A Fuzzy Neural Network and its Application to Motion Detection and Velocity Estimation

by

Abbas Zahedi Kouzani

B.Sc.

Thesis submitted in partial fulfilment of requirements for the degree of

Master of Engineering Science

The University of Adelaide

Faculty of Engineering

Department of Electrical and Electronic Engineering

January, 1995

TABLE OF CONTENTS

ABSTRACT

The advantages of fuzzy sets and neural networks in emulating the human brain capabilities motivated the development of fuzzy neural networks. Various models of fuzzy neurons have been proposed as the basic element of fuzzy neural networks. In this thesis, we introduce a generic fuzzy neuron as an extension of existing fuzzy neuron models. In our model, all the states of activity are given in terms of fuzzy sets with relative grades of membership distributed over the interval $[0, 1]$. The inputs and outputs are fuzzy sets over different universes of discourse. The connection, aggregation, and activation functions, which determine the operation of the neuron, are fuzzy relations. When the inputs to a function are fuzzy sets over the same universe of discourse, the function can be any fuzzy operation in class of triangular norms or triangular conorms. To evaluate the operation of the fuzzy neuron, a fuzzy neural network architecture based on the generic fuzzy neuron has been developed for motion estimation. The five-layer feedforward fuzzy neural network emulates a spatio-temporal image-matching algorithm. Seven simplified versions of fuzzy neurons are defined and utilized in the fuzzy neural network. The results of simulations on thousands of 64×64, 6-bit image frames containing moving objects under different conditions are reported.

DECLARATION

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the University of Adelaide Library, being available for loan and photocopying.

Signature: ---

Date: 31/1/95

ACKNOWLEDGMENTS

I gratefully acknowledge the assistance and support of my principal supervisor, Dr. A. Bouzerdoum. His unlimited support, patience, encouragement and friendship have been vital to the completion of this research. I learned from him much more than I can confess in these few lines.

I also appreciate the guidance and encouragement of my second supervisor, Mr. M.J. Liebelt, which has been a great asset for the progress of this research.

The Ministry of Culture and Higher Education of I.R. Iran should also be thanked for providing me the financial support during this study.

Finally, I must express my gratitude to my mother and father who have been my constant support.

Abbas Zahedi Kouzani

January, 1995

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background and Motivation

One of the last frontiers of science, perhaps its ultimate challenge, is understanding the biological basis of mentation and cognition - how we think, reason, learn, remember, perceive and act, and implementing artificial systems that do the kinds of things we do. From mechanical automata in past centuries to electronic devices now, we have tried to make hardware and software that act like us, or at least some significant part of us. Much of current research is devoted to modelling various aspects of the human brain, the thing which seems to be the most highly developed in humans.

The human brain is superior to all kinds of modern day computers in processing cognitive information, the information acquired by the peripheral nervous system. Whereas most of the traditional mathematical tools are based upon some absolute measures of information, the cognitive information is in the form of relative grades. Unlike the computational functions of traditional computers, the human brain acts upon the relative grades of raw information acquired by the neural sensory system.

1.2 Fuzzy Sets

To deal properly with uncertainties and imprecisions which arise from human thinking, mentation, cognition and perception, some special tools and techniques are required. In 1965, Lotfi A. Zadeh published his first celebrated paper on *fuzzy sets* as a means for representing uncertainty [58]. The type of uncertainty that this theory was meant to handle has as its roots

the type of imprecision and ambiguity which is prevalent in human discourse and thought. The theory of fuzzy sets is based upon the notion of a relative graded membership, and so is the function of the mentation and cognition. Zadeh writes: "What is central about fuzzy theory is that, it aims of modelling the imprecise modes of reasoning that play an essential role in the remarkable human ability to make rational decisions in an environment of uncertainty and imprecision. This ability depends, in turn, on our ability to infer an approximate answer to a question based on a store of knowledge, that is inexact, incomplete, or not totally reliable" [57].

Fuzzy set theory emerged by challenging the basic assumptions of three theories: sharp boundaries in classical set theory, classical logic that each proposition must either be true or false, and additivity in classical measure theory, particularly probability theory. The first challenge to classical set theory came in the 1965 with the fuzzy set theory [58]. Next came fuzzy logic, which emerged as an outgrowth of fuzzy set theory [19], and a generalization of the Lukasiewicz infinitive-valued logic defined on the unit interval [43]. The third challenge appeared in 1974, when fuzzy measure theory was introduced [45].

Although fuzzy theory encountered lots of skepticism, it became quite strong in the 1970s. New important concepts were introduced such as fuzzy numbers, fuzzy topology, and various kinds of fuzzy relations. An extension principle appeared in 1975, by which other concepts and theories of classical mathematics can readily be fuzzified [60]. Researchers developed a theory of dynamic fuzzy systems, investigated operators for aggregating fuzzy sets in a comprehensive way, introduced fuzzy sets of more general types, and formulated various categories of fuzzy sets and relations. Some ideas of prospective applications of fuzzy theory also emerged in the 1970s: fuzzy control [37][55][56][59], fuzzy decision making [61], and fuzzy pattern recognition [5]. In spite of a general lack of interest, fuzzy set theory continued to advance rapidly. Applications of the theory, however, lagged behind the theory itself, until the 1980s. Since then, hundreds of articles on fuzzy set theory and its applications have been published in more than a dozen fields.

1.3 Artificial Neural Networks

Another difference between the human brain and conventional digital computers is its

structure. It is believed that the brain consists of an enormous number of neurons highly interconnected by links with variable strengths, operating in parallel. Conventional computers, on the other hand, execute sets of instructions sequentially. To achieve the massively parallel distributed processing features of the brain, neural networks have been studied extensively since the influential work of McCulloch and Pitts, on neurons modelled as discrete decision-making elements with threshold logic outputs in 1943 [38]. In its simplest conception, the neural network may be described as a collection of neurons which interact among themselves through a highly interconnected synaptic network. The most striking aspect of such a network is the highly distributed manner in which information is stored and the high degree of parallelism by which it is processed.

The limitations of McCulloch and Pitts model were recognized by Rosenblatt [44] in 1950's. First, the behaviour of the model was difficult to predict analytically. Second, the available computational resources of the day were not adequate to simulate the proposed system. Rosenblatt stirred considerable interest and activity in the field when he designed and developed the Perceptron [44]. The Perceptron had three layers which could learn to associate a given input to an output. However, the system exhibited complex adaptive behaviour. The ADALINE network was developed shortly after the Perceptron by Widrow and Hoff in the 1960's [51]. It employed a more sophisticated learning procedure than the Perceptron learning rule. In 1967, Amari established a mathematical basis for a learning theory (error-correction method) [1]. In 1988, he and Maginu developed a self-organizing network as a model for associative memory [2]. Initial work in associative memory was published by Anderson et al. [3], and by Kohonen [20][21]. The former developed a neural network called *Brain-State-in-a-Box*, and the latter developed *Kohonen's self-organizing feature maps* and *learning vector quantizor*. The Hopfield network [10][11][12][47] is also a well-known associative network today. *Back-propagation networks* are probably the most well-used neural network today. Several people published the back-propagation technique independently. The earliest was published by Werbos [50]. *Adaptive-Resonance-Theory networks* (ART) based on biologically plausible models of learning was introduced by Grossberg [7] and developed by Carpenter and Grossberg [6]. In contrast to feedforward networks, ART networks are much more comprehensive set of neurophysiological phenomena. Bart Kosko has developed several lines of research with his *hetero-associative networks*, which includes the *Bidirectional-Associative-Memory* (BAM), the *Fuzzy-Cognitive-Maps*, and the *Fuzzy-*

Associative-Memory [22][23][24][25][26]. Since McCulloch and Pitts publication, significant progress has been made in the field of neural networks on many fronts, and many publications have emerged which attracted a great deal of attention and funding for further research.

1.4 Fuzzy Neural Networks

To emulate the capability of the human brain (learning, remembering, reasoning, intelligence, perceiving, etc.) in a machine, the attempt to utilize fuzzy sets in the context of neural networks began soon after the inception of the fuzzy set theory. The first ideas for developing a hybrid architecture, to enhance computing capabilities of fuzzy sets by accepting mechanisms of learning and using architectures of neural networks, were published by Wee and Fu [49] in 1969. In their work, a class of fuzzy automata was formulated and a nonsupervised learning scheme for automatic control and pattern recognition was proposed. In 1975, S.C. Lee and E.T. Lee extended the McCulloch-Pitts model of a neuron to a fuzzy neuron with a fuzzy activity rather than an all-or-none process [33]. An architecture of a fuzzy neural network based on the McCulloch-Pitts architecture was proposed in which the components were fuzzy neurons. In 1985, Keller and Hunt introduced a fuzzy perceptron [17]. The proposed fuzzy perceptron was used in pattern recognition to determine linear decision boundaries. Yamakawa and Tomoda described a simple fuzzy neuron model and used it in a neural network architecture for character recognition [52]. They did not present a specific learning algorithm for the network; however, Yamakawa and Furukawa described later an example-based learning algorithm for membership functions of the fuzzy neuron [54]. The proposed algorithm was tested for recognition of hand-written characters. Pedrycz studied a problem of learning in neural networks with max-min composition operations [41]. He indicated analogies between relational structures and a certain class of neural networks. Takagi and Hayashi reported a formulation for determining fuzzy inference rules and a method of fuzzy reasoning using neural network models [46]; a structure of a neural-network-driven fuzzy reasoning system was given and two applications of this method were presented.

An *Adaptive-Network-based Fuzzy Inference System* (ANFIS), a fuzzy inference system implemented in the framework of adaptive networks, was introduced by Jang [15]. The ANFIS could construct an input-output mapping based on both human knowledge (fuzzy if-

then rules) and stipulated input-output data pairs. He employed the ANFIS architecture to model nonlinear functions, identify nonlinear components on-line in a control system, and predict a chaotic time series. Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks was published by Keller and Tahani [18] in 1992. Kosko proposed a fuzzy associative memory (FAM) which defines mappings between fuzzy sets [26]. FAM used fuzzy matrices instead of fuzzy neurons to represent fuzzy associations.

Besides the activities reported above, some recently developed fuzzy neurons and fuzzy neural networks have been found more attractive. Gupta and Knopf proposed a mathematical model of a fuzzy neuron and neural network architecture for control application [8]. In the proposed network each neuron represented a fuzzy inference rule. The neurons could learn from experience by adapting the synaptic weighting functions. Pedrycz and Rocha hypothesized the models of neurons based on logic-oriented processing mechanisms of fuzzy sets [42]. Two broad classes of aggregative (named AND and OR neurons) and referential neurons (such as matching, dominance, and inclusion neurons) have been presented. Furthermore, learning procedure for basic logic neurons and various topologies of neural networks have been discussed. Hirota and Pedrycz introduced a neural network-based model of logical connectives [9]. The basic processing element of this network consists of two types of OR and AND neurons structured into a three layer network. A supervised learning for the OR/AND neuron and a realization of a pseudo median filter in which the OR/AND neurons play an important role have been studied. Lin and Song proposed a fuzzy neural network with a simple structure of three layers with different types of fuzzy neurons [35]. They evaluated the fuzzy neural network in a simulation study of inverse kinematics of a two degrees of freedom manipulator. Kwan and Cai defined a fuzzy neuron, introduced four types of fuzzy neurons, and proposed a four-layer feedforward fuzzy neural network with its learning algorithm [28]. They have applied the fuzzy neural network to recognize shifted and distorted training patterns.

In this thesis, a generic model of a fuzzy neuron will be introduced as the basic element of our fuzzy neural system. The generic fuzzy neuron, which is inspired by Gupta and Knopf's fuzzy neuron, is a generalization of the existing models of fuzzy neurons. The architecture of a five-layer feedforward fuzzy neural network is proposed for motion estimation. We define seven simplified types of fuzzy neurons to be employed in different layers of the proposed fuzzy

neural network architecture. We also present and discuss simulation results performed on synthetic images.

1.5 Overview

The thesis is organised into 7 chapters including the current introductory chapter. Chapter 2 summarizes some of the basic concepts of fuzzy set theory which will be needed in this thesis. Chapter 3 provides an introduction to the field of neural networks and summarizes the theoretical results concerning multilayer feedforward neural networks. A discussion of four developed models of fuzzy neurons and fuzzy neural networks is given in Chapter 4. To our knowledge, they are the most important models among existing fuzzy neural systems. Chapter 5 discusses the proposed generic fuzzy neuron and the fuzzy neural network architecture for motion estimation. An algorithm for motion estimation is also presented in this chapter. Chapter 6 discusses the simulation results. Finally, the concluding remarks and future directions of this research are given in Chapter 7.

Chapter 2

THE BASICS OF FUZZY SET THEORY

2.1 Introduction

Fuzzy set theory was introduced by L.A. Zadeh in his paper "Fuzzy Sets" published in 1965 [58] as a means for representing uncertainty. The type of uncertainty that this theory was meant to handle has the type of imprecision and ambiguity which is prevalent in human discourse and thought. The theory has now matured into a wide-ranging collection of concepts and techniques for dealing with complex phenomena which do not lend themselves to analysis by classical methods. The aim of this chapter is to summarize some of the basic concepts of fuzzy set theory which will be needed in this thesis. The chapter is organised as follows: First the idea of fuzzy sets is presented. Then basic operations with fuzzy sets are explained. After that, the definition of fuzzy relations and fuzzy relational equations are given. Next, the concepts of linguistic variables, fuzzy logic, and fuzzy approximate reasoning are discussed. Finally, a brief review of fuzzy inference systems and their components is provided. The contents of this chapter are based on the following publications: [29][31][32][48][53][57][58][62]. The reader is referred to the mentioned publications where exact definitions and more detailed explanations are provided.

2.2 Quantification of Ambiguity

Linguistic terms and numerical values are classified into three categories in accordance with their meanings. Deterministic words, e.g., "male" and "female," "dead" and "alive," have truth values of 0 or 1 corresponding to NO or YES, respectively. In other words, if one is asked, "Are you male?," "Are you alive?," then the answer can be made with a "YES" or "NO"

statement. Exact numerical values, e.g., "exactly $25°C$," "$43g$," etc., are in the same category; that is they have truth values of 0 and 1. The deterministic words and numerical values have neither flexibility nor intervals. They are characterized by a characteristic function as shown in Figure 2.1(a). The word "exactly $15°C$" means only one single point of temperature at $15°C$, thus, this type of term or value is called a *singleton.*

Even in the scientific analysis, when exact values are preferred, numerical intervals are sometimes used to represent flexible values. For example, "The comfortable room temperature for human beings is $20°C$-$30°C$." The characteristic function representing a numerical interval "$20°C$-$30°C$" is shown in Figure 2.1(b). Truth values for temperatures are given by YES or NO in this case as well as the case of a singleton. This interval can be regarded as a set of singletons. Thus, this type of deterministic interval is called a *crisp set.* Crisp sets are also adopted to represent linguistic terms in knowledge in artificial intelligence.

Natural languages are used for easy and efficient communication in our life. Whereas numerical values and deterministic linguistic terms used in artificial intelligence are well-defined, the natural language is usually intuitive and includes some kind of uncertainty, called ill-defined, e.g., "Cool it a little bit." This type of natural language is referred to as a *fuzzy linguistic term.* The meaning of the fuzzy linguistic term is defined by a characteristic function as shown in Figure 2.1(c). This function is specifically called a *membership function,* because it indicates a grade of membership of each element in a fuzzy linguistic term. A membership function exhibits a continuous curve from 0 to 1 or vice versa. For instance, the fuzzy linguistic terms "cold," "a little cool," "cool," "warm," "hot," and "very hot" are indicated in Figure 2.1(c).

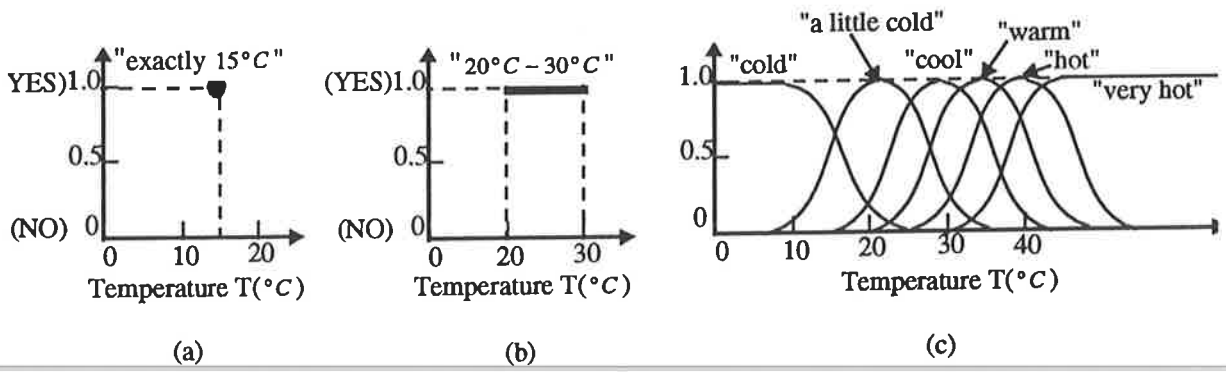


Figure 2.1: (a) A singleton. (b) A crisp interval. (c) Fuzzy sets.

2.3 Fuzzy Sets

Let X be a space of points (objects) which could be discrete or continuous. X is called the *universe of discourse* and x represents the generic element of X.

Definition 2-1: A *fuzzy set* A in a universe of discourse X is characterized by a *membership function* μ_A, which associates with each point in X a real number in the interval [0,1] namely, $\mu_A : X \rightarrow [0,1]$. The value of $\mu_A(x)$ at x represents the "grade of membership" of x in A. A fuzzy set may be viewed as a generalization of the concept of an ordinary set whose membership function takes two values $\{0, 1\}$. Thus a fuzzy set A in X may be represented as a set of ordered pairs of a generic element x and its grade of membership function

$$A = \{(x,\mu_A(x)) : x \in X\} \tag{2.1}$$

Definition 2-2: The *support* of a fuzzy set A, $S(A)$, is the crisp set

$$S(A) = \{x \in X : \mu_A(x) > 0\} \tag{2.2}$$

Definition 2-3: The *crossover point* is the element x in X at which

$$\mu_A(x) = 0.5 \tag{2.3}$$

Definition 2-4: A *fuzzy singleton* is a fuzzy set whose support is a single point in X. If A is a fuzzy singleton whose support is the point x, we write

$$A = \mu/x \tag{2.4}$$

where μ is the grade of membership of x in A.

A fuzzy set A may be viewed as the union of its constituent singletons. On this basis, A may be represented in the form

$$A = \int_X \mu_A(x)/x \tag{2.5}$$

where the integral sign stands for the union of the fuzzy singletons $\mu_A(x)/x$. If A has a finite support $\{x_1, x_2, \ldots, x_n\}$, then (2.5) may be replaced by

$$A = \mu_1/x_1 + \mu_2/x_2 + \ldots + \mu_n/x_n \tag{2.6}$$

or, more compactly

$$A = \sum_{i=1}^{n} \mu_i / x_i \tag{2.7}$$

in which μ_i is the grade of membership of x_i in A. It should be noted that the "+" sign in (2.6) denotes the union rather than the arithmetic sum.

Definition 2-5: A fuzzy set is *empty* if and only if its membership function is identically zero on X.

Definition 2-6: The α-*level set* or α-*cut* of a fuzzy set A is a crisp subset of X and is denoted by (Figure 2.2 shows a continuous case)

$$A_\alpha = \{x \in X : \mu_A \geq \alpha\} \tag{2.8}$$

The "strong α-level set" or "strong α-cut" is denoted by

$$A_\alpha = \{x \in X : \mu_A > \alpha\} \tag{2.9}$$

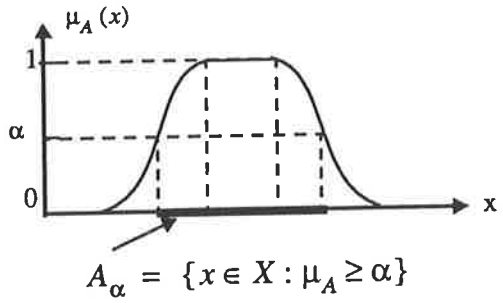


Figure 2.2: An α-level set.

Definition 2-7: A fuzzy set A is *convex* if and only if:

$$\mu_A [\lambda x_1 + (1-\lambda) x_2] \geq min [\mu_A (x_1), \mu_A (x_2)] \tag{2.10}$$

for all x_1 and x_2 in λ and all λ in [0,1]. Alternatively, a fuzzy set is convex if all α-cuts are convex. Figure 2.3 shows an example of a convex fuzzy set (a), and an example of a non-convex fuzzy set (b).

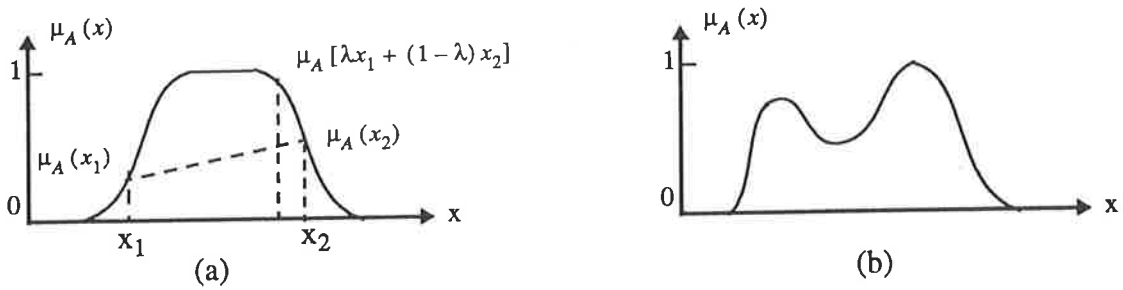


Figure 2.3: (a) A convex fuzzy set. (b) A non-convex fuzzy set.

Definition 2-8: A fuzzy set A is *normal* if and only if supremum of $\mu_A(x)$ over X is unity. A fuzzy set is *subnormal* if it is not normal.

Definition 2-9: For a normal and convex fuzzy set, if α-cut is a closed interval, it is called a *fuzzy number*. Figure 2.4 shows such fuzzy sets, which could be called "approximately 3" and "approximately 7."

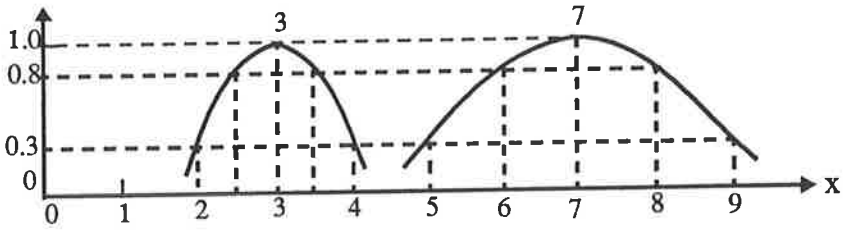


Figure 2.4: Fuzzy numbers 3 and 7.

2.4 Operations With Fuzzy Sets

Let A and B be two fuzzy sets in X with membership functions μ_A and μ_B respectively. The set-theoretic operations with fuzzy sets are defined via their membership functions.

Definition 2-10: Two fuzzy sets A and B are *equal*, $A = B$, if and only if

$$\mu_A(x) = \mu_B(x) \quad \text{for all } x \in X \tag{2.11}$$

Definition 2-11: A is *contained in* B, $A \subseteq B$, if and only if

$$\mu_A(x) \leq \mu_B(x) \tag{2.12}$$

Definition 2-12: The *complement* of a fuzzy set A is denoted by $\bar{A}$ and its membership

function is defined by

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{2.13}$$

Definition 2-13: The *union* of two fuzzy sets A and B, $A \cup B$, is the fuzzy set defined by the following membership function

$$\mu_{A \cup B}(x) = max[\mu_A(x), \mu_B(x)] \tag{2.14}$$

Definition 2-14: The *intersection* of two fuzzy sets A and B, $A \cap B$, is the fuzzy set defined by the following membership function

$$\mu_{A \cap B}(x) = min[\mu_A(x), \mu_B(x)] \tag{2.15}$$

Definition 2-15: The *algebraic sum* of two fuzzy sets A and B, $A + B$, is the fuzzy set defined by the following membership function

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x) \tag{2.16}$$

Definition 2-16: The *algebraic product* of two fuzzy sets A and B, $A \cdot B$, is the fuzzy set defined by the following membership function

$$\mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x) \tag{2.17}$$

Definition 2-17: The *bounded sum* of two fuzzy sets A and B, $A \oplus B$, is the fuzzy set defined by the following membership function

$$\mu_{A \oplus B}(x) = min[1, \mu_A(x) + \mu_B(x)] \tag{2.18}$$

Definition 2-18: The *bounded difference* of two fuzzy sets A and B, $A \ominus B$, is the fuzzy set defined by the following membership function

$$\mu_{A \ominus B}(x) = max[0, \mu_A(x) - \mu_B(x)] \tag{2.19}$$

Definition 2-19: The *absolute difference* of two fuzzy sets A and B, $A \boxminus B$, is the fuzzy set defined by the following membership function

$$\mu_{A \boxminus B}(x) = |\mu_A(x) - \mu_B(x)| \tag{2.20}$$

Definition 2-20: The *algebraic division* of two fuzzy sets A and B, $A \boxslash B$, is the fuzzy set defined by the following membership function

$$\mu_{A \boxslash B}(x) = min\left\{1, \frac{\mu_A(x)}{\mu_B(x)}\right\} \tag{2.21}$$

Definition 2-21: If $A_1, A_2, \ldots, A_n$ are fuzzy sets in $X_1, X_2, \ldots, X_n$, respectively, the *cartesian product* of $A_1, A_2, \ldots, A_n$, $A_1 \times A_2 \times \ldots \times A_n$, is a fuzzy set in the product space

$X_1 \times X_2 \times \ldots \times X_n$ with the following membership function:

$$\mu_{A_1 \times A_2 \times \ldots \times A_n}(x_1, x_2, \ldots, x_n) = min\,[\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n)] \quad (2.22)$$

A general class of aggregation operators for the intersection of fuzzy sets is called *triangular norms* or *t-norms*. The min, cartesian product, and absolute difference operators considered above belong to this class. They can be characterized as follows:

Definition 2-22: *t-norms* are two-valued functions from $[0, 1] \times [0, 1]$ into $[0, 1]$, which satisfy the following conditions:

$$1\text{-}\ t(0, 0) = 0;\ t(\mu_A(x), 1) = t(1, \mu_A(x)) = \mu_A(x),\ x \in X \quad (2.23)$$

$$2\text{-}\ t(\mu_A(x), \mu_B(x)) \leq t(\mu_C(x), \mu_D(x))\ if(\mu_A(x) \leq \mu_C(x) \text{ and } \mu_B(x) \leq \mu_D(x)) \quad (2.24)$$

$$3\text{-}\ t(\mu_A(x), \mu_B(x)) = t(\mu_B(x), \mu_A(x)) \quad (2.25)$$

$$4\text{-}\ t(\mu_A(x), t(\mu_B(x), \mu_C(x))) = t(t(\mu_A(x), \mu_B(x)), \mu_C(x)) \quad (2.26)$$

Corresponding to the class of intersection operators, a general class of aggregation operators for the union of fuzzy sets is called *triangular conorms* or *s-norms* (*t-conorms*). The max, algebraic sum, and bounded sum belong to this class.

Definition 2-23: *s-norms* are two-placed functions s, which map from $[0, 1] \times [0, 1]$ into $[0, 1]$. These properties are formulated with the following conditions:

$$1\text{-}\ s(1, 1) = 1;\ s(\mu_A(x), 0) = s(0, \mu_A(x)) = \mu_A(x),\ x \in X \quad (2.27)$$

$$2\text{-}\ s(\mu_A(x), \mu_B(x)) \leq s(\mu_C(x), \mu_D(x))\ if(\mu_A(x) \leq \mu_C(x) \text{ and } \mu_B(x) \leq \mu_D(x)) \quad (2.28)$$

$$3\text{-}\ s(\mu_A(x), \mu_B(x)) = s(\mu_B(x), \mu_A(x)) \quad (2.29)$$

$$4\text{-}\ s(\mu_A(x), s(\mu_B(x), \mu_C(x))) = s(s(\mu_A(x), \mu_B(x)), \mu_C(x)) \quad (2.30)$$

2.5 Fuzzy Relations

Ambiguous relationships such as "x and y are almost equal," "x is much taller than y," and "x and y look very simillar," are often topics of everyday conversation, however, expressing these kinds of ambiguous relationships in terms of ordinary relations is very difficult. *Fuzzy relations* are what makes it possible to express these ambiguous relationships.

Definition 2-24: *Fuzzy relation* R from set X to set Y is a fuzzy set in the direct product $X \times Y = \{ (x, y) : x \in X, y \in Y \}$, and is characterized by the following membership function:

$\mu_R : X \times Y \rightarrow [0,1]$. When $X = Y$, R is known as a fuzzy relation on X. As a generalization of fuzzy relations, the *n-ary fuzzy relation* R in $X_1 \times X_2 \times \ldots \times X_n$ is a fuzzy set expressed as:

$$R = \{ ((x_1, x_2, \ldots, x_n), \mu_R (x_1, x_2, \ldots, x_n)) : (x_1, x_2, \ldots, x_n) \in X_1 \times X_2 \times \ldots \times X_n \} \quad (2.31)$$

Definition 2-25: A fuzzy relation R in $X \times Y$ can be expressed by an $m \times n$ matrix called a *fuzzy matrix*, $M = [m_{ij}]$, where m_{ij} has a value in the interval $[0, 1]$. The m_{ij} indicates $\mu_R (x_i, y_j)$ in the matrix.

Definition 2-26: A *fuzzy graph* expresses a fuzzy relation R in $X \times Y$, in a graph. For each $\mu_R (x_i, y_j)$ we make x_i and y_i vertices and add the grade $\mu_R (x_i, y_j)$ to the arc from x_i to y_i.

Definition 2-27: If R is fuzzy relation in $X \times Y$ and S is fuzzy relation in $Y \times Z$, the *composition* of R and S, $R \circ S$, is a fuzzy relation in $X \times Z$ defined as follows

$$R \circ S \leftrightarrow \mu_{R \circ S} (x, z) = \max_y \{ min \{ \mu_R (x, y), \mu_S (y, z) \} \} \quad (2.32)$$

Definition 2-28: If R is fuzzy relation in $X \times Y$ and S is fuzzy relation in $Y \times Z$, the *max-* composition* of R and S, $R \overset{\circ}{*} S$, is a fuzzy relation in $X \times Z$ defined as follows

$$R \overset{\circ}{*} S \leftrightarrow \mu_{R \overset{\circ}{*} S} (x, z) = \max_y \{ \mu_R (x, y) * \mu_S (y, z) \} \quad (2.33)$$

where * could be any operator in class of triangular norms, namely, minimum, cartesian product, or bounded difference.

Definition 2-29: If R is fuzzy relation in $X \times Y$ and S is fuzzy relation in $Y \times Z$, the *-* *composition* of R and S, $R \overset{*}{\underset{*}{\circ}} S$, is a fuzzy relation in $X \times Z$ defined as follows

$$R \overset{*}{\underset{*}{\circ}} S \leftrightarrow \mu_{R \overset{*}{\underset{*}{\circ}} S} (x, z) = \overset{*}{y} \{ \mu_R (x, y) * \mu_S (y, z) \} \quad (2.34)$$

where * could be any operator in class of triangular norms or triangular conorms.

2.6 Fuzzy Relational Equations

As described in the previous section, if we let A be a fuzzy set in X and R be a fuzzy relation

in $X \times Y$, the composition of A and R, $A \circ R$, is defined as

$$\mu_{A \circ R}(y) = \max_{x} \{ min \{ \mu_A(x), \mu_R(x, y) \} \} \tag{2.35}$$

and is a fuzzy set in Y which is called B. In other words, $B = A \circ R$. If we view A as the fuzzy input, B as the fuzzy output, and R as the fuzzy system, we can think of $B = A \circ R$ as expressing the input-output relation of a system with fuzzy input and fuzzy output. If the fuzzy input A and fuzzy relation R are given, the fuzzy output B can be founded by taking the composition of A and R. The equation $B = A \circ R$ is called a *fuzzy relational equation.*

2.7 Linguistic Variables

A linguistic variable can be regarded as a variable whose value is not numbers but words or sentences in a natural or artificial language.

Definition 2-30: A *linguistic variable* is characterized by a quintuple $(x, T(x), U, G, M)$ in which x is the name of the variable; $T(x)$ denotes the term set of x, that is, the set of names of linguistic variables of x, with each value being a fuzzy variable denoted generically by x and ranging over a universe of discourse U which is associated with the base variable u; G is a syntactic rule for generating the name, X, of values of x; and M is a *semantic rule* for associating with each X its meaning, $M(X)$, which is a fuzzy subset of U. A particular X, that is a name generated by G, is called a *term.*

For instance, if "temperature" is interpreted as a linguistic variable, then its term set could be T(temperature) = {cold, a little cold, cool, warm, hot, very hot} where each term in T(temperature) is characterized by a fuzzy set in a universe of discourse U= [0, 100]. These terms can be characterized as fuzzy sets whose membership functions are shown in Figure 2.1(c).

2.8 Fuzzy Logic and Approximate Reasoning

Logics as bases for reasoning can be distinguished by their three context-independent items:

truth values, operators, and tautologies (reasoning procedure).

In Boolean logic, truth values can be true or false and by means of these truth values the operators are defined via truth tables. In multivalued logic, truth values can be true or false or have an intermediate value, which may be an element of a finite or infinite truth value set T. The uniqueness of interpretation of truth tables, which is very convenient in Boolean logic, disappears immediately because many truth tables in multivalued logic are similar. In fuzzy logic, the truth values are allowed to range over the fuzzy subsets of T. Whereas operators in Boolean logic and multivalued logic are normally defined by the tabulation of truth values in truth tables, in fuzzy logic, the tabulation of the truth values for operators is not possible because the number of truth values is infinite. However, truth values can be tabulated in terms of the linguistic variable "Truth" for a finite number of terms, such as true, not true, very true, false, more or less true, an so on.

In traditional logic the main tools of reasoning are tautologies, for instance, the *modus ponens*, that is

Premise	A is true
Implication	If A then B
Conclusion	B is true

A and B are statements or propositions (crisply defined) and the B in the conditional statement is identical to the B of the conclusion. Two generalizations of the modus ponens are

1- To allow statements that are characterized by fuzzy sets.

2- To relax (slightly) the identity of the "B s" in the implication and the conclusion.

In approximate reasoning there are two important fuzzy implication inference rules named the *generalized modus ponens* (GMP) and the *generalized modus tollens* (GMT):

Premise	x is A'
Implication	If x is A then y is B
Conclusion	y is B'

(GMP)

Premise	y is B'
Implication	If x is A then y is B
Conclusion	x is A'

(GMT)

The fuzzy implication inference is based on the compositional rule of inference for approximate reasoning suggested by Zadeh in 1973 [59]. Here fuzzy sets A, A', B, B' are

introduced via linguistic variables x, y instead of crisp sets in the traditional logic. The GMP, which reduces to modus ponens when $A = A'$ and $B = B'$, is closely related to the forward data-driven inference which is particularly used in fuzzy logic control. The GMT, which reduces to modus tollens, when $A'=$ not A and $B'=$ not B, is closely related to the backward goal-driven inference which is commonly used in expert systems, especially in the area of medical diagnosis.

Definition 2-31: Let $R(x)$, $R(x, y)$, and $R(y)$, $x \in X, y \in Y$, be fuzzy relations in X, $X \times Y$, and Y respectively. Let A and B denote particular fuzzy sets in X and $X \times Y$. The *compositional rule of inference* asserts, that the solution of the relational assignment equations $R(x) = A$ and $R(x, y) = B$ is given by $R(y) = A \circ B$, where $A \circ B$ is the composition of A and B. If the max-* composition is used, $R(y) = A \overset{\circ}{*} B$, this represents *max-* compositional rule of inference.*

2.9 Fuzzy Inference Systems

Fuzzy inference systems are also known as *fuzzy-rule-based systems*, *fuzzy associative memories*, or *fuzzy controllers* when used as controllers. As shown in Figure 2.5 a fuzzy inference system consists of five functional blocks: fuzzification interface, rule base, database, decision-making unit, and defuzzification interface. Usually the rule base and the data base are jointly referred to as the knowledge base. The above components are discussed in more detail in the following.

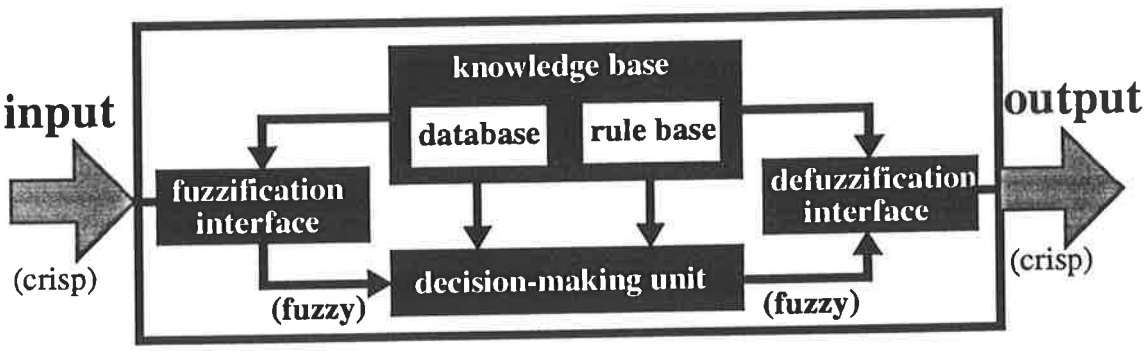


Figure 2.5: Fuzzy inference system.

2.9.1 The Fuzzification Interface

Fuzzification is related to the vagueness and imprecision in a natural language. It is a subjective valuation which transforms a measurement into a valuation of a subjective value, and hence it could be defined as a mapping from an observed input space to fuzzy sets in a certain universe of discourse. A fuzzification operator conceptually converts a crisp value into a fuzzy singleton. It interprets an input x_0 as a fuzzy set A whose membership function $\mu_A(x)$ is zero except at the point x_0, at which $\mu_A(x_0)$ is one.

2.9.2 The Rule Base

In a fuzzy interface system, the dynamic behaviour is characterized by a set of linguistic description rules based on expert knowledge. The expert knowledge is usually of the form

IF a set of conditions are satisfied THEN a set of consequences can be inferred (2.36)

Since the antecedents and the consequences of these if-then rules are associated with fuzzy concepts (linguistic terms), they are called "fuzzy rules." Several linguistic variables might be involved in the antecedents (multi-input) and the conclusions (multi-output) of these rules. *Multi-input-multi-output fuzzy systems* are referred to as MIMO fuzzy systems and *multi-input-single-output* as MISO fuzzy systems. The rule base for a MISO fuzzy system could have the form:

$$\begin{aligned}
&\text{rule 1 : if } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } z \text{ is } C_1,\\
&\text{rule 2 : if } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } z \text{ is } C_2,\\
&\qquad\vdots\\
&\text{rule } n \text{ : if } x \text{ is } A_n \text{ and } y \text{ is } B_n \text{ then } z \text{ is } C_n.
\end{aligned} \tag{2.37}$$

where x, y, and z are linguistic variables representing two input variables and one output variable. A_i, B_i, and C_i are linguistic terms of the linguistic variables x, y, and z in the universe of discourse U, V, and W, respectively, with $i = 1, 2, \ldots, n$; and an implicit sentence connective also links the rules into a rule base.

A fuzzy rule, such as "if (x is A_i and y is B_i) then (z is C_i)," is implemented by a fuzzy

implication (fuzzy relation) R_i and is defined as follows:

$$\mu_{R_i} = \mu\,(A_i \text{ and } B_i \rightarrow C_i)\ (u, v, w) = [\mu_{A_i}(u) \text{ and } \mu_{B_i}(v)] \rightarrow \mu_{C_i}(w) \qquad (2.38)$$

where $(A_i \text{ and } B_i)$ is fuzzy set $A_i \times B_i$ in $U \times V$; $R_i = (A_i \text{ and } B_i) \rightarrow C_i$ is a fuzzy relation in $U \times V \times W$; and $\rightarrow$ denotes a fuzzy implication function.

2.9.3 The Database

The database defines the membership functions of the fuzzy sets (linguistic terms) used in the fuzzy rules. There are two methods used for defining fuzzy sets depending on whether the universe of discourse is discrete or continuous: i) numerical, ii) functional.

In the former, the grade of membership function of a fuzzy set is represented as a vector or numbers whose dimension depends on the degree of discretization. In the latter, the membership function of a fuzzy set is described in a functional form, typically a bell-shaped (gaussian) function, triangle-shaped function, or trapezoid-shaped function. The functional definition can readily be adapted to a change in the normalization of a universe.

2.9.4 The Decision-Making Logic

The decision making logic is the kernel of a fuzzy inference system. It infers the output of the fuzzy inference system (in terms of fuzzy sets) on the basis of fuzzified input and the knowledge base. Several types of fuzzy reasoning (inference operations upon fuzzy rules) have been proposed. Depending on the types of fuzzy reasoning and fuzzy if-then rules employed, most inference systems can be classified into four types described in the following (see Figure 2.6). For simplicity assume that we have two fuzzy rules as follows:

$$\begin{aligned} &\text{rule 1: if } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } z \text{ is } C_1, \\ &\text{rule 2: if } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } z \text{ is } C_2, \end{aligned} \qquad (2.39)$$

The inputs are usually measured by sensors and are crisp. In some cases it may be expedient to convert the input data into fuzzy sets. In general, however, a crisp value may be treated as a fuzzy singleton. Then the firing strengths α_1 and α_2 of the first and second rules may be expressed as

$$\alpha_1 = \min\ (\mu_{A_1}(x_0), \mu_{B_1}(y_0)) \tag{2.40}$$

$$\alpha_2 = \min\ (\mu_{A_2}(x_0), \mu_{B_2}(y_0)) \tag{2.41}$$

where $\mu_{A_i}(x_0)$ and $\mu_{B_i}(y_0)$ play the role of the degrees of partial match between the user-supplied data and the data in the rule base. These relations play a central role in the four types of fuzzy reasoning described in the following:

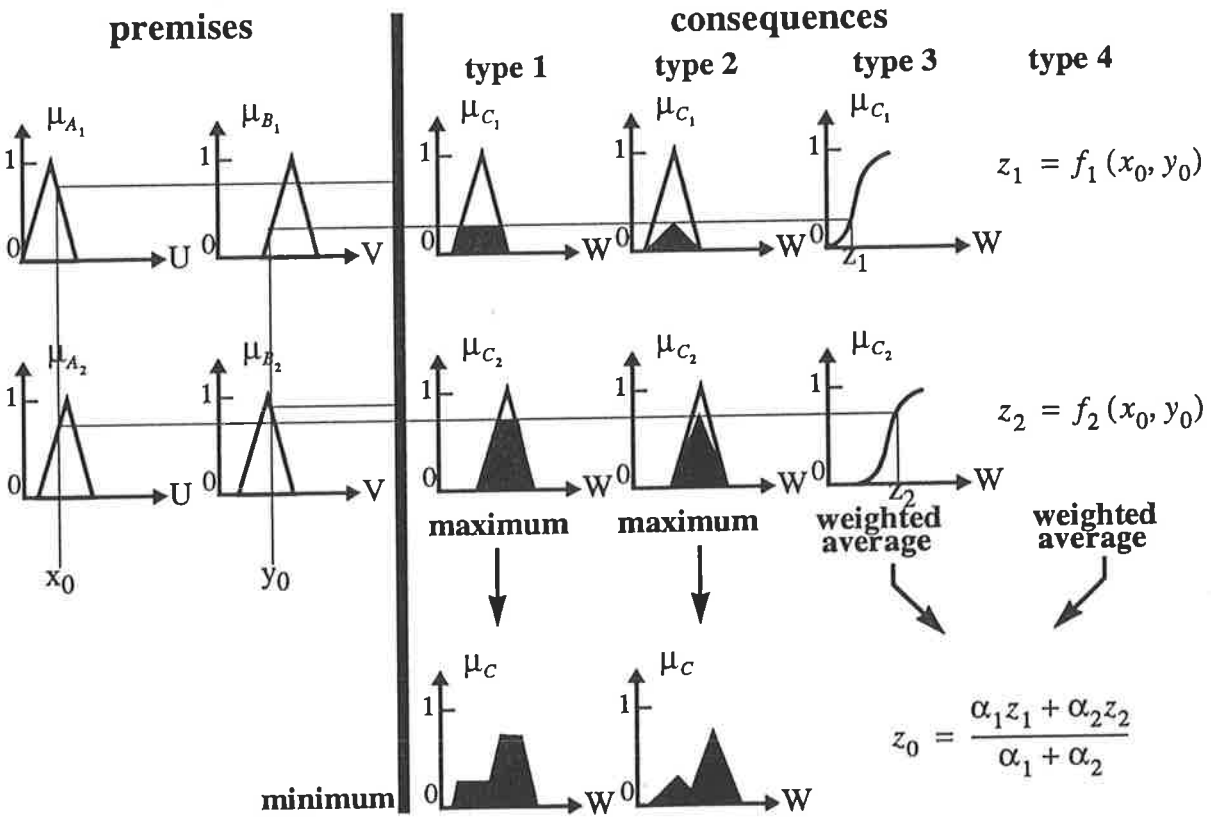


Figure 2.6: Diagrammatic representation of fuzzy reasonings.

