UNIVERSITE DE GENEVE

CENTRE UNIVERSITAIRE
D'INFORMATIQUE
GROUPE VISION

Date: April, 1997
N° 97.04

TECHNICAL REPORT

VISION

Invariance Signatures: Characterizing contours by their departures from invariance

David McG. Squire Terry M. Caelli

Computer Vision Group
School of Computing, Curtin University of Technology
Bentley, W.A., AUSTRALIA

Computer Vision Group
Computing Science Center, University of Geneva
24 rue du Général Dufour, CH - 1211 Geneva 4 SWITZERLAND
e-mail: squire@cui.unige.ch tmc@cs.curtin.edu.au

Abstract

In this paper, a new invariant feature of two-dimensional contours is reported: the Invariance Signature. The Invariance Signature is a measure of the degree to which a contour is invariant under a variety of transformations, derived from the theory of Lie transformation groups. Since it is derived from local properties of the contour, it is well-suited to a neural network implementation. It is shown that a Model-Based Neural Network (MBNN) [7, 36] can be constructed which computes the Invariance Signature of a contour, and classifies patterns on this basis. Experiments demonstrate that Invariance Signature networks can be employed successfully for shift-, rotation- and scale-invariant optical character recognition.

1 Introduction

The ability to perceive the permanent features of the visual environment is something which humans take for granted. It is hard to imagine a world in which we could not do so. In this paper we are concerned in particular with the invariant perception of two-dimensional patterns under shift, rotation and scaling in the plane. This corresponds to the ability of humans to recognize patterns such as typed or handwritten characters independently of their size, orientation or position, which they do unthinkingly when reading a document such as an engineering or architectural drawing.

The aim of any invariant pattern recognition technique is to obtain a representation of the pattern in a form that is invariant under some transformations of the original image. It is often claimed that an ideal technique would produce a representation of the pattern which was not only invariant, but which also uniquely characterized the input pattern. This goal is not always as desirable as it might seem.

In many pattern recognition problems, the aim is to produce a system which classifies input patterns as belonging to a particular *class*, rather than to identify uniquely every input pattern presented. In such cases a unique representation for each possible input pattern can actually be a disadvantage. All that is required is an invariant representation which retains enough information for distinct classes to be distinguished. Indeed, if all members of a class are identical, or nearly so, in the invariant representation this can greatly reduce the size of the training set required by many recognition algorithms. The Invariance Signature provides a means of realizing this goal.

1.1 Prior Methods for Invariant Pattern Recognition

A large number of techniques for invariant pattern recognition has been proposed. They vary greatly. Some treat the image as a pattern of pixels, others operate on higher level representations, such as contours. Many are expensive in computation time and/or space. They have varying degrees of sensitivity to noise. A brief review of popular techniques follows.

An approach that has long been popular is to seek an integral transform of the image which is invariant under some specified transformations. Most such techniques are based upon the Fourier transform: in the transform domain the amplitude spectrum is invariant under shifts of the image and the phase encodes the shift. Other invariances can be obtained by first performing a coordinate transform so that the transformation with respect to which invariance is desired becomes a shift in the new coordinate system [13, 29, 33]. Such transforms, in various guises, form the basis of many invariant pattern recognition techniques [1, 6, 12, 23]. It is important to note that the transformed image is of (at least) the same dimensionality as the original. The invariant component does not uniquely characterize the input, and the issue of actually recognizing it is not addressed at all.

Integrals can also be used to compute geometrical moments of the image. Certain functions of the moments are invariant under transformations of the image [18]. Geometrical moments are not fault-tolerant, and generally produce disappointing pattern recognition results [39]. Better results have been obtained using alternative sets, such as the Zernike moments [19]. Again, computing a set of moments is expensive.

Matched Filtering is one of the oldest techniques for pattern recognition [2, 28]. Combined with convolution it becomes cross-correlation: effectively an exhaustive search of the image for all possible transformed versions of a template pattern. Some version of cross-correlation plays a role in many invariant pattern recognition techniques [1, 6, 20, 23, 26, 40], despite the fact that it is

computationally-expensive, and this cost scales exponentially with the number of transformations with respect to which invariance is desired.

In the above techniques, an invariant representation of the whole image is sought. Matching is done on a pixel-by-pixel basis. A different approach is to view an image as a set of parts and relationships. For example, the angle between two lines is invariant under shifts, rotations and scalings. The lines are parts, and the angle is their relationship. In "parts and relationships" techniques, matching is done between abstracted properties of the image, rather than between pixels. These techniques require some form of sub-graph matching [21], which is known to be an NP-complete problem [2]. The challenge is thus to find an algorithm that can obtain an acceptable solution in reasonable time. Explicit search may be satisfactory for sufficiently small graphs. Another approach is relaxation labeling [21].

In this paper we are specifically interested in the invariant recognition of contours. There are two approaches to contour recognition: the contour may be represented by an algebraic expression fitted to the image, or treated as a group of pixels. Contour recognition is of particular interest because there are many applications in which patterns naturally consist of line drawings (*e.g.* character recognition, circuit diagrams, engineering drawings). Moreover, there is evidence that the human visual system applies a contour-based approach to pattern recognition even when no contour exists in the image: an implied contour is interpolated [8].

Perhaps the simplest contour-based technique is the Hough transform [2]. Variants of the Hough transform occur frequently in the invariant pattern recognition literature [10, 22, 26]. The Hough transform and its generalizations can be interpreted as cross-correlation techniques, where the template is a parametric curve rather than an image [26].

Another approach is to compute invariants of the contour. This promises to avoid the expenses of exhaustive methods such as the Hough transform. Algebraic invariants are well suited for use with contours which can be expressed by an implicit polynomial $f(x, y) = 0$. An example is the family of conic sections, which can be defined in matrix form: $\mathbf{X}^{\mathrm{T}}\mathbf{A}\mathbf{X} = 0$. The coordinates can be transformed so that $\mathbf{A}$ is diagonal. Thus properties of $\mathbf{A}$ invariant under similarity transforms are invariant descriptors of the conic. One such feature is the determinant. In fact, any symmetric function of the eigenvalues of $\mathbf{A}$ is an invariant. $\mathbf{A}$ must be obtained by fitting a polynomial to the image – a difficult and potentially computationally-expensive problem. A high resolution image of the curve is required for accurate coefficient estimation. The matching process, however, is cheap.

Differential Invariants arise when the points on a curve, $\mathbf{x}$, are expressed as a function of a parameter t, $\mathbf{x} = \mathbf{x}(t)$, rather than by an implicit function. The natural shape descriptors in such a representation are the derivatives $\frac{d^n x_i}{dt^n}$. These descriptors are *local*, since they are evaluated at t, unlike the global descriptors derived using algebraic invariants. A differential invariant is a function of the $\frac{d^n x_i}{dt^n}$ which does not change under transformations of $\mathbf{x}$ and t. Various differential invariants have been applied: curvature, torsion and Gaussian curvature, for instance, are all invariant under Euclidean transformations [14]. Differential invariants are complete: a small set of invariants contains all the essential information about the curve. Also, their locality makes them insensitive to occlusion. Whilst elegant, its application requires the computation of high derivatives of the contour, which is known to be error-prone. Moreover, these derivatives are raised to high powers, magnifying the estimation error.

2 Lie Transformation Groups and Invariance

One approach to invariant pattern recognition is to consider how local features change under global transformations. This leads naturally to the study of Lie transformation groups, which have been a component of many, varied invariant pattern recognition techniques [11, 12, 15, 16, 29, 33, 37].

We derive a new shift-, rotation- and scale- invariant function of a two-dimensional contour, the *Invariance Measure Density Function*. It is shown that several such functions can be combined to yield an *Invariance Signature* for the contour. This Invariance Signature has several properties that make it attractive for implementation in an MBNN: it is based on local properties of the contour, so initial calculations are inherently parallel; it is statistical in nature, and its resolution can be chosen at the designer's discretion, allowing direct control over the dimensionality of the network implementation. The use of the Invariance Signature, however, is not limited to neural

implementations.

Whilst patterns are not represented uniquely by the Invariance Signature, it will be shown in Section 5 that *classes* of patterns are represented sufficiently differently for optical character recognition applications.

2.1 One Parameter Lie Groups in Two Dimensions

A Lie group is a continuous transformation group with a differentiable structure [29]. For our purposes, the most interesting groups are the one-parameter Lie groups defined on the plane. These include rotation, dilation and translation. They are smooth transformations of the form

$$x' = \mu(x, y, \varepsilon) \qquad y' = \nu(x, y, \varepsilon). \tag{1}$$

The parameter ε determines which element of the group the transformation is. For instance, if ε_0 corresponds to the identity element, we have

$$x' = \mu(x, y, \varepsilon_0) = x \qquad y' = \nu(x, y, \varepsilon_0) = y. \tag{2}$$

There is a vector field $\vec{g} = \begin{bmatrix} g_x & g_y \end{bmatrix}^{\mathrm{T}}$ associated with each Lie group G, which gives the direction in which a point (x, y) is "dragged" by an infinitesimal transformation under the group. It is given by

$$g_x(x, y) = \left.\frac{\partial \mu}{\partial \varepsilon}\right|_{\varepsilon=\varepsilon_0} \qquad g_y(x, y) = \left.\frac{\partial \nu}{\partial \varepsilon}\right|_{\varepsilon=\varepsilon_0}. \tag{3}$$

This vector field $\vec{g}$ allows an operator $\mathscr{L}_G$ to be defined,

$$\mathscr{L}_G = g_x \frac{\partial}{\partial x} + g_y \frac{\partial}{\partial y}. \tag{4}$$

$\mathscr{L}_G$is called the *generator* of G, because it can be used to construct the finite transformation corresponding to the infinitesimal dragging described in Eqs. (3).

2.2 From Infinitesimal to Finite Transformations

Consider the case in which the vector field specifying the infinitesimal transformation at each point (Eqs. (3)) is known. We wish to construct Eqs. (1), specifying the finite transformation. We will consider the transformation of x in detail. For a small change in the group parameter from the identity element, $\varepsilon = \varepsilon_0 + \Delta\varepsilon$, we can approximate the change in x by

$$x' = x + \Delta x \approx x + \Delta\varepsilon \left.\frac{\partial \mu}{\partial \varepsilon}\right|_{\varepsilon=\varepsilon_0}. \tag{5}$$

We now wish to find a finite transformation corresponding to n applications of the $\Delta\varepsilon$ transformation. This will approximate the finite transformation corresponding to the group element specified by parameter $\varepsilon = n\Delta\varepsilon$. Let x_i be the value of x' after i applications of the $\Delta\varepsilon$ transformation. We obtain

$$\begin{aligned} x_0 &= x \\ x_1 &= x_0 + \frac{\varepsilon}{n}\mathscr{L}_G x_0 = \left(1 + \frac{\varepsilon}{n}\mathscr{L}_G\right) x \\ x_2 &= x_1 + \frac{\varepsilon}{n}\mathscr{L}_G x_1 \\ &= \left(1 + \frac{\varepsilon}{n}\mathscr{L}_G\right) x_1 \\ &= \left(1 + \frac{\varepsilon}{n}\mathscr{L}_G\right)^2 x \end{aligned} \tag{6}$$

and thus

$$x_n = \left(1 + \frac{\varepsilon}{n}\mathscr{L}_G\right)^n x.$$

In the limit as $n \to \infty$, the approximation becomes exact, giving the finite transformations

$$\mu(x, y, \varepsilon) = \lim_{n\to\infty} \left(1 + \frac{\varepsilon}{n}\mathscr{L}_G\right)^n x \qquad \nu(x, y, \varepsilon) = \lim_{n\to\infty} \left(1 + \frac{\varepsilon}{n}\mathscr{L}_G\right)^n y. \tag{7}$$

An example of the direct application of the derivation of the familiar rotation transformation using Eqs. (7) may be found in [36].

2.3 Functions Invariant Under Lie Transformations

A function is said to be invariant under a transformation if all points of the function are mapped into other points of the function by the transformation. Consider a function $F(x, y)$. We wish to determine its invariance with respect to a Lie transformation group G. Let

$$\vec{g}(x, y) = \left[\begin{array}{cc} g_x(x, y) & g_y(x, y) \end{array}\right]^{\mathrm{T}} \tag{8}$$

be the associated vector field, and $\mathscr{L}_G$ be the generator of G. F is constant with respect to the action of the generator if

$$\mathscr{L}_G F = 0. \tag{9}$$

This can be written in terms of the vector field as

$$\nabla F \cdot \vec{g}(x, y) = 0. \tag{10}$$

Now consider a contour C parameterized by t specified by the implicit function

$$\forall t \;\; F(x(t), y(t)) = K. \tag{11}$$

Since F is constant on the contour, we have

$$\frac{dF}{dt} = \frac{\partial F}{\partial x}\frac{dx}{dt} + \frac{\partial F}{\partial y}\frac{dy}{dt} = 0. \tag{12}$$

Combining Eqs. (10) and (12) shows that F is invariant under the Lie transformation generated by $\mathscr{L}_G$ if

$$\frac{dy}{dx} = \frac{g_y}{g_x}. \tag{13}$$

everywhere on the contour.

