Real-time Trajectory analysis using stacked invariance estimators

Brendan Kitts[†] Sheila R. Cooke[‡] and Robert F. Sekuler[¿]

[†]*Department of Computer Science,* [‡]*Department of Neuro-science,* [¿]*Department of Psychology,*
Brandeis University, Waltham, MA. 02254
http://www.cs.brandeis.edu/~brendy
Email: brendy@cs.brandies.edu

Abstract

This paper addresses the problem of analysing real-time trajectories ignoring factors such as rotation, translation, and scale. The problem was tackled using an ensemble of techniques including Fourier analysis, a method from statistics called Canonical analysis, and combination methods such as averaging and stacking. Traditional methods performed poorly when measured against human scores, but combination methods produced significantly higher results. This supports theoretical work (Tumer, 1996) which shows that large ensembles of uncorrelated classifiers boosts estimator performance.

1.0 Introduction

This paper addresses the problem of analysing the similarity of real-time trajectories invariant to rotation, translation and scale. The application for this research is open ended, and could be applied to identifying signature aircraft trajectories, handwriting identification, or recognition of sign gestures. The particular application focused on in this paper is testing the ability of humans to imitate visually observed trajectories using a pen and tablet device.

Because of the need to find scores of smilarity invariant to rotation, translation and scale, a large part of this paper is an investigation of the methods used to remove these aspects from the signal. It is shown that although a large number of transforms are available, in practice their usefulness varies, and even sound methods have implications for the form of the signal produced. This paper proposes to solve this problem by applying a technique from machine learning called "stacking" (Briemann, 1992) which averages together a large number of estimators to minimize the bias incurred through using a single invariance method.

2.0 The Trajectory analysis problem

The problem to be addressed is to find a reliable similarity index between two 2D trajectories. This problem has a wide range of applications. The scoring procedure can be used to find a trajectory in a database which best matches an observed trajectory, and so can be used for *trajectory recognition*. The application in this article is a little different. Scores are being used to calculate how well a human being can imitate a real-time computer generated target. In either case, the calculation of a similarity index is clearly a very useful, and very hard problem. Moreover, concentrating on the smoothness, and other aspects of the scores themselves provides a far more stringent criteria for assessment, than simply looking at identification accuracy on some test set.

Data was collected by first creating a set of target or template trajectories, which consisted of (x,y,t) vector for a moving shape which is shown on a computer screen. Human subjects viewed these targets and then had to imitate as best they could by moving the pen on the graphics tablet. Subjects' positions, called *responses*, were recorded by custom software at a rate of 100MHz.

The primary difficulty in analysing these stimuli was that there were virtually no constraints on the targets and responses. There were an arbitrary number of templates, and they could take any form or size. This made it is prohibitive to use neural networks or any other parametric approximation method, to train on a set of prototypes and then calculate the mismatch with the response. Therefore emphasis was placed in finding general mathematical scoring methods.

Figure 1: Two test stimuli. The shape on the left is an idealized shape, the shape on the right is an image drawn by a human being.

2.1 Overview of Scoring procedure

Analysis of trajectories was performed in three steps. First, stimuli were preprocessed to reduce the amount of incomming data, and to smooth over high frequency noise. Second, invariance methods were used to transform each trajectory into a rotation, scale, translation invariant signal. Finally, the two invariant signals were matched using vector cross-correlation to calculate their overall similarity.

3.0 Pre-processing

Before employing the invariance transforms, a number of domain specific effects needed to be dealt with. Firstly it was found that hand tremor in the form of low ampliltude, high frequency noise, was very common and was essentially irrelevant to the match. Hand tremor was removed by applying a process of decimation, interpolation, and smoothing to both target and response. Decimation and smoothing was accomplished by performing a fast fourier interpolation at ×8 reduced resolution. This was favoured over the ordinary method of sliding-average smoothing, which was found in experiments to tend to remove high amplitude information, especially the corners of trajectories (Duda and Hart, 1973). This was followed by single pass of sliding-average smoothing to remove some residual "ringing" which was characteristic to having sharp boundaries approximated using the fast fourier method.

The complete preprocessing involved a single pass of spatial smoothing with *s=2* (simply taking the average of neighbouring points) applied to the function *g* interpolated using *2M* points to give a reduced vector *r*.

$$g(sx) = \frac{1}{s} \sum_{i = x - \frac{s}{2}}^{x + \frac{s}{2}} r(i)$$

$$\text{where } r = F^{-1}(F(g), 2IM)$$

The final vector for *g* has only *IM* points. F^{-1} is the inverse discrete fourier transform taken with 2*IM* equally spaced points.

This procedure also removed the existence of speed-up and slow-down in different segmentes of the trajectory. Subjects typically speeded up on long straight sections, and slowed down on turns, resulting in a greater density of points being recorded in turning regions. Because the target moved at a constant speed, matches based on *(x,y,t)* trajectories were quite poor. Interpolating both trajectories uniformly gave far more reliable empirical performance. Because of interpolation, the *(x,y,t)* vectors became a series of *(x,y)* vectors with time implicit in the number of data components columnwise.

4.0 Invariance Methods

Invariance methods are used widely throughout areas of machine learning, pattern recognition, and signal processing. They are used so widely because in theory they give a system the ability to work with a vector representing some object, which is not affected by how that object finds itself transformed in the environment.

Methods for rotation, scale and translation invariance are discussed in the following section. Some of these turned out to not be good invariance methods, whilst others transformed the signal in ways which distorted the order of neighbouring signals, leading to unsmooth relative classifications for stimuli. Biased signal transforms can cause problems, for instance, improperly compressing the dynamic range of some stimuli so that they appear closer together than they actually are, and performing other warpings of the "stimuli neighbourhood space". This effect warrants particular attention since these methods are routinely used as front-ends to pre-processing stages for pattern recognition tasks, in which such neighbourhood relations are crucial.

<u>Notation</u>

In the following sections $f=(x,y)$ is a column vector of datapoints representing the xy position of the target, and $g=(x,y)$ is a response signal which may be rotated, translated, scaled, and subject to noise. Scale, Translation and Rotation functions are defined as.

$$S(x, y, \Delta) = (x, y) \cdot \begin{bmatrix} \Delta & 0 \\ 0 & \Delta \end{bmatrix}^T$$

$$R(x, y, \Delta) = (x, y) \cdot \begin{bmatrix} \cos(\Delta) & -\sin(\Delta) \\ \sin(\Delta) & \cos(\Delta) \end{bmatrix}^T$$

$$T(x, y, \Delta_x, \Delta_y) = (x', y') \text{ where } (x', y', o) = (x, y, 1) \cdot \begin{bmatrix} 1 & 0 & \Delta_x \\ 0 & 1 & \Delta_y \\ 0 & 0 & 1 \end{bmatrix}^T$$

Functions which are invariant to these transforms are referred to as S^i, T^i, R^{i}.