1) Type 1

Fuzzy reasoning of this type is associated with the use of minimum operation rule R_c as a fuzzy implication function. In this mode of reasoning the ith rule leads to the decision

$$\mu_{C_i'}(w) = \min\ (\alpha_i, \mu_{C_i}(w)) \tag{2.42}$$

which implies that the membership function μ_C of the inferred consequences C is pointwise given by

$$\mu_C(w) = \max(\mu_{C_1'}, \mu_{C_2'}) = \max([\min(\alpha_1, \mu_{C_1}(w))], [\min(\alpha_2, \mu_{C_2}(w))]) \quad (2.43)$$

2) Type 2

Fuzzy reasoning of this type is based on the use of product operation rule as a fuzzy implication function. In this mode of reasoning the ith rule leads to the decision

$$\mu_{C_i'}(w) = \alpha_i \cdot \mu_{C_i}(w) \quad (2.44)$$

Consequently, the membership function μ_C of the inferred consequence C is pointwise given by

$$\mu_C(w) = \max(\mu_{C_1'}, \mu_{C_2'}) = \max([\alpha_1 \cdot \mu_{C_1}(w)], [\alpha_2 \cdot \mu_{C_2}(w)]) \quad (2.45)$$

3) Type 3

Fuzzy reasoning of this type is a simplified method based on the fuzzy reasoning of the first type in which the membership functions of fuzzy sets A_i, B_i, and C_i are monotonic. The result inferred from the first rule is α_1 such that $\alpha_1 = C_1(z_1)$. The result inferred by from the second rule is α_2 such that $\alpha_2 = C_2(z_2)$. Correspondingly, a crisp action may be expressed as the weighted combination

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} \quad (2.46)$$

4) Type 4

In this type of fuzzy reasoning, the ith rule is the form of

$$\text{rule } i\text{: if } (x \text{ is } A_i, \ldots, y \text{ is } B_i) \text{ then } z = f_i(x, \ldots, y) \quad (2.47)$$

where $x, \ldots, y$ and z are linguistic variables representing process state variables. $A_i, \ldots, B_i$ are linguistic terms of the linguistic variables $x, \ldots, y$ in the universe of discourse $U, \ldots, V$, respectively, with $i = 1, 2, \ldots, n$; and f_i is a function of the variables $x, \ldots, y$ defined in the input subspaces. For simplicity, assume that we have two fuzzy control rules as follows:

$$\begin{aligned} &\text{rule 1: if } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } z = f_1(x, y), \\ &\text{rule 2: if } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } z = f_2(x, y). \end{aligned} \quad (2.48)$$

The inferred values from the rules are $\alpha_1 f_1(x_0, y_0)$ and $\alpha_2 f_2(x_0, y_0)$. Correspondingly, a crisp action is given by (2.49)

$$z_0 = \frac{\alpha_1 f_1(x_0, y_0) + \alpha_2 f_2(x_0, y_0)}{\alpha_1 + \alpha_2} \tag{2.50}$$

2.9.5 The Defuzzification Interface

Basically, defuzzification maps output fuzzy sets defined over an output universe of discourse to crisp outputs. It is employed because in many practical applications a crisp output is required. A defuzzification strategy is aimed at producing the nonfuzzy output that best represents the possibility distribution of an inferred fuzzy output. At present, the commonly used strategies may be described as the following:

1) The Max Criterion Method

The max criterion method produces the point at which the possibility distribution of the fuzzy output reaches a maximum value.

2) The Mean of Maximum Method

The mean of maximum generates an output which represents the mean value of all local inferred fuzzy outputs whose membership functions reach the maximum. In the case of a discrete universe, the inferred fuzzy output may be expressed as

$$z_0 = \sum_{j=1}^{l} \frac{w_j}{l} \tag{2.51}$$

where w_j is the support value at which the membership function reaches the maximum value $\mu_z(w_j)$, and l is the number of such support values.

3) The Centre of Area Method

The centre of area generates the centre of gravity of the possibility distribution of the inferred fuzzy output. In the case of a discrete universe, this method yields

$$z_0 = \frac{\sum_{j=1}^{n} \mu_z(w_j)\, w_j}{\sum_{j=1}^{n} \mu_z(w_j)} \tag{2.52}$$

where n is the number of quantization levels of the output.

Figure 2.7 shows a graphical interpretation of various defuzzification strategies.

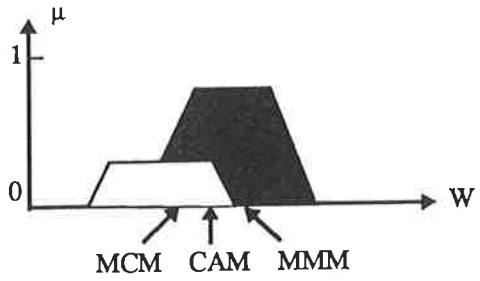


Figure 2.7: Diagrammatic representation of various defuzzification strategies.

2.10 Summary

In this chapter, some basic concepts of fuzzy set theory, which are required for understanding the contents of this thesis, were summarized. These concepts will be employed to build the proposed model of the generic fuzzy neuron and also the architecture of the fuzzy neural network. The basic operations with fuzzy sets, fuzzy relations, fuzzy relational equations, linguistic variables, and fuzzy approximate reasoning are among the topics which will be utilized in the following chapters.

Chapter 3

FUNDAMENTALS OF ARTIFICIAL NEURAL NETWORKS

3.1 Introduction

The biologically inspired artificial neural network models attempt to achieve good performance via dense interconnection of simple computational elements. They have a good potential in areas where many hypotheses are pursued in parallel and high computation rates are required. The potential benefits of neural networks extend beyond the high computation rates provided by massive parallelism. They typically provide a greater degree of robustness or fault tolerance than sequential computers because there are many more processing nodes, each with primarily local connections. Moreover, they can modify their behaviour in response to their environment, i.e., they can learn from environment.

This chapter provides an introduction to the field of neural networks and summarizes the theoretical results concerning multilayer feedforward neural networks. It is organised as follows: First a review of the biological neural networks is presented. Then the artificial neuron model is explained. Next, the architecture of the multilayer feedforward neural network is described. Finally, the backpropagation learning rule is discussed. The contents of this chapter are based on the following publications: [4][8][13][16][36][63].

3.2 The Biological Prototype

Artificial neural networks have undoubtedly been biologically inspired, but the close correspondence between them and real neural systems is still rather weak. Vast discrepancies

exist between both the architectures and capabilities of artificial and natural neural networks. Knowledge about actual brain functions is very limited, so there is little to guide those who would try to emulate them.

The human brain is only a metaphor for a wide variety of neural network configurations that have been developed. It consists of approximately 10^{11} computing elements called *neurons*. Although these neurons can be classified into perhaps as many as 10000 different types, they share many common features. These neurons with basically similar properties are able to produce very different actions because of precise connections with each other and with sensory receptors and muscles. Each neuron is connected to about 10^4 other neurons. A typical neuron has four morphologically defined regions: the cell body, which is also called soma, dendrites, axon, and synaptic terminals. The *cell body* is the metabolic centre of the neuron. It usually gives rise to two types of processes called the *dendrites* and the *axon*. Dendrites form a dendritic tree, which is a very fine bush of thin fibres around the neuron's body. Dendrites receive information from neurons through axons - long fibres that serve as transmission lines. An axon is a long cylindrical connection that carries impulses from the neuron. The end part of an axon splits into a fine arborization. Each branch of the axon terminates in a small endbulb almost touching the dendrites of neighbouring neurons. The axon-dendrite contact organ is called *synapse*. The synapse is where the neuron introduces its signal to the neighbouring neuron. The signals reaching a synapse and received by dendrites are in the form of electrical impulses. The interneural transmission is sometimes electrical but is usually effected by the release of chemical transmitters at the synapse.

The neuron is able to respond to the total of its inputs aggregated within a short time interval called the *period of latent summation*. If the total potential of the neuron's membrane reaches a certain level, it fires. The membrane can be considered as a shell, which aggregates the magnitude of the incoming signals over some duration.

Incoming impulses can be *excitatory* if they cause the firing, or *inhibitory* if they prevent the firing of the response. A more precise condition for firing is that the excitation should exceed the inhibition by the amount called the *threshold* of the neuron. Since a synaptic connection causes the excitation or inhibition of the receiving neuron, it is practical to assign positive and negative unity weight values to such connections respectively. The neuron fires when the total

input impulses exceeds the threshold value during the latent summation period.

After carrying a pulse, an axon is in a state of complete non-excitability for a certain time called *refractory period.* For this time interval the nerve does not conduct any signals. Therefore, we may divide the time scale into consecutive intervals, each equal to the length of the refractory period. This enables a discrete-time description of the neuron's performance in terms of their states at discrete time instance. The time units for modelling biological neurons can be taken to be of the order of a millisecond.

3.3 Learning in Biological Systems

Learning is acquired when modifications are made to the effective coupling between one cell and another, at the synaptic junction. There is no direct linkage across the junction; rather, it is a temporary chemical linkage. The synapse releases chemical materials called *neurotransmitters* when its potential is raised sufficiently by the action potential. It may take the arrival of more than one action potential before the synapse is triggered. The neurotransmitters that are released by the synapse diffuse across the gap, and chemically activate gates on the dendrites, which, when open, allow charged ions to flow. It is this flow of ions that alters the dendric potential, and provides a voltage pulse on the dendrite, which is the conducted along into the next neuron. At the synaptic junction, the number of gates opened on the dendrite depends on the number of neurotransmitters released. This adjustment of coupling so as to favourably reinforce good connections is an important feature of neural network models.

3.4 The Artificial Neuron Model

The artificial neuron model designed to mimic the characteristics of the biological neuron is shown in Figure 3.1. Each neuron model consists of a processing element with synaptic input connections and a simple output. The neuron fires an output response when the aggregate activity of all inputs exceeds some predefined threshold level. From a structural perspective, as shown in Figure 3.1, the response activity y_j of a single neuron at location j can be expressed as

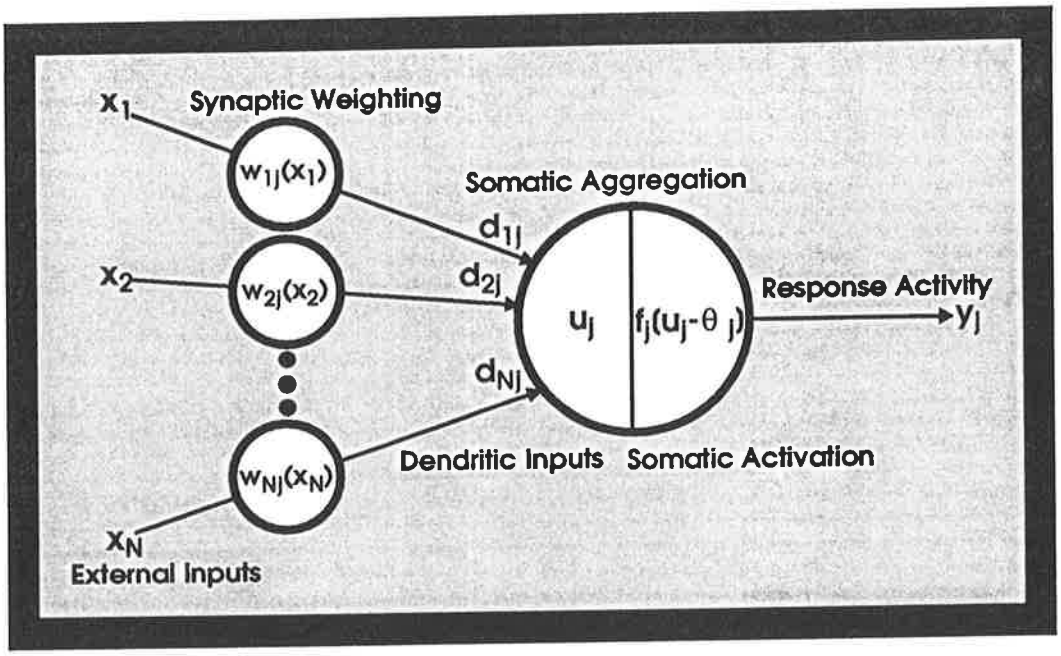


Figure 3.1: The basic structure of the artificial neuron.

$$y_j = f_j[u_j - \theta_j] \tag{3.1}$$

where the function $f_j[.]$ is often referred to as a *somatic activation function* that describes the degree to which the jth neuron is active or firing an output response. Its domain is the set of activation values of the neuron model defined as

$$net_j = u_j - \theta_j \tag{3.2}$$

where u_j is the total *aggregate input activity* incident on the cell body of the neuron and θ_j is the *threshold* level for this neuron. This aggregate input activity may be expressed as

$$u_j = \mathop{\mathrm{A}}_{i=1}^{N} d_{ij} \tag{3.3}$$

where A is some aggregation operator and N is the number of dendritic inputs to the neuron j. Each dendritic input d_{ij} to the neural cell is a transformed version of an external input x_i. This transformation is the result of a weighting function $w_{ij}(.)$ used to describe the *synaptic terminal* between the axon of a transmitting neuron and the dendrite of the receiving neuron. In terms of the synapse, x_i is the synaptic input and d_{ij} is the corresponding synaptic output.

The synaptic output, or dendritic input to the jth neural cell from the ith neuron, can be written as

$$d_{ij} = w_{ij}(x_i) \tag{3.4}$$

The general neuron model, shown in Figure 3.1 and described with expressions (3.1)-(3.4) is commonly used in the neural network literature. However, different artificial neural network classes make use of different definitions of $f(net)$. Also, even within the same class of networks, the neurons are sometimes considered to perform differently during different phases of network operation. Observe from (3.1) that the neuron as a processing node performs the operation of aggregation of its weighted inputs to obtain net. Subsequently, it performs the non-linear operation $f(net)$ through its activation function. Typical activation functions are the hyperbolic tangent

$$f(net) = \frac{1-\exp(-\lambda net)}{1+\exp(-\lambda net)} = \frac{2}{1+\exp(-\lambda net)} - 1 \tag{3.5}$$

and sign function

$$f(net) = \operatorname{sgn}(net) = \begin{cases} +1, & (net \geq 0) \\ -1, & (net < 0) \end{cases} \tag{3.6}$$

where $\lambda > 0$ determines the steepness of the continuous function $f(net)$ near $net = 0$. As $\lambda \to \infty$, the limit of the continuous function becomes the $\operatorname{sgn}(net)$ function defined in (3.6). Activation functions (3.5) and (3.6) are called *bipolar continuous and bipolar binary functions*, respectively.

By shifting and scaling the bipolar activation functions defined by (3.5) and (3.6), *unipolar continuous and unipolar binary activation functions* can be obtained, respectively, as

$$f(net) = \frac{1}{1+\exp(-\lambda net)} \tag{3.7}$$

and

$$f(net) = \operatorname{sgn}(net) = \begin{cases} +1, & (net \geq 0) \\ 0, & (net < 0) \end{cases} \tag{3.8}$$

The unipolar binary function is the limit of $f(net)$ in (3.7) when $\lambda \to \infty$. The *soft-limiting activation functions* (3.5) and (3.7) are often called *sigmoidal activation function*, as opposed to the *hard-limiting activation functions* given in (3.6) and (3.8).

One of the simplest artificial neuron models is the Perceptron. In terms of Equation (3.1) this model can be expressed as

$$y_j = \text{sgn}\left[\sum_{i=1}^{N} w_{ij}x_i - \theta_j\right] \tag{3.9}$$

where the activation function is a hard-limiting function, the aggregation operator is the summation of weighted inputs, and the threshold θ_j is a bias value. In this model the synaptic outputs, $w_{ij}x_i$, are assumed to be linearly proportional to the external input. The corresponding synaptic weights, w_{ij}, may be either positive (excitatory) or negative (inhibitory) real numbers. Both the synaptic weights and the threshold level are assigned to the neuron during the learning procedure. For this simplistic model all external inputs and the resultant output response are assumed to be bipolar binary.

3.5 Artificial Neural networks

The artificial neural network can be defined as an interconnection of neurons, such that neuron outputs are connected, through weights, to all other neurons including themselves. The network models are partitioned into two basic categories: static networks and dynamic networks. *Static networks* are characterized by node equations that are memoryless, that is, their output is a function only of the current input, not past or future inputs or outputs. *Dynamic networks*, on the other hand, are systems with memory. Their node equations are typically described by differential or difference equations. In the following, the multilayer feedforward network which is a static network will be only discussed because it is the type of neural network used to realize the fuzzy neural systems will be described in the next chapter and also our proposed fuzzy neural system.

3.6 Multilayer Feedforward Neural Networks

The multilayer feedforward neural network is a static network consists of an input layer, an output layer, and one or more layers of nodes between the input layer and the output layers. Layers with outputs not visible to the external observer are called *hidden layers* and their units are called *hidden units*. Often the nodes between adjacent layers are fully connected, i.e.,

every node in layer l is connected to every node in layer $l+1$. An L-layer feedforward network with n inputs, m outputs and $L-2$ layers of hidden units is shown in Figure 3.2.

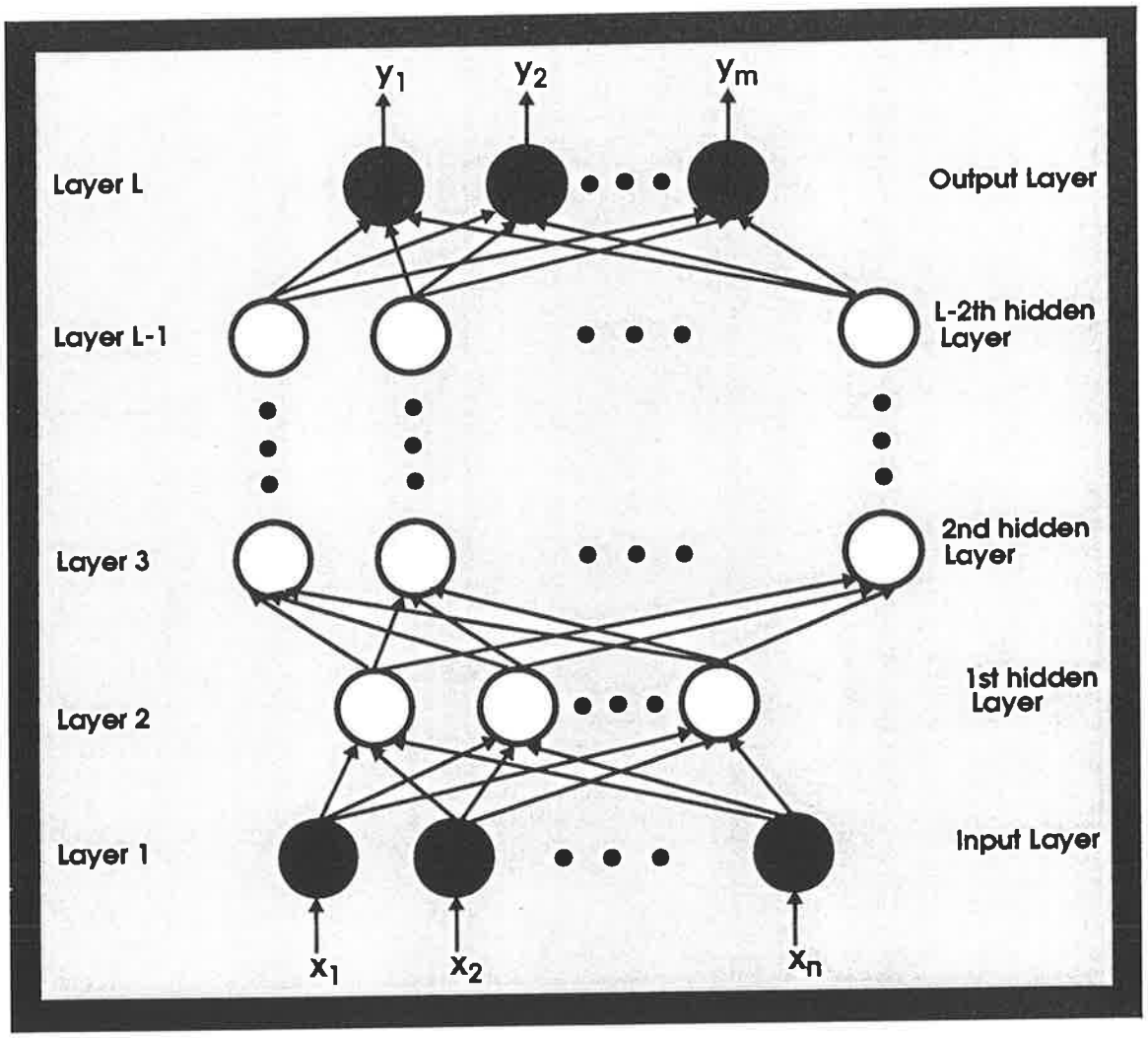


Figure 3.2: The multilayer feedforward neural network topology.

The input layer neurons simply distribute the signal along multiple paths to hidden layer neurons. A weight is associated with a connection between layers. All connections and data flow in the network go from the input to the output. This is why this network is called feedforward network. There are no feedback loops, even from a unit to itself, in a feedforward network. The way that total input to a neuron is calculated and the way that a neuron calculates its output as a function of its inputs depends on the types of neurons being used in the network. In the next section, a learning rule for the multilayer feedforward neural network is presented.

3.7 The Generalised Delta Learning Rule

The learning rule for the multilayer feedforward neural network is called the *generalised delta rule* or *backpropagation rule*. It provides the method for adjusting the weights in the network. When we show the untrained network an input pattern, it will produce a random output. We need to define an error function that represents the difference between the network's current output and the correct output that we want it to produce. Because we need to know the "correct" output, this type of learning is known as *supervised learning*. In order to learn successfully we want to make the output of the network approach the desired output, that is, we want to continually reduce the value of this error function. This is achieved by adjusting the weights on the links between the units. The generalised delta rule does this by calculating the value of the error function for that particular input, and then back-propagating the error from one layer to the previous one. Each unit in the network has its weights adjusted so that it reduces the value of the error function; for units on the output layer, their output and the desired output are known, so adjusting the weights is fairly simple, but for units in the middle layers, the adjusting is not so obvious. The mathematics show that the weights for a particular node should be adjusted in direct proportion to the error in the units to which it is connected. In this way the error function is reduced and the network learns.

Before we discuss the learning algorithm, let us introduce the following notations: (notation refers to a set of symbols)

Notation	Meaning
n	number of input nodes
m	number of output nodes
p	training pattern index
x_i	ith component of input vector
t_i	ith component of target vector
y_i	ith component of output vector
$X_p = x_1, x_2, \ldots, x_n$	pth training pattern
$T_p = t_1, t_2, \ldots, t_m$	pth target output
$Y_p = y_1, y_2, \ldots, y_m$	pth actual output
w_{ij}^{l}	weight which connects the ith node in layer l-1 to the jth node in layer l

N_l	number of nodes in layer l
L	number of layers
P	number of training patterns
E_p	error function for pattern p

Let us define the error function to be proportional to the square of the difference between the actual and desired output, for all the patterns to be learnt:

$$E_p = \frac{1}{2}\sum_{j=1}^{m}(t_{pj} - y_{pj})^2 \tag{3.10}$$

The $\frac{1}{2}$ makes the mathematics simpler and brings this specific error function into line with other similar measures. The activation of each unit j, for pattern p, can be written as

$$net_{pj} = \sum_i w_{ij} y_{pi} \tag{3.11}$$

i.e. simply the weighted sum, as in the single-layer perceptron. The output from each unit j is the threshold function f_i acting on the weighted sum:

$$y_{pj} = f_j(net_{pj}) \tag{3.12}$$

We can write

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ij}} \tag{3.13}$$

by the chain rule. Looking at the second term in (3.13), and substituting in (3.11)

$$\frac{\partial net_{pj}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_k w_{kj} y_{pk} = \sum_k \frac{\partial w_{jk}}{\partial w_{ij}} y_{pk} = y_{pi} \tag{3.14}$$

since $\frac{\partial w_{jk}}{\partial w_{ij}} = 0$ except when $k = i$ it equals 1. The change in error can be defined as a function of the change in the *net* inputs to a unit as

$$-\frac{\partial E_p}{\partial net_{pj}} = \delta_{pj} \tag{3.15}$$

and therefore (3.13) becomes

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} y_{pi} \tag{3.16}$$

Decreasing the value of E_p means making the weight changes proportional to $\delta_{pj}y_{pi}$, i.e.

$$\Delta_p w_{ij} = \eta \delta_{pj} y_{pi} \tag{3.17}$$

where η is a positive constant. To be able to decrease E, we need to know what δ_{pj} is for each unit. Using (3.15) and the chain rule, we can write

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial y_{pj}} \frac{\partial y_{pj}}{\partial net_{pj}} \tag{3.18}$$

Consider the second term, and from (3.12)

$$\frac{\partial y_{pj}}{\partial net_{pj}} = f_j'(net_{pj}) \tag{3.19}$$

Now consider the first term in (3.18). From (3.10), we can differentiate E_p with respect to y_{pj}, giving

$$\frac{\partial E_p}{\partial y_{pj}} = -(t_{pj} - y_{pj}) \tag{3.20}$$

Therefore

$$\delta_{pj} = f_j'(net_{pj})\ (t_{pj} - y_{pj}) \tag{3.21}$$

This is useful for the output units, since the target and output are both available, but not for the hidden units, since their targets are not known. Thus, if unit j is not an output unit, we can write

$$\frac{\partial E_p}{\partial y_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial y_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial y_{pj}} \sum_i w_{ik} y_{pi} = -\sum_k \delta_{pk} w_{jk} \tag{3.22}$$

using (3.11) and (3.15), and noticing that the sum drops out since the partial differential is non-zero for only one value, just as in (3.14). Substituting (3.22) in (3.18), we get finally

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{jk} \tag{3.23}$$

This equation represents the change in the error function, with respect to the weights in the network. This provides a method for changing the error function so as to be sure of reducing it. The function is proportional to the errors δ_{pk} in subsequent units, so the error has to be calculated in the output units first and then passed back through the network to the earlier units to allow them to alter their connection weights. It is the passing back of this error value that leads to the networks being referred to as *back-propagation networks*. Equations (3.21)

and (3.23) together define how the multilayer network could be trained.

Using the sigmoid function as the non-linear threshold function makes the implementation of the back-propagation system much easier since it has a simple derivative. Given that the output of a unit, y_{pj} is given by

$$y_{pj} = f(net) = \frac{1}{1 + e^{-k\ net}} \tag{3.24}$$

the derivative with respect to the unit is given by

$$f'(net) = \frac{ke^{-k\ net}}{\left(1 + e^{-k\ net}\right)^2} = k\ f(net)\ (1 - f(net)) = ky_{pj}(1 - y_{pj}) \tag{3.25}$$

The derivative is therefore a simple function of the outputs.

3.8 Summary

In this chapter, an introduction to the field of artificial neural networks was presented. The emphasis was on multilayer feedforward neural network architectures, which form the basis of fuzzy neural systems introduced in this thesis. In the next chapter we will describe four developed fuzzy neural systems which have feedforward multilayer architectures.

The generalised delta learning rule for multilayer feedforward neural networks was also described in this chapter. This was done because most of the researchers who have addressed learning algorithms for their proposed fuzzy neural systems, have employed learning rules which are mostly inspired by the generalised delta rule. In the next chapter we will present some of these approaches. Although we will not present a learning algorithm for the proposed fuzzy neural network in this thesis, we will employ a generalised-delta-rule based algorithm for training our network in future work.

Chapter 4

FUZZY NEURAL SYSTEMS

4.1 Introduction

Fuzzy neural systems result from the fusion of fuzzy set theory and neural networks. Thus, the advantages of both approaches are merged. Research on fuzzy neural systems has been pursued from two different directions. Some researchers have utilized conventional neuron models to develop neural networks which are functionally equivalent to fuzzy inference systems [15][18][26][46]. These types of neural networks are trained using the learning rules which are mostly derived from the backpropagation algorithm. Other researchers, on the other hand, have developed neurons with fuzzy functions and fuzzy computations. They have applied logical equations or if-then rules with either fuzzy or crisp input values to describe fuzzy neurons, and replaced the conventional neurons with these fuzzy neurons in a neural network [8][9][28][33][35][41][42][52][54]. Most of those who have addressed learning algorithms, employed the learning rules which are mostly inspired by the backpropagation algorithm. This thesis deals with the second group of fuzzy neural systems.

In this chapter, a brief survey of four different models of fuzzy neurons, related fuzzy neural networks and their associated learning algorithms are presented. These four models have been found more attractive than others. They belong to the second category of fuzzy neural systems. The Gupta-Knopf fuzzy neuron is the most powerful fuzzy neuron among the four models described in this chapter. The generic fuzzy neuron which will be introduced in the next chapter is inspired by this fuzzy neuron.

4.2 Gupta and Knopf's Fuzzy Neural Network

A mathematical model for an adaptive fuzzy neuron and a fuzzy neural network has been presented by Gupta and Knopf [8]. A brief review of the proposed fuzzy neuron and fuzzy neural network is given here.

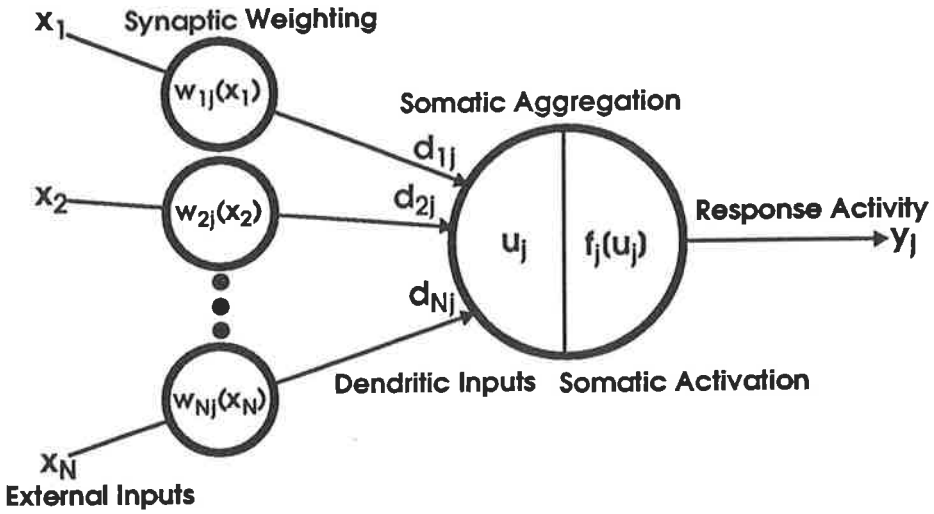


Figure 4.1: The Gupta-Knopf fuzzy neuron.

4.2.1 Fuzzy Neuron

The Gupta-Knopf fuzzy neuron is shown in Figure 4.1. For the fuzzy neuron all states of activity are given in terms of fuzzy sets with relative graded memberships distributed over the interval [0, 1]. These neural states of activity are:

Synaptic Inputs: $x_i = \{\alpha_i, \mu_{x_i}(\alpha_i)\}, \quad \forall i = 1, 2, \ldots, N$ (4.1)

Synaptic Outputs (Somatic Inputs): $d_{ij} = \{\beta_j, \mu_{d_{ij}}(\beta_j)\}, \quad \forall j = 1, 2, \ldots, M$ (4.2)

Aggregated Value of the Somatic Inputs: $u_j = \{\beta_j, \mu_{u_j}(\beta_j)\}, \quad \forall j = 1, 2, \ldots, M$ (4.3)

jth Neuron Output: $y_j = \{\beta_j, \mu_{y_j}(\beta_j)\}, \quad \forall j = 1, 2, \ldots, M$ (4.4)

where N is the total number of dendritic inputs to the jth neuron and M is the total number of neurons in the neural network architecture. In linguistic terms, each fuzzy neuron

represents a single concept such as "small," "big," etc. The output response of the neuron y_j is a fuzzy set representing the degree to which the applied external inputs are able to generate this concept.

The synaptic weighting function which transforms a synaptic input x_i into a synaptic output d_{ij} for the j th neuron is defined by the fuzzy relation between these two fuzzy sets. Therefore, the synaptic weighting function is written as

$$w_{ij} = x_i \times d_{ij} \tag{4.5}$$

where $\times$ is the cartesian product of the fuzzy sets x_i and d_{ij}. The definition for this synaptic weighting function is illustrated in Figure 4.2.

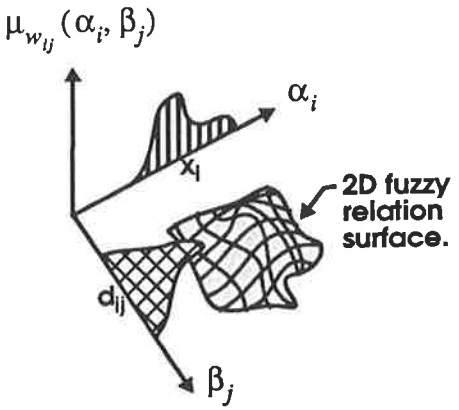


Figure 4.2: The relationship between the synaptic input x_i and the corresponding output d_{ij}.

The synaptic outputs, d_{ij}, may be classified as either excitatory or inhibitory. δ_{ij} is defined as the dendritic input received directly by the soma, where

$$\delta_{ij} = \psi[d_{ij}] = \begin{cases} d_{ij} & \text{if the dendrite transmits an excitatory input} \\ N(d_{ij}) & \text{if the dendrite transmits an inhibitory input} \end{cases} \tag{4.6}$$

The inhibitory inputs undergo a fuzzy logic negation which is defined as

$$N(\mu_{d_{ij}}(\beta_j)) = 1 - \mu_{d_{ij}}(\beta_j) \tag{4.7}$$

The aggregation operator is assumed to be a t-norm operator

$$u_j = \mathop{\mathrm{T}}_{i=1}^{N} \delta_{ij} \tag{4.8}$$

No threshold operation is employed by the fuzzy neuron model. The decision function $f[.]$ is defined as a mapping operator that transforms the membership of the aggregate fuzzy set u_j into the fuzzy set for the neuron response y_j. The role of this modification is to enhance, as shown in Figure 4.3, or diminish the degree to which the external inputs give rise to the concept represented by the neuron j. The modified output becomes an external input to the neighbouring neurons. Thus, a general expression for the response of the jth fuzzy neuron may be written as

$$y_j = f_j[u_j] = f_j\left[\mathop{\mathrm{T}}_{i=1}^{N} \delta_{ij}\right] \tag{4.9}$$

Each dendritic input δ_{ij} is given by

$$\delta_{ij} = \psi[d_{ij}] = \psi[x_i \circ w_{ij}] \tag{4.10}$$

where "$\circ$" is the compositional operator.

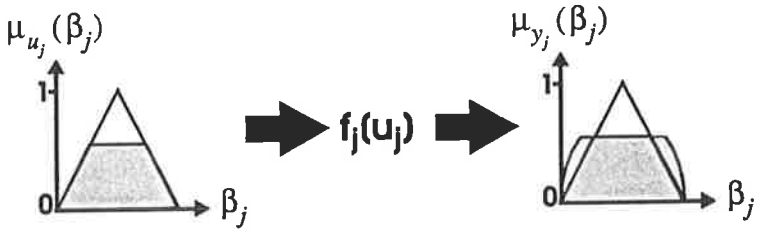


Figure 4.3: The mapping operator $f_j[.]$ that modifies the response of the fuzzy neuron.

4.2.2 The Architecture of the Fuzzy Neural Network

The somatic operation of a fuzzy neuron provides the inference mechanism in this network. The computational structure of a fuzzy neuron represents a single fuzzy inference rule, Equations (4.9) and (4.10), which may be stated linguistically as

$$\text{IF } \delta_{1j} \text{ AND } \delta_{2j} \text{ AND } \ldots \text{ AND } \delta_{Nj} \text{ THEN } y_j \tag{4.11}$$

where each fuzzy input is given by compositional operation at the synapse as described by Equation (4.10).

The inference procedure associated with an unknown input x_i^* arises at the synapse such that the new synaptic output is given by

$$d_{ij}^* = x_i^* \circ w_{ij} \tag{4.12}$$

or in terms of the membership functions Equation (4.12) is rewritten as follows

$$\mu_{d_{ij}}^*(\beta_j) = max\left[min\{\mu_{x_i}^*(\alpha_i), \mu_{w_{ij}}(\alpha_i, \beta_j)\} \right] \tag{4.13}$$

The resultant response of the fuzzy neuron to N inputs is given by Equations (4.9) and (4.10).

A practical fuzzy inference system requires a set of M rules. This inference system may be achieved by three contiguous layers of fuzzy neurons. The first layer consists of M fuzzy neuron, each representing one of the rules in the set. The output of each fuzzy neuron becomes an external input to a single neuron in the second layer. Since the output of each neuron is over the same universe, β, the responses from these M fuzzy neurons must be combined by an s-norm operation. Two cascaded fuzzy neurons with inhibitory inputs correspond to an s-norm operation for the fuzzy inputs $(y_1, y_2, \ldots, y_N)$. This three-layer neural network architecture can be used to derive an output to N fuzzy inputs that are applied to M fuzzy inference rules, as illustrated in Figure 4.4.

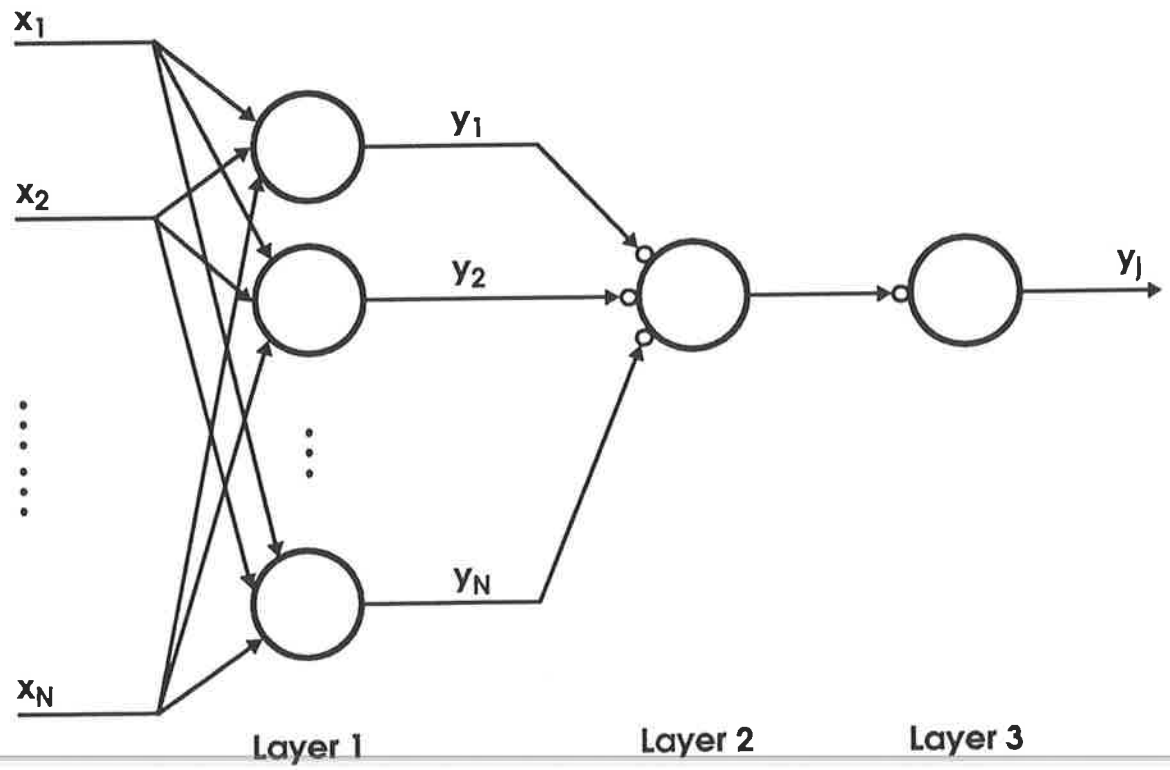


Figure 4.4: The architecture of the Gupta-Knopf fuzzy neural network.

4.2.3 Learning by Experience

The learning procedure for a fuzzy neuron involves changing the surface of the two-dimensional fuzzy relation employed at each synapse. A single synaptic connection to the j th neuron is shown in Figure 4.5. For a given external input to this synapse at time k, $x_i(k)$, the corresponding fuzzy relation, $w_{ij}(k)$, should be determined. This gives a minimum error, $e_j(k)$, between the fuzzy neuron response, $y_j(k)$, and the desired response, $D_j(k)$. In order to achieve this property, the following adaptation rule may be employed to modify the fuzzy relation surface

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k) \tag{4.14}$$

The term $\Delta w_{ij}(k)$ is the change in fuzzy relation surface given as a function, $F[.]$, of the error

$$\Delta w_{ij}(k) = F[e_j(k)] = F[D_j(k) - y_j(k)] \tag{4.15}$$

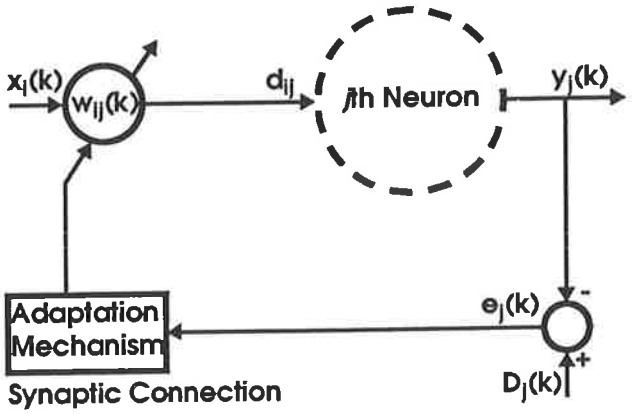


Figure 4.5: The synaptic diagram of a single synaptic connection with learning.

4.3 Fuzzy-Set Based Methods of Neurons and Knowledge-Based Networks

Pedrycz and Rocha introduced the models of neurons based on logic-oriented processing mechanisms of fuzzy sets [42]. Different classes of logic-driven neurons, their properties, learning mechanisms for the discussed types of neurons, and development of neural networks are described below.

4.3.1 Aggregative Neurons: OR and AND Logical Computing Nodes

Two basic types of logical neurons performing aggregation of the OR and AND types are discussed in this section. Their key feature is that all processing faculties realized by these neurons are completed with the use of standard fuzzy set operations. All input and output signals as well as the interconnections of the network are assumed to be coded in the unit interval. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a finite set of input nodes and let $Y = \{y_1, y_2, \ldots, y_n\}$ denote a set of output nodes.

a) OR computing node: Its input signals $x_1, x_2, \ldots, x_n$ are combined successively with weights $w_1, w_2, \ldots, w_n$ by employing first the AND logical connective to each pair (x_i, w_i). Subsequently, these partial results are aggregated by means of the OR logical operator producing in this way the output y of the neuron. This transformation is written as

$$y = OR(X;W) \tag{4.16}$$

The explicit formula is based on the t-s composition of the fuzzy sets X and W:

$$y = \underset{i=1}{\overset{n}{S}} \left(w_i t x_i \right) \tag{4.17}$$

A bias term is added as an additional term in (4.17) that is driven by a constant input signal equal to 1. Thus, the neuron incorporating the bias is given by:

$$y = \underset{i=0}{\overset{n}{S}} \left(w_i t x_i \right) \tag{4.18}$$

where, w_0 stands for the connection of the bias term and x_0 is a constant input equal to 1. The role of the bias is to shift the output values of the neuron.

In particular, for the max and min operations the output of the OR neuron is given by

$$y = \max_i \{ min(w_i, x_i) \} \tag{4.19}$$

b) AND computing node: The AND computing node constitutes a structural dual to that given previously. The input signals are first combined by completing the OR operation over a collection of the connections of the neuron. Then, the AND node combines the results by AND-ing them:

$$y = AND(X;W) \tag{4.20}$$

$$y = \underset{i=0}{\overset{n}{T}}\left(w_i S x_i\right) \tag{4.21}$$

where, w_0 stands for the connection of the bias term, x_0, which is now kept constant at zero value. In particular, for the *min* and *max* operations:

$$y = \min_i \{max(w_i, x_i)\} \tag{4.22}$$

To add an inhibitory effect to the construct, Pedrycz and Rocha extended the inputs by including complementary values of x_i, say $1 - x_i$: $X = [x_1\ x_2\ \ldots\ x_n \mid \bar{x}_1\ \bar{x}_2\ \ldots\ \bar{x}_n]$. The OR and AND neurons with this extended vector of inputs make it possible to admit both the inhibitory and excitatory characters of their behaviour, depending on the numerical values of the connections. Figure 4.6 illustrates the graphical notation of OR and AND neurons. In this figure, $x_1, x_2, \ldots, x_n$ denote the excitatory inputs and $w_1, w_2, \ldots, w_n$ denote their weights. $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ are the input complements employed to handle inhibitory features of the neurons, and $v_1, v_2, \ldots, v_n$ are their weights.