The condition derived in Eq. (13) has a very natural interpretation: a contour is invariant under a transformation group G if the tangent to the contour at each point is in the same direction as the vector field $\vec{g}$ corresponding to the infinitesimal transformation that generates the group.

3 The Invariance Signature: From Local Invariance Measures to Global Invariance

We now propose a new shift-, rotation- and dilation-invariant signature for contours. We call this an *Invariance Signature*, since it is derived from the degree to which a given contour is consistent with invariance under a set of Lie transformation groups.

3.1 The Local Measure of Consistency

We have seen in Eq. (13) that in order for a contour C to be invariant under a transformation group G the tangent to the contour must be everywhere parallel to the vector field defined by the generator of the group. We now define the *Local Measure of Consistency* with invariance under a transformation group G at a point (x, y) on C, $\iota_G(x, y)$.

$$\iota_G(x, y) = \left|\hat{\theta}(x, y) \cdot \hat{g}_G(x, y)\right| \tag{14}$$

The absolute value is used because only the orientation of the tangent vector is significant, not the direction. At each point both the tangent vector to the contour, $\vec{\theta}(x,y)$ and the vector field $\vec{g}(x,y)$ are normalized:

$$\hat{g}(x,y) = \frac{g_x(x,y)\hat{\imath} + g_y(x,y)\hat{\jmath}}{\sqrt[+]{g_x^2(x,y) + g_y^2(x,y)}} \tag{15}$$

and

$$\hat{\theta}(x,y) = \frac{\hat{\imath} + \frac{dy}{dx}\hat{\jmath}}{\sqrt[+]{1 + \left(\frac{dy}{dx}\right)^2}}, \tag{16}$$

where $\hat{\imath}$ and $\hat{\jmath}$ are unit vectors in the x and y directions respectively. Substituting Eqs. (15) and (16) in Eq. (14), we obtain

$$\iota_G(x,y) = \frac{1}{\sqrt[+]{1 + \left[\frac{g_y(x,y) - g_x(x,y)\frac{dy}{dx}}{g_x(x,y) + g_y(x,y)\frac{dy}{dx}}\right]^2}}. \tag{17}$$

3.2 The Invariance Measure Density Function

Eq. (14) is a mapping $C \mapsto [0,1]$, which gives a measure of the degree to which the tangent at each point is consistent with invariance under G. We now seek a function which characterizes the consistency of the entire contour C with invariance under G. Such a function is the density function for the value of ι_G in $[0,1]$, $I(\iota_G)$, which we will call the *Invariance Measure Density Function*. The more points from C that are mapped close to 1 by Eq. (14), the more consistent C is with invariance under G. $I(\iota_G)$ is a descriptor of C, and we will show that $I(\iota_G)$ is invariant under rotations and dilations of C. Translation invariance is obtained by choosing the centroid of C as the origin of coordinates.

It is interesting to note that there is evidence from psychophysical experiments that a measure of the *degree of invariance* of a pattern with respect to the similarity group of transformations (rotations, translations and dilations) is important in human pattern recognition [5]. The measure proposed here might be seen as a mathematical formalization of this notion. Moreover, its implementation in a neural network architecture is consistent with Caelli and Dodwell's statement [5, p. 159] of a proposal due to Hoffman [15, 16]:

> Hoffman's fundamental postulate was that the coding of orientation at various positions of the retinotopic map by the visual system, discovered by Hubel and Wiesel [17] and others, actually provides the visual system with "vector field" information. That is, the visual system, on detecting specific orientation and position states ("Φ/P codes"), spontaneously extracts the path curves (interpreted as visual contours) of which the local vectors are tangential elements.

First, however, we must establish the form of $I(\iota_G)$. Let C be parameterized by $t : t \in [t_0, T]$. The arc length s along C is

$$s(t) = \int_{t_0}^{t} \sqrt{\left(\left.\frac{dx}{dt}\right|_\tau\right)^2 + \left(\left.\frac{dy}{dt}\right|_\tau\right)^2}\, d\tau. \tag{18}$$

The total length of C is thus $S = s(T) = \oint_C ds$. For well-behaved functions $F(x,y)$, we can construct $s(t)$ such that we can reparameterize C in terms of s. Thus we can rewrite Eq. (14) to give ι_G in terms of s. For simplicity, we will first consider the case in which ι_G is a monotonic function of s, as shown in Figure 1. The Invariance Measure Density is:

$$I(\iota_G) = \lim_{\Delta\iota_G \to 0} \left|\frac{\Delta s}{S\Delta\iota_G}\right| = \frac{1}{S}\left|\frac{ds}{d\iota_G}\right|. \tag{19}$$

$I(\iota_G)$ can be interpreted as the probability density function for ι_G at points $(x(s), y(s))$, where s is a random variable uniformly distributed on $[0, S]$. It is clear that for the general case the

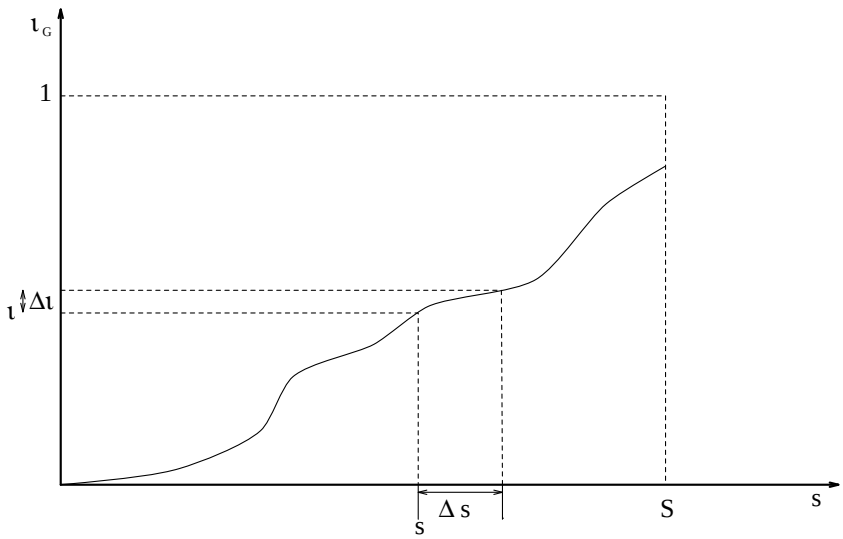


Figure 1: Local Measure of Consistency as a function of arc length.

function could be broken into piecewise monotonic intervals and their contributions to the density summed. The general form for a specific value $\iota_G{}'$ is thus

$$I\left(\iota_G{}'\right) = \frac{1}{S} \sum_{s\in[0,S]:\iota_G(s)=\iota_G{}'} \left|\frac{ds}{d\iota_G}\right|. \tag{20}$$

Theorem 1 *The Invariance Measure Density, $I\left(\iota_G\right)$, is invariant under translations, rotations and dilations of the contour C with respect to which it is calculated.*

Proof That $I\left(\iota_G\right)$, defined in Eq. (20), is invariant under translations of the contour C is trivial, since, as defined in Section 3.2, ι_G is calculated with the origin at the centroid of the contour C. Rotation and dilation invariance can be proved by construction. Since the transformations for rotation and dilation in the plane commute, we can consider a two-parameter Abelian group corresponding to a rotation by an angle ϕ and a dilation by a positive factor β. The coordinates x and y are transformed according to

$$x' = \beta\left(x\cos\phi - y\sin\phi\right) \qquad\qquad y' = \beta\left(x\sin\phi + y\cos\phi\right). \tag{21}$$

Consider the relationship between the arc length $s(t)$ for the original parameterized contour $(x(t), y(t))$ and $s'(t)$, after the coordinates are transformed according to Eqs. (21). We find that

$$\frac{dx'}{dt} = \beta\cos\phi\frac{dx}{dt} - \beta\sin\phi\frac{dy}{dt} \qquad\qquad \frac{dy'}{dt} = \beta\sin\phi\frac{dx}{dt} + \beta\cos\phi\frac{dy}{dt}. \tag{22}$$

Combining these, we obtain

$$\left(\frac{dx'}{dt}\right)^2 + \left(\frac{dy'}{dt}\right)^2 = \beta^2\left[\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right]. \tag{23}$$

This result can be substituted into Eq. (18), giving

$$s'(t) = \beta s(t). \tag{24}$$

This indicates that the total arc length is $S' = \beta S$. The derivative of $s(t)$ is also scaled. Substituting

into Eq. (20), we obtain

$$\begin{aligned} I'(\iota_G{}') &= \frac{1}{S'} \sum_{\iota_G(s)=\iota_G{}'} \left| \frac{ds'}{d\iota_G} \right| \\ &= \frac{1}{\beta S} \sum_{\iota_G(s)=\iota_G{}'} \beta \left| \frac{ds}{d\iota_G} \right| \\ &= \frac{1}{S} \sum_{\iota_G(s)=\iota_G{}'} \left| \frac{ds}{d\iota_G} \right| \\ &= I(\iota_G{}') \qquad \square \end{aligned} \tag{25}$$

Thus $I(\iota_G{}')$ is invariant under rotations and dilations of C.

3.3 Invariance Measure Densities For Specific Contours

To demonstrate the application of the Invariance Measure Density, we will evaluate $I(\iota_G)$ for a specific contour. Let C be a square of side $2L$ centred at the origin, as shown in Figure 2. We

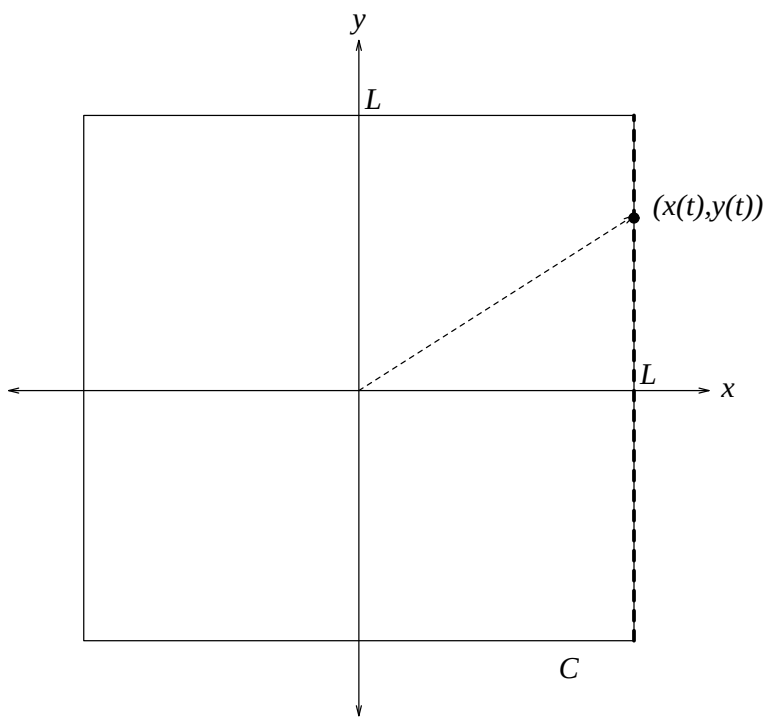


Figure 2: Square of side 2L.

will find the Invariance Measure Density for C with respect to rotation, $I_{C_{rot}}(\iota)$. By symmetry, we need only find $I_{C_{rot}}$ for one side of the square. On the side indicated by the dashed line in Figure 2, x and y can be expressed in terms of a parameter t as

$$x = L \qquad\qquad y = t, \qquad -L \leq t \leq L. \tag{26}$$

Thus the arc length is $s(t) = t + L$, and the total is $S = 2L$. If $\dot{x}(t)$ and $\dot{y}(t)$ are the derivatives of x and y with respect to t, Eq. (17) can be rewritten as

$$\iota_{C_{rot}}(s) = \frac{1}{\sqrt{1 + \left[\frac{g_y(s)\dot{x}(s) - g_x(s)\dot{y}(s)}{g_x(s)\dot{x}(s) + g_y(s)\dot{y}(s)}\right]^2}} \tag{27}$$

Here $\dot{x}(t) = 0$ and $\dot{y}(t) = 1$. The generator of the rotation group is

$$\mathscr{L}_R = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}. \tag{28}$$

Eqs. (28) and (26) can be substituted into Eq. (27) to give

$$\iota_{C_{rot}}(s) = \frac{1}{\sqrt{1 + \left(\frac{s-L}{L}\right)^2}}, \tag{29}$$

which can be inverted to yield

$$s = L\left(1 + \sqrt{\frac{1}{\iota_{C_{rot}}^2} - 1}\right); \tag{30}$$

differentiating,

$$\frac{ds}{d\iota_{C_{rot}}} = \frac{-L}{\iota_{C_{rot}}^2\left(1 - \iota_{C_{rot}}^2\right)}. \tag{31}$$

Using Eq. (20), we arrive at our final result:

$$\begin{aligned} I_{C_{rot}}\left(\iota_{C_{rot}}\right) &= \frac{1}{2L}\sum_{\iota'_{C_{rot}}=\iota_{C_{rot}}}\left|\frac{ds}{d\iota_{C_{rot}}}\right|_{\iota'_{C_{rot}}} \\ &= \frac{1}{2L} \times 2 \times \frac{L}{\iota_{C_{rot}}^2\left(1 - \iota_{C_{rot}}^2\right)} \\ &= \frac{1}{\iota_{C_{rot}}^2\left(1 - \iota_{C_{rot}}^2\right)}, \qquad \iota_{C_{rot}} \in \left[\frac{1}{\sqrt{2}}, 1\right]. \end{aligned} \tag{32}$$

Note that, as required, the scale of the square L does not appear. The factor of 2 arises because $\iota_{C_{rot}}$ is a symmetric function of s, so the sum has two terms. This function, shown in Figure 3, is characteristic of the square.

