Pergamon

Pattern Recognition, Vol. 29, No. 1, pp. 1-17, 1996
Elsevier Science Ltd
Pattern Recognition Society
Printed in Great Britain.
0031-3203/96 $15.00+.00

0031-3203(95)00069-0

INVARIANT PATTERN RECOGNITION: A REVIEW*

JEFFREY WOOD
Department of Computer Science, Royal Holloway, University of London, Egham, Surrey, TW20 0EX, U.K.

(*Received* 4 *October* 1994; *received for publication* 15 *May* 1995)

Abstract—In this document we review and compare some of the classical and modern techniques for solving the problem of invariant pattern recognition. Such techniques include integral transforms, construction of algebraic moments and the use of structured neural networks. In all cases we assume that the nature of the invariance group is known *a priori*. Many of the methods described apply to specific geometrical transformation groups; however some of the techniques are highly general and applicable to large classes of invariance groups. We also review some results regarding the existence and structure of invariants under certain kinds of groups.

Invariance   Pattern classification   Group theory   Integral transforms
Fourier transforms   Moment invariants   Neural networks   Weight sharing
Higher order networks

1. INTRODUCTION

The problem of invariant pattern recognition is recognized as being a highly complex and difficult one. It is not surprising, therefore, that a wide variety of techniques have been invented to deal with specific or general instances of this problem. Our aim in this paper is to give an overview of some of the methods that have been used to solve this problem, and to compare and contrast these methods.

We refer often to group-theoretic terms, particularly in Section 7. The foreknowledge we assume may be found in the standard literature; see for example[1-3] for background in finite groups, or references (4-7) for an introduction to Lie group theory. The reader familiar with the basic concepts of group theory should have little difficulty with this document.

1.1. *The problem of invariance*

Formally, the issue in question is as follows.

We have a set V of possible patterns. A pattern is assumed to be a function (real- or complex-valued) on some set X. In some parts of this paper we will make certain further assumptions about V, for example that its elements are square-integrable over X.

Each pattern $f \in V$ has a desired classification, $c(f)$. The general pattern recognition problem is to construct a system which takes as input an element f of V and computes a value $s(f)$, with the intention that $s(f) = c(f)$ for all $f \in V$.

* This research was funded by the Science and Engineering Research Council and the Defence Research Agency under CASE award no. 92567042.

In the pattern classification problems which we will be considering, there is a group $\mathcal{G}$ which acts upon the set X. Through this action, $\mathcal{G}$ also has an action on the space V, given for any $g \in G$, $f \in V$ by

$$(gf)(x) = f(g^{-1}x) \quad \forall x \in X$$

The use of g^{-1} instead of g ensures that $(g_1g_2)f$ equals $g_1(g_2f)$ and not $g_2(g_1f)$.

The group $\mathcal{G}$ could be a geometrical group such as the group of all translations, rotations, scalings, etc, or a more abstract group such as a cyclic translation or other finite permutation group. The desired classification of a pattern is invariant under the action of the group, i.e.

$$c(gf) = c(f) \quad \forall g \in \mathcal{G}, f \in V$$

We assume that the *invariance group* $\mathcal{G}$ is known *a priori*. Webber has designed a system[8] for learning a pattern set with unknown invariances. We will not consider such an approach here.

Since we have prior knowledge of the classification problem, we should be able to improve the generalization ability of any given pattern classifier by incorporating this knowledge into the classification system. Thus we are now aiming to build constraints on the function s which correspond to those on c; i.e. we desire that

$$s(gf) = s(f) \quad \forall g \in \mathcal{G}, f \in V$$

This is the problem of invariant pattern recognition.

1.2. *Approaches to the invariance problem*

There are two broad approaches to the problem as described above. Others exist, for example the transformation reconstruction method of Fayn *et al.*[9] which we will discuss briefly later in the paper.

(1) The classification process can be broken down into the distinct phases of invariant feature extraction and feature classification. In other words, we can first calculate a set of features of the input, each of which is invariant under the group $\mathcal{G}$, and then we can classify the patterns from the information given in their feature vectors, without needing to consider the symmetries of the actual pattern set. The feature classification can thus be done using any conventional pattern recognition technique, for example multi-layer perceptron classification.

(2) We can attempt to combine these two phases, essentially by constructing an invariant which is parameterized. The parameters of the invariant can be adapted in order to perform the desired classification. Such techniques lend themselves naturally to neural network style approaches.

We now list several issues which we need to consider when constructing a system for invariant pattern recognition. Some of these are clearly general considerations in building any pattern classifier, but must be taken into account when choosing a scheme for producing invariance.

- Invariance versus tolerance. We have stated that we require the function s to be invariant under $\mathcal{G}$; in practice only an approximate invariance may be obtainable. This may arise through computational inaccuracies combined with the continuous nature of some transformation groups. It is likely that we will prefer a function s which is "as invariant as possible". However, we may be satisfied with approximate invariance (which we might call *transformation tolerance*). We might even prefer a transformation-tolerant-system, on the grounds of biological realism (the eyes, for example, cannot easily read writing when written upside-down, so our own visual system is not truly rotation-invariant).
- Discriminability. Invariance alone is not a sufficient group-theoretic requirement for a pattern classifying system applied to a symmetrical pattern set.

We assume without loss of generality that the output $s(f)$ of our classification system is a function Λ of a set of invariants $\{I_1(f), I_2(f), \ldots\}$. This scheme includes both basic approaches listed above, since the function Λ could be a simple thresholding function of a single (possibly parameterized) invariant I_1.

It is important that the system be able to distinguish patterns which are not related by a group transformation; otherwise the desired classification function c may not be obtainable. In other words, we desire that for at least one of our invariants I_k we have

$$I_k(f_1) = I_k(f_2) \Rightarrow \exists g \in \mathcal{G} \text{ s.t. } gf_1 = f_2$$

for all patterns $f_1, f_2 \in V$. Note that the converse of this requirement is the criterion for I_k to be an invariant. If an invariant I_k obeys this rule, we say that it *discriminates perfectly*. A classification system for which at least one invariant I_k obeys the rule is said to have *perfect discriminability*.

We may be satisfied if at least one of any two patterns f_1, f_2 breaking the above rule is highly unlikely to be presented to the system in practice, or if we can be sure that any two such patterns will have the same desired classification $c(f)$.

- Computational complexity. It is also important that the system should not take an unreasonable amount of time to compute s, and that it should not occupy an unreasonable amount of space.
- Ease and speed of training. In practice, a pattern classification system will be trained on a finite number of prototypes. We desire this process to be as simple and as fast to execute as possible. This training speed should not be confused with the processing speed just mentioned.
- Generalization ability. It is important that the system generalizes well to the patterns which it does not see during training. In particular, it is desirable in most applications that the system should be noise tolerant.
- Flexibility. It is desirable to be able to easily change the structure of the system to accommodate changes in the classification problem, in particular changes in the nature of the invariance group. There is however a natural trade-off here between the flexibility of the system and its simplicity. Thus some systems are applicable to many problems and many invariance groups, but these are more complex than those tailored to invariance under a specific group (and perhaps with regard to a specific problem).
- Transformation retrieval. Another possible requirement of the system is that, when a pattern is identified as being in the orbit of a prototype, the group transformation giving the pattern from the prototype should be retrievable from the classification system. We will not give much attention to this last issue.

1.3. *Overview of the paper*

The rest of the paper is arranged as follows.

In Section 2, we discuss integral transforms which are used as invariant feature extractors. These transforms are based largely on Fourier analysis, and are probably the most common tools used in invariant pattern recogniton. In Section 3, we review the theory of moments, including Zernike moments. Moment functions are also commonly used as invariant features. Section 4 deals with the use of neural networks in the invariance problem. This includes higher-order neural networks, and also first-order networks calculating feature hierarchies, such as the neocognitron. These connectionist methods make no distinction between feature extraction and classification.

In Section 5 we cover various techniques which do not fit easily into any of the above categories. Section 6 compares some of the methods described in the earlier sections, with reference to the practical issues listed above. Section 7 contains some highly theoretical results on the existence and form of group invariants, and Section 8 concludes.

2. INTEGRAL INVARIANTS

In this section we describe invariants which are computable by integral transformations. Such transformations are generally based on the Fourier Transform and its variants. In fact, Ferraro and Caelli have proved that any integral transform used to produce invariants under a two-dimensional Lie group must have the form of a Fourier Transform. This result is detailed in Section 2.6.

2.1. *General integral transforms*

Let X and W be two sets. Let f be a complex-valued function on the set X, k a complex-valued function on the set $X \times W$. Then the general integral transform of the function f over the set X is given by the formula

$$z(w) = \int_X f(x)k(w,x)\,dx \qquad (1)$$

Here z is a complex-valued function on the set W. The function k is called the *kernel* of the transform. Note that, for a finite set X, the integral above becomes a summation.

We now assume that there is a group $\mathcal{G}$ acting on the set X.

For use in invariant feature extraction, we generally require that the integral (as determined by its kernel) meets the following criteria [see Caelli and Liu[10]].

(1) The transform must be invertible.

(2) The modulus of the transform at any point w must be invariant under the action of $\mathcal{G}$ on X.

The first requirement simply ensures that there is no information loss. The second specifies an arbitrarily large set of invariant features: the moduli of the values of z evaluated at any given set of points $w_1, \ldots, w_m$. Note also that the integral transform is linear and hence easy to compute (in its discrete or approximate form, that is). We will return to the general integral transform at the end of Section 2. First, however, we wish to consider a number of specific integral transforms.

2.2. *The Fourier Transform*

The largest body of knowledge concerning preprocessing to extract group invariants undoubtably lies in the area of Fourier and similar Transforms. We will consider here first classical Fourier Transforms and then their variants, used to produce invariance under rotations and/or scalings (dilations). One of the main advantages of Fourier Transform methods in invariant pattern recogniton is the fact that the power spectrum, thanks to the use of the Fast Fourier Transform, can be computed in $O(n \log n)$ time. This compares well to many other mechanisms for producing invariance. For more information on Fourier Transforms, consult the standard literature.[11,12] A group-theoretic approach to the Discrete Fourier Transform is given for example in Clausen and Baum.[13] Another review of Fourier Transform methods for invariant pattern recognition appear in Wechsler.[14]

2.2.1. *The continuous Fourier Transform.* The *continuous Fourier Transform* (FT) of the function f is defined by the equation

$$z(w) = \int_{-\infty}^{\infty} f(x)e^{-2\pi iwx}\,dx$$

This is clearly an integral transform as defined by equation (1) with kernel $k(w,x) = e^{-2\pi iwx}$. It is also invertible.