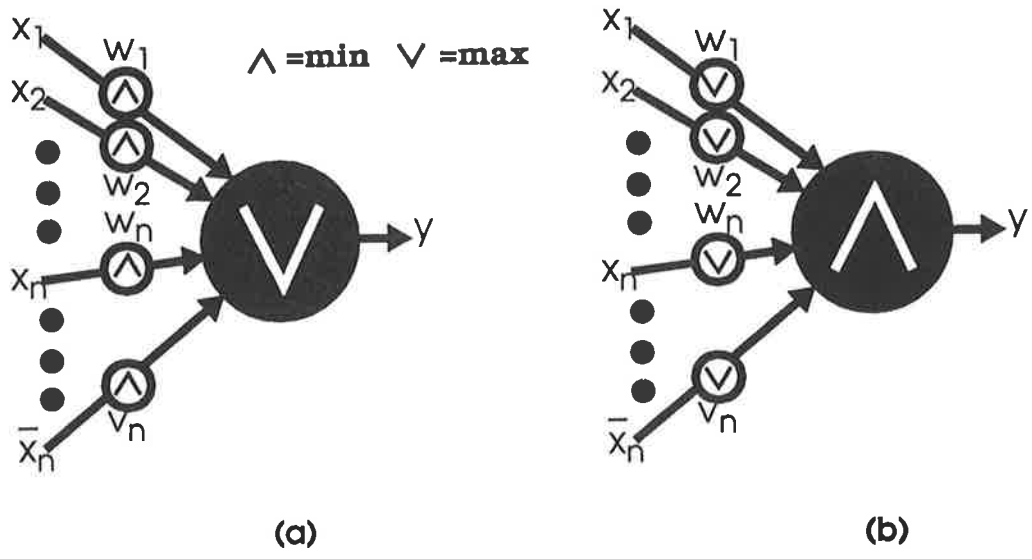


Figure 4.6: Graphical notation of (a) OR and (b) AND neurons.

4.3.2 Reference Neurons

The role of a reference neuron is to perform a transformation of the input signals with respect to a given reference point. Let the reference point be denoted by $R = \{r_1, r_2, \ldots, r_n\}$. The transformation is denoted by

$$y = \mathcal{F}(X;W,R) \tag{4.23}$$

We describe here the following two reference neurons:

a) Matching neuron:

$$y = MATCH(X;W,R) \tag{4.24}$$

$$y = \sum_{i=1}^{n} \left[w_i t (x_i \equiv r_i) \right] \tag{4.25}$$

where the equality index, denoted by $\equiv$, can be defined as

$$a \equiv b = \frac{1}{2} \{ (a \rightarrow b) \wedge (b \rightarrow a) + (\bar{a} \rightarrow \bar{b}) \wedge (\bar{b} \rightarrow \bar{a}) \} \tag{4.26}$$

that returns a level of matching for any two degrees of membership $a, b \in [0, 1]$. In this formula, the $\wedge$ denotes the *min* operator, and $\rightarrow$ is modelled by the φ operation as follows:

$$a \varphi b = min\left(1, \frac{a}{b}\right) \tag{4.27}$$

When $w_i = 1$, the matching along this coordinate is essential; lower values of w_i indicate that the result of matching derived becomes less important.

b) Difference neuron:

$$y = DIFFER(X;W,R) \tag{4.28}$$

$$y = \sum_{i=1}^{n} \left[w_i t (x_i \equiv | r_i) \right] \tag{4.29}$$

where the difference operator $\equiv |$ is taken as complement of the equality index

$$(a \equiv | b) = 1 - (a \equiv b) \tag{4.30}$$

4.3.3 Learning Procedure for Basic Logic Neurons

The issue of learning for the discussed types of neurons is addressed as a problem of supervised training controlled by the gradient of the predefined performance index. The generic performance index used here is the mean squared error between the output of the neuron and a target value. For the series of training data organised as input/output pairs,

$$(X_1, t_1)$$
$$(X_2, t_2)$$
$$\vdots$$
$$(X_N, t_N)$$

Within the process, the adjustable parameters of the neuron (w_i, v_i, and/or its reference value r_i) are modified in order to minimize the given performance index. The optimization is done by minimizing the performance index expressed as follows:

$$E = \sum_{k=1}^{N} (t_k - y_k)^2 \tag{4.31}$$

where y_k stands for the actual output of the neuron. The standard iterative scheme of adjustments of the connections, say w_i, is read as:

$$w_i = w_i - \alpha \frac{\partial E}{\partial w_i} \qquad \text{i= 1,2,..., n} \tag{4.32}$$

The learning rate α controls successive increments of the connections.

4.3.4 Multilevel Neural Networks

The networks which are composed of logic-based neurons are called *heterogeneous networks*. This means that they include several neurons of different computational characteristics. The neurons are organized into layers. A three layer neural structure is discussed here. It consists of OR and AND neurons described previously. Each layer consists of neurons of the same logical type. Two types of the networks are defined (Figure 4.7). The first class is composed of AND neurons situated in the hidden layer, while the output layer consists of a single OR neuron. The second category has OR neurons in the hidden layer and a single AND neuron in the output layer.

For the first class of the network (OR-AND):

- an input layer consists of $2n$ nodes including both the direct and the complementary versions of the inputs. The role of these nodes is to distribute the signals to all the nodes of the hidden layer.

- a hidden layer is composed of h AND neurons. The intermediate signals z_l produced there, are described as:

$$z_l = AND(W_l, X) \quad l = 1, 2, \ldots, h \tag{4.33}$$

The vector of connections, W_l, summarises all the connections between the lth node of the hidden layer and the input nodes. In other words, this relationship is obtained as follows:

$$z_l = \mathop{T}_{i=1}^{n}\left(w_{li} S x_i\right) t \mathop{T}_{i=1}^{n}\left(w_{l(n+i)} S \bar{x}_i\right) \qquad l=1,2,\ldots, h \tag{4.34}$$

The output layer contains a single OR neuron which performs an aggregation of z_l's:

$$y = \mathop{S}_{i=1}^{h}\left(v_l t z_l\right) \tag{4.35}$$

where v_l's denote the weights. The first type of the network implements any two-valued function as a sum of minterms (SOM). The second architecture develops a product of maxterms (POM).

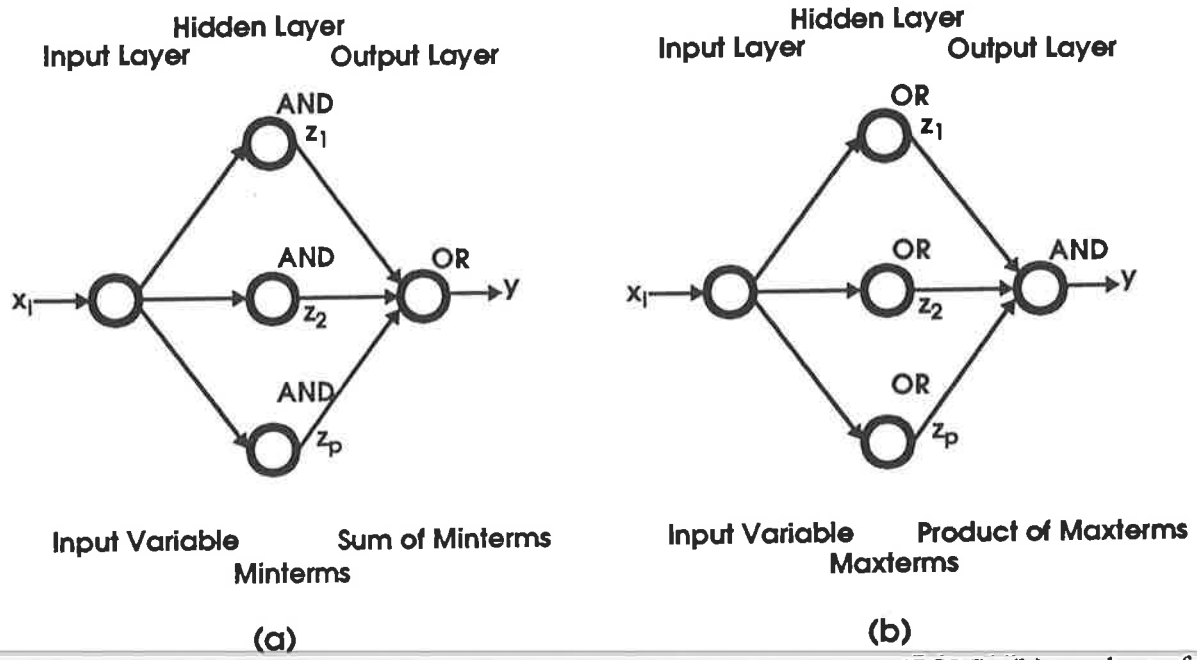


Figure 4.7: Two types of logic-based networks (a) sum of minterms (SOM), (b) product of maxterms (POM).

4.4 Kwan and Cai's Fuzzy Neural Network

Kwan and Cai have defined a fuzzy neuron and proposed the structure of a four-layer feedforward fuzzy neural network and its associated learning algorithm [28]. A brief review of the proposed neuron, the fuzzy neural network and its learning algorithm are discussed here.

4.4.1 Fuzzy Neuron

The Kwan-Cai fuzzy neuron (as shown in Figure 4.8) has N weighted inputs, x_i for $i = 1, 2, \ldots, N$, with w_i $(i = 1, 2, \ldots, N)$ weights, and M outputs, y_j for $j = 1, 2, \ldots, M$. All the inputs and weights are real values in interval $[0, 1]$. Each output could be associated with the membership value of a fuzzy concept. Moreover, we have:

$$z = h[w_1x_1, w_2x_2, \ldots, w_Nx_N] \tag{4.36}$$

$$s = f[z - T] \tag{4.37}$$

$$y_j = g_j[s] \quad \text{for } j = 1, 2, \ldots, M \tag{4.38}$$

where z is the *net* input of the fuzzy neuron; $h[.]$ is the aggregation function; s is the state of the fuzzy neuron; $f[.]$ is the activation function; T is the activating threshold; and $\{g[.], j = 1, 2, \ldots, M\}$ are the M output functions of the fuzzy neuron which represent the membership functions of the input pattern $\{x_1, x_2, \ldots, x_N\}$ in all the M fuzzy sets. Four types of fuzzy neurons (FNs) are defined by changing the neuron's functions:

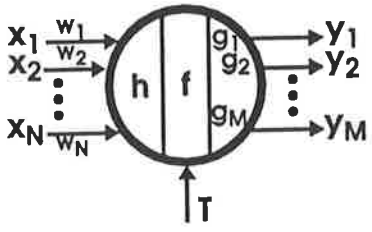


Figure 4.8: The Kwan-Cai fuzzy neuron.

a) Input-FN:

If a fuzzy neuron is used in the input layer of a fuzzy neural network and it has only one input x, such that

$$z = x \tag{4.39}$$

then it is called an Input-FN.

b) Maximum-FN:

If a maximum function is used as the aggregation function of a fuzzy neuron such that

$$z = \max_{i=1}^{N} (w_i x_i) \tag{4.40}$$

then it is called a Max-FN.

c) Minimum-FN:

If a minimum function is used as the aggregation function of a fuzzy neuron such that

$$z = \min_{i=1}^{N} (w_i x_i) \tag{4.41}$$

then it is called a Min-FN.

d) Competitive-FN:

A competitive fuzzy neuron (Comp-FN) has a variable threshold T and only one output such that

$$y = g[s-T] = \begin{cases} 0 & \text{if } s < T \\ 1 & \text{if } s \geq T \end{cases} \tag{4.42}$$

$$T = t[c_1, c_2, \ldots, c_K] \tag{4.43}$$

where s is the state of the fuzzy neuron; $t[.]$ is the threshold function; and c_k (k=1 to K) are the competitive variables of the fuzzy neuron.

4.4.2 Structure of the Fuzzy Neural Network

The four-layer feedforward fuzzy neural network proposed by Kwan and Cai is shown in Figure 4.9. The first layer, which is arranged in a two-dimensional array, has $N_1 \times N_2$ Input-FNs. The formulation of the (i, j)th Input-FN is as follows

$$s_{ij}^{[1]} = z_{ij}^{[1]} = x_{ij} \quad \text{for i} = 1, \ldots, N_1, \text{ j} = 1, \ldots, N_2 \tag{4.44}$$

$$y_{ij}^{[1]} = \frac{s_{ij}^{[1]}}{P_{vmax}} \quad \text{for i} = 1, \ldots, N_1, \text{ j} = 1, \ldots, N_2 \tag{4.45}$$

where x_{ij} is the (i, j)th input value ($x_{ij} \geq 0$) and P_{vmax} is the maximum input value among all inputs.

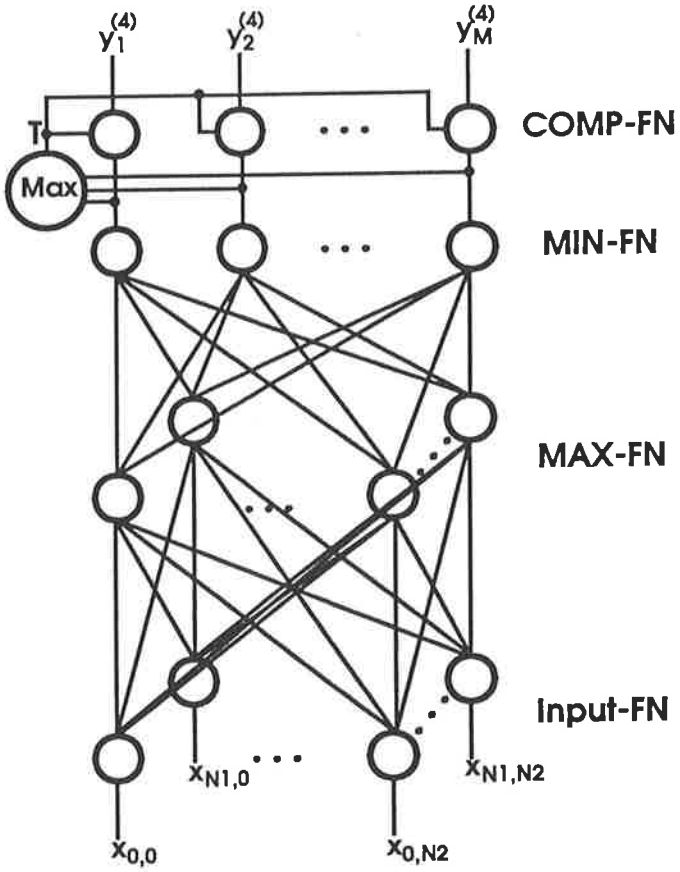


Figure 4.9: The Kwan-Cai fuzzy neural network.

The second layer is also arranged in a two-dimensional array, and consists of $N_1 \times N_2$ Max-FNs. The purpose of this layer is to fuzzify the input through a weight function $w[m, n]$. The state of the (p, q)th Max-FN in this layer is:

$$s_{pq}^{[2]} = \max_{i=1}^{N_1}\left(\max_{j=1}^{N_2}\left(w[p-i, q-j]\, y_{ij}^{[1]}\right)\right) \qquad (4.46)$$

$$\text{for } p = 1, \ldots, N_1,\ q = 1, \ldots, N_2$$

where $w[p-i, q-j]$ is the weight connecting the (i, j)th Input-FN in the first layer to the (p, q)th Max-FN in the second layer and is defined by

$$w[m, n] = \exp\left(-\beta^2\left(m^2 + n^2\right)\right) \qquad (4.47)$$
$$\text{for } m = -(N_1 - 1) \text{ to } (N_1 - 1),\ n = -(N_2 - 1) \text{ to } (N_2 - 1)$$

The value of β is determined by the learning algorithm. Each Max-FN has M different

outputs, one for each fuzzy neuron in the third layer. The outputs of the (p, q)th Max-FN in this layer are:

$$y_{pqm}^{[2]} = g_{pqm}\left[s_{pq}^{[2]}\right] \quad \text{for } p = 1, \ldots, N_1, q = 1, \ldots, N_2, m = 1, \ldots M \tag{4.48}$$

where $y_{pqm}^{[2]}$ is the mth output of the (p, q)th Max-FN which is to be connected to the mth Min-FN in the third layer. The output function $g_{pqm}\left[s_{pq}^{[2]}\right]$ is to be determined by the learning algorithm. For simplicity, the triangles with heights equal to 1 and base length equal to α (shown in Figure 4.10) are chosen as the output functions of the Max-FNs in the second layer. Hence

$$y_{pqm}^{[2]} = g_{pqm}\left[s_{pq}^{[2]}\right] = \begin{cases} 1 - 2\left|s_{pq}^{[2]} - \Theta_{pqm}\right|/\alpha & \text{if } \alpha/2 \geq \left|s_{pq}^{[2]} - \Theta_{pqm}\right| \geq 0 \\ 0 & \text{if otherwise} \end{cases} \tag{4.49}$$

$$\text{for } \alpha \geq 0, p = 1, \ldots, N_1, q = 1, \ldots, N_2, m = 1, \ldots, M$$

where Θ_{pqm} is the central point of the base of function $g_{pqm}\left[s_{pq}^{[2]}\right]$. The corresponding α and Θ_{pqm} for every set of p, q and m must be determined by the learning algorithm.

In the third layer which is one-dimensional, the output of the mth Min-FN defined as

$$y_m^{[3]} = s_m^{[3]} = \min_{p=1}^{N_1}\left(\min_{q=1}^{N_2}\left(y_{pqm}^{[2]}\right)\right) \quad \text{for } m = 1 \text{ to } M \tag{4.50}$$

where $s_m^{[3]}$ represents the state of the mth Min-FN in the third layer.

In the output layer, M Comp-FNs are used to provide nonfuzzy outputs. The algorithm of the mth Comp-FN is:

$$s_m^{[4]} = z_m^{[4]} = y_m^{[3]} \quad \text{for } m = 1 \text{ to } M \tag{4.51}$$

$$y_m^{[4]} = g\left[s_m^{[4]} - T\right] = \begin{cases} 0 & \text{if } s_m^{[4]} < T \\ 1 & \text{if } s_m^{[4]} \geq T \end{cases} \quad \text{for } m = 1, \ldots, M \tag{4.52}$$

$$T = \max_{m=1}^{M}\left(y_m^{[3]}\right) \quad \text{for } m = 1 \text{ to } M \tag{4.53}$$

where T is the activation threshold of all the Comp-FNs in the fourth layer.

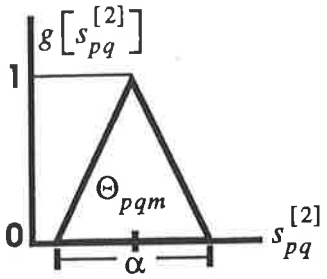


Figure 4.10: Output function of a Max-FN in the second layer.

4.4.3 Learning Algorithm of the Fuzzy Neural Network

The learning algorithm determines the following parameters: The parameters of the output functions of the Max-FNs in the second layer, α and Θ_{pqm}; the parameter of the fuzzification function, β; and the number of fuzzy neurons in each of the third and fourth layers, M. Let T_f be the fault tolerance of the fuzzy neural network ($0 \le T_f \le 1$) and K be the total number of training patterns. The steps of learning algorithm are:

Step 1. Create $N_1 \times N_2$ Input-FNs in the first layer and $N_1 \times N_2$ Max-FNs in the second layer. Choose a value for α ($\alpha \ge 0$) and a value for β.

Step 2. Set $M = 0$ and $k = 1$.

Step 3. Set $M = M + 1$. Create the Mth Min-FN in the third layer and the Mth Comp-FN in the fourth layer. Set:

$$\Theta_{pqM} = s_{pqM}^{[2]} = \max_{i=1}^{N_1}\left(\max_{j=1}^{N_2} (w[p-i, q-j]\, x_{ijk}) \right) \tag{4.54}$$

for $p = 1, \ldots, N_1, q = 1, \ldots, N_2$

where Θ_{pqM} is the central point of the Mth output function of the (p, q)th Max-FN in the second layer, and $X_k = \{x_{ijk}\}$ is the kth training pattern.

Step 4. Set $k = k + 1$. If $k > K$, then the learning procedure is finished. Otherwise, input the kth training pattern to the network and compute the output of the current fuzzy

neural network (with M fuzzy neurons in the third and fourth layers). Set:

$$\sigma = 1 - \max_{j=1}^{M}\left(y_{jk}^{[3]}\right) \tag{4.55}$$

where $y_{jk}^{[3]}$ is the output of the jth Min-FN in the third layer for the kth training pattern X_k. If $\sigma \leq T_f$, go to Step 4. If $\sigma > T_f$, go to Step 3.

4.5 Lin and Song's Fuzzy Neural Network

Lin and Song have proposed a fuzzy neuron and defined a three-layer fuzzy neural network and its associated learning algorithm [35]. A brief review of the proposed neuron, the fuzzy neural network and its learning algorithm is discussed below.

4.5.1 Fuzzy Neuron

The proposed fuzzy neuron is shown in Figure 4.11. The inputs to the fuzzy neuron are fuzzy sets $u_1, u_2, \ldots, u_p$ in the universes of discourse $U_1, U_2, \ldots, U_p$, respectively. These fuzzy sets can be labelled by such linguistic terms as "very small," "small," "large," etc. The inputs are then weighted based on a fuzzy computation called "integration operation," instead of conventional weighted summation. Then, the weighted inputs go through an activation unit. The activation unit performs fuzzy logic computations instead of conventional activation operations such as the sigmoidal function.

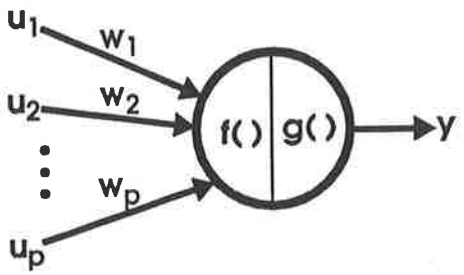


Figure 4.11: The Lin-Song fuzzy neuron.

As shown in Figure 4.11, the fuzzy neuron has an integration unit and an activation unit. The integration function f combines information, activities, or evidences from other neurons and

provides the net input for this fuzzy neuron

$$net = f(u_1, u_2, \ldots, u_p ; w_1, w_2, \ldots, w_p) \tag{4.56}$$

An activation function g output a value as a function of its net input

$$\text{output} = g(net) \tag{4.57}$$

4.5.2 The Fuzzy Neural Network Architecture

The proposed fuzzy neural network has three layers as shown in Figure 4.12. Layer 1 is the input layer, layer 2 is the fuzzy rule base layer, and layer 3 is the output layer. The fuzzy neurons used in each layer have different integration and activation functions.

In the first layer, the fuzzy neurons have the following integration and activation functions:

$$f_i^{(1)} = -\frac{\left(x_i^{(1)} - m_i^{(1)}\right)^2}{\left(s_i^{(1)}\right)^2} \tag{4.58}$$

$$g_i^{(1)} = \exp\left(f_i^{(1)}\right) \quad i = 1, 2, \ldots, (n_1 + n_2 + \ldots + n_N) \tag{4.59}$$

where $m_i^{(1)}$ and $s_i^{(1)}$ are the mean and variance of the i th bell-shaped function of the inputs. Suppose that the dimension of the input variables is N and each input variable is divided into n_j linguistic term levels ($j = 1, 2, \ldots, N$), then the total number of inputs of the first layer is $n_1 + n_2 + \ldots + n_N$. For crisp inputs, each input variable will be fed into n_j input nodes in the first layer. In this layer, the $m_i^{(1)}$'s and $s_i^{(1)}$'s can be interpreted as weights. If some of the input variables are fuzzy linguistic variables, the corresponding input functions will be set to 1.

The fuzzy neurons in the second layer have the following integration and activation functions:

$$f_j^{(2)} = x_{j1}^{(2)} x_{j2}^{(2)} \ldots x_{jN}^{(2)} \tag{4.60}$$

$$g_j^{(2)} = f_j^{(2)} \quad j = 1, 2, \ldots, L \quad L = n_1 \times n_2 \times \ldots \times n_N \tag{4.61}$$

In the third layer, the fuzzy neurons have the following integration and activation functions:

$$f_k^{(3)} = \sum m_{jk}^{(3)} x_j^{(3)} \quad \text{for } k = 1, 2, \ldots, M \tag{4.62}$$

$$g_k^{(3)} = \frac{f_k^{(3)}}{\sum x_j^{(3)}} \quad \text{for } k = 1, 2, \ldots, M \tag{4.63}$$

$$y_k = g_k^{(3)} \quad \text{for } k = 1, 2, \ldots, M \tag{4.64}$$

Similarly, the weights of neurons in this layer can be interpreted as $m_{jk}^{(3)}$'s. The outputs of this layer are all crisp.

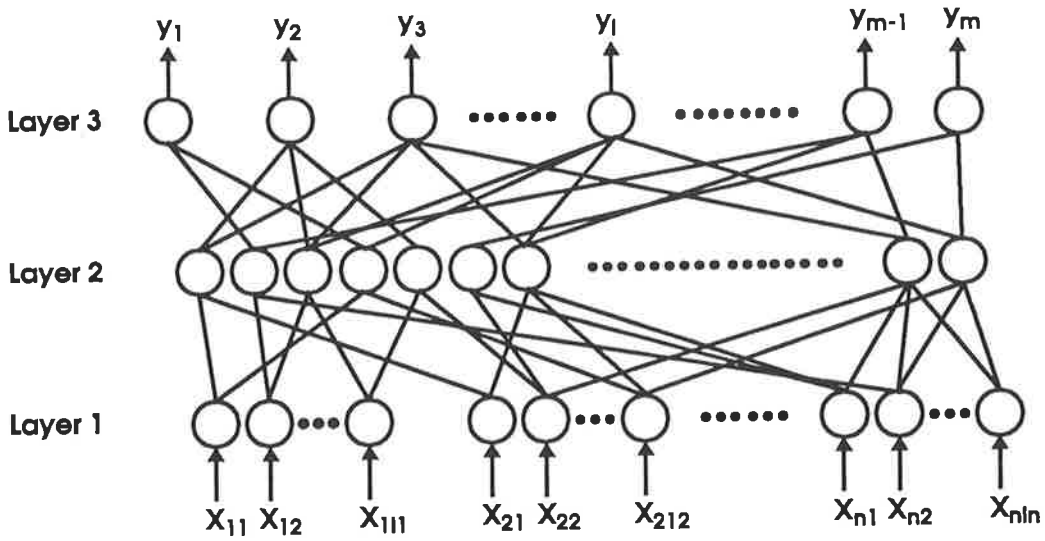


Figure 4.12: The Lin-Song fuzzy neural network architecture.

4.5.3 The Fuzzy Neural Network Learning Algorithm

In order to adjust and optimize the parameters of the fuzzy neural network, the backpropagation algorithm is utilized. The objective is to minimize the output error function:

$$E = \frac{1}{2}\|y_d - y\|^2 = \frac{1}{2}\|y_d - g^{(3)}\|^2 \tag{4.65}$$

where y_d is the desired output and y is the fuzzy neural network output. The general learning algorithm used for adjusting the network parameters w_{ij}'s is

$$w_{ij}(t+1) = w_{ij}(t) + \beta\left(-\frac{\partial E}{\partial w_{ij}}\right) \tag{4.66}$$

where β is the learning rate. In detail, the learning algorithm can be described as follows:

Layer 3:

$$m_{jk}^{(3)}(t+1) = m_{jk}^{(3)}(t) - \beta\left(\frac{\partial E}{\partial m_{jk}^{(3)}}\right) \tag{4.67}$$

$$\frac{\partial E}{\partial m_{jk}^{(3)}} = -\left(y_{dk} - g_k^{(3)}\right)\left(\frac{x_j^{(3)}}{\sum x_i^{(3)}}\right) \tag{4.68}$$

$$j = 1, 2, \ldots, L \quad k = 1, 2, \ldots, M$$

Layer 2:

The weights are all unity and no weight updating is needed. Error propagation through this layer will be treated directly in Layer 1.

Layer 1:

$$m_i^{(1)}(t+1) = m_i^{(1)}(t) - \beta\left(\frac{\partial E}{\partial m_i^{(1)}}\right) \tag{4.69}$$

$$s_i^{(1)}(t+1) = s_i^{(1)}(t) - \beta\left(\frac{\partial E}{\partial s_i^{(1)}}\right) \tag{4.70}$$

$$\frac{\partial E}{\partial m_i^{(1)}} = \sum_k -\left\{\left(y_{dk} - g_k^{(3)}\right)\left(\frac{1}{\sum x_j^{(3)}}\right)\left(\sum_j m_{jk}^{(3)}\left(\frac{\partial f_j^{(2)}}{g_i^{(1)}}\right)\exp\left(f_i^{(1)}\right)2\frac{\left(x_i^{(1)} - m_i^{(1)}\right)}{\left(s_i^{(1)}\right)^2}\right)\right\} \tag{4.71}$$

$$\frac{\partial E}{\partial s_i^{(1)}} = \sum_k -\left\{\left(y_{dk} - g_k^{(3)}\right)\left(\frac{1}{\sum x_j^{(3)}}\right)\left(\sum_j m_{jk}^{(3)}\left(\frac{\partial f_j^{(2)}}{g_i^{(1)}}\right)\exp\left(f_i^{(1)}\right)2\frac{\left(x_i^{(1)} - m_i^{(1)}\right)^2}{\left(s_i^{(1)}\right)^3}\right)\right\} \tag{4.72}$$

$$\frac{\partial f_j^{(2)}}{g_i^{(1)}} = \begin{cases} 0 & \text{if } j_1 \neq i, j_2 \neq i, \ldots, j_N \neq i \\ \prod_{l=1, j_l \neq i}^{N} x_{jl}^{(2)} & \text{otherwise} \end{cases} \tag{4.73}$$

$$i = 1, 2, \ldots, (n_1 + n_2 + \ldots + n_N)$$

4.6 Conclusions

This chapter discussed briefly four different models of fuzzy neural systems including fuzzy neurons, fuzzy neural networks, and their learning algorithms. All of the discussed fuzzy neural systems are categorized into the second class of fuzzy neural systems described before. Among the described models of fuzzy neurons, Gupta-Knopf's fuzzy neuron is the most

powerful one in processing of fuzzy input information, so that the neural network which is built using this fuzzy neuron will have the smallest number of neurons and layers. However, this fuzzy neuron is the most complex neuron. Gupta and Knopf employed their fuzzy neuron to implement a fuzzy neural network for a steering control of an automobile application. The simplest fuzzy neuron was the Pedrycz-Rocha fuzzy neuron. It is capable of implementing some simple fuzzy set operations like AND, OR, or NOT. To achieve a fuzzy decision making system, we should employ a greater number of neurons of this type than Gupta-Knopf neurons, and also use more interconnections between nodes. However, the structure of this fuzzy neuron is very simple and its learning and recall speed are fast. Pedrycz and Rocha applied their fuzzy neuron in decision making, diagnostic, and mappings problems. The other two fuzzy neurons are not as strong as the Gupta-Knopf fuzzy neuron or as simple as the Pedrycz-Rocha one. Kwan and Cai employed their fuzzy neural network in pattern recognition, whereas, Lin and Song used their fuzzy neural system in an inverse kinematics manipulator with two degree of freedom. In the next chapter, a generic model of a fuzzy neuron inspired by Gupta and Knopf's fuzzy neuron, and the architecture of a feedforward multilayer fuzzy neural network for motion detection and velocity estimation will be introduced. Although the generic fuzzy neuron is more complex than all the described fuzzy neurons, it is the most powerful.

Chapter 5

THE PROPOSED FUZZY NEURAL SYSTEM

5.1 Introduction

After the brief discussion of various models of fuzzy neural systems in the previous chapter, the proposed fuzzy neural system is discussed in this chapter. The contents of this chapter is organised as follows: First, we introduce our generic model of a fuzzy neuron as the basic element of our fuzzy neural network. Then, we look at the bases of our approach for motion estimation before we propose our fuzzy neural system. In this relation, we review how fuzziness is measured in a grey-tone image. We also introduce an algorithm for motion detection and velocity estimation. Moreover, we will study how velocity vectors consisting of speed and motion direction are calculated at each pixel in the input image. In the last section of this chapter, we propose the architecture of a five-layer fuzzy neural network which emulates the motion estimation algorithm. Since different types of fuzzy neurons are used in different layers of the network, the definition of the fuzzy neurons, which are simplified versions of the generic fuzzy neuron, will be given together with the operation of the layers.

5.2 The Generic Fuzzy Neuron

In this section, a generic fuzzy neuron is introduced which forms the basic computational element of the fuzzy neural network that will be discussed later. The generic fuzzy neuron model is a generalization of the existing models of fuzzy neurons. Structural similarity between the generic fuzzy neuron and the Gupta-Knopf fuzzy neuron is not just coincidental; the author owes much inspiration to Gupta and Knopf's paper [8]. However, the generic

fuzzy neuron differs from that of Gupta and Knopf and other models in the following ways:

a) All the variables involved in the generic fuzzy neuron are allowed to be fuzzy or crisp sets over different universes of discourse.

b) All the functions that specify the characteristics of the neuron are chosen to be fuzzy relations (when the universes of discourse are similar, the fuzzy relation can be replaced by any fuzzy operation).

c) The output of each function, which is a fuzzy set, is obtained using the *-* composition of inputs to the function and the corresponding fuzzy relation.

d) Each fuzzy neuron can be simplified to represent an entire fuzzy inference rule with any number of propositions.

These differences make it possible to carry out any fuzzy computation on the input data and express any kind of ambiguous relationship. This leads to the following definition:

Definition 5-1: A *generic fuzzy neuron* (Figure 5.1) consists of connection functions, an aggregation function and an activation function. It has N inputs $I_1, I_2, \ldots, I_N$, each input consisting of a finite set of elements. Let $X_1 = \{x_{11}, x_{12}, \ldots, x_{1n_1}\}$, $X_2 = \{x_{21}, x_{22}, \ldots, x_{2n_2}\}$,..., $X_N = \{x_{N1}, x_{N2}, \ldots, x_{Nn_N}\}$ denote the N finite sets of inputs. The inputs $I_1, I_2, \ldots, I_N$ are fuzzy sets in the universes of discourse $X_1, X_2, \ldots, X_N$, characterized by membership functions $\mu_{I_1}, \mu_{I_2}, \ldots, \mu_{I_N}$. The inputs are weighted with $W_1, W_2, \ldots, W_N$, which are fuzzy sets in the universes of discourse $S_1 = \{s_{11}, s_{12}, \ldots, s_{1i_1}\}$, $S_2 = \{s_{21}, s_{22}, \ldots, s_{2i_2}\}$,..., $S_N = \{s_{N1}, s_{N2}, \ldots, s_{Ni_N}\}$. The weighting operation is done through *connection functions* of the form:

$$A_i = c_i(X_i, W_i) \tag{5.1}$$

Let C_i be a 3-ary fuzzy relation in $[X_i] \times [S_i] \times [U_i]$; the output of a connection function is expressed by the *-* composition of X_i, W_i, and C_i which is the fuzzy set A_i in the universe of discourse $U_i = \{u_{i1}, u_{i2}, \ldots, u_{ij_i}\}$. The connection outputs, A_i's, may be classified as either excitatory or inhibitory. Let B_i's be the direct inputs to the neuron, where

$$B_i = \begin{cases} A_i & \text{an excitatory input} \\ \overline{A}_i & \text{an inhibitory input} \end{cases} \tag{5.2}$$

$\bar{A}_i$ denotes complement of the fuzzy set A_i, which is defined by the following membership function

$$\mu_{\bar{A}_i}(x) = 1 - \mu_{A_i}(x) \tag{5.3}$$

The set B_i is a fuzzy set in the same universe, U_i, as A_i.

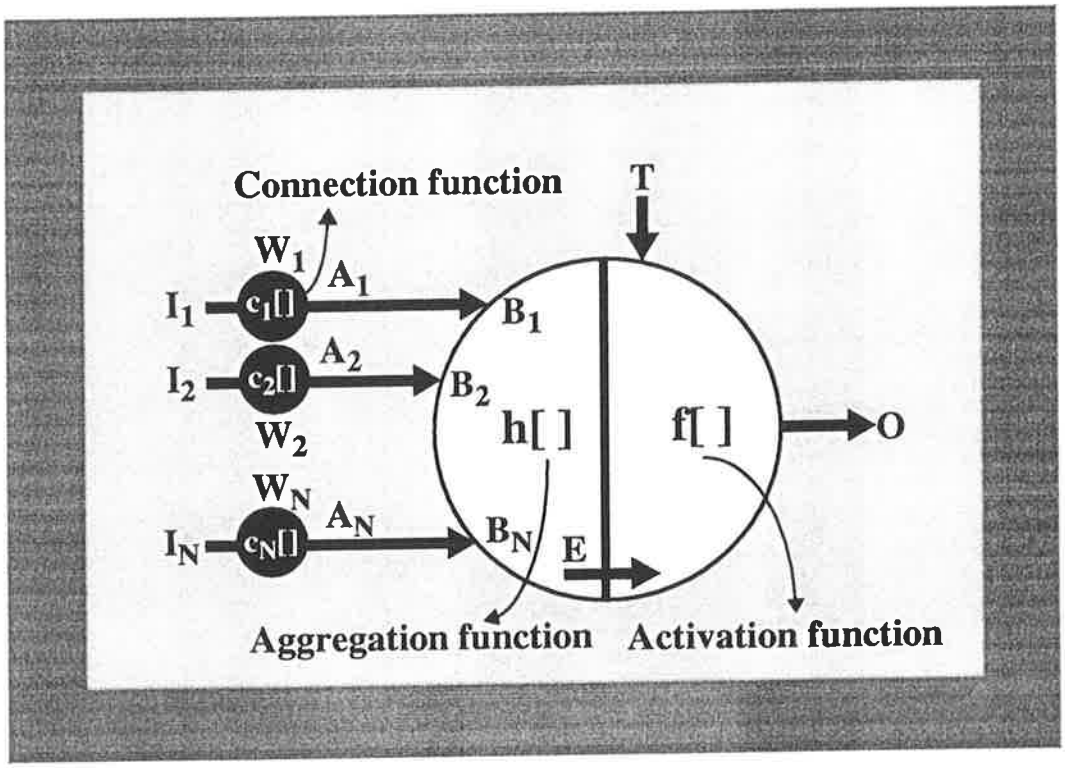


Figure 5.1: The generic fuzzy neuron.

In the next stage, an *aggregation function* h [.] provides an input for the last stage as follows

$$E = h(B_1, B_2, \ldots, B_N) \tag{5.4}$$

Let H be an (N+1)-ary fuzzy relation in $[U_1] \times [U_2] \times \ldots \times [U_N] \times [V]$; the output of the aggregation function is expressed by the *-* composition of $B_1, B_2, \ldots, B_N$, and H, which is the fuzzy set E in the universe of discourse $V = \{v_1, v_2, \ldots, v_k\}$.

The output of the fuzzy neuron is determined by an *activation function,* in the last stage, as follows

$$O = f(E, T) \tag{5.5}$$

where f is the activation function and T is the threshold input which is a fuzzy set over the universe of discourse $Z = \{z_1, z_2, \ldots, z_l\}$. Let F be a 3-ary fuzzy relation in $[V] \times [Z] \times [Y]$. The output O is determined by *-* composition of E, T, and F, and is a fuzzy set in the universe of discourse $Y = \{y_1, y_2, \ldots, y_m\}$.

The generic fuzzy neuron can be simplified if some of the fuzzy sets I_i, W_i, A_i, B_i, E, T, and O are defined over the same universe of discourse, e.g. $X = \{x_1, x_2, \ldots, x_n\}$. In this case, the related fuzzy relations $C_i, H,$ or F can be replaced with any operator from the two basic classes of operations on fuzzy sets, which are the triangular norms (t-norms) and the triangular conorms (t-conorms or s-norms), such as min, max, bounded-difference, algebraic product, and so on. Many types of fuzzy neurons can be defined by changing the functions $c[.], h[.]$, and $f[.]$. In the next section, seven types of fuzzy neurons utilized in the construction of a fuzzy neural network for motion estimation are defined.

Having defined the generic fuzzy neuron (GFN), we describe here how it can be simplified to the four fuzzy neuron models described in Chapter 4. The simplification that should be made in the generic fuzzy neuron parameters to achieve the function of a given fuzzy neuron are as follows:

1) Gupta-Knopf fuzzy neuron (GKFN)

- N inputs are allocated to the GFN: $I_1, I_2, \ldots, I_N$.
- The inputs are defined as fuzzy sets in the same universe of discourse, X.
- The GKFN inputs, x_i $(i = 1, \ldots, N)$, are represented by I_i's.
- The N weight fuzzy sets in the GFN, W_i $(i = 1, \ldots, N)$, are omitted.
- The fuzzy relations in the GFN connection functions, C_i $(i = 1, \ldots, N)$, are set to the corresponding synaptic weighting functions in the GKFN, w_i $(i = 1, \ldots, N)$.
- The output of the connection function is specified by the min-max compositional rule of inference.
- The A_i's in the GFN, which now express the d_i's in the GKFN, are simplified to be fuzzy

sets over the same universe of discourse, U.

- The excitatory and inhibitory inputs to the GKFN are treated, respectively, as excitatory and inhibitory inputs to the GKFN.
- The B_i's in the GFN are equal to the corresponding δ_i's in the GKFN and defined over the universe of discourse U.
- The fuzzy relation H and its *-* compositional rule which denote the aggregation function in the GFN are simplified to the triangular norm operator utilized in the GKFN.
- The fuzzy set E in the GFN represents the fuzzy set u in the GKFN; it is defined in the universe of discourse U.
- The threshold input T in the GFN is omitted.
- Based on the mapping operator $f[.]$ used in the GKFN, a $\beta \times \beta$ fuzzy matrix as the fuzzy relation F, and the max-min composition as the rule of inference, will be employed in the GFN. β is the number of elements in the universe of discourse of the GKFN's output.
- The GFN's output O, represents the GKFN's output, y.

2) Pedrycz-Rocha OR and AND neurons (PROR, PRAND)

- The number of inputs are reduced to one, I_1, in the GFN.
- The input is defined as a fuzzy set in the universe of discourse X, which has $2n$ elements.
- The $2n$ PROR or PRAND inputs are represented by the $2n$ elements of I_1.
- One weight fuzzy set W_1 is allocated in the GFN over the universe of discourse, X.
- The fuzzy relation C_1 and its *-* compositional rule in the GFN are simplified to a minimum operator for the PROR or a maximum operator for the PRAND.
- A_1 in the GFN, which is a fuzzy set in X, expresses the neuron input which is an excitatory input.
- The fuzzy relation H and its *-* compositional rule which denote the aggregation function in the GFN are simplified to the H_1 fuzzy matrix and max-min compositional rule for the PROR or the H_2 fuzzy matrix and min-max compositional rule for the PRAND, where

$$H_1 = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} \tag{5.6}$$

- The fuzzy set E in the GFN is defined over the universe of discourse V which has only one element.
- The threshold input T and the activation function in the GFN are omitted.
- The GFN's output O, which has only one element, represents the output of the PROR or the PRAND.

3) Kwan-Cai fuzzy neuron (KCFN)

- The number of inputs are reduced to one, I_1, in the GFN.
- The input is defined as a fuzzy set in the universe of discourse X, which has N elements.
- The N KCFN's inputs, x_i $(i = 1, \ldots, N)$, are represented by the N elements of I_1.
- One weight fuzzy set W_1 is allocated in the GFN over the universe of discourse, X.
- The fuzzy relation in the GFN's connection functions, C_1, and its *-* compositional rule is simplified to the algebraic product operator for the KCFN.
- A_1 in the GFN which is a fuzzy set in X, expresses the neuron input which is excitatory.
- Based on the aggregation function $h[.]$ used in the KCFN, a suitable fuzzy relation H and compositional rule will be employed in the GFN to implement the function $h[.]$.
- The fuzzy set E in the GFN represents z in the KCFN; it has only one element.
- The threshold input T in the GFN, which has only one element, is used as the threshold input in the KCFN.
- Based on the $f[.]$ and $g_i[.]$ functions used in the KCFN, a suitable 3-ary fuzzy relation H and compositional rule will be employed in the GFN to implement the functions $f[.]$ and $g_i[.]$.
- The GFN's output O, which is a fuzzy set in an universe of discourse with M elements, represents the KCFN's outputs. Each element of O expresses one of the M outputs in the KCFN.

4) Lin-Song fuzzy neuron (LSFN)

- P inputs are allocated to the GFN: $I_1, I_1, \ldots, I_P$.
- The inputs are defined as fuzzy sets over different universes of discourse, $X_1, X_2, \ldots, X_P$.
- The LSFN inputs, u_i $(i = 1, \ldots, P)$, are represented by I_i's.
- P weight fuzzy sets in the GFN, W_i $(i = 1, \ldots, P)$, are allocated to represent the weights in the LSFN, w_i $(i = 1, \ldots, P)$.
- Based on the integration function $f[.]$ used in the LSFN, the fuzzy relations C_i $(i = 1, \ldots, P)$, and H and suitable compositional rules will be employed in the GFN to implement the function $f[.]$.
- The fuzzy set E in the GFN represents the "*net* input" fuzzy set in the LSFN.
- The threshold input T in the GFN is omitted.
- Based on the activation function $g[.]$ used in the LSFN, a suitable fuzzy relation F and compositional rule will be employed in the GFN to implement the function $g[.]$.
- The GFN's output O, represents the LSFN's output.

In general, the weights, the connection functions, the aggregation function, the threshold, and the activation function could be tuned during the learning phase. Therefore, the fuzzy neural network which will be constructed with neurons of this type can learn from experience.

5.3 Architecture of the Fuzzy Neural Network for Motion Estimation

In this section, we introduce an architecture of a fuzzy neural network designed for detection of moving objects and estimation of their velocity. But before describing the detailed architecture of the network, let us first introduce our approach to motion information processing and present details of the algorithm we propose for motion estimation.