4.1 Log-polar Transform with central moments

The Log-polar transform works by converting each *(x,y)* coordinate pair in the original image into polar co-ordinates:

$Polar(x,y) = (r,\theta)$ where

$$r = \sqrt{x^2 + y^2}$$
$$\theta = tan^{-1}(y/x)$$

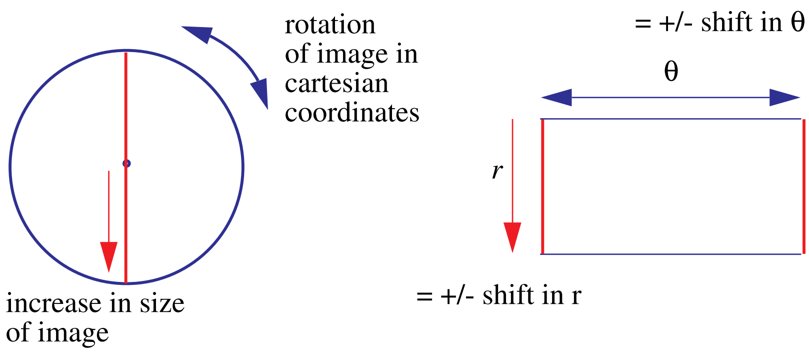

Figure 3: Rotation and scale in xy coordinates are converted into translation in polar coordinates

In polar co-ordinates, rotation in cartesian coordinates results in θ in being shifted by Δθ.

$$R(r,\theta,\Delta\theta) = (r,\theta+\Delta\theta)$$

Similarly, scale changes in cartesian coordinates cause *r* to be shifted by Δ*s* in polar coordinates, converting scale into translation:

$$S(r,\theta,\Delta s) = (\Delta s.r,\theta)$$

Since both rotation and scale cause translation in polar coordinates, a method which cancels translation from a polar representation will remove scale and rotation from the trajectory entirely. Scaling (translation in *r* dimension in polar coordinates) can be addressed by putting *r* onto a logarithmic scale, which forces images which are larger than the template to only have very small differences in their *r* values. This gives the Log-Polar transform:

$$LogPolar(x,y) = (r,\theta) \quad \text{where}$$
$$r = \log\left(\sqrt{x^2 + y^2}\right)$$
$$\theta = tan^{-1}(y/x)$$

Figure 2: log-polar transform of the author

Removing rotational effects (shift in θ in polar coordinates) from the image however is a more difficult problem. Because θ has a cyclical coordinate system (eg. 460 degrees = 100 degrees), shifts in θ can cause an image to be shifted translationally, and if the resulting values of θ exceed π, ie. $\theta+\Delta\theta > \pi$, then points in the image cycle around to appear at the opposite axis $>-\pi$. An example of this effect can be seen in figure 3.

Figure3: Example of the cycling effect found under high relative difference in rotation between one image and another. Both stimuli are identical, however the blue stimulus is rotated 120 degrees from the yellow (a). A rotation invariant transform should show the stimuli to be identical, however the log-polar map of both stimuli shifts the blue stimulus up and past the 2π boundary, with its extremity cycling around and appearing at the bottom of the figure.

Translation invariance for θ can be implemented by centering the data by subtracting the image mean, giving rise to the Centered Logpolar transform:

$$R^i S^i T^i(x,y) = LogPolar(x-\bar{x}, y-\bar{y})$$

$$\text{where } \bar{x} = \frac{1}{n}\sum x \quad \text{and similarly for } \bar{y}$$

Unfortunatley, if cycling occurs however (meaning the data extends over π), averaging will not find the center of the image. In this case other procedures such as the Fourier transform which assume periodicity in the data will provide a better solution.

4.2 Fourier-Melin Transform

In *Théorie Analytique de la Chaleur* (*The Analytical Theory of Heat*, 1822) Joseph Fourier showed that any function could be transformed into a sum of component sin and cos vectors. A Fourier Transform breaks up an image into these constituent sines and cosines.

If a function is given by $g(t)$, then the fourier transform $F(g(t))$ is:

$$F(g) = a_0 + \sum_{n=1}^{\infty} (a_n \cos(n\omega t) + b_n \sin(n\omega t))$$

or equivalently

$$F(g) = \sum_{t=-\infty}^{\infty} g(t) \cdot e^{-2i\pi ft}$$

where $e^{in\omega t} = \cos n\omega t + i \sin n\omega t$ and $e^{-in\omega t} = \cos n\omega t - i \sin n\omega t$ (Euler's formula)

The second equation writes the first as a vector having real and imaginary part, with the real part corresponding to the cosine component, and imaginary part corresponding to the sine component of a particular period. In a two-dimensional image, there will be two frequency dimensions.

Since the real and imaginary parts of the fourier signal correspond to sin and cos, it can be seen that translating the image spatially, ie. $T(g(x),\Delta) = g(x+\Delta)$ causes the fourier transform to advance the cos and sin components by Δ, ie.

$$T(f(g(x)),\Delta) = a_n \cos(n\pi [x+\Delta]) + b_n \sin(n\pi [x+\Delta]),\ 1<n<\infty$$

Right: a) a Fourier transform of a duck. b) an inverse fourier transform with high frequencies subtracted. c) an inverse Fourier transform with low frequencies subtracted. These images were created by Kevin Cowtan. Fourier images use darkness to represent amplitude (dark = high, white = close to zero), and colour (blue to red) to represent phase.