The modulus of the Fourier Transform (i.e. the power spectrum) at any point w is invariant under translation of the function f. This follows from the convolution theorem or by a group-theoretic argument, but can also be seen as follows. Here z' denotes the FT of the translated function.

$$\begin{aligned} z'(w) &= \int_{-\infty}^{\infty} f(x-a)e^{-2\pi iwx}\,dx \\ &= \int_{-\infty}^{\infty} f(x)e^{-2\pi iw(x+a)}\,dx \\ &= e^{-2\pi iwa}z(w) \end{aligned}$$

Hence $\|z'(w)\| = \|z(w)\|$ and the Fourier Transform satisfies our requirements for a general integral transform.

We also have the circular Fourier Transform,[15] defined for a periodic function f with period 2π (or equivalently a function f defined on the unit circle):

$$z(l) = \frac{1}{2\pi}\int_0^{2\pi} f(\theta)e^{-il\theta}\,d\theta = \frac{e^{il\phi}}{2\pi}\int_0^{2\pi} f(\theta-\phi)e^{-il\theta}\,d\theta \qquad (2)$$

Here l is normally taken to be an integer; the $z(l)$'s are by definition the Fourier coefficients of the function f, and so the set of values $\{z(l)\}$ for all integers l characterize f without loss of information.

The power spectrum of this function z is invariant under translation of the function (equivalently under rotation, if θ is viewed as an angle). Often the circular Fourier Transform is written

$$f_l(r) = \frac{1}{2\pi}\int_0^{2\pi} f(r,\theta)e^{-il\theta}\,d\theta, \qquad (3)$$

where $f(r,\theta)$ is a function of the real plane, expressed in polar coordinates, and $f_l(r)$ is a function of the radius only.

2.2.2. *The discrete Fourier Transform.* The properties of the Discrete Fourier Transform (DFT) are analogous to those of the continuous Fourier Transform. The power spectrum of the DFT is invariant under cyclic translation of the input vector. Again this can be seen either from the cyclic convolution theorem, or from an interpretation of the DFT as a homomorphism between representations of the cyclic translation group.[13] The Discrete Fourier Transform is of course also invertible, and hence again satisfies the conditions for an integral transform invariant.

2.2.3. *The multi-dimensional Fourier Transform.* The Fourier Transform can of course be generalized to functions defined on any real space (and similarly for the Discrete Fourier Transform). Formally, we have

$$z(w) = \int_X f(x)e^{-2\pi i\langle w,x\rangle}\,dx,$$

where $X = \mathbb{R}^n$ for some n and w, x denote vector elements of X with inner product $\langle w,x\rangle$. Again the power spectrum of the transform is an invariant under the n-dimensional translation group.

2.3. *The Mellin Transform*

Let $f(r)$ denote a function of the positive real line. The *Mellin Transform*[16–18] of f is denoted by the equation

$$z(u) = \int_0^\infty f(r)r^{u-1}\,dr, \tag{4}$$

where u is a complex number. When $\Re e(u) > 0$ (and also f is defined on compact set), the Mellin integral is guaranteed to converge. However it is usual to prefer imaginary values of u, so a compromise is usually made by taking $u = \varepsilon + iw$, for a small value of ε. When u is imaginary, we obtain the alternative form of the Mellin Transform:[10,15]

$$z(w) = \int_0^\infty f(r)r^{iw-1}\,dr, \tag{5}$$

where w is real. Here we can make the substitution $r = e^{-2\pi a}$ to obtain the equation

$$z(w) = \int_0^\infty f_1(a)e^{-2\pi i a w}\,da,$$

where f_1 is defined by $f_1(a) = -2\pi f(e^{-2\pi a})$. This coordinate mapping therefore turns the Mellin Transform into a Fourier Transform. We can alternatively say that the Mellin Transform is a Fourier Transform evaluated over an exponential scale.

Furthermore, applying a scale factor K to the function f results via this coordinate mapping in a shift proportional to $\ln K$ of f_1. Consequently the magnitude of this form of the Mellin Transform at a point w is invariant (by the invariance laws of the Fourier Transform) under scaling. Note that this law can only be applied when $u = iw$ is imaginary.

The Mellin Transform can also be applied in two dimensions:

Let f be a function defined on the positive quadrant of the real plane, indexed by coordinates (x_1, x_2).

The two-dimensional *Mellin Transform*[16] of the function f is defined by

$$z(w_1, w_2) = \int_0^\infty\int_0^\infty f(x_1,x_2)x_1^{iw_1-1}x_2^{iw_2-1}\,dx_1\,dx_2,$$

again with w_1, w_2 real. By the mapping $x_1 = e^{-2\pi a_1}$, $x_2 = e^{-2\pi a_2}$, we obtain another Fourier Transform. Thus the magnitude of the two-dimensional Mellin Transform is invariant under one-axis scaling of f.

2.3.1. *The Fourier-Mellin Transform.* Let $f(r,\theta)$ be a function of the two-dimensional plane expressed in polar coordinates. By applying the circular Fourier Transform [equation (3)] to f we obtain a function of r and the variable l. To this we can apply the Mellin Transform in form (5), which gives us the following transform:

$$z(l,w) = \frac{1}{2\pi}\int_0^\infty\int_0^{2\pi} f(r,\theta)e^{-il\theta}r^{iw-1}\,d\theta\,dr$$

This is called the *Circular–Fourier Radial–Mellin Transform*, or simply the *Fourier–Mellin Transform*. By the *log-polar* or *complex-log* mapping $(r,\theta)\mapsto(\ln r,\theta)$, we realise this transform as a two-dimensional Fourier transform. The modulus of $z(l,w)$ is thus an invariant under both rotation and scaling. The coefficients $z(l,w)$ are often referred to as *Fourier–Mellin descriptors*. Their magnitudes are often used in practice as invariants under the two-dimensional rotation and scaling group.[17,19–22]

The function f can be reconstructed from its Fourier–Mellin descriptors $z(l,w)$ as follows:

$$f(r,\theta) = \int_{-\infty}^{\infty}\sum_{l=-\infty}^{\infty} z(l,w)e^{il\theta}r^{-iw}\,dw$$

This is called the *log-polar circular harmonic decomposition* of f, as published by Caelli and Liu.[10] Thus the Fourier–Mellin Transform in this form satisfies the conditions of an integral invariant (see Section 2.1). De Bourgrenet de la Tocnaye and Ghorbel[19] have shown how the parameters of a rotation and scaling operation can be recovered from the Fourier–Mellin descriptors of an original and transformed pattern.

Returning to our original form of the Mellin Transform [equation (4)], we have an analogous form of the Fourier–Mellin Transform, as follows:

$$z(l,u) = \frac{1}{2\pi}\int_0^\infty\int_0^{2\pi} f(r,\theta)e^{-il\theta}r^{u-1}\,d\theta\,dr \tag{6}$$

This form has been used by Li[16] and by Sheng and Arsenault.[17] The modulus of this transform is invariant under rotation but not under scaling. When u is taken to be real, however, it can be normalized in a number of different ways. Li used the method

$$\hat{z}(l,u) = \frac{z(l,u)}{m_{0,0}^{u/2}}, \tag{7}$$

where $m_{0,0}$ is the zeroth order moment, to be defined in Section 3.1, equal to the integral of the function f. The magnitude of $\hat{z}(l,u)$ is invariant under both rotation and scaling for any real l, u.

2.3.2. *Fourier techniques for larger invariance groups.* We have shown how it is possible to use Fourier transforms to find a set of features which are invariant either under two-dimensional translation or under rotation and scaling. This leaves this question: how is it possible to find a set of invariants under all three kinds of transformation?

One method is to centralize the data to compensate for shifts, and then to apply a Fourier–Mellin transform. The following alternative was used by Casasent and Psaltis:[23,24]

(1) Calculate the power spectrum of the Fourier Transform of the two-dimensional input. This is invariant under translation.

(2) Convert the power spectrum to polar coordinates.

(3) Perform a complex-log mapping.

(4) Calculate another two-dimensional Fourier Transform power spectrum. This will also be rotation- and scale-invariant.

2.4. *Cross correlation*

The following is largely taken from Pintsov.[25]

Let f denote a pattern or image, t a template. Both are complex-valued functions on a set X, upon which the group $\mathcal{G}$ acts. Hence $\mathcal{G}$ has a defined action on the pattern and template. The *cross correlation* of f and t with respect to the group $\mathcal{G}$ is defined by:

$$C(g) = \int_X f(x)t(g^{-1}x)\,dx \tag{8}$$

Thus we have a mapping C from $\mathcal{G}$ into the complex numbers.

This is an integral transform as of the manner of Section 2.1, though in general it satisfies neither of the constraints listed in that section. It does however have the following property for any $h \in \mathcal{G}$:

$$\begin{aligned} C(g)(hf) &= \int_X f(h^{-1}x)t(g^{-1}x)\,dx \\ &= \int_X f(x)t(g^{-1}hx)\,dx \\ &= C(h^{-1}g)(f), \end{aligned}$$

assuming the use of an invariant integral.

Thus the cross correlation coefficients are permuted upon the application of transformation h to the input pattern.

Fixing f and varying g and t, within a constant factor we find that $C(g)$ is maximized when gt is equal to f. Thus, using prototypes as templates, we can discover which prototype matches most closely to the input pattern, and by which group element the prototype would have to be transformed to obtain this closest match.

Classical cross correlation occurs as cross correlation with respect to the two-dimensional translation group:

$$C(w_1, w_2) = \int_{\mathbb{R}^2} f(x_1, x_2)t(x_1 - w_1, x_2 - w_2)\,dx_1\,dx_2$$

2.4.1. *The Radon Transform.* Pintsov[25] has applied the generalized cross correlation formula to obtain the classical Radon Transform.[26]

Let $\mathcal{G}$ denote the group of rigid motions of the plane (i.e. rotations followed by translations). Let the template t be the x_2-axis; thus $t(x_1, x_2) = \delta(x_1)$. The template transformed by the element $g = (\theta, a_1, a_2)$ (i.e. a rotation about the origin by θ followed by a translation of (a_1, a_2)) is given by

$$\begin{aligned} (gt)(x) &= t(g^{-1}x) \\ &= \delta(x_1 \cos\theta + x_2 \sin\theta - a_1 \cos\theta - a_2 \sin\theta) \end{aligned}$$

Substitution into the cross correlation equation (8) gives us the Radon transform:

$$C(g) = \int_{\mathbb{R}^2} f(x)\delta(x_1 \cos\theta + x_2 \sin\theta - p)\,dx_1\,dx_2,$$

where $p = a_1 \cos\theta + a_2 \sin\theta$.