5.3.1 Measures of Fuzziness in an Image

A gray-tone image possesses ambiguity within each pixel because of the possible multivalued

levels of brightness a pixel can have [39]. With the concept of fuzzy sets, an image X of size $M \times N$ and L gray-levels can be considered as an array of fuzzy singletons, each having a value of membership denoting its degree of brightness relative to some brightness level l, $l = 0, 1, \ldots, L-1$. In the notion of fuzzy sets, we may therefore write

$$X = \{\mu_X(x_{mn}) = \mu_{mn}/x_{mn};\ m = 1, 2, \ldots M,\ n = 1, 2, \ldots N\} \tag{5.7}$$

or in union form

$$X = \bigcup_m \bigcup_n \mu_{mn}/x_{mn};\ m = 1, 2, \ldots M,\ n = 1, 2, \ldots N \tag{5.8}$$

where $\mu_X(x_{mn})$ or μ_{mn}/x_{mn} $(0 \le \mu_{mn} \le 1)$ denotes the grade of possessing some brightness property μ_{mn} by the (m, n)th pixel intensity x_{mn} [40]. It should be noted that the "/" sign in (5.7) and (5.8) does not denote the arithmetic division. As it was described in Chapter 2, μ_{mn}/x_{mn} expresses a fuzzy singleton.

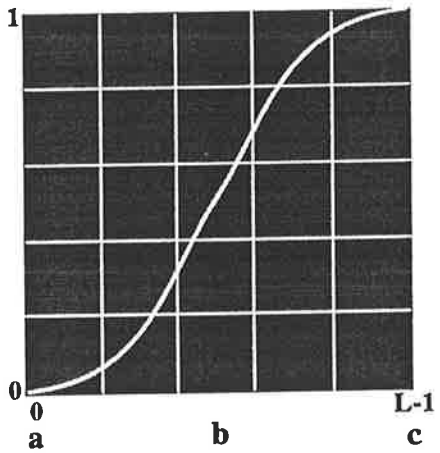


Figure 5.2: Second-order S function.

The fuzzy property μ_{mn} may be defined in a number of ways with respect to any brightness level depending on the problems at hand. In this work, a second-order S function [34] (Figure 5.2) has been used as follows

$$S(x;a,b,c) = \begin{cases} 0 & \text{if } x \le a \\ S_1 & \text{if a} < x \le b \\ S_2 & \text{if b} < x \le c \\ 1 & \text{if } x > c \end{cases} \tag{5.9}$$

where

$$S_1(x;a,b,c) = \frac{(x-a)^2}{(b-a)(c-a)} \tag{5.10}$$

and

$$S_2(x;a,b,c) = 1 - \frac{(x-c)^2}{(c-b)(c-a)} \tag{5.11}$$

5.3.2 Spatio-Temporal Motion Evaluation

Motion can be regarded as orientation in the spatio-temporal domain [14]. This fundamental fact has formed the basis of the algorithm designed for motion estimation in this work. In the proposed method, motion evaluation does not depend on object detection. We assume that the time distance between consecutive images is very small and no significant change occurs between two consecutive frames, i.e., each object moves a maximum of one pixel between two consecutive frames. Our approach is formulated bearing in mind the complexity of the algorithm and the amount of hardware needed for implementing the algorithm in VLSI technology. Since it is intended that the proposed system will form part of a real-time VLSI micro-sensor for motion detection and velocity estimation, to achieve the simplest architecture, we have to establish the mentioned assumption. However, the assumption made is not a limitation of the approach. Furthermore, since on-chip integrated photodetectors will be employed as the image acquisition system hardware, and because the output of photodetector cells can be sampled at different rates, the choice of an appropriate sampling rate makes it possible to take successive image frames within small time intervals. The sampling rate can be calculated according to the maximum velocity an object can have at the nearest distance, and also the resolution of the velocity values we intend to measure. This assumption will significantly reduce the number of connections and nodes in the fuzzy neural network that will be introduced later. Therefore, the system implementation in VLSI technology will be feasible and the cost of the implementation will be cheaper. However, the proposed fuzzy neural network can be easily expanded to cope better with objects that move more than one pixel per frame. The other hypotheses made for this work are:

a) An object possesses a finite speed and so it can not jump from a given pixel to a nonadjacent one.

b) The object can only change direction smoothly and gradually.

c) Noise does not possess properties a and b. It does not also follow the aforementioned assumption.

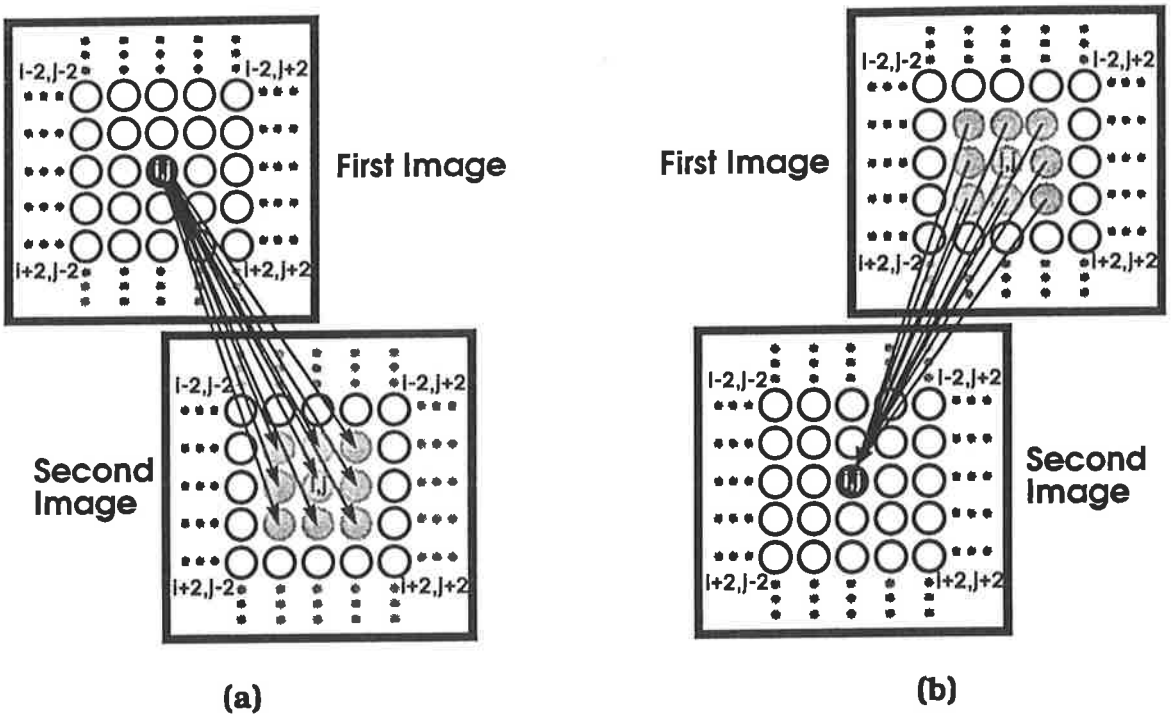


Figure 5.3: The nine possible movements between two consecutive images, (a) a given pixel in the first image can only move to nine positions in the second image, i.e., (b) a given pixel in the second image can only receive a movement from the nine neighbouring pixels in the first image.

In the proposed algorithm, calculation of the velocity vector, consisting of average velocity and motion direction, is done for each pixel individually. According to the aforementioned assumption, each pixel in the first image can move to one of nine possible positions in the second image, as shown in Figure 5.3 (a). Similarly, only one of nine possible pixels in the first image, shown in Figure 5.3(b), can only move to a given pixel in the second image. When noise is present in the input images, it is impossible to find out which of the nine possible pixels in the first image has moved to a given pixel in the second image if the matching criterion is based on a pixel intensity value only. A good solution to this problem is to assign a small sector of the image, consisting of the given pixel and its neighbouring pixels, as a representative for the given pixel, then compare the sector from the second image with sectors of equal size from the first image within a certain search area. In this area the search is done for the position of the maximum similarity between the two sectors. In our approach, the sector is selected as the 3×3-neighbourhood of a pixel as shown in Figure 5.4. It consists of the pixel and its eight neighbouring pixels. This sector is considered as a representative for the pixel.

The search range is limited to the 3×3-neighbouring sectors as shown in Figure 5.5. In other words, to find out which of the nine possible pixels in the first image has moved to a given pixel in the second image, we search for the maximum similarity between the representative sector of the given pixel and the sectors representing pixels in the first image which can move to the given pixel. Obviously, it is feasible to extend the size of the sector and/or the area of the search. The sector size and the search area have been chosen for the sake of simplicity only.

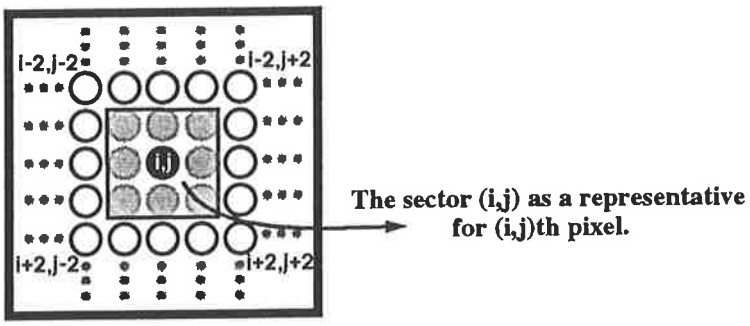


Figure 5.4: The related representative sector of a given pixel which consists of its nine neighbouring pixels.

5.3.3 Similarity Measures

Before proceeding with the details of how the similarity is measured for a given sector in the second image and its nine neighbouring sectors in the first image, let us describe the parameters we employ to measure the similarity and the variables we use to store the average velocity and motion direction of a given pixel.

As mentioned earlier, the input information is fed to the system in the form of two-dimensional image frames, each frame consists of $M \times N$ pixels. As shown in Figure 5.3, between two consecutive frames a pixel can move in nine possible direction only with an average velocity value ranging from 0 to $L-1$. To keep the velocity vector calculation results for a pixel, the following variables and parameters are set for each pixel (see Figure 5.6):

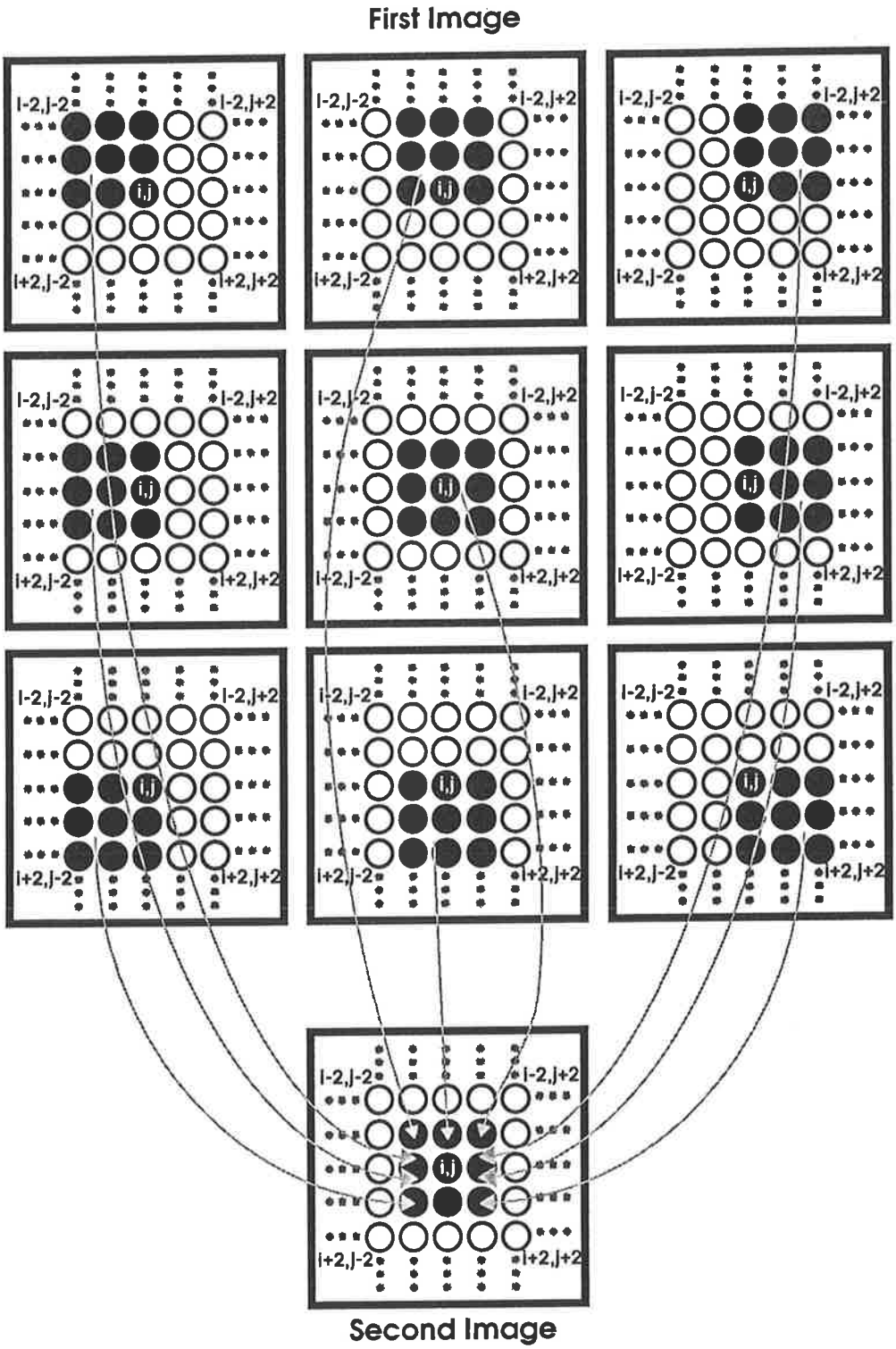


Figure 5.5: The search range. A given sector in the second image can be best matched with one of the nine possible neighbouring sectors in the first image.

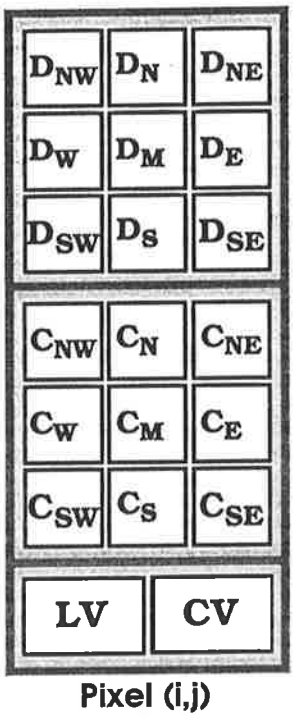


Figure 5.6: Variables which are set aside for each pixel.

1) Last Velocity

LV is a six-bit register denotes the last velocity calculated for a given pixel. The content of this variable is updated when a new movement to the pixel is detected.

2) Current Velocity

CV is a seven-bit counter which denotes the current velocity being measured for a given pixel. The content of this variable is updated once each sampling interval. Whenever a new movement to the pixel is detected, the content of its CV variable is copied into the LV register and then its CV variable is reset to zero.

3) Direction Variables

D_W, D_{NW}, D_N, D_{NE}, D_E, D_{SE}, D_S, D_{SW}, and D_M are nine 6-bit registers called *direction variables*. Each direction variable denotes a membership degree of possessing the velocity value stored in LV by the pixel in the corresponding direction, e.g. north east, middle, south west, etc. These variables are updated together with LV when a new movement to the pixel is detected. A higher value denotes a higher degree of certainty that the system acquires for estimating the motion direction. Calculation of these variables is done using the following function

$$y = \exp\left(-\left(\beta^2 x^2\right)\right) \tag{5.12}$$

where β is a parameter that can be selected by a learning algorithm, and x is an integer in the range 0 to 5. Figure 5.7 illustrates plots of Equation (5.12) for different values of β.

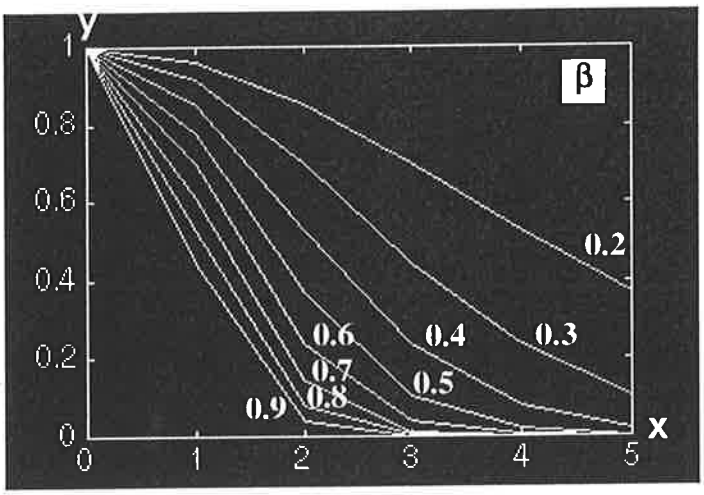


Figure 5.7: Graphical representation of $y = \exp\left(-\beta^2 x^2\right)$ for $\beta = 0.2$ to 0.9.

When a movement to a given pixel is detected, its nine direction variables are numbered from 0 to 5 based on the motion direction. Figure 5.8 demonstrates two examples of this numbering. Then the obtained number of each variable is substituted in Equation (5.12) and the result specifies the new value of the variable.

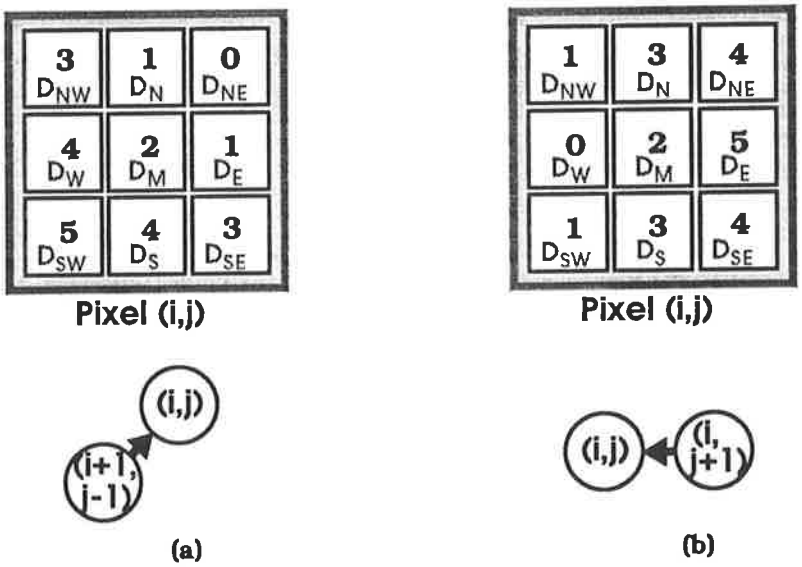


Figure 5.8: Numbering example of the direction variables. (a) Movement from the south-west pixel. (b) Movement from the east pixel.

4) Control Parameters

C_W, C_{NW}, C_N, C_{NE}, C_E, C_{SE}, C_S, C_{SW}, and C_M are nine 6-bit registers called *control parameters*. These parameters are used in the calculation of the similarity measure. They are updated once each sampling interval. In the updating process, the values of three variables are used to calculate the new value of each control parameter: LV, CV, and the corresponding direction variable. Figure 5.9 shows the variables that are needed in the calculation of the north-west control parameter of a given pixel.

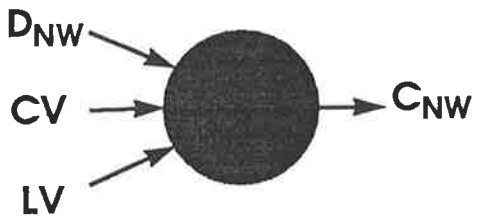


Figure 5.9: The variables which are needed in calculation of a control parameter.

The updating mechanism is formulated as follow:

$$C_X = \begin{cases} \frac{D_X}{LV} \cdot CV & \text{if } 0 \le CV \le LV \\ D_X\left(1 - \frac{CV - LV}{64}\right) & \text{if } LV < CV \le LV + 63 \\ 0 & \text{if } LV + 63 < CV \end{cases} \quad (5.13)$$

where $0 \le D_X \le 63, 0 < LV \le 63, 0 \le CV \le 127$

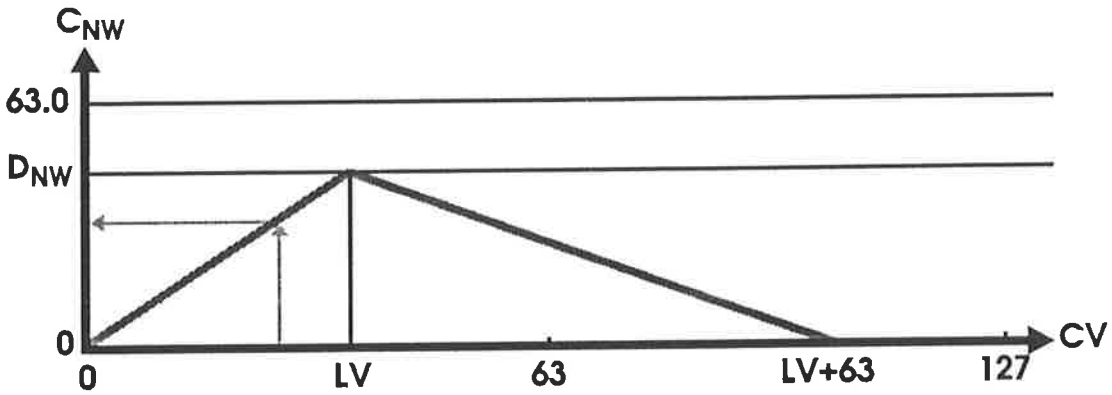


Figure 5.10: Graphical representation of a control parameter updating operation.

Figure 5.10 illustrates a graphical representation of the operation described by Equation (5.13) for updating the control parameters. The main purpose using the control parameters is to suppress the effect of noise in the input information. Moreover, when there is more than one

sector in the second image with patterns similar to the sector in the first image, the best match can be found. The reason is that the time domain as well as the spatial domain is included in the calculation of the best-matched sector. In the spatial domain the similarity is found using the brightness patterns. In the time domain, all the calculations which have been done from the beginning to track the real motion, will produce more certain results.

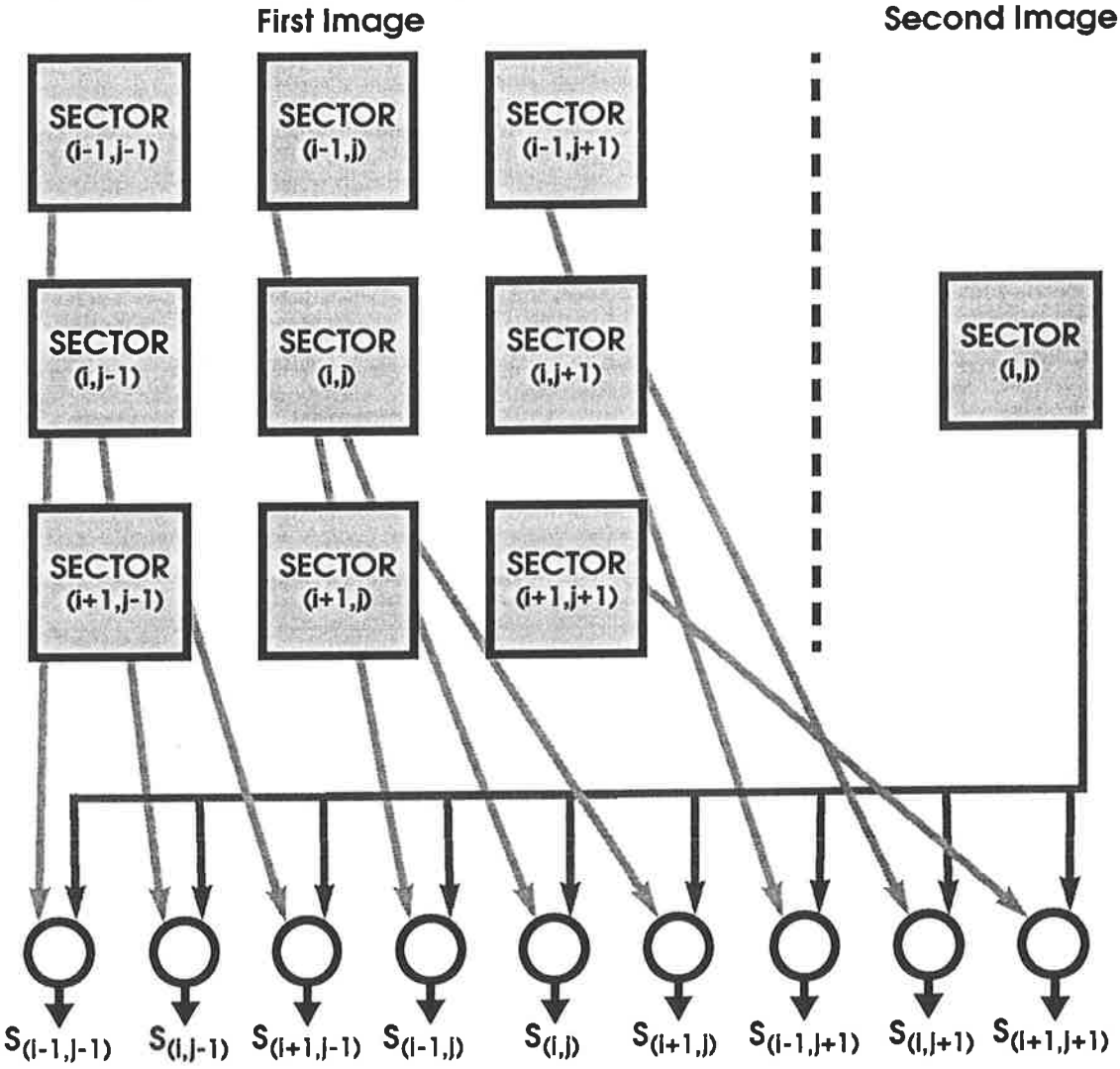


Figure 5.11: The parameters which are employed in the calculation of similarity between the representative sector of a pixel in the second image and its nine neighbouring sectors in the first image.

As it is shown in Figure 5.10, a control variable is a dynamic parameter whose value is changed in each sampling interval. Since the sampling interval is small enough, an object can only change speed and direction smoothly and gradually. When a movement to a given pixel occurs, the movement will be transferred to the next pixel with approximately the same speed and same direction. Therefore, the possibility of transferring the motion from the receiving

pixel to the next pixel is highest when the time index kept in the CV counter of the receiving pixel is nearly equal to the content of the LV register. This fuzzy mechanism reinforces the similarity measure in the next pixel at the appropriate time and weakens it other times. The maximum reinforcement is achieved at the same direction where D_X has its maximum value specified by Equation (5.12). The mechanism of extracting the possibility index, described by Equation (5.13), is illustrated in Figure 5.10.

Having extracted the control parameters, let us describe how the similarity measure is calculated. First, all the similarities between the representative sector of a given pixel in the second image and its nine neighbouring sectors in the first image are measured. The operation on the sector's brightness patterns is done using a fuzzy operator that will be described in the next section. As a result, nine similarity values are obtained for each pixel (see Figure 5.11). Then the nine measured similarities will be strengthened by nine control parameters in each pixel. The control parameters which are used for this purpose come from the neighbouring pixels except the middle control parameter which is taken from the pixel itself (see Figure 5.12).

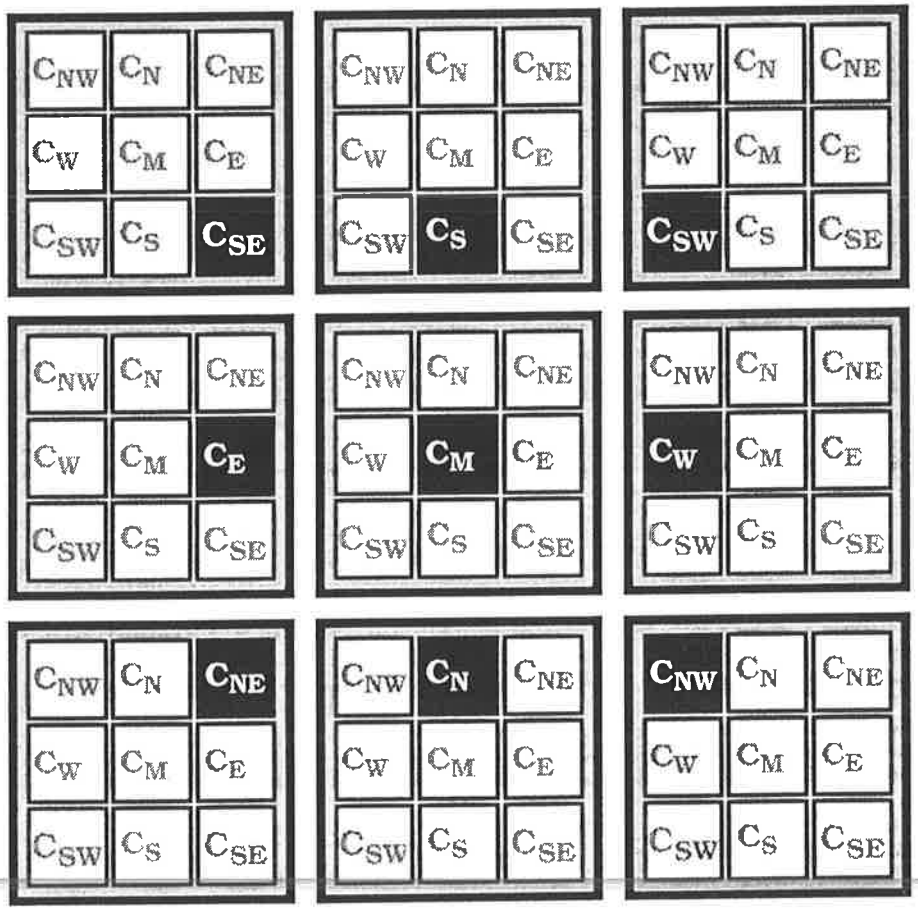


Figure 5.12: Control parameters which are utilized in calculation of the similarity measure for the (i,j)th pixel.

The strengthening operation is a fuzzy operation which will be described in the next section. Therefore, nine match indicators are calculated for each pixel as shown in Figure 5.13. Each match indicator expresses the final degree of similarity between a representative sector of a pixel in the second image and its nine neighbouring sectors in the first image. In the last stage, the best-matched sector will be easily specified using a fuzzy operation discussed in the next section.

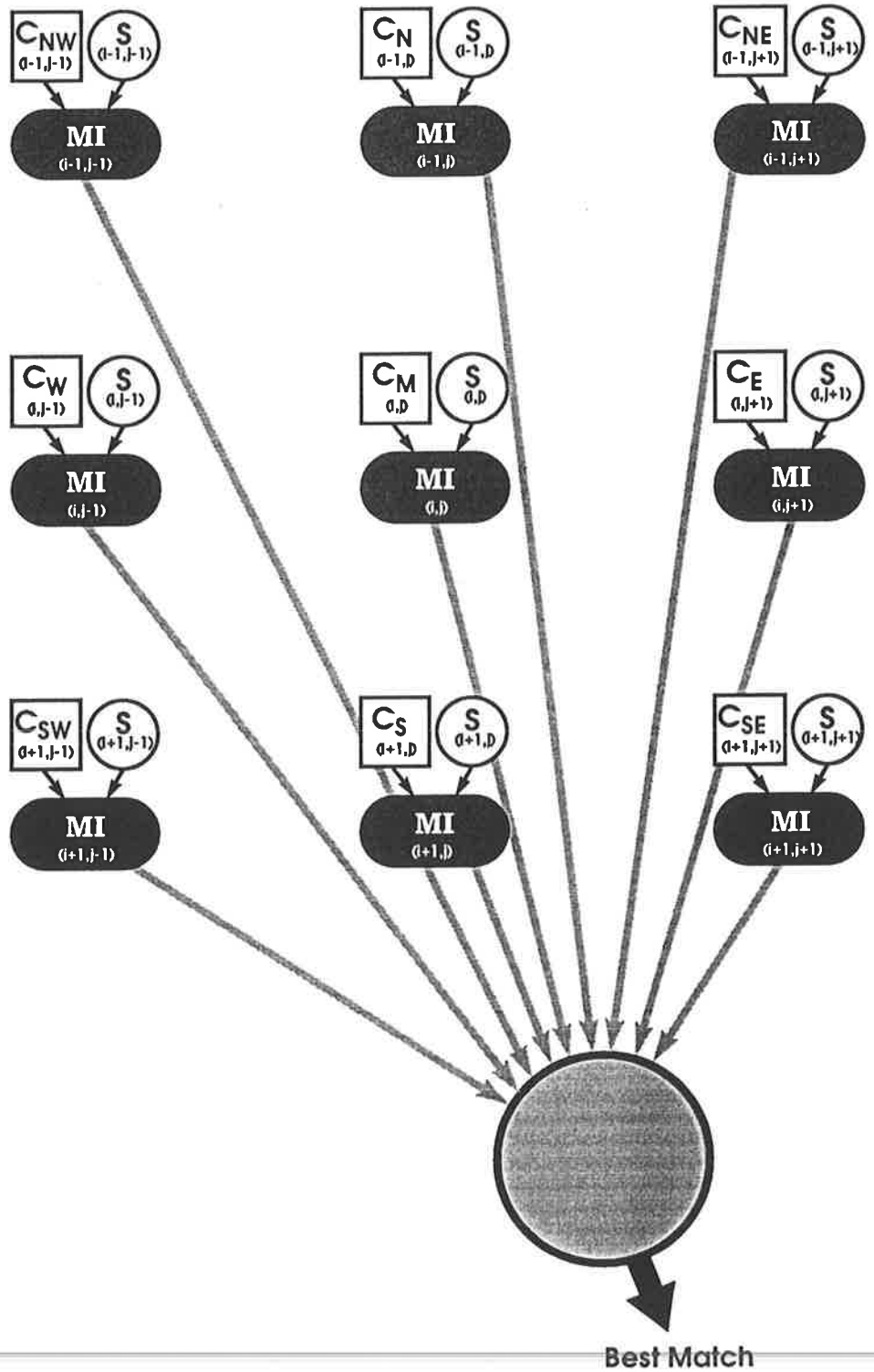


Figure 5.13: The parameters used for extracting the best match between the representative sector of the (i,j)th pixel and its nine neighbouring sectors.

Having extracted the best-matched sector for a given pixel, the average velocity and movement direction are now readily available. If for the (i, j)th pixel in the second image, the (i, j)th sector in the first image is the best match, then no movement to this pixel has occurred. In this case, the CV counter will be incremented. If the content of the CV counter exceeds 127, there is not any movement to this pixel; and hence, the content of the related LV is reset to zero.

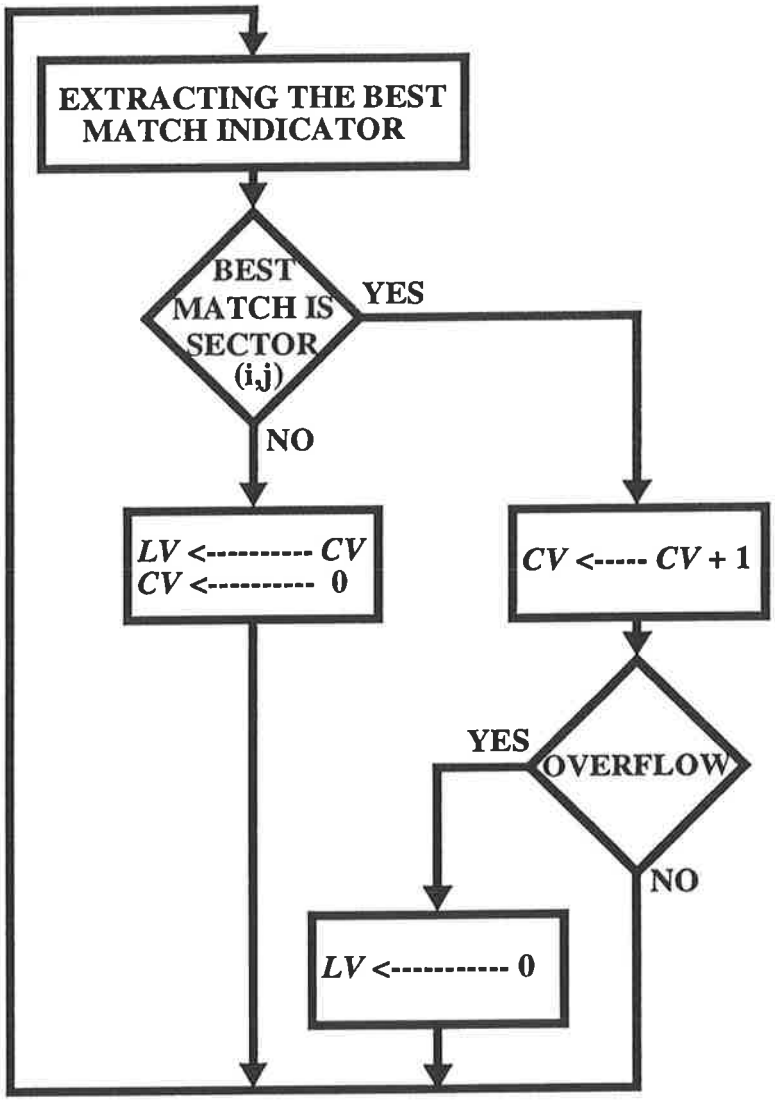


Figure 5.14: Flowchart diagram of the average velocity calculation.

If the best match is not the (i, j)th sector, this means that a movement has occurred from the pixel with the best match index to the given pixel. In this case, the content of CV which represents the time of travel, is transferred into LV. The CV counter will then be reset to zero. To find out the exact amount of the average velocity, the content of LV should be multiplied by the period of the sampling interval. A flowchart of this operation is shown in Figure 5.14.

The features of the proposed algorithm for motion evaluation are

1) Suppressing the effect of noise.
2) Solving the correspondence problem, i.e., the problem of finding the correct match among other possible matches in brightness patterns.
3) Robustness against changes in illumination.

In the next section the architecture of a fuzzy neural network which implements this algorithm is described.

5.3.4 Architecture of the Fuzzy Neural Network

The proposed fuzzy neural network for motion estimation is a six-layer feedforward network with a hierarchical structure as shown in Figure 5.15. The inputs to the network are two matrices of 64×64 pixels of 64 gray-levels. The outputs are velocity vectors consisting of speed and motion direction for each individual pixel of the image. In the following we describe each layer of the fuzzy neural network individually:

5.3.4.1 First Layer

The first layer is the input layer which accepts a pattern into the network. It consists of two sets of 64×64 input fuzzy neurons (INPUT-FNs) shown in Figure 5.16. The first set of INPUT-FNs is allocated to the current frame. The second set is for the previous frame. There is a 64×64 memory unit reserved for storing the previous frame. In each sampling interval, when a new image frame is acquired, the current image is overwritten into the memory forming the previous image frame.

Each INPUT-FN in this layer corresponds to one-pixel of the previous or current frame. To express the input image in terms of fuzzy sets, we consider the image as an array of fuzzy singletons, each having a membership value denoting its degree of brightness in the interval [0,1] (Equations (5.7) and (5.8)). To determine the membership value, μ_{mn}, we use the second order S function (Equations (5.9), (5.10), and (5.11)) as follow:

$$\mu(x) = S(x;0, 31, 63) \tag{5.14}$$

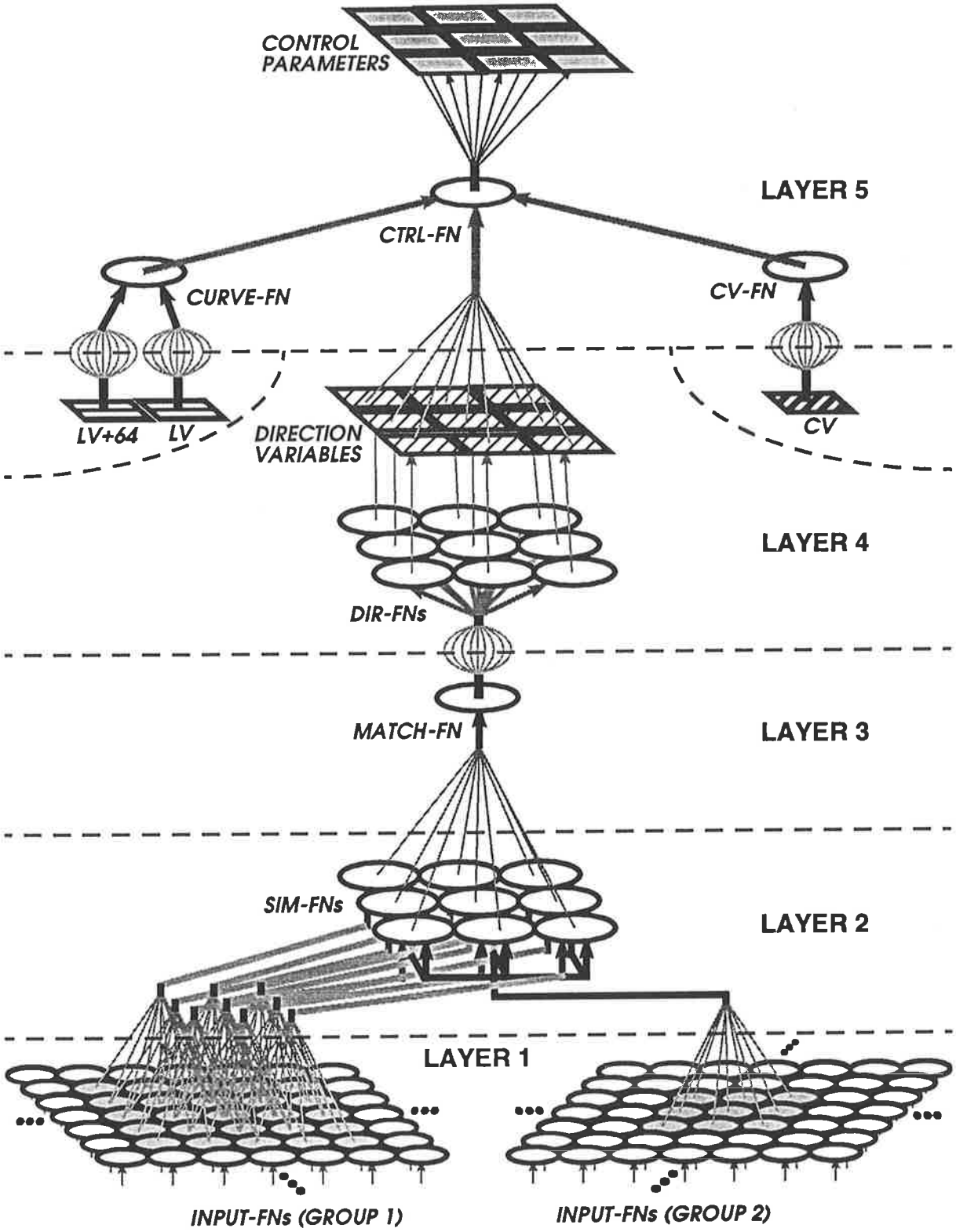


Figure 5.15: Architecture of the proposed fuzzy neural network for motion estimation.

where x is the pixel intensity in the interval $[0, 63]$. Each INPUT-FN determines a membership value for its input pixel by implementing the function described by Equation (5.14). The INPUT-FN has one input which is a fuzzy set over a universe of discourse, X, with 64 elements. All the elements are connected to the given input pixel. The weight is a fuzzy set over the same universe of discourse whose elements are set as follows:

$$W = \{0, 1, \ldots, 63\} \tag{5.15}$$

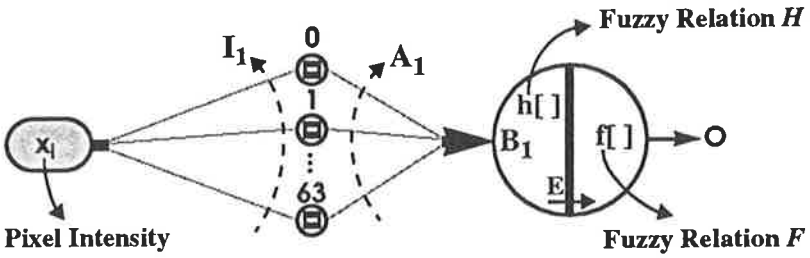


Figure 5.16: Input fuzzy neuron (INPUT-FN).

Since the input and the weight are fuzzy sets over the same universe of discourse, we employ the absolute difference fuzzy operator, ⊟, as connection functions. The input A_1 to the neuron is excitatory, therefore

$$B_1 = A_1 \tag{5.16}$$

The aggregation function $h[.]$ is a 2-ary fuzzy relation, H, which is defined by a 64×64 fuzzy matrix as follows:

$$H = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \tag{5.17}$$

The output of the aggregation function is the fuzzy set E determined by the fuzzy max-min composition of B_1 and H. The fuzzy set E will have 63 elements whose values are 1, and one element whose value is 0. The intensity of the input pixel specifies which elements should be zero. The INPUT-FN has no threshold. The output of INPUT-FN is determined by the

activation function $f[.]$ which is a 2-ary fuzzy relation, F, defined by the following 64×1 fuzzy matrix:

$$F = \begin{bmatrix} S(0;0, 31, 63) \\ S(1;0, 31, 63) \\ \cdots \\ S(63;0, 31, 63) \end{bmatrix} \tag{5.18}$$

The output O, is a fuzzy set in the universe of discourse Y with only one element. It is determined by the fuzzy min-max composition of E and F; it expresses the membership degree of the input brightness in the interval [0,1].

