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(Ⅱ)

計畫編號：NSC-90-CS-7-009-005

執行期間：90 年 1 月 1 日至 90 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 90 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(Ⅱ)

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(Ⅱ)

Professor：Sheng Fuu Lin 林昇甫

NSC-90-CS-7-009-005

Institute of Electrical and Control Engineering

National Chiao Tung University

Hsinchu, Taiwan, Republic of China

Dec 31. 2001

參與計畫人員

相對主持人：	金傳文
研究人員　：	林昇甫
	林進燈
	陳韋酉
	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。上年度工作重點是從攝影機所擷取的影像中判斷是否有來襲目標物及基本引導邏輯的研究。在上年度中我們完成了，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。在今年度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。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 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 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. At this year, 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.

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

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.

Since the weather condition is a key for the imagery target detection, we will try to find a suitable methodology which can reduce the weather effect.

1.2 Organization of the Report

The following is a brief description of the organization of this reports. In Chapter 2, some image processing techniques including the image thresholding, median filter and region growing, image difference. Furthermore, to consider the location of the desired target in the image and to detect from the raw images is presented in Chapter 3. In order to improve the efficiency of the detection, we use another method based on the velocity field 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 summarized 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 included in the trajectory of the target

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. Median filter is introduced in Section 2.1.2. Low pass filter is introduced in Section 2.1.3. Image difference is introduced in Section 2.1.4.

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

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

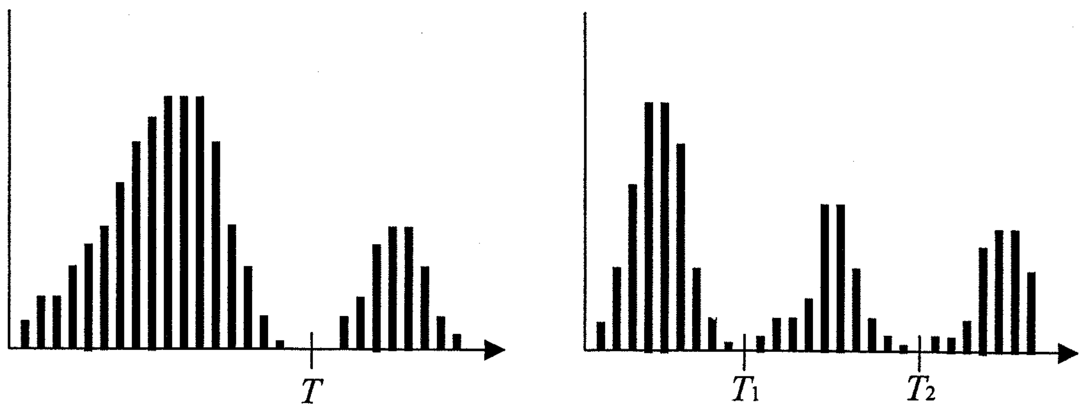


Fig. 2.1 Gray-level histograms that can be partitioned by (a) a single threshold. (b) multiple thresholds.

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, 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 times 3 neighborhood the median is the 5th largest value, in a 5 times 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 times 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 times 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

1/9 ×

1	1	1
1	1	1
1	1	1

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 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,

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

which $image1(x,y)$ means the gray value of the former image,

$image2(x,y)$ 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：

if $Dif(x,y) < 20$, then $Dif(x,y) = 0$, 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：

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

If $Result(x,y) \neq 0$, 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.

Chapter 3

Targets Detection from Images

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.

3.1 Movable Camera

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. Judge whether it is the object.

3.1.1 Search for Possible Targets

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,

SRH is the height of the searching range.

3.1.2 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.

3.1.3 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,

$SSRH$ is the height of the small searching range.

3.1.4 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 position 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. 3.1 shows that we do not know it is a broken point of the video section, and Fig. 3.2 shows that we know it is a broken point of the video section then change to use the whole image range.

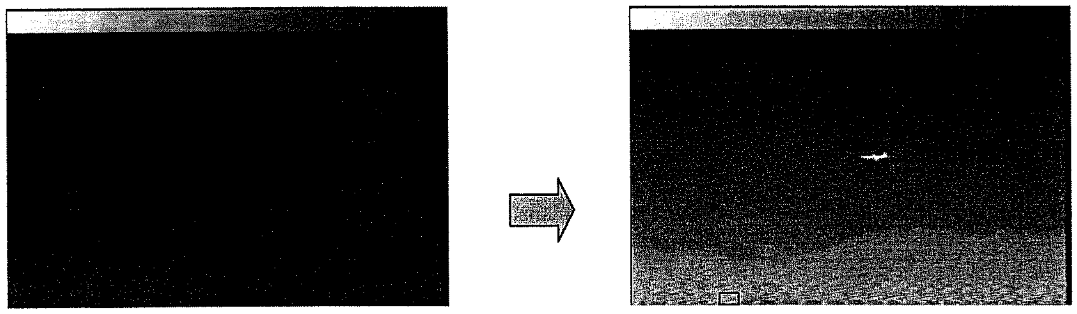


Fig. 3.1 Only using the small range searching.

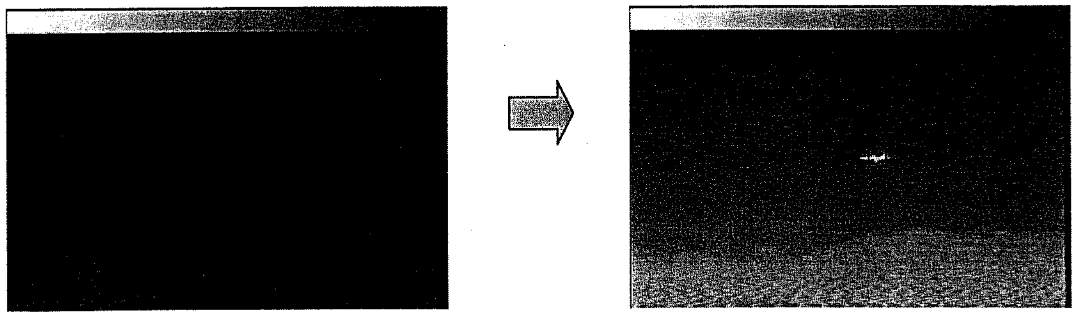


Fig. 3.2 Decide it is a broken point of the video section then change to use the whole image range searching.

3.1.5 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. Fig.3.3 shows some images of the result of the experiment. We obtain the series of images from the original video sequence per 1/30 sec. In all, we obtain 7290 images. By our 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 whole detection efficiency in the next chapter.

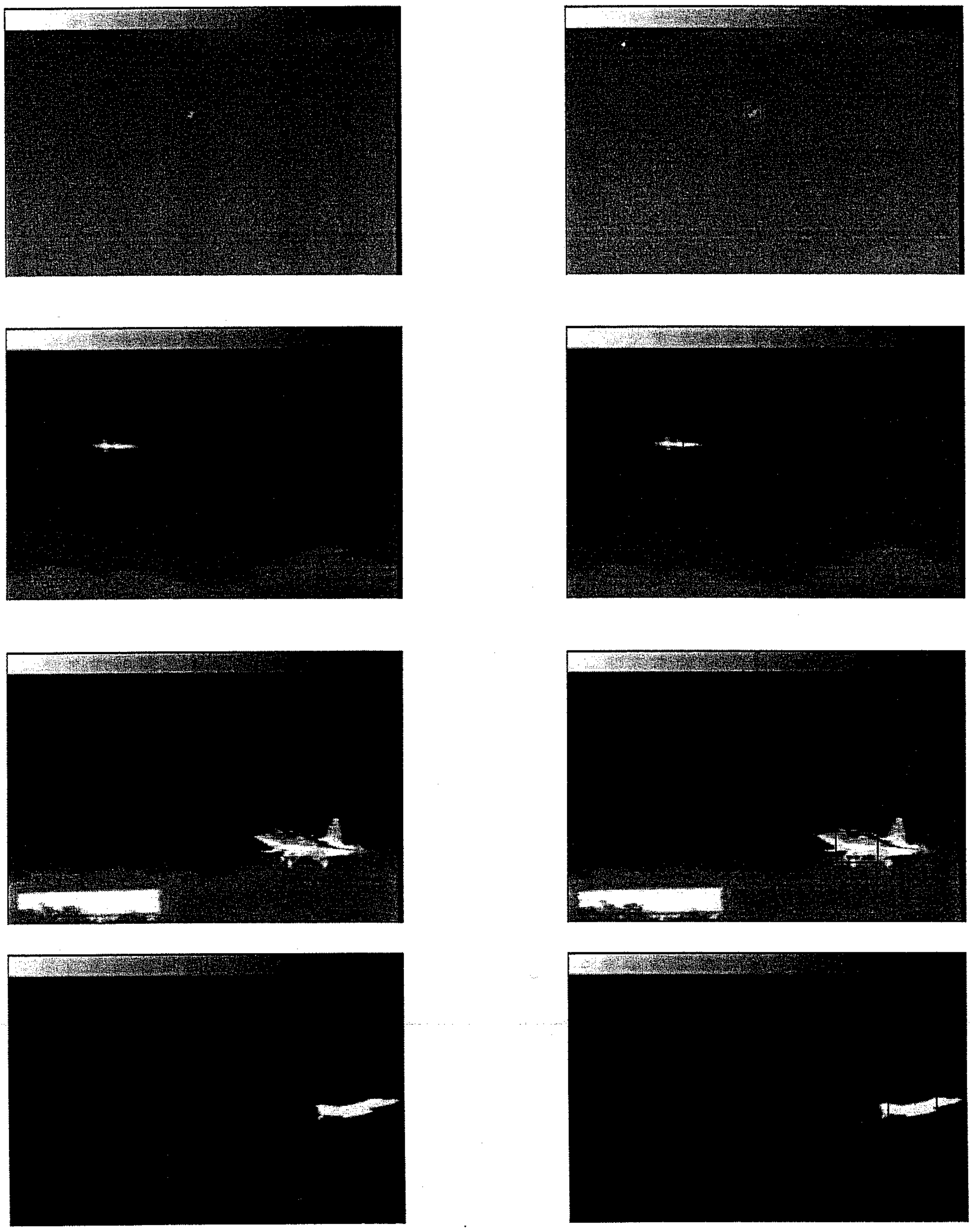


Fig. 3.3 Some final image results

Chapter 4

Target Detection from Velocity Estimation

From Chap3, 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. We hope this algorithm would be more efficient and practical in target detection.

4.1 Image Flow and Optic Flow

An image-flow field depicts, at each point on the imaging surface, a 2D projection of the instantaneous 3D velocity of the corresponding in the scene. The scenario in Fig 4.1 can be used to explain this definition. A rigid body B is undergoing an arbitrary motion relative to the imaging surface. A point $\boldsymbol{P}$ on the rigid body has a velocity **S** with respect to a *world coordinate system* (X,Y,Z) fixed at the origin O_w. The imaging surface, a plane in this case, is fixed with respect to the world coordinate system in such a way that O_w is the center of projection, or the viewpoint [40]. An *image coordinate system* (x,y) is attached to the image plane with its origin at O_i. The point *p* on the image plane is the projection of the scene point ***P***. Likewise, the vector $\mathbf{V} = (u,v)$ in the image plane, originating at *p*, is the projection of the velocity vector **S**. By the definition given above, the vector **V** is the image-flow vector at the point *p*.