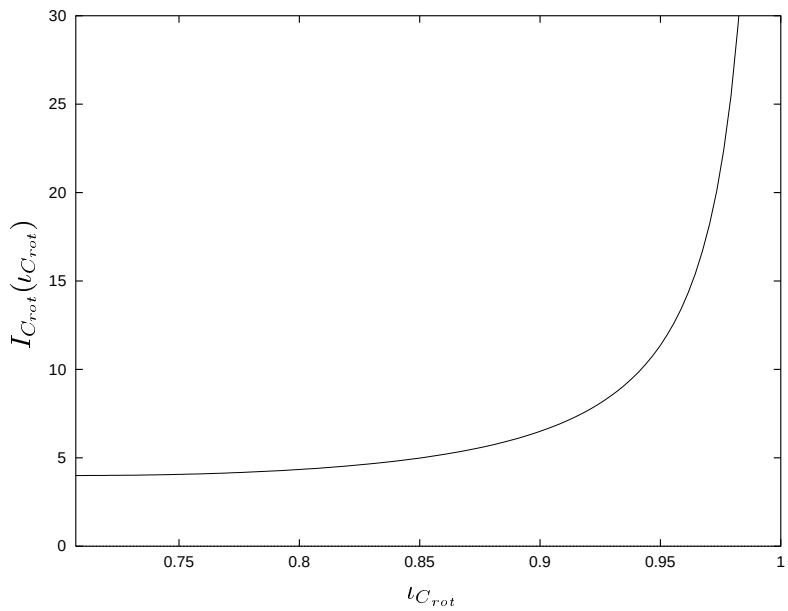


Figure 3: Invariance Density Measure with respect to rotation for a square.

3.4 Invariance Space: Combining Invariance Measure Densities

We now consider the case in which the Invariance Measure Density Function is calculated with respect to a number of groups, and the results combined to provide a more complete characterization of the transformational properties of a contour C. This operation maps each point from the two dimensional image space to the interior of a unit hypercube in an n-dimensional *invariance space*, where each of the n dimensions corresponds to a particular sort of invariance. Eq. (33) shows this for the case of a three-dimensional invariance space where the dimensions correspond to the Local Measure of Consistency ι with respect to rotation, dilation and translation:

$$(x, y) \mapsto \begin{bmatrix} \iota_{rot} & \iota_{dil} & \iota_{trans} \end{bmatrix}^{\mathrm{T}}. \tag{33}$$

The distribution of points in this invariance space is characteristic of the contour C. This particular three-dimensional invariance space will be used for the experimental application of Invariance Signatures. Since each of the component invariance measure densities is invariant, this n-dimensional Invariance Signature is invariant under rotations, dilations, translations (and reflections) of the input image.

The vector fields for the generators of the transformation groups for rotation, dilation and translation are given in normalized form. All can be derived using Eq. (3): for rotation invariance

$$\vec{g}_{rot}(x, y) = \frac{1}{\sqrt{x^2 + y^2}} \begin{bmatrix} -y & x \end{bmatrix}^{\mathrm{T}}, \tag{34}$$

and for dilation invariance

$$\vec{g}_{dil}(x, y) = \frac{1}{\sqrt{x^2 + y^2}} \begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}. \tag{35}$$

The translation invariance case is somewhat different. What is measured is the degree to which the contour is "linear". The vector field used is constant for all (x, y), and the direction is given by the unit eigenvector corresponding to the largest eigenvalue, $\vec{e}_1$, of the coordinate covariance matrix of all points in the contour. The direction of this eigenvector is the principal direction of the contour. Since $\vec{e}_1$ is calculated from the image each time it is required, this measure is invariant under rotations, dilations and translations of the image. The vector field for the translation invariance case is thus:

$$\vec{g}_{trans}(x, y) = \begin{bmatrix} e_{1x} & e_{1y} \end{bmatrix}^{\mathrm{T}} \tag{36}$$

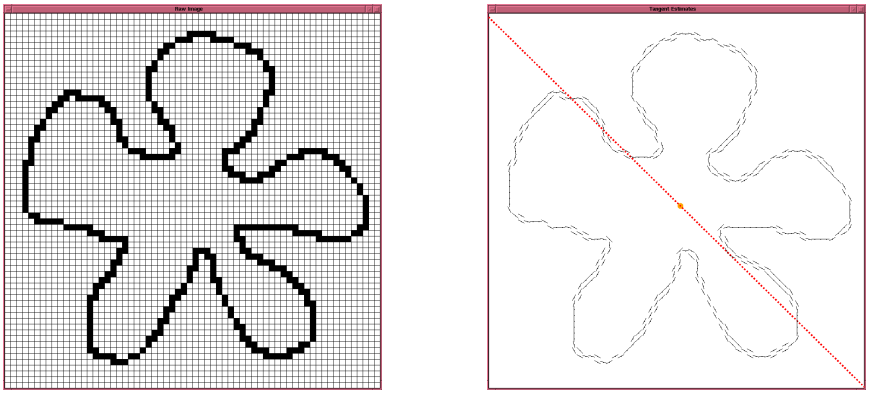
It should be noted that this representation of the image is not unique. The combination of individual Invariance Measure Densities into an Invariance Space does, however, increase its discriminating properties. As an example, removing two opposite sides of a square will not alter its rotation or dilation Invariance Signatures, but it will change the translation Invariance Signature. Likewise, a single straight line has the same translation Invariance Signature as any number of parallel straight lines, however they are spaced. The rotation and dilation Invariance Signatures, however, are sensitive to these changes.

3.5 Discrete Invariance Signatures

For a computer application of Invariance Signatures, a discrete version is required. The natural choice is the frequency histogram of ι_G. For a continuous contour, this is obtained by dividing the interval $[0, 1]$ into n "bins" and integrating $I(\iota_G)$ over each bin. For bins numbered from b_0 to b_{n-1}, the value in bin k is

$$b_k = \int_{\frac{k}{n}}^{\frac{k+1}{n}} I(\iota_G) d\iota_G. \tag{37}$$

Since $I(\iota_G)$ is a probability density function, the sum of the values of the bins must be one.

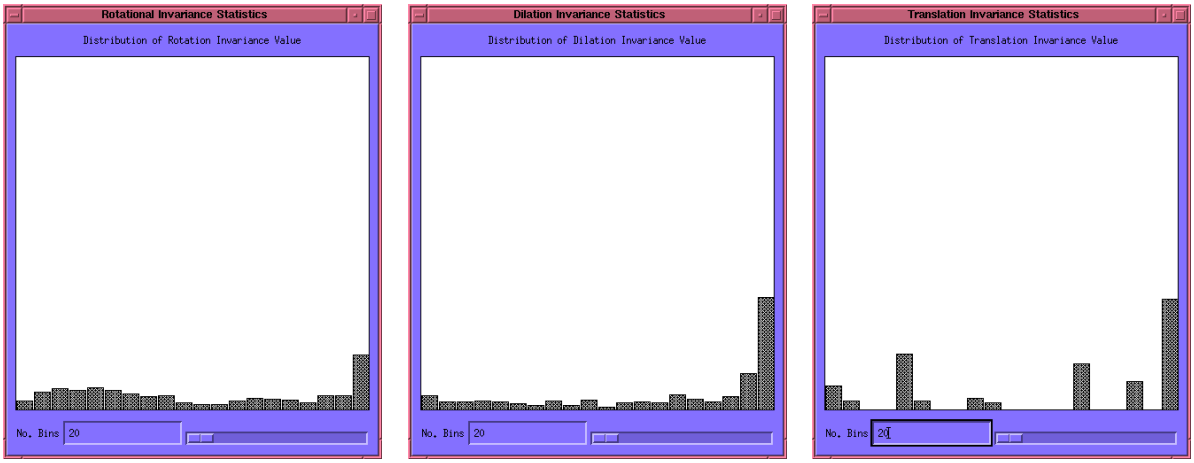


1. Original Contour 2. Tangent Estimates

Figure 4: Example of a sampled contour and its estimated tangents.

When using sampled images, a true frequency histogram of the estimated local measures of consistency may be used. The system designer must choose the number of bins, n, into which the data is grouped. It will be seen in Section 5 that this choice is not arbitrary.

An example of a sampled contour and the estimated tangent vectors at each point is shown in Figure 4. The circle indicates the centroid of the contour, and the dashed line shows the direction of $\vec{e}_1$. The estimated discrete Invariance Signatures are shown in Figure 5, for 20 bins. It would



1. Rotation 2. Dilation 3. Translation

Figure 5: 20 bin discrete Invariance Signatures for the contour in Figure 4.

be expected that this "flower"-shaped contour would have Invariance Signatures which reflect a quite strong dilation-invariant component corresponding to the approximately radial edges of the "petals", and also a significant rotation-invariant component due to the ends of the petals which are approximately tangential to the radial edges. This is indeed what is observed in Figure 5.

4 The Invariance Signature Neural Network Classifier

We propose a Model-Based Neural Network to compute the discrete Invariance Signature of an input pattern and to classify it on that basis. It consists of a system of neural network modules, some hand-coded and some trained on sub-tasks. A schematic diagram is shown in Figure 6. This system will be referred to as the Invariance Signature Neural Network Classifier (ISNNC).

Whilst the ISNNC appears complex, it retains the basic characteristics of a traditional feed-

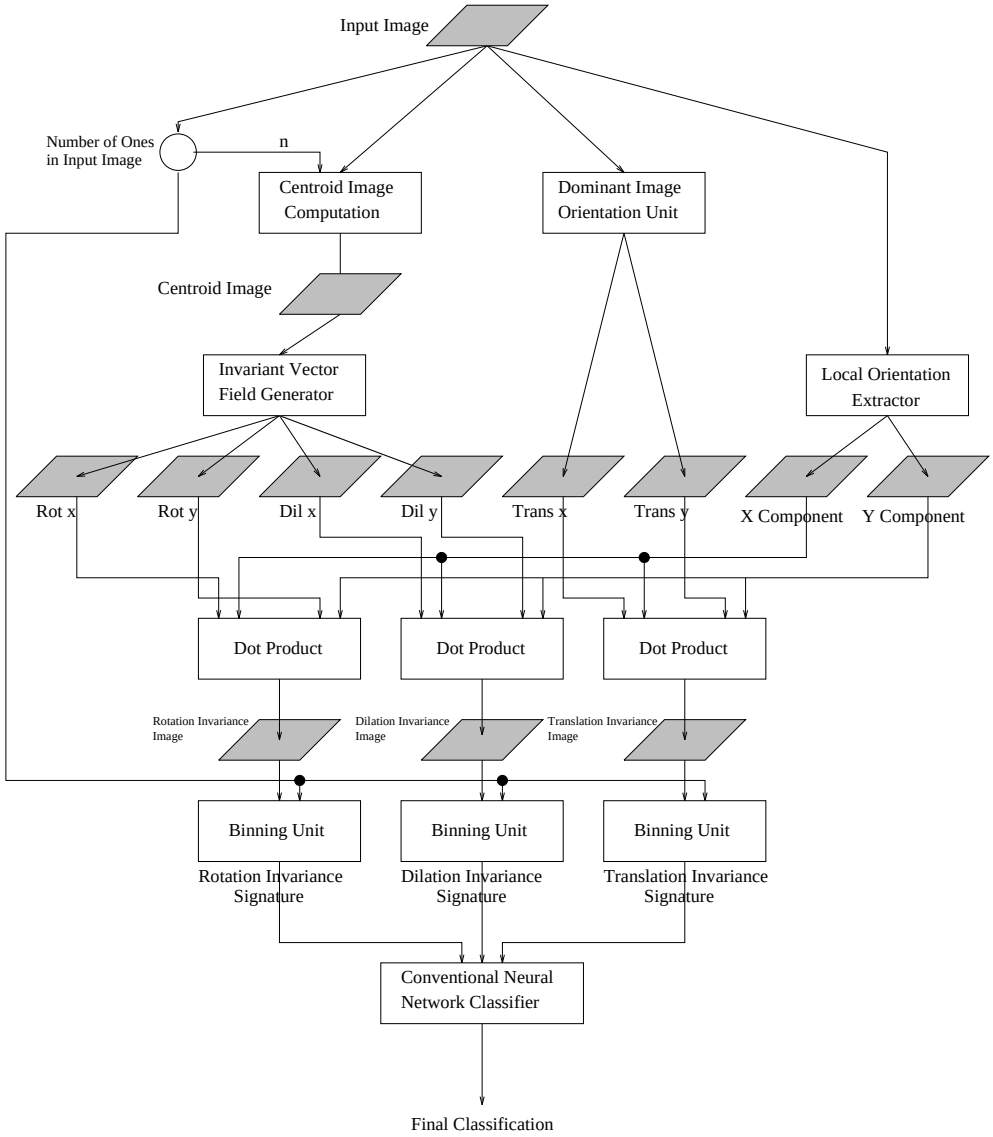


Figure 6: Invariance Signature-based contour recognition system.

forward neural network, or TNN (TNN will be used to describe a fully-connected, feed-forward multilayer perceptron). It consists entirely of simple nodes joined by weighted connections. (with the exception of the *Dominant Image Orientation Unit*, for which a neural network solution is still to be developed.) Each node i in the network computes the sum of its j weighted inputs, $\text{net}_i = \sum_j w_{ij} x_j$. This is used as the input to a transfer function f, which is either linear, $f(\text{net}_i) = \text{net}_i$, or the standard sigmoid, $f(\text{net}_i) = \frac{1}{1+e^{-\text{net}_i}}$. There is a number of distinct levels. Computation at one level must be completed before computation at the next level begins.

