Continuous *n*-tuple classifier and its application to real-time face recognition

S.M.Lucas

Indexing terms: Real-time face recognition, N-tuple classifiers, Pattern recognition

Abstract: The continuous *n*-tuple classifier was recently proposed by the author as a new type of *n*-tuple classifier that is ideally suited to problems where the input is continuous or multi-level rather than binary. Results on a widely used face database show the continuous *n*-tuple classifier to be as accurate as any method reported in the literature, while having the advantages of speed and simplicity over other methods. The author summarises the previous work, provides new insight into how the system works and discusses its applicability to real-time face recognition.

1 Introduction

N-tuple classifiers have established a reputation as being the fastest possible classifiers for a wide variety of pattern recognition problems. The continuous *n*-tuple classifier offers a simple way of applying *n*-tuple classifiers to problems where the input is continuous or multi-level. In such cases, the continuous *n*-tuple classifier can offer superior recognition accuracy over the standard *n*-tuple classifier. The continuous *n*-tuple classifier can be used in two ways; compiled and uncompiled, as explained below. If the compiled version is used, then training is slower than the standard *n*-tuple classifier, but recognition speed is the same. Conversely, in the uncompiled version, training speed is the same as the standard *n*-tuple classifier but recognition is slower.

1.1 Face recognition and identity verification systems

Identity verification can be based on three principles:

- what you have (e.g. a door key or credit card)
- what you know (e.g. a pin number or password)
- what you are (e.g. fingerprint, face or some other biometric)

Systems based on the first two principles involve low (or zero) computational overhead, but can be insecure and expensive to administrate. Biometric systems involve some kind of signal transducer to capture some aspect of the human form or behaviour, which forms the input to a pattern recognition/verification system. For the case of face recognition, only recently has the combination of low-cost video/image capture hardware and low-cost high-performance computing hardware made face recognition a viable alternative.

Compared to other biometrics, face recognition has several distinctive features:

- can be used unobtrusively/covertly, in ways that iris scans or fingerprints cannot
- can potentially be used in conjunction with E-fit (electronic photofit) to search for faces based on a witness description
- a face recognition system could as a byproduct of its normal function store a video stream for later human analysis, if a system was suspected of being invaded
- Many PCs already have a monitor-top camera for use in video-conferencing; this can be used in conjunction with face recognition software to protect access to the PC.

This potential is already starting to be realised, but more work is needed to make systems faster and more accurate in real-world circumstances.

Face recognition can be decomposed into two problems; finding a face (or faces) in an image, and recognising the identity of that face. A related problem is that of facial gesture recognition, where the aim is to detect whether someone is smiling, laughing or frowning, etc.

Finding a face in an image is also known as face registration or face localisation. The degree of difficulty of this problem depends on several factors, such as the control one has over lighting and background conditions, whether the images are colour or monochrome and whether the images are still or on a video stream. In the case of a video stream, for example, motion detection can be used to help identify a face as it moves into the frame, or if the subjects can be made to perform a certain gesture (such as blinking) then this can be used as a location cue. If the image is colour, then the range of possible skin tones for a given illumination condition is a small subset of the set of all possible colours. If the background can be controlled, then it might be possible to extract the head very simply as one of the main sources of brightness in the image. Hence, depending on the control one has over these factors, face localisation can be a hard or an easy problem.

There are two versions of the face recognition problem (this is true for most biometrics); the recognition and the verification problem. In the case of verification, one only has to test the likelihood that the face is that of who it clalms to be, hence this involves testing the quality of match of an image against a single

IEE Proceedings online no. 19982317

Paper first received 18th February and in revised form 20th May 1998

The author is with the Department of Electronic Systems Engineering, University of Essex, Colchester CO4 3SQ, UK

model. In the case of face recognition, the problem is to find the best match of an unknown image against a database of face models, or to decide that it does not match any of them well. The practical importance of this distinction is the speed required; generally, if there are C subjects (i.e. people) in the database, then the recognition process will be C times slower than the verification process. This may place practical limits on the algorithm used.

1.2 N-tuple classifiers

Conventional n-tuple systems [1] have the desirable features of superfast single-pass training, superfast recognition, conceptual simplicity, straightforward hardware and software implementations and accuracy that is often competitive with other more complex, slower methods. Due to their attractive features, n-tuple methods have been the subject of much research. See Rohwer and Morciniec [2] for a thorough review.