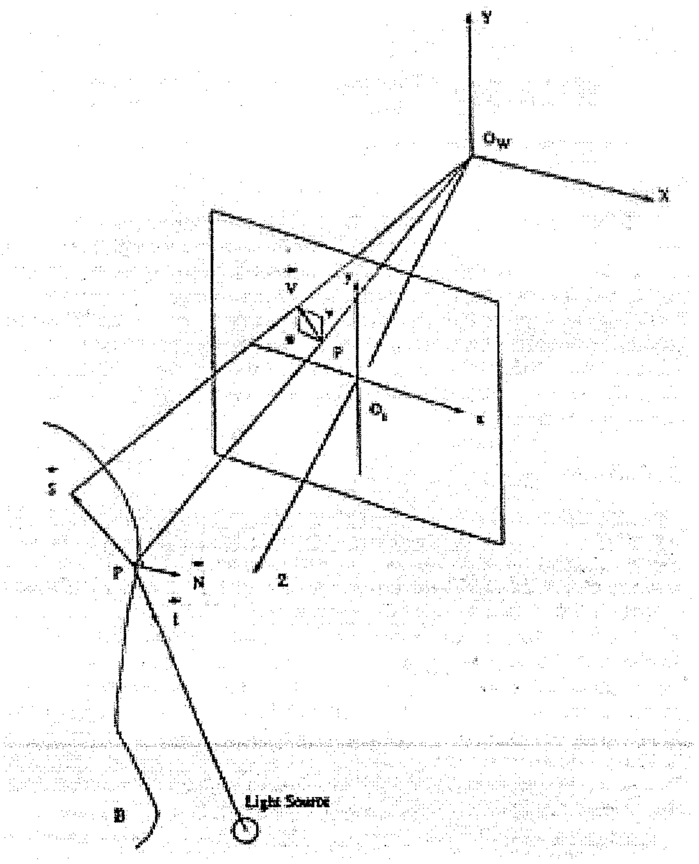


Fig. 4.1 The imaging geometry.

The *optic-flow field* is the 2D distribution of *apparent velocities* that can be associated with the variation of brightness patterns on the image [41]. When the light source in the image is nonuniform moving, the optic-flow fiel is nonzero. Since the scene is stationary, the image flow will be zero throughout the image. However, since the light source is moving, the brightness patterns on the image will result a nonzero optic-flow field. As the definitions imply, image flow and optic flow are generally not equal

4.2 Two Approaches to Optic-Flow Estimation

4.2.1 Gradient-Based Approach

Techniques based on the gradient-based approach [41,42,43,44,45] typically work on the assumption of *conservation of image intensity*. These techniques assume that for a given scene point the *intensity* I at the corresponding image point remains constant over time. That is, if a scene point projects onto the image point (x, y) at time t and onto the image point $(x+\delta x, y+\delta y)$ at time $(t+\delta t)$, we can write

$$I(x,y,t) \cong I(x+\delta x, y+\delta y, t+\delta t)\,. \tag{4.1}$$

Expanding the right-hand side by a Taylor series about (x, y, t) and ignoring the second and higher order terms, we obtain

$$I(x+\delta x, y+\delta y, t+\delta t) = I(x,y,t) + \delta x\frac{\partial I}{\partial x} + \delta y\frac{\partial I}{\partial y} + \delta t\frac{\partial I}{\partial t}\,. \tag{4.2}$$

Combing the two equations results in the following expression：

$$\delta x\frac{\partial I}{\partial x}+\delta y\frac{\partial I}{\partial y}+\delta t\frac{\partial I}{\partial t}=0. \tag{4.3}$$

Dividing throughout by δt ; (u,v) is the local velocity vector, I_x, I_y and I_t is the partial derivatives.

$$I_x u+I_y v+I_t=0. \tag{4.4}$$

The equation (4.4) denotes the motion constraint line, and it can be plotted in *uv* space.

4.2.2 Correlation-Based Approach

Correlation-based techniques typically use the assumption of *conservation of local intensity distribution.* In essence, for each pixel in the first image, they search for a matching pixel in the second image. The output is a displacement vector for each pixel in the first image. Typically, such a search involves finding the *best match* for the pixel under consideration among some interesting pixels in the second image. If the best match can be found uniquely, the exact displacement vector is immediately known.

4.3 Detection of the target

Fig 4.2 shows some images of the series video. From above statements, we can calculate velocity field of the image by correlation-based approach method. And we can get lots of information from the velocity field. So we propose this method to solve this problem, it can be explained as the following steps：

1. Get the image intensity
2. Compute error distribution and response it
3. Find the target from velocity field
4. Check the computing range
5. Decide the target is real or not

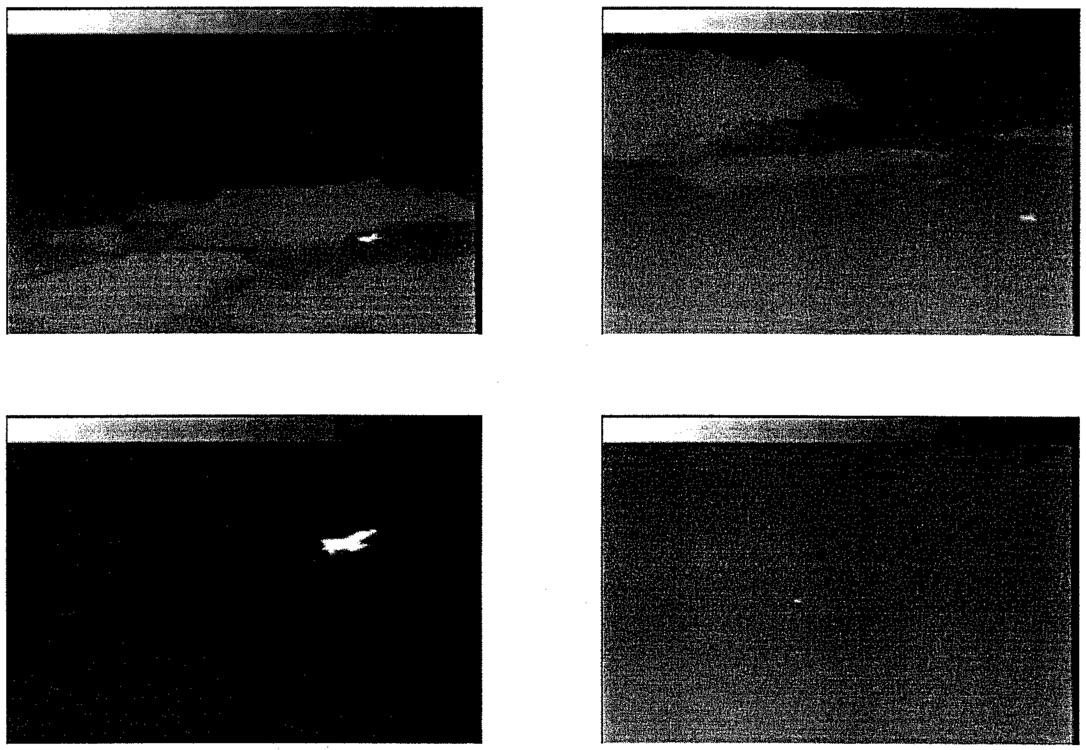


Fig. 4.2 Some images of the series video.

4.3.1 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：

$$Intensity(x,y) = \tfrac{1}{3}[image(x,y) + image(x+1,y) + image(x+2,y)]. \qquad (4.5)$$

4.3.2 Computation of the Error Distribution

As discussed in the qualitative description, we use sum of squared differences (SSD) as the error function. For simplicity, we use only two images, I_1 and I_2, to compute error distribution and response it over the search area. For each pixel $P(x,y)$ at location (x,y) in the image I_1, we form a window W_p of size 3×3 around the pixel. We establish the search window W_s of size 5×5 around the pixel at location (x,y) in the image I_2. The 5×5 sample of error distribution, whose elements represent the dissimilarity between W_p and a 3×3 window around each pixel in W_s, is computed as

$$E_c(u,v) = \sum_{i=-1}^{1}\sum_{j=-1}^{1}(I_1(x+i,y+j) - I_2(x+u+i,y+v+j))^2,$$

$$u,v = -2,-1,0,1,2. \qquad (4.6)$$

In this expression, $I_1(x,y)$ and $I_2(x,y)$ refer to the intensities at location (x,y) in images I_1 and I_2, respectively. The 5×5 sample of response distribution, whose elements represent the similarity between W_p and a 3×3 window around each pixel in W_s, is computed as the exponential of the negated error. That is,

$$R_c(u,v) = e^{-k}[\sum_{i=-1}^{1}\sum_{j=-1}^{1}(I_1(x+i,y+j) - I_2(x+u+i,y+v+j))^2],$$

$$u,v = -2,-1,0,1,2\,. \tag{4.7}$$

The choice of an exponential function for converting error distribution into response distribution is based primarily on computational considerations. First, it is well-behaved when error approaches zero. A function that uses, for instance, the reciprocal of error, tends to infinity as error approaches zero. Therefore, it is computationally harder to manipulate. Secondly, the response obtained with an exponential function varies continuously between zero and unity over the entire range of error. We use $k = 10^{-4}$ in this method to make the response be the positive value, and the total loop times in one image are $(u2 - u1 + 1) \times (v2 - v1 + 1)$ where $(u2, u1)$ is the range of the $P(x, y)$ in X direction, and $(v2, v1)$ is the range of it in Y direction.

4.3.3 Velocity Field

From the above Fig4.3 (c), we can find the best match pixel from the maximum response pixel, but we still can't know whether it is the target we need or not. Therefore we still have to calculate the velocity field to segment what we need. And

20	30	121
16	27	119
81	79	111

(a)

4	6	8	9	16	19	22
5	6	8	10	12	20	22
11	11	10	23	32	124	126
14	15	16	18	26	121	121
58	60	65	82	83	116	117
98	100	105	110	112	123	124
100	104	110	110	112	124	124

(b)

9	12	18	63	82
11	17	26	154	78
67	107	162	987	143
171	194	180	369	96
157	153	106	87	39

(c)

Fig. 4.3 An illustration of response distribution computation of one example : (a) the 3×3 window W_p. (b) the 5×5 window W_s (a one-pixel-wide annulus around this window is shown because it is required to compute the correlation at boundary pixels). (c) the 5×5 response distribution($\times10^3$).

then 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.

A very important step in the current framework is interpreting the response distribution. Each point (u,v) in the search window (in uv space) is a candidate for the true velocity. However, a point with a small response is less likely to represent the true velocity than is a point with a high response. Thus, response distribution could be interpreted as a frequency distribution in velocity space — the response at a point depicting the frequency of occurrence, or the likelihood, of the corresponding value of velocity. This interpretation lets us use a variety of estimation-theoretic techniques to compute velocity and associate a notion of confidence with it.

Specifically, the quantity we are trying to compute is the true velocity(u_t, v_t). With the interpretation given above, we know the frequency of occurrence $R_c(u,v)$ of various values of velocity $(u,v)=(u_t,v_t)+(e_u,e_v)$ over the search area. The quantity (e_u,e_v) is the error associated with the point (u,v), that is, its deviation from the true velocity. We can obtain an *estimate* of the true velocity using a weighted least-squares approach [46]. This estimate, denoted by $U_{cc}=(u_{cc},v_{cc})$, is given by

$$u_{cc}=\frac{\sum_u\sum_v R_c(u,v)u}{\sum_u\sum_v R_c(u,v)} ,$$

$$v_{cc} = \frac{\sum_u \sum_v R_c(u,v)v}{\sum_u \sum_v R_c(u,v)} , \tag{4.8}$$

where the summation is $u,v = -2,-1,0,1,2$. For example, the u_{cc} and v_{cc} corresponding to the response distribution of Fig 4.3(c) is $(0.35,-0.42)$.

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.

$$diff = \sum_{i=-1}^{1} \sum_{j=-1}^{1} [U(x+i, y+j) - mean\_x]^2 + [V(x+i, y+j) - mean\_y]^2 \tag{4.9}$$

where $U(x, y)$ is the velocity in X direction of I_1(x,y) ; $V(x, y)$ is in Y direction, and $mean\_x$ is the mean velocity in X direction; $mean\_y$ is in Y direction. Fig 4.4 shows the velocity fields of some images.