The only departure from a traditional neural network is that some weights are calculated at runtime by nodes at prior levels. We call these *dynamic weights*. They allow the ISNNC to compute dot products, and for nodes to act as gates controlling the transmission of the output of another node. Since connection weights in any implementation are only references to a stored value, this should not present any difficulty. Alternatively, the same functionality can be achieved by allowing nodes which multiply their inputs, as used in Higher Order Neural Networks [25, 35].

4.1 Lie Vector Field Generation

4.1.1 Calculation of a Centroid Image

The first step is to compute the centroid of the image, since this must be the origin of coordinates. This is done using a neural module which takes as its input a binary image, and outputs an image which is zero everywhere except at the centroid, where it is one. The weights of the Centroid Image Computation module are entirely hand-coded.

A quantity needed for this operation, and also later in the ISNNC, is N_{on}, the total number of "on" pixels in the input binary image $\mathcal{I}(x_i, y_j)$. This is given by

$$N_{\text{on}} = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \mathcal{I}(x_i, y_j). \tag{38}$$

This can easily be computed by a node with a linear transfer function which has inputs from all input nodes with weights of 1. The value used in later levels of the ISNNC is $\frac{1}{N_{\text{on}}}$. Since the input image size is fixed, it is trivial to build a neural module which interpolates $f(x) = \frac{1}{x}$ with any desired accuracy for the integer values $\{x : x \in [0, N_x N_y]\}$ [4]. It will be assumed that $\frac{1}{N_{\text{on}}}$ is available.

The coordinates, $(\bar{x}, \bar{y})$, of the centroid of $\mathcal{I}(x_i, y_j)$ are

$$\bar{x} = \frac{1}{N_{\text{on}}} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \mathcal{I}(x_i, y_j)\, x_i \qquad \bar{y} = \frac{1}{N_{\text{on}}} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \mathcal{I}(x_i, y_j)\, y_i \tag{39}$$

A system of two neurons can compute these quantities (see Figure 7). This illustrates the use of dynamic weights. The weight on the connection between Nodes A and B is dynamic, as it is computed from the number of ones in the input image. The weights on the connections to Node A are the x-coordinates of the input nodes.These are fixed during the design process: there is no sense in which a node needs to "know" its coordinates. An identical system computes $\bar{y}$.

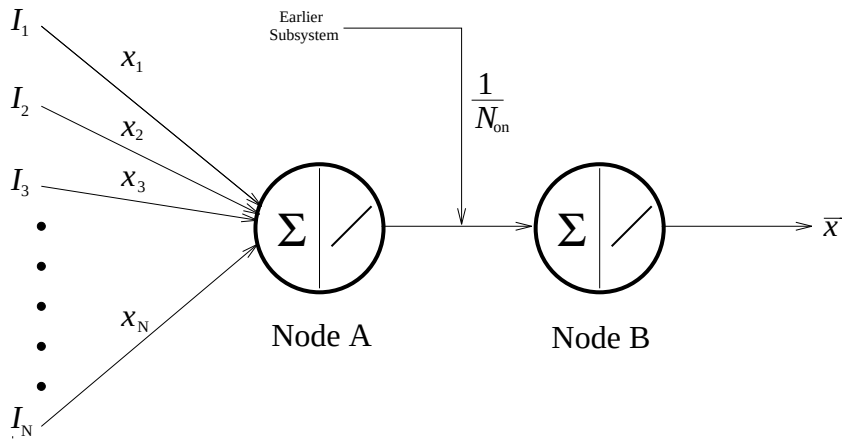


Figure 7: $\bar{x}$ module.

Once $\bar{x}$ and $\bar{y}$ have been calculated, the Centroid Image is generated using two nodes for each input node, as shown in Figure 8. The parameter α in Figure 8 (and similar figures) determines the gain of the sigmoids. In this study it was set to 100. The weights and inputs to Nodes A and B are set so that they "fire" only if their coordinate is within θ of the centroid. $\theta = 0.5$ was used, since input nodes were assigned unit spacing. A neural AND of these nodes is calculated by Node C.

This results in two images, one for the x coordinate and one for y. Each has a straight line of activation corresponding to the centroid value for that coordinate. A neural AND of these images is gives the final Centroid Image, $\mathcal{I}_C$, in which only the centroid node is one.

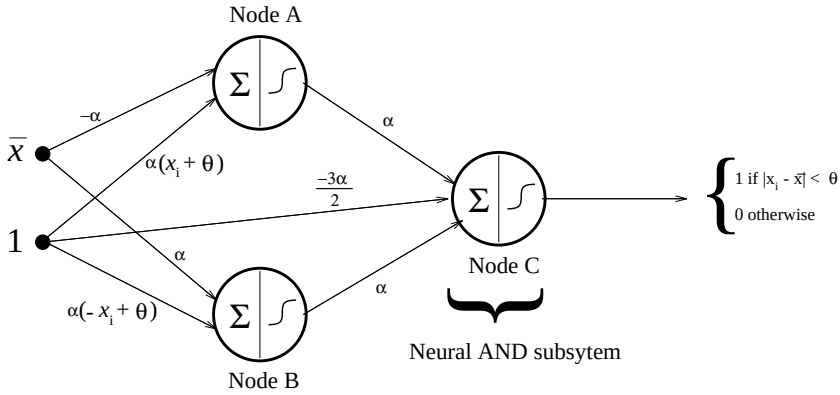


Figure 8: Coordinate matching module for node x_i. Output is 1 when $|x_i - \bar{x}| < \theta$.

This admittedly complicated neural procedure for computing the centroid is of course unnecessary if the system is being simulated on a digital computer. A hybrid system in which some calculations are not neural is a more efficient solution in that case. The goal in this section, however, is to show that parallel, neural solutions to these problems are possible.

4.1.2 Gating of Vector Fields

The Centroid Image is used to produce the rotation and dilation vector fields: four images in all. The weights from the Centroid Image are derived from Eqs. (34) and (35). Each centroid image node has weights going to all the (linear) nodes in each vector field component layer. Each weight has the value appropriate for the vector field at the destination node if the source node is the centroid. Since all nodes of the Centroid Image are zero except for the centroid node, only weights from the centroid node contribute to the vector field component images. Thus weights corresponding to the vector fields for all possible centroid positions are present in the network, but only the appropriate ones contribute. Eq. (40) shows the function computed by a node (k, l) in the rotation invariance vector field x-component image:

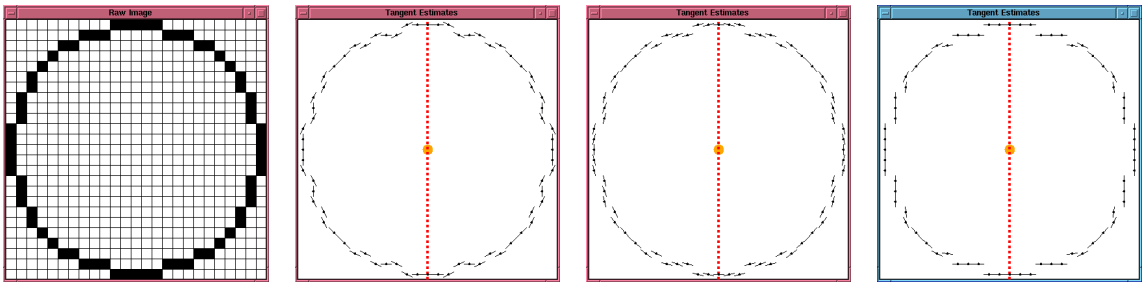
$$\mathrm{Rot}_{x_{(k,l)}} = \sum_{i=0}^{x_{max}} \sum_{j=0}^{y_{max}} w_{(i,j)(k,l)} \mathcal{I}_{C_{(i,j)}} \qquad \text{where} \qquad w_{(i,j)(k,l)} = \frac{-(j-l)}{\sqrt{(j-l)^2 + (i-k)^2}}. \tag{40}$$

It does not matter which node is chosen as the origin of coordinates when the weights are set, only that it is consistent between layers. Similar weighting functions are used for the other vector component images.

4.2 Local Orientation Extraction

4.2.1 A Tangent Estimate

To compute the Invariance Signature of a contour, it is necessary to estimate the tangent vector at each point. A simple and robust estimate of the tangent vector at a point is the eigenvector corresponding to the largest eigenvalue of the covariance matrix of a square window centred on that point. The size of the window defines the number of orientations possible. Figure 9 shows the tangent orientations estimated by this method for an image of a circle using a variety of window sizes. The tangents estimated using a 5×5 window appear better than those from the 3×3, but this comes at a computational cost, since the number of calculations is proportional to the square of the window size. Moreover, the "best" choice of window size depends upon the scale of the input contour. If the window size becomes large compared to the contour, the dominant eigenvector of the covariance matrix becomes a poor estimator of the tangent, since the window is likely to include extraneous parts of the contour. This is clear from the tangent estimates shown in Figure 9(d). Consequently, we choose to use a 3×3 window, both for its computational advantage and because it makes no assumptions about the scale of the input contour.



(a) Original Image (b) 3×3 Window (c) 5×5 Window (d) 35×35 Window

Figure 9: Tangent estimation with varying window sizes, using the eigenvector corresponding to the largest eigenvalue of the covariance matrix.

4.2.2 Discontinuities

Estimating this mapping is difficult for a neural network, as it has two discontinuities. The first arises when the dominant eigenvalue changes, and the orientation of the tangent estimate jumps by $\frac{\pi}{2}$ radians. The second is due to the change in sign of the tangent vector when the orientation goes from π back to 0.

The first discontinuity can be avoided by using a weighted tangent estimate: the magnitude of the estimated tangent vector corresponds to "orientation strength". Let the eigenvalues of the covariance matrix be λ_1 and λ_2, where $\lambda_1 \geq \lambda_2$. The corresponding unit eigenvectors are $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$. The weighted tangent vector estimate $\mathbf{s}$ is given by

$$\mathbf{s} = \begin{cases} \left(1 - \left|\frac{\lambda_2}{\lambda_1}\right|\right) \hat{\mathbf{e}}_1 & \text{if } \lambda_1 \neq 0, \\ 0 & \text{if } \lambda_1 = 0. \end{cases} \tag{41}$$

This weighting cause the magnitude of the estimated tangent vector to go to zero as $\left|\frac{\lambda_2}{\lambda_1}\right| \to 1$, and thus the $\frac{\pi}{2}$ jump is avoided.

4.2.3 Training a Neural Orientation Extraction Module

A neural orientation extraction module (NOEM) is required to estimate this mapping. It will be replicated across the input layer to estimate the tangent at every point, resulting in two "tangent images", one for each vector component. The values are gated by the input image (used as dynamic weights), so that values are only transmitted for ones in the input . It was decided to produce this module by training a network on this task.

A training set was produced consisting of all 256 possible binary 3×3 input windows with a centre value of 1, and, for each, the two outputs given by Eqs. (41). The performance measure chosen, E, was the ratio of the sum squared error to a variance measure for the training set,

$$E = \frac{\sum_{c=1}^{N} \sum_{j=1}^{M} \left(t_{cj} - y_{cj}\right)^2}{\sum_{c=1}^{N} \sum_{j=1}^{M} \left(\overline{t_{cj}} - t_{cj}\right)^2} \tag{42}$$

where N is the number of training exemplars, M the number of nodes in the output layer, t_{cj} the desired output value of the node and y_{cj} the observed output value.

A variety of multilayer perceptrons (MLPs), with single hidden layers of differing sizes, was trained, using backpropagation [30]. The accuracy of the approximation improved with increasing hidden layer size (as expected for such a highly nonlinear function), and the training time increased. The residual error did not stabilize, but continued to decrease steadily, albeit very slowly, as training was continued. For the work reported in [37] a network with a 20 node hidden layer was used. After 6×10^6 iterations with a learning rate of 0.0005, a value of $E = 3.11\%$ was reached.

Further investigations (see Section 5.1) have shown that such seemingly small errors in the Local Orientation Extraction module can adversely affect classification performance. Consequently, a more accurate module was sought.