2.5. *Triple correlation*

Triple correlation was used by Delopoulos, Tirakis and Kollias[27] to obtain invariance under the translation, scaling and rotation group of the plane. We denote this group here by $\mathcal{G}$. *Triple correlation* is a natural generalization of autocorrelation and is defined by

$$C_3(a,b) = \frac{1}{T^n}\int_X f(x)f(x+a)f(x+b)\,dx,$$

where f is a real-valued function defined on the interval $X = [0, T]^n$ and a and b are displacement vectors. Each pair of vectors (a,b) can be naturally identified with a triangle.

It is easy to see that the triple correlation is invariant under translation of f within the set X. It is also true that it commutes with both scalings and rotations (i.e. scaling or rotation of the function f causes a permutation of the triple correlation values). Each orbit of the correlation coefficients under $\mathcal{G}$ is the set of pairs (a,b) corresponding to a set of mutually similar triangles.

To obtain a set of invariant features under $\mathcal{G}$, Delopoulos *et al.* suggest the following scheme:

(1) Perform a triple correlation of the input function f.

(2) Within each correlation coefficient orbit, fix a vector pair (a,b). Now the coefficient $C_3(a',b')$ within a given orbit can be indexed by the transform parameters (K,θ) which describe the group transformation required to obtain the vector (a',b') from the corresponding representative pair (a,b).

(3) Perform a log-polar mapping $(K,\theta) \mapsto (p = \ln K, \theta)$ on each orbit.

(4) Execute a 2D Fast Fourier Transform.

The power spectrum obtained for a given orbit must then be a set of rotation-, scaling- and translation-invariant features. To form a perfectly discriminating set of invariants, other features must be added to these power spectra.[27] However, for reasons of practical computability, it is suggested that only the power spectra themselves (or possibly even the first component of each power spectrum) are used.

A final note of Delopoulos *et al.* on the triple correlation is that a third order neural network (see Section 4.3.2) can be realized as a first order weighted sum of the triple correlation coefficients (the weights being constrained to be the same for coefficients corresponding to similar triangles).

2.6. *Existence of Integral Transform Invariants*

Ferraro and Caelli[28] have considered the general two-dimensional integral transform, given by

$$z(w_1, w_2) = \int_{-\infty}^{\infty}\int f(x_1, x_2) k(w_1, w_2; x_1, x_2)\, dx_1\, dx_2$$

The problem discussed by Ferraro and Caelli is the existence of an integral transform invariant under some given Lie group composed of two one-parameter subgroups. They prove that an invariant integral of this form exists if and only if the infinitesimal operators of the two one-parameter subgroups commute and are linearly independent. When this condition is satisfied, the integral above can be written with a suitable change of coordinates as a Fourier Transform. The details are as follows.

Let L_α and L_β denote the infinitesimal operators of the invariance group, defined in terms of the coordinate system (x_1, x_2). A suitable change of coordinates (x_1, x_2) to (a_1, a_2) is given by

$$L_\alpha a_1 = 1 \quad L_\alpha a_2 = 0$$
$$L_\beta a_1 = 0 \quad L_\beta a_2 = 1$$

Denoting by $|J(a_1, a_2; x_1, x_2)|$ the Jacobian of the coordinate change, the integral transform in terms still of the original coordinates becomes.

$$z(w_1, w_2) = \int_{-\infty}^{\infty}\int f(x_1, x_2) \left| J\begin{pmatrix} a_1 & a_2 \\ x_1 & x_2 \end{pmatrix} \right| \exp(-i[a_1 w_1 + a_2 w_2])\, dx_1\, dx_2$$

This clearly has the form of a Fourier Transform. We have already seen an example of this transformation in the Fourier–Mellin Transform (Section 2.3.1).

Note also that, as a consequence of the existence result above, an invertible integral transform with an invariant modulus under either (i) translation and rotation or (ii) translation and scaling does not exist. This is because the infinitesimal generators of these groups do not commute.

3. ALGEBRAIC INVARIANTS

Algebraic invariants, or moment invariants, are obtained by taking quotients and powers of moments. A moment is a weighted sum of the pattern $f(x)$ over the whole input field, with weights equal to some polynomial in x. The simplest kind of moments are the regular moments, which we will consider first. We will then deal with Zernike moments, the generalized moments proposed by Shvedov, Shmidt and Yakubovich and finally Flusser and Suk's moment formulae for affine transformation invariance.

3.1. *Regular moments*

For non-negative integers p, q, the $(p+q)$th order *moments* of a pattern $f(x, y)$ defined on the plane $\mathbb{R}^2$ are given by the following:[14,16,29]

$$m_{p,q} = \int_{-\infty}^{\infty}\int x^p y^q f(x, y)\, dx\, dy \tag{9}$$

$f(x, y)$ is assumed to be a piecewise continuous function which is non-zero in only a finite part of the plane, and p and q are taken to be non-negative integers. We also have the discrete form

$$m_{p,q} = \sum_{j=1}^{N}\sum_{k=1}^{N} x_j^p y_k^q f(x_j, y_k),$$

where f is a function taking non-zero values in the range $\{x_1, \ldots, x_N\} \times \{y_1, \ldots, y_N\}$ only. The *centroid* (x_0, y_0) of f is defined by $x_0 = (m_{1,0}/m_{0,0})$, $y_0 = (m_{0,1}/m_{0,0})$. If the image f is translated by a vector t, the centroid will also be translated by t. Thus the centralized image $f_T(x, y) = f(x + x_0, y + y_0)$ is translation-invariant:

The *central moments* are defined by:

$$v_{p,q} = \int_{-\infty}^{\infty}\int x^p y^q f(x + x_0, y + y_0)\, dx\, dy \tag{10}$$

Discrete central moments can be defined analogously.

As moments of the centralized image, central moments are translation-invariant.

We now define (N.B. $v_{0,0} = m_{0,0}$)

$$\mu_{p,q} = \frac{v_{p,q}}{v_{0,0}^{1+(p+q)/2}} \tag{11}$$

It can be shown that these *normalized moments* are also scale-invariants.

From these functions, Hu[30] derived seven moment invariants which are also translation-, scale- and also rotation-invariant. These are as follows.

$$\phi_1 = \mu_{2,0} + \mu_{0,2}$$

$$\phi_2 = (\mu_{2,0} - \mu_{0,2})^2 + 4\mu_{1,1}^2$$

$$\phi_3 = (\mu_{3,0} - 3\mu_{1,2})^2 + (3\mu_{2,1} - \mu_{0,3})^2$$

$$\phi_4 = (\mu_{3,0} + \mu_{1,2})^2 + (\mu_{2,1} + \mu_{0,3})^2$$

$$\begin{aligned}\phi_5 = {} & (\mu_{3,0} - 3\mu_{1,2})(\mu_{3,0} + \mu_{1,2})[(\mu_{3,0} + \mu_{1,2})^2 \\ & - 3(\mu_{2,1} + \mu_{0,3})^2] + (3\mu_{2,1} - \mu_{0,3})(\mu_{2,1} + \mu_{0,3}) \\ & \times [3(\mu_{3,0} + \mu_{1,2})^2 - (\mu_{2,1} + \mu_{0,3})^2]\end{aligned}$$

$$\begin{aligned}\phi_6 = {} & (\mu_{2,0} - \mu_{0,2})[(\mu_{3,0} + \mu_{1,2})^2 - (\mu_{2,1} + \mu_{0,3})^2] \\ & + 4\mu_{1,1}(\mu_{3,0} + \mu_{1,2})(\mu_{2,1} + \mu_{0,3})\end{aligned}$$

$$\phi_7 = (3\mu_{2,1} - \mu_{0,3})(\mu_{3,0} + \mu_{1,2})[(\mu_{3,0} + \mu_{1,2})^2 - 3(\mu_{2,1} + \mu_{0,3}{}^2] - (\mu_{3,0} - 3\mu_{1,2})(\mu_{2,1} + \mu_{0,3}) \times [3(\mu_{3,0} + \mu_{1,2})^2 - (\mu_{2,1} + \mu_{0,3})^2]$$

3.2. *Moments and Fourier–Mellin descriptors*

Li[16] has identified a relationship between Fourier–Mellin descriptors (see section 2.3.1) and normalized moments. Here $\hat{z}(l,u)$ denotes a normalized Fourier–Mellin descriptor defined by equations (6) and (7), however without the 2π factor:

$$\hat{z}(l,u) = \frac{1}{m_{0,0}^{u/2}} \int_0^\infty \int_0^{2\pi} f(r,\theta) e^{-il\theta} r^{u-1}\, d\theta\, dr,$$

with integral values of l and u for which $u - l$ is even.

We now denote by t the quantity $((u-l)/2) - 1$, which is an integer. Constraining p and q by the equation $p + q + 2 = l$, Li obtains the following law:

$$\hat{z}(l,u) = \hat{z}_r(l,u) - i\hat{z}_i(l,u),$$

$$\hat{z}_r(l,u) = \sum_{q=0,2,\ldots}^{u-2} \sum_{k=0,2,\ldots}^{l} (-1)^k \binom{l}{k} \binom{t}{\frac{(q-k)}{2}} \mu_{p,q}$$

$$\hat{z}_i(l,u) = \sum_{q=1,3,\ldots}^{u-2} \sum_{k=1,3,\ldots}^{l} (-1)^{(k-1)/2} \binom{l}{k} \binom{t}{\frac{(q-k)}{2}} \mu_{p,q}$$

Thus the normalized Fourier–Mellin descriptors are linear combinations of some normalized regular moments. Also, Sheng and Duvernoy[21] note that Hu's moment invariants are specific normalized Fourier–Mellin descriptors.

3.3. *Zernike moments*

Other sets of moments have been used. These include Legendre, rotational and complex moments. However the class of moments which have been shown to be superior to others are Zernike moments. These have been investigated by Perantonis and Lisboa[31] and by Khotanzad and Lu,[32] amongst others. The following summary is based on that of Perantonis and Lisboa.

The *Zernike polynomials* are defined by

$$R_{dm}(r) = \sum_{l=0}^{(d-|m|)/2} (-1)^l \times \frac{(d-1)!}{l![(d+|m|)/2 - 1]![(d-|m|)/2+1]!} r^{d-2l}$$

Any two of these polynomials $R_{d_1m_1}$ and $R_{d_2m_2}$ are orthogonal.