In conventional n-tuple-based image recognition systems, the locations specified by each n-tuple are used to identify an address in a lookup table. The contents of this address either use a single bit to indicate whether or not this address was accessed during training, or store a count of how many times that address occurred.

While the traditional n-tuple classifier deals with binary-valued input vectors, methods for using n-tuple systems with integer-valued inputs have also been developed. Allinson and Kolcz [3] have developed a method of mapping scalar attributes into bit strings based on a combination of CMAC and Gray coding methods. This method has the property that for small differences in the arithmetic values of the attributes, the Hamming distance between the bit strings is equal to the arithmetic difference. For larger values of the arithmetic distance, the Hamming distance is guaranteed to be above a certain threshold. However, where practical, the continuous n-tuple method described in this paper should be preferable, since it incorporates the exact arithmetic distance between attributes.

The continuous n-tuple method also shares some similarity at the architectural level with the single-layer look-up perceptron of Tattersall and Johnston [4], though they differ in the way the class outputs are calculated and in the training methods used to configure the contents of the look-up tables (RAMs). The continuous n-tuple method was first proposed by the author [5], and since tested more thoroughly [6]. This paper summarises the previous work, provides fresh insight into how the system works and discusses its applicability to real time face recognition on video streams.

2 The continuous n-tuple classifier

The d-dimensional input space is sampled by m n-tuples. Each n-tuple defines a fixed set of locations in the input space. Let the set of locations defining the jth n-tuple be

$$n_j = \{a_{j1}, a_{j2}, \ldots, a_{jn} | 1 \leq a_{ji} \leq d\} \qquad (1)$$

where each a_{ji} is chosen as a random integer in the specified range.

The continuous n-tuple classifier stores the vector defined by each n-tuple on each image explicitly. Hence, each n-tuple simply defines a sparse subset of the original image vector. This is illustrated in Fig. 1, which shows a continuous n-tuple sampling of a face for the case of $n = 3$ and $m = 3$; note that in practice m has to be much larger than this to get good performance.

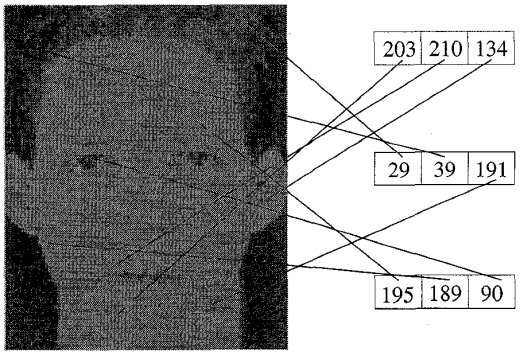

Fig. 1 *Illustration of the continuous n-tuple sampling process*
A set of random n-tuples are chosen, where each n-tuple defines a set of n locations in image space. In the continuous n-tuple method, the grey-level values at each point are extracted and stored without any further processing. For the compiled version of the continuous n-tuple method, some further quantisation is necessary, for example, mapping from 256 levels of grey down to 8

By sampling the image in this way, we can deal directly with patterns defined as vectors of real numbers, or as grey scale images, as we have for the face recognition experiment described below.

Hence, for a given pattern vector $\mathbf{x} = x(1) \ldots x(d)$ we form a projection of this

$$\mathbf{y}_j(\mathbf{x}) = x(a_{j1}) \ldots x(a_{jn}) \qquad (2)$$

Denote the kth training vector of n-tuple j for the cth class as $\mathbf{y}^c_{jk}$.

The 'training algorithm' is simply to extract and store all these vectors from the set of training images.

Recognition of an unknown image x is performed as follows: For the j-th projected vector, $\mathbf{z}_j(\mathbf{x}) = x(a_{j1}) \ldots x(a_{jn})$ we find the closest (under distance metric D) stored vector for each class. We then sum (over all j) these distances to find the recognition score r_c

$$r_c = \sum_{j=1}^{m} \min_k D(\mathbf{y}^c_{jk}, \mathbf{z}_j) \qquad (3)$$

for each class c, and assign class membership to the class with the minimum total. This algorithm is summarised in Table 1.