Although single hidden layer MLPs are universal approximators, it has been suggested that networks with multiple hidden layers can be useful for extremely nonlinear problems [32, 34]. Consequently, several four-layer MLPs were tried. They converged more rapidly, and to a smaller residual error. However, as with the single hidden layer networks, they were very sensitive to the learning rate, suggesting that a more sophisticated learning algorithm might be more appropriate. After an initial large reduction, the error continued to decrease very slowly throughout training. Final accuracy was determined by how long one was prepared to wait, as much as by the number of hidden nodes. The network finally chosen had a residual error of 1.07%.

This problem is similar to edge extraction, although edge extraction is usually performed on grey-scale images rather than binary, thin contours. Srinivasan *et al.* [38] developed a neural network edge detector which produced a weighted vector output much like that described in Eq. (41). They used an encoder stage which was trained competitively, followed by a backpropagation-trained stage. The encoder produced weight distributions resembling Gabor filters of various orientations and phases. A more compact and accurate tangent estimator might be developed using an MBNN incorporating a stage with Gabor weighting functions, as used in [7].

4.3 Calculation of the Local Measure of Consistency

The next stage of the network computes the Local Measure of Consistency at each point with with respect to each of the Lie vector fields. The output is an *Invariance Image* for each Lie transformation: an image in which the value at each point is the absolute value of the dot product of the estimated tangent vector and the Lie vector field (see Eq. (17)).

In the neural implementation, (see Figure 6), the x-component image from the Local Orientation Extractor provides dynamic weights to be combined with the x-component of each vector fields. The same is done for the y-components. For each Lie transformation, there is thus a layer of neurons, each of which have two inputs: one from each the vector field component image. These are weighted by the corresponding Local Orientation images. Zeros in the input image have tangent estimates of zero magnitude and thus make no contribution.

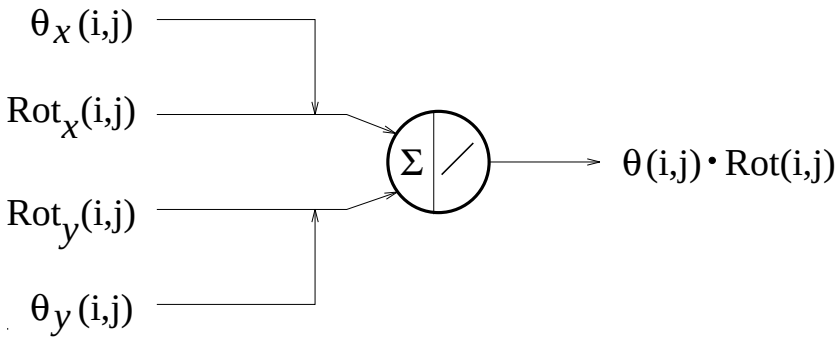


Figure 10: Calculation of the dot product of the tangent estimate θ and the vector field corresponding to rotation, for the image point at coordinates (i, j).

Consider the subsystem corresponding to input image point (i, j). Figure 10 shows the first stage of the calculation. Modules of this type are replicated for each point in the input image, and for each Lie vector field image. The outputs of these modules are then passed through modules which calculate their absolute values.

Figure 11 shows a neural module which calculates a good approximation of the absolute value of its input. It is less accurate for inputs very close to zero, but in this application, where possible orientations are coarsely quantized, this does not cause any problems. This completes the neural calculation of the Local Measure of Consistency for each point with respect to each Lie vector field. All that remains is to combine these into an Invariance Signature.

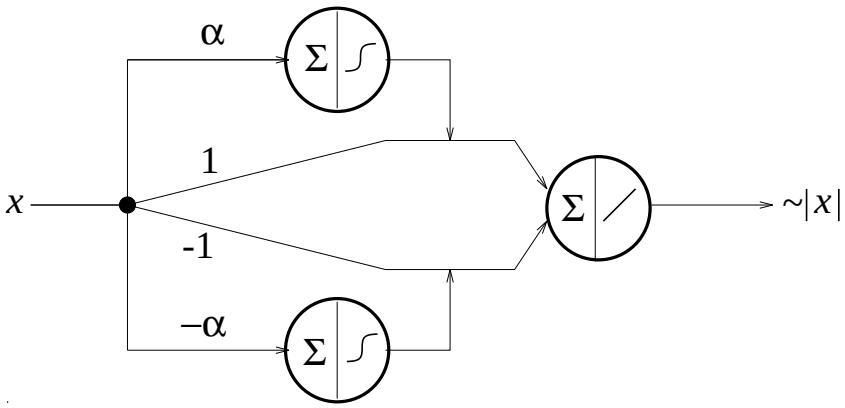


Figure 11: Neural module which calculates the absolute value of its input.

4.4 Calculation of the Invariance Signature

The MBNN approach can produce modules that perform tasks not usually associated with neural networks, as illustrated by the binning module in Figure 12. There is one such module for each of the n bins of the Invariance Signature histogram for each invariance class. Each binning module is connected to all nodes in the Invariance Image. The inputs are gated by the input image, ensuring that only the N "on" nodes contribute.

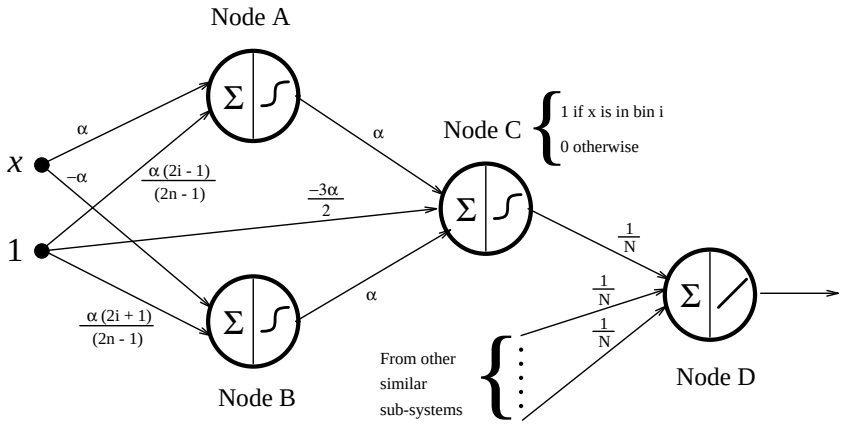


Figure 12: Neural binning module.

The n bins have width $\frac{1}{n-1}$, since the first bin is centred on 0, and the last on 1 (Bins ending at the extrema of the range would miss contributions from the extreme values, since the edges of neural bins are not vertical). Nodes A and B have an output of 1 only when the input x is within bin i, that is:

$$\frac{2i-1}{2(n-1)} < x < \frac{2i+1}{2(n-1)}. \tag{43}$$

In order to detect this condition, the activations of Nodes A and B are set to:

$$net_A = \alpha x - \frac{\alpha(2i-1)}{2(n-1)} \tag{44}$$

$$net_B = -\alpha x + \frac{\alpha(2i+1)}{2(n-1)}. \tag{45}$$

The outputs of these nodes go to Node C, which computes a neural AND. Node D sums the contributions to bin i from all N nodes.

This concludes the calculation of the Invariance Signature. The result is a signature consisting of $3n$ values, where there are n bins in the histogram for each invariance class. This calculation has been achieved using a modular MBNN, where module functions are pre-defined, and are independent of any training data. The building blocks of the modules are restricted to simple summation artificial neurons, with either linear or sigmoidal transfer functions. The sole departure from standard neural network components is the introduction of dynamic weights, but these can be eliminated if product neurons are used, as in Higher Order Neural Networks [25, 35]. This shows the power of the modular approach to neural network design, and demonstrates that modules can be designed which perform tasks that are not often considered to be part of the neural networks domain, such as binning data.

A final module can be added, which is trained to classify patterns on the basis of the Invariance Signature, rather than on the input patterns themselves. Classification performance will thus be guaranteed to be invariant under shift, rotation and scaling. The advantages of this approach are demonstrated in Section 5.

5 Character Recognition with Invariance Signature Networks

It remains now to demonstrate that Invariance Signatures retain enough information to be usable for pattern recognition, and that they are not unduly sensitive to the noisy data encountered in real applications. To this end, the system is applied to the classification of Roman alphabetic characters, both for "perfect" machine-generated training and test data, and for scanned data. The term "perfect" will be used throughout to describe data which is both noise-free and unambiguous.

5.1 Perfect Data

5.1.1 Departures from Exact Invariance

Despite the proven invariance properties of Invariance Signatures calculated for continuous contours in the plane, departures from invariance occur in real applications in several ways. Scanned data contains noise from the sensor, although the present quality of scanners makes this negligible for this application. More important sources of error are discussed below.

Quantization Noise Noise is introduced into the tangent estimation procedure by the sampling of the contour. Since the estimated orientation is quantized, the Local Measure of Consistency can change when a contour is quantized at a new orientation. It is possible to compensate partially for this effect by using sufficiently wide bins when calculating the Invariance Signature, but errors still arise when the new estimated orientation moves the resultant ι_G across bin boundaries.

Ambiguous Characters In many fonts some letters are rotated or reflected versions of others, such as {b, d, p, q} and {n, u}. Consequently, it is impossible to classify isolated characters into 26 classes if shift, rotation, scale and reflection invariance is desired. Admittedly, reflection invariance is not usually desired, but it is an characteristic of the ISNNC. In commercial OCR systems, context information (*i.e.* surrounding letter classifications and a dictionary) is used to resolve ambiguities, which occur even in systems without inherent invariances. This approach would be equally applicable as a post-processing stage for the ISNNC.

5.1.2 The Data Set

A computer can be used to produce a perfect data set, which is free from quantization noise and contains no ambiguous characters. This set can be used to show that Invariance Signatures retain sufficient information for classification in the absence of noise, and these results can be used to assess the performance of the system on real data.

A training data set was created using a screen version of the Helvetica font. Only the letters {a, b, c, e, f, g, h, i, j, k, l, m, n, o, r, s, t, v, w, x, y, z} were used, so that ambiguity was avoided.

An 18×18 binary image of each letter was produced. This training data set is shown in Figure 13.

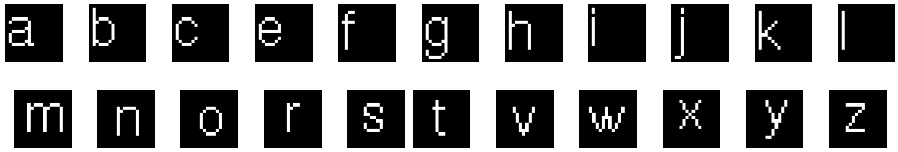


Figure 13: Training set of canonical examples of unambiguous characters.

A perfect test data set was created by computing reflected and rotated versions of the training data, where rotations were by multiples of $\frac{\pi}{2}$ radians, so that there was no quantization error. This test data set is shown in Figure 14.

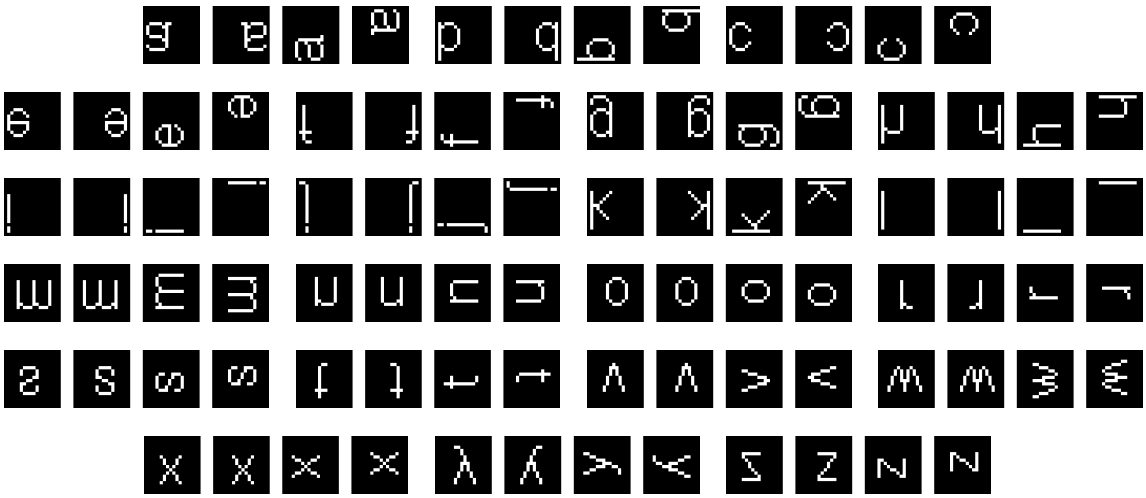


Figure 14: Test set of ideally shifted, rotated and reflected letters.

5.1.3 Selected Networks Applied to this Problem

Simulations showed that this training set could be learnt by a network with no hidden layer: it is linearly separable [24]. This was was also be true of the Invariance Signatures calculated from these data. Two different network architectures were constructed for comparison. The first was a MLP with a 18×18 input layer, no hidden layers, and a 1×22 output layer. The other was an ISNNC, with an 18×18 input layer. Within this ISNNC, the Invariance Signature was calculated using 5 bins for each transformation. The resultant 15 node Invariance Signature layer was connected to a 1×22 output layer, forming a linear classifier sub-network.

