Article:
Hodge, V.J. orcid.org/0000-0002-2469-0224, O'Keefe, S. orcid.org/0000-0001-5957-2474 and Austin, J. orcid.org/0000-0001-5762-8614 (2009) A binary neural shape matcher using Johnson Counters and chain codes. Neurocomputing. pp. 693-703. ISSN 0925-2312

https://doi.org/10.1016/j.neucom.2008.07.013

A Binary Neural Shape Matcher using Johnson Counters and Chain Codes

Victoria J. Hodge	Simon O'Keefe	Jim Austin
Department of Computer Science University of York	Department of Computer Science University of York	Department of Computer Science University of York

Contact Author: **Dr. Victoria Hodge,**
Advanced Computer Architecture Group
Department of Computer Science
University of York, York, YO10 5DD, UK
Fax: +44 1904 432767
Email: vicky@cs.york.ac.uk

Abstract

Images may be matched as whole images or using shape matching. Shape matching requires: identifying edges in the image, finding shapes using the edges and representing the shapes using a suitable metric. A Laplacian edge detector is simple and efficient for identifying the edges of shapes. Chain codes describe shapes using sequences of numbers and may be matched simply, accurately and flexibly. We couple this with the efficiency of a binary associative-memory neural network. We demonstrate shape matching using the neural network to index and match chain codes where the chain code elements are represented by Johnson codes.

Keywords

Neural, Associative Memory, Binary Encoding, Shape Matcher, Figurative Image Retrieval.

1 Introduction

There has recently been an enormous growth in the storage and processing of digital images. Examples of image processing applications include: trademark retrieval; medical image analysis and diagnosis; and photograph archiving and searching. Content-based Image Retrieval (CBIR) is a sub-area of image processing. In CBIR the aim is to retrieve stored images from an image database that are similar to a query image. Trademark image searching is potentially one of the most important application areas for CBIR techniques. The work described in this paper is aimed primarily at matching and retrieving trademark images and other similar images such as clip art images.

Trademark databases are large with millions of stored trademarks which require accurate and fast indexing and retrieval to allow the databases to be searched. As new trademarks are designed, they need to be matched against existing trademarks to ensure there is no copyright conflict. Searching for potentially conflicting trademarks among databases comprising solely of image data is a difficult task [9]. A trademark retrieval system must be able to find all stored trademark images which might be judged similar to any new trademark. Current trademark retrieval systems rely on relatively low-level features within an image such as texture, colour or shape for image matching and retrieval. They attempt to match a query image against stored images in one of two ways: (1) comparing features generated from the images as a whole, or (2) matching features from individual image components (shapes) [10].

In this paper we focus on the latter approach: finding the constituent components (shapes) in a figurative image to allow those shapes to be used for image matching. Most experts in the field agree that shape similarity is the single most important determining factor regarding human similarity judgement [9]. Systems that compute the image similarity between a query image and the database of trademark images using features calculated from the *whole image* similarity are inconsistent **when** an additional component (shape) is present or when a component is missing from the image. Features such as the invariant moments [17] produce rather different statistics when additional components are added or components removed. Therefore, such a system will not be robust if the query consists of multiple components.

CBIR systems such as Mojsilović et al. [26] and Biederman et al. [7] propose that human image recognition works on various levels and that an agglomerative technique is used. Mojsilović et al. detect the edges in an image and Biederman uses lines and curves. Both approaches aggregate these primitives following perceptual principles to produce compound objects – shapes – that need to be represented and matched quickly, efficiently and accurately.

Example trademark systems that compute image similarity using component (shape-based) similarity include the STAR system [34] which uses features such as Fourier descriptors, grey level projection and moment invariants to represent the shapes identified by human subjects in the trademark image database. In the ARTISAN system [10], the shape boundaries are extracted using edge detection and the shapes are compared using a distance measure calculated using shape features, size and absolute position. In ARTISAN, each component is considered independently and the spatial relation between components within the same image is not considered. The trademark retrieval system of Alwis [3] performs multiple components matching by categorizing each component into a line, an arc or a closed figure and matching images using component similarity. Peng and Chen [27] represent each shape as a set of closed contours and represent each closed contour as a list of angles (chain code). Our proposed approach takes its inspiration from this approach and uses the AURA matching system [4] to perform fast shape similarity assessment with chain codes. Chain codes have been widely proposed and investigated in the literature [19, 25, 29, 30, 31] for shape matching and many other applications as they are simple with a low storage requirement and other shape features may be computed directly from the chain code.

The main contribution of our proposed method is speed of processing. Using AURA to underpin the chain code processing introduces a 15 times speedup compared to a standard chain code implementation and the method produces shorter vector representations than grid-based chain codes [28, 29] for shapes with sides longer than 24 pixels. Additionally, the matching results from the AURA method are identical to the results from a standard method. Many techniques proposed in the literature to speed up a standard algorithm do not produce identical results to the standard algorithm, for example [15]. Hence, a user cannot be sure that they will get the results they expect when they use the faster algorithm. With the speedup proposed in this paper, the user will always get the same matching results from the faster approach that they were expecting from the standard approach. The recall accuracy of chain codes has been documented elsewhere in the literature [31] who determined that the recall accuracy for chain code shape retrieval was just below Zernike moments and well above invariant moments and edge histograms. Zernike moments are complimentary to chain codes as Zernike moments are region-based and represent the features of the whole shape whereas chain codes are boundary-based and represent the features of the shape's outline [25] so the two could be used in tandem to represent shapes. Eakins et al [10] identified that using

combinations of multiple shape measures to represent images outperforms both single image representations (such as moments) and single shape measures (such as Fourier descriptors) for recall accuracy. Sanchez-Cruz et al [30] compared various chain code schemes to the JBIG compression standard for compression using letters, shapes and molecules. The chain codes generally outperform the JBIG standard with respect to compression in particular the A8 chain code scheme described later. Kato et al [19] demonstrated that the match time for face recognition system is seven times faster for a chain code representation compared to an image-based template representation.