4.3.4 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 range this way, we set that the center is $I_1(x, y)$, it is the target position, and one 5×5 window 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 equation (4.10). Some velocity fields as follows：

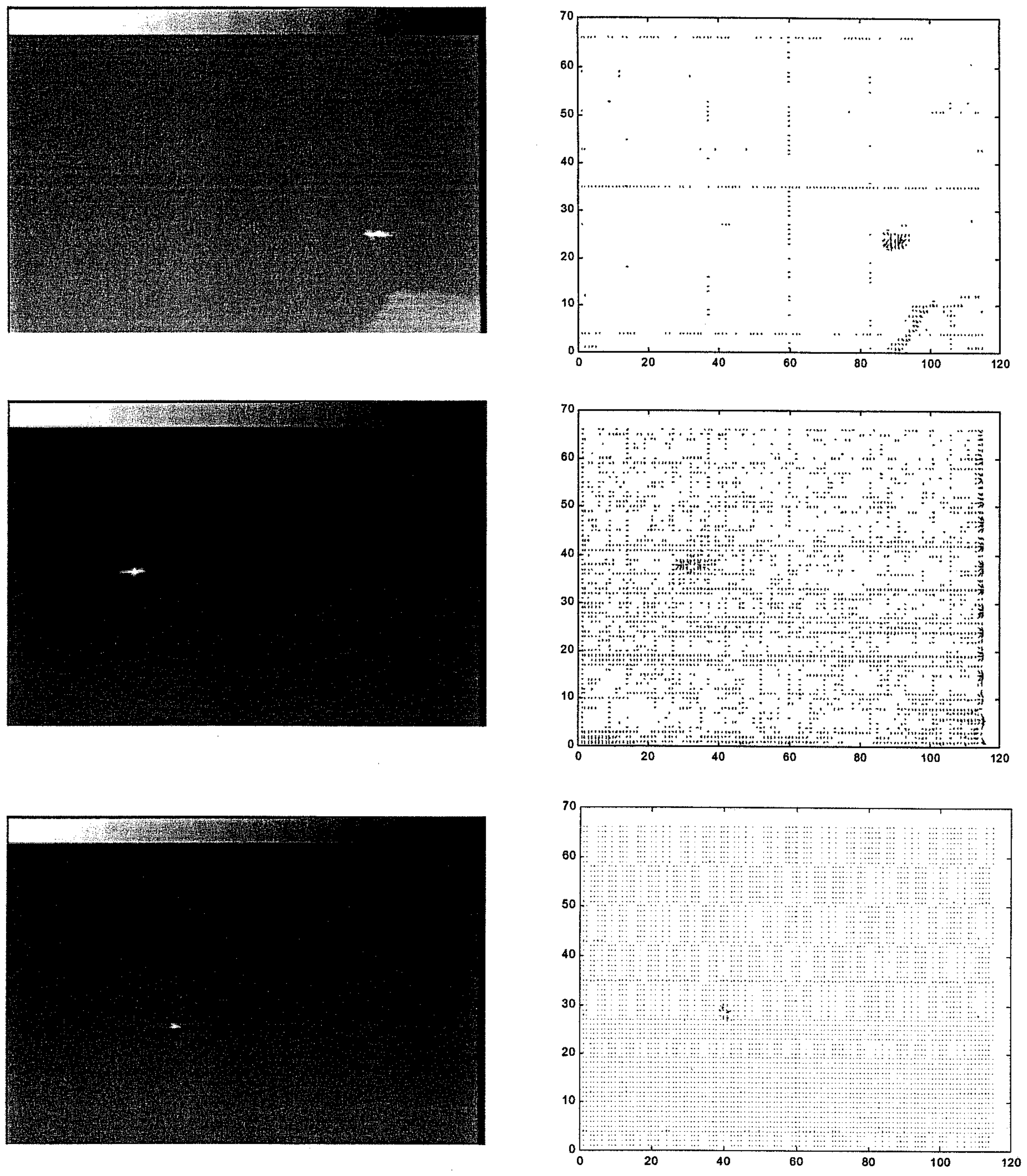


Fig. 4.4 The velocity fields of some images.

$$\begin{cases} u2 = x + 5 \\ u1 = x - 5 \end{cases}$$
$$\begin{cases} v2 = y + 5 \\ v1 = y - 5 \end{cases} \qquad (4.10)$$

4.3.5 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.

4.3.6 Experimental results

In this section, we use another method to detect the target, and Fig 4.5 shows some final result. We obtain these images per $\frac{1}{30}$ second one image. In all of this sequence, we obtain more than 7000 images, and the rate of success is higher than 98.5% based on the viewpoint of the velocity field. So the detection of the target is workable by this method. Fig. 4.5 shows some results by this 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. Even if the iteration number reaches the upper bound, we still velocity, even the target is not the lightest pixel in the image, we still can find out the target in the most case. Even if the iteration number reaches the upper bound, we still can detect the target in one second. In the future, we wish that more testing videos can be provided to test the proposed algorithm.

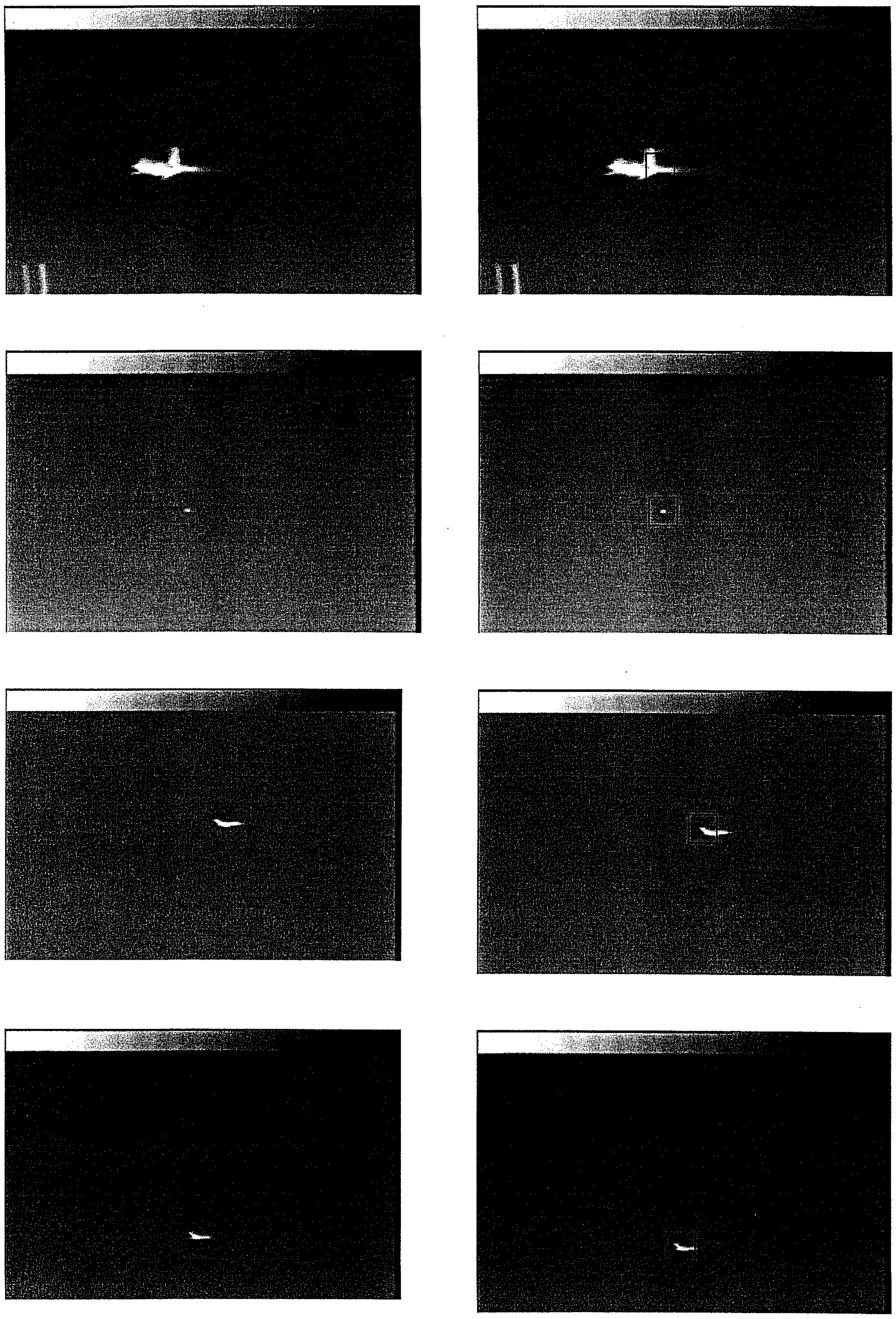


Fig. 4.5 Some final image results.

Chapter 5

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

Estimating the DOA of signals is a significant problem in the field of array signal processing. Many conventional DOA estimation methods have been proposed, including the multiple signal classification (MUSIC) method of Schmidt, and the maximum likelihood (ML) technique. Of both these available methods, the ML technique has the best performance. Nonetheless, because of the high computational load of the multivariate nonlinear maximization problem involved, the ML technique did not become popular. On the contrary, the suboptimal MUSIC method is more prevalent than the ML technique when the signal-to-noise ratio and the number of samples are both not too small, because the suboptimal method involves solving only a one-dimensional maximization problem and finding a subspace (signal subspace or noise subspace). However, the MUSIC has to perform the eigen-decomposition. Hence the major computational burden lies in finding the signal subspace or noise subspace. In summary, these methods are computationally intensive and difficult to implement in real time. The DOA estimation is viewed as a mapping problem from a different view of point in this thesis. Therefore, we propose one scheme to cope with the mapping problems. At first, we propose a six-layered neural fuzzy network (NFN) with on-line learning and anti-noise abilities. The network structure keeps integrating expert knowledge (or fuzzy rules) and neural network's learning abilities. There are no rules initially in the NFN. They are created and adapted as on-line learning proceeds through simultaneous structure and parameter learning. In the structure learning of the precondition part, the input space is partitioned in a flexible way according to an aligned clustering-based algorithm. As to the structure learning of consequent part, only a singleton value selected by a clustering method is assigned to each rule initially. The combined precondition and consequent structure learning scheme can set up an effective and dynamically growing network. Based on the constructing network, the associated parameter learning algorithms are derived. The precondition parameters are tuned by the backpropagation algorithm and the consequent parameters are tuned optimally by either least mean squares (LMS) or recursive least squares (RLS) algorithms. Both structure and parameter learning are done simultaneously to form a fast learning scheme. More notably, only the proper training data need to be provided from the outside world to achieve the structure and parameter learning without giving any initial rule in this learning method. To increase the estimation accuracy, we take the phase differences (PD) from the output of the interferometer as network input. Therefore, we can realize an optimal neuro-fuzzy estimation system to deal with the defects of low accuracy, high computational burden, and real-time estimation for the conventional estimation methods. Finally, the performances comparison of both the NFN and the RBFN in terms of convergence accuracy, estimation accuracy, sensitivity to noise, and network size are performed by various examples.

In addition, an optimal identifier is proposed for tackling the emitter identification problems. The associated learning algorithms is also derived in this thesis. Similar to the above processing method, we reconsider the problem of emitter identification as a mapping problem again. Hence, we construct a three-layered network so-called vector neural network (VNN) to identify emitter types. Each of the input variables is selected by the feature vector which is composed of the upper limit and lower limit values of the characteristic parameters in detecting signal. The input feature vectors include the *radio frequency* (RF), *pulse width* (PW), and *pulse repetition interval* (PRI). Based on the supervised learning algorithm, both conventional vector-type backpropagation (CVTBP) and new vector-type backpropagation (NVTBP) algorithms are introduced and used to derive the parameter learning algorithms. The parameter learning algorithms are also used to tune the network parameters, find the optimal connection weights and increase the correction rate of emitter identification. Capabilities and performances of both algorithms (CVTBP and NVTBP) are verified and compared by various computer simulations. Simulation results show that the NVTBP outperforms the CVTBP not only in terms of convergence rate but also in terms of correction rate. It is seen that the constructing VNN is feasible to solve the problem of the EID. For the complex signals, we may add signal parameters as input variables to keep high discrimination rate in the future.