5.1.4 Reduction of Data Dimensionality

Although the Invariance Signature calculation stage of the ISNNC has to be run every time a new pattern is classified, Invariance Signatures of the training and test data need only be calculated once. The classification stage of the ISNNC can then be trained as a separate module. This can lead to a great reduction in the training time. The number of weights n_p in a TNN is:

$$n_p = \sum_{i=1}^{N-1} (\text{nodes in layer})_{i-1} \times (\text{nodes in layer})_i \tag{46}$$

where N is the total number of layers, and i is the layer number, (the input layer is layer 0). The iteration time during training is proportional to n_p, so, for this example, each iteration for the Invariance Signature classification module will be $\frac{18 \times 18}{3 \times 5} = 21.6$ times faster than for the TNN.

The calculation of the Invariance Signatures is time-consuming, but this time is recouped during training when the input image is large, which is typically the case in real applications. ISNNCs can reduce the dimensionality of the training data significantly, and thus the training time for the classification network. Moreover, an ISNNC simulation on a sequential computer cannot take advantage of the parallel, local computations that characterize many of the ISNNC modules. A parallel implementation would be much faster.

5.1.5 Perfect and Network-Estimated Local Orientation

Since the NOEM (Section 4.2) has some residual error, two versions of the Invariance Signature training and test data were created, one with the tangents calculated directly from the covariance matrix eigenvectors, and the other using the NOEM. Results from these are compared to evaluate the importance of accurate tangent extraction.

Ten instances of each network were made, each with a different parameter initialization. All were trained for 1000 iterations, using backpropagation. Patterns were assigned to the class corresponding to the highest output node value, so no patterns were rejected.

Results for Traditional Neural Networks The results obtained with TNNs are summarized in Table 1. As expected, TNNs do not exhibit invariance, since the training data contained no transformed versions of the patterns.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
$\mu \pm \sigma$	100.00 ± 0.00	15.00 ± 0.45	100.00 ± 0.00	13.64 ± 0.00

Table 1: Classification performance (% correct) of traditional neural network classifiers trained for 1000 iterations with the data shown in Figure 13 and tested with the perfect data set in Figure 14.

The final performance of the TNNs is better than chance ($\frac{1}{22} = 4.5454\%$). It might be thought that this is because some transformations of highly symmetrical training patterns resulted in patterns very similar to the untransformed version (*e.g.*o, s, x and z). Analysis, however, shows that this is not the case. The 12 correctly classified test patterns are shown in Figure 15.

Figure 15: Test patterns classified correctly by the TNNs.

No reason for these patterns being classified correctly is apparent. It seems clear that chance must play a part. For instance, i4 shares no "on" pixels with the training example of i. Marginally better performance on the test data could have been achieved by employing an early-stopping scheme. This would, however, have been at the cost of less than 100% correct performance on the training data. It should be noted that the sum squared-error on the test set decreased throughout training. A scheme based on the test set error would not improve performance in this case.

It must be acknowledged that TNNs could not be expected to perform better than chance on this task. Their architecture provides no invariances, and generalization cannot be expected unless transformed versions of the patterns are included in the training set. This argument can be used against all the comparisons between TNNs and MBNNs in this paper: they are not fair. Nevertheless, these comparisons between naive applications of TNNs and specifically-designed MBNNs demonstrate that MBNNs can perform successfully using training sets completely inadequate for TNNs. Moreover, these MBNNs are of lower dimensionality than the TNNs. Providing the TNNs with sufficiently large training sets would only make their training still more computationally-expensive, with no *guarantee* of invariant performance.

Results with Perfect Local Orientation The results for the ten ISNNCs which used perfect Local Orientation Extraction are summarized in Table 2. The average number of iterations for 100% correct classification to be achieved was 220. Since the problem is linearly-separable, it could in fact be solved directly, using a technique such as singular-valued decomposition [27]. Since weights may set by any method at all in the MBNN paradigm, this makes the comparison of convergence times somewhat irrelevant. Nevertheless the MBNN modules, although taking on average 4.4 times as many iterations to converge as the TNNs, were 4.9 times faster to train, due to their lower dimensionality.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
$\mu \pm \sigma$	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00

Table 2: Classification performance (% correct) of Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 13 and tested with the perfect data set in Figure 14.

The ISNNCs generalize perfectly to the the test set. The network architecture constrains the system to be shift-, rotation-, scale- and reflection-invariant in the absence of quantization noise, so this is no surprise. Importantly, the result indicates that sufficient information is retained in the 5 bin Invariance Signatures for all 22 unambiguous letters of the alphabet to be distinguished. Inspection of the sum squared error values after each iteration indicated that the error on the test set was indeed identical to that on the training set: for perfect data, the ISNNC produces perfect results.

Results using the NOEM The results above were obtained using a hybrid system, which used a non-neural module to calculate the tangent at each point. The NOEM used here was trained using backpropagation until 98.93% of the variance in the training data was explained (see Eq. (42)). The Invariance Signatures for the test and training data were recalculated using this module, and classification modules were trained using these Invariance Signatures. Systems were produced with both 5 and 10 bin Invariance Signatures. The results obtained are summarized in Tables 3 and 4.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
$\mu \pm \sigma$	100.00 ± 0.00	96.591 ± 0.00	100.00 ± 0.00	96.591 ± 0.00

Table 3: Classification performance (% correct) of 5 Bin Invariance Signature Neural Network Classifiers (with neural Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 13 and tested with the perfect data set in Figure 14.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
$\mu \pm \sigma$	100.00 ± 0.00	95.455 ± 0.00	100.00 ± 0.00	93.182 ± 0.00

Table 4: Classification performance (% correct) of 10 Bin Invariance Signature Neural Network Classifiers (with neural Local Orientation Extraction) trained for 1000 iterations with the data shown in Figure 13 and tested with the perfect data set in Figure 14.

The misclassified patterns for the 5 bin ISNNCs are shown in Figure 16. t1 and t2 were classified as f, which is understandable, since there is very little difference between the patterns. t4 was misclassified as a. All ten networks had these same misclassifications.

These results show that residual error in the NOEM causes a degradation of the classification performance of the ISNNCs. They are, however, still far superior to those for the TNNs. Moreover, there is no reason that the NOEM could not be trained further. It is expected that performance would continue to approach 100% as the accuracy of the module was improved.

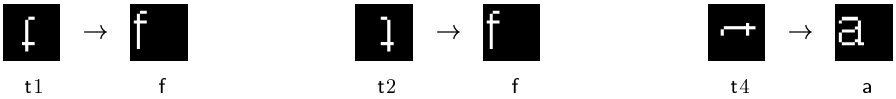


Figure 16: Test Patterns Misclassified by the 5 Bin Invariance Signature Neural Network Classifiers, and the training examples as which they were incorrectly classified.

The results for the 10 bin system in Table 4 show that the effects of inaccuracies in the NOEM are greater when the number of bins is increased. This is due to the fact that the errors can cause the consistency measure at a point to change bins more easily this way, thus altering the Invariance Signature histogram.

5.2 Optical Character Recognition

Having demonstrated that Invariance Signatures retain sufficient information for the classification of "perfect" data, it remains to show that the system can be used for transformation invariant pattern recognition in the presence of sensor and quantization noise. To this end, it was decided to apply ISNNCs to the classification of scanned images of shifted and rotated printed alphabetic characters.

5.2.1 The Data Set

The training and test sets were created using all 26 letters of the English alphabet, with each character appearing in 18 different orientations, at increments of 20 degrees. Shifts arose also, because characters were extracted from the scanned image of all these characters using a technique which took no account of centroid position.

An A4 page with these characters printed on it by a laser printer at 300 dots per inch was scanned at 75 dots per inch using a UMAX Vista-S6 Scanner. The connected regions in this image were detected, and the minimum bounding-box for the largest character was calculated (56×57 pixels). The image was then segmented into separate characters, and each character was thinned using an algorithm due to Chen and Hsu [9]. The training set consisted of the 234 characters rotated by angles in the range $[0°, 160°]$ relative to the upright characters, and the test set of those in the range $[180°, 340°]$. Examples of the resultant scanned, extracted and thinned characters are shown in Figures 17 and 18.

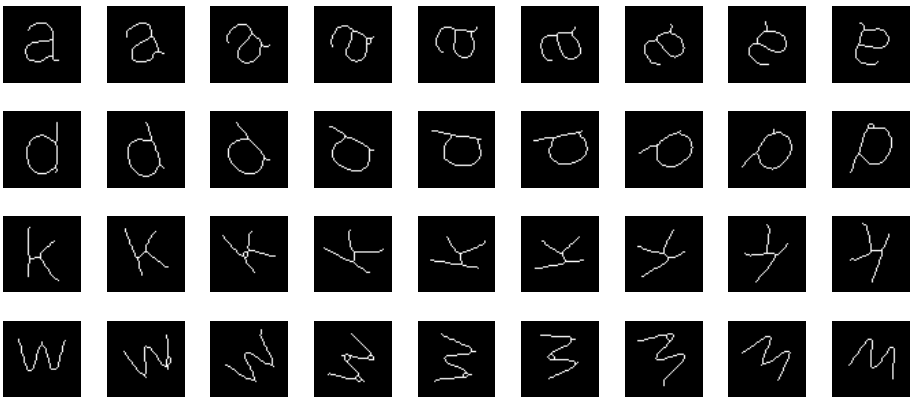


Figure 17: Examples from the training set of thinned, scanned characters.

It is interesting to compare the subjective visual similarity between the Invariance Signatures both within and between classes for these data. Figure 19 shows the first four training examples of the letter a, accompanied by images showing the tangent estimates. The tangent estimate is represented by a line segment with the orientation of the estimated tangent. The weight assigned to each estimated tangent is not shown.

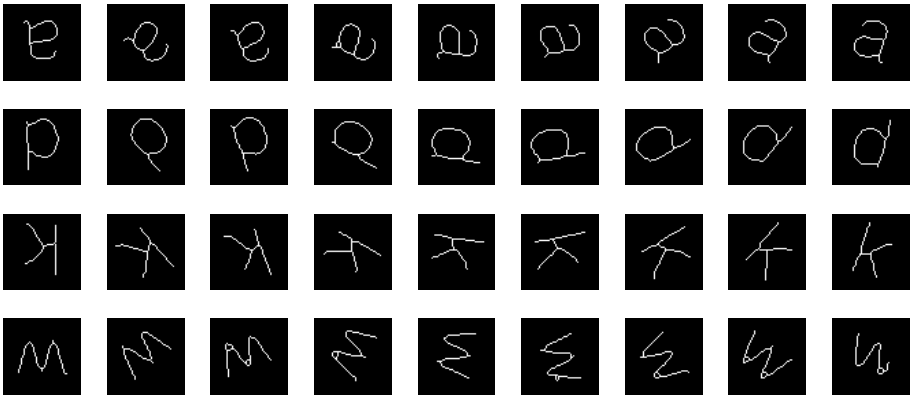


Figure 18: Examples from the test set of thinned, scanned characters.

It is not easy to interpret the similarity between these tangent representations of the contours. For this, it is necessary to see the Invariance Signatures. Figure 20 shows the Invariance Signature histograms for the patterns in Figure 19. It is apparent that there is a great (subjective) similarity between these representations. They are not identical, as a result of the noise discussed in Section 5.1. For classification purposes, however, it is necessary only that the signatures be more similar within each letter class than between classes. This must be determined by experiment.

Figures 21 and 22 show the equivalent data for the letter x. These again show the marked within-class similarity, and are distinctly different to the signatures for the letter a: these letters were deliberately chosen since a is "quite rotationally-invariant", whereas x is "quite dilationally-invariant".

5.2.2 Selected Networks Employed for this Problem

The methodology employed was identical to that used for the synthetic data, described in Section 5.1.3. The TNN used had a 56×57 node input layer, and a 1×26 output layer, giving a massive 83018 independent weights to be estimated. With only 234 training patterns and 83018 parameters, this problem is almost certain to be linearly separable. This was verified by simulation.

A variety of classification modules was tried, for ISNNCs with both 5 and 10 bin Invariance Signatures. These included linear classifier, and a variety of MLPs with differing hidden layer sizes. These experiments indicated that 5 bin Invariance Signatures were insufficient for this problem, and that it was not linearly separable. It also became clear that the errors introduced by the slightly inaccurate NOEM caused a significant departure from invariance.

For these reasons, the results presented are for 10 bin ISNNCs with directly-MLP classification module. The MLP classifier had a 3×10 node input layer, a 15 node hidden layer, and a 26 node output layer. This classifier has only 881 weights to be estimated, a reduction of 99% compared to the TNN linear classifier. This translates to a dramatic reduction in both the storage space and the training time required. Only five TNNs were trained, partly because of the training time needed, and partly because the results were so consistent.

5.2.3 Results for Traditional Neural Networks

It might have been expected that the TNNs would perform better on this task than on that described in Section 5.1.3, since this training set contains differently transformed versions of the canonical untransformed characters. As can be seen from Table 5, the results are in fact slightly worse (average best percent correct of 13.932 ± 0.300 compared with 15.000 ± 0.454). Although better than chance (3.85% correct), these results are completely inadequate for an optical character recognition system.