Table 1: Algorithm for performing pattern classification with the continuous n-tuple classifier

Continuous n-tuple recognition algorithm Classifies pattern x in input array into class $c \in C$	
Step 1:	Initialise recognition vector $\mathbf{r}$ is a $\|C\|$-dimensional vector of real numbers For each class $c \in C$ Set $r_c = 0.0$
Step 2:	Sum distances over all projected vectors for x each class For $j = 1$ to m Set $\mathbf{z}_j(\mathbf{x}) = x(a_{j1}) \ldots x(a_{jn})$ For each class $c \in C$ Set $r_c = r_c + \min_k D(\mathbf{y}^c_{jk}, \mathbf{z}_j)$
Step 3:	Classification Assign $\mathbf{x}$ to class c where $r_c \leq r_d \forall d \neq c$

In the experiments reported below we choose a Manhattan (city-block) distance metric

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \qquad (4)$$

i.e. the sum of the absolute differences along each dimension of the vector. Experiments were also made using an unweighted Euclidean distance metric, but with significantly poorer results.

The advantages of the continuous n-tuple method are twofold. First, it directly incorporates a useful distance metric, and therefore offers better generalisation than the basic n-tuple method. This is because previously unseen points in the input space of a particular n-tuple are assigned values according to the closest recorded point from the training data. Evidence of this can be found in Table 3 and Fig. 3. Secondly, it allows us to deal directly with continuous or multi-level input spaces. There is no sacrifice in training speed (training speed should theoretically improve slightly, but this was not measurable in practice). However, recognition time given a direct implementation of the algorithm is significantly poorer, and gets worse linearly with respect to the number of stored exemplars. One solution to this that may be practical is to perform some quantisation of the input space (denote the number of quantisation levels as σ) and pre-compile the minimum distances to each stored class vector for each n-tuple from each possible address location. The algorithm for performing this simple compilation is given in Table 2. In this way, we get exactly the same recognition speed as the conventional n-tuple classifier, with something close to the recognition accuracy of the continuous n-tuple classifier. The only sacrifice now is training time; we have to step through all possible addresses (of which there are σ^n) in the address space of each n-tuple to set up the distance values for those addresses, and this must be repeated for each n-tuple and each class. For the face recognition problem described below, the results indicate that this is not only practical, but actually benefits the accuracy as well. Also, see Section 4.3 for discussion of how either a lazy or an approximate version of this compilation algorithm can be performed online.

Table 2: Algorithm for mapping a continuous n-tuple classifier into a standard n-tuple classifier

```
Algorithm to transform continuous n-tuple
into standard n-tuple system
transforms set of vectors y^c_j into look-up table n_cj
q(x) is a quantisation function that
maps each continuous scalar value x_i
into an integer in the range 0...(σ – 1)
t is an n-dimensional vector use as temporary storage
Algorithm:
For j := 1 to m
  For a := 0 to (σ^n – 1)
    For i := 0 to (n – 1)
      t(i) := (a/σ^i) mod σ
    For each class c ∈ C
      Set n_cj[a] = min_k (D(q(y^c_jk), t))
```

The continuous n-tuple method clearly has a close relationship with nearest-neighbour classifiers. In fact, each n-tuple acts as a kind of distance-weighted nearest-neighbour classifier for that subset of the input space. However, due to the fact that each n-tuple is a tiny projection of the original pattern (and indeed, typically only about 10–20% in total of the original pattern space needs to be sampled for optimum performance), the continuous n-tuple method is much more efficient. It also tends to perform more accurately. Note that if we use parameters of $n = d$ (in this case $d = 92 \times 112 = 10304$) and $m = 1$, then the continuous n-tuple method exactly implements a one-nearest-neighbour classifier.

2.1 Visualisation

Fig. 2 illustrates a typical compiled continuous n-tuple memory. These plots show the contents of particular duple sampling pixel locations (11, 41) and (33, 54). Each one has been trained on five samples from subjects (classes) 0 and 1, respectively, the first two people in the ORL (see below) database. The black circles indicate the values that occurred for each of the five training samples for each class. The blackness of each square is proportional to the Manhattan distance between that square and the closest circle. While the contents of each continuous n-tuple memory are different for each class, there often exists a good deal of overlap between classes. Despite this, when the aggregate is taken over hundreds of different n-tuples, the classes do generally become clearly separable.