5.1 The introduction of Vector Neural Network (VNN) for Emitter Identification

Modern radars have been widely used to detect aircrafts, ships, or land vehicles, or they can be used for searching, tracking, guidance, navigation, and weather forecasting. In military operation, radar has become an important equipment and also used to guide weaponry. Hence, an electronic support measure (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 to warn the crew of an immediate threat with enough information to take evasive action. To accomplish this function, a powerful emitter identification (EID) function must be involved in the RWR system. As the signal pulse density increases, further demands will be put on the EID function. Clearly, the EID function must be sophisticated enough to face the complex surroundings [47].

Many conventional signal recognition techniques including *k*-nearest neighbor classification and template matching rely on algorithms which are computationally intensive and require a key man to validate and verify the analysis [48]. At present, a histogramming approach is accessed by radio frequency (RF), pulse width (PW), and pulse repetition interval (PRI) of the collected pulse descriptor words (PDWs). This approach is used in a current EID system designed for sorting and comparing tabulated emitter parameters with measured signal parameters. However, these techniques are inefficiency and time-consuming for solving EID problems; they are often difficult to identify signals under high signal density environment in near real time.

For many practical problems, including pattern matching and classification, function approximation, optimization, vector quantization, data clustering and forecasting, neural networks have drawn much attention and been applied successfully in recent years [48]-[51]. Neural networks have a large number of highly interconnected nodes that usually operate in parallel and are configured in regular architectures. The massive parallelism results in the high computation rate of neural networks and makes the real-time processing of large data feasible. In this chapter, the EID problem is considered as a nonlinear mapping problem. The input features, including RF, PW, and PRI, are extracted from 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. The VNN can accept either interval-value or scalar-value input and produce scalar output. The proposed VNN is used to construct a functional mapping from the space of the interval-value features to the space of emitter types. The input and output of the VNN are related through interval arithmetics. To train the VNN, a suitable learning algorithm should be developed. The training goal is to find a set of optimal weights in the VNN such that the trained VNN can perform the function described by a training set of if-then rules. Most existing learning methods in neural networks are designed for processing numerical data [52]-[54]. 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 [55]. This BP algorithm was derived based on an error function defined by the difference of fuzzy actual output and the corresponding nonfuzzy target output through fuzzy arithmetics. Similar to their approach, we derive a vector-type BP algorithm for training the proposed VNN. Although this algorithm can train the VNN for the if-then-type training data, it has the problems of slow convergence and bad local minima as a normal scalar-type BP algorithm does. To obtain better learning results and efficiency for the VNN, we further propose a modified vector-type BP algorithm derived from a different form of error function. This learning algorithm has higher convergence rate and is not easily stuck in bad local minima. After training, the VNN can be used to identify the emitter type of the sensed scalar-value features from a real-time received emitter signal. The representation power of the VNN and the effectiveness of the modified BP learning algorithm are demonstrated on several EID problems, including the two-emitter identification problem and the multi-emitter identification problem with/without additive noise.

5.2 Problem Formulation

In general, the problem of emitter signal classification is performed in a two-step process as illustrated in Fig. 5.1. The first step is called deinterleaving (or sorting), which sorts received pulse trains into ``bins" according to the specific emitter from the composite set of pulse trains received from passive receivers in the RWR system.

After deinterleaving, the second step is to infer the emitter type by each bin of received pulses to differentiate one type from another type. A pulse descriptor word (PDW) is generated from the sorting process. Typical measurements include radio frequency (RF), pulse width (PW), time of arrival (TOA), and pulse repetition interval (PRI) as illustrated in Fig. 5.2. Thus, a PDW describes a state vector in a multidimensional space. The primary focus of the chapter will be on the problem of emitter identification (EID). Emitter parameters and performance are affected by the RF band in which they operate. Likewise, the range of frequency band chosen for a specific emitter is determined by the radar's mission and specifications. The frequency information is very important for both sorting and jamming. By comparing the frequency of the received pulses, the pulse trains can be sorted out and identified for different radars. When the frequency of the victim radar is known, the jammer can concentrate its energy in the desired frequency range. The parameter PW can be used to provide coarse information on the type of radars. For example, generally speaking, weapon radars have short pulses. Another parameter of interest in electronic warfare (EW) receiver measurements is the PRI. The information is the time difference between the leading edge of consecutive transmission waves and is the reciprocal of pulse repetition frequency (PRF). The parameter varies for different radars.

In this chapter, 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(x_1), PRI(x_2), and PW(x_3), are used to form the feature vector $[x_1, x_2, x_3]$ in this problem.

Such a nonlinear mapping function can be approximated by a suitable neural network. However, these parameters operate in interval ranges for a specific radar emitter; for example, RF ranges from 15.6 GHz to 16.6 GHz, PRI ranges from 809 μs to 960 μs, and PW ranges from 1.8 μs to 3.6 μs for some specific emitter type. 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 N_t samples of training pairs $(\tilde{\mathbf{x}}_p; \mathbf{d}_p)$ for the EID problem, where p=1,..., N_t indicating the pth training pair, $\tilde{\mathbf{x}}_p = [\tilde{x}_{p1}, \tilde{x}_{p2}, \tilde{x}_{p3}]$. In each training pair, $\tilde{x}_{pi}$ is an interval value represented by $[x_{pi}^L, x_{pi}^U]$, and $\mathbf{d}_p$ is a m-dimensional $\{0,1\}$ vector containing only one 1 to indicate the emitter type among m candidates. Hence, the VNN has 3 input nodes with each node receiving one feature's value, and m output nodes with each node representing one emitter type. The input-output relationship of the VNN is denoted by

$$\mathbf{y}_p = \hat{f}(\tilde{\mathbf{x}}_p) \tag{5.1}$$

where $\mathbf{y}_p$ 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:

$$\text{IF } x_{p1} \text{ is in } [x_{p1}^L, x_{p1}^U] \text{ and } \dots \text{ and } x_{pn} \text{ is in } [x_{pn}^L, x_{pn}^U]$$

$$\text{THEN } \mathbf{x}_p = [x_{p1}, \dots, x_{pn}] \text{ belongs to } C_k, \qquad (5.2)$$

where C_k denotes the kth emitter type.

The objective of learning is to obtain an approximated model $\hat{f}(\cdot)$ for the mapping in Eq. (5.1) and Eq. (5.2) such that the error function indicating the difference between $\mathbf{d}_p$ and $\mathbf{y}_p$, p=1,…,N_t, is minimized. Two different error functions are used in this chapter, one is the common root-mean-square error function, and the other is

$$E(w) = -\sum_{p=1}^{N_t} \{\mathbf{d}_p \ln \mathbf{y}_p + (1-\mathbf{d}_p)\ln(1-\mathbf{y}_p)\}. \qquad (5.3)$$

After training, the trained VNN can be used in the functional phase or the so-called testing phase. In this phase, the VNN on-line accepts a feature vector, $\mathbf{x} = [x_1, x_2, x_3]$ containing scalar values x_i from the sensors, and produces an output vector $\mathbf{y}_p$ with the highest-value element in $\mathbf{y}_p$ indicating the identified emitter type.

5.3 Structure of Vector Neural Network (VNN)

In this section, we shall introduce the structure and function of the vector neural network (VNN) which can process interval-value as well as scalar-value data. Before doing so, let us review some operations of *interval arithmetic* that will be used later. Let A=[a^L, a^U] and B=[b^L, b^U]$ be intervals, where the superscripts L and U represent the lower limit and upper limit, respectively. Then we have

$$A+B=\{[a^L,\ a^U]\}+\{[\ b^L,\ b^U\ ]\}=\{[\ a^L+b^L, a^U+b^U\ ]\}$$

and

$$k\cdot A=k\cdot[\ a^L,\ a^U\ ]\}=\{[\,ka^L, ka^U\,\}]=\begin{cases}[ka^L, ka^u], \text{ if } k\geq 0\\ [ka^L, ka^u], \text{ if } k<0,\end{cases}$$

where k is a real number. The activation function of a neuron can also be extended to an interval input-output relation as

$$f(Net_{in})=f([Net_{in}^L, Net_{in}^U])=[f(Net_{in}^L), f(Net_{in}^U)],$$

where $Net_{in}=[Net_{in}^L, Net_{in}^U]$ is interval-valued and $f(\cdot)$ is a sigmoid function. The sigmoid function is denoted by $f(Net_{in})=1/(1+\exp(-Net_{in}))$. The interval activation function defined by Eq. (5.6) is illustrated in Fig. 5.2.

We shall now describe the function of the VNN using the above *interval arithmetic operations*. The general structure of VNN is shown in Fig. 5.3, where the solid lines show theforward propagation of signals, and the dashed lines show the backward propagation of errors. In order to identify any n-dimensional interval-value vector, we employ a VNN that has n input nodes, l hidden nodes, and m output nodes. When the interval-value input vector $\tilde{x}_p=(\tilde{x}_{p1},\ldots,\tilde{x}_{pn})$ is presented to the input layer of

VNN, the input-output relation of each node of VNN is explicitly calculated as follows, where $\tilde{x}_{pi}=[x_{pi}^{L},x_{pi}^{U}]$.

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

Hidden nodes:

$$\tilde{z}_{pj}=[z_{pj}^{L},z_{pj}^{U}]=[f(net_{pj}^{L}),f(net_{pj}^{U})],\ j=1,\ldots,l \tag{5.7}$$

$$net_{pj}^{L}=\sum_{\substack{i=1\\ w_{ji}^{(1)}\geq 0}}^{n} w_{ji}^{(1)}x_{pi}^{L}+\sum_{\substack{i=1\\ w_{ji}^{(1)}<0}}^{n} w_{ji}^{(1)}x_{pi}^{U}+\theta_{j}, \tag{5.8}$$

$$net_{pj}^{U}=\sum_{\substack{i=1\\ w_{ji}^{(1)}\geq 0}}^{n} w_{ji}^{(1)}x_{pi}^{U}+\sum_{\substack{i=1\\ w_{ji}^{(1)}<0}}^{n} w_{ji}^{(1)}x_{pi}^{L}+\theta_{j}, \tag{5.9}$$

Output nodes:

$$\tilde{y}_{pj}=[y_{pk}^{L},z_{pk}^{U}]=[f(net_{pk}^{L}),f(net_{pk}^{U})],\ \ k=1,\ldots,m \tag{5.10}$$

$$net_{pk}^{L}=\sum_{\substack{j=1\\ w_{kj}^{(2)}\geq 0}}^{n} w_{kj}^{(2)}x_{pj}^{L}+\sum_{\substack{j=1\\ w_{kj}^{(2)}<0}}^{n} w_{kj}^{(2)}x_{pj}^{U}+\theta_{k}, \tag{5.11}$$

$$net_{pk}^{U}=\sum_{\substack{j=1\\ w_{kj}^{(2)}\geq 0}}^{n} w_{kj}^{(2)}x_{pj}^{U}+\sum_{\substack{j=1\\ w_{kj}^{(2)}<0}}^{n} w_{kj}^{(2)}x_{pj}^{L}+\theta_{k}, \tag{5.12}$$

where the weights $w_{ji}^{(1)}$, $w_{kj}^{(2)}$ and the biases θ_{j}, θ_{k} 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 $x_{pi}^{L}=x_{pi}^{U}=x_{pi}$, where x_{pi} is the scalar-value input. Correspondingly, the VNN can produce scalar output, $y_{pk}^{L}=y_{pk}^{U}=y_{pk}$.