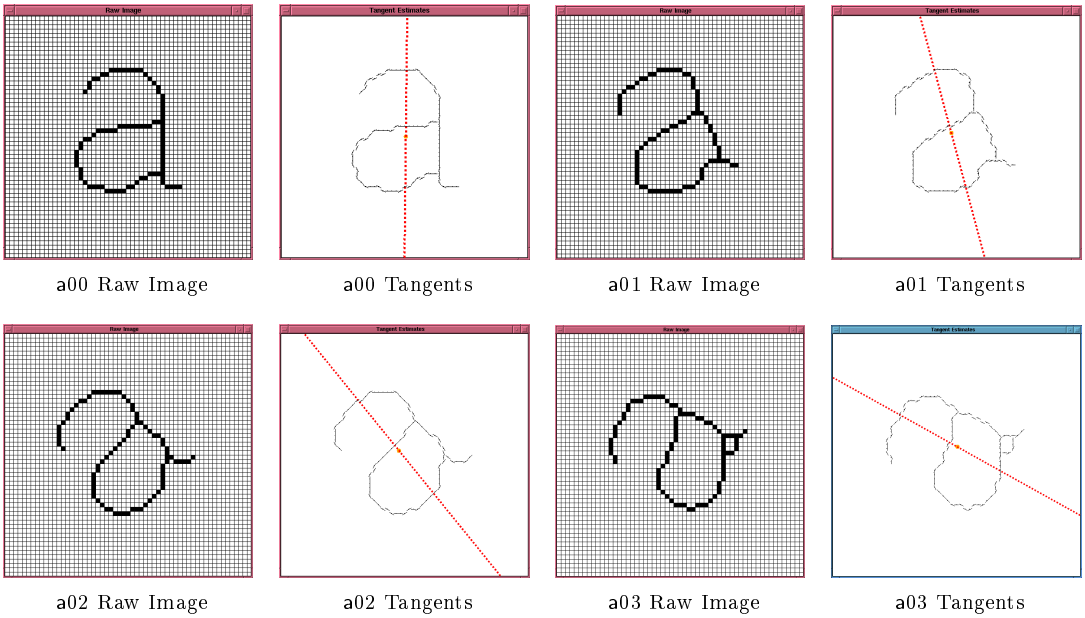


Figure 19: Tangents estimated for training examples of the letter a.

5.2.4 Results for Invariance Signature Neural Network Classifiers

The results obtained with ISNNCs appear in Table 6. The ISNNCs achieved a much higher correct classification rate on the test set than the TNNs. The failure of the ISNNCs to achieve 100% correct classification of the training set is not surprising. The test and training sets used for this problem have each character of the alphabet mapped to a separate class. Yet, as was discussed in Section 5.1, the sets of characters {b, d, p, q} and {n, u} are identical under rotations and reflections: transformations under which the ISNNC output is invariant. The expected training set performance for noise-free data is thus $100 \times (20 + 0.25 \times 4 + 0.5 \times 2)/26 = 85\%$ correct. Any performance on the training data better than this must be the result of the fitting of noise in the training data.

In order to assess this effect, training and test sets were created in which {b, d, p, q} were assigned the same label, as were {n, u}. The results are shown in Table 7. The average final test set performance was improved by 14.8% by this re-labeling, which is very close to the maximum possible 15.4% achievable if this were the only source of error.

The residual difference between training and test set error is generalization error, rather than invariance error. These networks were trained with only 9 examples of each character, and these examples are quite noisy. There is the unavoidable quantization noise, but there are also some quite marked artifacts, such as loops introduced by the thinning algorithm, one of which can be seen in pattern a03 in Figure 20. There are also two erroneous test patterns, the result of clipping in the segmentation process. These were retained, as such errors can and do occur in practical applications.

5.2.5 Failure Analysis

The errors made by the ISNNCs are not random. To illustrate this, a failure analysis is presented in Table 8 for Network 5 from Table 7, showing how the test patterns were misclassified. Patterns which are perceptually similar are responsible for many of the misclassifications. This means that prior information about likely errors could be used in conjunction with these classifications to aid error correction. This analysis indicates that ISNNCs often make errors that appear "human", which is a promising indication that the Invariance Signature measures contour similarity in a way similar to humans.

Inspection of the thinned patterns used indicated that the patterns for the letters {i, j, l} were

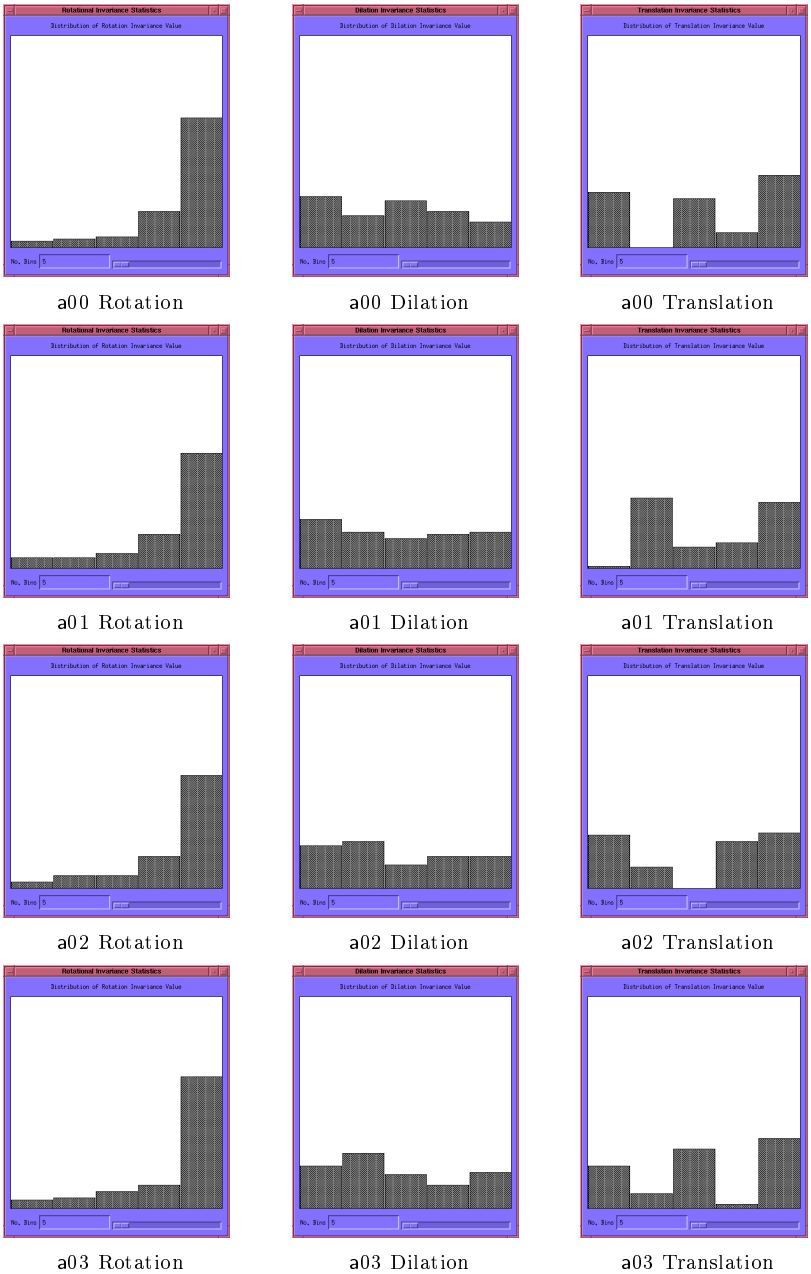


Figure 20: 5 bin Invariance Signatures for training examples of the letter a.

little more than straight lines. If these were to be re-labeled as the same character, final test set performance for Network 5 would improve to 91.026% correct. If the same were done for {f, t}, which are also extremely similar, test set performance would be 93.162% correct.

The misclassifications of patterns to the classes b and n may be due to the fact that the re-labeled dataset implies a non-uniform prior probability distribution for these classes: b was used as the target class for input patterns corresponding to {b, d, p, q}, and thus occurs four times more frequently in the training data than standard classes. Similarly, n was used as the target for inputs {n, u}. Networks will tend to "guess" these classes more frequently than the others [3].

Given the extremely small training set, this is a remarkable result, comparable with character recognition results achieved by others with thousands of training patterns. Such re-relabeling is an essential feature of a *truly* invariant optical character recognition system, since some characters are inherently ambiguous. Others are extremely similar, and noise can render them indistinguishable.

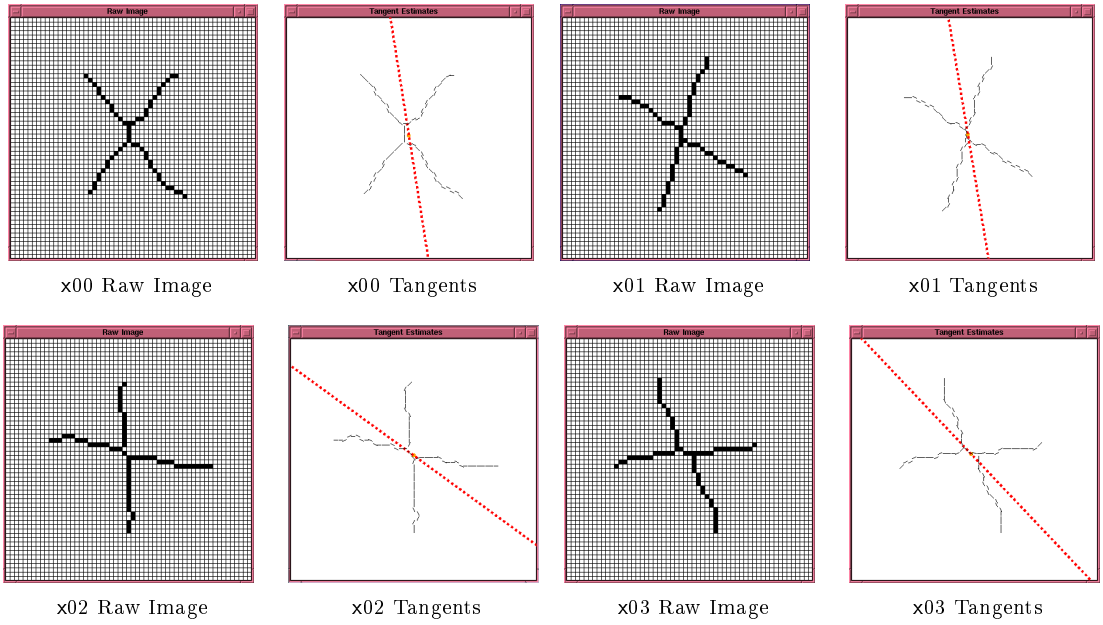


Figure 21: Tangents estimated for training examples of the letter x.

In practical optical character recognition systems, a dictionary is used to verify recognized words, and context is used to correct erroneously labeled patterns. Alternatively, the orientation of correctly classified unambiguous characters could be used to infer the correct labeling of ambiguous characters.

Perhaps most importantly, these results show that ISNNCs can correctly classify patterns which have been transformed by arbitrarily large amounts. The shifts and rotations of the input images are not restricted to small perturbations of the training patterns. This indicates that the ISNNCs are performing truly invariant pattern recognition, rather than interpolation-based generalization.

This study should be considered to be a "proof of concept", both for the Invariance Signature as a contour descriptor, and for MBNNs which classify on that basis. It is not intended to be a large-scale experiment which corresponds to a real application. Such a study would require much greater quantities of data than are used here. We believe, however, that the experiments presented demonstrate that the Invariance Signatures are genuinely useful for robust invariant pattern recognition. The quality of the results obtained is very pleasing, when the size of the training sets is compared to those used in other neural network approaches to optical character recognition.

6 Conclusion

In Section 2, a new invariant feature of two-dimensional contours was developed: the Invariance Signature. We believe that the Invariance Signature is a powerful descriptor of contour shape, which is closely-related to measures employed in human perception. Its application is by no means limited to neural networks.

The development of the Invariance Signature was inspired by the desire to find an invariant contour descriptor which was suitable for calculation in a neural network, and which corresponded well to theories of human contour perception. Since Lie group theory provides the link between the local changes in the positions of points under the action of a transformation and the global specification of the transformation, it provides the natural starting point. The Invariance Signature is a global measure of the degree of invariance of a given contour with respect to a set of Lie transformations, which, however, is constructed from *local* calculations. It is this that makes the Invariance Signature attractive for use in an MBNN.