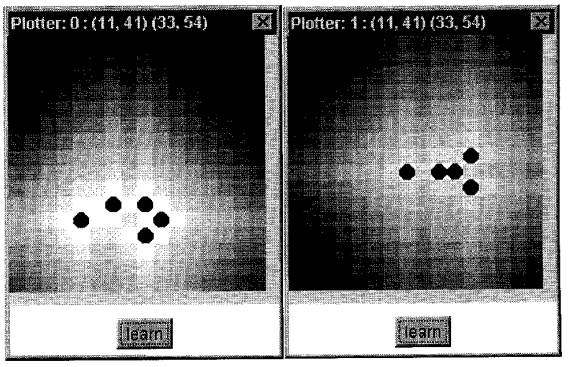

Fig. 2 *Illustration of a typical compiled continuous n-tuple memory*
These plots show the contents of particular tuple sampling pixel locations (11, 41) and (33, 54) with $\sigma = 16$, therefore each memory is a 16 × 16 matrix. Each memory has been trained on five samples from subjects 0 and 1, the first two subjects in the ORL database

3 Results

The continuous n-tuple classifier was tested on the Olivetti Research Laboratory (ORL) database, available from http://www.orl.co.uk/facedatabase.html. The database consists of four hundred images; ten each from 40 people, each image is 92 × 112 pixels, and there is considerable intra-subject variation. The database has been widely used by other researchers, which makes it a useful benchmark. All the systems quoted in Table 3 use five images for training and five for testing, but many of these are only based on one or two partitions of the dataset. To make the results presented in this paper (i.e. the n-tuple results) statistically significant, each result is based on 100 random partitions of the dataset (again, into five images for training and five for testing). N-tuple points were drawn from a uniform distribution over the entire image, using the default pseudorandom number generator built into the java.-math library.

The probabilistic decision-based neural net (PDBNN) results are taken from Lin *et al.* [7]. Self-organising map results combined with convolutional neural net (SOM + CN) results, together with the results of eigenfaces [8], top-down HMM and pseudo-2D HMM are taken from Lawrence *et al.* [9] and Samaria and Harter [10]. The humble nearest-neighbour classifier actually performs surprisingly well. This

Table 3: Error rates on the test data together with training and classification times for the ORL database

Method	Error rate (%)	Training time	Classification time
PDBNN	4.0	20 min	< 0.1s
SOM + CN	3.8	4h	< 0.5s
Top-down HMM	13.0	n/a	n/a
Pseudo-2d HMM	5.0	n/a	240s
Eigenfaces	10.0	n/a	n/a
n-tuple (4:500:2)	11.6 (2.8 : 5.5 : 21.5)	0.9s	0.025s
cont *n*-tuple (4:500:256)	3.79 (1.7 : 0.5 : 7.5)	0.9s	0.33s
cont *n*-tuple* (2:500:16)	3.59 (1.4 : 1.0 : 7.0)	2 min	0.013s
1-NN (10304:1:256	4.1 (1.6 : 1.0 : 9.5)	0	1s

Cont *n*-tuple* indicates the compiled version of the continuous *n*-tuple with 16 levels of quantisation. The *n*-tuple methods are suffixed with the parameters ($n : m : s$) in parentheses. The results show the mean for each experiment together with the standard deviation, and the minimum and the maximum error in parentheses. All results quoted in the Table use five images per class for training and the remaining five per class for testing. The various *n*-tuple results and one-nearest-neighbour (1-NN) classifier are each based on 100 experiments, each one using a different random partition of the data

is based on a city-block distance metric; a Euclidean distance version performs significantly worse. The figures in parentheses indicate that the nearest-neighbour classifier was implemented as a continuous *n*-tuple system with $n = 10304$ and $m = 1$. Of particular note is the fact that the compiled version (cont-*n*-tuple*) with the quantised input space ($\sigma = 16$), and cut down *n*-tuple size ($n = 2$) actually gives the best performance, and is also the fastest method for recognition, capable of recognising 76 images per second. The two-minute training time seems a reasonable price to pay for this, and is significantly faster than training the PDBNN or the SOM + CN architecture. All *n*-tuple timing results are based on a Java implementation running on a 200 MHz Pentium PC.