5.4 Supervised Learning Algorithms for VNN

In this section, we shall derive a conventional vector-type backpropagation (CVTBP) learning algorithm and a new vector-type backpropagation (NVTBP) learning algorithm for the proposed VNN with interval-value input data. The comparisons ofthese two methods are also made.

5.4.1 Conventional Vector-Type Backpropagation Learning Algorithm

In this subsection, a cost function, E_{pk}, is defined, using the interval output $\tilde{y}_{pk} = [y_{pk}^L, y_{pk}^U]$ and the corresponding desired output d_{pk} for the pth input pattern, as

$$E_{pk} = \begin{cases} (d_{pk} - y_{pk}^L)^2/2, & \text{if } d_{pk} = 1 \\ (d_{pk} - y_{pk}^U)^2/2, & \text{if } d_{pk} = 0 \end{cases} \tag{5.13}$$

for the case of an interval-value input vector and a crisp desired output.

The training of VNN involves the minimization of the cost function in Eq.(5.13). As in the BP algorithm, the weight changes $\Delta\omega_{kj}^{(2)}$ and $\Delta\omega_{ji}^{(1)}$ are updated according to the following rules:

$$\Delta\omega_{kj}^{(2)}(t+1) = \alpha \quad \Delta\omega_{kj}^{(2)}(t) + \eta \quad (-\partial E_{pk} / \partial\omega_{kj}^{(2)}), \tag{5.14}$$

$$\Delta\omega_{ji}^{(1)}(t+1) = \alpha \quad \Delta\omega_{ji}^{(1)}(t) + \eta \quad (-\partial E_{pk} / \partial\omega_{ji}^{(1)}), \tag{5.15}$$

where η is the learning rate, α is the momentum constant, and the gradient can be derived by the chain rule:

$$\frac{\partial E_{pk}}{\partial \omega} = \frac{\partial E_{pk}}{\partial f}\frac{\partial f}{\partial \omega} \tag{5.16}$$

where f is the activation function. The weight updating rules for the VNN are illustrated in Fig. (5.4). To show the learning rules, we shall calculate the computation of $\frac{\partial E_{pk}}{\partial \omega}$ layer by layer along the dashed lines in Fig. (5.3), and start the derivation from the output nodes.

Layer 3: Using Eqs. (5.10)-(5.13) to calculate $\frac{\partial E_{pk}}{\partial \omega_{kj}^{(2)}}$ for various values of the weights and desired output.

1. If $d_{pk} = 1$ and $\omega_{kj}^{(2)} \geq 0$, then

$$\frac{\partial E_{pk}}{\partial \omega_{kj}^{(2)}} = -(d_{pk} - y_{pk}^{L})y_{pk}^{L}(1 - y_{pk}^{L})z_{pj}^{L} \triangleq -\delta_{pk}^{L} z_{pj}^{L}, \tag{5.17}$$

where $\frac{\partial y_{pk}^{L}}{\partial net_{pk}^{L}} = \partial f(net_{pk}^{L}) \Big/ \partial net_{pk}^{L}$ and $f(net_{pk}^{L}) = 1 \Big/ (1 + \exp(-net_{pk}^{L}))$.

2. If $d_{pk} = 1$ and $\omega_{kj}^{(2)} < 0$, then

$$\frac{\partial E_{pk}}{\partial \omega_{kj}^{(2)}} = -\delta_{pk}^{L} z_{pj}^{U}, \tag{5.18}$$

3. If $d_{pk} = 0$ and $\omega_{kj}^{(2)} \geq 0$, then

$$\frac{\partial E_{pk}}{\partial \omega_{kj}^{(2)}} = -(d_{pk} - y_{pk}^{U})y_{pk}^{U}(1 - y_{pk}^{U})z_{pj}^{U} \triangleq -\delta_{pk}^{U} z_{pj}^{U},$$

where $\dfrac{\partial y_{pk}^{U}}{\partial net_{pk}^{U}} = \partial f(net_{pk}^{U}) \Big/ \partial net_{pk}^{U}$ and $f(net_{pk}^{U}) = 1 \Big/ (1 + \exp(-net_{pk}^{U}))$.

4. If $d_{pk} = 0$ and $\omega_{kj}^{(2)} \geq 0$, then

$$\frac{\partial E_{pk}}{\partial \omega_{kj}^{(2)}} = -\delta_{pk}^{U} z_{pj}^{L}, \tag{5.20}$$

Layer 2: Using Eqs. (5.7)-(5.9) and (5.13) to calculate for different values of the weights and desired output.

1. If $d_{pk} = 1$, $\omega_{kj}^{(2)} \geq 0$ and $\omega_{ji}^{(1)} \geq 0$, then

$$\frac{\partial E_{pk}}{\partial \omega_{ji}^{(1)}} = -(d_{pk} - y_{pk}^{L})y_{pk}^{L}(1 - y_{pk}^{L})\omega_{kj}^{(2)} z_{pj}^{L}(1 - z_{pj}^{L})x_{pi}^{L} \triangleq -\delta_{pk}^{L} \omega_{kj}^{(2)} z_{pj}^{L}(1 - z_{pj}^{L})x_{pi}^{L}, \tag{5.21}$$

2. If $d_{pk} = 1$, $\omega_{kj}^{(2)} \geq 0$ and $\omega_{ji}^{(1)} < 0$, then

$$\frac{\partial E_{pk}}{\partial \omega_{ji}^{(1)}} = -\delta_{pk}^{L} \omega_{kj}^{(2)} z_{pj}^{L}(1 - z_{pj}^{L})x_{pi}^{U}, \tag{5.22}$$

3. If $d_{pk} = 1$, $\omega_{kj}^{(2)} < 0$ and $\omega_{ji}^{(1)} \geq 0$, then

$$\frac{\partial E_{pk}}{\partial \omega_{ji}^{(1)}} = -\delta_{pk}^{L} \omega_{kj}^{(2)} z_{pj}^{U}(1 - z_{pj}^{U})x_{pi}^{U}, \tag{5.23}$$

4. If $d_{pk} = 1$, $\omega_{kj}^{(2)} < 0$ and $\omega_{ji}^{(1)} < 0$, then

$$\frac{\partial E_{pk}}{\partial \omega_{ji}^{(1)}} = -\delta_{pk}^{L} \omega_{kj}^{(2)} z_{pj}^{L} (1 - z_{pj}^{U}) x_{pi}^{L}, \tag{5.24}$$

5. If $d_{pk} = 0, w_{kj}^{(2)} \geq 0$ and $w_{ji}^{(1)} \geq 0$, then

$$\begin{aligned}\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} &= -(d_{pk} - y_{pk}^{U}) y_{pk}^{U} (1 - y_{pk}^{U}) w_{kj}^{(2)} z_{pj}^{U} (1 - z_{pj}^{U}) x_{pi}^{U} \\ &\triangleq -\delta_{pk}^{U} w_{kj}^{(2)} z_{pj}^{U} (1 - z_{pj}^{U}) x_{pi}^{U}\end{aligned} \tag{5.25}$$

6. If $d_{pk} = 0, w_{kj}^{(2)} \geq 0$ and $w_{ji}^{(1)} < 0$, then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta_{pk}^{U} w_{kj}^{(2)} z_{pj}^{U} (1 - z_{pj}^{U}) x_{pi}^{L} \tag{5.26}$$

7. If $d_{pk} = 0, w_{kj}^{(2)} < 0$ and $w_{ji}^{(1)} \geq 0$, then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta_{pk}^{U} w_{kj}^{(2)} z_{pj}^{L} (1 - z_{pj}^{L}) x_{pi}^{L} \tag{5.27}$$

8. If $d_{pk} = 0, w_{kj}^{(2)} < 0$ and $w_{ji}^{(1)} < 0$, then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta_{pk}^{U} w_{kj}^{(2)} z_{pj}^{L} (1 - z_{pj}^{L}) x_{pi}^{U} \tag{5.28}$$

$$\delta_{pk}^{L} \triangleq \frac{\partial E_{pk}}{\partial net_{pk}^{L}} = \frac{\partial E_{pk}}{\partial y_{pk}^{L}} \frac{\partial y_{pk}^{L}}{\partial net_{pk}^{L}} = (d_{pk} - y_{pk}^{L}) y_{pk}^{L} (1 - y_{pk}^{L}), \tag{5.29}$$

$$\delta_{pk}^{U} \triangleq \frac{\partial E_{pk}}{\partial net_{pk}^{U}} = \frac{\partial E_{pk}}{\partial y_{pk}^{U}} \frac{\partial y_{pk}^{U}}{\partial net_{pk}^{U}} = (d_{pk} - y_{pk}^{U}) y_{pk}^{U} (1 - y_{pk}^{U}). \tag{5.30}$$

Clearly, the value of δ_{pk} is proportional to the amount of $(d_{pk} - y_{pk}) y_{pk} (1 - y_{pk})$.

5.4.2 New Vector-Type Backpropagation Learning Algorithm

The aim of this subsection is to modify the CVTBP learning algorithm derived in the last subsection to enhance the identification power of VNN. To achieve this, we propose another error function instead of the squares of the differences between the actual interval output $\tilde{y}_{pk}$ and the corresponding desired output $\tilde{d}_{pk}$ as in Eq.(5.13), where the subscript pk represents the pth input pattern and kth output node. The new error function is defined as

$$E_{pk} = \begin{cases} -d_{pk} \ln y_{pk}{}^{L} - (1-d_{pk}) \ln(1-y_{pk}{}^{L}), \text{ if } d_{pk}=1 \\ -d_{pk} \ln y_{pk}{}^{U} - (1-d_{pk}) \ln(1-y_{pk}{}^{U}), \text{ if } d_{pk}=0. \end{cases} \tag{5.31}$$

The meaning of this error function will be described later. The learning objective is to minimize the error function in Eq.(5.31). In the new vector-type backpropagation (NVTBP) learning algorithm, the partial derivative of E_{pk} with respect to y_{pk} is,

$$\begin{aligned} \frac{\partial E_{pk}}{\partial y_{pk}{}^{L}} &= -\frac{d_{pk}}{y_{pk}{}^{L}} + \frac{1-d_{pk}}{1-y_{pk}{}^{L}} \\ &= \frac{y_{pk}{}^{L} - d_{pk}}{(1-d_{pk})y_{pk}{}^{L}}, \end{aligned} \tag{5.32}$$

and

$$\begin{aligned} \frac{\partial E_{pk}}{\partial y_{pk}{}^{U}} &= -\frac{d_{pk}}{y_{pk}{}^{U}} + \frac{1-d_{pk}}{1-y_{pk}{}^{U}} \\ &= \frac{y_{pk}{}^{U} - d_{pk}}{(1-d_{pk})y_{pk}{}^{U}}, \end{aligned} \tag{5.33}$$

Likewise, Eqs. (5.14) and (5.15) describe the weight changes $\Delta w_{kj}{}^{(2)}$ and $\Delta w_{ji}{}^{(1)}$ in the modified BP algorithm. We shall calculate the values of $\partial E_{pk} / \partial w_{kj}^{(2)}$ for each connection from the hidden layer to the output layer. Similarly, we shall calculate the values of $\partial E_{pk} / \partial w_{ji}^{(1)}$ for each connection from the input layer to the hidden layer. Then the connection weights are updated according to these gradient values. The procedure of the derivation is layer by layer along the dashed lines in Fig. 4, and start the derivation from the output nodes.

Layer 3: Using Eqs. (5.10)-(5.12), (5.32), and (5.33) to calculate $\partial E_{pk} / \partial w_{kj}^{(2)}$ for different values of the weights and desired output.