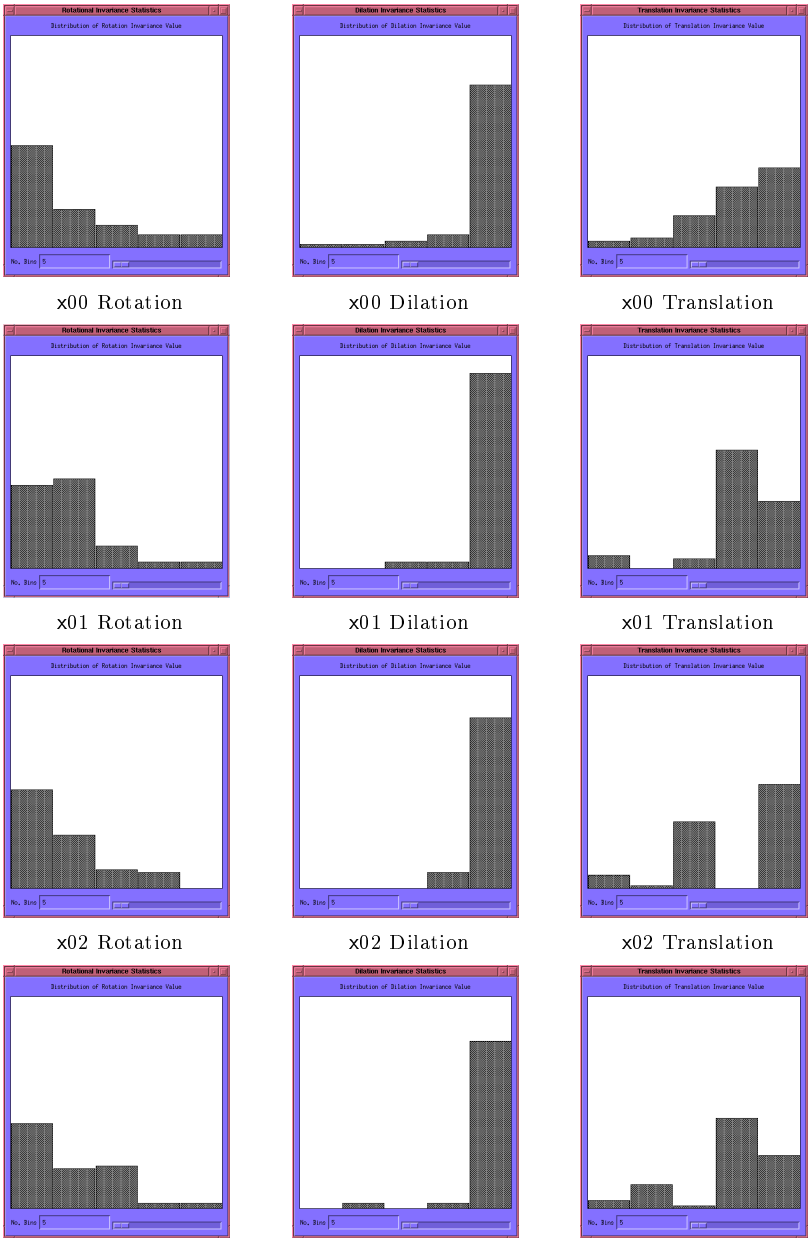


Figure 22: 5 bin Invariance Signatures for training examples of the letter x.

The core of the Invariance Signature approach is this: rather than seeking individual invariant features of a contour, the Invariance Signature measures the *degree to which the contour is invariant* under a transformation. The statistics of these departures from invariance are themselves an invariant descriptor of the contour.

In Section 4, it was shown that an MBNN could be constructed which calculated the Invariance Signature of a contour, and used this as the basis for classification. Since the Invariance Signature is shift-, rotation-, scale- and reflection-invariant, the output of this network is *guaranteed* to be invariant under these transformations. This network is called the Invariance Signature Neural Network Classifier. The ISNNC consists of a wide variety of modules, which perform tasks as diverse as tangent vector estimation and the binning of data. The weights of all modules in the ISNNC except the final MLP classifier are specified independently of the training set; some are specified directly, others are determined by training the module on a sub-task. The result is a

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
$\mu \pm \sigma$	100.00 ± 0.00	13.932 ± 0.300	100.00 ± 0.00	10.940 ± 0.209

Table 5: Classification performance (% correct) of Traditional Neural Network Classifiers trained for 200 iterations with the data described in Section 5.2.1.

collection of simple nodes joined by weighted connections, as in traditional neural network.

In order to be useful, the Invariance Signature must not only be invariant, but must retain sufficient information for contour classes to be distinguished. That this is so is demonstrated in Section 5. When applied to noise-free, unambiguous data, the ISNNC produced perfect results. Excellent results were also obtained using scanned data. It might be said that the comparisons were unfair, since the training sets used were inadequate for good TNN performance to be expected. That only emphasizes one of the key advantages of the MBNN approach: large training sets are not required.

For both the perfect data and the scanned character task, the dimensionality of the classification module was very much lower for the ISNNCs than it was for the TNNs. The training time for the ISNNCs was correspondingly shorter. Moreover, the dimensionality of the ISNNC classifier can be varied by the designer, by changing the number of bins for the discrete Invariance Signatures.

These experiments indicate that the Invariance Signature can be successfully employed for the recognition of scanned characters independent of rotations and shifts, and that this technique can be implemented in a MBNN. Since the ISNNCs are guaranteed to be invariant under shift, scaling, rotation and reflection, and training set performance on the noisy data was 99.06 ± 0.63 percent correct, it can be concluded that test set performance below this level is due to failure to generalize in the presence of noise, not failure to generalize in the sense of invariance. If the size of the training set were increased, test set performance could be expected to approach 100 percent correct.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	95.726	72.650	95.726	71.795
Network 2	96.154	72.222	96.154	70.940
Network 3	96.154	73.077	96.154	69.658
Network 4	95.299	73.932	95.299	70.513
Network 5	94.872	71.795	94.872	71.368
Network 6	95.726	72.650	95.726	68.803
Network 7	96.154	73.504	96.154	69.658
Network 8	97.436	72.222	97.436	70.513
Network 9	94.444	74.359	94.444	69.658
Network 10	96.154	70.940	96.154	70.940
$\mu \pm \sigma$	95.812 ± 0.783	72.735 ± 0.971	95.812 ± 0.783	70.385 ± 0.877

Table 6: Classification performance (% correct) of 10 bin Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 40000 iterations with the data described in Section 5.2.1.

	Best Performance		Final Performance	
	Training Data	Test Data	Training Data	Test Data
Network 1	98.718	87.179	98.718	87.179
Network 2	99.145	83.761	99.145	82.906
Network 3	99.573	86.752	99.145	85.897
Network 4	99.573	86.752	99.145	85.043
Network 5	98.718	88.034	98.718	88.034
Network 6	99.573	85.897	99.145	85.470
Network 7	100.00	85.897	100.00	83.761
Network 8	98.291	86.325	98.291	85.043
Network 9	97.863	86.752	97.863	85.043
Network 10	99.145	84.615	99.145	83.761
$\mu \pm \sigma$	99.056 ± 0.628	86.196 ± 1.179	99.056 ± 0.628	85.214 ± 1.483

Table 7: Classification performance (% correct) of 10 bin Invariance Signature Neural Network Classifiers (with perfect Local Orientation Extraction) trained for 10000 iterations with the data described in Section 5.2.1, modified to label characters which can be transformed into each other as the same character.

Misclassifications						
d → n	f → t	f → j	i → l	i → l	i → l	i → l
i → l	i → l	j → r	k → f	k → r	k → f	l → i
m → h	n → b	n → b	q → n	r → f	r → y	r → k
r → i	t → f	t → f	t → f	t → f	u → h	w → m

Table 8: Failure analysis for Network 5 from Table 7.

References

[1] Jürgen Altmann and Herbert J.P. Reitböck. A fast correlation method for scale- and translation-invariant pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):46–57, January 1984.

[2] Dana Harry Ballard and Christopher M. Brown. *Computer Vision*. Prentice–Hall, Englewood Cliffs, New Jersey 07632, 1982.

[3] Etienne Barnard and Elizabeth C. Botha. Back-propagation uses prior information efficiently. *IEEE Transactions on Neural Networks*, 4(5):794–802, September 1993.

[4] A. Bulsari. Some analytical solutions to the general approximation problem for feedforward neural networks. *Neural Networks*, 6(7):991–996, 1993.

[5] Terry Caelli and Peter Dodwell. Orientation-position coding and invariance characteristics of pattern discrimination. *Perception and Psychophysics*, 36(2):159–168, 1984.

[6] Terry M. Caelli and Zhi-Qiang Liu. On the minimum number of templates required for shift, rotation and size invariant pattern recognition. *Pattern Recognition*, 21(3):205–216, 1988.

[7] Terry M. Caelli, David McG. Squire, and Tom P.J. Wild. Model-based neural networks. *Neural Networks*, 6:613–625, 1993.

[8] T.M. Caelli, G.A.N. Preston, and E.R. Howell. Implications of spatial summation models for processes of contour perception: A geometric perspective. *Vision Research*, 18:723–734, 1978.

[9] Y-S. Chen and W-H. Hsu. A modified fast parallel algorithm for thinning digital patterns. *Pattern Recognition*, 7:99–106, 1988.

[10] Gloria Chow and Xiaobo Li. Towards a system for automatic facial feature detection. *Pattern Recognition*, 1993.

[11] James B. Cole, Hiroshi Murase, and Seiichiro Naito. A Lie group theoretic approach to the invariance problem in feature extraction and object recognition. *Pattern Recognition Letters*, 12:519–523, September 1991.

[12] M. Ferraro and T. Caelli. The relationship between integral transform invariances and Lie group theory. *Journal of the Optical Society of America (A)*, 5:738–742, 1988.

[13] Mario Ferraro and Terry M. Caelli. Lie transform groups, integral transforms, and invariant pattern recognition. *Spatial Vision*, 8(1):33–44, 1994.

[14] David Forsyth, Joseph L. Mundy, and Andrew Zisserman. Transformational invariance - a primer. *Image and Vision Computing*, 10(1):39–45, 1992.

[15] W. C. Hoffman. The Lie algebra of visual perception. *Journal of Mathematical Psychology*, 3:65–98, 1966.

[16] W. C. Hoffman. The Lie transformation group approach to visual neuropsychology. In E. Leewenberg and H. Buffart, editors, *Formal theories of visual perception*, pages 27–66. Wiley, New York, 1978.

[17] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[18] Anil K. Jain. *Fundamentals of digital image processing*. Prentice-Hall information and system sciences series. Prentice-Hall International, London, 1989.

[19] Alireza Khotanzad and Jiin-Her Lu. Classification of invariant image representations using a neural network. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(6):1028–1038, June 1990.

[20] Reiner Lenz. Group invariant pattern recognition. *Pattern Recognition*, 23(1/2):199–217, 1990.

[21] S.Z. Li. Matching: Invariant to translations, rotations and scale changes. *Pattern Recognition*, 25(6):583–594, 1992.

[22] Xiaobo Li and Nicholas Roeder. Experiments in detecting face contours. In *Vision Interface Conference*, May 1994.

[23] Feng Lin and Robert D. Brandt. Towards absolute invariants of images under translation, rotation and dilation. *Pattern Recognition Letters*, 14:369–379, May 1993.

[24] Marvin Minsky and Seymour Papert. *Perceptrons. An introduction to computational geometry.* The MIT Press, Cambridge, London, 1969.

[25] Stavros J. Perantonis and Paulo J.G. Lisboa. Translation, rotation, and scale invariant pattern recognition by higher-order neural networks and moment classifiers. *IEEE Transactions on Neural Networks*, 3(2):241–251, March 1992.

[26] David A. Pintsov. Invariant pattern recognition, symmetry, and radon transforms. *Journal of the Optical Society of America (A)*, 6(10):1544–1554, October 1989.

[27] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing.* Press Syndicate of the University of Cambridge, Cambridge, U.K., 2 edition, 1992.

[28] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, Orlando, FL., 1982.

[29] Jacob Rubinstein, Joseph Segman, and Yehoshua Zeevi. Recognition of distorted patterns by invariance kernels. *Pattern Recognition*, 24(10):959–967, 1991.

[30] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[31] S.L. Salas, Einar Hille, and John T. Anderson. *Calculus: One and several variables, with analytic geometry.* John Wiley & Sons, New York, fifth edition, 1986.

[32] Warren S. Sarle. Neural networks and statistical models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, April 1994.

[33] Joseph Segman, Jacob Rubinstein, and Yehoshua Y. Zeevi. The canonical coordinates method for pattern deformation: Theoretical and computational considerations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(12):1171–1183, December 1992.

[34] Eduardo D. Sontag. Feedback stabilization using two-hidden-layer nets. *IEEE Transactions on Neural Networks*, 3:981–990, 1992.

[35] Lilly Spirkovska and Max B. Reid. Higher-order neural networks applied to 2D and 3D object recognition. *Machine Learning*, 15(2):169–199, 1994.

[36] David McG. Squire. *Model-based Neural Networks for Invariant Pattern Recognition.* PhD thesis, School of Computing, Curtin University of Technology, Perth, Western Australia, October 1996.

[37] David McG. Squire and Terry M. Caelli. Shift, rotation and scale invariant signatures for two-dimensional contours, in a neural network architecture. In Stephen W. Ellacott, John C. Mason, and Iain J. Anderson, editors, *Mathematics of Neural Networks: Models Algorithms and Applications*, Statistics and OR, pages 344–348, Boston, July 1995. Kluwer Academic Publishers.

[38] V. Srinivasan, P. Bhatia, and S.H. Ong. Edge detection using a neural network. *Pattern Recognition*, 27(12):1653–1662, 1994.

[39] Harry Wechsler. *Computational Vision*. Academic Press Inc., 1250 Sixth Avenue, San Diego, CA 92101, 1990.

[40] Christopher Zetzsche and Terry Caelli. Invariant pattern recognition using multiple filter image represtations. *Computer Vision, Graphics and Image Processing*, 45:251–262, 1989.