5.3.4.2 Second Layer

The purpose of the second layer is to determine the similarity values between the representative sector of each pixel in the current frame and its neighbouring sectors in the previous frame. The absolute difference fuzzy operator, ⊟, is employed for this purpose. There are nine similarity fuzzy neurons (SIM-FNs) in this layer for each pixel. For the 64×64-pixel input images, a total number of $9 \times 64 \times 64$ SIM-FNs are allocated to the second layer. We choose a 3×3 universe of discourse containing 9 elements (pixels), $X = \{x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}\}$, to express the representative sector of each pixel as described before. For a given pixel, the single output of its corresponding INPUT-FN plus eight single outputs of its neighbouring pixels determine the nine elements of the universe of discourse, i.e., the pixel's representative sector.

As illustrated in Figure 5.17, each SIM-FN has two inputs, I_1 and I_2, which are fuzzy sets over the same universe of discourse, X. The input I_1 is connected to the outputs of the corresponding INPUT-FNs in the previous frame, whereas, the input I_2 is fed by the related current frame INPUT-FN outputs. There is no weighting operation for both I_1 and I_2, i.e. the connection functions are null functions. The input to the neurons are both excitatory thus we have:

$$B_1 = A_1 = I_1 \quad \text{and} \quad B_2 = A_2 = I_2 \tag{5.19}$$

Since both inputs are fuzzy sets over the same universe of discourse, we employ the absolute difference fuzzy operator as the aggregation function of SIM-FNs. The result is a fuzzy set E in the same universe of discourse. E has nine elements each denoting the absolute difference between the corresponding input elements.

The output of the SIM-FN is determined by the neuron's activation function. The employed activation function is a 2-ary fuzzy relation, F, defined by the following 9×1 fuzzy matrix:

$$F = \begin{bmatrix} 1/9 \\ 1/9 \\ \cdots \\ 1/9 \end{bmatrix} \tag{5.20}$$

The output O, is a fuzzy set in the universe of discourse Y with only one element. It denotes the bounded sum of the nine elements of the fuzzy set E. This is done with the aid of the bounded-sum -algebraic-product composition of E and F. The single output expresses the similarity value between its two input sectors in the interval [0, 1]. When two sectors are exactly the same, O will be equal to one.

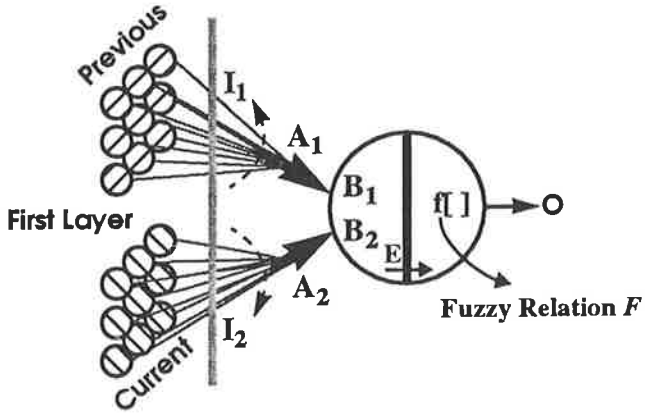


Figure 5.17: Similarity fuzzy neuron (SIM-FN).

5.3.4.3 Third Layer

The third layer is used to determine the match indexes between the representative sector of a pixel in the current frame and its neighbouring sectors in the previous frame. This is done using the similarity values and control parameters described before. For each pixel, the nine

single-element similarities measured by nine SIM-FNs in the second layer are strengthened with nine control parameters. This operation is carried out through a match fuzzy neuron (MATCH-FN). There are 64×64 MATCH-FNs in the third layer, with each neuron allocated to a given pixel. We choose a 3×3 universe of discourse X in the input, containing 9 elements. For a given pixel, the nine SIM-FNs single outputs form the nine elements of the universe of discourse.

As demonstrated in Figure 5.18, each MATCH-FN has only one input, I_1. It is a fuzzy set in the universe of discourse, X. Each element of the fuzzy set I_1 is connected to the corresponding single output of a SIM-FN in the second layer. I_1 is strengthened through connection functions using weights. The weight is a fuzzy set over the universe of discourse X and its elements are set as follows:

$$W = \{c_{SE_{(i-1,j-1)}}, c_{S_{(i-1,j)}}, c_{SW_{(i-1,j+1)}}, c_{E_{(i,j-1)}}, c_{M_{(i,j)}}, c_{W_{(i,j+1)}}, c_{NE_{(i+1,j-1)}}, c_{N_{(i+1,j)}}, c_{NW_{(i+1,j+1)}}\} \tag{5.21}$$

The control parameters and therefore the weight W are changed in each sampling interval. Since the input and weight are fuzzy sets over the same universe of discourse, we employ the algebraic product fuzzy operator as the connection function. The output of the connection function is a fuzzy set A_1 treated as an excitatory input to the fuzzy neuron; therefore,

$$B_1 = A_1 \tag{5.22}$$

The aggregation function is a 2-ary fuzzy relation, H, which is defined by a 9×1 fuzzy matrix as follows:

$$H = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix} \tag{5.23}$$

The output of the aggregation function is a fuzzy set E determined by the fuzzy max-min composition of B_1 and H. The fuzzy set E will have only one element whose value denotes the maximum value of the nine elements of the fuzzy set B_1.

The threshold input T in the MATCH-FN is a fuzzy set in the universe of discourse X as the input. It has nine elements, each element is connected to the corresponding element of fuzzy set A_1. Therefore we have

$$T = A_1 \tag{5.24}$$

This arrangement is to determine the difference between each match indicator, which is an element of the fuzzy set A_1, with the maximum value of the nine match indicators. The output of the MATCH-FN is determined by the activation function which is a 3-ary fuzzy relation, F, defined by the following 9×9 fuzzy matrix

$$F = \begin{bmatrix} 0 & 1 & 1 & \dots & 1 \\ 1 & 0 & 1 & \dots & 1 \\ 1 & 1 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & 1 \\ 1 & 1 & 1 & \dots & 0 \end{bmatrix} \tag{5.25}$$

The output O, is a fuzzy set in the universe of discourse X with nine elements. It is determined by the fuzzy max-bounded-difference composition of E, T, and F. Each element of the output fuzzy set denotes the difference between the corresponding element of the input to the neuron and the element that carries the maximum value.

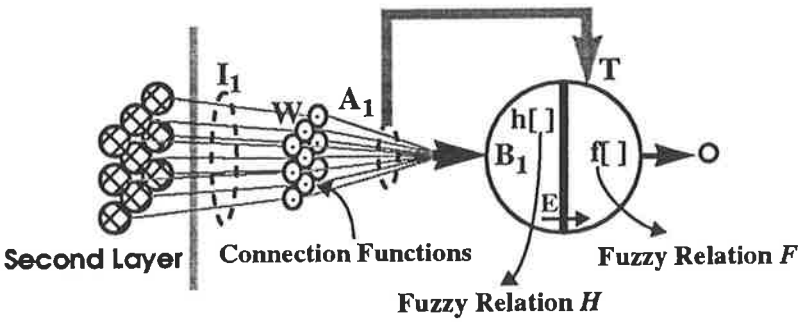


Figure 5.18: Match fuzzy neuron (MATCH-FN).

5.3.4.4 Fourth Layer

The calculation of the direction variables, D_X, is carried out in the fourth layer. This operation

is carried out through nine direction fuzzy neurons (DIR-FNs). There are $9 \times 64 \times 64$ DIR-FNs in the forth layer; each set of nine neurons is allocated to a given pixel. We choose a 3×3 universe of discourse X for the input, containing nine elements.

As shown in Figure 5.19, each DIR-FN has only one input, I_1. It is a fuzzy set in the universe of discourse, X. The connection between the Layer 3 outputs and the Layer 4 inputs is defined as follows:

$$I_{11} = I_{12} = I_{13} = I_{14} = I_{15} = I_{16} = I_{17} = I_{18} = I_{19} = O \tag{5.26}$$

There is no weighting operation for I_1, i.e. the connection functions are identity functions. The input to each neuron is inhibitory thus we have:

$$B_1 = \overline{A_1} \tag{5.27}$$

The aggregation function is a 2-ary fuzzy relation, H, which is defined by a 1×9 fuzzy matrix. The contents of the fuzzy matrices are different for the nine DIR-FNs. The nine fuzzy matrices of the nine DIR-FNs, for a given pixel in the forth layer, are defined as follow:

$$H_1 = [\exp(-\beta^2 0^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 2^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 5^2)]$$
$$H_2 = [\exp(-\beta^2 1^2)\ \exp(-\beta^2 0^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 2^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 5^2)\ \exp(-\beta^2 4^2)]$$
$$H_3 = [\exp(-\beta^2 3^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 0^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 2^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 5^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 3^2)]$$
$$H_4 = [\exp(-\beta^2 1^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 0^2)\ \exp(-\beta^2 2^2)\ \exp(-\beta^2 5^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 4^2)]$$
$$H_5 = [\exp(-\beta^2 4^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 0^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 4^2)]$$
$$H_6 = [\exp(-\beta^2 4^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 5^2)\ \exp(-\beta^2 2^2)\ \exp(-\beta^2 0^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 1^2)]$$
$$H_7 = [\exp(-\beta^2 3^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 5^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 2^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 0^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 3^2)]$$
$$H_8 = [\exp(-\beta^2 4^2)\ \exp(-\beta^2 5^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 2^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 0^2)\ \exp(-\beta^2 1^2)]$$
$$H_9 = [\exp(-\beta^2 5^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 4^2)\ \exp(-\beta^2 2^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 3^2)\ \exp(-\beta^2 1^2)\ \exp(-\beta^2 0^2)] \tag{5.28}$$

The output of the aggregation function is determined by the fuzzy max-algebraic-product composition of B_1 and H, which is a fuzzy set E in the universe of discourse V containing only one element. E determines a value for the corresponding direction variable of a pixel using $\exp(-\beta^2 x^2)$. As it can be seen from the contents of the fuzzy matrices, in a given DIR-FN, this arrangement allows the neuron to behave like a lens so that the neuron focuses on one

of the input elements which is directly related to the direction value controlled by the neuron. However, the neuron uses the other elements in the computation of the direction value with lower degree of importance. β is assigned to 0.5 in the system simulations, based on a few experiments which were conducted to obtain a suitable value for β; however, the best value for this parameter should be extracted from a learning process. We will introduce a learning algorithm for this purpose in our future work.

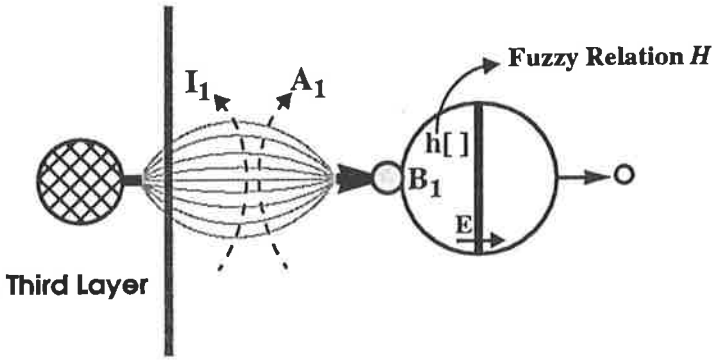


Figure 5.19: Direction fuzzy neuron (DIR-FN).

There is no threshold input and no activation function for DIR-FN, i.e., the neuron's output is equal to the aggregation function output

$$O = E \tag{5.29}$$

The output O has one element that denotes a value for the corresponding direction variable of a pixel in the interval [0, 1]. For a given pixel, Figure 5.20 shows how each direction variable is driven by a DIR-FN. The direction variables which are memory cells accept the DIR-FN outputs when a movement to a pixel is detected, i.e., the output of the fifth neuron is not the maximum one.

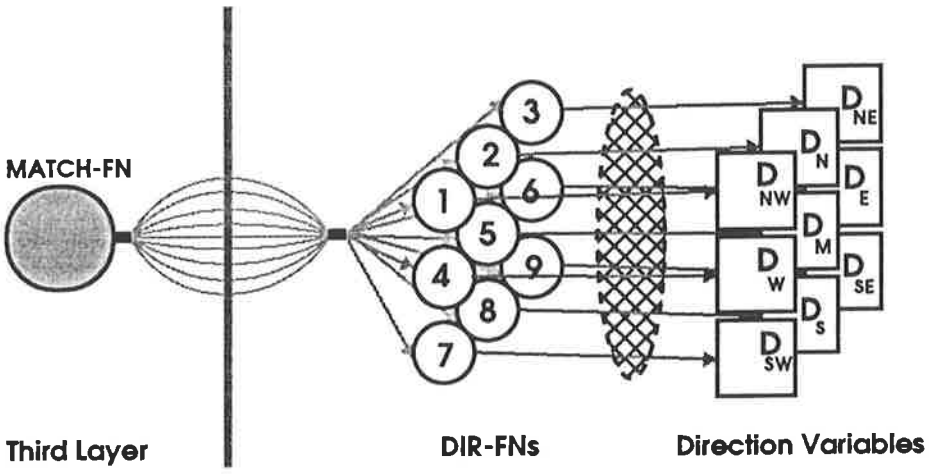


Figure 5.20: The connections between DIR-FNs and direction variables.

5.3.4.5 Fifth Layer

The fifth layer is the last layer of the fuzzy neural network. The calculation and updating of the control parameters, C_X's, are carried out in this layer. There are three different types of fuzzy neurons used for extracting the control parameters of a given pixel. Therefore, the total number of the neurons in Layer 5 are $3 \times 64 \times 64$. For a given pixel, the inputs to this layer are the pixel's LV register, the pixel's CV counter, and nine direction parameters. The output of this layer are nine control parameters whose values vary in the interval [0, 1]. As mentioned in the previous section, the calculation of the control parameters is done using Equation (5.13). This function is implemented by a curve fuzzy neuron (CURVE-FN), a current velocity fuzzy neuron (CV-FN), and a control fuzzy neuron (CTRL-FN).

A CURVE-FN builds the curve shown in Figure 5.10 using the LV value. It has two inputs I_1 and I_2 in the same universe of discourse X with 128 elements. All the elements of the first input are connected to LV and the second input to $LV + 64$. Since LV is a binary number between 1 and 63, we can simply add an extra bit which is always "high" as the MSB. Therefore, we will have $LV + 64$ as the result. There are two weights W_1 and W_2 which are fuzzy sets over the same universe of discourse, X. W_1 and W_2 elements are set as follows:

$$W_1 = W_2 = \{0, 1, \ldots, 127\} \tag{5.30}$$

We employ the algebraic divide fuzzy operator, $\boxed{/}$, as connection functions. The input A_1 to the neuron is excitatory whereas the input A_2 is inhibitory; therefore,

$$B_1 = A_1 \ , \ B_2 = \overline{A_2} \tag{5.31}$$

The fuzzy set B_1 forms the left part of the curve and the fuzzy set B_2 forms its right part. The aggregation function is a min fuzzy operator. The output of the aggregation function is a fuzzy set E over the same universe of discourse X with 128 elements. It contains 128 elements of the whole curve. There is no threshold input and no activation function for the CURVE-FN. The output O of the neuron is defined as follows:

$$O = E \tag{5.32}$$

It represents the curve shown in Figure 5.10 without considering the scaling effect of the

corresponding direction variable. This curve will be scaled down in the CTRL-FN to achieve the curve described by Equation (5.13).

The CV-FN is similar to the INPUT-FN but it does not employ an activation function. Each CV-FN has one input which is a fuzzy set over the universe of discourse X with 128 elements. All the elements are connected to the CV's output. The weight is a fuzzy set over the same universe of discourse X and its elements are set out as follows:

$$W = \{0, 1, \ldots, 127\} \tag{5.33}$$

We employ the absolute difference fuzzy operator, ⊟, as connection functions. The input A_1 to the neuron is excitatory; therefore,

$$B_1 = A_1 \tag{5.34}$$

The aggregation function $h[.]$ is a 2-ary fuzzy relation, H, which is defined by a 128×128 fuzzy matrix as follows:

$$H = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \tag{5.35}$$

The output of the aggregation function is a fuzzy set E determined by the fuzzy max-min composition of B_1 and H. The fuzzy set E will have 127 elements whose values are 1, and one element whose value is 0. The CV-FN has no threshold input and no activation function; output is defined as

$$O = E \tag{5.36}$$

It is a fuzzy set with 128 elements which denotes the input in terms of fuzzy sets.

A CTRL-FN has two inputs I_1 and I_2 in the same universe of discourse X with 128 elements. The elements of the first and the second inputs are connected to the corresponding elements of the CURVE-FNs output and the CV-FNs output respectively. There are no connection functions and the inputs are excitatory and inhibitory, respectively. We have

$$B_1 = A_1 \quad B_2 = \overline{A_2} \tag{5.37}$$

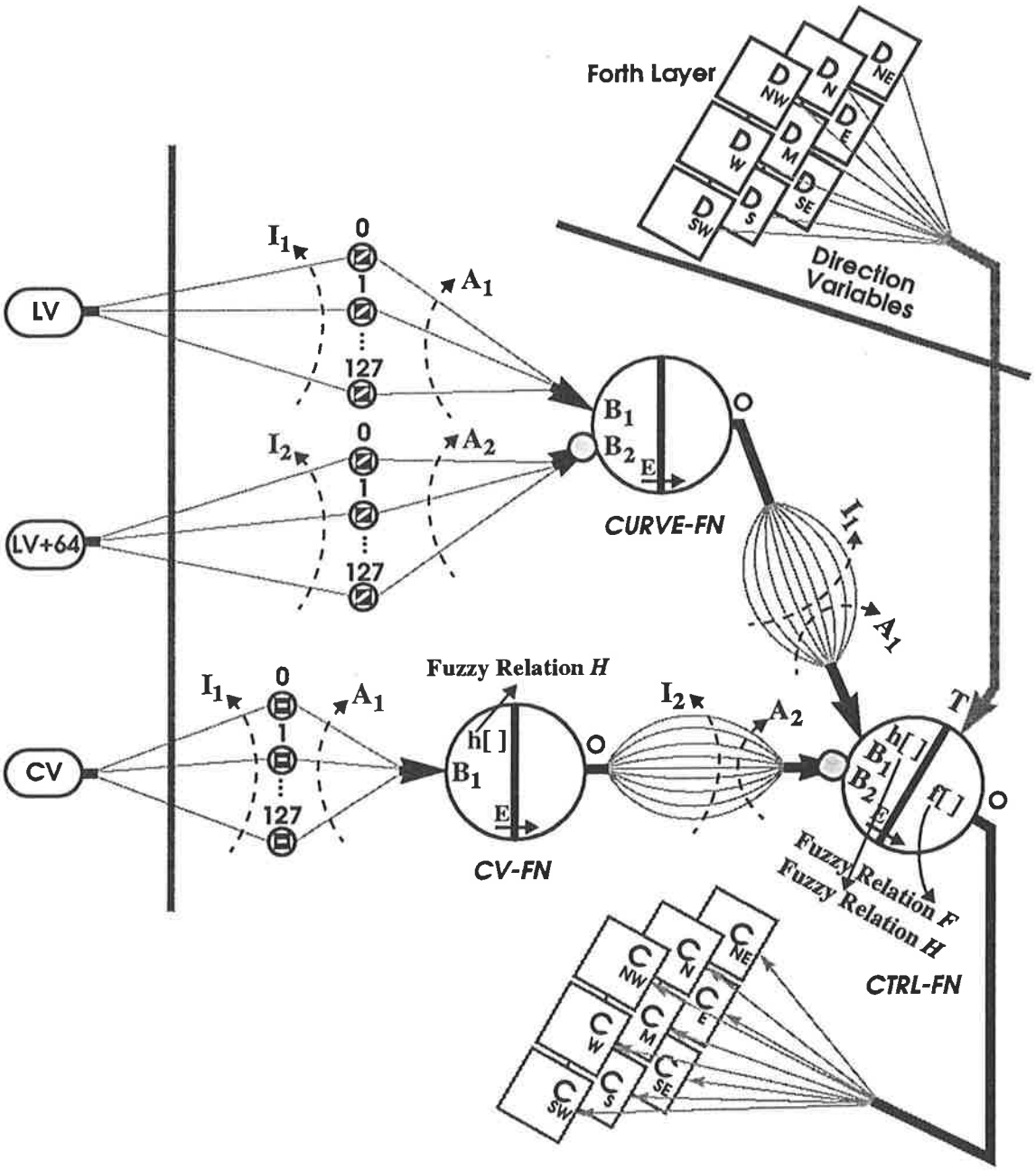


Figure 5.21: CURVE-FN, CV-FN, and CTRL-FN plus their connections.

The aggregation function is a 3-ary fuzzy relation, H, which is defined by a $128 \times 128 \times 1$ fuzzy matrix as follows:

$$H = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \tag{5.38}$$

The output of the aggregation function is a fuzzy set E determined by the fuzzy max-min composition of B_1, B_2, and H. The fuzzy set E will only have one element whose value denotes the output of the function described by Equation (5.13) to its input which is driven by CV. This value is scaled down by nine direction variables producing the new values of nine control parameters; this is done by the activation function. The threshold input T is a fuzzy set in the universe of discourse Z with nine elements. Each element of the threshold input is connected to one direction variable. The activation function is a 3-ary fuzzy relation, F, defined by the following $1 \times 9 \times 9$ fuzzy matrix:

$$F = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \tag{5.39}$$

The output O, is a fuzzy set in the universe of discourse Y with nine elements. It is determined by the max-algebraic-product composition of E, T, and F. Each element of the output fuzzy set denotes a value for the corresponding control parameters. Figure 5.21 illustrates the fuzzy neurons defined for Layer 5 and their connections.

5.4 Conclusions

This chapter constitutes the core of the thesis. In this chapter, a generic model of a fuzzy neuron was defined as an extended model of the existing fuzzy neurons. It was shown that the generic fuzzy neuron can be simplified to four models of fuzzy neurons described in Chapter 4. Then the bases of the approach we employed for motion estimation was presented. In the proposed algorithm, motion estimation is carried out not only using a comparison between brightness patterns in consecutive image frames by fuzzy relations, but also a control mechanism is established to strengthen or weaken the results of the comparisons in brightness patterns. The main features of the algorithm are:

1) Suppressing the effect of noise.
2) Solving the correspondence problem, i.e., the problem of finding the correct match among other possible matches in brightness patterns.
3) Robustness against changes in illumination.

The only limitation of the algorithm is the assumption that the time distance between consecutive images is very small and no significant change occurs between two consecutive images, i.e., each object moves a maximum of one pixel per frame. We explained that this assumption, is not a limitation of the approach, but a simplification. The approach can be easily extended to deal with the objects travelling more than one pixel per frame.

Next, the architecture of a fuzzy neural network which was designed to implement the proposed motion estimation algorithm was given. It has a five-layer feedforward structure. Seven different fuzzy neurons were defined and utilized in different layers of the fuzzy neural network. The fuzzy neurons are simplified versions of the generic fuzzy neuron. Although it was mentioned that the fuzzy neural network constructed with the generic fuzzy neurons can learn from experience, we did not present a learning algorithm for training the system. We will define and use a simple learning algorithm based on the generalized delta rule for this purpose in our future work.

In the next chapter we will study the results of more than thirty simulations of the proposed fuzzy neural network. An evaluation of the performance of the fuzzy neural network under various conditions will be presented.

Chapter 6

EXPERIMENTAL RESULTS

6.1 Introduction

Simulation studies have been conducted to demonstrate the performance of the proposed fuzzy neural network for motion estimation. Two different experiments were carried out. The first consisted of five simulations for moving objects with different speeds and trajectories. The second consisted of twenty four simulations for a moving object with six different velocities under four different noise conditions. In this chapter the experimental results are reported and discussed but first we describe the procedure for conducting the experiments.

6.2 Experimental Procedure

The experiments were conducted using a simulation program developed by the author and run on a SPARC station. The software consists of two C programs: one for producing input image frames and another one for implementing the network. The first program is capable of producing 64×64, 6-bit image frames containing moving objects. The number of moving objects, their shapes, their brightness patterns, their speeds, their trajectories, and also the background pattern can be changed. The outputs of the program are binary image files, each file represents one image frame. The program is also capable of adding noise to the images. For this purpose, it reads external files containing noise data and adds them to the images. The program can easily be modified to produce image frames with higher resolution, e.g. 256×256, 8-bit images.

The second program is the fuzzy neural network simulator. It reads the input binary image file, performs the simulation, and outputs the results in each clock cycle. The calculation of the average velocity is done based on the frequency of this clock signal. The output consists of average speeds and motion directions of all pixels in each clock cycle. The output appears in the form of a binary file. The program allows the parameter β of the DIR-FNs used in the fourth layer of the network to be changed by the user.

Two main experiments were conducted, each experiment containing a different number of simulations. The first consisted of five simulations for moving objects with different speeds and trajectories. The objective of this experiment is merely to test the simulator and evaluate the motion estimator. The second experiment consisted of twenty four simulations for a moving object with six different velocities under four different noise conditions. The objective of this experiment was to measure the error values in estimated average velocities in noise free and noisy environments.

6.3 First Experiment

This experiment was done to test the performance of the system in detecting moving objects and estimating their average velocities. We considered objects of different shapes, brightness patterns, and velocities. In this experiment we have conducted five different simulations. We present one figure for each simulation in this thesis. Each figure is divided into three sections: top, middle, and bottom. In the top part of the figure we present ten frames from the input image sequence which are selected from ten equal time slices. The frame number is presented under each image.

In the middle part of each figure ten pictures are presented, with each picture illustrating the last average velocities of the 64×64 pixels. The darker the pixel is, the higher is its average velocity. The fuzzy neural network calculates the 6-bit average velocities in terms of *pixel per frame* (ppf). The maximum average velocity an object can possess and be detectable by the system is 1 ppf. As the number under the picture shows, the picture is the output of the fuzzy neural network after processing the corresponding input frame shown in the top section of the figure.

In the bottom section of the figure two graphs are presented. Each graph contains arrows of different size and direction. The size of each arrow represents the last average velocity of the associated pixel; the smaller the arrow is, the lower is the pixel's average velocity. The direction of an arrow shows the direction of the last movement to the pixel. For the pixels with no arrows displayed, the average velocity is zero. The left graph represents the output of the fuzzy neural network after processing the last image frame in each simulation. The right graph shows a selected part of the left graph containing 20×20 pixels. As can be seen from the figures, the arrows are displayed bigger to make the difference between the arrow sizes more distinguishable.

Simulation 1.1: In this simulation we examined the response of the fuzzy neural network to an object moving horizontally from left to right. During movement, the object increases its speed continuously. When the object leaves the scene in the last frame, it possesses the maximum possible speed, i.e., 1 ppf. We used 1739 image frames in this simulation. Figure 6.1 shows the results of this simulation.

Simulation 1.2: This simulation was done on a scene with three objects moving horizontally. The objects have different sizes, speeds (0.1 ppf, 0.04 ppf, and 0.02857 ppf), and directions. The middle object has the smallest size but the largest speed (0.1 ppf), and travels from right to left. The other two objects travel from left to right with the bottom having the smallest speed (0.02857 ppf). We used 2430 image frames to conduct this simulation. The results of this simulation are represented in Figure 6.2.

Simulation 1.3: In this simulation, there are two objects moving diagonally from top to bottom in opposite directions. The object which starts from top-left corner and finishes at bottom-right corner, travels with a lower speed (0.06667 ppf) than the other object which moves with a speed of 0.2 ppf. We used a nonuniform background pattern and different object shapes to show the ability of the system to detect motion under different background conditions. We used 770 image frames in this simulation. Figure 6.3 shows the results obtained from this simulation.

Simulation 1.4: Figure 6.4 shows the results of this simulation. The input image sequence contains a single object moving around a circle with a constant speed of 0.2 ppf. The

background brightness varies continuously from the dark to bright. We employed 741 image frames in this simulation.

Simulation 1.5: Figure 6.5 shows the response of the network to the last simulation of this experiment. Here an object is moving circularly with varying speed. The object starts travelling clockwise from top of the circle with the lowest speed in this experiment (0.02381 ppf). The object increases its speed continuously. When the object rotates 90°, the speed reaches 0.2 ppf (which is the highest speed in this simulation). Then the object starts decelerating, and the speed is decreased continuously until the object reaches the bottom of the trajectory. This trend is repeated again until the object returns to the starting point (i.e., top of the circle). The total number of 3624 image frames were used in this simulation.

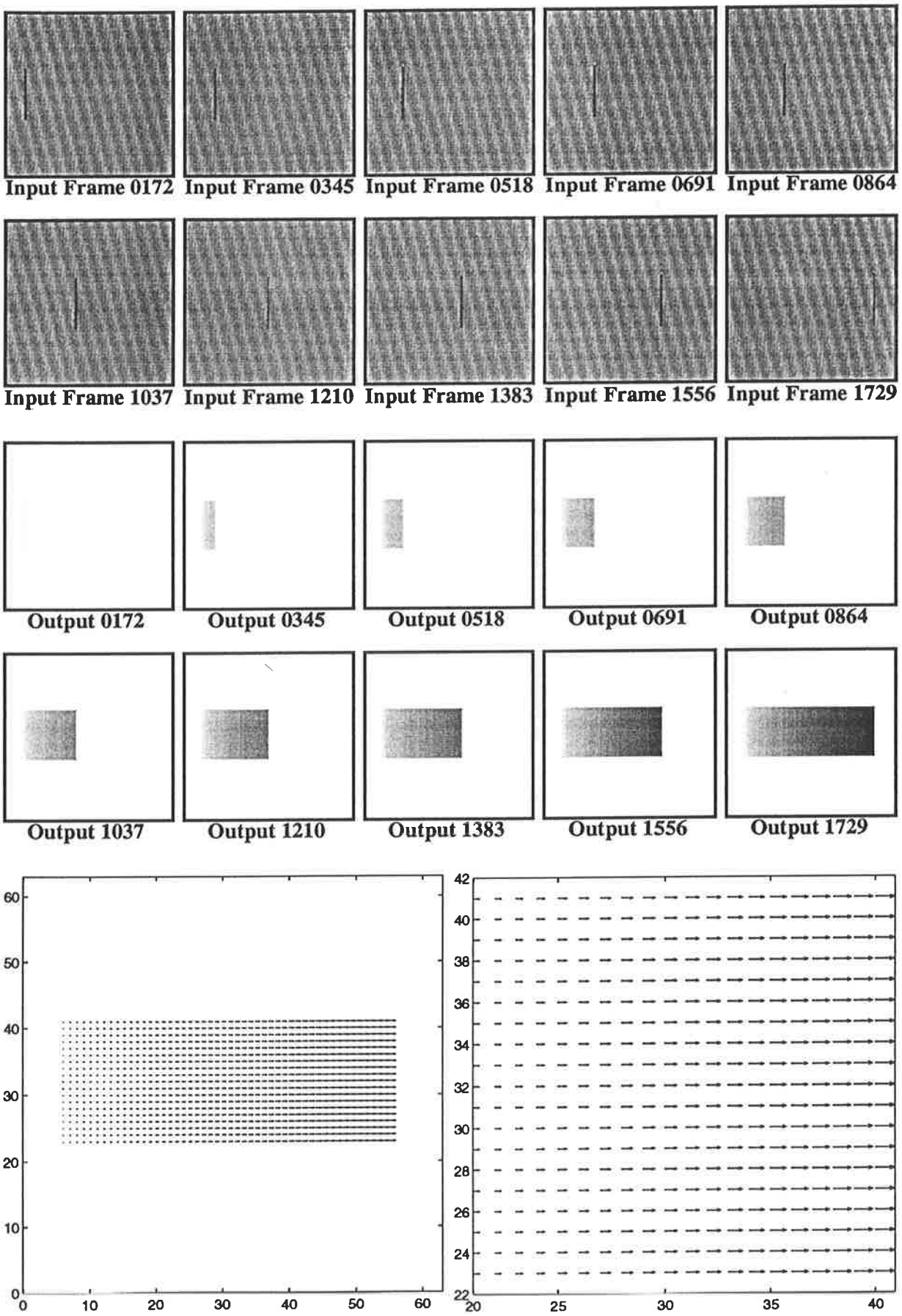


Figure 6.1: An object moving horizontally from left to right with varying speed.

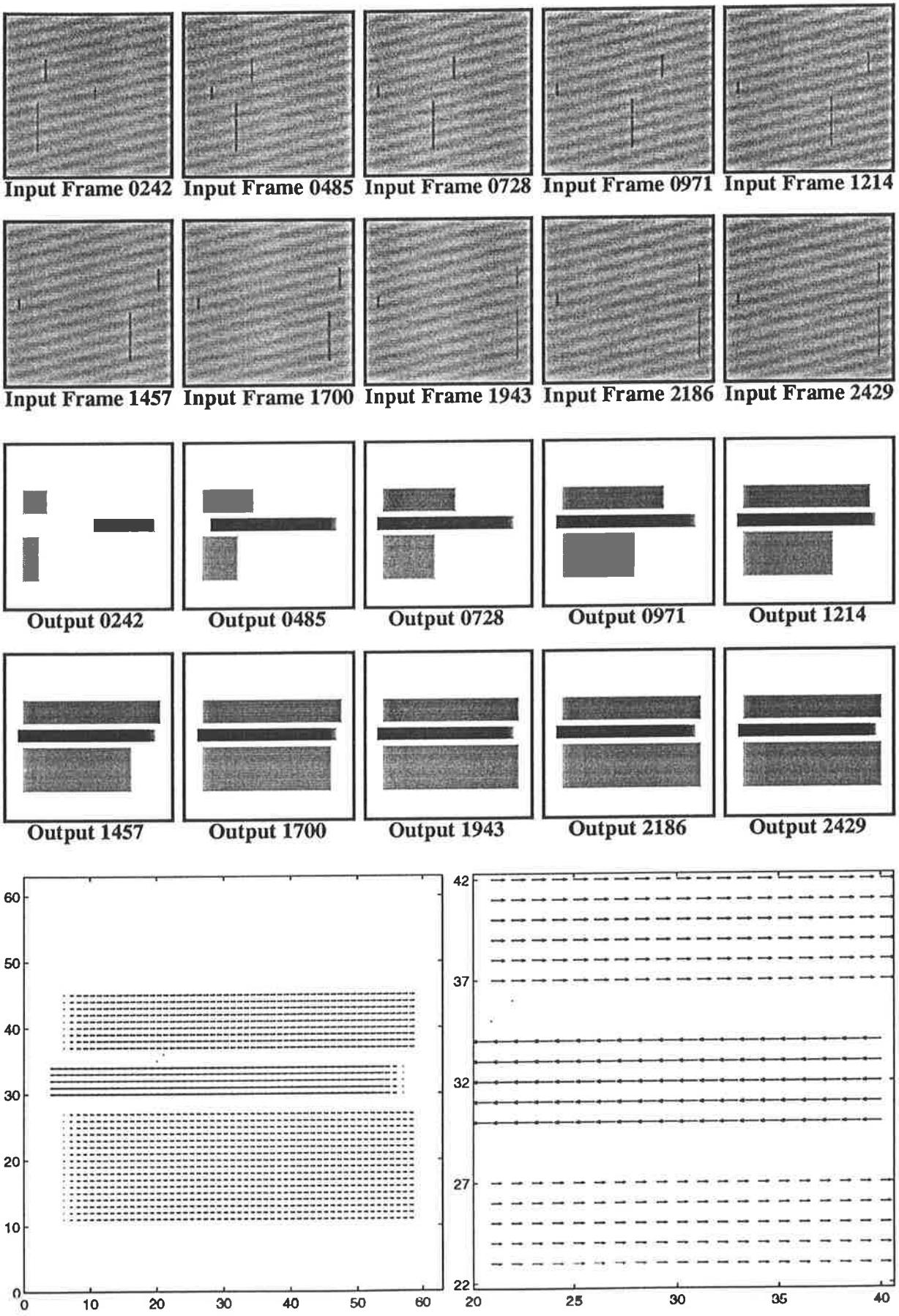


Figure 6.2: Three objects moving horizontally with different speeds and directions.

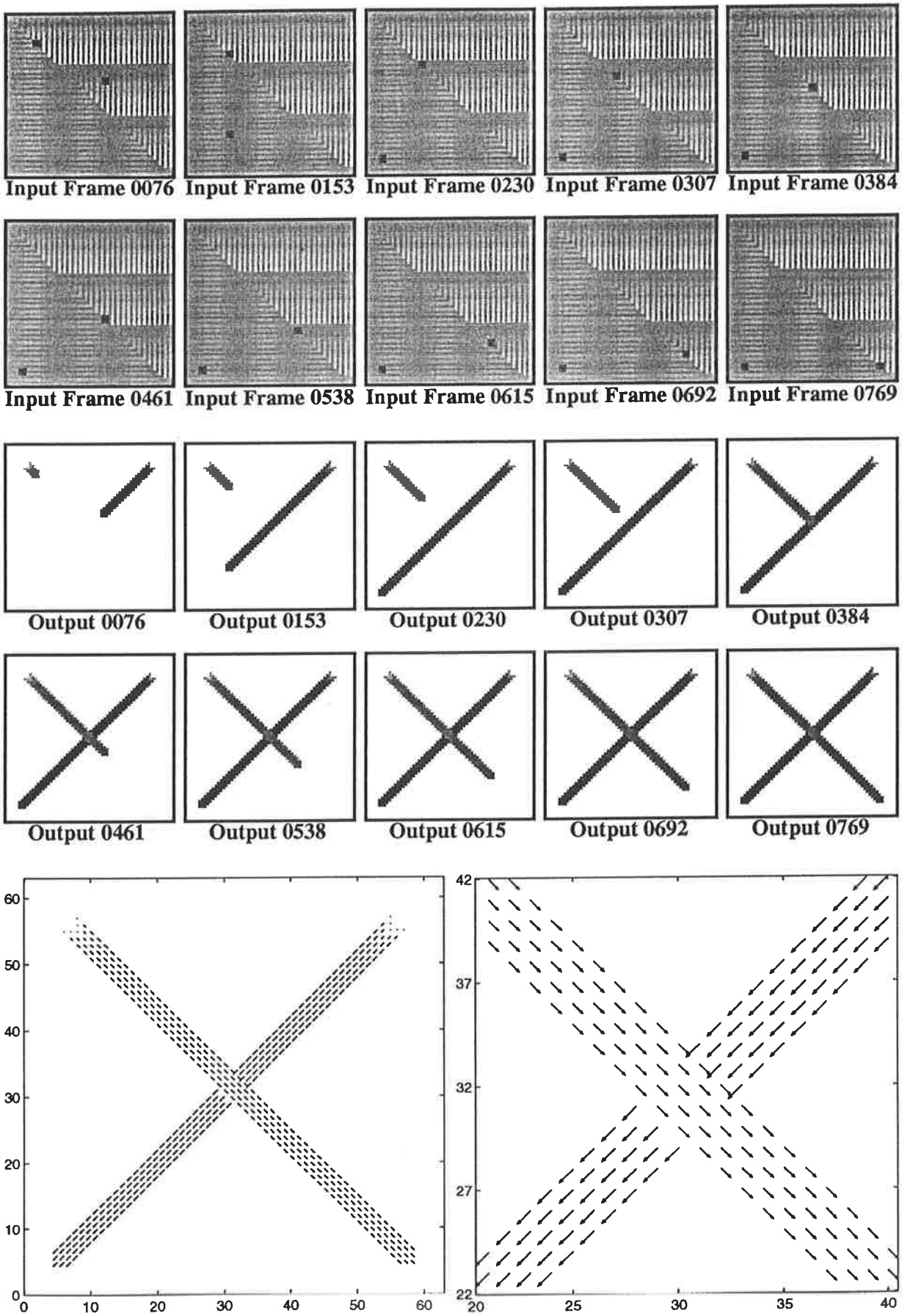


Figure 6.3: Two objects moving diagonally in opposite directions with different speeds.

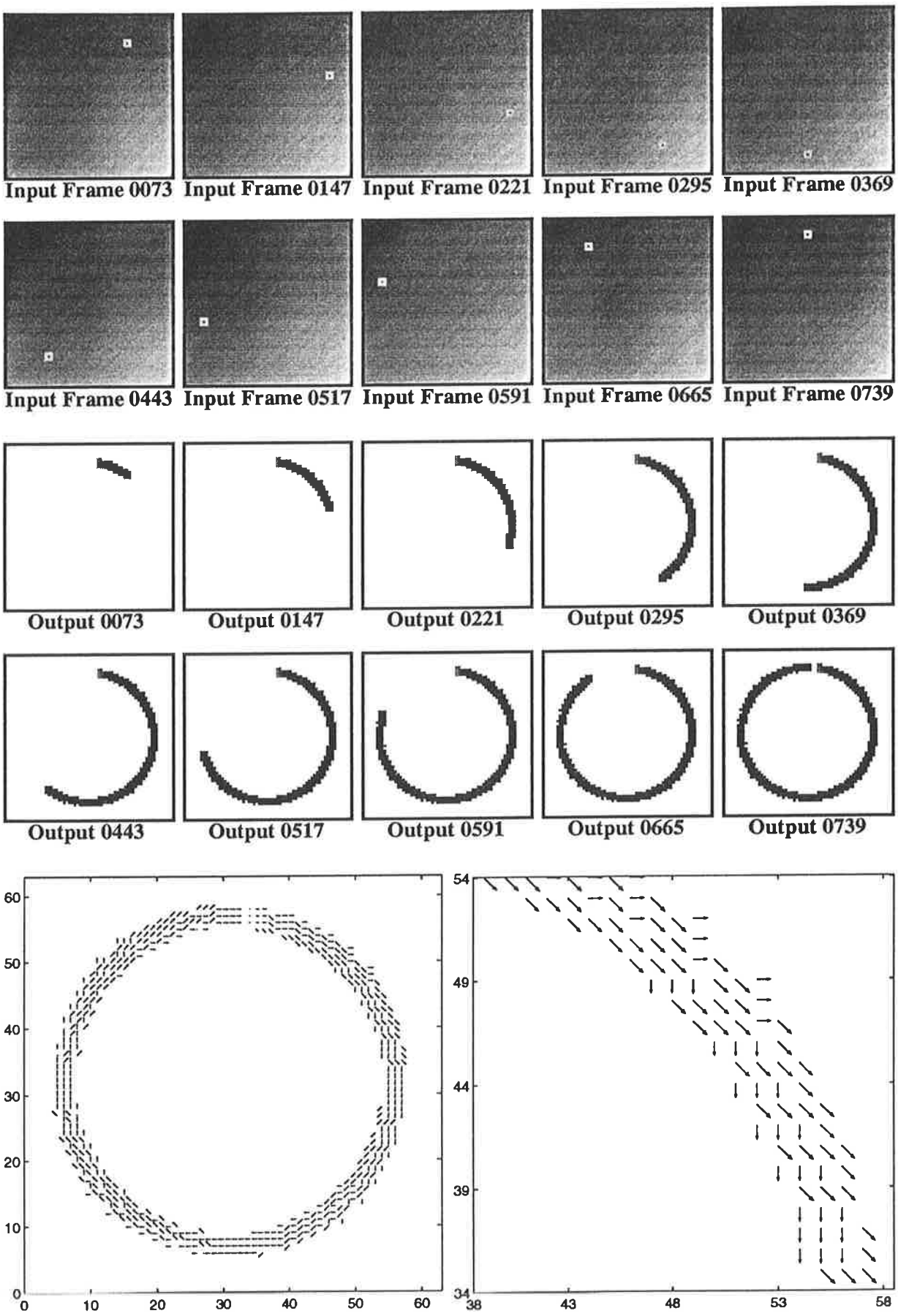


Figure 6.4: An object moving on a circular path with a constant speed.