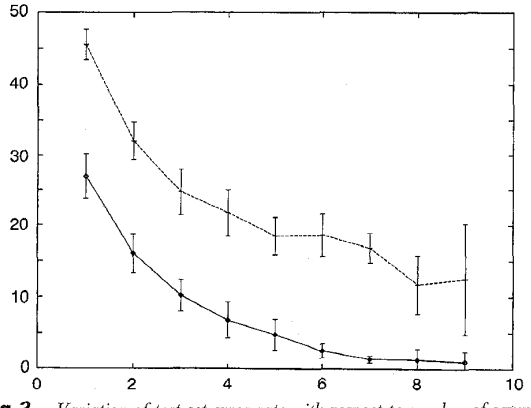

Fig. 3 *Variation of test set error rate with respect to number of samples used for training for the continuous n-tuple classifier (cont-n-tuple) and the standard n-tuple classifier (n-tuple)*
For both systems, values of $n = 3$ and $m = 200$ were used. This gives poorer results than with $n = 4$ and $m = 500$ for example, but allows the experiments to be run more quickly and exhibits the same trend. Each point on the graph represents the mean of five experiments, with error bars ±1 standard deviation from the mean. Each experiment used a different random partition of the dataset, and a different set of randomly chosen *n*-tuples
—◇— cont-*n*-tuple
– – – – *n*-tuple

The significance of the difference between the cont-*n*-tuple (4:500:256) and the 1-NN method was tested as follows. For 100 random partitions of the data set, the error rate of each method was measured to give 100 paired samples. A single-tailed *t*-test was then made to test the significance of the hypothesis that the continuous *n*-tuple classifier performs more accurately than the 1-NN method on this task. The test showed an acceptance of this hypothesis with a probability of 0.992 (i.e. confidence of 99.2%).

3.1 Variation of error rate with number of training samples

An investigation was made into how the error rate varied for the continuous and the standard *n*-tuple classifier with respect to the number of samples used for training. To speed things up, values of $n = 3$ and $m = 200$ were used. Fig. 3 shows the average test-set error rate for one training image per class through to nine training images per class. Each point on the graph is the mean of five experiments, each experiment based on a different random partition of the data.

4 Discussion

4.1 Memory requirements

For the ORL dataset, extensive tests have shown that recognition accuracy is not highly dependent on the value of *n*, providing that $n > 1$. As shown above, good results are obtained with $n = 2$, $m = 500$ and $\sigma = 16$. Consequently, the amount of RAM needed per class (i.e. per person) is ($16^2 * 500 = 125$ kbytes).

4.2 Recognition speed

Table 3 gives the recognition speed per image for the compiled continuous *n*-tuple classifier as being 0.013s per face image, given a 40-class problem. This is already fast enough to classify 76 images per second, but for verification purposes where a face is matched against the model of a claimant this can be multiplied by a factor of 40, i.e. it could verify approx 3000 images per second. There are two cases where this unprecedented speed might still be inadequate:

- if it was required to match against thousands of subjects in real-time
- if the face is at an unknown location, scale and rotation in the image; a complete recognition system might comprise a simple face locator (perhaps based on a continuous *n*-tuple classifier trained on thousands of faces) which returns multiple-face registration hypotheses.

There are many algorithms for face localisation. These may be based on general facial properties [11, 12], or on finding the optimal affine transformation between a particular face image and the image to be recognised. It is interesting to note that an efficient implementation of the latter method is given by Matas *et al.* [13] which shares a random sampling methodology with the continuous *n*-tuple system. The system of Matas *et al.* uses a Sobel distribution of sample points, as opposed to the uniform distribution employed here.

Hence, there may arise situations where it is necessary either to further increase the speed of the continuous *n*-tuple system, or to make the system more robust in the face of small registration errors (and hence able to cope with imprecise face registration). There are several ways of of achieving this. The first is to make the software faster or use a faster PC; the current implementation is written in Java and the timing measurements were made on a 200MHz PC. A C++ version might gain a factor of two in speed. PCs are continuing to follow Moore's law in their exponential speed increase against time.

Secondly, when the continuous *n*-tuple system has been compiled, it is just as straightforward to build in parallel hardware as a conventional *n*-tuple classifier. The implementation consists of some RAM, one summation unit per class (or one to cycle over all classes) and some simple logic to pick the winning class or decide if there is a conflict.