Since sin and cos are both periodic, both are advanced equally, but the ratio between the two is preserved. Another way of capturing this “ratio” is to just take the magnitude of the complex vectors, which is the total length of the vector at this frequency. The magnitude is therefore invariant to spatial translation in the image, and so is often used to achieve translation invariance.

$$T^i(x) = |F(x)|$$

A scale change applied to an image corresponds to multiplying all frequencies by a function of the scale change:

$$\mathrm{S}(F(x,y),\Delta a,\Delta b) = \frac{1}{|\Delta a\Delta b|}F\left(\frac{x}{\Delta a},\frac{y}{\Delta b}\right)$$

Therefore, scale can be eliminated by normalising by the first harmonic (which contains the $\frac{1}{|ab|}$ term), and then putting the frequencies produced by the fourier transform onto a log scale, or simply by putting frequencies onto a log scale without the first step. (Cox, 1992). This gives a scale and translation invariant representation of the original image:

$$S^iT^i(x,y) = \log\left(\frac{|F[(x,y)]|}{|F[(0,0)]|}\right)$$

Rotation can be eliminated by first applying a log-polar transform to the image, which itself transforms rotation into translation. Fourier magnitude then creates a translation invariant signal. This complete transform, is called the Circular-Fourier, Radial-Melin, or the Fourier-Melin transform.

$$FourierMelin(x,y) = |F[Polar(x,y)]|$$

Figure 5: Fourier power spectra for template (left) and hand-drawn stimuli (right). Both images show absolute size of sin and cos components at various frequencies.

4.3 Hu invariants

A moment is a description of some features of a particular distribution, and therefore can be used to compare one trajectory distribution against another. A moment M(p,q,(x,y) of a distribution (x,y) of order p in dimension x and q in dimension y is defined as follows:

$$M(p, q, (x, y)) = \sum x^p y^q$$

Central moments U(p,q,(x,y)) are identical, but each value of x and y has the mean subtracted,

$$U(p, q, (x, y)) = \sum (x - \bar{x})^p (y - \bar{y})^q$$

Normalised moments V(p,q,(x,y)) take central moments and then normalised for size by dividing the moment score by the total area of the shape.

$$V(p, q, (x, y)) = \frac{U(p, q, (x, y))}{\sqrt{M(0, 0, (x, y))}^{(p+q+2)}}$$

Moments have important statistical and physics analogues. The first moment is the mean E[g], the second central moment is the variance $E[(g-\bar{g})^2]$, the third central moment is skew, the fourth central moment is kurtosis. Each of these moments measures particular aspects of the shape, for instance, variance tells us spread, skew tells us if there is a tail (if one end of the shape has less area than the other), kurtosis gives us information on the size of the tail and so on. In physics moments are .

Hu (1962) identified a series of famous shape moments which were invariant to rotation, scale and translation. There were 10 moments in all, although using normalized moments fixed three of these to constant values. This leaves us with six moments which are invariant to reflection, and a seventh which changes sign under reflection. They are defined below:

$$h1 = V(2,0) + V(0,2)$$

$$h2 = [V(2,0) - V(0,2)]^2 + 4[V(1,1)]^2$$

$$h3 = [V(3,0) - 3V(1,2)]^2 + [V(0,3) - 3V(2,1)]^2$$

$$h4 = [V(3,0) + V(1,2)]^2 + [V(0,3) - V(2,1)]^2$$

$$h5 = [V(3,0) - 3V(1,2)]\,[V(3,0) + V(1,2)]\,([V(3,0) + V(1,2)]^2 - 3[V(0,3) + V(2,1)]^2) + [3V(2,1) - V(0,3)]\,[V(0,3)+V(2,1)]\,(3[V(3,0) + V(1,2)]^2 - [V(0,3) + V(2,1)]^2)$$

$$h6 = [V(2,0) - V(0,2)]\,([V(3,0) + V(1,2)]^2 - [V(0,3) + V(2,1)]2) + 4V(1,1)\,[V(3,0) + V(1,2)]\,[V(0,3) + V(2,1)]$$

$$h7 = [3V(2,1) - V(0,3)]\,[V(3,0) + V(1,2)]\,([V(3,0) + V(1,2)]^2 - 3[V(0,3) + V(2,1)]^2) + [3V(1,2) - V(3,0)]\,[V(0,3) + V(2,1)]\,(3[V(3,0) + V(1,2)]^2 - [V(0,3) + V(2,1)]^2)$$

These variables have a very large dynamic range, and therefore for experiments, a set of compressing transformations proposed by Masters (1994) were applied to these moments.

2.6 Eigenvector Rotation

Centering and normalising data has traditionally been done by subtracting the mean and dividing by the norm of the data - as for instance is done in normalized moments. (although even this procedure is susceptible to problems and may not be accurate for all problems[1]). These methods address translation and scale, but do not address rotation. One strategy to achieve rotation invariance has been to use the principal axis of a shape or trajectory to infer the "standard orientation" of that shape or data, and then rotate the shape based on its principal axis into this standard orientation for pattern matching. The procedure is as follows.

First the eigenvectors of the stimulus are found. This can be done by performing a singular value decomposition on the covariance matrix of the data g.

$$eig(g) = diag(S) \text{ where}$$

$Cov(g,g) = USV^T$ where U and V are orthogonal and S is a diagonal matrix of eigenvalues.

Next, the eigenvector with the maximum eigenvalue, the first column of U, is taken as the principal axis, and is rotated $-\theta$ degrees into a horizontal position. Since rotation is symmetric, this can be done by rotating by the transpose of the rotation matrix θ degrees.

$$[r,\theta] = \mathrm{polar}(U_{1..n,1}{}^{T})$$

$$(x, y) = R(x, y, -\theta) = (x, y) \cdot \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

This aligns the data such that its direction of maximum spread is along the horizontal of the coordinate frame being used. Note in this orientation, $M(1,1) = 0$. Next, if the data is not skewed in a standard direction - generally to the right - it is flipped across the axis of the second eigenvector. This can be done by inspecting the third moment of the data, and if negative, flipping the data across the left-right axis by multiplying by the x values of data -1.

$$\text{if } M(3,0) < 0, \quad (x, y) = (x, y) \cdot \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

The result of this process is the original data, rotated about its axis until it is in a standard orientation with its largest direction of spread/mass on the horizontal, and the direction of next maximum mass orientated in the left, with its tail pointing to the right. If the candidate stimulus to be compared preserves these basic properties of variance and skewness,

1.

then they will both be orientated into exactly the same position for pattern matching.