For each pair of values of m and non-negative d such that $d - |m|$ is even and non-negative, a *Zernike moment of order d* with repetition m is defined by

$$A_{dm} = \frac{d+1}{\pi} \int \int_{x^2+y^2 \le 1} f(x,y) k(d,m;x,y)\, dx\, dy,$$

where the kernel $k(d,m;x,y)$ expressed in polar coordinates is given by

$$k(d,m;r,\theta) = R_{dm}(r) e^{-im\theta}$$

The modulus of a Zernike moment is rotation-invariant. As suggested by Khotanzad and Hong,[33] translation and scale invariance can also be achieved by normalizing the image f first using regular moments; i.e.

$$f(x,y) \to f(x(m_{0,0}/\alpha)^{1/2} + m_{1,0}/m_{0,0}, y(m_{0,0}/\alpha)^{1/2} + m_{0,1}/m_{0,0}),$$

where α is a constant equal to the zeroth order regular moment of the normalized image.

3.4. *Generalized moments*

Shvedov, Shmidt and Yakubovich[34] have proposed the following as a generalization of the concept of moments, which they show (see Section 7.4) are a very general class of invariants.

Let $\mathcal{G}$ be a group acting on a manifold X, and T be an arbitrary representation of $\mathcal{G}$. Let $\mathcal{H}$ be the subgroup stabilizing some point in X. An *essential vector* v of the representation $T(g)$ is defined by the equation $T(h)v = v$ for all $h \in \mathcal{H}$. An *essential column* of the representation $T(g)$ is any column t_1 such that $t_1(g) = T(g)v$ for some essential vector v. The sum t of all essential columns of T is constant on each coset of $\mathcal{H}$ and therefore effectively a function on the space X. An *essential representation* T of $\mathcal{G}$ is one for which the mapping t is continuous and injective.

Let $\times$ denote the direct product (tensor product) of vectors. A vector made up of the maximum possible number of linearly independent components of the vector $\times^d u$ is called its *symmetrized d-th degree* and denoted by $Sym^d u$. We can define the symmetrized degree $Sym^d A$ of the matrix A by the formula

$$Sym^d(Au) = Sym^d(A) Sym^d(u) \quad \text{for any } u$$

We can now define the *generalized moments* $M_d(f)$ of some function f on X by the formula

$$M_d(f) = \int_X f(x) Sym^d t(x)\, d\mu(x),$$

where t is obtained from an essential representation T of $\mathcal{G}$. Here μ denotes a *relatively invariant measure* on the space X (see Section 7.1). Note that each generalized moment is a vector-valued function of f.

Shvedov *et al.* prove that transforming the function f by a group element g is equivalent to multiplying the generalized moment M_k by the matrix $\chi(g) Sym^d T(g)$, where χ, a complex one-dimensional representation of $\mathcal{G}$, is the weight of measure μ (see Section 7.1). We will state another result of Shvedov *et al.* concerning generalized moments in Section 7.4.

3.5. *Affine moment invariants*

To conclude the section on moments, we summarize the theory of Flusser and Suk,[35] in which a method of constructing invariants under arbitrary affine transformations of the plane is presented.

An affine transformation can be decomposed into six one-parameter transforms:

$$\begin{array}{llllll}
(1) & x\mapsto x+\alpha & (2) & x\mapsto x & (3) & x\mapsto \omega x \\
 & y\mapsto y & & y\mapsto y+\beta & & y\mapsto \omega y \\
(4) & x\mapsto \delta x & (5) & x\mapsto x+\zeta y & (6) & x\mapsto x \\
 & y\mapsto y & & y\mapsto y & & y\mapsto \eta x+y
\end{array}$$

Defining the central moments $v_{p,q}$ and normalized moments $\mu_{p,q}$ as in equations (10) and (11), we note that the function $\mu_{p,q} = v_{p,q}/v_{0,0}^{(1+(p+q)/2}$ is an invariant under transforms #1, #2 and #3.

We therefore assume that the invariant function F which we are trying to construct is of the form

$$F = \sum_k c_k v_{p_1}(k), q_1(k) \cdots v_{pn(k)}(k), q_{n(k)}(k)/v_{0,0}^{m(k)},$$

where

$$m(k) = \left(\sum_{j=1}^{n(k)} (p_j(k) + q_j(k)) \right) \Big/ 2 + n(k)$$

and our task is to find constraints on c_k, p_j, q_j and n that ensure the invariance of F under the whole affine group.

The requirements turn out to be as follows.

For invariance under transformation #4, we require

$$\sum_{j=1}^{n(k)} p_j(k) = \sum_{j=1}^{n(k)} q_j(k) \quad \forall k$$

For invariance under transformation #5, we require

$$\sum_{p=0}^{\infty} \sum_{q=0}^{\infty} p v_{p-1,q+1} \frac{\delta F}{\delta v_{p,q}} = 0$$

Similarly for invariance under transformation #6,

$$\sum_{p=0}^{\infty} \sum_{q=0}^{\infty} q v_{p+1,q-1} \frac{\delta F}{\delta v_{p,q}} = 0$$

Solving these equations gives us a complete set of constraints for invariance of this form under the affine group. Flusser and Suk used a set of three second and third order moment invariants in experiments on recognition of letters, geometric shapes and satellite images. In all cases the moment invariants performed well.

4. NEURAL NETWORK APPROACHES

In this section we discuss the use of neural networks[36–39] to produce group invariance. With the exception of those in the first section, the methods we will discuss rely on the presence of symmetries among the network connections. The techniques in this section use feedforward rather than recurrent network structures.

4.1. *Learned invariance*

One extremely simple (but not very efficient) method of using a neural network as an invariant pattern recognition system is to present to the network, during training, each input pattern in a number of positions in its group orbit. The interpolative abilities of the network should then allow the network to give the correct output for any pattern in the whole orbit. In effect, it can *learn* the invariance.

This scheme does not really fit into our current context of invariance very well, since there is no guarantee of the trained network's output being invariant. At best, it can only be transformation tolerant (though possibly to a sufficiently high degree of accuracy that the distinction becomes irrelevant). The crucial difference between this method and the majority of techniques described in this paper is that this method of learned invariance fails to incorporate the *a priori* knowledge of the problem into the recognition system. On the other hand, no complex modifications to the pattern classifier are required to adapt to the problem of invariance.

Another quite different approach to the use of neural networks for learning invariance is presented by Simard *et al.*[40] Their system, known as *Tangent Prop*, works by modifying the backpropagation algorithm so that the network's output is trained to be insensitive to infinitesimal transformations of the (continuous) invariance group.

4.2. *Weight sharing*

In a conventional neural network (feedforward or otherwise), each connection in the network has an individual weight, which is typically modified during training.

The principle of *weight sharing* is to force certain connections in the network to have the same weight. By doing this, certain constraints are effectively placed on the network's output, such as enforcing invariance under some given group. The number of free parameters in such a structured network is greatly less than that in an ordinary network of the same connectivity, so training of the network should be faster. Furthermore, since the constrained network has *a priori* knowledge of the learning problem incorporated into its structure, it will generalize better than the unconstrained network.

Rumelhart *et al.*[39] used weight sharing to solve the T–C problem, in which two simple T and C shapes have to be distinguished, independently of translation and 90° rotation. Learning algorithms, such as backpropagation,[39] can readily be adapted to enforce weight sharing.

Another way of using weight-sharing networks is to fix the weights at certain values, and to add a variable-weight standard network to the output nodes, which performs the learning. Effectively, this is performing an invariant feature extraction followed by pattern classification, all within the context of a neural network. We will not consider approaches of this kind in this section, since many possible methods of invariant feature extraction are covered elsewhere in the paper.

Le Cun[41] has described a generalization of weight sharing which he calls *Weight Space Transformation*. The idea of Weight Space Transformation is that each connection's weight is described as a function of some parameters, which are modifiable by a training algorithm. The intention of this technique is again to improve learning speed and generalization.

We now discuss a number of neural network approaches to the invariant pattern recognition problem, all of which use weight sharing to produce their invariance.

4.3. *Higher order neural networks*

Higher order neural networks[42] have been used by many researchers for solving the problem of invariant pattern recognition. Each neuron in a higher order network computes a weighted sum of products of its inputs. The maximum degree of an allowable product is called the *order* of the neuron. The complexity of a neuron is such that only a single neuron network is usually used in applications of invariant pattern recognition. Such a one node network can learn easily and quickly through an algorithm similar to the perceptron convergence algorithm.

Let $\mathscr{P}$ denote the power set of the set $\{1,\ldots,n\}$. Formally, a *higher order neuron of degree m* with inputs $(x_1, x_2, \ldots, x_n)$ has the functionality

$$F(x) = \psi\left(\sum_{S\in\mathscr{P},|S|=m} w_s \prod_{j\in S} x_j\right)$$

or alternatively

$$F(x) = \psi\left(\sum_{S\in\mathscr{P},|S|\leq m} w_s \prod_{j\in S} x_j\right),$$

where the w_s's are weight, ψ denotes some activation function and $F(x)$ denotes the output of the neuron. Note that we depart, for the sake of convention, from our usual notation for an input pattern, using x instead of f. In this section we will restrict our consideration to second and third order networks.

4.3.1. *Second order neural networks.* The output of a single node second order network is given by

$$F(x) = \psi\left(\sum_{j,k} w_{jk} x_j x_k\right),$$

where the sum is taken to be over all pairs of distinct indices.

Such a network can be made invariant under translation and scaling over a two dimensional input grid. Each input x_j can be identified with a point in this grid; hence each pair of inputs (x_j, x_k) can be identified with a line in the grid. The invariance is achieved by enforcing w_{jk} to be equal to w_{rs} whenever the pair (x_j, x_k) defines a line of the same gradient as that defined by the pair (x_r, x_s).

With this weight sharing in place, the output of the network is guaranteed to be invariant under translation and tolerant to scaling. This is because such a transformation permutes the pairs (x_j, x_k) within their parallel line classes, thus having no effect on the weighted sum which determines $F(x)$. Alternatively, second order networks can be made invariant under translation and tolerant to rotation. This works by sharing two weights w_{jk} and w_{rs} whenever their associated coordinate pairs define lines of the same length.

Invariance (as opposed to tolerance) under scaling or rotation can only be approximate due to the inaccuracies of the pixel grid representation. Invariance is improved by increasing the resolution of the input, and by allowing an error factor in the calculation of line gradient when deciding on weights to be shared.