The third possibility would be to build a tangent–distance metric [14] into the system. This would make the system less sensitive to small shifts in image registration, and this possibility is currently being investigated.

Finally, in the case that a video stream of images is available, there are many ways in which the dynamic information implicit in the video sequence can be exploited; see McKenna *et al.* [15] for an example.

4.3 Real-time training

A real-time training algorithm can be outlined as follows. The case of $n = 2$ is particularly easy to visualise, as shown in Fig. 2. The contents of each continuous *n*-tuple memory for each class may be plotted on a two-dimensional grid consisting of $\sigma \times \sigma$ squares. Prior to training, all squares on the grid are set to infinity. When a particular pair of input values occur (i.e. the grey level values for the points indexed by this *n*-tuple are extracted from a training image), these index a particular square on the grid which is set to zero. All other squares are given values equal to their distance (e.g. Manhattan distance) on the grid from the indexed point. The cost of this operation is proportional to $\sigma \times \sigma$ (or in general σ^n) for the first training pattern. Thereafter, the cost decreases every time that a square not previously referenced is indexed, since it is only necessary to update the values at each square in each direction from the indexed point until a square is reached that already has a lower distance value than the distance to the indexed point. This means that when training the system on a video sequence of a new face, the first few frames would take a relatively long time to train (but still very fast compared to other architectures, and depending on the setup parameters and host system, perhaps still in real time), with the speed increasing until a face is so well known that new frames only cause minor, inexpensive updates to continuous *n*-tuple memory.

A simplified version of this algorithm has already been implemented, whereby each *n*-tuple training vector updates only the set of points within a given radius *r* of the training vector. Using $r = 10$, $\sigma = 32$, $n = 2$ and $m = 100$ it was possible to train on 200 images in 1.1s of CPU time. Hence, even for $m = 500$ it should be possible to train at a rate of 36 frames per second, which is more frames per second than most PC video capture systems can supply.

4.4 Possible extensions

Conventional *n*-tuple classifiers have already been given probabilistic interpretations [16–19], but these have been based on the frequency of occurrence of particular combinations of *n*-tuple values. There are some interesting possibilities for combining probabilistic approaches with the explicit distance metric method used in the continuous *n*-tuple classifier.

The current implementation uses a distance metric that operates directly on intensity information, which is sensitive to variations in contrast and brightness. The system might therefore perform even better if the images were first subjected to some simple global or local normalisation operators. However, such pre-processing would slow the system down, and any improvements have to be traded off against the loss of speed. The lighting conditions in the ORL database are quite consistent, and this may explain why the system performs so well without any pre-processing on this database.

It can be argued that different regions of the image provide more recognition information than others. One suggestion (by the anonymous referees and others) has been to modify the sampling process to reflect this. This would be an attractive way to improve performance, since it would not imply any extra computational cost; indeed, if we sampled the space more efficiently, we might even be able to get better accuracy with fewer *n*-tuples, hence further improving the speed. An initial experiment was made to explore this idea, where a radial elliptical sampling process was used. Each point was defined by a randomly generated angle (from a uniform distribution in the range $0–2\pi$), and a randomly generated length in the range 0–1. These points in polar co-ordinates were then mapped onto an ellipse centred on the image. The motivation behind this was that faces are roughly elliptical, and that more information might be in the centre of the image, which is naturally favoured in the radial sampling process. Ten experiments were then run to compare the radial elliptical (RE) sampling with the uniform random (UR) sampling, using $n = 2$ and $m = 100$. The RE method had an average error rate of 17.6% while the UR method had an average error of 5.2%. From this we can conclude that the RE method is significantly worse than the UR method. Perhaps some other sampling method might prove to be even better than UR, but this remains to be seen.

5 Conclusions

A new *n*-tuple classification method has been described, called the continuous *n*-tuple classifier. The system is conceptually simple and straightforward to implement either in hardware or software. A comprehensive set of results on the ORL face database demonstrate that this method is extremely competitive with other approaches, in terms of accuracy, training time and

recognition time. In particular, with $n = 2$ and $m = 500$ we can train the system at a rate of 36 frames per second, perform classification (for a 40-class problem) at 76 frames per second, and achieve an average error rate of just 3.59% on the ORL database.

