
1	99.60	99.87	99.93	99.80	82.71	95.48	93.70	90.63
0	99.63	99.95	99.94	99.84	83.36	95.80	94.07	91.08

Chapter 6

Conclusion

In last two years, we proposed an algorithm for the target detection from infrared images, and detected the target in two parts. Firstly, the infrared images obtained from an immovable camera would be discussed by using the *image difference*, *run length*, and *image thresholding*. Secondly, detect the target from images obtained from a movable camera; according to the result, the rate of success is higher than 98%. In last year, in order to improve the system efficiency, we proposed another method that based on the velocity differences of the image objects. By this method, the detection correctness rate is more than 98.5%. In this year, we proposed another velocity estimation method to improve the system efficiency. By this method, the detection correctness rate is more than 98.7%.

Compare these two methods, the proposed method is more efficient, the method can deal with multi-targets uses less process time. The SSD method can deal with multi-targets, but it needs too much time. The lightest point method assumes the target is the lightest pixel in the image. However the lightest pixel is not always the target, it might be something else. At the beginning, it must take more than two images to find the real target. In this year we decrease the brightness effect to the image. Due to the optical flow, the nature scene is unexpected, the standard methods may not hold for large displacements. To solve the problem and to improve the process performance an incremental estimation of optical flow based on multiresolution strategy was proposed. Even the target detection is wrong, we still can detect correctly in less one second. In the future, we would determine the distance between the sensor and the object based on perspective transformation and the algorithm should be deal with more targets.

A novel neural fuzzy network (NFN) is proposed to estimate the direction of arrival of moving targets based on the phase differences from an interferometer. The main contribution of the proposed NFN is that it always produces an economical networks size, and the learning speed and modeling ability are superior to ordinary neural networks. In this report, we use two networks, RBFN and NFN, to estimate DOA and compare their performance under either with noise or without noise conditions. With Simulation results, we believe that our proposed networks (NFN/VNN) may be applied for solving the problems of the signal detection, signal localization, signal tracking, and beamforming in the military applications (such as reconnaissance and surveillance).

Bibliography

[1] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Reading, MA: Addision Wesley, 1993.

[2] T. I. Hentea, “Algorithm for automatic threshold determination for image segmentation,” *Canadian Conf. On Electrical and Computer Engineering*, Vamcouver, Canada, vol. 1, pp. 535-538, 1993.

[3] Y. Zou and W. T. M. Dunsmuir, “Generalized max/median filtering,” *Proc. Intl Conf. Image Processing*, pp.428-431, Santa Barbara, California, 1997.

[4] T. S. Huang, G. J. Yang, “A fast two-dimensional median filtering algorithm,” *IEEE Trans. Acoust, Speech, Signal Processing*, vol. Assp-27, no. 1, pp.13-18, 1979.

[5] T. R. Benedict and G. W. Bordner, “Synthesis of an optimal set of radar track-while-scan smoothing equations, ”*IEEE Trans. Image Processing,* vol.AC-7, pp.27-32,July 1962.

[6] S. S. Blackman, *Multiple Target Tracking with Radar Applications*, Norwood, MA: Artech House, 1986.

[7] C. B. Chang and J. A. Tabaczynski, "Application of state estimation to target tracking, "*IEEE Trans. Automat. Contr*, vol. 29, pp.98-109, Feb 1984.

[8] Y. -L. Chang and X. Li, "Adaptive image region-growing, "*IEEE Trans. Imaeg Processing*, vol. 3, no. 6, pp. 868-872, 1994.

[9] D. Chetiverikov and J. Verestoy, "Tracking feaure points: a new algorithm, "*Proc. Fourteenth Int'l Conf. Patttern Recognition*, vol. 2, pp. 1436-1438, Brisbane, Australia 1998.

[10] K. S. Fu, R. C. Gonzalez and C.S.G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, New York: McGraw-Hill, 1987.

[11] V. S. Hwang, "Tracking feature points in time-varying images using an opportunistic selection approach, " *Pattern Recognition*, vol. 22, no. 3, pp. 247-256, 1989.

[12] C. T. Lin and C. S. Lee, *Neural Fuzzy System*, New Work: Prentice-Hall, 1996.