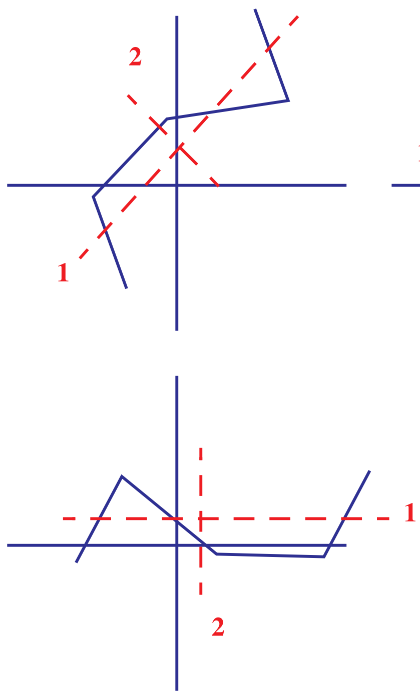

Figure 4: Eigenvector rotation method finds the principal axis of a trajectory or shape, and then rotates the trajectory until that principal axis is horizontal with resspect to the basic coordinate system taken. If there is more trajectory mass on the right (calculated by finding the third moment of the trajectory in this orientation), the trajectory vector is rotated a full 180 degrees again so that the smaller area component of the image is pointing towards the right.

A problem with this method is that for a target trajectory the ratio of the largest eigenvalue to the smallest might be very close to 1. This is referred to in the literature as *an ill-conditioned problem*. In this situation, if the response trajectory deviates just slightly from the target, the first and second eigenvalues could be swapped, resulting in a completely orthogonal axis of maximum variance, and so a stimulus may be rotated into a standard position almost 90 degrees from the stimulus it is being compared against.

This problem was addressed by testing the match between the two stimuli at each of the three additional 90 degree rotations from the standard position. Unfortunately, even using this technique, the correlation was quite low - lower in fact, than just normalising the trajectories in their original orientations and calculating a cross-correlation. This poor result indicates that the act of finding the direction of maximum spread and rotating this into position itself is not a good method for orientating stimuli in this domain. A reason for this is that noisy deviations in trajectory, and especially beginning and end transients (entire line segments added by the user in finding an initial point, or trailing after an experiment) significantly distorted the variance of the data, leading to the maximum variance being in a new direction, however, with the best direction of correlation was still being in the standard direction. Another factor contributing to this was the fact that stimuli were generally not rotated by any large degree in this domain.

Because large amounts of noise was common in this application, this method suffered greatly in terms of inappropriate rotation and correspondingly poor matches. Other methods such as fourier-melin degrade only in proportion to the noise in the figure, and therefore were better suited to this particular problem.

2.5 Canonical Analysis

Canonical correlation is a multidimensional extension of the ordinary 1-dimensional vector correlation. Canonical correlation can be defined in a number of interesting ways. The canonical correlation between two multi-dimensional vectors *f* and *g*, is that it is the maximum ordinary 1 dimensional correlation between a linear combination of *f* and a linear combination of *g*.

$$f_{comb} = a_1 f_1 + a_2 f_2 + \ldots + a_d$$

$$g_{comb} = b_1 g_1 + b_2 g_2 + \ldots + b_d$$

$$CanCorr = \max_{(a_1,a_2,..,a_d),(b_1,b_2,...,b_d)} Corr(f_{comb}, g_{comb})$$

where

$$Corr(f_{comb}, g_{comb}) = \frac{\left(f_{comb} - \bar{f}_{comb}\right) \cdot \left(g_{comb} - \bar{g}_{comb}\right)}{\left|f_{comb} - \bar{f}_{comb}\right| \cdot \left|g_{comb} - \bar{g}_{comb}\right|}$$

The maximum correlation between the two linear combinations is said to be the first correlate, and this can be used as the "correlation". It is also possible to then subtract out f_{comb} and g_{comb} from the *f* and *g* data, and then to run the canonical procedure again (finding another f_{comb} and g_{comb} which maximise the standard correlation). If there are *d* dimensions in the data, then there are *d* such correlations.

Algebraically Canonical Correlation can also be defined as

$$A = Cov(f,f)^{-1/2}\, Cov(f,g)\, Cov(g,g)^{-1/2} \quad (1)$$

$$c = eig(A) \tag{2}$$

where $Cov(f,f)^{-1/2}$ refers to the inverse of the matrix square root of the covariance matrix of *f* with *f*, and *Cov* is defined as:

$$Cov(f, g) = \frac{1}{N}(f - \bar{f}) \cdot (g - \bar{g})^T$$

(Eric Bohlman and Roger Laffose, 1997; Bernstein, 1988). The equation for *A* is a natural extension of the ordinary formula for correlation between 1 dimension vectors, which is essentially the covariance of each vector divided by the norm of that covariance. *c* is a positive vector of eigenvalues between 0 and 1.

Figure 8: The Canonical correlation of two vectors *f* and *g* is the maximum correlation between a projection from *f* onto one coordinate axes (a line through the space), and a similar projection from *g*. There are *d* axes, and so there are *d* possible correlations. The eigenvalues for this example are .9968 (eigenvector [-.9091 .4166]) and .9831 (eigenvector [-.4166 -.9091]), which show a strong relationship in both principal axes.

The first procedure for calculating canonical correlation is reminicent of the procedure used for calculating principle components. In fact, canonical correlation is a multidimensional extension of principal components. The Principal components of a vector *f* are the eigenvalues of the variance matrix between *f* and *f*, that is, how each dimension of the vector varies with every other dimension of itself. Canonical correlates on the other hand, are the eigenvalues of the covariance matrix between *f* and some other vector *g* normalised by the size of the vector. This can be demonstrated as follows:

Principal components are defined as the eigenvalues of the Covariance matrix of *f* with itself. Therefore, $PC(f) = eig(Cov(f,f))$. From the definition of Canonical Correlation, we now subsitute *f=f* and *g=f* into the canonical correlation equation:

$$CanCorr(f,f) = eig(A)$$

where

$$A = Cov(f,f)^{-1/2}\ Cov(f,f)\ Cov(f,f)^{-1/2}$$

Therefore, the canonical auto-correlation with *f* with *f* is precisely the principal components of *f* but with the small change in that it is normed by the magnitude of the covariance matrix.

The main usefulness of this procedure is that it gives the algebraic correlation between a pair of multidimensional target and response trajectories. This correlation also happens to be invariant to rotation, translation, and scale. This is proved below.

Theorem: Canonical Correlation is invariant to rotation, scale and translation.