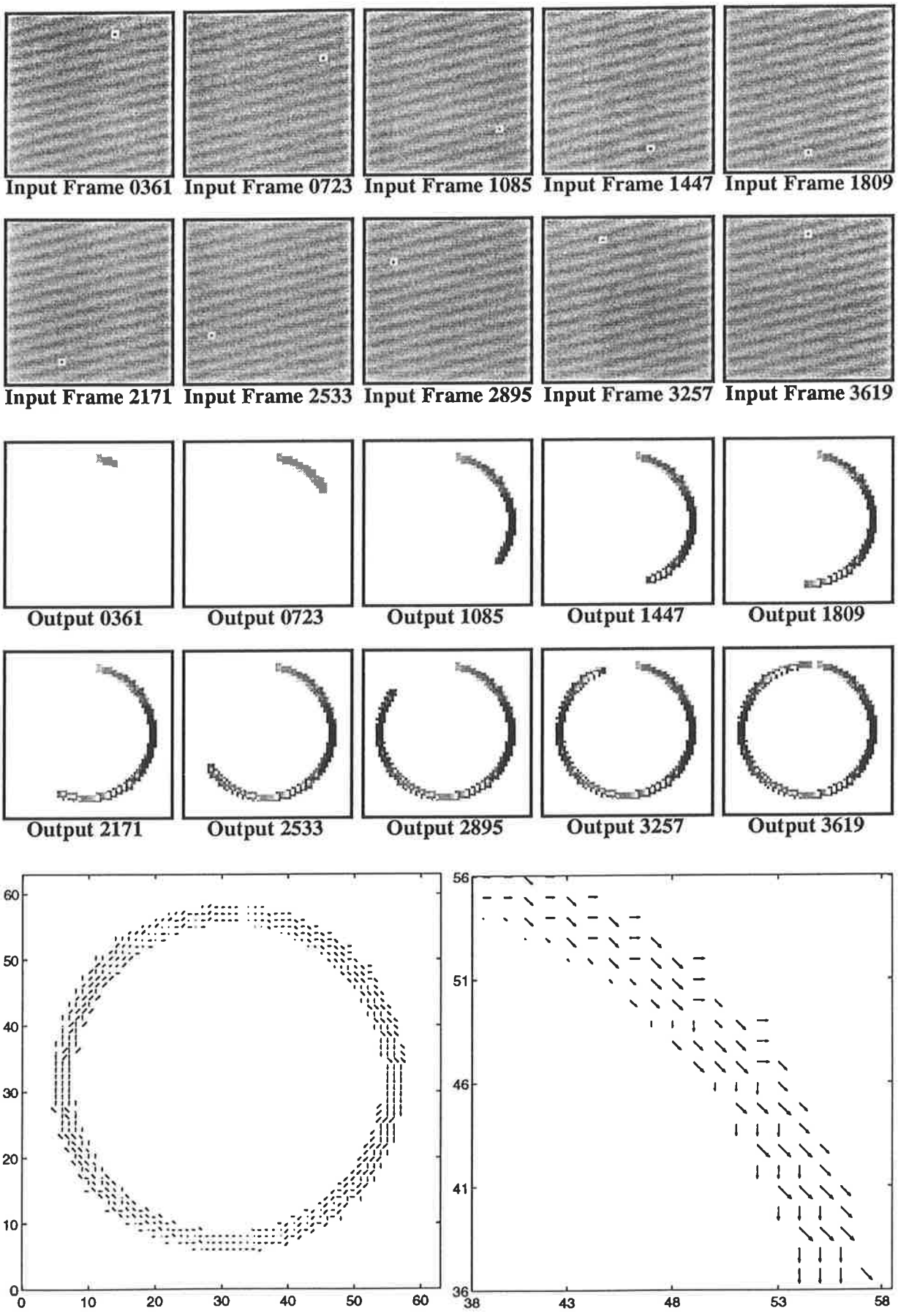


Figure 6.5: An object moving circularly with a varying speed.

6.4 Second Experiment

This experiment determines the influence of noise on the system performance by measuring the output error under different noise conditions. We have carried out twenty four simulations in this experiment which are classified into six groups. In each group, there is a moving object with a given constant speed moving horizontally from left to right. The six selected velocity values are samples from the range of possible velocities the system can detect (from 0 ppf to 1 ppf). In each group, we conducted four different simulations with: 1) noise free input images, 2) input images with $SNR = 22dB$, 3) input images with $SNR = 17dB$, 4) input images with $SNR = 12dB$. One figure is presented for each simulation. Each figure is divided into three parts. Similar to the figures presented in the first experiment, in the top section of each figure ten frames from the input sequence, and in the middle part ten pictures which illustrate the last average velocity of the pixels, are presented. However, in the bottom part of each figure we present two graphs that indicate the percentage relative error of estimated average velocities. The left graph shows percentage relative error versus frame number calculated for only the pixels which receive a movement in each sampling interval; this is to find out the relation between the system performance versus velocity values. The right graph indicates the error versus frame number averaged for all the pixels in the image; this is to measure the overall performance of the system.

To measure the performance of the system in the presence of noise, we employed gaussian white noise with different signal to noise ratios. The signal to noise ratio (*SNR*) is defined as follows

$$SNR = 10 \log_{10} \left(\frac{\sigma_s}{\sigma_n} \right)^2 \tag{6.1}$$

where σ_s is the standard deviation of the image and σ_n is the standard deviation of noise. We made thousands of noise files for three different *SNR* levels, 12 dB, 17 dB, and 22 dB. Then we added the noise information to the input image frames in the image maker program. The results were the noisy image frames used in this experiment.

Simulations 2.1.1-2.1.4: In this group of simulations, the object moves horizontally from left to right with a constant speed of 0.2 ppf. The results of the simulations are presented in

Figures 6.6-6.9. Figure 6.6 illustrates the results for a noise free situation, and Figures 6.7-6.9 illustrate the noisy cases with a *SNR* of 22 dB, 17 dB, and 12 dB respectively. We used 263 image frames in these simulations.

Simulations 2.2.1-2.2.4: In this group of simulations, the object moves horizontally from left to right with a speed of 0.06667 ppf. The simulation results under the following conditions are presented in Figures 6.10-6.13 respectively: 1) input images are noise free, 2) input images with 22dB signal to noise ratio, 3) input images with 17dB signal to noise ratio, 4) input images with 12dB signal to noise ratio. We employed 793 image frames in these simulations.

Simulations 2.3.1-2.3.4: Figures 6.14 to 6.17 indicate the results of the third group of simulations. The object possesses a constant speed of 0.04 ppf and the input frames are mixed with different levels of noise as mentioned in the previous cases. We utilized 1323 image frames in these simulations.

Simulations 2.4.1-2.4.4: In this group, the object moves horizontally from left to right with the speed of 0.02857 ppf. The results of the simulations are presented in Figures 6.18-6.21. Figure 6.18 shows the results for a noise free situation, and Figures 6.19-6.21 illustrate the noisy cases with *SNR* of 22 dB, 17 dB, and 12 dB respectively. We used 1853 image frames in these simulations.

Simulations 2.5.1-2.5.4: The object moves with a constant speed of 0.02222 ppf in this group of simulations. Figures 6.22-6.25 show the simulation results under four different noise conditions as described before. We employed 2383 image frames in these simulations.

Simulations 2.6.1-2.6.4: In this group of simulations, the object travels horizontally from left to right with the speed of 0.01818 ppf. The simulation results under the following noise conditions are presented in Figures 6.26-6.29 respectively: 1) input images are noise free, 2) input images with 22dB signal to noise ratio, 3) input images with 17dB signal to noise ratio, 4) input images with 12dB signal to noise ratio. We utilized 2912 image frames in these simulations.

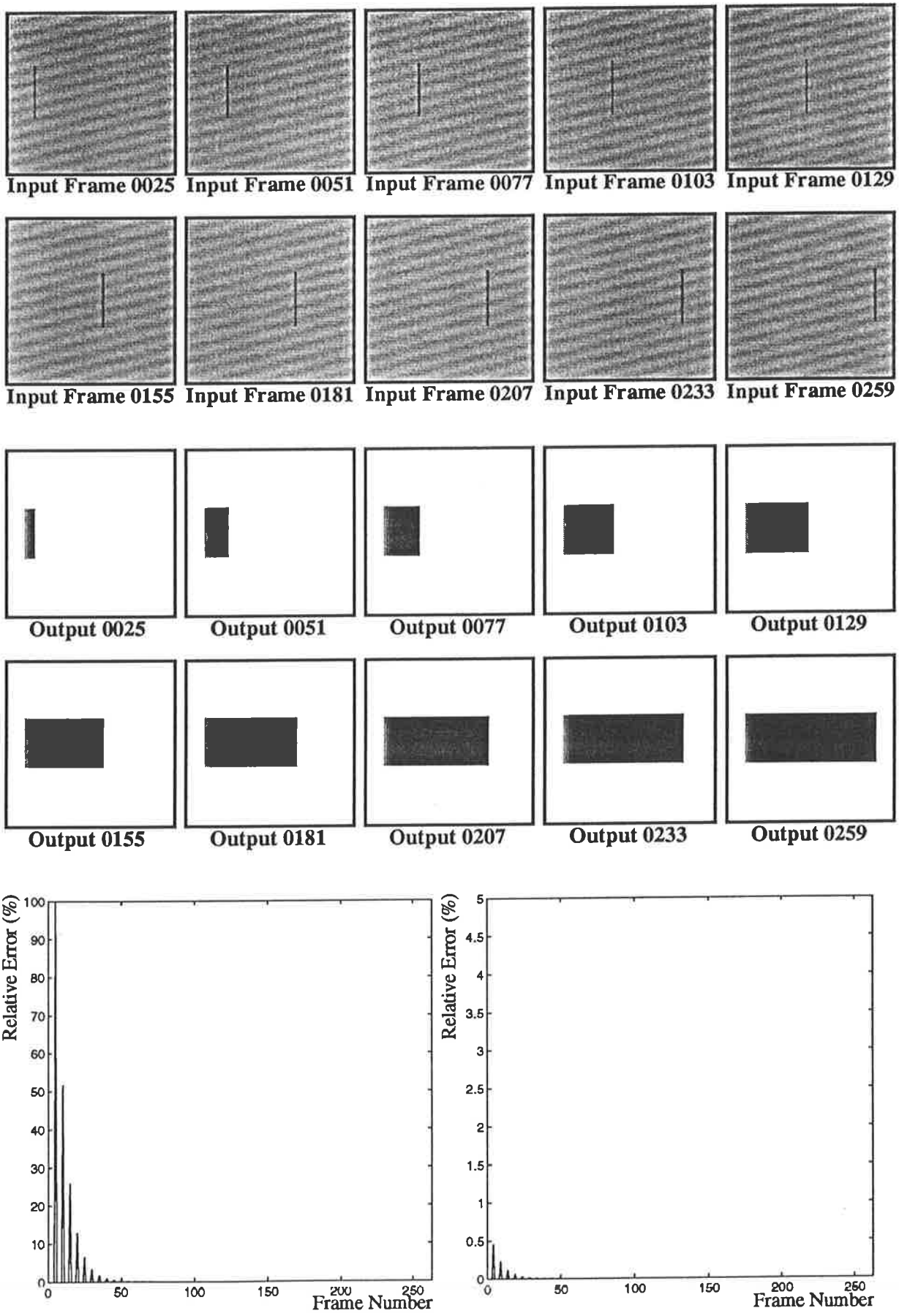


Figure 6.6: Moving object speed = 0.2 ppf. Noise free input information.

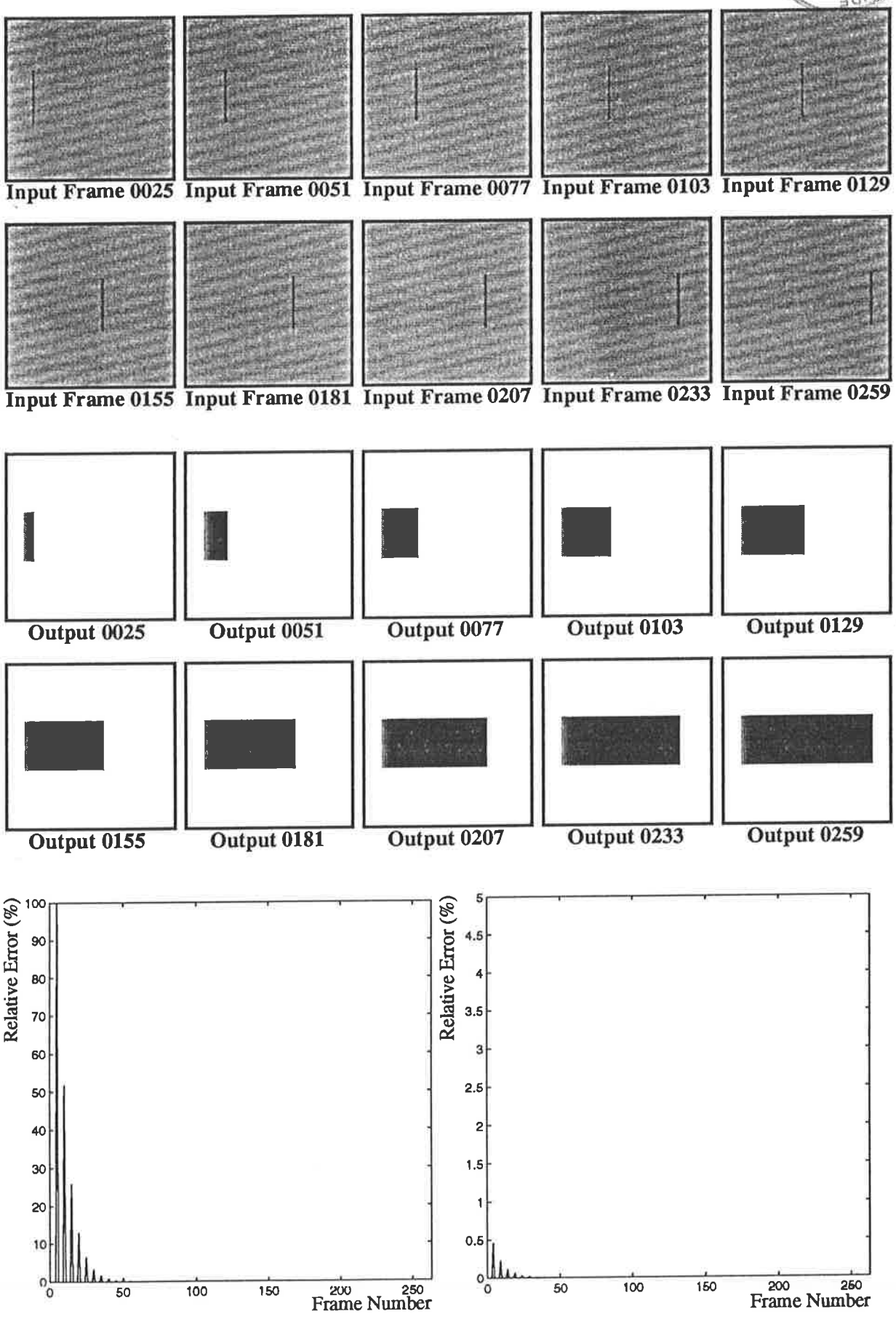
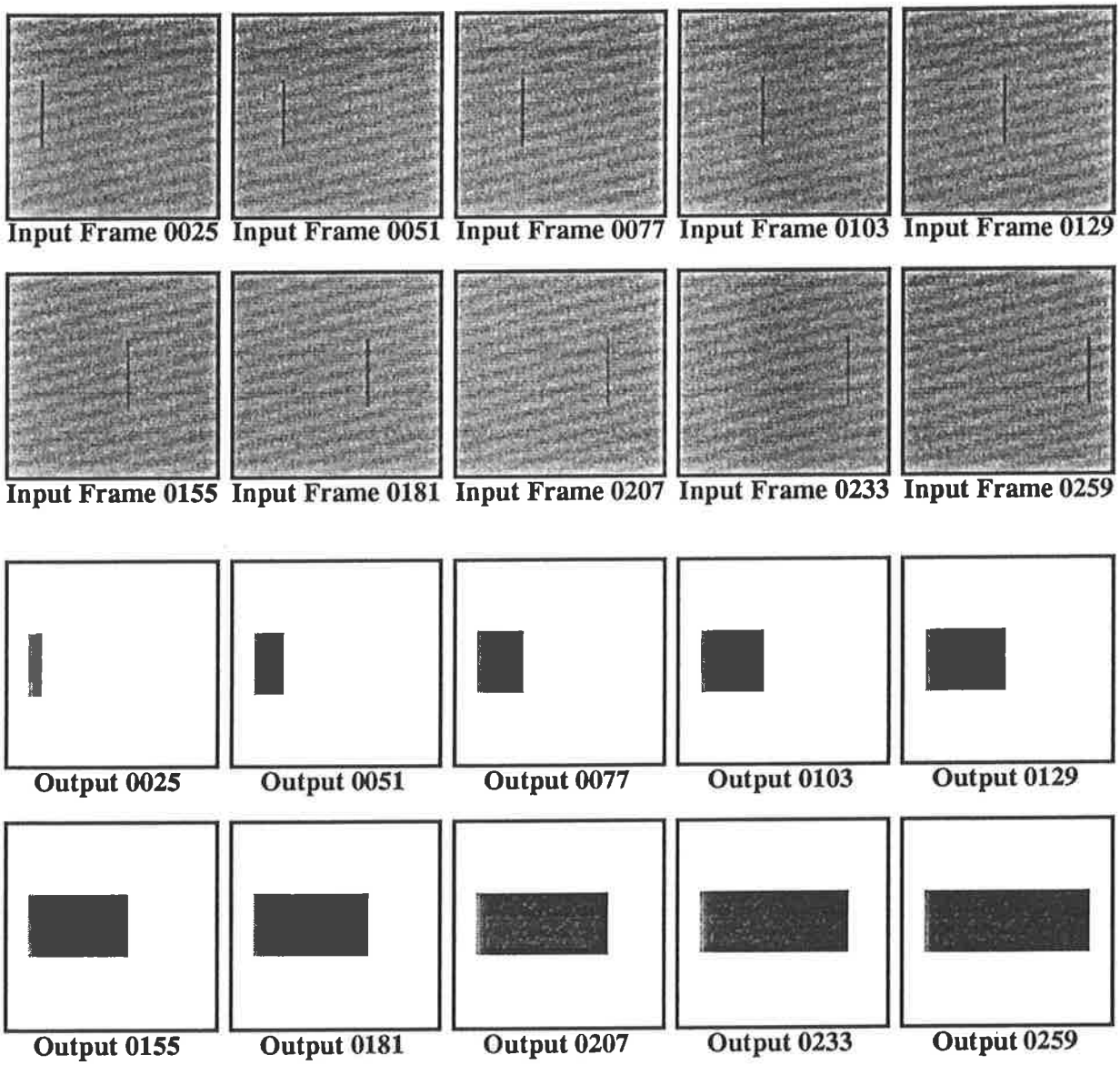


Figure 6.7: Moving object speed = 0.2 ppf. Input information SNR = 22 dB.



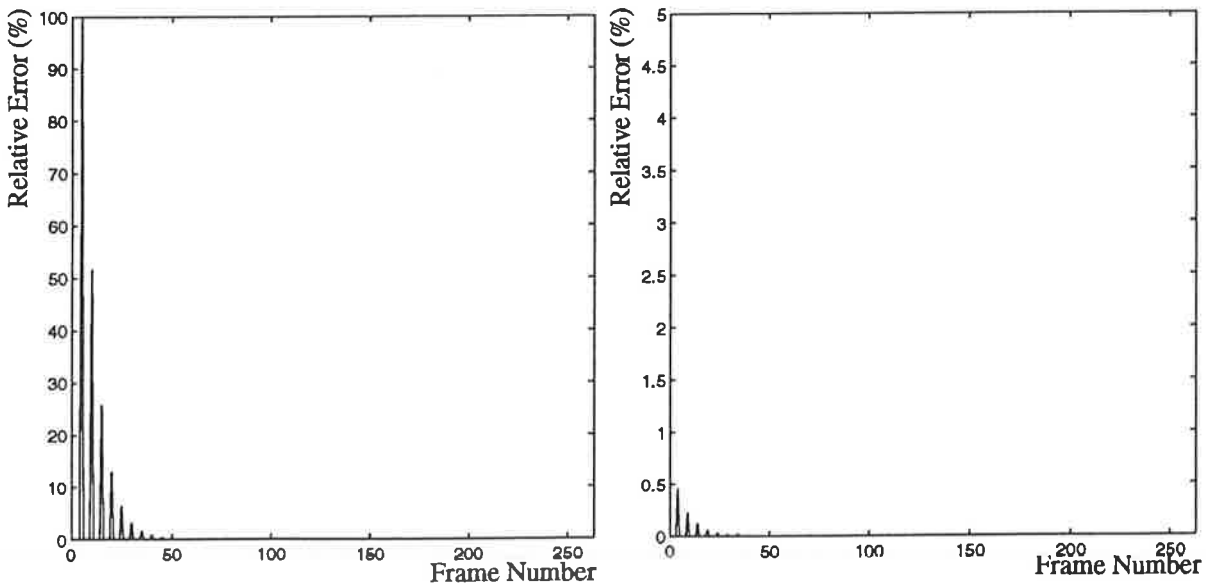


Figure 6.8: Moving object speed = 0.2 ppf. Input information SNR = 17 dB.

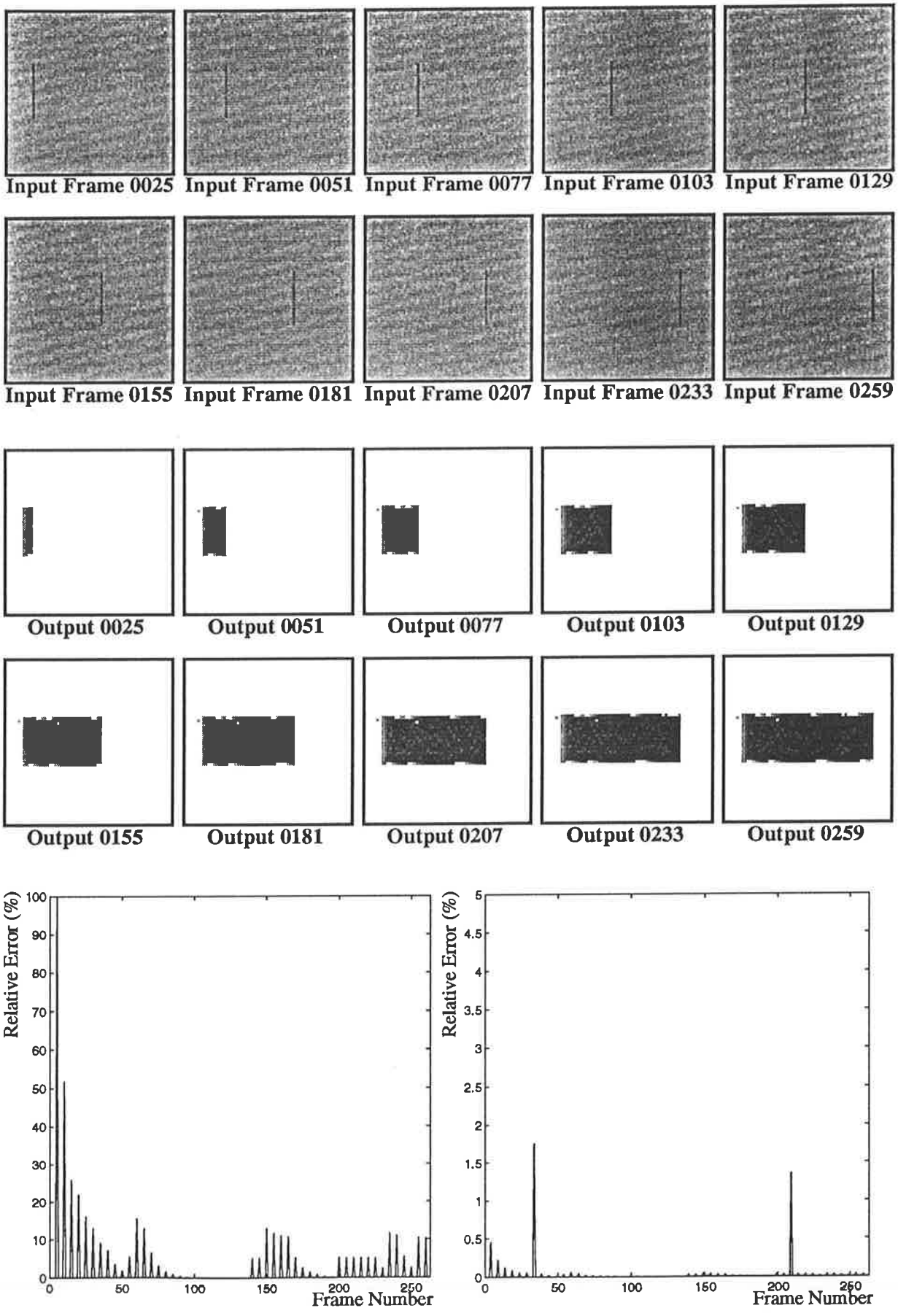
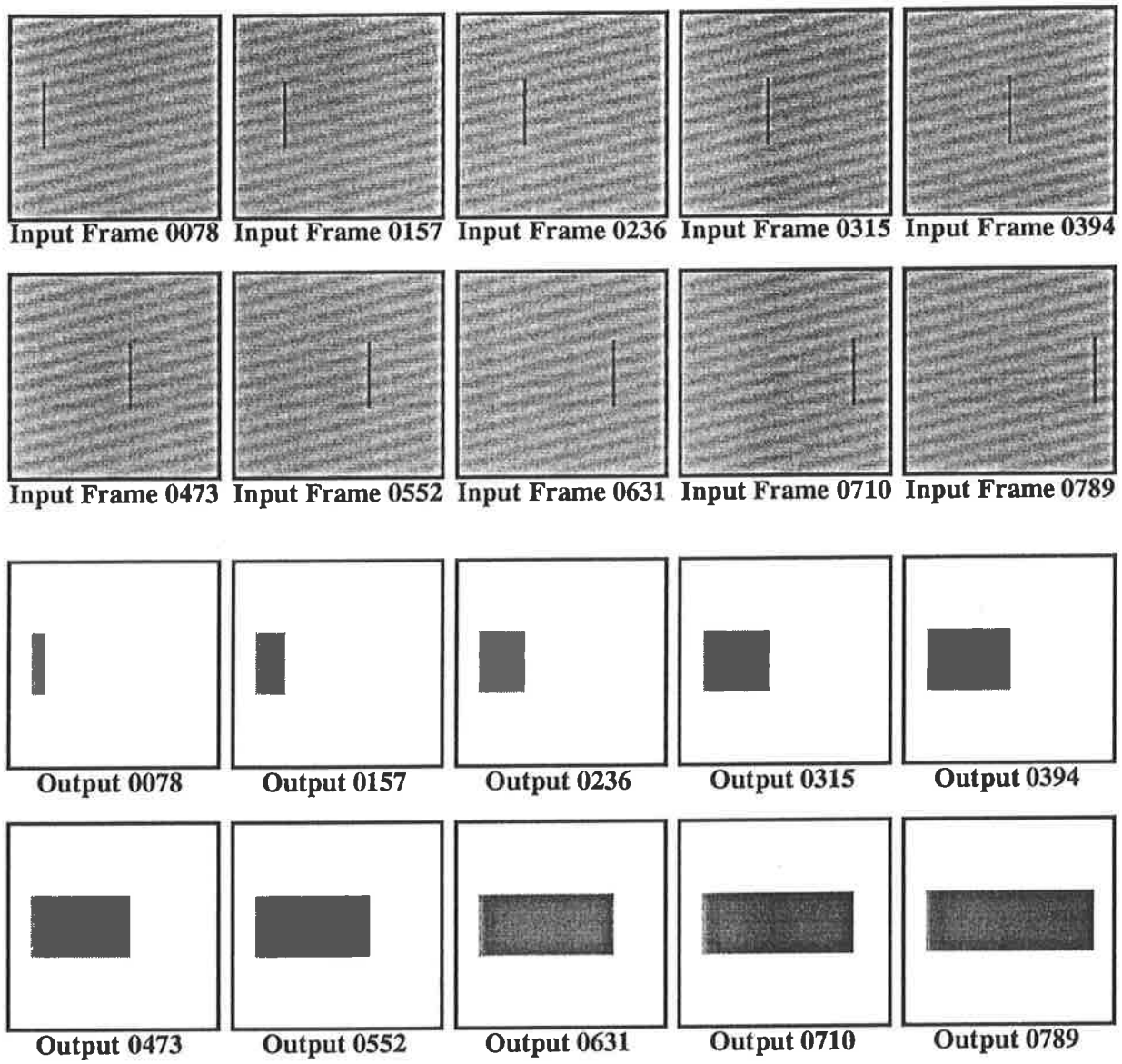


Figure 6.9: Moving object speed = 0.2 ppf. Input information SNR = 12 dB.



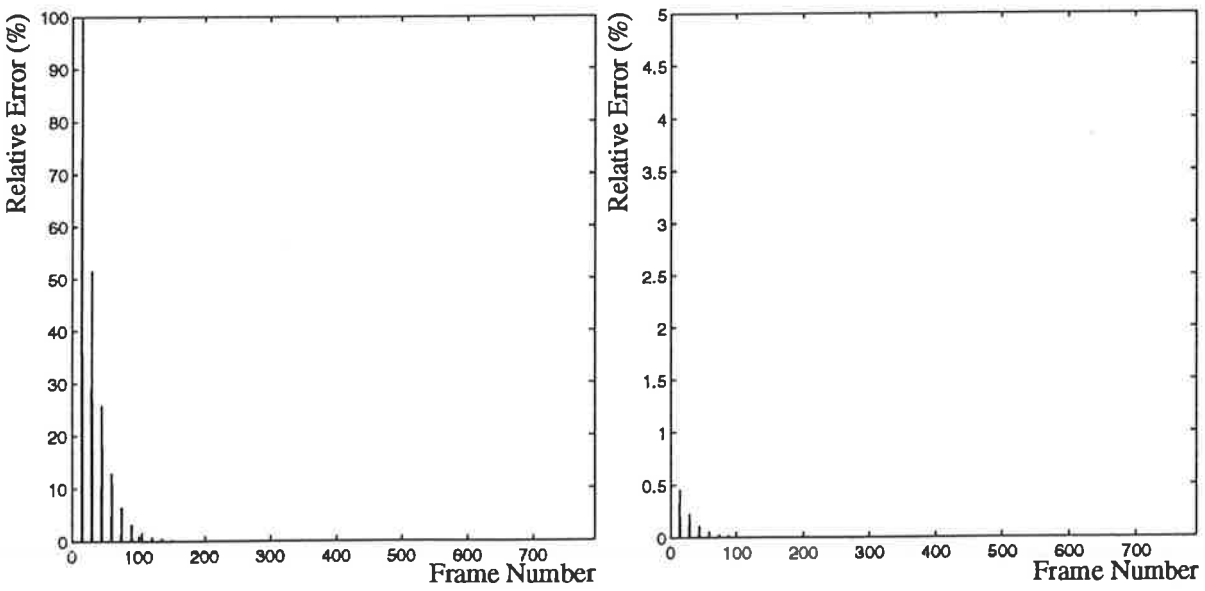


Figure 6.10: Moving object speed = 0.06667 ppf. Noise free input information.

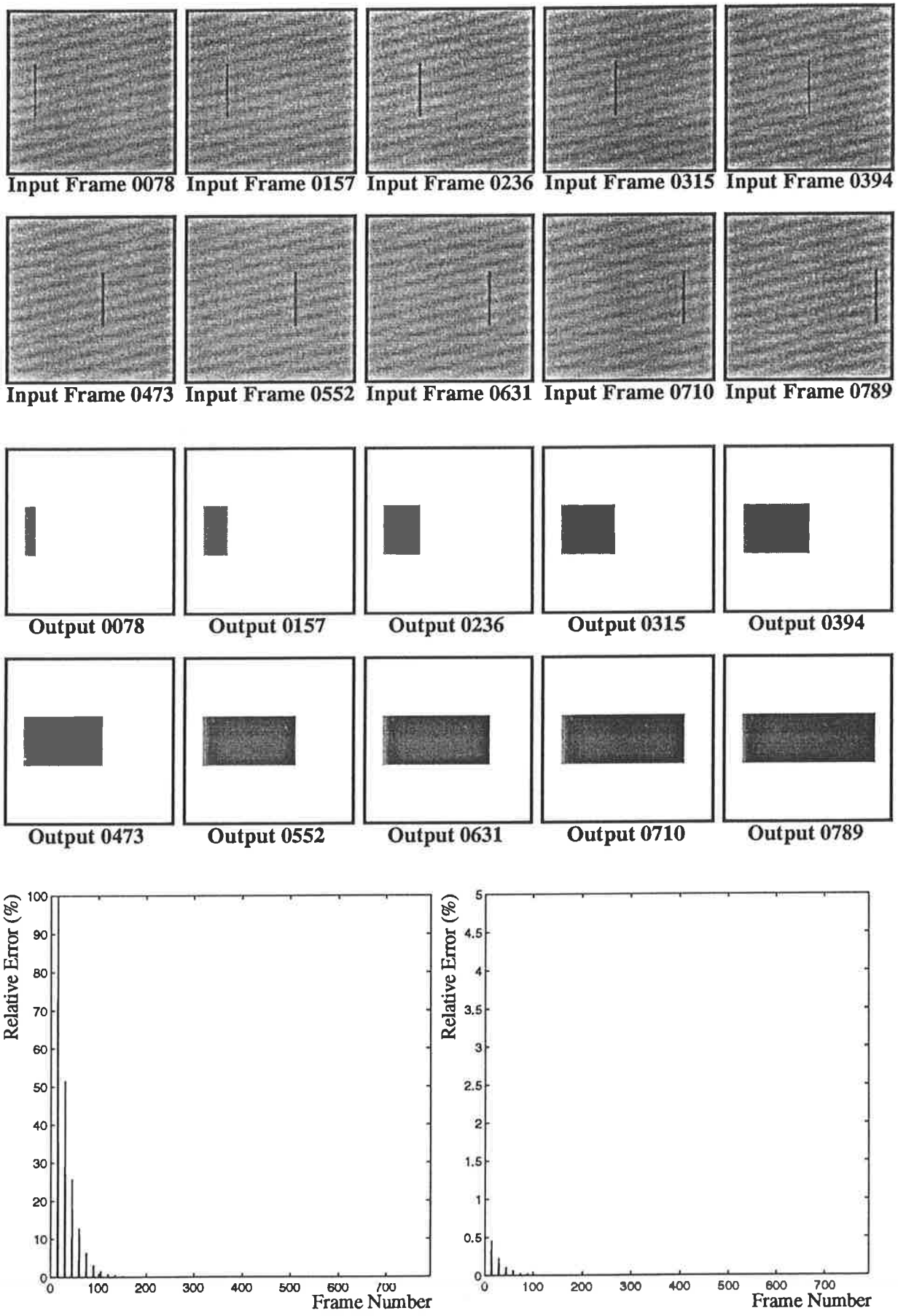
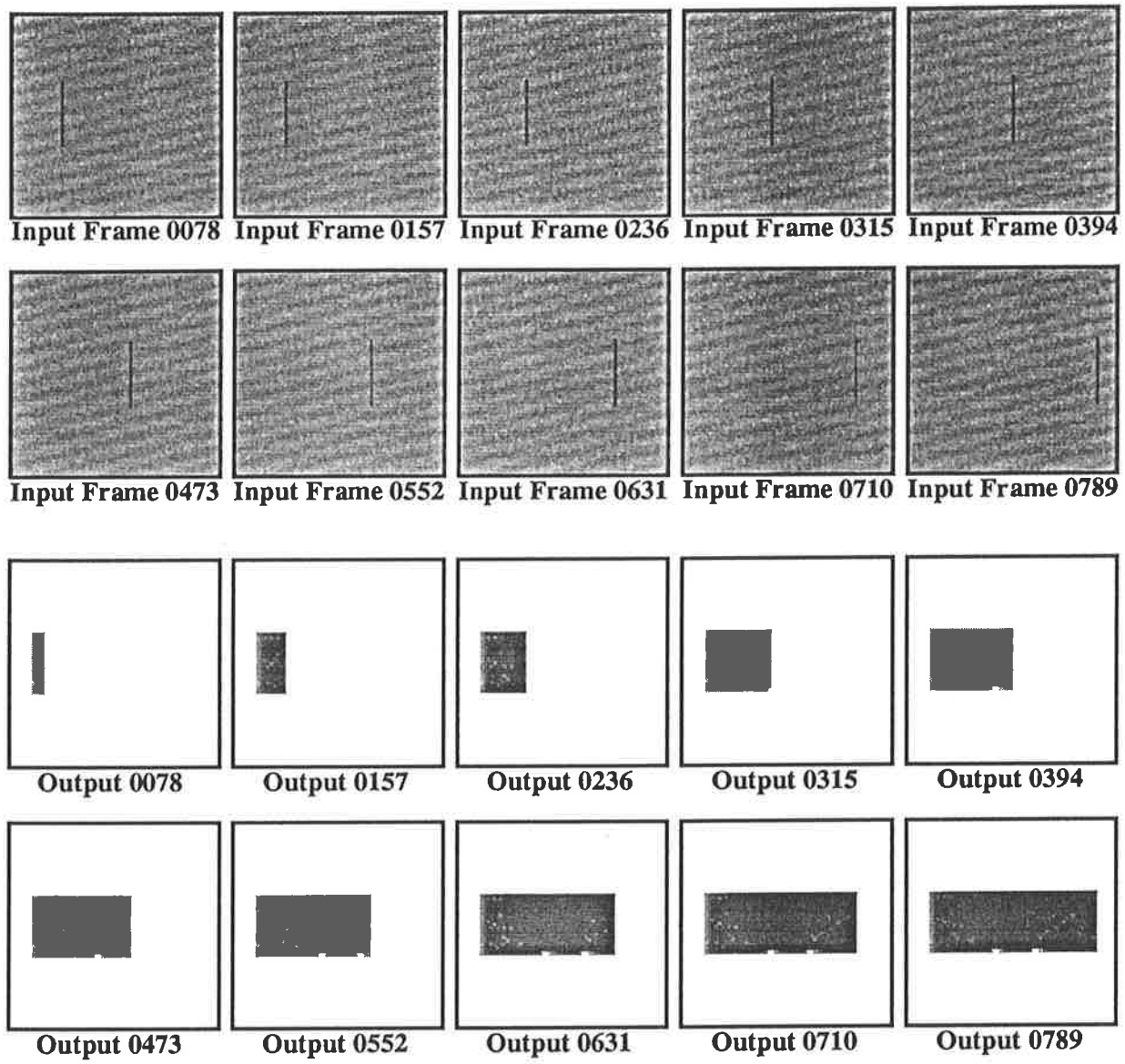


Figure 6.11: Moving object speed = 0.06667 ppf. Input information SNR = 22 dB.



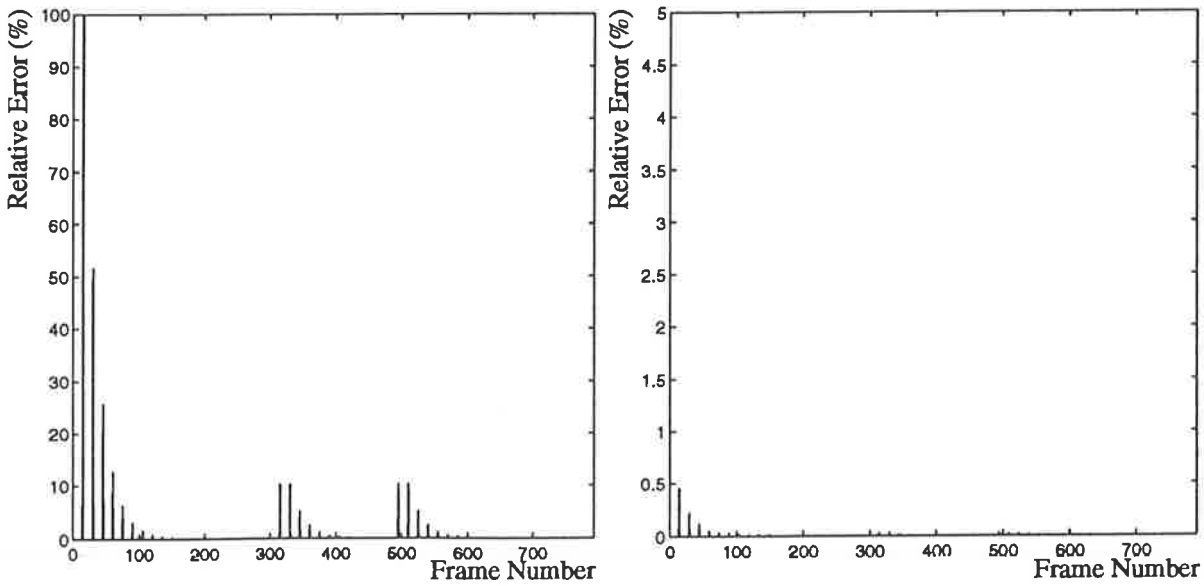


Figure 6.12: Moving object speed = 0.06667 ppf. Input information SNR = 17 dB.

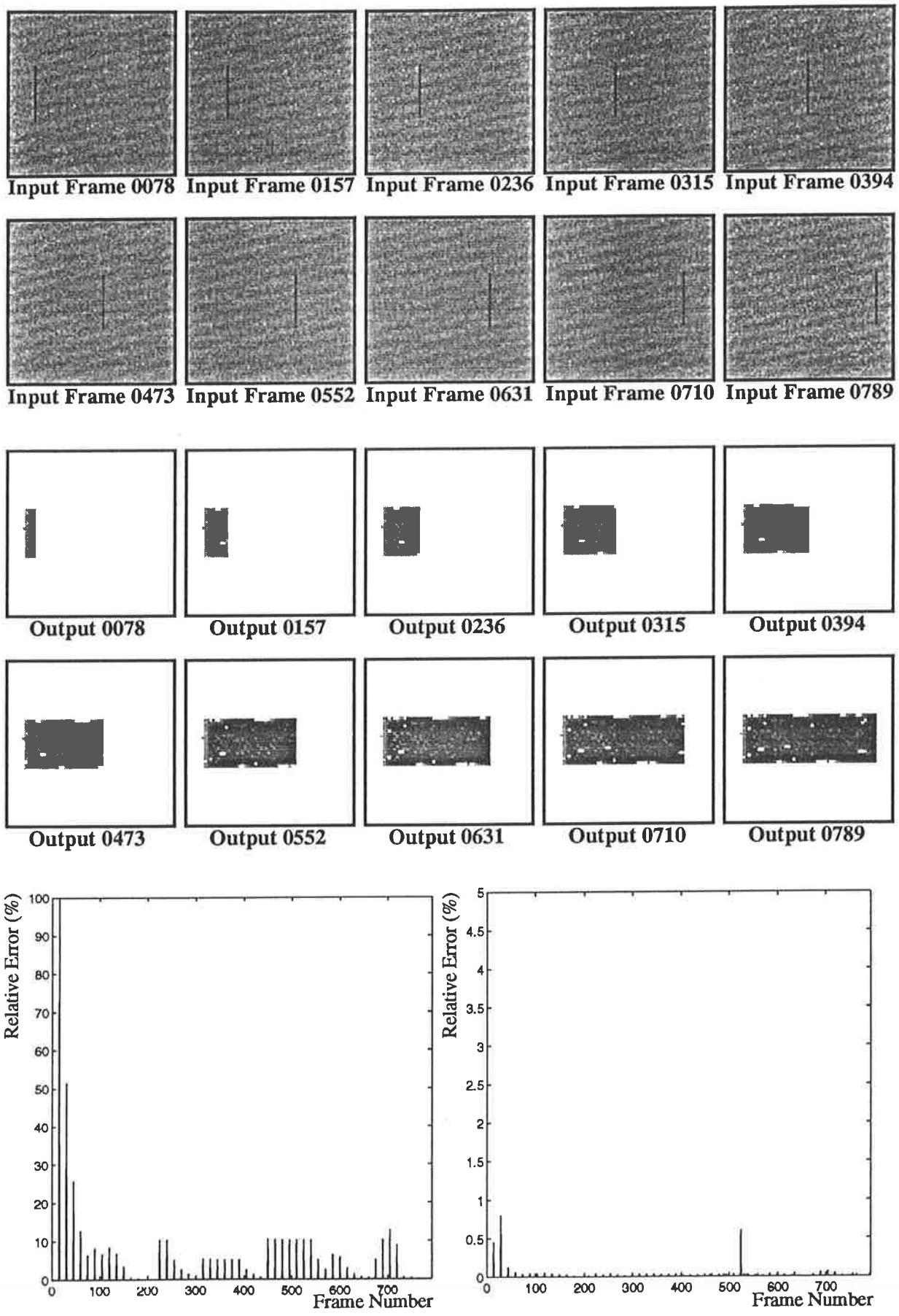


Figure 6.13: Moving object speed = 0.06667 ppf. Input information SNR = 12 dB.

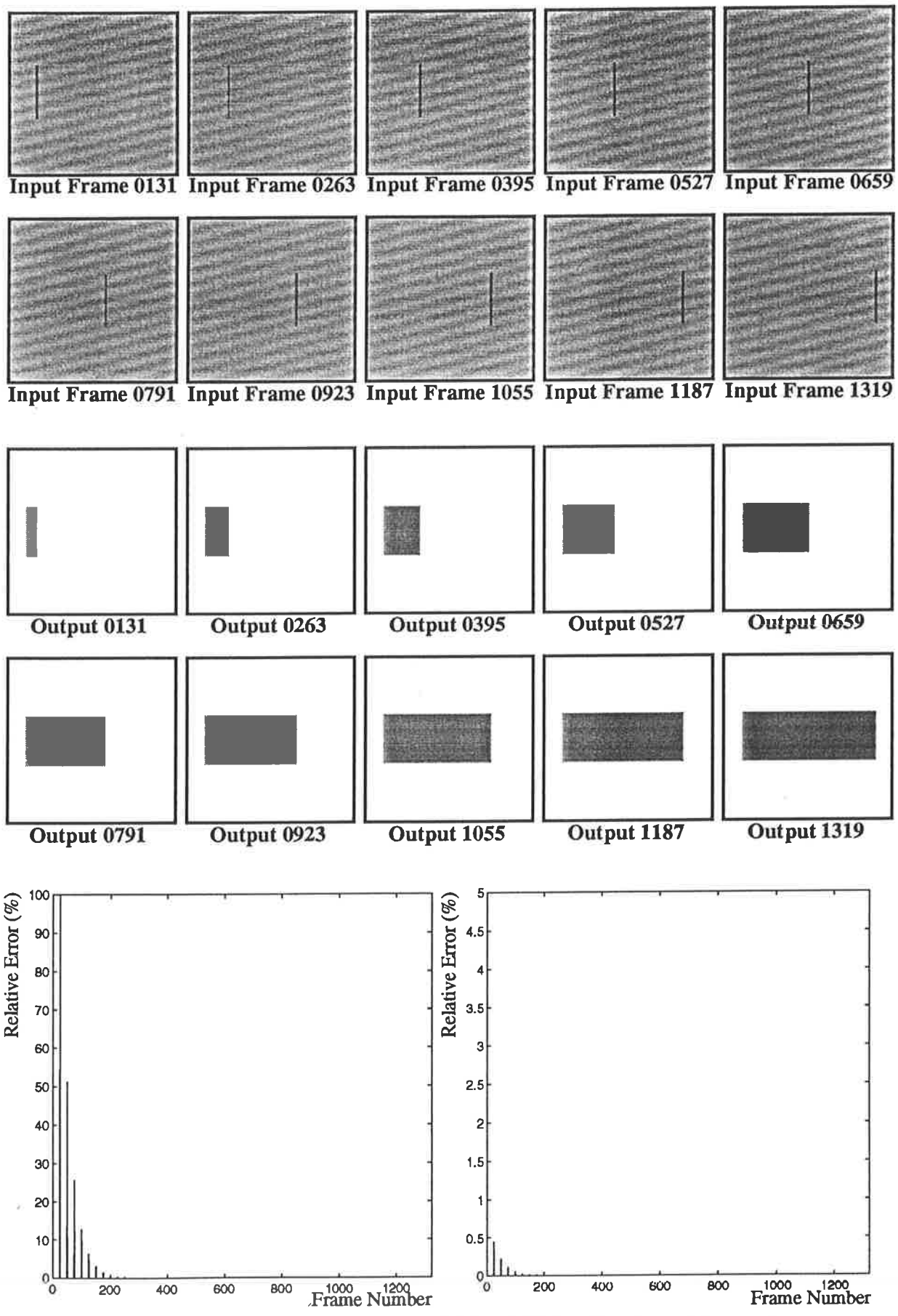


Figure 6.14: Moving object speed = 0.04 ppf. Noise free input information.