In its current form, the system is capable of performing multi-class face recognition in real-time on video streams, providing that the face can be located in the image. This is a significant proviso, however, and one future avenue of research would be to test continuous n-tuple systems for locating faces.

Finally, there is nothing explicitly related to faces built into the continuous n-tuple classifier, although the fact that the system works so well when only sampling 10% of the input space seems to corroborate the notion that neighbouring pixels are highly correlated. It would therefore be interesting to test the system on other image recognition problems.

6 References

1 ALEKSANDER, I., and STONHAM, T.: 'Guide to pattern recognition using random-access memories', *IEE Proc.-E, Comput. Digit. Tech.*, 1979, **2**, pp. 29–40

2 ROHWER, R., and MORCINIEC, M.: 'A theoretical and experimental account of n-tuple classifier performance', *Neural Comput.*, 1996, **8**, pp. 629–642

3 ALLINSON, N., and KOLCZ, A.: 'Application of the cmac input encoding scheme in the n-tuple approximation network', *IEE Proc. Comput. Digit. Tech.*, 1994, **141**, pp. 177–183

4 TATTERSALL, G., and JOHNSTON, R.: 'Single-layer look-up perceptrons', *IEE Proc. F, Radar Signal Process.*, 1991, **138**, pp. 46–54

5 LUCAS, S.: 'Face recognition with the continuous n-tuple classifier'. Proceedings of the British Machine Vision Conference, 1997, pp. 222–231

6 LUCAS, S.: 'The continuous n-tuple classifier and its application to face recognition', *Electron. Lett.*, 1997, **33**, (20), pp. 1676–1678

7 LIN, S., KUNG, S., and LIN, L.: 'Face recognition/detection by a probabilistic decision-based neural network', *IEEE Trans. Neural Netw.*, 1997, **8**, pp. 114–132

8 TURK, M., and PENTLAND, A.: 'Eigenfaces for recognition', *J. Cognitive Neurosc.*, 1991, **3**, pp. 71–86

9 LAWRENCE, S., GILES, C., TSOI, A., and BACK, A.: 'Face recognition: a convolutional neural network approach', *IEEE Trans. Neural Netw.*, 1997, **8**, pp. 98–113

10 SAMARIA, F., and HARTER, A.: 'Parameterisation of a stochastic model for human face identification', Proceedings of 2nd IEEE workshop on applications of computer vision, Sarasoto, FL, 1994

11 CHEN, C., and CHIANG, S.: 'Detection of human faces in colour images', *IEE Proc. Vis., Image Signal Process.*, 1997, **144**, pp. 384–388

12 YANG, G., and HUANG, T.: 'Human face detection in complex backgrounds', *Pattern Recognit.*, 1994, **27**, pp. 53–63

13 MATAS, J., JONSSON, K., and KITTLER, J.: 'Fast face localisation and verification'. Proceedings of the British Machine Vision Conference, 1997, pp. 152–161

14 SIMARD, P., LE CUNN, Y., and DENKER, J.: 'Efficient pattern recognition using a new transformation distance', in HANSON, S., COWAN, J., and GILES, C. (Eds.): 'Advances in neural information processing systems 5' (MORGAN KAUFMAN, San Mateo, CA, 1993)

15 MCKENNA, S., GONG, S., and RAJA, Y.: 'Face recognition in dynamic scenes'. Proceedings of the British Machine Vision Conference, 1997, pp. 140–151

16 SIXSMITH, M., TATTERSALL, G., and ROLLETT, J.: 'Speech recognition using n-tuple techniques', *Br. Telecom Technol. J.*, 1990, **8**, pp. 50–60

17 ROHWER, R.: 'Two bayesian treatments of the n-tuple recognition method'. Proceedings of IEE 4th International Conference on Artificial Neural Networks, IEE, London, 1995, pp. 171–176

18 LUCAS, S., and AMIRI, A.: 'Statistical syntactic methods for high performance ocr', *IEE Proc. Vis., Image Signal Process.*, 1996, **143**, pp. 23–30

19 LUCAS, S., and AMIRI, A.: 'Recognition of chain-coded handwritten characters with the scanning n-tuple method', *Electron. Lett.*, 1995, **31**, (24), pp. 2088–2089