Proof of rotation invariance
Let R be the rotation matrix defined in (4).

$$A\ (f, fR)\ =\ \left(\frac{f^T f}{N}\right)^{-\frac{1}{2}} \cdot \left(\frac{f^T fR}{N}\right) \cdot \left(\frac{(fR)^T fR}{N}\right)^{-\frac{1}{2}}$$

$$= \frac{1}{N}\left(f^T f\right)^{-\frac{1}{2}} \cdot \frac{1}{N}\left(f^T fR\right) \cdot \frac{1}{N}\left(R^T f^T fR\right)^{-\frac{1}{2}} \text{ distributing transpose}$$

$$= \frac{1}{N^3}\left(f^T f\right)^{-\frac{1}{2}} \cdot \left(f^T fR\right) \cdot \left(R^T f^T fRR^T f^T fR\right)^{-1} \text{ inverse of the square root}$$

$$= \frac{1}{N^3}\left(f^T f\right)^{-\frac{1}{2}} \cdot \left(f^T fR\right) \cdot \left(R^T \left(f^T f\right)^2 R\right)^{-1} \text{because } RR^T{=}I$$

$$= \frac{1}{N^3}\left(f^T f\right)^{-\frac{1}{2}} \cdot f^T fR \cdot R^{-1}\left(f^T f\right)^{-\frac{1}{2}} R \quad \text{distributing inverse and } (R^T)^{-1} = R$$

$$= \frac{1}{N^3}\left(f^T f\right)^{-\frac{1}{2}} \cdot f^T f \cdot \left(f^T f\right)^{-\frac{1}{2}} R$$

$$= Cov\,(f,f)^{-\frac{1}{2}} \cdot Cov\,(f,f) \cdot Cov\,(f,f)^{-\frac{1}{2}} \cdot R$$

$$= A\,(f,f) \cdot R$$

Because eigenvalues are unchanged under rotation, *eig[A(f,fR)]=eig[A(f,f)]* the canonical correlation between two rotated stimuli are identical regardless of rotation.

Proof of scale invariance

Let g be equal to f under a scale transformation, $g = Sf$. Then

$$A\,(f, Sf) = \left(\frac{f^T f}{N}\right)^{-\frac{1}{2}} \cdot \left(\frac{f^T Sf}{N}\right) \cdot \left(\frac{Sf^T Sf}{N}\right)^{-\frac{1}{2}} \quad \text{(from 1)}$$

$$= \frac{\sqrt{S^2}}{S} Cov\,(f,f)\, Cov\,(f,f)\, Cov\,(f,f)$$

$$= A(f,f)$$

Since $A(f,f) = A(f,Sf)$, the eigenvalues of $A(f,Sf)$ are unchanged.

Proof of translation invariance

Since the mean is subtracted from both f and f (in the calculation of *Cov*), then *Can-Corr(f,g)* must be invariant to translation.

Similarly it can also be shown that Canonical Correlation is invariant to reflection, but not invariant to shear.

2.6 Casesant fourier

Several variants have been suggested on the basic Fourier-Melin procedure implemented here. A good survey can be found in Wood (1996). One such procedure is a method introduced by Casasent and Psaltis (1976) as an alternative to the standard Fourier-Melin procedure. The primary difference is that instead of using moments to center the image, (ie. just subtracting the image mean), a Fourier power spectrum is calculated (which is translation invariant). The algorithm as follows: (1) calculate fourier transform of image and take magnitude. This takes care of translation in the image. (2) perform log-polar transform on the power spectra. This converts rotation to translation and nullifies scaling effects. (3). Perform a second fourier transform and take magnitude. This nullifies rotation in the image.

2.7 Aspect ratio

Aspect ratio gives the ratio between the maximum and minimum eigenvalues, and so is invariant to rotation, translation, and scale in the data. However, this provides only very general information about a signal. This method is included in this paper only for comparison, and because it can assist in the combining stage discussed later.

2.8 Other methods

Perantonis and Lisboa (1992) have also introduced the *Zernike moment*, which multiplies the image by Zernike polynomials which are expressed in polar coordinates. This provides rotation invariance. Some authors have also presented interesting theories on how affine transformation invariance could be achieved. Assuming the general form of the invariant function is known, it becomes a matter of finding parameters which minimise in a least squares sense, the invariance function. Further details can be found in Flusser and Suk (1993) and Rothe et. al. (1994). A comprehensive survey of invariance methods can be found in

3.0 Matching

After transforming the trajectory to make it invariant to rotation, translation and scaling, a match needs to be calculated between the target and response trajectories. This matching was done differently depending on the method used. Hu invariants were matched by calculating the mean squared difference between the target and resposne. Canonical correlation was calculated a match implicitly. All of the other methods used a novel application of cross-correlation typically applied in image processing. This is described in this section.

Let $Image_f(i,j) \quad = 1, \text{ if } (i,j) \in f$

$=0$, otherwise

This is a function similar to an image function in image processing, which gives a 1 in every location where there is a pixel, and a 0 everywhere else. We will use this function to compute a grid-wise cross-correlation between two trajectories.

$$CrossCorr(f, g) = \frac{\sum_{i,j} u_f(i,j) \cdot u_g(i,j)}{\sqrt{\sum_{i,j} u_f(i,j)^2 \cdot \sum_{i,j} u_g(i,j)^2}}$$

where $u_f(i,j) = Image_f(i,j) - \frac{1}{n}\sum_{i,j} Image_f(i,j)$ and similarly for u_g.

.

Figure 2: Surface plot of the convolved template (on the left) and image (on the right). Smoothing is necessary to ensure a good range of matches.

Cross-correlation was performed in a number of stages (figure 5). Vectors were first projected onto an $IM \times IM$ size 2D mesh. (for 3D trajectories, this would be a discretized volume). This was done by first calculating the smallest bounding rectangle which would enclose a trajectory *f*, by subtracting the minimum *x* and *y* value from the maximum value in *f*.

$$range(f) = max(f)\text{-}min(f)$$

Next, the trajectory vectors were normalized to a range of *[0..IM]* by dividing all *(x,y)* entries in *f* by *range(f)*, to normalize their value to the range 0..1, and then multiplying by *IM*. Finally, if there was a mismatch between the size of *range(f)* and *range(g)*, the smaller was rescaled and padded with zeros, by dividing by *range(larger)/range(smaller)*. The trajectories were then converted into a grid representation by applying the $Image_f$ function, giving rise to a discretized, mesh version of the original trajectory.