Another problem with scale invariance is that a scaling operation will increase or decrease the number of active pixels in the grid, and will thus change the number of active pairs (x_j, x_k). Kanoaka *et al.*[43] have proposed an approach (which they applied to a third order network) in which only feature points (e.g. corners and intersections of edges) are activated for any given pattern.

Second order neural networks have also been used for cyclic translation invariance (see Duren and Peikari[44]), and together with a complex-log mapping for rotation and scale invariance.[45,46]

4.3.2. *Third order neural networks.* Third order networks are normally used in preference to second order, because they lend themselves naturally to invariance under translation, rotation and scaling simultaneously. The output of a single node third order network is given by

$$F(x) = \psi\left(\sum_{j,k,l} w_{jkl} x_j x_k x_l\right),$$

where again all indices j, k, l are taken to be distinct.

In a third order neuron, the triples (x_j, x_k, x_l) are identified naturally with triangles in the input domain. Weight sharing is enforced among triples corresponding to similar triangles. This allows exact translation invariance, and approximate rotation and scale invariance. The degree of invariance can again be improved by allowing weight sharing among "almost similar" triangles as well as by increasing input resolution. Again there is also a problem as with second order networks of scaling operations changing the number of active points.

Third order neural networks have also been used by Perantonis and Lisboa,[31] Spirkovska and Reid[47] and Takano *et al.*[48] amongst others. Generally such networks perform very well with translated and/or rotated input patterns, and acceptably well under scaling. We would like to repeat the point of Delopoulos *et al.*,[27] that a third order neuron is performing a function based on triple order correlation (see Section 2.5).

Coarse Coding. One of the main problems in using third order neural nets is the combinatorially ($O(n^3)$) large number of weights required. This often forbids use of these networks in practical applications.

An approach to get round this problem has been proposed by Spirkovska and Reid.[47] This involves *coarse coding* of the input grid. Essentially, the input grid is projected onto a set of partially overlapping

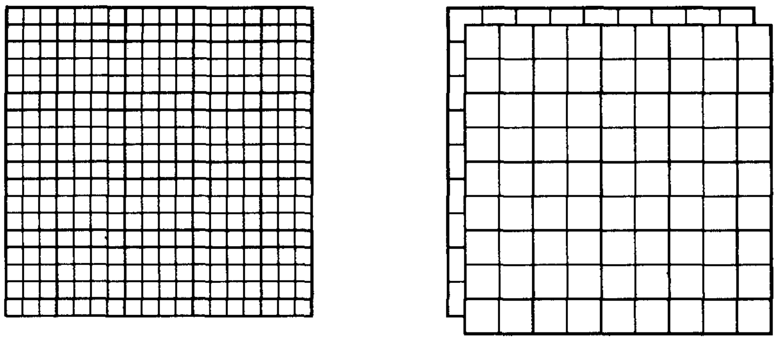

Fig. 1. Coarse coded input grids.

coarser grids (see Fig. 1). Thus each active pixel in the original grid (except for pixels around the edge) causes one pixel in each of the coarse grids to be activated. A computational neuron will then have a connection from every triple of input pixels within each coarse grid. This method can dramatically reduce the number of connections required.

4.4. *Symmetric first order networks*

In this section, we consider firsr order neural networks, which can be used to produce invariance by weight sharing in a similar manner to that used in higher order networks. With the practical exception of the neocognitron, these networks can be trained by standard connectionist algorithms such as backpropagation.[39]

4.4.1. *A rotation-invariant first order network.* Fukumi *et al.*[49] have proposed the following network for rotation invariance in two dimensions. Each input image is represented by a discrete polar-coordinate grid. Invariance is achieved under a finite, though arbitrarily large, rotation group. This works by constructing a first hidden layer of nodes, each of which is connected to the input with weights that are shared among pixels with the same radius from the image center. The network structure from the first layer upwards has no weight sharing, because the output of the first hidden layer is an invariant feature vector.

Fukumi *et al.* used their network in a simple experiment on coin recognition, in which it performed very well. A disadvantage of this network is that it discriminates imperfectly. This is because the invariant features calculated by the first hidden layer are functions of the total activation in each radial circle of pixels. An arbitrary rearrangement of the pixel intensities within any given circle would have no effect on the network output.

4.4.2. *A network for digit recognition.* Le Cun *et al.*[50] have used a symmetric feedforward network for handwritten zip code recognition. The layers in their network tend to learn hierarchies of features (with higher level features being represented in higher layers). The nodes in a given hidden layer are divided into two-dimensional grids known as "feature maps".

The nodes in a given map take input from some of the maps in the preceding layer. Specifically, each node takes input from a localized "window" of neurons in each of the preceding maps to which it is connected. Furthermore, the weight of the connection from a given node A to node B is a function solely of the position of A in the input window of node B. Thus the connections between any given pair of connected maps will share weights.

A sigmoid function is used as activation function throughout the network; however each node has a different threshold. The network was translation tolerant, and performed excellently on a large training and test set. We find it strange that the neuron thresholds are not fixed across a given feature map, since this would improve the degree of translation tolerance.

4.4.3. *Fukushima's neocognitron.* The neocognitron was developed by Fukushima.[51,52] An extensive analysis of the neocognitron appears in Lovell's thesis,[53] and another summary is given by Fausett.[36] Our description is primarily obtained from these two sources. **Network Architecture.** Essentially the neocognitron is a fixed-architecture network with thousands of neurons and hundreds of thousands of connections, some of which are fixed. The network structure is highly similar to the one we have just described of Le Cun *et al.*

The nodes are organized into layers, which (after the input layer), alternate between *S-layers* and *C-layers*. The layers are also divided into planes called *S-planes* and *C-planes*. The neurons themselves are called *S-cells* and *C-cells*. There are also *V-cells*, which are inhibitory and occur in single planes per *S-layer*, but we will omit a discussion of these. The prefixes S and C stand for *simple* and *complex*, and derive from biological cells.

Each S-plane is trained to respond to a particular feature. The S-planes in the lower layers are trained with lower level features. Two planes in the same layer are trained to respond to similar but distinct features. Each node in a given plane picks out the same feature, though in a different position. Each S-node is connected to a window of neurons in each of the immediately preceding C-planes. The weights are variable and are a function of the position of the C-cell within the S-cell's receptive input window, as in the network of Le Cun *et al.* Thus weights are shared between S-cells in the same plane. The output of an S-cell will effectively be a measure of similarity between the input and corresponding weights.

A given C-plane takes input from either one or two S-planes in the preceding layer (in the latter case, the features picked out by those S-planes will be similar and one function of the C-plane is to combine them). A given C-cell has an input window as of the manner of a S-cell. Input windows to both S-cells and C-cells overlap each other. The weights leading into a C-cell are fixed, positive and are strongest in the center of the window of attention. This results in the C-plane appearing to "blur" its S-plane input.

Due to its use of weight sharing, the neocognitron is partially invariant under translation [see Barnard and Casasent[54]]. We will not discuss the exact functionality of the necognitron, since it is irrelevant to our current concern of invariance.

Neocognitron Training. The neocognitron is designed to be trained by either supervised or unsupervised learning. The unsupervised version behaves less successfully but is more biologically plausible.

In the supervised learning case, the network is trained from the first hidden layer upwards. Each S-*plane* in the hidden layers has to be identified by the teacher with a visual feature, which it must try to learn. When training a given S-plane, the cell in the center of a given plane is designated as the "seed cell" to be updated by the Hebbian rule. Weight sharing is again constantly enforced. The neocognitron can be trained in this way far more efficiently than by using conventional neural network learning algorithms such as backpropagation.

4.4.4. *Time-Delay Neural Networks.* Time Delay Neural Networks have been used in speech processing[55,56] to recognize patterns independent of their occurrence in time; i.e. with invariance under the one-dimensional translation group. These feedforward networks use weight sharing to achieve invariance in a similar but more basic way to that described for the neocognitron above.

4.4.5. *Symmetry Networks.* The Symmetry Network was introduced by Shawe-Taylor,[57,58] based on the Group Invariance Theorem of Minsky and Papert.[38] Essentially, a Symmetry Network is a feedforward network, the output of which is invariant under some given group $\mathcal{G}$ due to weight sharing among its connections. $\mathcal{G}$ can be any finite group acting by permutation on the network's inputs.

A Symmetry Network is constructed by identifying each layer of the network with a subgroup of $\mathcal{G}$ [the output node(s) being each identified with the improper subgroup $\mathcal{G}$, and the subgroup associated with the input layer being determined by the target problem]. Each node in a given layer is then associated with a coset of the corresponding subgroup, and the nature of the weight sharing in the network's connections is determined by group-theoretic principles. In particular the threshold must be the same for all neurons in the same hidden layer. Activation functions must also be the same for all neurons in the same hidden layer, though there are no other restrictions on them.

The structure of a Symmetry Network is such that if transformation $g \in \mathcal{G}$ is applied to the input then the outputs of the nodes of each hidden layer are permuted according to the action of g on the cosets of the corresponding subgroup. Symmetry Networks have been applied with considerable success to the problem of graph isomorphism.[58] The same paper addresses the issue of discriminability of Symmetry Networks, this property being dependent upon the network structure chosen.

The rotation-invariant network of Fukumi *et al.* (see Section 4.4.1) is a Symmetry Network in which each computational node has been identified with the whole group $\mathcal{G}$. The network used by Le Cun *et al.*, the Time-Delay Neural Networks and the neocognitron are all close relatives of the Symmetry Network. Their inclusion in the Symmetry Network model is inexact since the translations under which these networks are transformation-tolerant do not form a group over the space on which the inputs are defined.

The second order networks used for cyclic translation invariance [see[44]] can also be implemented inside a Symmetry Network, assuming activation functions such as $\psi(x) = x^2$ are allowed, by virtue of the formula $x_j x_k = \frac{1}{2}((x_j + x_k)^2 - x_j^2 - x_k^2)$.

5. MISCELLANEOUS APPROACHES

The techniques described in this section are varied and do not fit naturally into the framework of the proceeding sections.

5.1. *Image normalization*

Yuceer and Oflazer[59] have proposed the following preprocessing technique for normalizing an input image in a rotation-, translation- and scale-invariant manner. We denote by f the input pattern, which is assumed to be a binary-valued function of the pixel grid $\{1, \ldots, N\} \times \{1, \ldots, N\}$.