[13] R. Mehrotra, "Establishing motion-based feature point correspondence, "*Pattern Recognition*, vol. 31, no. 1, pp. 23-30, 1998.

[14] D. Murray and A. Basu, "Motion tracking with an active camera, " *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 449-459, 1994.

[15] M. A. Rahgozar and J. P. Allebach, “Motion estimation based on time-sequentially sampled imagery, ”*IEEE Trans. Image Processing*, vol. 4, no. 1, pp. 48-65, 1995.

[16] K. Rangarajan and M. Shah, “Establishing motion correspondence, ” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 56-73, Lahaina, Maui, Hawaii, 1991.

[17] V. Salari and I. K. Sethi, “Feature point correspondence in the presence of occlusion, ” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 87-91, 1990.

[18] I. K. Sethi and R. Jain, “Finding trajectories of feature points in a monocular image sequence, ” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 56-73, 1987.

[19] J. –Y Shieh, H. Zhuang, and R. Sudhakar, “Motion estimation from a sequence of stereo images: a direct method, ” *IEEE Trans. System, Man, Cybern*, vol. 24, no. 7, pp. 1044-1053, 1994.

[20] J. Weng, P. Cohen, and N. Rebibo, “Motion and structure estimation from stereo image sequence, ” *IEEE Trans. Robotics. Robotics and Automation*, vol. 8, no. 3, pp. 362-382, 1992.

[21] T. C. Wang and P. K. Varshney, “A tracking algorithm for maneuvering targets, ”

IEEE Trans. Aerosp. Electron. Syst, vol. 29, pp. 910-924, July 1993.

[22] J. M. White, and G. D. Rohrer, “Image Thresholding for Optimal Character Recognition and Application Requiring Character Image Extraction, ” *IBM, J. Research Devel*, vol. 27, no. 4, pp. 400-411, 1983.

[23] S. B. Xu, “ Qualitative depth from monoscopic cues, ” *Image Processing and its Application*, pp. 437-440, 1992.

[24] Y. Jae-Woong and O. Jun-Ho, “Estimation of depth and 3D motion parameter of moving object with multiple stereo images, ” *Image And Vision Computing*, vol. 14, no. 7, pp. 501-516, 1996.

[25] B. Y. Tsui, “Microwave Receivers with Electronic Warfare Applications, ” New York,Wiley,1986.

[26] M. H. Hayes, “Statistical Digital Signal Processing and Modeling, ” New York, John Wiley and Sons, Inc.1996.

[27] B. Widrow and S. D. Stearns, “Adaptive Signal Processing,” rentice-Hall. Inc. 1985.

[28] S. M. Kay and S. L. Marple, Jr. , “Spectrum analysis-A modern perspective, ” *Proc. IEEE*, vol.69, pp.1380-1419, Nov.1981.

[29] D. H. Johnson and S. R. DeGraaf, “Improving resolution of bearing in passive sonar ar-rays by eigenvalue analysis, ” *IEEE Trans. Acoust. Speech Signal*

Processing, vol.ASSP-30,Aug.1982.

[30] R. O. Schmidt, "Multiple emitter location and signal parameter estimation, " *IEEE Trans. Antennas Propagat*, vol.Ap-34, no.3, pp.276-280, Mar.1986.

[31] A. Paulraj and T. Kailath, "Eigenstructure methods for direction of arrival esti-mation in the presence of unknown noise fields, " *IEEE Trans. Acoust., Speech, Signal Processing* vol.ASSP-34,Feb.1986.

[32] R.Roy and T.Kailath, "ESPRIT-estimation of signal parameters via rotational invari-ance techniques, " *IEEE Trans. Acoust., Speech, Signal Processing,* vol.37,no.7 pp.984-995,July 1989.

[33] S.Jha and T.S.Durrani, "Direction of arrival estimation using artificial neural net -works, " *IEEE Trans. Syst. Man Cybarn,* vol.21, no.5, pp.1192-1201, Sept./ Oct.1991.

[34] H. L. Southall, J. A. Simmers, and T. H. O'Donnell, "Direction finding in phased arrays with a neural network beamformer, " *IEEE Trans. Antennas Propagat.*, vol.43,no.12, pp.1369-1374, Dec.1995.