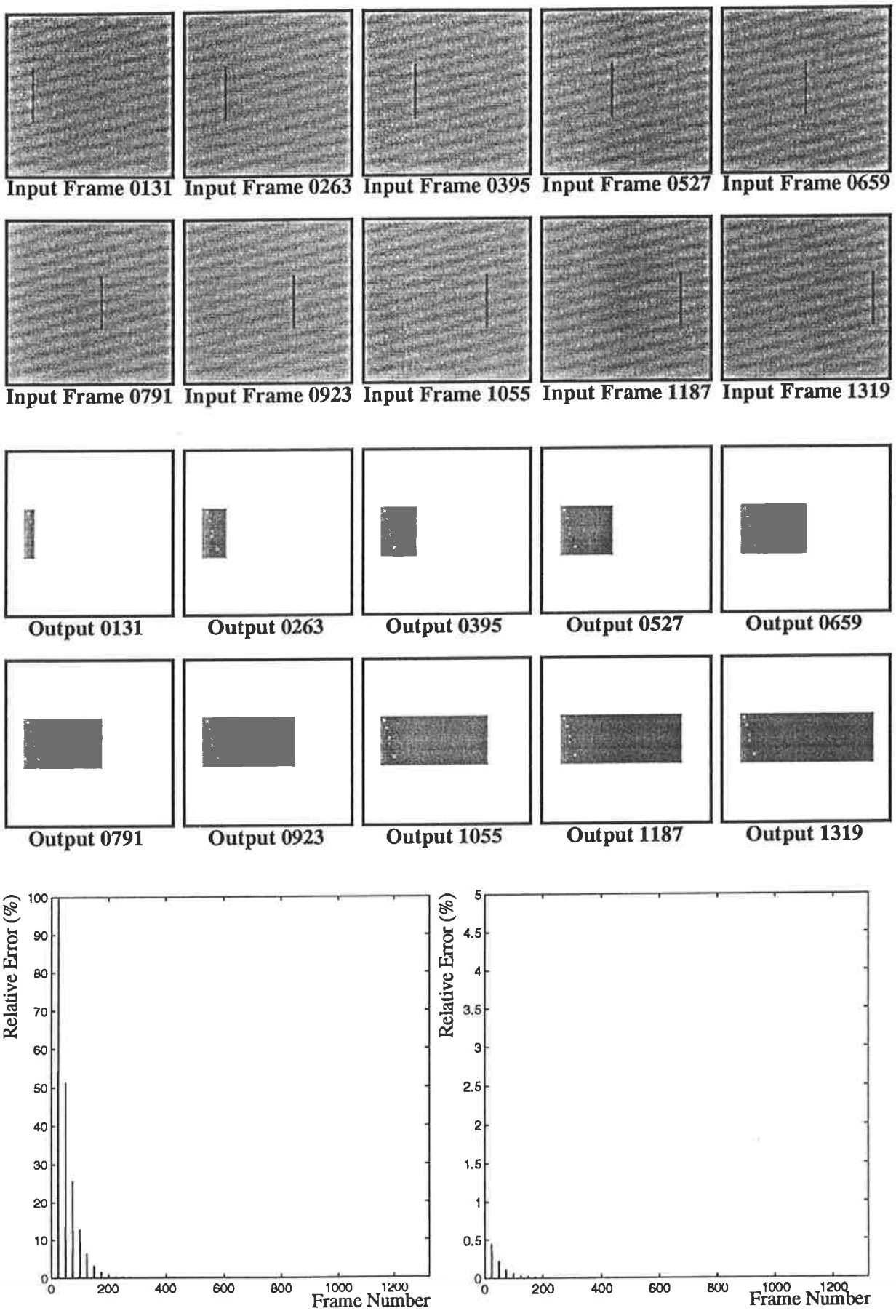


Figure 6.15: Moving object speed = 0.04 ppf. Input information SNR = 22 dB.

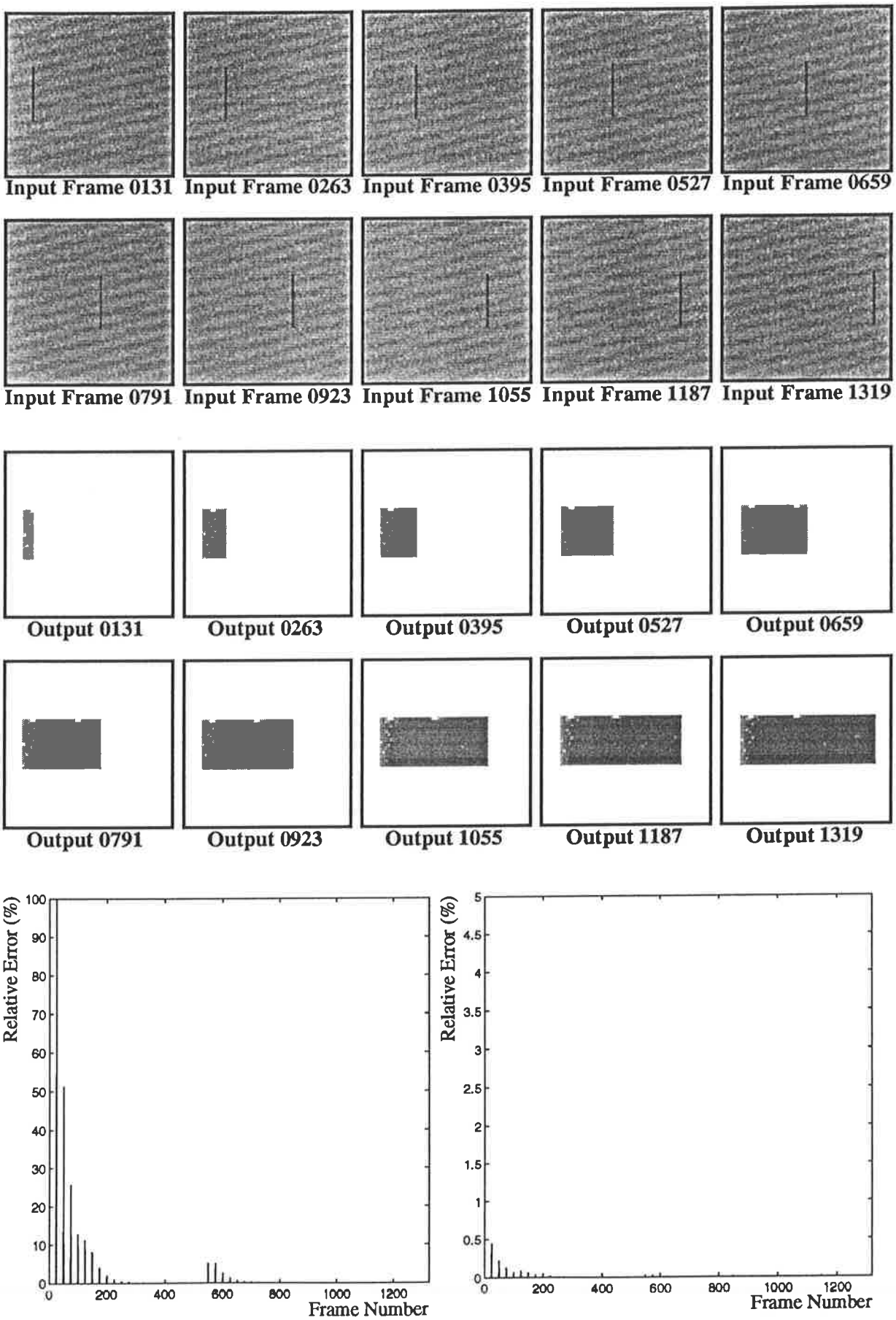


Figure 6.16: Moving object speed = 0.04 ppf. Input information SNR = 17 dB.

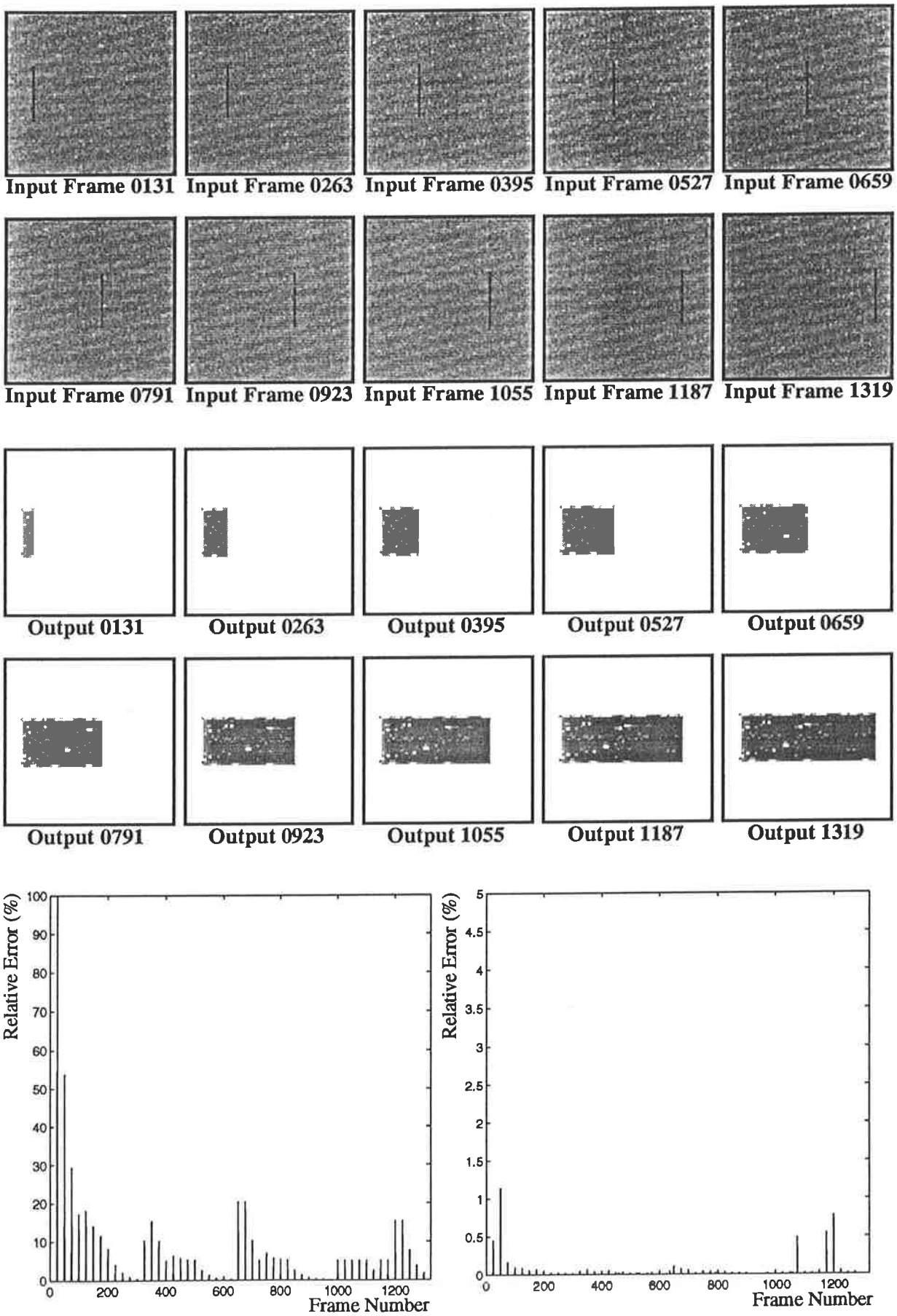


Figure 6.17: Moving object speed = 0.04 ppf. Input information SNR = 12 dB.

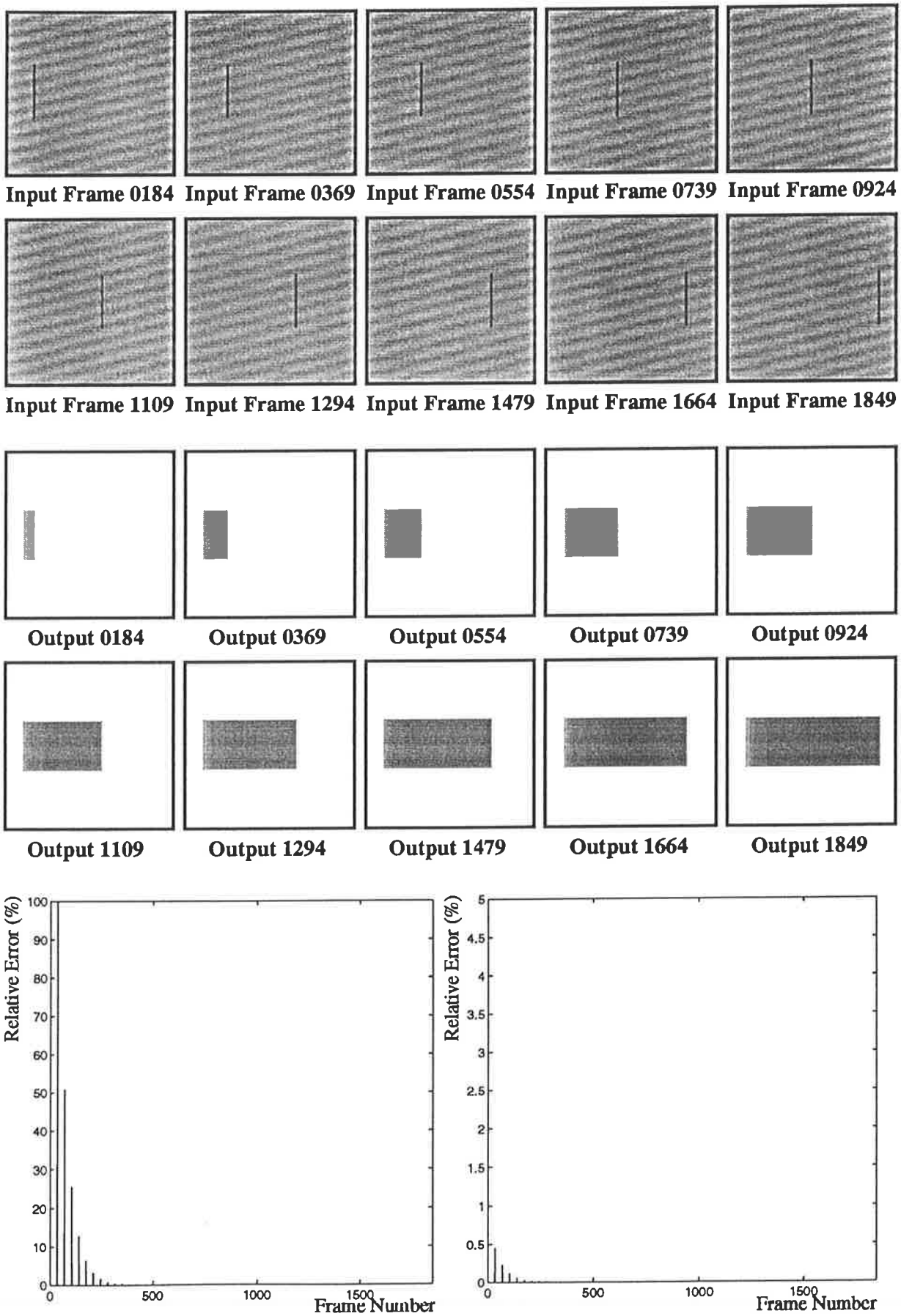
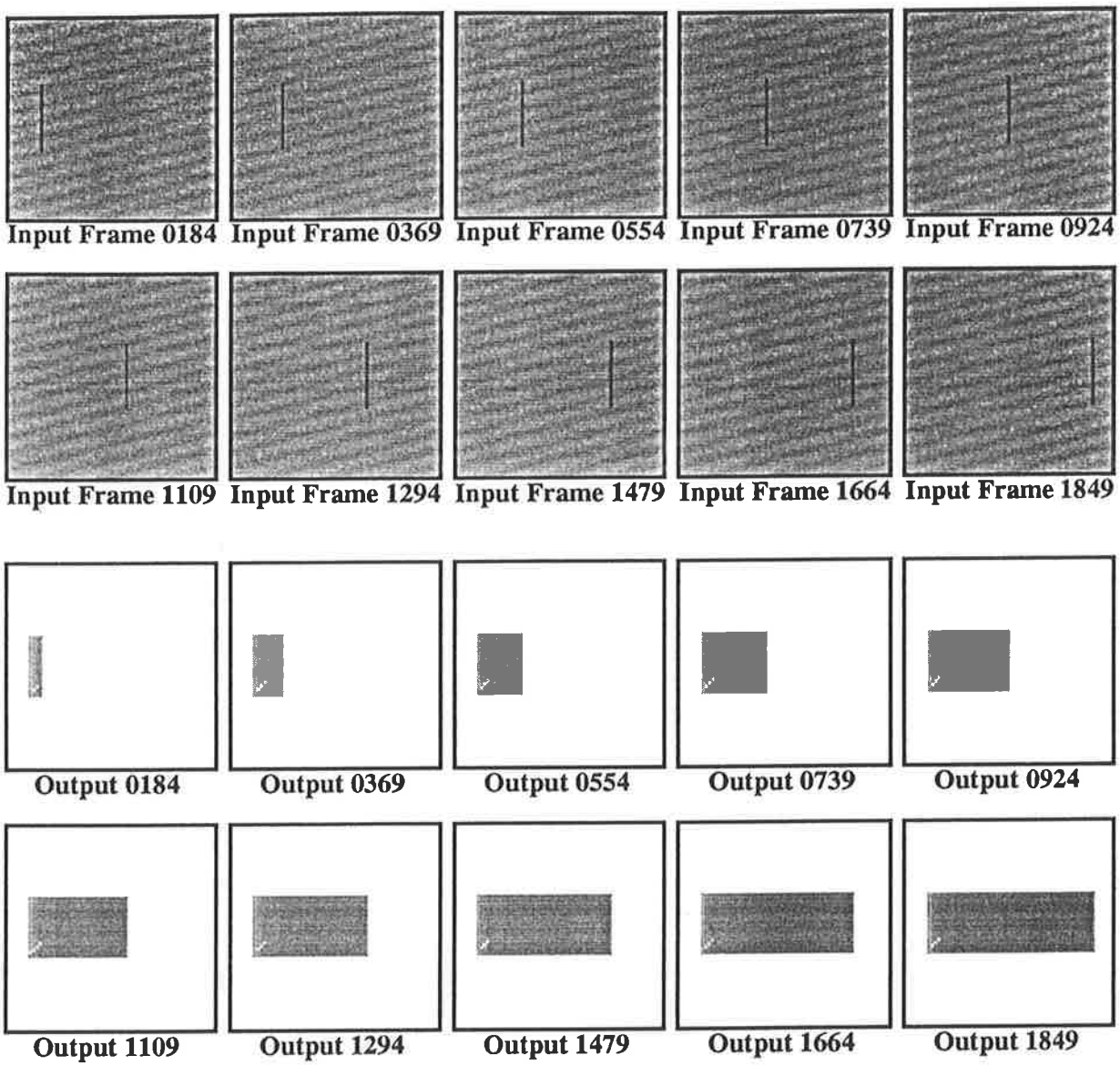


Figure 6.18: Moving object speed = 0.02857 ppf. Noise free input information.



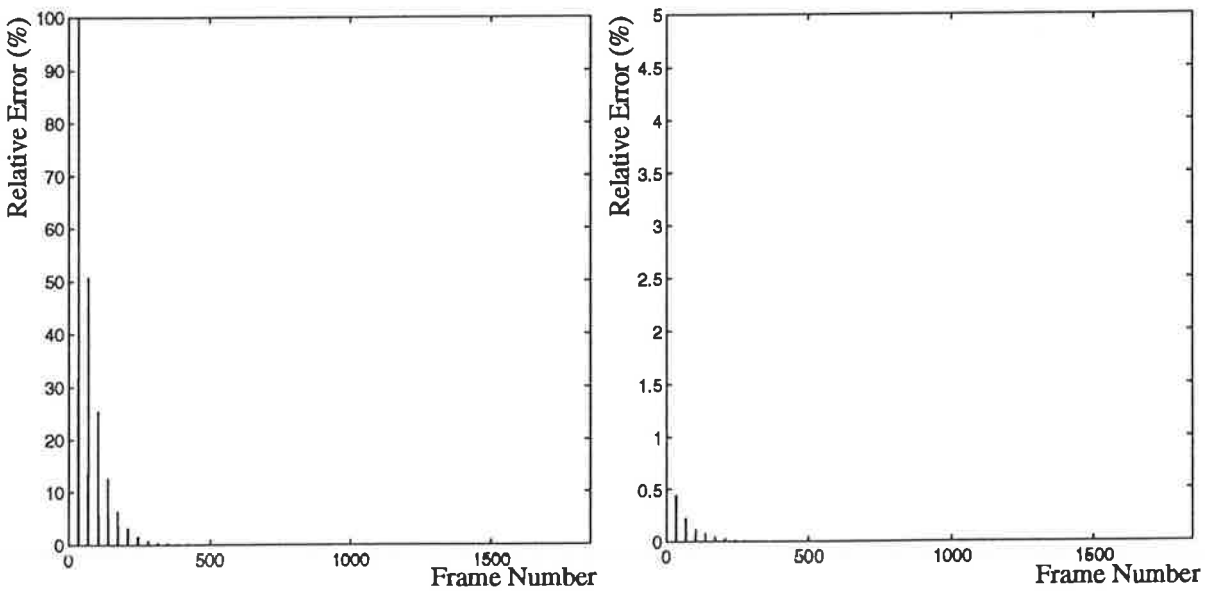
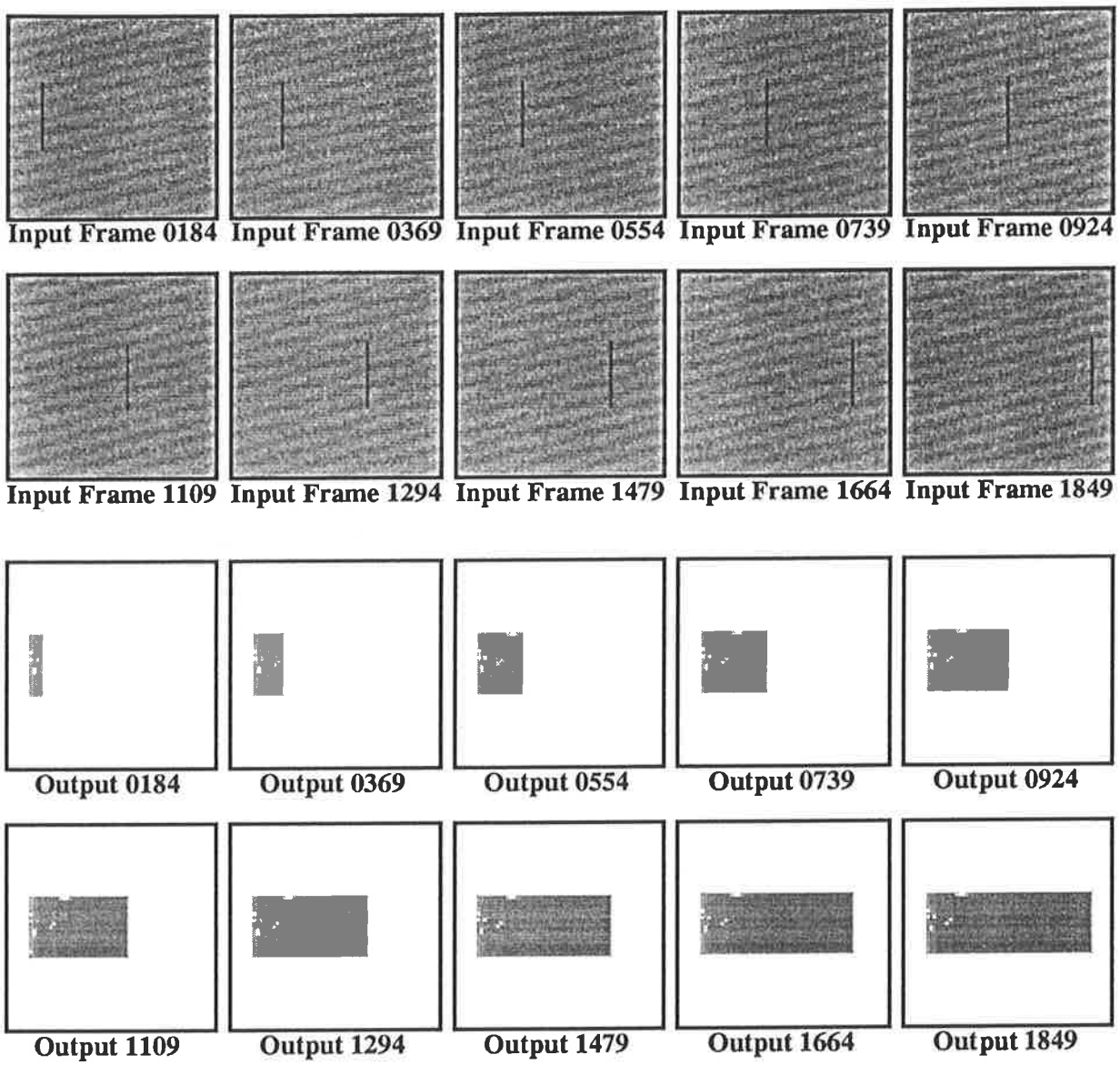


Figure 6.19: Moving object speed = 0.02857 ppf. Input information SNR = 22 dB.



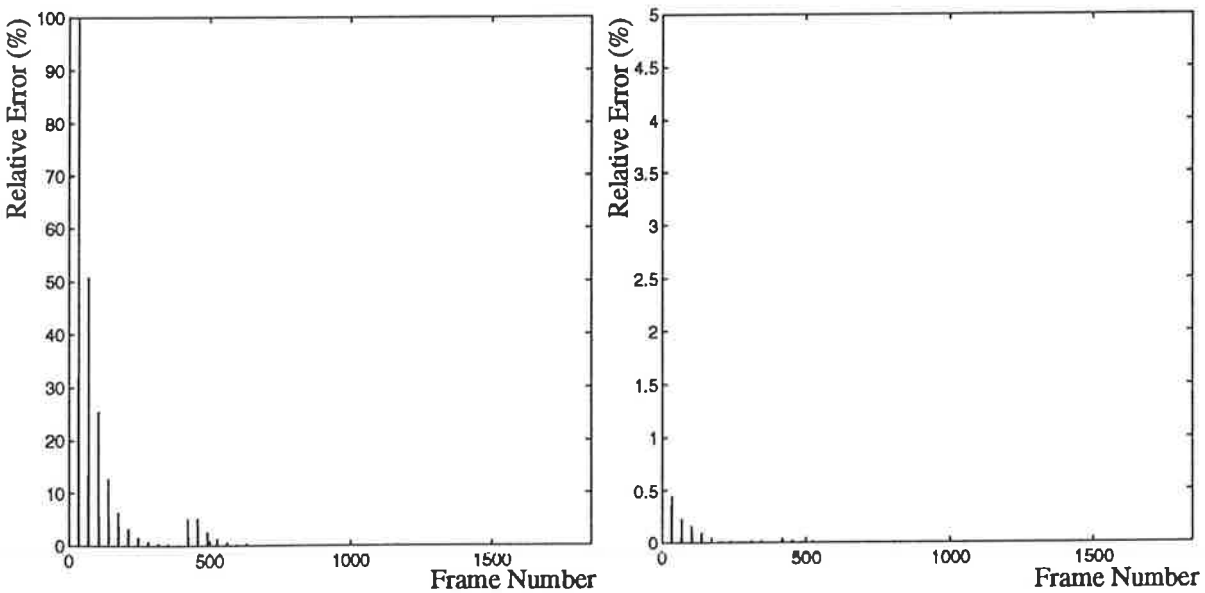
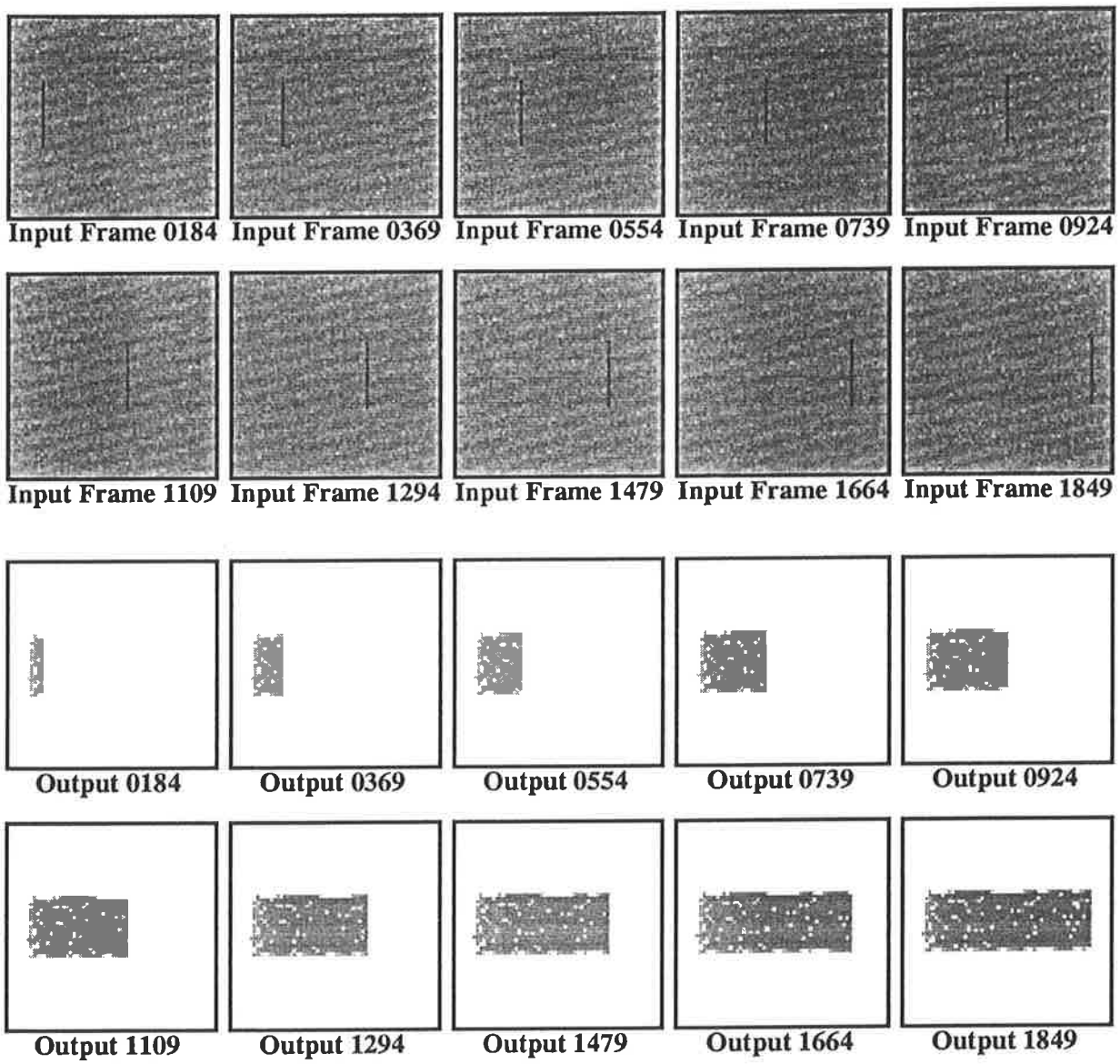


Figure 6.20: Moving object speed = 0.02857 ppf. Input information SNR = 17 dB.



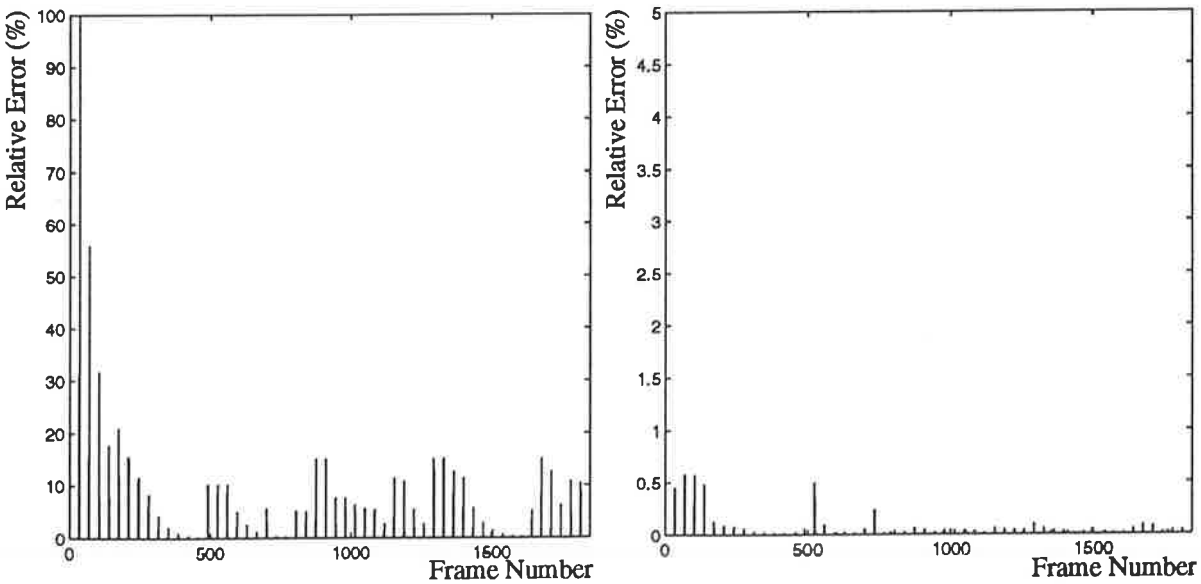


Figure 6.21: Moving object speed = 0.02857 ppf. Input information SNR = 12 dB.

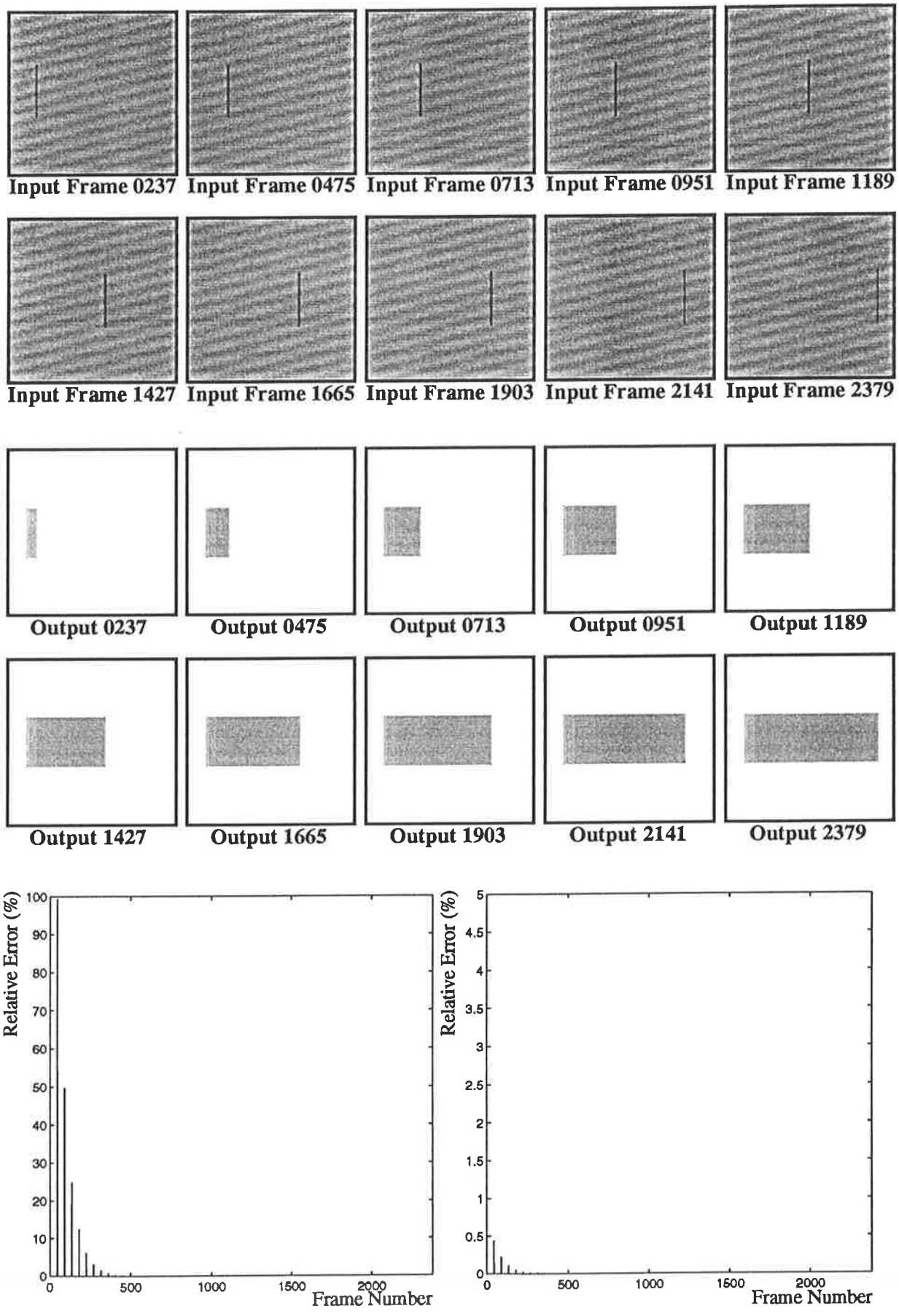


Figure 6.22: Moving object speed = 0.02222 ppf. Noise free input information.

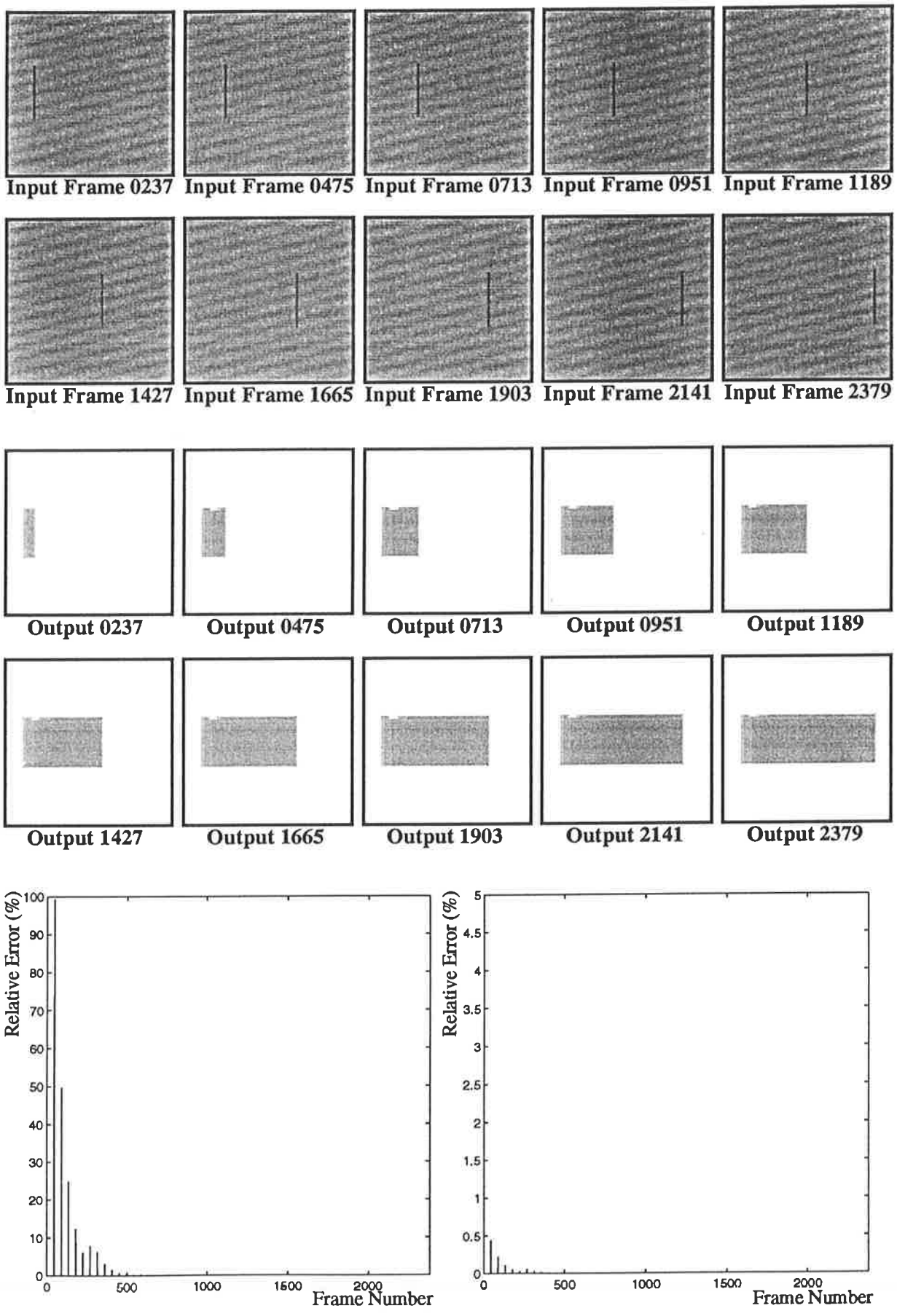


Figure 6.23: Moving object speed = 0.02222 ppf. Input information SNR = 22 dB.

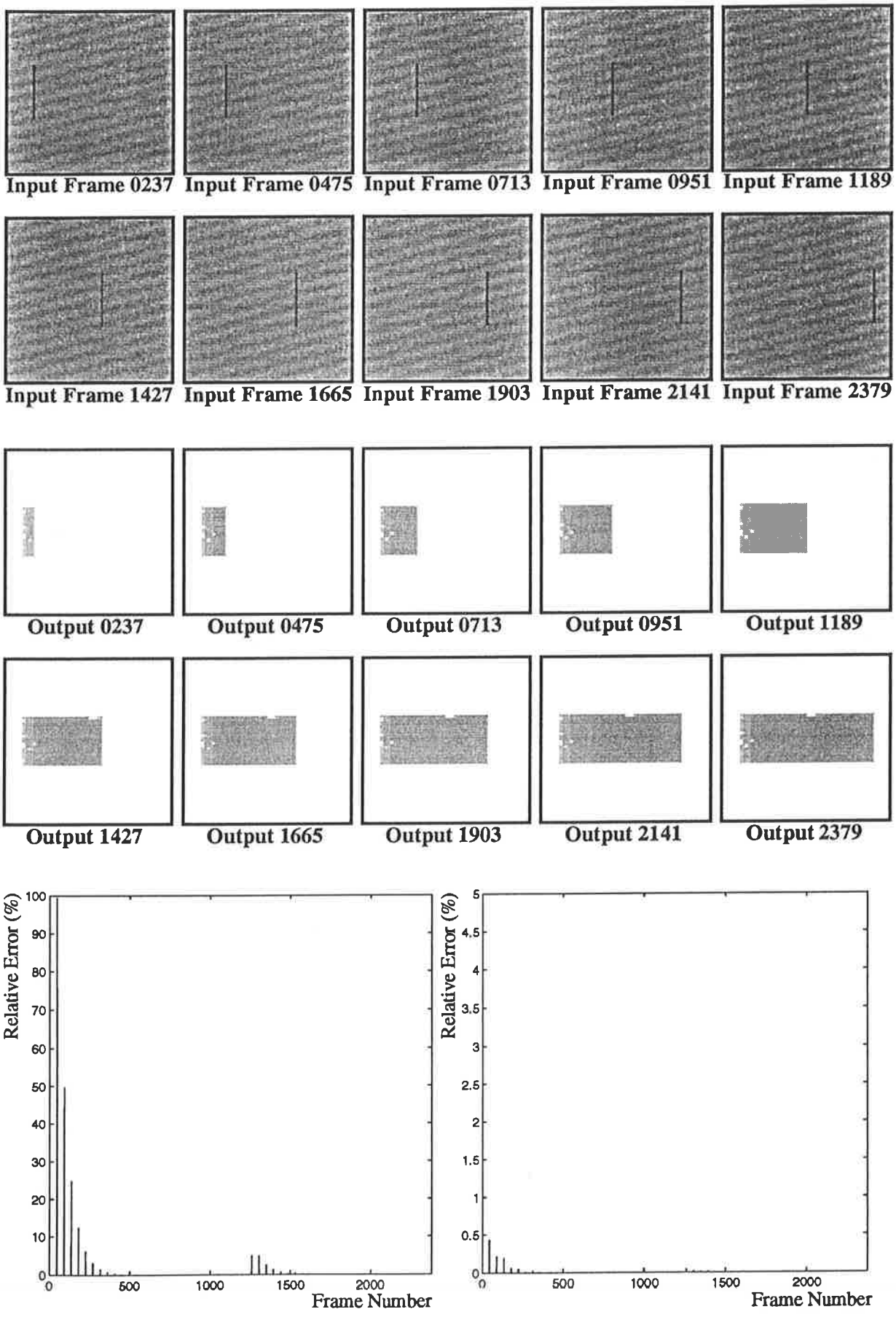


Figure 6.24: Moving object speed = 0.02222 ppf. Input information SNR = 17 dB.