The first step is to achieve translation invariance. This works by computing the center of gravity of the input pattern and translating the origin to that point. In other words, given the input pattern $f(x, y)$ and letting $m_{j,k}$ denote the discrete regular moments defined by equation (9) in Section 3.1, we have

$$f_T(x,y) = f\left(x + \frac{m_{1,0}}{m_{0,0}}, y + \frac{m_{0,1}}{m_{0,0}}\right) \quad \forall x,y \in 1 \cdots N$$

This was presented earlier, in the section on moments.

The next step is to normalize the image so that the average radius r_0 for the activated pixels is a quarter of the input grid dimension, i.e. we have

$$r_0 = \frac{1}{m_{0,0}} \sum_{j=1}^{N} \sum_{k=1}^{N} f_T(x_j, y_k) \cdot \sqrt{(x_j^2 + y_k^2)}$$

$$f_{TS}(x,y) = f_T\left(\frac{4r_0}{N}x, \frac{4r_0}{N}y\right) \quad \forall x,y \in 0 \cdots N-1$$

The function f_{TS} is now translation- and scale-invariant. This normalization process is similar to that used by Perantonis and Lisboa for use with Zernike moments (see Section 3.3).

The rotation invariance is achieved by rotating the image so that the direction of maximum variance coincides with the x-axis. The direction of maximum variance is given by the eigenvector corresponding to the largest eigenvalue of the covariance matrix of the set of vectors for activated pixels. Yuceer and Oflazer derive an expression for the required angle of rotation, thus obtaining a function f_{TSR} which is invariant under all three kinds of transformation. One minor problem with this scheme is that there are always two directions of maximum variance (in opposite directions). Hence two normalized images (rather than just one) have to be used to define each input orbit.

A more serious problem is that the method as a whole is not particularly efficient; in particular it requires two passes over the input image. To alleviate this problem with speed, Yuceer and Oflazer proposed a modification to the above algorithm which involves reversing the order of obtaining scale and rotation invariance. This produced slightly less reliable results, but requires only a single pass over the input image.

5.2. *Symmetric filter systems*

Lenz has introduced the following system[60,61] for invariant feature extraction. Let the group $\mathscr{G}$ act on the set X, and let V denote the Hilbert space of complex-valued square-integrable functions defined on X. Thus $\mathscr{G}$ has a representation R given by its action on V, which derives in turn from its action on X. We take the representation R to be unitary. Let $f \in V$ denote a given input pattern.

The feature extractor performs a series of linear transformations by taking inner products with *filter functions* ω_i to produce a feature vector Ω:

$$\Omega(f) = (\omega_j(f))_{j \in I},$$

where the index set I is generally taken to be finite.

The filter functions $\omega_1, \omega_2, \ldots, \omega_m$ are said to define a *$\mathscr{G}$-symmetric filter system* if the ω_j's are orthonormal and the subspace they span is a module under $\mathscr{G}$. When the invariance group is a rotation group we refer to a *rotationally symmetric filter system.* Henceforth assume that the filter functions form a $\mathscr{G}$-symmetric filter system. Now let T be the representation of $\mathscr{G}$ defined by the group's action on the filter functions. We find that $\Omega(gf) = T(g^{-1})\Omega(f)$, and that T is also a unitary representation. Thus the magnitude of the feature vector $\Omega(f)$ will be the same for all patterns in an orbit. Lenz therefore proposes to use this quantity as an invariant feature.

For the specific case of a rotation group, we have the following result:[60]

Theorem 5.1. *Any complete, orthonormal set of eigenfunctions of the Fourier Transform belonging to one eigenvalue, form a rotationally symmetric filter system.*

This theory was applied to a problem of distinguishing two faces, invariant under the two-dimensional rotation group. The faces were distorted with varying amounts of noise but were all correctly classified by Lenz's system.

5.3. *A preprocessing method for cyclic invariance*

Mas and Ramos[62] proposed the following scheme for preprocessing of binary inputs to produce a set of features invariant under cyclic translation. The main advantage of this system is its efficiency, arising from its use of binary operations.

(1) Let x denote the input vector of dimension n. For each $k \in \{1, \ldots, n/2\}$, we compute a n-tuple y^k by

$$y_j^k = x_j \oplus x_{(j+k \bmod n)} \quad (j \in 1 \cdots n),$$

where $\oplus$ denotes addition modulo 2.

(2) We now compute an $(n/2)$-dimensional vector z ($(n-1)/2$-dimensional when n is odd), where z_k is the sum (not modulo 2) over all j of y_j^k.

(3) We finally perform a thresholding operation on z (with threshold $n/2$) to obtain a binary vector $\hat{x}$; i.e. $\hat{x}_j = 1$ if $z_j \geq n/2$ and $\hat{x}_j = 0$ otherwise.

The vector $\hat{x}$ is invariant under cyclic shifts of x. This is because such a shift in the input vector induces the same translation in each y^k vector. Thus the sum z_k of the components of y^k for a given k is invariant as desired. This system unfortunately suffers from discriminability problems. For example, it is also invariant under reversal of x and under simultaneous inversion of all bits in x.

Mas and Ramos used this feature vector as a set of invariant features for input into a pattern recognition system (specifically a Hopfield network). We would like to make the point, however, that the preprocessing described above can be calculated within a sparsely-connected feedforward network with set weights.

This network would have one hidden layer to compute each y^k vector. The output layer would compute the vector $\hat{x}$. All weights would be set to 1 or 0. We assume that $\oplus$ is allowed in the hidden layers instead of normal addition; alternatively, $\oplus$ can be replaced with $+$ without affecting the invariance of the output vector $\hat{x}$ (actually, this would improve the discriminability). We would also like to point out that the network just described is a Symmetry Network (see Section 4.4.5) with fixed weights of 1 and 0. Any choice of activation functions and weight assignments within this Symmetry Network would also produce an invariant set of features. Alternatively, the network could be trained with constrained weight sharing to perform some desired classification function.

5.4. *Fast translation–invariant transforms*

Another technique for achieving cyclic translation invariance is that of *fast translation–invariant transforms*, introduced by Wagh and Kanetkar;[63] see also Moharir.[64]

These are transforms which can be computed using the scheme illustrated in Fig. 2. Here the leftmost set of nodes are the input nodes, and the information flow is from left to right. All connections in the diagram have a weight of 1, and ψ_1 and ψ_2 can be any commutative functions of their two inputs.

The output of a fast translation–invariant transform is (not surprisingly!) invariant under cyclic translation. This can be seen in the example in Fig. 2. A cyclic permutation in the input layer of the diagram induces a pair of permutations in the two separate sets of nodes in the second layer. This in turn induces a permutation in each of the four sets of nodes in the next layer, and finally a (trivial) permutation in the eight singleton sets of nodes in the output layer. The *R-transform* is defined by the substitution $\psi_1(x_1, x_2) = x_1 + x_2$, $\psi_2(x_1, x_2) = |x_1 - x_2|$. The *M-transform* is defined by $\psi_1(x_1, x_2) = max(x_1, x_2)$, $\psi_2(x_1, x_2) = min(x_1, x_2)$. The power spectrum of the Walsh–Hadamard transform can also be expressed as a fast translation–invariant transform. Duren and Peikari[44] define the *fast correlation transform* by the substitution $\psi_1(x_1, x_2) = x_1 + x_2$, $\psi_2(x_1, x_2) = x_1 x_2$. They also prove that the fast correlation transform discriminates better than the *R*-transform. The subclass of fast translation–invariant transforms for which ψ_1 and ψ_2 are functions of the sum of their inputs can be readily implemented within a feedforward neural network. Again we find that such a network is a Symmetry Network (see Section 4.4.5) with fixed weights of 1 and 0. Furthermore, if we extend our neural network model to allow the neuron to compute any commutative function of its weight inputs (rather than just their sum), then we see that we can construct any fast transform within a Symmetry Network.

One unfortunate aspect of the fast translation–invariant transforms is that they do not discriminate fully; in particular they show invariance under dyadic translation (i.e. bitwise addition of an integer $0 \cdots n-1$ to all input indices, where n is the number of inputs).

5.5. *Continuous transformation reconstruction*

The following approach was suggested by Fayn, Sorokin and Vaynshteyn.[9]

Assume that patterns with the same classification are assumed to be images of each other under some continuous (or piecewise-continuous) transformation. Characteristic points are picked out from a test pattern; these could be local extrema of the pattern. The relationship between these and the characteristics of each given prototype pattern then allows a piecewise-continuous transformation to be constructed which maps from one set of characteristic points to the other. Then the calculated transform can be applied to the test pattern, and the result compared to the prototype pattern.

The classification of a given test pattern is given by that of whichever prototype pattern is matched most closely with the transformed test pattern, assuming that there is a prototype with a sufficiently high correlation.

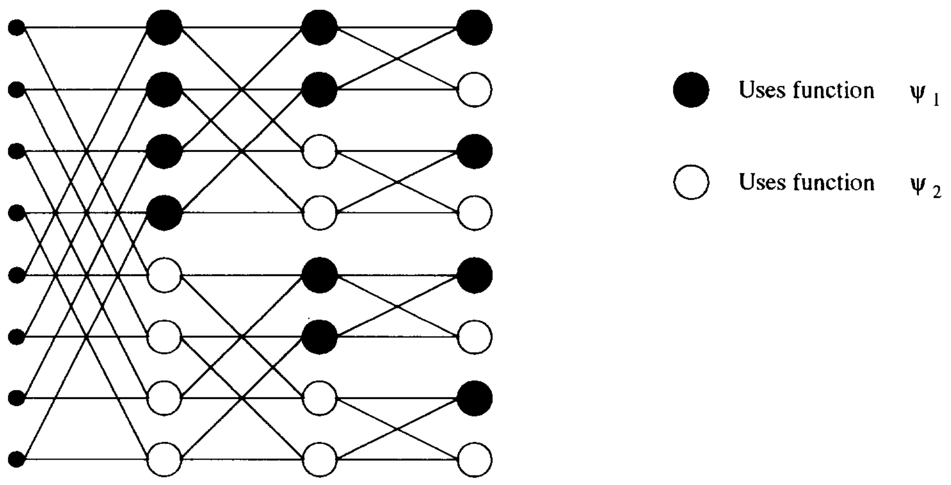

Fig. 2. Fast translation–invariant transform.

The theory described was used in a variety of complex visual and auditory recognition experiments; the results of these being extremely good.