[35] J. Park and I. W. Sandberg, "Universal approximation using radial basis function net-works, " *Neural Computa.*, vol.3,pp.246-257,1991.

[36] T. Lo , L. Henry, and L. John, "Radial basis function neural network for direction of arrivals estimation, " *IEEE Signal Processing Letters*, vol.1, no.2, pp.45-47,

Feb.1994.

[37] A. H. El Zooghby, C. G. Christodoulou, and M. Georgiopoulos, "Performance of radial-basis function networks for direction of arrival estimation with antenna arrays, "*IEEE trans. Antennas Propagat.,* vol.45, no.11, pp.1611-1617, Nov.1997.

[38] S. Chen, C. F. N.Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial-basis function networks," *IEEE Trans. Neural Networks,* vol.2, no.2, pp.302-309, Mar.1991.

[39] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its application, " *IEEE Trans. Fuzzy Systems*, vol.6, no.1, pp.12- 32, Feb.1998.

[40] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs,N.J., 1982.

[41] B. K. P Horn and B. Schunck. "Determining optical flow." *Artificial Intelligence*, 17: pp185-203, 1981.

[42] N. Cornelius and T. Kanade, " Adapting optical flow to measure object motion in reflectance and x-ray image sequence. " *In Proc. ACM Siggraph/Sigart Interdisciplinary Workshop on Motion*, Toronto, pp. 50-58, 1983.

[43] H. H. Nagel, "Displacement vectors derived from second order intensity variations in image sequences." *Computer Vision, Graphics and Image*

Processing, 21:pp85-117, 1983.

[44] H. H. Nagel, "On the estimation of dense displacement maps from image sequence. " In *Proc. ACM Motion Workshop*, Toronto, pp59-65, 1983.

[45] W. B. Thompson and S. T. Barnard, "Lower level estimation and interpretation of visual motion, " *Computer*, 20(8):20-28, 1987.

[46] J. V. Beck and K. J. Arnold. *Parameter estimation in engineering and science.* John Wiley, New York, 1977.

[47] D. C. Schleher, *Introduction to Electronic Warfare*, New York, Artech House, Inc., 1986.

[48] G. B. Willson, "Radar classification using a neural network," *Applications of Artificial Neural Networks, SPIE*, Vol. 1294, pp. 200-210, 1990.

[49] I. Howitt, "Radar warning receiver emitter identification processing utilizing artificial neural networks," *Applications of Artificial Neural Networks, SPIE,* Vol. 1294, pp. 211-216, 1990.

[50] K. Mehrotra, C. K, Mohan, and S. Ranka, *Elements of Artificial Neural Networks*, Cambridge, MA: MIT press, 1997.

[51] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, Cambridge, MA: MIT press, 1995.

[52] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets and classification,"

IEEE Trans. Neural Networks, Vol. 3, No. 5, pp. 683-696, Feb. 1992.

[53] J. M. Keller and H. Tahani, "Backpropagation neural networks for fuzzy logic," *Inform. Sci.*, Vol. 62, pp. 205-221, 1992.

[54] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm," *IEEE Trans. Neural Networks,* Vol. 3, No. 5, pp. 801-806, 1992.

[55] H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy if-then rules," *IEEE Trans. Fuzzy Syst.*, Vol. 1, No. 2, pp. 85-97, May 1993.

[56] Golland, P., Optical Flow Estimation using Color Images, Master Thesis, Artificial Intelligence Lab, MIT, 1995.

[57] Barron, J., D. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, Vol.12, NO.1, pp.43-77,1994.

[58] Nagel, H. H.,"Displacement vectors derived from second-order intensity variations in image sequences," *Computer Vision and Image Understanding*, Vol.65, No.2, pp.259-268, 1997.

[59] Lucas, B. and T. Kanade, "Optical navigation by the method of differences" in *Proc. 7th Int. Joint Conf. Artificial Intelligence*, 1985, pp.981-984.

[60] Irani, M., B. Rousso, and S. Peleg "Computing occluding and transparent

motions" International Journal of Computer Vision, Vol.12, No.1,pp.5-16, 1994.

[61] Kenner, M. and T.-C. Pong, "Motion analysis of long image sequence flow," Pattern Recognition Letters, Vol.11, No.1,pp.123-131,1990.