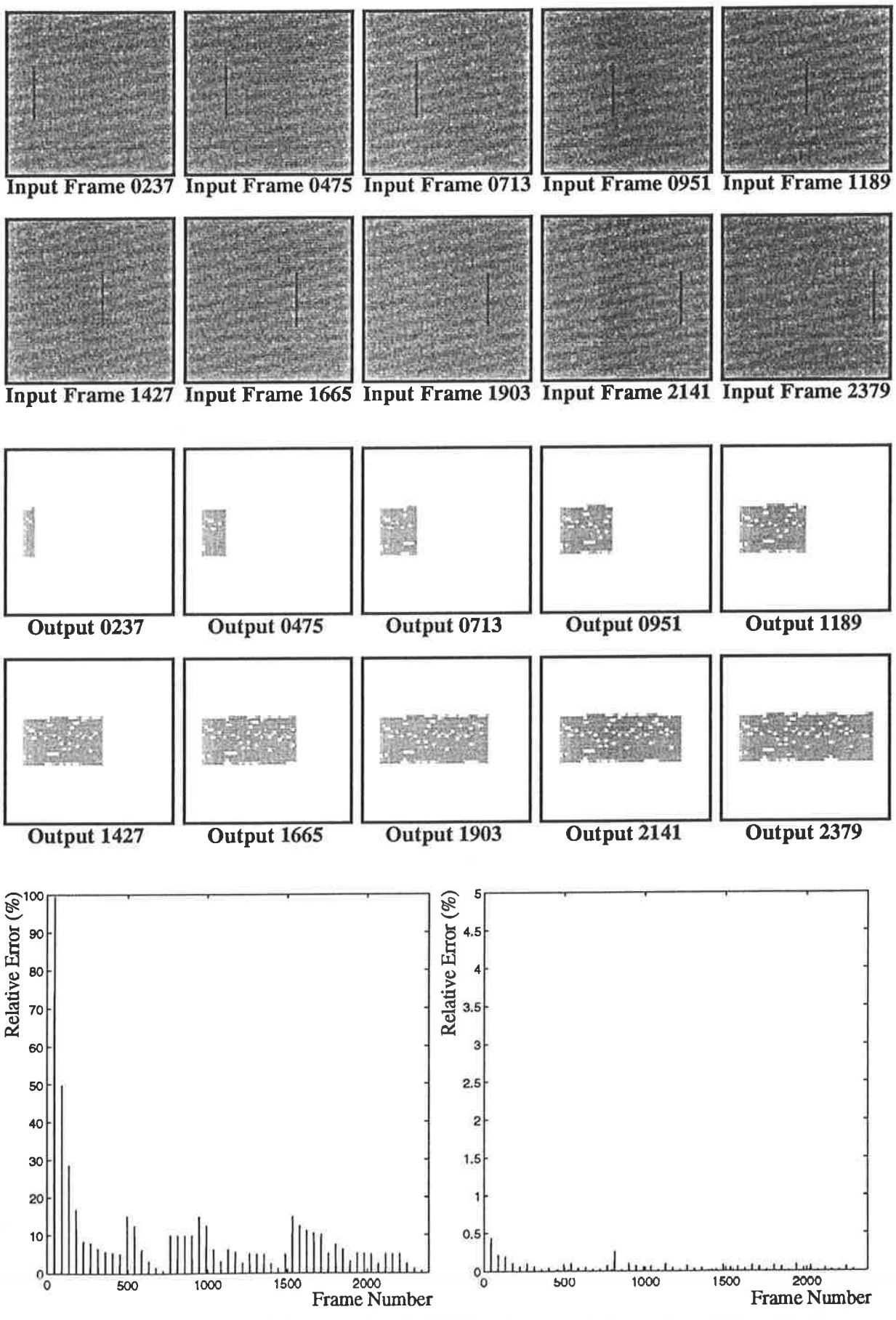


Figure 6.25: Moving object speed = 0.02222 ppf. Input information SNR = 12 dB.

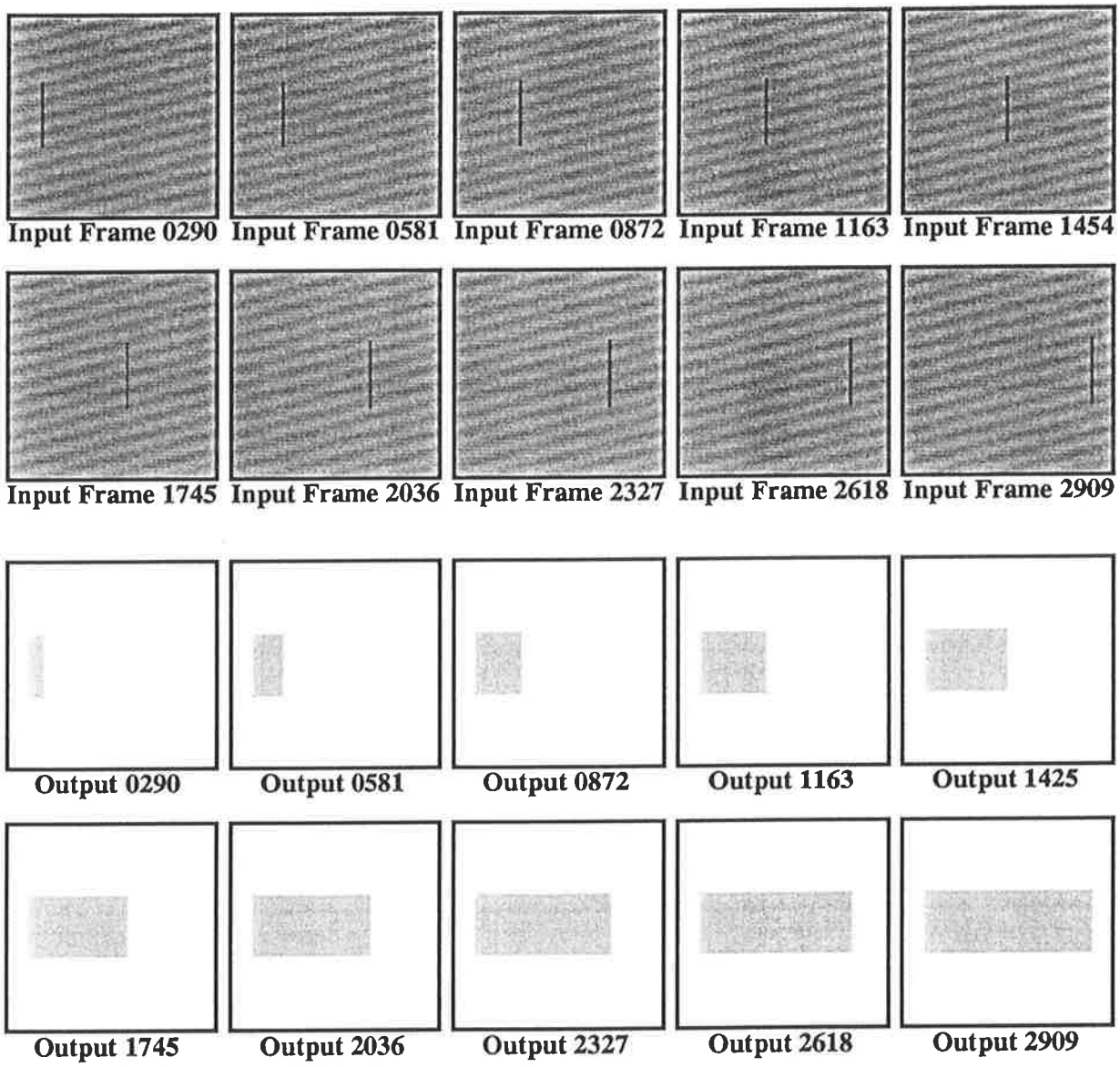


Figure 6.26: Moving object speed = 0.01818 ppf. Noise free input information.

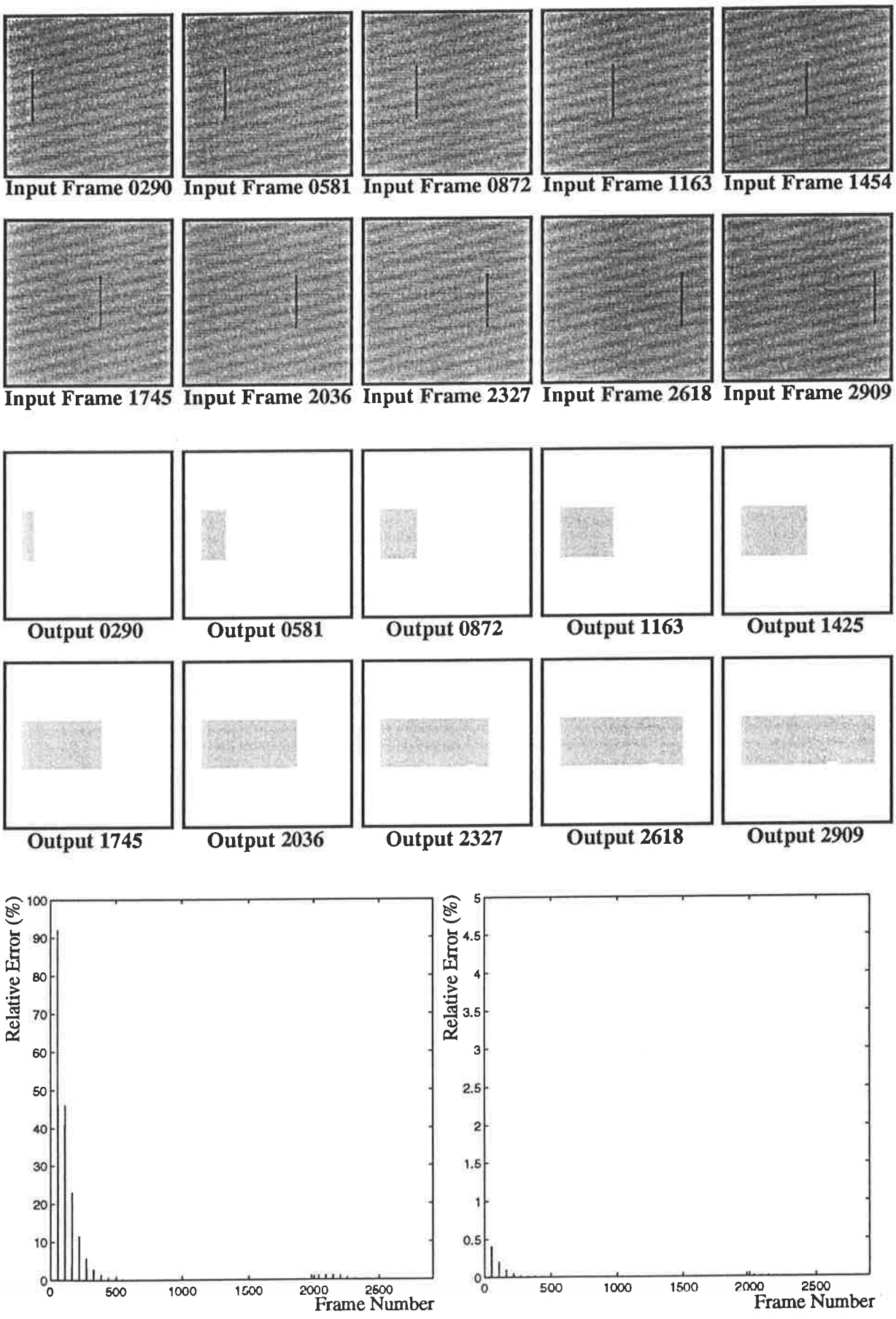


Figure 6.27: Moving object speed = 0.01818 ppf. Input information SNR = 22 dB.

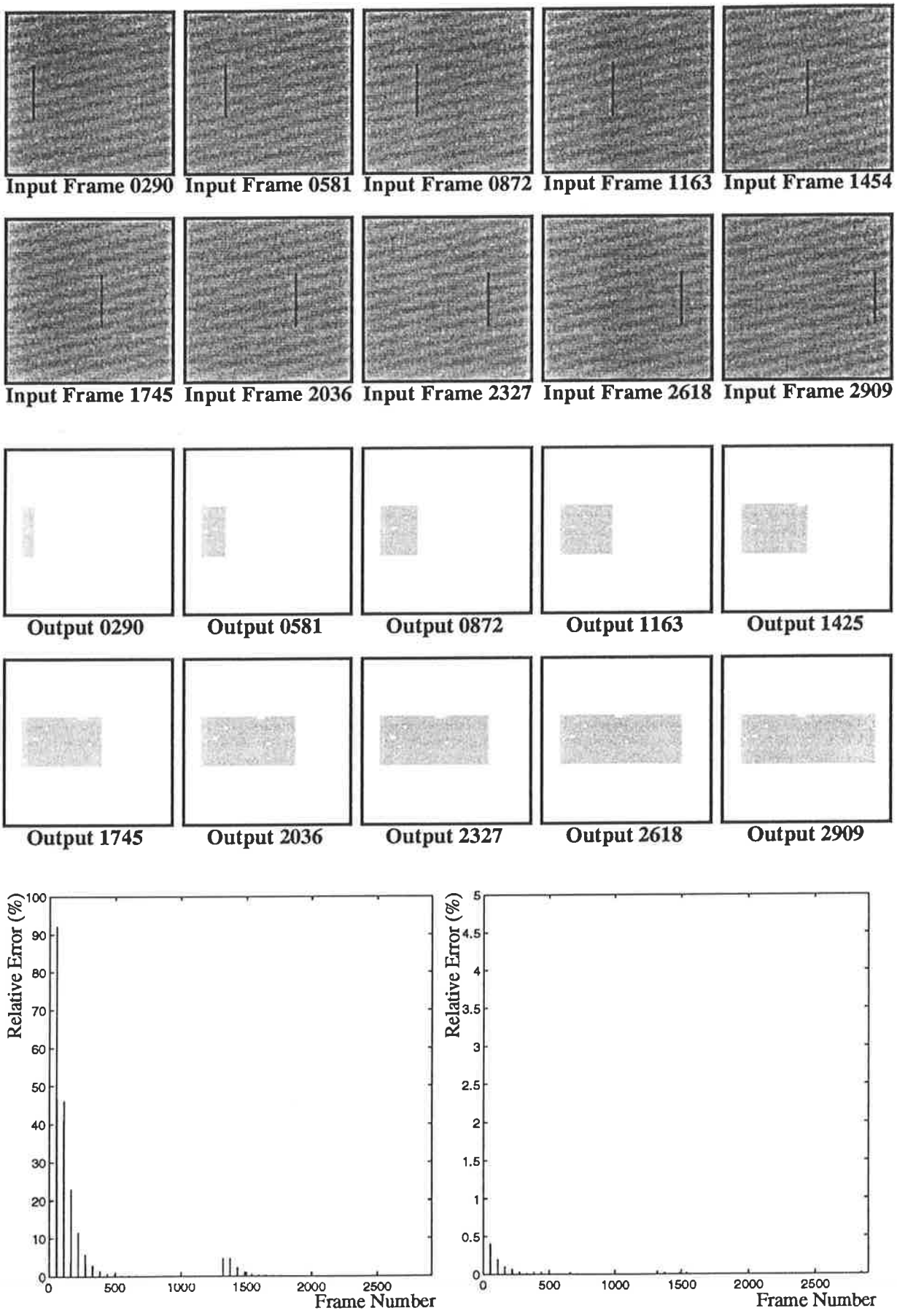


Figure 6.28: Moving object speed = 0.01818 ppf. Input information SNR = 17 dB.

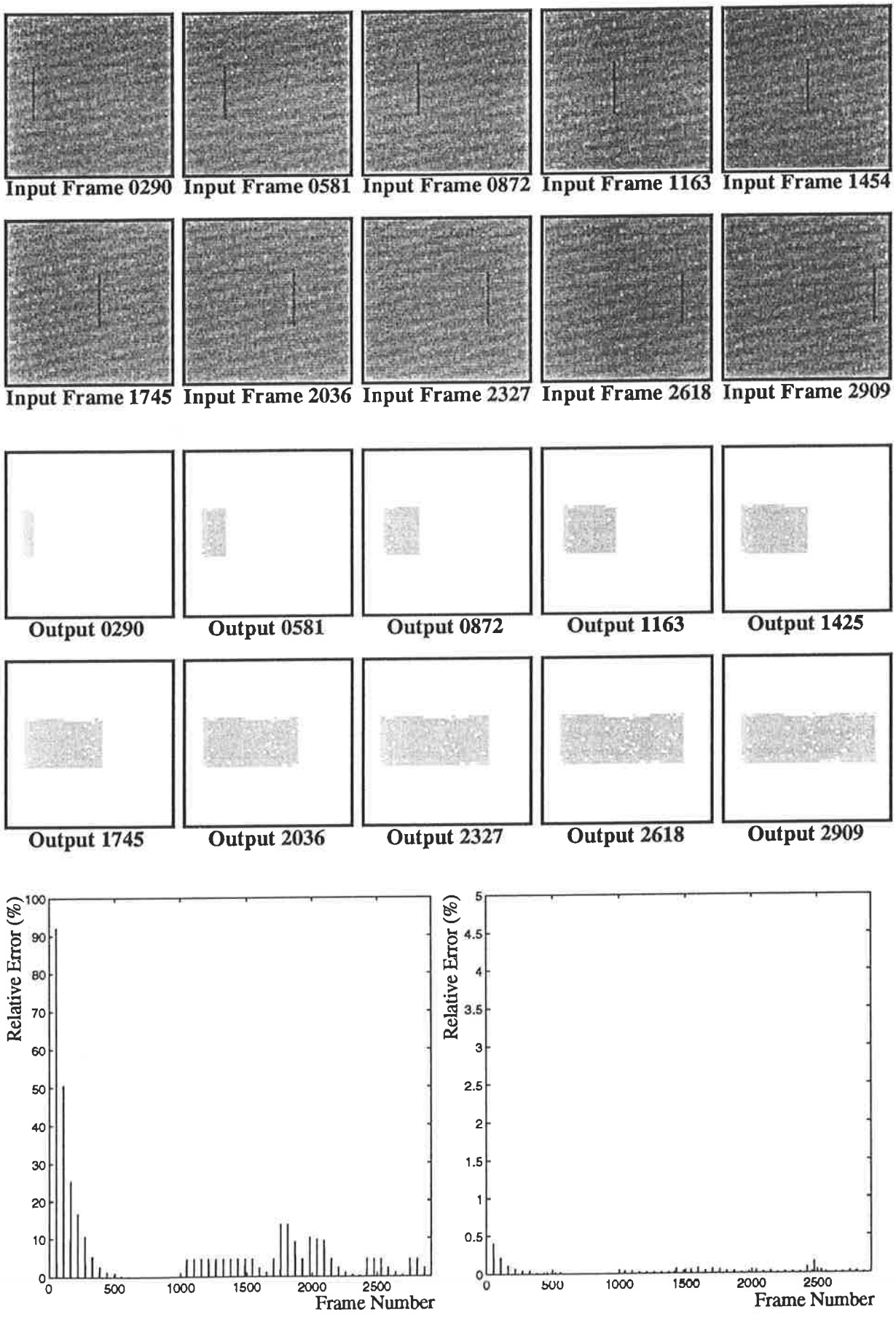


Figure 6.29: Moving object speed = 0.01818 ppf. Input information SNR = 12 dB.

6.5 Discussion

In the first experiment we simulated the fuzzy neural network and tested it on different image sequences to make sure that the system works well under different conditions. We utilized different object shapes, brightness patterns, speeds and trajectories, and also employed different background patterns. Although our choices are not the worst cases for proving the performance of the fuzzy neural network, they are not the easiest cases either. They are reasonable examples of conditions in which the detection of a moving object and estimation of its velocity is neither very easy nor very difficult, even for human beings. All the obtained results are reasonably good and fulfil our expectations.

In the second experiment, the moving object had the same shape, brightness pattern, and trajectory in all simulations, but the object speed and the level of noise in the input images varied from simulation to simulation. We selected six different velocities, which cover the entire range of possible velocities the system can detect. This gives us a good indication of the system performance for various velocities. As can be seen from the relative error graphs in Figures 6.6-6.29, there is a transient error at the beginning of each simulation. This error occurs because of the nature of the system. As was mentioned in the previous chapter, the system calculates the average velocity, based on previously obtained velocities. Since at the beginning of each simulation, there are no previously calculated velocities, the error is expected to be large. This error disappears quickly as more frames are processed, and decreases more rapidly if the velocity is high (see Figures 6.6, 6.10, 6.14, 6.18, 6.22, and 6.26). This transient behaviour of the system can be improved by changing the initial values of the variables which keep the last velocity values of the pixels. This can be done in a learning procedure.

As can be seen from the results, for the six noise free simulations (Figures 6.6, 6.10, 6.14, 6.18, 6.22, 6.26), there is no error in calculation of the velocity vectors (except for transition errors). This shows that the system works well in the absence of noise.

The system also works well if the noise level is moderate. Figures 6.7, 6.11, 6.15, 6.19, 6.23 and 6.27 illustrate the effect of moderate noise (SNR = 22 dB) on estimated velocities. The few spots in the output pictures, which are presented in the middle part of each figure, indicate

very small errors unrecognizable in both relative error graphs. Comparisons of the relative error graphs with their noise free counterparts show that there are no significant differences in error levels between the two cases. This proves the robustness of the algorithm to noisy conditions.

Furthermore, it was found that as the noise levels increase from moderate to high the system performance deteriorates slowly and gracefully. For a *SNR* of 17 dB, the results are given in Figures 6.8, 6.12, 6.16, 6.20, 6.24 and 6.28. Note that the relative errors for pixels which should receive motion (left graphs) do not exceed 10% (except in the transient phase), and the errors averaged over the entire frame (right graphs) are below 0.05%. For high noise levels (*SNR* = 12 dB), the results are presented in Figures 6.9, 6.13, 6.17, 6.21, 6.25, 6.29. The relative errors, in these figures, are below 20% for pixels receiving motion, and typically are around 10%; the errors averaged over the entire frame are below 2%. The sharp changes in error levels, that occurred in these simulations, are due to similar noise patterns occurring in two consecutive frames. When this happens, it causes a false motion detection in some pixels; thereby, inducing a large error in the velocity estimate of these pixels. This effect can be reduced by increasing the size of the sector representing a pixel, or by introducing some preprocessing such as low-pass filtering.

According to the simulation results discussed above, the fuzzy neural network showed a good performance in detecting moving objects and estimating their velocities. Although the noise levels were high, especially in the last case, the error levels are reasonably low. In addition, the results show that the performance of the system does not depend on velocity values of the moving objects.

Moreover, the parameter β of the DIR-FNs used in the fourth layer of the network and assigned 0.5 in our experiments can be adjusted during a learning process to achieve yet better results. To investigate the impact of this parameter on system performance, we have conducted an extra experiment in which the object speed was 0.2 ppf, the signal to noise ratio was 12 dB, and β was equal to 0.3 and 0.7 respectively. Upon comparing the results of this experiment with those obtained for simulation 2.1.4, (see Figure 6.30), it can be concluded that if the parameter β is increased, the error values for pixels that receive motion are increased whilst the error values for the overall pixels are decreased, and vice versa if the

value of β is decreased. This means that as the value of β is decreased, the system becomes more accurate in detecting motion but more sensitive to noise. The optimum value of β can be found with learning.

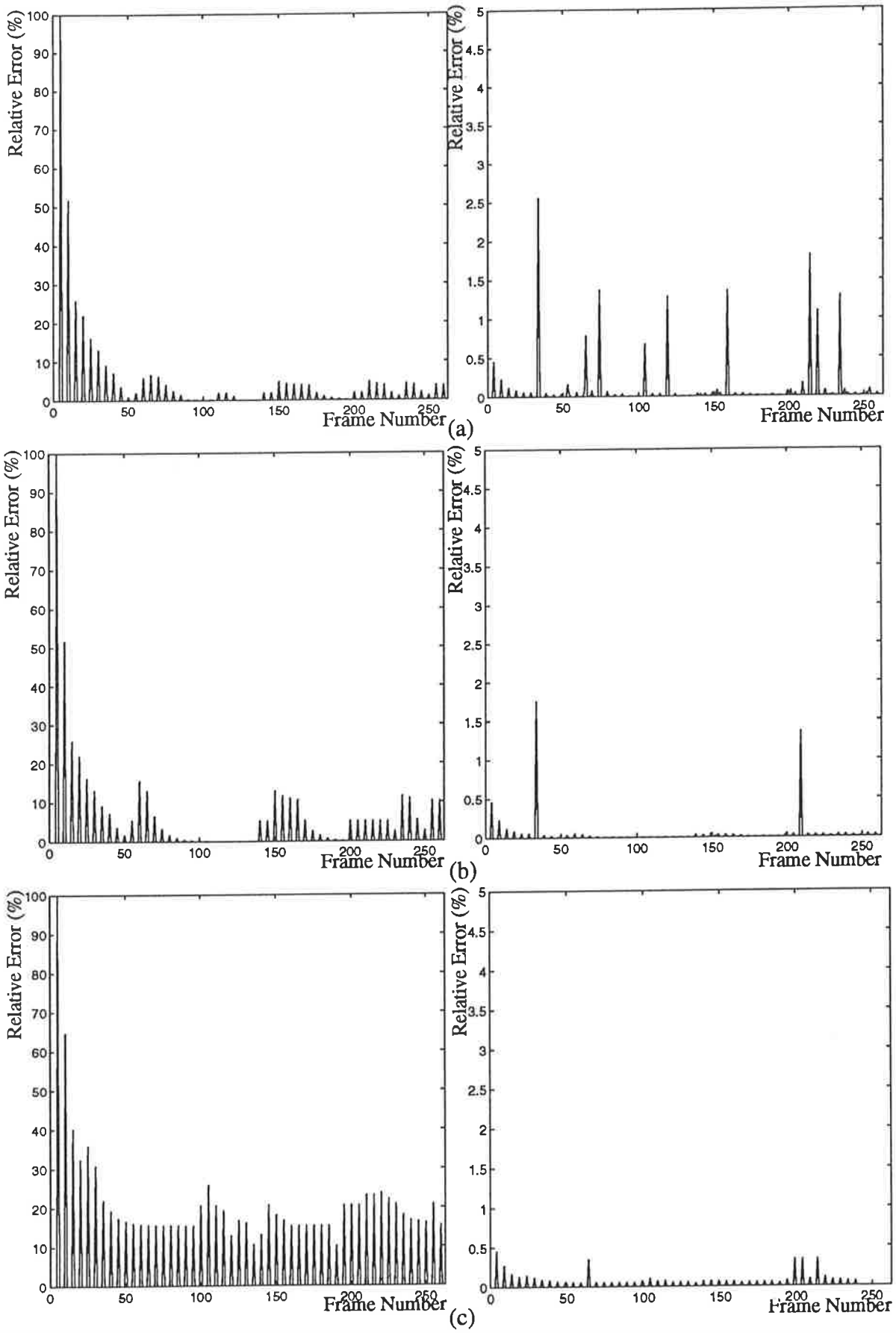


Figure 6.30: Error graphs for only the pixels that receive motion (left). Error graphs for all pixels (right). SNR is 12 dB. a) $\beta = 0.3$, b) $\beta = 0.5$, c) $\beta = 0.7$.

6.6 Conclusions

In this chapter, the experimental results were reported and discussed. First we explained the experimental procedure including simulation software which was developed for conducting the experiments. Two main experiments were conducted, each experiment contained different number of simulations. The first consisted of five simulations for objects moving with different speeds and directions. The objective of this experiment was to examine the quality of the motion estimator; that is, how the fuzzy neural network detects moving objects with different shapes, brightness patterns, and velocities. The second experiment consisted of twenty four simulations which contained a single moving object with six different velocities under four different noise conditions (noise free images, images with *SNR*=22 dB, images with *SNR*=17 dB, and images with *SNR*=12 dB). The objective of this experiment was to assess the performance of the system in noise free and noisy environments. The fuzzy neural network showed a good performance in detecting moving objects and estimating of their velocities. Although the noise levels were high, the error levels were reasonably low. In addition, the results showed that the performance of the system does not depend on the velocity values of the moving objects nor does it depend on the background conditions. In conclusion, the system performed well, and it can be utilized as a good motion estimator.

Chapter 7

CONCLUSIONS AND FUTURE DIRECTIONS

7.1 Overview

Fuzzy neural systems result from the fusion of fuzzy set theory and neural networks. Thus, the advantage of both approaches are merged. Research on fuzzy neural systems has been pursued in two different directions. Some researchers have utilized conventional artificial neuron models to develop neural networks which are functionally equivalent to fuzzy inference systems. Other researchers have developed neurons with fuzzy functions and fuzzy computations. This thesis dealt with the fuzzy neural networks of the second category. Seven chapters were presented in this thesis.

An introduction to the subject of fuzzy neural networks was given in Chapter 1. In Chapter 2, some of the basic concepts of fuzzy set theory which were required for understanding this thesis were explained. In Chapter 3, an introduction to the field of artificial neural networks was presented. The emphasis was on multilayer feedforward neural network architectures, which formed the basis of the fuzzy neural systems introduced in this thesis. A survey of four different types of fuzzy neurons and their related fuzzy neural networks were presented in Chapter 4. These four models have been found to be the most powerful fuzzy neural systems. In Chapter 5, which was the kernel of this thesis, a generic fuzzy neuron was defined as an extended model of existing fuzzy neurons. Then the foundation of the approach we employed for motion estimation was presented. The architecture of a fuzzy neural network was introduced to emulate the proposed motion estimation algorithm. Seven different fuzzy neurons were defined and utilized in different layers of the fuzzy neural network; they are simplified versions of the

generic fuzzy neuron. In chapter 6, we studied and discussed the results of more than thirty simulations of the proposed fuzzy neural network.

7.2 Conclusions

The first main contribution in this thesis was the generic model of a fuzzy neuron proposed as the basic element of our fuzzy neural system. It is a generalization of existing fuzzy neurons. In Chapter 5 it was shown that the generic fuzzy neuron can be simplified to the four models of fuzzy neurons described in Chapter 4. Moreover, we discussed the differences between the generic fuzzy neuron and the other models. It was described that it is possible to carry out any fuzzy computation and also express any kind of ambiguous relationship using the generic fuzzy neuron. It is claimed that the generic fuzzy neuron is the most powerful fuzzy neuron capable of fuzzy information processing. This is because we employed fuzzy relations in the generic fuzzy neuron functions (connection, aggregation, and activation). Moreover, we utilized the *-* compositional rule of inference, introduced in Chapter 2, to obtain the solution of the relational assignment equations used in the generic fuzzy neuron. In this way, any type of fuzzy operation in the class of triangular norms or triangular conorms can be utilized to derive the output of a generic fuzzy neuron function. This gives a high fuzzy processing power to the corresponding function and therefore to the generic fuzzy neuron.

The algorithm for motion detection and velocity estimation which was introduced in Chapter 5, formed the second main contribution of this thesis. In this algorithm, the direction and speed of motion are estimated not only using a comparison between brightness patterns in consecutive image frames by fuzzy relations, but also a control mechanism was established to strengthen or weaken the results of the comparisons. It was found that the proposed algorithm possesses the following features:

1) Suppressing the effect of noise.
2) Solving the correspondence problem, i.e., the problem of finding the correct match among other possible matches in brightness patterns.
3) Robustness against changes in illumination.

The only limitation of the algorithm is the assumption that the time interval between consecutive images is very small and no significant change occurs between two consecutive

frames, i.e., each object moves a maximum of one pixel between two consecutive frames. The assumption is to make the system as simple as possible. It significantly reduces the number of connections and nodes in the fuzzy neural network. As a result, the system's construction in VLSI technology will be feasible and the cost of the implementation will be cheaper. The proposed fuzzy neural network can be easily expanded to process frames containing moving objects travelling more than one pixel between consecutive images. Therefore, the assumption made is not a limitation of the system, but a simplification.

The third main contribution of this thesis is the five-layer feedforward fuzzy neural network architecture proposed for motion estimation. We evaluated the performance of the fuzzy neural network in simulation studies described in Chapter 6. According to the simulation results, the fuzzy neural network showed a significant performance in detection of moving objects and estimation of their velocities. Although the noise levels were high, the error levels were reasonably low. In addition, the results showed that the performance of the system does not depend on the velocity values of the moving objects nor does it depend on the background conditions. We found out that increasing the parameter β in DIR-FNs, increases the error values for the pixels receiving movement whilst it decreases the error values for the overall pixels. We concluded that a learning algorithm should be used to find the best β value.

The simulation results show that the fuzzy neural network has a great potential to be used in motion estimation which contains uncertainty and imprecision. The results of the demonstrated system suggest that the formulation of a fuzzy neural network by combining fuzzy set theory and neural networks is a fruitful one. The described fuzzy neural network may also be used for other computer vision applications.

7.3 Future Directions

In chapter 5 it is mentioned that the weights, the connection functions, the aggregation function, the threshold, and the activation function in the generic fuzzy neuron could be adjusted during a learning process. As a result, the fuzzy neural network constructed with neurons of this type can learn from the environment. In the proposed fuzzy neural network only β requires to be adjusted. This parameter was employed in the aggregation functions of the fourth layer DIR-FNs, and was utilized in the calculation of their corresponding fuzzy matrices. The simulations

were conducted with a fixed value of $\beta = 0 \cdot 5$, which was determined by trial and error. We did not present a learning algorithm for finding the value of β which gives an optimum performance. However, we intend to define and use a simple learning algorithm for this purpose in our future work. Moreover, it is intended the proposed system will form part of a real-time VLSI micro-sensor for motion detection and velocity estimation.

BIBLIOGRAPHY

[1] S. Amari, "A Theory of Adaptive Pattern Classifiers," *IEEE Trans. on Electronic Computers*, EC-16, pp. 299-307, 1967.

[2] S. Amari, K. Maginu,"Statistical Neurodynamics of Associative Memory," *Neural Networks*, 1, pp. 63-74, 1988.

[3] J.A. Anderson, J.W. Silverstein, S.A. Ritz, R.S. Jones, "Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model," *Psych. Review*, vol. 84, pp. 413-451, 1977, and reprinted in J.A. Anderson, E. Rosenfeld (Eds.), *Neurocomputing*, MIT Press, Cambridge, MA, 1988.

[4] R. Beale, T. Jackson, *Neural Computing: An Introduction*. IOP Publishing Ltd, 1990.

[5] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New york, 1981.

[6] G.A. Carpenter, S. Grossberg, "The ART of Adaptive Pattern Recognition," *Computer*, pp.77-88, March, 1988.

[7] S. Grossberg, "Adaptive Pattern Classification and Universal Recording, II: Feedback, Expectation, Olfaction, and Illusions," *Bilogical Cybernetics*, vol. 23, pp. 187-202, 1976(b).

[8] M.M. Gupta, G.K. Knopf, "Fuzzy Neural Network Approach to Control Systems," B.M. Ayyub, M.M. Gupta, L.N. Kanal, *Analysis and Management of Uncertainty: Theory and Applications*. Elsevier Science Publishers B.V., 1992.

[9] K. Hirota, W. Pedrycz, "OR/AND Neuron in Modeling Fuzzy Set Connectives," *IEEE Trans. on Fuzzy Systems*, vol. 2, no. 2, pp. 151-161, 1994.

[10] J.J. Hopfield, "Neural Networks and Physical Systems With Emergent Collective Computational Abilities," *Proc. of the National Academy of Sciences, USA*, vol. 79, pp. 2554-2558, 1982.

[11] J.J. Hopfield, "Neurons With Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proc.of the National Academy of Sciences, USA*, vol. 81, pp. 3088-3092, 1984.

[12] J.J. Hopfield, D. Tank, "Computing With Neural Circuits: A Model," *Sciences,* vol. 233, pp. 625-633, 1986.

[13] D.R. Hush, B.G. Horne, "Progress in Supervised Neural Networks," *IEEE Signal Processing Magazine*, pp. 8-39, January, 1993.

[14] B. Jahne, *Spatio-temporal Imege Processing: Theory and Scientific Applications.* Springer Verlag, Berlin New York, 1993.

[15] J.S.R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference Systems," *IEEE Trans. on Systems, Man, and Cybernetics,* vol. SMC-3, pp. 28-44, 1992.

[16] E.R. Kandel, J.H. Schwartz, *Principles of Neural Science*. New York: Elsevier, 1991.

[17] J.M. Keller, D.J. Hunt, "Incorporating Fuzzy Membership Functions into the Perceptron Algorithm," *IEEE Trans. on Pattern Anal. Machine Intell.*, vol. PAMI-7, no. 6, pp. 693-699, 1985.

[18] J.M. Keller, H. Tahani, "Implementation of Conjuctive and Disjunctive Fuzzy Logic Rules With Neural Networks," *Int'l J. Approximate Reasoning*, vol. 6, pp. 221-240, 1992.

[19] G.J. Klir, T.A. Folger, *Fuzzy Sets, Uncertainty, and Information,* Prentice Hall, Englewood Cliffs, N.J., 1988.

[20] T. Kohonen, "Correlation Matrix Memories," *IEEE Trans. on Computer*, C-21, pp. 353-359, 1972.

[21] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, pp. 59-60, 1972.

[22] B. Kosko, "Adaptive Bidirectional Associative Memories," *Applied Optics,* vol. 26, pp. 4947, 1987.

[23] B. Kosko, "Constructing an Associative Memory," *BYTE*, pp. 137-144, September, 1987.

[24] B. Kosko, "Fuzzy Associative Memories," *Fuzzy Expert Systems,* Addison-Wesley, Reading, MA, 1987.

[25] B. Kosko, "Fuzzy Cognitive Maps," *Intelligent Journal of Man-Machine Studies,* vol. 24, pp. 65-75, 1986.

[26] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1992.

[27] A.Z. Kouzani, A. Bouzerdoum, M.J. Liebelt, "Motion Detection and Velocity Estimation Using Cellular Automata," *Proc. of the Second Australian and New Zealand Conference on Intelligent Information Systems*, pp. 327-331, 30 Nov.-2 Dec. 1994, Brisbane, Australia.

[28] H. K. Kwan, Y. Cai, "A Fuzzy Neural Network and its Application to Pattern Recognition," *IEEE Trans. on Fuzzy Systems*, vol. 2, no. 3, pp. 185-193, 1994.

[29] Y.J. Lai, C.L. Hwang, *Fuzzy Mathematical Programming: Methods and Applications.* Springer-Verlag, Berlin, Heidelberg, 1992.

[30] M.A. Lee, H. Takagi, "Integration Design Stages of Fuzzy Systems Using Genetic Algorithms." *Second IEEE International Conference on Fuzzy Systems, San Fransisco, California, March 28 -April 1, 1993.* IEEE Press, 1993.

[31] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part I," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404-418, 1990.

[32] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part II," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 419-435, 1990.

[33] S.C. Lee, E.T. Lee, "Fuzzy Neural Networks," *Mathematical Biosciences*, vol. 23, pp. 151-177, 1975.

[34] H. Li, H.S. Yang, "Fast and Reliable Image Enhancement Using Fuzzy Relaxation Technique," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-19, no. 5, pp. 1276-1281, 1989.

[35] J.N. Lin, S.M. Song, "A Novel Neural Network for the Control of Complex Systems," *IEEE International Conference on Neural Networks, June 28 - July 2, 1994, Orlando, Florida.* IEEE Press, 1994.

[36] R.P. Lippmann, "An Introduction to Computing With Neural Nets," *IEEE Magazine on Acoustics, Signals, and Speech Processing*, pp. 4-22, April, 1987.

[37] E.H. Mamdani, S. Assilian, "A Case Study on Application of Fuzzy Set Theory to Automatic Control," *Proc. IFAC Stochastic Control Symp.*, Budapest, 1974.

[38] W.S. McCulloch, W.H. Pitts, "A Logical Calculus of Ideas Immanent In Nervous Activity," *Bull. Math. Biophys.*, vol. 5, pp. 115-133, 1943.

[39] S.K. Pal, R.A. King, "Image Enhancement Using Smoothing with Fuzzy Sets," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 7, pp. 494-501, 1981.

[40] S.K. Pal, A. Rosenfeld, "Image Enhancement and Thresholding by Optimization of Fuzzy Compactness," *Pattern Recognition Letters,* vol. 7, pp. 77-86, 1988.

[41] W. Pedrycz, "Neurocomputing in Relational Systems," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 13, no. 3, pp. 289-296, 1991.

[42] W. Pedrycz, A.F. Rocha, "Fuzzy-Set Based Models of Neurons and Knowledge-Based Networks," *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 4, pp. 254-266, 1993.

[43] N. Rescher, *Many-Valued Logic*, McGraw-Hill, New York, 1969.

[44] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psych. Review*, vol. 65, pp. 386-408, 1958, and reprinted in J.A. Anderson, E. Rosenfeld (Eds.), *Neurocomputing*, MIT Press, Cambridge, MA, 1988.

[45] M. Sugeno, "Theory of Fuzzy Integrals and Its Applications," PhD Thesis, Tokyo Institute of Technology, 1974.

[46] H. Takagi, I. Hayashi, "NN-Driven Fuzzy Reasoning." *Int'l J. Approximate Reasoning,* vol. 5, no. 3, pp. 191-212, 1991.

[47] D.W. Tank, J.J. Hopfield, "Collective Computation in Neurolike Circuits," *Scientific American*, vol. 257, pp. 104-114 & 158, 1987.

[48] T. Terano, K. Asai, M. Sugeno, *Fuzzy Systems Theory and Its Applications.* Academic Press, Inc., San Diego, CA, 1992.

[49] W.G. Wee, K.S. Fu, "A Formulation of Fuzzy Automata and its Application as a Model of Learning Systems," *IEEE Trans. Syst. Sci. & Cybern.,* vol. SSC-5, pp. 215-223, 1969.

[50] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences.* PhD Dissertation, Harvard University, Cambridge, MA, 1974.

[51] B. Widrow, M.E. Hoff, "Adaptive Switching Circuits," *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record*, Part 4, pp. 96-104, and reprinted in *Neurocomputing*, J. Anderson, E. Rosenfeld (Eds.), MIT Press, Cambridge, MA 126-134, 1988.

[52] T. Yamakawa, S. Tomoda, "A Fuzzy Neuron and Its Application to Pattern Recognition," *Proc. Third IFSA Congress,* Washington, pp. 943-948, August 6-11, 1989.

[53] T. Yamakawa, "A Fuzzy Inference Engine in Nonlinear Analog Mode and Its Application to a Fuzzy Logic Control," *IEEE Trans. on Neural Networks,* vol. 4, no. 3, pp. 496-522, 1993.

[54] T. Yamakawa, M. Furukawa, "A Design Algorithm of Membership Functions for A Fuzzy Neuron using Example-Based Learning," *IEEE International Conference on Fuzzy Systems, March 8-12, 1992, San Diego, California.* IEEE Press, 1992.

[55] L.A. Zadeh, "A Rationale for Fuzzy Control," *J. Dynamic Systems, Measurment and Control,* vol. 94, Series G, pp. 3-4, 1972.

[56] L.A. Zadeh, "Fuzzy Algorithms," *Inform. Control,* vol. 12, pp. 94-102, 1968.

[57] L.A. Zadeh, "Fuzzy Logic," *Computer,* vol. 21, no. 4, pp. 83-93, 1988.

[58] L.A. Zadeh, "Fuzzy Sets," *Inform. Control,* vol. 8, pp. 338-353, 1965.

[59] L.A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Trans. on Systems, Man, and Cybernetics,* vol. SMC-3, pp. 28-44, 1973.

[60] L.A. Zadeh, "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning," *Information Sciences,* vol. 8, pp. 199-249 & 301-357, vol. 9, pp. 43-80, 1975.

[61] H.J. Zimmermann, *Fuzzy Set, Decision Making and Expert Systems.* Kluwer Academic Publishers, Boston, 1987.

[62] H.J. Zimmermann, *Fuzzy Set Theory- And Its Applications, Second Edition.* Kluwer Academic Publishers, Boston, 1991.

[63] J.M. Zurada, *Introduction to Artificial Neural Systems.* West Publishing Co., 1992.