6. PRACTICAL COMPARISONS

Numerous comparisons have been made between different methods of invariant pattern recognition by many researchers. We summarize some of their findings below.

6.1. *Invariance under translation, rotation and scaling*

It is widely accepted that Zernike moments outperform other kinds of moment (including regular moments), especially when noise is added.(31,32) Regular moments are highly noise-sensitive. Another problem with moment invariants in general is that their evaluation is not very efficient.

In contrast, one of the main advantages of Fourier-based methods is that they can be applied efficiently using Fast Fourier Transform techniques. Fourier–Mellin descriptors perform well under noise,(15) but unlike moments are not translation–invariant. However there is no reason why the input pattern could not be centralized before the Fourier–Mellin descriptors are calculated. Alternatively, a hybrid scheme such as that suggested by Casasent and Psaltis(23) could be used to produce invariance under all three transformations.

Higher-order neural networks performed better than a Zernike moment classifier in experiments done by Perantonis and Lisboa.(31) Spirkovska and Reid(47) demonstrated a third order network's remarkable tolerance to occlusion of the input pattern. However, higher order networks do suffer from problems of noise sensitivity.(31,47) There is also a severe problem with the amount of computational space they require, particularly in the case of third order networks, though this can be alleviated using methods such as coarse coding.(47) A major advantage of higher order networks is that they can be trained quickly.

Dobnikar *et al.*(65) used a neural network with both feedforward and recurrent parts to learn a set of rotated and scaled TETRIS shapes. This was compared to an approach in which the input was scale-normalized and a Discrete Fourier Transform applied to the sampled contour to obtain rotation invariance. The neural network performed better both with and without noise distortion of the inputs.

The rotation-invariant first order network used by Fukumi *et al.*(49) achieved 100% correct recognition rates on a problem of distinguishing two coins (either face up). It does however have a problem with discriminability (see Section 4.4.1).

The neocognitron,(51) though an immense network, can be trained quickly by a supervisor and has performed reasonably well on digit recognition.(53) Considerably better results have however been obtained by Le Cun *et al.*(50) on the smaller network described in Section 4.4.2, which was trained by the slower method of backpropagation. It is worth noting that neither of these networks was designed with invariance as the principle objective, and neither is truly invariant under translation.

6.2. *Invariance under discrete translation*

The preprocessing method used by Mas and Ramos(62) (see Section 5.3) possesses poor discriminability but is efficient to calculate. It tends to behave well under structured noise (e.g. block permutations) but badly under more random errors.

In Duren and Peikari's paper,(44) it is pointed out that both second order neural networks and Discrete Fourier Transforms can be used to produce linear (as opposed to cyclic) discrete translation invariance. In the DFT case, this works by appending a sequence of zeros to the end of any input string. It is also shown that such a method cannot work for any fast translation–invariant transform. This results from the imperfect discriminability properties of this class of transforms.

Efficiency of processing is another important consideration. Whilst the Fast Fourier Transform is more efficient than the evaluation of the output of a second order network, it requires complex-valued arithmetic. Second order neurons, on the other hand, could be implemented using integer arithmetic or in certain cases merely binary.

We could also design Symmetry Networks(58) for invariance under discrete translation groups. The efficiency and discriminability (amongst other properties) of such networks would depend upon their precise structure.

7. THE FORM OF INVARIANTS

In this section we give definitions and results from several papers on the existence and structure of invariants under certain groups. This section is more mathematically formal than those preceding, and is largely extracted from the papers of Shvedov *et al.*(34) Dunn(66) and Newman.(67)

7.1. *Invariants and relative invariants*

Let $\mathscr{G}$ denote a group and V a finite-dimensional module under $\mathscr{G}$.

Any function $\mu: V \mapsto \mathbb{R}^n$ for some n is called a *measurement function*.

A measurement function μ is called *relatively invariant* with *weight* (or *modulus*) χ if

$$\mu(gv) = \chi(g)\mu(v) \quad \forall g \in \mathscr{G}, v \in V$$

It can be proved that χ is a representation of $\mathscr{G}$ in $GL(n, \mathbb{R})$.

A relatively invariant measurement μ is called *invariant* if its weight χ is trivial, i.e.

$$\mu(gv) = \mu(v) \quad \forall g \in \mathscr{G}, v \in V$$

When considering the action of the group on itself (i.e.

$V = \mathcal{G}$), a distinction is made between a *left invariant* measure as defined above, and a *right invariant* measure, for which

$$\mu(g_1 g) = \mu(g_1) \quad \forall g, g_1 \in \mathcal{G}$$

The left and right invariant measurer on $\mathcal{G}$ with range $\mathbb{R}$, when they exist, are called *Haar measures.*

7.2. *Complete systems of invariants*

Let $\{I_1, I_2, \ldots\}$ denote a set (not necessarily finite) of invariants under the action of the group $\mathcal{G}$ on the vector space V. Then from Shvedov *et al.*[34] we have the following definition:

This set is called a *complete system of invariants* if the following property holds: for every $\varepsilon > 0$, there exists an integer N such that if $I_k(f_1) = I_k(f_2)$ for $k = 1 \cdots N$ and for some $f_1, f_2 \in V$, then there exists $g \in \mathcal{G}$ with $|f_1 - g f_2| < \varepsilon$.

Informally, this says that a complete system of invariants must be able to distinguish with arbitrary accuracy between any two vectors not in the same orbit under $\mathcal{G}$; i.e. the system must possess perfect discriminability.

7.3. *The invariant integral*

Invariants under a finite group can be constructed by averaging some suitable function of the group over the whole group. To be useful, the function must be chosen so that its mean value over the group is not constant. The concept of group averaging can be generalized to certain continuous groups by the introduction of an invariant integral.[6,61,68] The following definition is based on that of Newman.[67]

Let $\mathcal{G}$ denote a topological and compact group, τ a mapping from the group into $\mathbb{R}^n$ for some n and let μ denote the left Haar measure. The mean value of the function τ is defined by the equation

$$M(\tau) = \frac{1}{\mu(\mathcal{G})} \int_{\mathcal{G}} \tau(g)\, d\mu(g)$$

The integral used in the above definition is called an *invariant integral* or *Haar integral.* Newman's definition is a generalization to the case of a locally compact group. Let $L(V)$ denote the set of functions which have a mean as defined by the above equation. Newman proves that M is an invariant linear transform of $L(V)$ into $\mathbb{R}^n$. He also presents a similar result for relative invariants.

As a special case of another result by Newman, we have the following:

Let R denote a measurement function from V to $\mathbb{R}^n$ such that for all $f \in V$ the function $f^r : \mathcal{G} \mapsto \mathbb{R}^n$ defined by $f^r(g) = R(g^{-1} f)$ is in $L(V)$. Now the mean of f^r is an invariant measure, and furthermore every invariant is of this form.

7.4. *Existence of invariants*

Shvedov *et al.*[34] have shown that a complete set of continuous invariants under a given representation of a given group does not always exist. The example they use is that of the one-dimensional projective group acting on two specific functions which are in distinct orbits. The sum of these functions, however, can be transformed arbitrarily closely to either function. The lack of a continuous and complete set of invariants follows from this. In the same paper are presented the following results, which relate to the generalized moments and essential representations defined earlier (see Section 3.4).

Theorem 7.1. *Let $\mathcal{G}$ be a connected finite-dimensional real Lie group, X a manifold on which $\mathcal{G}$ acts continuously and V a normed space of functions on X on which $\mathcal{G}$ also acts continuously through its action on X. Furthermore, let μ denote a relatively invariant measure on X and T an essential representation of $\mathcal{G}$. Then there exists a complete system of invariants $\{I_1, I_2, \ldots\}$, $I_k : L \mapsto \mathbb{R}$, each of which has the form*

$$I_k(f) = F_k[M_{d_1}(f), M_{d_2}(f), \ldots, M_{d_{n(k)}}(f)],\ n(k) < \infty, \tag{12}$$

where the F_k's are arbitrary functions and the M_d's are the generalized moments defined by

$$M_d(f) = \int_X f(x) Sym^d t(x)\, d\mu(x)$$

(see Section 3.4 *for an explanation of this formula)*

Theorem 7.2. *If a complete system of invariants of the form in equation* (12) *exists, where the F_k's are arbitrary functions and the M_i's are linear, then there also exists a relatively invariant measure on the manifold X and an essential representation of the invariance group.*

Finally, Richardson has derived some results[69] regarding the construction of invariant functions and functionals under Lie groups acting on a two-dimensional input space. Also, the number of elementary invariant functions (i.e. invariants in terms of which any other invariant can be written) under such groups is discussed.

8. SUMMARY

We have reviewed the main techniques used to obtain invariance in pattern recognition, and we have related some of these methods to each other. These methods include the use of Fourier Transform-based power spectra or algebraic moments as invariant features, and the design of neural networks with structured weight sharing. The available techniques differ widely in their simplicity and flexibility. In particular they can be divided into relatively straightforward mechanisms for producing invariance under a specific group, and highly general but less simple methods which can be applied to larger classes of invariance problems. There are also such issues as computational time and space, noise tolerance and discriminability to take into account. Some of the techniques we have discussed have particular strengths and weaknesses in some of these areas, and one important thing for any experimenters in the field to consider must be which of these qualities are of most and which of least importance to their particular problems.

Acknowledgement—I would like to thank my supervisor, John Shawe-Taylor, for supporting me in this work and for reading through and advising me upon this document.