$$f_{norm} = \frac{(f - min(f))}{range(f)} \cdot IM$$

$$f_{norm} = f_{norm} \cdot \frac{range(g)}{range(f)} \quad \text{, if } range(f) < range(g) \text{ and}$$

$$g_{norm} = g_{norm} \cdot \frac{range(f)}{range(g)} \quad \text{, otherwise}$$

The final step was to convolve the grid representation with a gaussian kernel *G* of width *w*, height *gsize*, centered at 0, to give a blurred and spread out version of the polygonal path, which allowed spreading out into this mesh. This was necessary because lines were only one grid-size wide, and so spreading improves the range of matches and the information on trajectories which are close but not identical to the original. The convolved mesh representations of target and response paths were then matched by the normalized cross-correlation algorithm. Cross correlation was used for calculating Log-Polar with centering, Fourier-Melin, eigenvector rotation, and normalised cartesian correspondances.

$$c = CrossCorr(Image_f(i,j)*G,\ Imageg(i,j)*G) \text{ where } i,j \in [1..IM] \text{ and}$$

$$G(j,i) = \frac{1}{w \cdot \sqrt{2\pi}} \cdot e^{\frac{-\left(\sqrt{(j-gsize)^2 + (i-gsize)^2}\right)^2}{(2 \cdot w^2)}}$$

Figure 1: Stages in calculating an image cross correlation. From left to right, top, (a) the original image is read in and sampled at low resolution, giving this discrete dot pattern. (b) the image is further decimated by x10 to create a low resolution representation of the image. (c) a Gaussian kernel used for smoothing (d) The low-resolution image after convolution, (e) a surface plot of the convolved image. (f) contour plot of the same image.

Although the mesh-based spreading step may seem like a crude form of getting information on trajectory vector correspondances, in practice this was found to be superior to other forms of matching experimented with. One alternative was was to find the minimum squared distance between each point in the target and points in the response. A problem with this method of matching was that it did not register "negative" matches, ie. features of a trajectory which appeared in the target and were not replicated in the response were not penalised for not being present in the way that cross-correlation penalised them.

4.0 Empirical Results

The ultimate validation for the scoring system was for its scores to be checked against human scores on aparticular test set. The pattern analysis system was used to analyse over 600 template response trajectories, and assign scores to each. Following this, human subjects were then asked to rate the same 600 by viewing the trajectories. N=3 subjects were used to give ratings.

Next a correlation was run between the match scores the program produced, and the mean human subject scores (p). These results are shown in table 1.

The correlation between human scores themselves was only 0.5. This was a very low correlation, and meant that human assessments of similarity were highly variable. It further meant that if a machine was to be comparable to human performance, it was unlikely any procedure could attain a score much higher than 0.5.

4.1 Results for Standard Methods

Fourier-Melin was the most successful technique of the eight, with Hu invariants the worst. In addition to the methods mentioned in Section 2, a domain specific match method was also introduced for final matching, which looked for features in the image and matched on that basis ("Feature-based"). Interestingly, this method perfomed well below other general methods.

Invariance method	Match method	*p*	SSE
Fourier-Melin	Normalized Cross-correlation	.4585	77.17
Cartesian centered and scaled	Normalized Cross-correlation	.4511	80.58
Smoothed Polygonal boundary	Vector Correlation	.4258	76.68
Feature-based	Vector Correlation	.3947	80.72
Polygonal boundary	Vector Correlation	.3840	80.01
Canonical Correlation	Canonical Correlation	.3194	115.60
Log-polar moment centering	Normalized Cross-correlation	.2427	109.47
Casesant fourier	Normalized Cross-correlation	.1509	136.64
Cartesian trajectory vector	Normalized Cross-correlation	0.4207	71.06
4-way Eigenvector rotation	Normalized Cross-correlation	0.3580	91.71
Aspect ratio	Squared difference	0.1688	137.08
Eigenvector rotation	Normalized Cross-correlation	0.1920	135.53

Hu invariants	Squared difference	0.0702	151.81

As discussed in earlier sections, eigenvector rotation is not well suited to this application,

Invariance method	Match method	*p*	SSE
Normalised xy	Cross correlation	0.4761	3665
Centered xy	Cross correlation	0.4404	3668
Smoothed polygonal boundary	Vector correlation	0.3650	3687
Fourier-melin	Cross correlation	0.3647	3725
4-way Eigenvector rotation	Cross correlation	0.3580	3666
Polygonal boundary	Vector correlation	0.3486	3690
Canonical Correlation		0.3402	3633
Feature correlation	Vector correlation	0.3072	3663
Fourier polygonal	Vector correlation	0.2998	3660
Log-polar centered	Cross correlation	0.2738	3678
Eigenvector rotation	Cross correlation	0.1920	3681
Aspect ratio	Sum squared difference	0.1688	8542
Casesant fourier	Cross correlation	0.1509	3679
Hu moments	Sum squared difference	0.0796	15548

Invariance method	Match method	*p*	SSE
Canonical Correlation		0.51	14634
Feature correlation	Vector correlation	0.48	14729
Fourier-Melin	Cross correlation	0.46	14981
Normalized xy	Cross correlation	0.41	14746
Polygonal boundary	Vector correlation	0.41	14851
Smoothed polygonal boundary	Vector correlation	0.40	14835

Fourier polygonal boundary	Vector correlation	0.33	14713
centered xy	Cross correlation	0.32	14761
4-way Eigenvector rotation	Cross correlation	0.30	14747
Eigenvector rotation	Cross correlation	0.25	14777
Casesant fourier	Cross correlation	0.24	14809
Hu invariants	sum of squared differences	0.22	61131
Log-polar centered	Cross correlation	0.18	14807
Aspect ratio	sum of squared differences	0.05	18127

because of the potential for large rotations under small amounts of noise. The poor performance of Hu invariants was more of a surprise however. This also deviates from results by Cohen (1994), where moments were used to identify 3D parts for a Computer Aided Design application. Cohen showed that in a correlation matrix, four major groups of objects (bolts, washers,) tended to have similar moments (grouped together by their correlation), and therefore, similarity seemed to be preserved. He also noted, however, that some salient aspects of shapes, for instance, the difference between a round-headed and square hexagonal bolt tended to be overlooked by moment-based methods, since these were much finer features of the object. A reason Hu invariants may have given poor performance in this application, was because there were a large number of different trajectories, which had an overall similar form. Fine features of the trajectories which were essential for discrimination, were not picked up. This suggests that higher-order moments might have been more successful.