1. If $d_{pk} = 1, w_{kj}^{(2)} \geq 0,$ then

$$\begin{aligned}\frac{\partial E_{pk}}{\partial w_{kj}^{(2)}} &= \frac{\partial}{\partial w_{kj}^{(2)}}\left\{-d_{pk} \ln y_{pk}^{L} - (1-d_{pk})\ln(1-y_{pk}^{L})\right\} \\ &= \frac{\partial E_{pk}}{\partial y_{pk}^{L}} \frac{\partial y_{pk}^{L}}{\partial net_{pk}^{L}} \frac{\partial net_{pk}^{L}}{\partial w_{kj}^{(2)}} \\ &= -\frac{(d_{pk} - y_{pk}^{L})}{y_{pk}^{L}(1-y_{pk}^{L})} y_{pk}^{L}(1-y_{pk}^{L}) z_{pj}^{L} \\ &= -(d_{pk} - y_{pk}^{L}) z_{pj}^{L} \\ &\triangleq -\delta'_{pk}{}^{L} z_{pj}^{L}\end{aligned} \qquad (5.34)$$

2. If $d_{pk} = 1, w_{kj}^{(2)} < 0,$ then

$$\begin{aligned}\frac{\partial E_{pk}}{\partial w_{kj}^{(2)}} &= -(d_{pk} - y_{pk}^{L}) z_{pj}^{U} \\ &\triangleq -\delta'_{pk}{}^{L} z_{pj}^{U}\end{aligned} \qquad (5.35)$$

3. If $d_{pk} = 0, w_{kj}^{(2)} \geq 0,$ then

$$\begin{aligned}\frac{\partial E_{pk}}{\partial w_{kj}^{(2)}} &= \frac{\partial}{\partial w_{kj}^{(2)}}\left\{-d_{pk}\ln y_{pk}^{U}-(1-d_{pk})\ln(1-y_{pk}^{U})\right\} \\ &= \frac{\partial E_{pk}}{\partial y_{pk}^{U}}\frac{\partial y_{pk}^{U}}{\partial net_{pk}^{U}}\frac{\partial net_{pk}^{U}}{\partial w_{kj}^{(2)}} \\ &= -\frac{(d_{pk}-y_{pk}^{U})}{y_{pk}^{U}(1-y_{pk}^{U})}y_{pk}^{U}(1-y_{pk}^{U})z_{pj}^{U} \\ &= -(d_{pk}-y_{pk}^{U})z_{pj}^{U} \\ &\triangleq -\delta'_{pk}{}^{U}z_{pj}^{U}\end{aligned} \tag{5.36}$$

4. If $d_{pk}=0, w_{kj}^{(2)}<0,$ then

$$\begin{aligned}\frac{\partial E_{pk}}{\partial w_{kj}^{(2)}} &= -(d_{pk}-y_{pk}^{U})z_{pj}^{L} \\ &\triangleq -\delta'_{pk}{}^{U}z_{pj}^{L}\end{aligned} \tag{5.37}$$

Layer 2: Using Eqs. (5.7)-(5.9), (5.32), and (5.33) to calculate $\partial E_{pk}/\partial w_{ji}^{(1)}$ for different values of the weights and desired output.

1. If $d_{pk}=1, w_{kj}^{(2)}\geq 0$ and $w_{ji}^{(1)}\geq 0,$ then

$$\begin{aligned}\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} &= \frac{\partial}{\partial w_{ji}^{(1)}}\left\{-d_{pk}\ln y_{pk}^{L}-(1-d_{pk})\ln(1-y_{pk}^{L})\right\} \\ &= \frac{\partial E_{pk}}{\partial y_{pk}^{L}}\frac{\partial y_{pk}^{L}}{\partial net_{pk}^{L}}\frac{\partial net_{pk}^{L}}{\partial y_{pj}^{L}}\frac{\partial y_{pj}^{L}}{\partial net_{pj}^{L}}\frac{\partial net_{pj}^{L}}{\partial w_{ji}^{(1)}} \\ &= -\frac{(d_{pk}-y_{pk}^{L})}{y_{pk}^{L}(1-y_{pk}^{L})}y_{pk}^{L}(1-y_{pk}^{L})w_{kj}^{(2)}z_{pj}^{L}(1-z_{pj}^{L})x_{pi}^{L} \\ &= -(d_{pk}-y_{pk}^{L})w_{kj}^{(2)}z_{pj}^{L}(1-z_{pj}^{L})x_{pi}^{L} \\ &\triangleq -\delta'_{pk}{}^{L}w_{kj}^{(2)}z_{pj}^{L}(1-z_{pj}^{L})x_{pi}^{L}\end{aligned} \tag{5.38}$$

2. If $d_{pk}=1, w_{kj}^{(2)}\geq 0$ and $w_{ji}^{(1)}<0,$ then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta'_{pk}{}^{L}w_{kj}^{(2)}z_{pj}^{L}(1-z_{pj}^{L})x_{pi}^{L} \tag{5.39}$$

3. If $d_{pk}=1, w_{kj}^{(2)}<0$ and $w_{ji}^{(1)}\geq 0,$ then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta'_{pk}{}^{L} w_{kj}^{(2)} z_{pj}{}^{U}(1-z_{pj}{}^{U}) x_{pi}{}^{U} \tag{5.40}$$

4. If $d_{pk}=1, w_{kj}^{(2)}<0$ and $w_{ji}^{(1)}<0$, then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta'_{pk}{}^{L} w_{kj}^{(2)} z_{pj}{}^{U}(1-z_{pj}{}^{U}) x_{pi}{}^{L} \tag{5.41}$$

5. If $d_{pk}=0, w_{kj}^{(2)}\geq 0$ and $w_{ji}^{(1)}\geq 0$, then

$$\begin{aligned}\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} &= \frac{\partial}{\partial w_{ji}^{(1)}}\left\{-d_{pk}\ln y_{pk}{}^{U}-(1-d_{pk})\ln(1-y_{pk}{}^{U})\right\}\\ &= \frac{\partial E_{pk}}{\partial y_{pk}{}^{U}}\frac{\partial y_{pk}{}^{U}}{\partial net_{pk}{}^{U}}\frac{\partial net_{pk}{}^{U}}{\partial y_{pj}{}^{U}}\frac{\partial y_{pj}{}^{U}}{\partial net_{pj}{}^{U}}\frac{\partial net_{pj}{}^{U}}{\partial w_{ji}^{(1)}}\\ &= -\frac{(d_{pk}-y_{pk}{}^{U})}{y_{pk}{}^{U}(1-y_{pk}{}^{U})} y_{pk}{}^{U}(1-y_{pk}{}^{U}) w_{kj}^{(2)} z_{pj}{}^{U}(1-z_{pj}{}^{U}) x_{pi}{}^{U}\\ &= -(d_{pk}-y_{pk}{}^{U}) w_{kj}^{(2)} z_{pj}{}^{U}(1-z_{pj}{}^{U}) x_{pi}{}^{U}\\ &\triangleq -\delta'_{pk}{}^{U} w_{kj}^{(2)} z_{pj}{}^{U}(1-z_{pj}{}^{U}) x_{pi}{}^{U}\end{aligned} \tag{5.42}$$

6. If $d_{pk}=0, w_{kj}^{(2)}\geq 0$ and $w_{ji}^{(1)}<0$, then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta'_{pk}{}^{U} w_{kj}^{(2)} z_{pj}{}^{U}(1-z_{pj}{}^{U}) x_{pi}{}^{L} \tag{5.43}$$

7. If $d_{pk}=0, w_{kj}^{(2)}<0$ and $w_{ji}^{(1)}\geq 0$, then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta'_{pk}{}^{U} w_{kj}^{(2)} z_{pj}{}^{L}(1-z_{pj}{}^{L}) x_{pi}{}^{L} \tag{5.44}$$

8. If $d_{pk}=0, w_{kj}^{(2)}<0$ and $w_{ji}^{(1)}<0$, then

$$\frac{\partial E_{pk}}{\partial w_{ji}^{(1)}} = -\delta'_{pk}{}^{U} w_{kj}^{(2)} z_{pj}{}^{L}(1-z_{pj}{}^{L}) x_{pi}{}^{U} \tag{5.45}$$

In the previous discussion, the notations $\delta'_{pk}{}^{L}$ and $\delta'_{pk}{}^{U}$ are defined as follows:

$$\delta'_{pk}{}^{L} \triangleq \frac{\partial E_{pk}}{\partial net_{pk}{}^{L}} = \frac{\partial E_{pk}}{\partial y_{pk}{}^{L}} \frac{\partial y_{pk}{}^{L}}{\partial net_{pk}{}^{L}} = (d_{pk} - y_{pk}{}^{L}), \tag{5.46}$$

$$\delta'_{pk}{}^{U} \triangleq \frac{\partial E_{pk}}{\partial net_{pk}{}^{U}} = \frac{\partial E_{pk}}{\partial y_{pk}{}^{U}} \frac{\partial y_{pk}{}^{U}}{\partial net_{pk}{}^{U}} = (d_{pk} - y_{pk}{}^{U}). \tag{5.47}$$

Clearly, the value of δ'_{pk} is proportional to the amount of $(d_{pk} - y_{pk})$ rather than $(d_{pk} - y_{pk})y_{pk}(1 - y_{pk})$ as in the conventional BP learning rule. When the actual output y_{pk} (representing $y_{pk}{}^{L}$ or $y_{pk}{}^{U}$) approaches the value of 1 or 0, the factor $y_{pk}(1 - y_{pk})$ makes the error signal δ_{pk} very small. This implies that an output node can be maximally wrong without producing a strong error signal with which the connection weights could be significantly adjusted. This decelerates the search for a minimum in the error. A detailed description of quantitative analysis can be found in.

In summary, the supervised learning algorithm for the VNN is outlined in the following.

Algorithm NVTBP/CVTBP:

Consider a 3-layered VNN with n input nodes, l hidden nodes, and m output nodes. The connection weight $w_{ji}{}^{(1)}$ is from node i of the input layer to the jth node of the hidden layer, and $w_{kj}{}^{(2)}$ is from the kth node of the hidden layer to the kth node of the output layer.

Input: A set of training pairs $\{(\tilde{x}_p; d_p), p = 1, \ldots, N_t\}$, where the input vectors are in interval

1. (Initialization): Choose $\eta > 0$ and $E_{\max}$ (maximum tolerable error). Initialize the

weights to small random values. Set E=0 and p=1.

2. (Training loop): Apply the *p*th input pattern $\tilde{x}_p$ to the input layer.

3. (Forward propagation): Propagate the signal forward through the network from the input layer to the output layer. Use Eqs.(4.7)-(4.9) to compute the net input (net_{pj}) and output ($\tilde{z}_{pj}$) of the *j*th hidden node, and use Eqs.(4.10)-(4.12) to compute the net input (net_{pk}) and output ($\tilde{y}_{pk}$) of the kth output node.

4. (Output error measure): Compute the error signal δ'_{pk} from Eqs. (4.46) and (4.47), and compute δ'_{pk} from Eqs. (5.29) and (5.30).

5. (Error backpropagation): Propagate the errors backward to update the weight changes $\Delta\omega_{kj}^{(2)}$ between hidden and output nodes by using Eq. (5.14), and update the weight changes $\Delta\omega_{ji}^{(1)}$ between input and hidden nodes by using Eq. (5.15).

6. (One epoch looping): Check whether the whole set of training data has been cycled once. If $p < N_t$, then p=p+1 and go to Step 1; otherwise, go to Step 7.

7. (Total error checking): Check whether the current total error is acceptable; if $E < E_{max}$, then terminate the training process and output the final weights; otherwise, set E=0, p=1, and initiate the new training epoch by going to Step 2.