REFERENCES

1. M. Kargapolov and Lu. Merzljakov, *Fundamentals of the Theory of Groups.* Springer-Verlag, New York (1979).
2. W. Ledermann, *Introduction to Group Characters.* Cambridge University Press, (1977).
3. W. Scott, *Group Theory.* Prentice-Hall, Englewood Cliffs (1964).
4. C. Chevalley, *Theory of Lie Groups.* Princeton University Press (1946).
5. W. Fulton and J. Harris. *Representation Theory (A First Course).* Springer-Verlag, New York (1991).
6. R. Gilmore, *Lie Groups, Lie Algebras and some of their Applications.* Wiley-Interscience (1974).
7. Y. Matsushima, *Differentiable Manifolds. Pure and Applied Mathematics.* Marcel Dekker Inc., New York (1972).
8. C. Webber, Self-organization of transformation-invariant neural detectors for constituents of perceptual patterns, *Network: Computation in Neural Syst.* **5**, 471–496 (November 1994).
9. V. Fayn, V. Sorokin and V. Vaynshteyn, Continuous-group pattern recognition, *Engng Cybern.* **6**, 97–106 (1969).
10. T. M. Caelli and Zhi-Qiang Liu, On the minimum number of templates required for shift, rotation and size invariant pattern recognition, *Pattern Recognition* **21**, 205–216 (1988).
11. R. N. Bracewell, *The Fourier Transform and Its Applications.* Electrical and Electronic Engineering Series, McGraw-Hill Book Company, (1978).
12. E. Brigham, *The Fast Fourier Transform and Its Applications.* Prentice-Hall (1988).
13. M. Clausen and U. Baum, *Fast Fourier Transforms.* Wissenschaftsverlag, Mannheim (1993).
14. H. Wechsler, Invariance in pattern recognition, *Advances in Electronics and Electron Physics*, R. W. Hawkes, ed., Vol. 69, pp. 262–322. Academic Press (1987).
15. A. Grace and M. Spann, A comparison between Fourier–Mellin descriptors and moment based features for invariant object recognition using neural networks, *Pattern Recognition Lett.* **12**, 635–643 (1991).
16. Y. Li, Reforming the theory of invariant moments for pattern recognition, *Pattern Recognition* **25**, 723–730 (1992).
17. Y. Sheng and H. H. Arsenault, Experiments on pattern recognition using invariant Fourier–Mellin descriptors, *J. Optical Soc. America A (Optics and Image Sci.)*, **3**, 771–776 (June 1986).
18. R. Wu and H. Stark, Three dimensional object recognition from multiple views, *J. Optical Soc. America A (Optics and Image Sci.)*, **3**, 1543–1557 (1986).
19. J. L. de Bougrenet de la Tocnaye and F. Ghorbel, A rotation, scaling and translation invariant pattern classification system, *Pattern Recognition Lett.* **8**, 55–58 (July 1988).
20. S. Kageyu, N. Ohnishi and N. Sugie, Augmented multilayer perceptron for rotation-and scale- invariant handwritten numeral recognition. *Proceedings of* 1991 *IEEE International Joint Conference on Neural Networks*, **1**, pp. 54–59. IEEE (1991).
21. Y. Sheng and J. Duvernoy, Circular Fourier radial Mellin transform descriptors for pattern recognition, *J. Optical Soc. America A (Optics and Image Sci.* **3**, 885–888 (1986).
22. Y. Sheng and C. Lejeune, Invariant pattern recognition using Fourier–Mellin transforms and neural networks, *J. Optics* **22**, 223–228 (1991).
23. D. Casasent and D. Psaltis, Position, rotation and scale-invariant optical correlation, *Appl. Optics* **15**, 1795–1799 (1976).
24. D. Casasent and D. Psaltis, Hybrid processor to compute invariant moments for pattern recognition, *Optics Lett.* **5**, 395–397 (1980).
25. D. Pintsov, Invariant pattern recognition, symmetry and Radon transforms, *J. Optical Soc. America A*, Vol. 6(10), 1544–1554 (October 1989).
26. S. Helgason, *The Radon Transform.* Birkhauser, Boston, (1980).
27. A. Delopoulos, A. Tirakis and S. Kollias, Invariant image classification using triple-correlation-based neural networks. *IEEE Trans. Neural Networks*, **5**, 392–408 (May 1994).
28. M. Ferraro and T. M. Caelli, Relationship between integral transform invariances and Lie group theory, *J. Optical Soc. America A*, **5**, 738–742 (May 1988).
29. R. P. Srivastava, Transformation and distortion tolerant recognition of numerals using neural networks, *Proc.* 1991 *ACM Computer Science Conference.* pp. 402–408 ACM, New York (1991).
30. M. K. Hu, Visual pattern recognition by moment invariants, *IEEE Trans. Inform. Theory* **8**, 179–187 (1962).
31. S. Perantonis and P. Lisboa, Translation, rotation and scale invariant pattern recognition by high-order neural networks and moment classifiers, *IEEE Trans. Neural Networks* **3**, 241–251 (March 1992).
32. A. Khotanzad and J.-H. Lu, Classification of invariant image representations using a neural network, *IEEE Trans. Acoustics, Speech* and *Signal Process.* **38**, 1028–1038 (June 1990).
33. A. Khotanzad and Y. H. Hong, Invariant image recognition by Zernike moments, *IEEE Trans. Pattern Analysis Mach. Intell.* **12**, 489–497 (1990).
34. A. Shvedov, A. Schmidt and V. Yakubovich, Invariant systems of features in pattern recognition, *Automation Remote Control* **40**, 131–142 (1979).
35. J. Flusser and T. Suk, Pattern recognition by affine moment invariants, *Pattern Recognition* **26**, 167–174 (1993).
36. L. Fausett, *Fundamentals of Neural Networks.* Prentice-Hall, Englewood Cliffs, 1994.
37. J. Hertz, A. Krogh and R. Palmer, *Introduction to the Theory of Neural Computation.* Addison-Wesley (1991).
38. M. Minsky and S. Papert, *Perceptrons.* MIT Press, Cambridge (1969).
39. D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representations by error propagation, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, D. E. Rumelhart and J. L. McClelland, eds, Vol. 1, pp. 318–362. MIT Press, Cambridge (1986).
40. P. Simard, B. Victorri, Y. Le Cun and J. Denker, Tangent prop—a formalism for specifying selected invariances in an adaptive network, *Proceedings of NIPS*-4 895–903 (1991).
41. Y. Le Cun, Generalization and network design strategies. Connectionism in perspective. R. Pfeifer, Z. Schreter, F. Fogelmansoulie and L. Steels, eds, Chap. 40, pp. 143–155. Elsevier Science, Amsterdam (1989).
42. C. L. Giles and T. Maxwell, Learning, invariance, and generalization in high-order neural networks, *Appl. Optics* **26**, 4972–4978 (December 1987).
43. T. Kanaoka, R. Chellappa, M. Yoshitaka and S. Tomita, A higher-order neural network for distortion invariant pattern recognition, *Pattern Recognition Lett.* **13**, 837–841 (December 1992).
44. R. Duren and B. Peikari, A comparison of second-order neural networks to transform-based method for transla-

tion- and orientation-invariant object recognition. *Proceedings of the* 1991 *IEEE Workshop on Neural Networks for Signal Processing* 236–245 (1991).
45. R. Duren and B. Peikari, A new neural network architecture for rotationally invariant object recognition, *Proceedings of the 34th Midwest Symposium on Circuits and Systems*, Vol. 1, pp. 320–323. IEEE (1992).
46. H. Y. Kwon, B. C. Kim, D. S. Cho and H. Y. Hwang, Scale and rotation invariant pattern recognition using complex-log mapping and augmented second order neural network. *Electronics Lett.* **29**(7), 620–621. (April 1993).
47. L. Spirkovska and M. Reid, Robust position, scale and rotation invariant object recognition using higher-order neural networks, *Pattern Recognition* **25**, 975–985 (1992).
48. M. Takano, T. Kanaoka, J. Skrzypek and S. Tomita, A note on a higher-order neural network for distortion invariant pattern recognition, *Pattern Recognition Lett.* **15**(6), 631–635 (June 1994).
49. M. Fukumi, S. Omatu, F. Takeda and T. Kosaka, Rotation-invariant neural pattern recognition system with application to coin recognition, *IEEE Trans. Neural Networks* **3**, 272–279. (March 1992).
50. Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, Backpropagation applied to handwritten zip code recognition, *Neural Comput.* **1**, 541–551 (1989).
51. K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by a shift in position, *Biol. Cybern.* **36**, 193–202 (1980).
52. K. Fukushima and S. Miyake, Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position, *Pattern Recognition* **15**(6), 445–469 (1982).
53. D. Lovell, *The Neocognitron as a System for Handwritten Character Recognition: Limitations and Improvements*, PhD thesis, University of Queenlsland (1994).
54. E. Barnard and D. Casasent, Shift invariance and the neocognitron. *Neural Networks* **3**, 403–410 (1990).
55. K. Lang and G. Hinton, The development of TDNN architecture for speech recognition, Technical Report CMU-CS-88-152, Carnegie-Mellon University (1988).
56. A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. Lang, Phoneme recognition using time-delay neural networks, *IEEE Trans. Acoustics, Speech Signal Process.* **37**(3), 328–339 (March 1989).
57. J. Shawe-Taylor, Building symmetries into feedforward networks, *Proceedings of First IEEE Conference on Artificial Neural Networks*, pp. 158–162 (1989).
58. J. Shawe-Taylor, Symmetries and discriminability in feedforward network architectures, *IEEE Trans. Neural Networks* **4**, 816–826 (September 1993).
59. C. Yuceer and K. Oflazer, A rotation, scaling and translation invariant pattern classification system, *Pattern Recognition* **26**(5), 687–710 (May 1993).
60. R. Lenz, Group invariant pattern recognition, *Pattern Recognition* **23**(1/2), 199–217 (1990).
61. R. Lenz, *Group Theoretical Methods in Image Processing*, volume 413 of *Lecture Notes in Computer Science*, Springer-Verlag (1990).
62. J. Mas and E. Ramos, Symmetry processing in neural network models, *J. Physics A* **22**(16), 3379–3391 (August 1989).
63. M. D. Wagh and S. V. Kanetkar, A class of translation invariant transforms, *IEEE Trans. Acoustics, Speech Signal Process.* **25**, 203, (April 1977).
64. P. Moharir, *Pattern Recognition Transforms*. Research Studies Press Ltd, Taunton, Somerset, U.K. (1992).
65. A. Dobnikar, J. Ficzko, D. Podbregar and U. Rezar, Invariant pattern classification neural network versus FT approach, *Microprocessing Microprogramming*, **33**, 161–168 (1991/92).
66. J. Dunn, Continuous group averaging and pattern classification problems, *SIAM J. Comput.* **2**(4), 253–272 (December 1973).
67. T. Newman, A group theoretic approach to invariance in pattern recognition, *Proceedings of* 1979 *IEEE Computer Society Conference on Pattern Recognition and Image Processing* 407–412 (1979).
68. M. Golubitsky and D. Schaeffer, *Singularities and Groups in Bifurcation Theory*, Vol. 51 of *Applied Mathematical Sciences*. Springer-Verlag (1988).
69. J. M. Richardson, Pattern recognition and group theory, *Frontiers of Pattern Recognition*. Academic Press, New York (1972).

About the Author—JEFFREY WOOD received the BSc degree in mathematics and computer science from Royal Holloway College, University of London, in 1992. He is now working towards a PhD in Computer Science at the same institution. His research interests include neural networks and applications of group theory.