4.2 Invariance?

Although many of the invariance methods had strong theoretical motivations, in practice their performance was certainly less than ideal. Figure 4,5,6 shows various methods compared on invariant benchmark data. It can be clearly seen that far from being invariant, the scores given by transforms certainly do change with rotations. The reason for this failure to implement invariance depends on the specific method in question. Centered Log-polar is susceptable to a cycling problem which makes estimation of the center and hence correct centering and alignment for matching, impossible. Eigenvector routines suffer from problems with noise which can cause spurious rotations between target and response. All methods which rely on estimation of the center, and normalization of the size of the image (such as Hu invariants, Cartesian normalization, Log-polar), are susceptable to problems. Normalization is susceptible to outliers. If a data cloud has its main density in a tiny region, and then has a single noisy point several standard deviations away from this dense region, then the entire distribution is normalized based on the “size” of the outlying point. This has effect of scaling the dense region to a tiny fraction of the whole. This problem is commonly experienced in image processing where normalization of image brightness can be undone by the presence of a few bright spots in the image which cause the main part of the image to appear dark. Centering by finding the mean can also provide a poor center for a datacloud if that datacloud is skewed. In this case, the true origin of rotation of the data may not be given by the mean.

Finally, The Fourier-Melin procedure, although one of the most successful transforms, may have suffered from boundary effects arising because the trajectory signal is not periodic. The influence of these many confounds is demonstrated clearly by the demonstration that all of these methods are poor at estimating invariance. From an estimation point of view, the differences in both degree of invariance, transform apparatus, scores, correlation of scores with human scores, also is suggestive that each procedure has a different *bias* associated with it.

4.3 Bias in Estimating Invariance

Bias just means "difference in accuracy between some estimator and a target distribution", and is normally expressed as the (E(Ti)-E(Tj))2. The assertion that two transforms have different biases can be proved by showing that an order relation over neighbouring images in T_i is not preserved in T_j. Something of the degree of distortion can be inferred by showing that two transforms are not homotopic ie. smoothly deformable, with each other.

Theorem 1: Neighbouring stimulus violation: An order relation on T_i may be violated in T_j if the difference between the two grows according to a non-linear function, or a linear function of the form $T_i\text{-}T_j=a(T_i)+c,\ c>0.$

Proof

If $T_i\text{-}T_j=a(T_i)+c \sim 0,$ *or* $T_i\text{-}T_j=a(T_j)+c \sim 0$ where $c>0$; where a is chosen to minimize the equation, (ie. T_i is a scalar multiple of T_j) then since < is invariant to multiplication (ie. $x<y \leftrightarrow lx<ly$) then any order relation between images $T_i(g_1),T_i(g_2),...,T_i(g_n)$ must also be preserved in $T_j(g_1),T_j(g_2),...,T_j(g_n)$. Therefore, if two transforms are scalar multiples, then they are guaranteed to have no effect on the order relation.

In the other case, (for instance, if $T_i\text{-}T_j=a(T_i)+c >> 0)$ the order relation, < cannot be guaranteed to be preserved. Therefore, "neighbouring trajectories" under T_i may not be neighbouring in T_j, and so the similarity space is not preserved between the two transforms.

Corollary: Invariance methods which are not linear with respect to each other. Canonical analysis, Fourier-Melin, and are all functions which are not linear functions of each other, and therefore can be expected to have different order relations on stimuli.

Definition: Homotopy between two transforms

A mapping f^t:$X \rightarrow Y$ is homotopic if for each X and Y they may be joined by a smoothly evolving family of maps f^t:$X \rightarrow Y$, and and if the inverse family of maps, f^{-1}:$Y \rightarrow X$ is also defined (diffeomorphism). In other words, on the unit interval $[0,1]$ in **R**, we say X and Y are homotopic if there exists a smooth map f:$X \times I \rightarrow Y$ such that $f(X,0) = X$ and $f(X,1) = Y$.

Lemma 3: Transforms which use **log**, and those which do not are non-homotopic. Transforms which use |.| and those which do not are non-homotopic with each other.

Proof

|.| is not invertable, and therefore, violates the inverse property of homotopy. With regard to log, $\lim_{t \rightarrow \infty} x \neq \lim_{t \rightarrow \infty} \log(x)$, therefore, this also breaks the space.

Corollary: Invariance methods which are not homotopic with each other

Fourier-Melin, Log-Polar, Casesant use log, and therefore are non-homotopic with the other transforms outlined in this paper. Fourier-Melin and Casesant both use magnitude and therefore are non-homotopic with the other transforms discussed in this paper.

4.4 Empirical evidence of bias: Distortion results

Further evidence of bias can be found by analysing the scoring performance of the procedures on a set of controlled stimuli. A group of 10 random trajectories were chosen as prototypes, and the following algorithm was used to add bivariate gaussian noise to the stimuli in increasingly larger amounts. This noise affected the angles and lengths of the prototype trajectories.

algorithm

This provided a family of trajectories with increasing amounts of distortion from the original prototype. Next, the scoring system calculated a score for the similarity between the prototype and the family of distorted trajectories based on that original prototype. There were 10 levels of randomness, leading to very high levels of distortion. A series of distorted trajectories are provided in figure 5.

A side-effect of adding bivariate random noise was that each time noise was added, it increased the overall length of the trajectory. At the highest level of randomness, the trajectory had increased from 200 pixels in size, to over 2000. It was important to only have

stimuli influenced by randomness, to get an idea of how matching degraded under randomness, and so how the similarity space was shaped for different methods.Size increases were controlled for by normalizing the trajectory by its size
f/max(f)

Scores for increasingly degraded stimuli are shown in figures 6 and 7. It can be shown that Hu invariants ... do not degrade smoothly on these stimuili, where-as fourier melin does.... It will be shown in the final section that combined methods such as stacking and averaging significantly eliminate this problem, giving substantially smoother similarity comparisons.

6.0 Combining Estimators

This above results show that many of the transforms listed in this paper are biased relative to each other. This still leaves the problem of maximising the accuracy of the similarity estimation. One simple approach to maximise estimation accuracy is simply to choose the best performing invariance method. However, this ignores a lot of information which is provided implicitly in the other methods' scores. A second approach is to take some average of the scores produced by each of the estimators.

The latter approach has recently been substantiated in some analytical work by Tumer and Ghosh (1993). Under assumptions that the variance of each of the estimators was roughly the same, these authors found that the error of the boundary region, E_{add} between the estimated distribution and the true distribution, for averaged classifiers is equal to

$$E_{add}^{ave} = \frac{1 + p\,(N-1)}{N} Eadd$$

where N is the number of estimators, E_{add} is the constant quantifying the base error for each of the estimators, and p is the correlation among the estimators. This indicates that designers should look for as many good uncorrelated estimators as possible. Interestingly, it is impossible to just continue to find good uncorrelated estimators. Each estimator which is applied to the data can be thought of as shattering the data space in some dimension. There are only a finite number of possible ways to shatter the data, and good experts will comprise an even smaller set. As a result, its likely to see a phenomenon where increasing numbers of colinear experts are found, increasing the top part of the equation and decreasing the total improvement.