The above algorithm adopts the *incremental* approach in updating the weights; that is, the weights are updated for each incoming training pattern. Finally, the optimal weights $w_{opt}^{(1)}$ and $w_{opt}^{(2)}$ can be obtained through the training procedure and expressed by

$$w_{opt}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & \cdots & w_{1n}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & \cdots & w_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ w_{l1}^{(1)} & w_{l2}^{(1)} & \cdots & w_{\mathrm{ln}}^{(1)} \end{bmatrix}, \tag{5.48}$$

and

$$w_{opt}^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} & \cdots & w_{1n}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & \cdots & w_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ w_{m1}^{(2)} & w_{m2}^{(2)} & \cdots & w_{mn}^{(2)} \end{bmatrix}, \tag{5.49}$$

5.5 Simulation Results

In this section, we employ the proposed VNN trained by the modified BP algorithm to handle the practical emitter identification problems in real-life; i.e., to map input patterns to their respective emitter types. An input pattern is determined to belong to the kth type if the kth output node produces a higher value than all the other output nodes when this input pattern is presented to the VNN. Before the input patterns are presented to the VNN, the range of each parameter must be normalized over the following bound to increase the network's learning ability:

RF : 2.0 GHz to 18.0 GHz

PRI : 1.0 μs to 10.0 μs

PW : 0.1 μs to 10.0 μs

Two problems are examined to demonstrate the identification capability of the proposed VNN in this section, the two-emitter identification problem and three-emitter identification problem. The performance is compared to that of the VNN trained by the conventional BP algorithm on the same training and testing data.

5.5.1 Performance Evaluation without Measurement Error

In this subsection, two experiments are performed for clean input data without measurement distortion to demonstrate the identification capability of the VNN with the NVTBP and CVTBP learning algorithms, respectively.

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). We set the learning rate as $\eta=0.01$ and momentum constant as $\alpha=0.99$ in Eqs. (5.14) and (5.15) of the NVTBP learning algorithm, and $\eta=0.05$, $\alpha=0.9$ in Eqs. (5.14) and (5.15) of the CVTBP learning algorithm. The 10 input-output training pairs (5 pairs for each type) are listed in Table 5.1. In the training phase, we use these training pairs to train two VNNs using the CVTBP and NVTBP learning algorithms, respectively, and find individually a set of optimal weights. In the testing phase, 80 testing patterns (40 patterns for each emitter type) are randomly selected from the ranges of emitter parameters and are presented to the trained VNNs for performance testing. Part of these testing patterns are shown in Table 5.2. Once a testing pattern is fed to the trained VNNs, the networks identify immediately its emitter type in near real time. The testing results show that the two trained VNNs achieve high identification rates. 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 5.5.

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. We first set the learning constant as η=0.02 and momentum constant as α=0.7 in both learning algorithms. The 15 input-output training pairs (5 pairs for each type) as listed in Table 5.3 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. Part of these testing patterns are shown in Table 5.4. Again, both networks show high identification capability, where the VNN trained by the NVTBP learning algorithm achieves an average identification rate of 99.84% nd the other VNN 91.08% as listed in the last row of Table 5.6. In the above two experiments, the results show that the two-emitter and three-emitter identification problems can be easily handled by the VNN with the derived NVTBP learning algorithm and the CVTBP learning algorithm in real time. However, the VNN trained by the NVTBP learning algorithm has better identification capability than that trained by the CVTBP learning algorithm. These experiments are performed for clean input data without measurement distortion. The robustness of the proposed scheme in noisy environment is tested in the following experiments.

5.5.2 Performance Evaluation with Measurement Error

In this subsection, two experiments are performed to evaluate the robustness of VNN with measurement distortion. In these experiments, the measurement distortion is simulated by adding noise. To perform the testing at different levels of adding noise, we define the error deviation level (EDL) by

$$\mathrm{EDL}_i(\%)=\frac{\zeta_{pi}}{\chi_{pi}}\times 100\%,\ i=1,2,3, \tag{5.50}$$

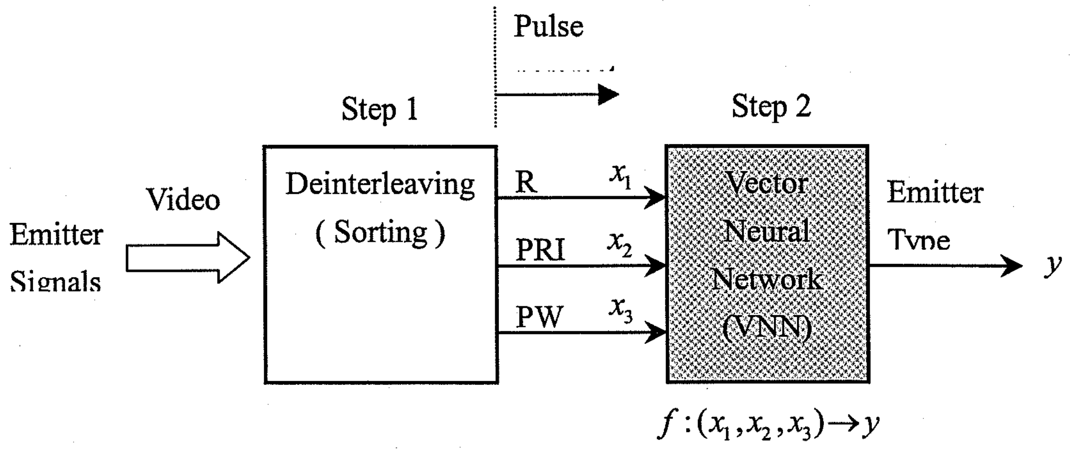
for ith input feature, where ζ_{pi} is a small randomly generated deviation for the pth input pattern.

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, ζ_{pi} (i=1, 2, 3), to each testing pattern ($\chi_{p1}, \chi_{p2}, \chi_{p3}$), p=1,..., 80, used in Experiment 1 to form the noise testing database, ($\chi_{p1}\pm\zeta_{p1},\ \chi_{p2}\pm\zeta_{p2},\ \chi_{p3}\pm\zeta_{p3}$). 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 5.5.

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, ζ_{pi} (i=1, 2, 3), to each testing pattern (χ_{p1}, χ_{p2}, χ_{p3}), p=1,..., 120, used in Experiment 2 to form the noise testing database, ($\chi_{p1} \pm \zeta_{p1}$, $\chi_{p2} \pm \zeta_{p2}$, $\chi_{p3} \pm \zeta_{p3}$). 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 5.6. The testing results in Table 5.5 and Table 5.6 indicate that, as expected, the VNN's identification ability decreases as EDL increasing in noisy environments. In conclusion, the proposed VNN trained by the NVTBP learning algorithm not only has higher identification capability, but is also relatively more insensitive to noise than that trained by the CVTBP learning algorithm.

5.6 Concluding Remarks

In this chapter, a vector neural network (VNN) along with a new vector-type backpropagation (NVTBP) learning algorithm was proposed to solve the emitter identification (EID) problem. The VNN can learn the teaching patterns in the form of interval-value input and scalar-value output in the training phase, and then operate in the way of scalar-value input and scalar-value output in the testing phase by means of interval arithmetics. The main contribution of this chapter is to propose an idea for integrating the processing of interval-value and scalar-value data into a single processing system and derive a NVTBP learning algorithm for solving the practical EID problem in real time. In fact, the proposed network with the NVTBP learning algorithm can not only solve the learning problem with interval-value data, but also improve the convergence of the CVTBP learning algorithm. The simulated results show that the proposed VNN can always produce high identification accuracy for the emitter signals. Also, it was demonstrated that the proposed VNN is quite insensitive to the additive error. With these results achieved in this chapter, the proposed VNN may be widely applied to military applications (such as reconnaissance and threat reaction) for achieving high power of identification for emitter signals to replace the EID function in the electronics support measures (such as RWR). In this chapter, we have showed that the proposed VNN can be used for identifying unambiguous emitters. In the future work, we will use the extra parameters of emitters such as angle of arrival and amplitude to form a new enlarged input feature vector for handling the problem of multiple ambiguous emitters; i.e., different types of emitters have similar characteristics.



Pulse
Step 1
Step 2
Emitter Signals
Video
Deinterleaving (Sorting)
R
PRI
PW
x_1
x_2
x_3
Vector Neural Network (VNN)
Emitter Type
y
$f:(x_1,x_2,x_3)\rightarrow y$

Figure. 5.1 Flow chart of emitter signal

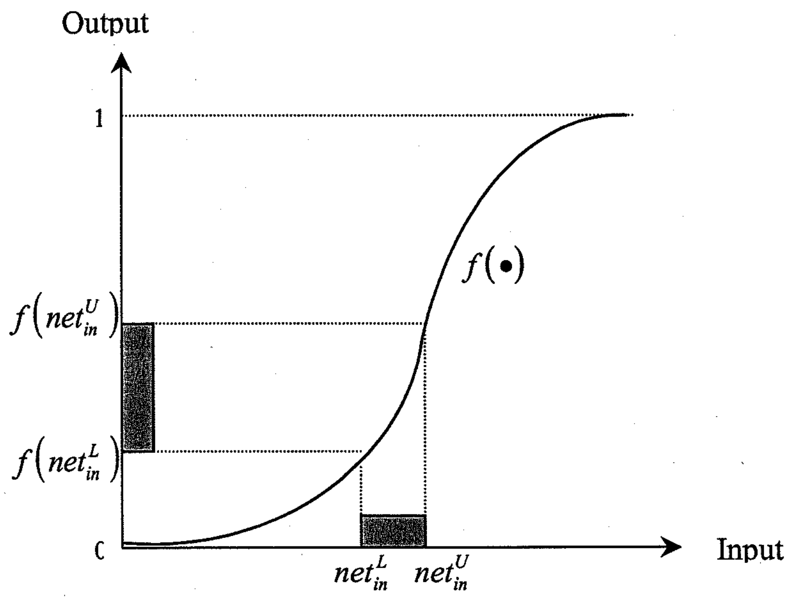


Figure. 5.2 Interval sigmoid function of each node in the VNN, where $\left(net_{in}^{L},net_{in}^{U}\right)$ and $f\left(net_{in}^{L},net_{in}^{U}\right)$ are an interval input and an interval output, respectively.

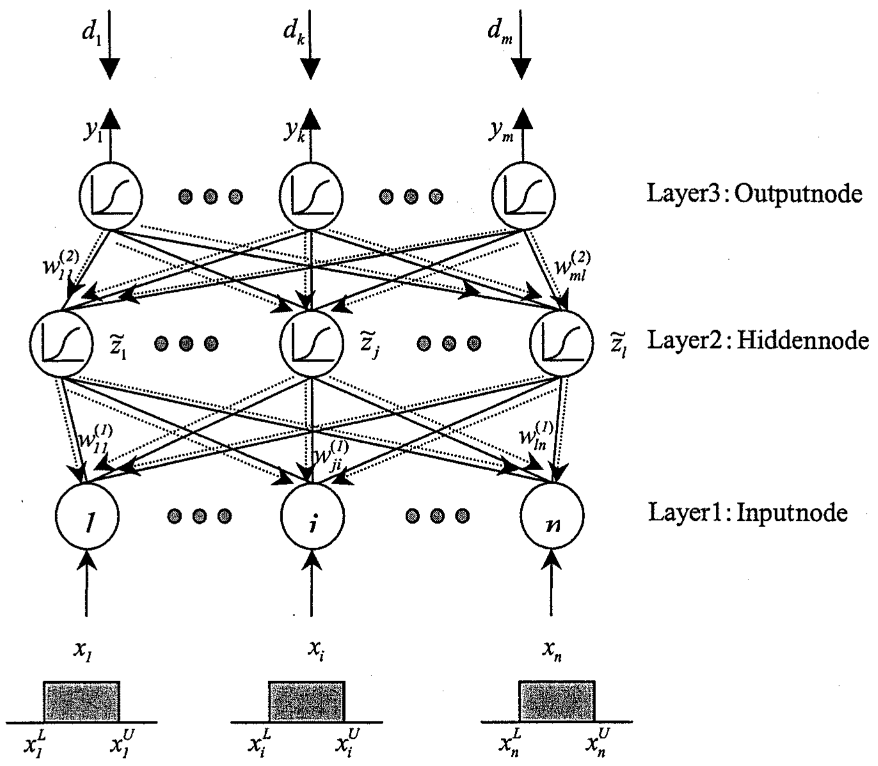


Figure. 5.3 The three-layered architecture of the proposed VNN.