Rescaling for combining

A problem with combining raw scores from estimators, was the fact that estimators tended to have a different range of numbers in which scores typically fell. For instance, low scores for canonical correlation tended to be rare, since it almost always found some axis through the data which gave a reasonably good correlation. Canonical Correlation tended to give scores in the range 0.7..1.0. Log-polar tended to give scores in the range 0 to 0.7. Sum of squares matches gave numbers in the range from 0 to some large number. Simply averaging these raw scores, therefore, would have been meaningless. Therefore, prior to combining, all estimator scores were normalized to Z-values which gave them unit variance, and centered them around 0. (This also put different dimensions on the same scale, and so made the least squares optimization problem used in stacking more soluable)

$$p = \frac{(p - \bar{p})}{\sqrt{\frac{1}{N-1}\sum (p - \bar{p})}}$$

4.2 Combining estimators by averaging

Averaging a group of estimators gives an ensemble classifier

$$P = \frac{1}{N}\sum_{i=1}^{N} p_i:$$

where there are a total of N estimators p_i. A problem with averaging is the fact that many of the estimators may be colinear with each other. In this case, the average found can become dominated by one estimator, and the benefits of using multiple estimators can be lost. One way to address this is to use a training set to find a set of weights which can be applied to each estimator in the average to minimize the error.

4.3 Combining estimators by stacking

Stacking was proposed by Wolpert (1992) as a method for improving the performance of a single estimator, by dividing its training set into different partitions and then combining the results of each. However, it quickly became apparent that stacking could be applied to any set of estimators, and indeed works best when the estimators are quite different from each other.

Where-as averaging gives equal weight to each estimator, Stacking can increase or decrease preference on particular experts, depending upon their performance on a training set. Stacking does this by essentially running a regression to find a set of coefficients which optimally map the set of estimator values (x,p(x)), to the a set of known correct training data (x,f(x)). This is normally a linear regression, and although high-order regressions are possible, this is not used in the literature, because high-order models increase the danger of overfitting based on a small training set. The result of this technique is that a set of coefficients are found which weigh each estimator, and are then used there-after to weigh their advice. Stacking is defined as:

$$P = \sum_{i=1}^{N} w_i p_i$$

Briemann (1992) suggested a way of improving the basic stacking method. Breimann observed that standard stacking seemed to overshoot and undershoot the range of training data y, by a considerable amount, giving wildly off results. Instead, Bremann proposed constraining the regression coefficients to the range [0..1] to force the combined estimator to stay within the maximum and minimum range indicated by the training data. Constrained stacking is defined as:

$$P = \sum_{i=1}^{N} w_i p_i$$

$$\text{where} \quad \sum_{i} w_i = 1$$

Combination method	p	SSE
Constrained stacking	.6062	50.06
Average	.6052	51.15
Stacking	.5374	55.32
Best classifier	.4585	77.17
Worst classifier	.2427	109.47

Table 2: Combination methods outperformed all standard methods. Results reported are for test set only.

The test set results from combination methods are shown in table 2. All methods were very effective, with constrained regression and averaging performing particularly well. Constrained Stacking is extremely robust, as even colinear, redundant, or negatively correlated experts - which would be factored into averaging - simply have their weights set to zero, and other experts will take up proportionately greater responsibility. As a consequence, constrained stacking provides a simple, robust, and effective method for boosting estimation.

One of the objectives in combining estimators was to eliminate particular biases, and get closer to an overall score for similarity. In addition to showing higher correlations with human ratings, performance on distortion data, which was a probe of how smoothly ratings degraded with randomly degrading stimuli, show that the data clearly degrade on a very smoothly sloping logarithmic curve.. This contrasts with results for individual estimators alone, which showed more variable results. As a result, it should be inferred that the bias problem has been reduced.

** either this or: Distortion results section showing that stacking and average give smoothly decaying similarity results (which was the intent)

Figure 10: In this diagram human subject scores have been sorted into values from smallest to largest. These form the yellow line in the upper part of each of these graphs. The blue irregular line is

the performance of a particular expert or combination. If the expert was perfect, it would perfectly follow the yellow line. The lower part of each figure shows the squared error on each of these sorted test cases. Smaller bar graphs mean better overall performance. The graphs are from top left to bottom right, (a) worst expert, (b) average of experts, (c) basic stacking, (d) constrained stacking. All cases shown here are from the test set.

7.0 Conclusion

This paper has demonstrated that combining methods are extremely effective on difficult pattern recognition problems. In fact, the situation is almost win-win. Unlike averaging, and assuming a representative training set, stacking can only improve performance, and even redundant, colinear or malicious experts can be taken out of the data by the procedure. After running a match, and additional methods can be given to the procedure to try to improve the performance.

References

Casasent, D. and Psaltis, D. (1976), Position, Rotation and Scale-invariant Optimacl Correlation, *Applied Optics*, Vol. 15, pp. 1795-1799.

Cowan, K. (1996), *Kevin Cowtan's Book of Fourier*, http://www.yorvic.york.ac.uk/~cowtan/fourier/fourier.html)

Cox, G. (1995), Template Matching and Measures of Match in Image Processing, Department of Electrical Engineering, University of Cape Town.

Cox, G. and de Jager, G. (1993), Invariance in Template Matching, *Proceedings of the Fourth South Affrican Workshop on Patter Recognition*, pp. 152-6. http://www.dip.ee.uct.ac.za/imageproc/pattern/

Masters, T. (1994),

Mokhtarian, F. and Mackworth, A. (1986), Scale-based description and recognition of planar curves and two-dimensional shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 1. pp. 34-43.

Perantonis, S. and Lisboa, P. (1992), Translation, rotation and scale invariant pattern recognition by high-order neural networks and moment classifiers, *IEEE Transactions on Neural Networks*, Vol. 3, pp. 241-251.

Rothe, I., Voss, K., et. al. (1994), A General Method to Determine Invariants, Fakultat fur Mathematik and Informatik, Friedrich-Schiller-Universitat Jena, Germany. http://www.sicc.co.kr/~yalkongs/archives/