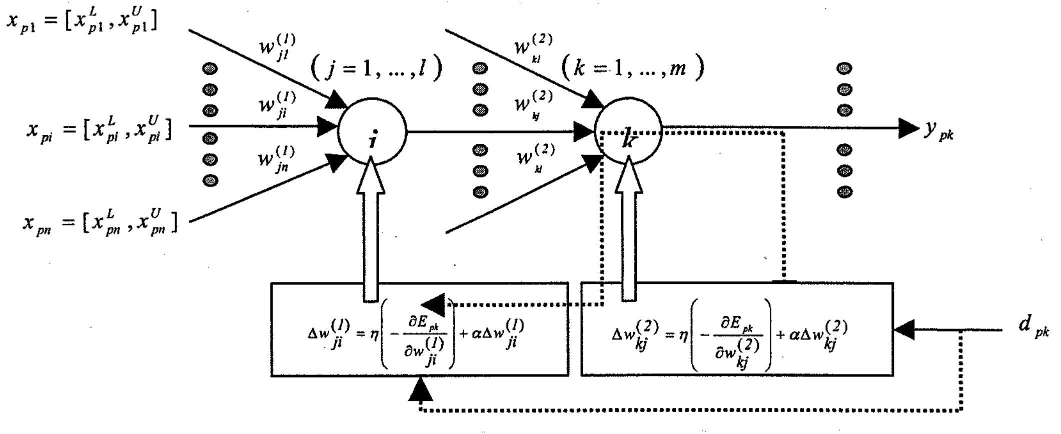

$x_{p1} = [x_{p1}^L, x_{p1}^U]$
$x_{pi} = [x_{pi}^L, x_{pi}^U]$
$x_{pn} = [x_{pn}^L, x_{pn}^U]$
$w_{j1}^{(1)}$
$w_{ji}^{(1)}$
$w_{jn}^{(1)}$
$(j = 1, \ldots, l)$
$w_{kl}^{(2)}$
$w_{kj}^{(2)}$
$w_{kl}^{(2)}$
$(k = 1, \ldots, m)$
i
k
y_{pk}
d_{pk}
$\Delta w_{ji}^{(1)} = \eta\left(-\frac{\partial E_{pk}}{\partial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

Figure. 5.4 Illustration of backpropagation learning rule for the VNN.

Pattern Number	RF (GHz)		PRI (μs)		PW (μs)		Emitter Type
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit	
1	15.6	16.6	8.09	9.60	1.80	3.60	1
2	4.2	5.0	2.20	4.50	2.45	4.75	2
3	4.3	4.9	2.30	4.40	2.50	4.70	2
4	4.4	4.8	2.40	4.30	2.55	4.65	2
5	15.70	16.50	8.19	9.50	1.90	3.50	1
6	15.80	16.40	8.29	9.40	2.00	3.40	1
7	4.50	4.70	2.50	4.20	2.60	4.60	2
8	15.90	16.30	8.39	9.30	2.10	3.30	1
9	16.00	16.20	8.49	9.20	2.20	3.20	1
10	4.55	4.60	2.60	4.10	2.65	4.55	2

Table. 5.1 Input-output training pairs for the two-emitter identification problem in Experiments 1 and 3.

Pattern Number	RF (GHz)		PRI (μs)		PW (μs)		Emitter Type
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit	
1	15.65	15.65	8.09	8.09	1.80	1.80	1
2	15.70	15.70	8.12	8.12	1.82	1.82	1
3	15.75	15.75	8.15	8.15	1.85	1.85	1
4	15.80	15.80	8.19	8.19	1.88	1.88	1
5	15.85	15.85	8.23	8.23	1.91	1.91	1
6	15.90	15.90	8.25	8.25	1.93	1.93	1
7	15.95	15.95	8.28	8.28	1.95	1.95	1
8	16.05	16.05	8.33	8.33	1.99	1.99	1
9	16.10	16.10	8.39	8.39	2.03	2.03	1
10	16.13	16.13	8.43	8.43	2.05	2.05	1
11	4.20	4.20	2.19	2.19	2.45	2.45	2
12	4.22	4.22	2.24	2.24	2.51	2.51	2
13	4.23	4.23	2.43	2.43	2.58	2.58	2
14	4.25	4.25	2.57	2.57	2.62	2.62	2
15	4.28	4.28	2.68	2.68	2.74	2.74	2
16	4.31	4.31	2.73	2.73	2.81	2.81	2
17	4.33	4.33	2.82	2.82	2.85	2.85	2
18	4.35	4.35	2.91	2.91	2.98	2.98	2
19	4.38	4.38	3.03	3.03	3.05	3.05	2
20	4.43	4.43	3.13	3.13	3.15	3.15	2

Table. 5.2 Part of the testing samples for the two-emitter identification problem in Experiments 1 and 3.

Pattern Number	RF (GHz)		PRI (μ s)		PW (μ s)		Emitter Type
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit	
1	15.6	16.6	8.09	9.60	1.80	3.60	1
2	4.20	5.00	2.20	4.50	2.45	4.75	2
3	4.30	4.90	2.30	4.40	2.50	4.70	2
4	16.35	17.45	5.05	6.90	5.35	7.85	3
5	15.70	16.50	8.19	9.50	1.90	3.50	1
6	15.80	16.40	8.29	9.40	2.00	3.40	1
7	16.45	17.35	5.15	6.80	5.45	7.75	3
8	16.55	17.25	5.25	6.70	5.55	7.65	3
9	4.50	4.70	2.50	4.20	2.60	4.60	2
10	15.90	16.30	8.39	9.30	2.10	3.30	1
11	16.00	16.20	8.49	9.20	2.20	3.20	1
12	16.75	17.15	5.35	6.60	5.65	7.55	3
13	4.40	4.80	2.40	4.30	2.55	4.65	2
14	16.85	17.05	5.45	6.5	5.75	7.45	3
15	4.55	4.60	2.60	4.10	2.65	4.55	2

Table. 5.3 Input-output training pairs for the three-emitter identification problem in Experiments 2 and 4.

Pattern Number	RF (GHz)		PRI (μ s)		PW (μ s)		Emitter Type
	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit	
1	15.65	15.65	8.09	8.09	1.80	1.80	1
2	15.70	15.70	8.12	8.12	1.82	1.82	1
3	15.75	15.75	8.15	8.15	1.85	1.85	1
4	15.80	15.80	8.19	8.19	1.88	1.88	1
5	15.85	15.85	8.23	8.23	1.91	1.91	1
6	15.90	15.90	8.25	8.25	1.93	1.93	1
7	15.95	15.95	8.28	8.28	1.95	1.95	1
8	16.05	16.05	8.33	8.33	1.99	1.99	1
9	16.10	16.10	8.39	8.39	2.03	2.03	1
10	16.13	16.13	8.43	8.43	2.05	2.05	1
11	4.20	4.20	2.19	2.19	2.45	2.45	2
12	4.22	4.22	2.24	2.24	2.51	2.51	2
13	4.23	4.23	2.43	2.43	2.58	2.58	2
14	4.25	4.25	2.57	2.57	2.62	2.62	2
15	4.28	4.28	2.68	2.68	2.74	2.74	2
16	4.31	4.31	2.73	2.73	2.81	2.81	2
17	4.33	4.33	2.82	2.82	2.85	2.85	2
18	4.35	4.35	2.91	2.91	2.98	2.98	2
19	4.38	4.38	3.03	3.03	3.05	3.05	2
20	16.35	16.35	5.05	5.05	5.35	5.35	3
21	16.41	16.41	5.13	5.13	5.42	5.42	3
22	16.45	16.45	5.17	5.17	5.55	5.55	3
23	16.51	16.51	5.25	5.25	5.88	5.88	3
24	16.55	16.55	5.28	5.28	5.91	5.91	3
25	16.61	16.61	5.35	5.35	5.93	5.93	3
26	16.65	16.65	5.39	5.39	5.95	5.95	3
27	16.69	16.69	5.43	5.43	5.99	5.99	3
28	16.72	16.72	5.59	5.59	6.03	6.03	3
29	16.83	16.83	5.63	5.63	6.05	6.05	3
30	4.43	4.43	3.13	3.13	3.15	3.15	2

Table. 5.4 Part of the testing samples for the three-emitter identification problem in Experiments 2 and 4.

Error Deviation Level	3-5-2 VNN trained by the			3-5-2 VNN trained by the		
	Average Correction		Total Average Correction	Average Correction		Total Average Correction
	Type	Type		Type	Type	
15	99.5	99.8	99.7	88.0	94.0	91.0
13	99.9	99.8	99.9	93.0	94.4	93.7
11	99.9	99.8	99.9	95.0	94.6	94.8
9	99.9	99.8	99.9	96.1	94.8	95.4
7	99.9	99.8	99.9	96.7	94.9	95.8
5	99.9	99.8	99.9	97.0	95.0	96.0
3	99.9	99.8	99.9	97.1	95.1	96.1
1	99.9	99.8	99.9	97.2	95.1	96.2
0	99.9	99.8	99.9	97.3	95.2	96.2

Table. 5.5 Testing results of the 3-5-2 VNN on the two-emitter identification problem with/without noise.

Error Deviation Level	3-8-3 VNN trained by the				3-8-3 VNN trained by the			
	Average Correction			Total Average Correction	Average Correction			Total Average Correction
	Type	Type	Type		Type	Type	Type	
15	63.0	89.3	74.8	75.7	57.8	87.9	70.7	72.2
13	72.2	90.3	74.9	79.1	59.0	89.3	70.8	73.1
11	74.2	92.2	74.9	80.4	60.0	90.2	70.9	73.7
9	79.1	93.7	79.3	84.0	60.8	92.1	75.5	76.1
7	86.3	96.1	85.8	89.4	67.1	93.1	80.5	80.2
5	96.0	97.9	94.1	96.0	75.1	94.0	88.6	85.9
3	99.3	99.3	99.6	99.4	80.5	94.8	92.1	89.1
1	99.6	99.8	99.9	99.8	82.7	95.4	93.7	90.6
0	99.6	99.9	99.9	99.8	83.3	95.8	94.0	91.0

Table 5.6: Testing results of the 3-8-3 VNN on the three-emitter identification problem with/without noise.

Chapter 6

Conclusion

In last year, 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 this 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%.

Compare these two methods, the proposed method is more efficient. The former 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, and it needs only two images to find out the target. Even the target detection is wrong, we still can detect correctly in less one second.

About the guidance system, we have proposed a neural fuzzy scheme for estimating the direction of arrival of moving targets based on the phase differences from an interferometer. In addition, to avoid the discontinuities caused by the input phase transition, we use the quadrature representation of the phase differences. Unlike conventional eigen-based DOA estimator, the proposed scheme requires no large amount of computations and does not need to model signal. With these results achieved in above chapter, the proposed neural fuzzy scheme could be widely applied to military applications (such as reconnaissance and threat reaction) for achieving high accurate DOA for certain electronics support measures. As the targets move, their motion is tracked through a SONFIN which uses the data provided by the most recent output of the sensor array to update the existing estimate of target angles.

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 estimation 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.