In this paper we focus on two tasks to identify and represent image shapes. Firstly, we consider edge detection to identify the edges in the image and allow us to find and trace the shapes (closed contours) contained in the image. Secondly, we consider how to represent the shapes so they may be matched efficiently. We focus on speed of retrieval. As the Johnson code chain code produces identical results to a standard chain code, we do not analyse the recall but concentrate on the speed up compared to the standard approach through using AURA to underpin the method.

2 Edge Detection

Initially, we need to identify the shape boundaries to allow us to form the chain code representation. Edge detection is a widely used technique to identify the outlines of image shapes. There are various edge detectors in common use such as Canny [8], Sobel [12] or Laplacian [12]. In this paper, we use the simple 3x3 Laplacian kernel given in Table 1 to detect edges. This has the advantages of being simple, efficient and accurate.

Table 1 3x3 Laplacian kernel used for edge detection

-1	-1	-1
-1	8	-1
-1	-1	-1

FIGURE 1 HERE

If the edges of the shapes are jagged then this will have an adverse effect on the quality of the shape representation. Jagged edges cause edge representations such as chain codes to reflect the small scale irregularities ("jaggies[1]") rather than the higher level shapes we need for matching. To improve this, anti-aliasing techniques may be used to tidy the edges [11].

The Laplacian works well on region-based images such as those in figure 1. However, it is not suitable for processing noisy or textured images. For these images we recommend a pre-processing method such as that proposed by [11, 12, 16, 18] to tidy noise and equalise textured regions. We can then run the Laplacian kernel edge detector on these pre-processed images to identify the edges.

To process coloured images, we recommend either converting the image to greyscale using for example the *ImageMagick 'convert'* utility. This generally works but may run into problems where there are adjacent coloured regions of equal intensity. The identical intensity regions will resolve to the same shade of grey and the boundary between them will be lost.

[1] Jaggies are "stairlike lines that appear where there should be smooth straight lines or curves" – Wikipedia.

Another alternative is to adapt the Laplacian edge detector to work on the three colour channels R, G and B and use the information extracted from all three channels in combination to identify the edges in the image [20]. Alternatively, converting a colour image to greyscale may be considered a dimensionality reduction problem and the three colour channels may be processed using Principal Component Analysis to project the three channels on to a single dimension – the principal axis which is effectively a greyscale image. This axis is very close to (but not coincident with) the axis of the intensity of the image. The resultant image can then be processed using conventional Laplacian processing.

Once we have identified the edges, we need to tidy them as the edges detected may be more than one pixel wide which prevents chain codes being correctly formulated. There will be multiple direction choices within the multiple pixel regions and we need a unique representation hence a single pixel width line. Edge thinning [18] is a morphological operation which reduces (thins) the edges to one pixel wide lines from which we can correctly then generate the chain code.

3 Shape Representation

Following edge thinning, we need to link the edge pixels to form shapes as shown in Figure 2. A shape representation for figurative image retrieval should be invariant to translation, rotation and scale and similar shapes should have similar representations. Ideally the similarity of the representation should decrease monotonically as the similarity of the shapes decreases. Matching in trademark systems is "best matching" rather than exact matching so a degree of match score is necessitated to assess the best match.

FIGURE 2 HERE

In figurative image processing systems such as trademark systems, shapes may be represented and matched using metrics that are either boundary-based or region-based [25]. Comparing two shapes using chain-codes requires all chain code elements to be matched and for large irregular shapes the chain code may be long. Therefore, a fast and efficient chain code matching implementation is desirable. However, chain codes are translation invariant, have high discriminatory power for matching shapes [9], are flexible and, for regular shapes, chain codes are compact. Chain codes link the edge pixels into chains and represent the chains using a numerical coding system. Chain codes may be 4-directional or 8-directional. In this paper we focus on **8-directional C8 chain codes** and the variant **angular 8-directional A8 chain codes** although our methodology is applicable to any variant – it is merely a case of adapting the Johnson code to handle the directionality. An example C8 8-directional chain code is shown in Figure 3 & Figure 4.

FIGURE 3 HERE

FIGURE 4 HERE

In the C8 approach, each direction is represented by a number in the range *[0-7]*. The chain code traces the boundary of the shape, representing the direction across each grid cell by the appropriate direction number as shown in Figure 3. We note that the higher the definition of the image, the more grid cells, the longer the chain code and hence the more accurate the chain code.

An example A8 angular 8-directional chain code is shown in Figure 5 & Figure 6.

FIGURE 5 HERE

FIGURE 6 HERE

In the A8 representation, each angular direction is represented by a number in the range *[0-7]*. The chain code traces the angular changes in the boundary of the shape, representing the angular change at each corner by the appropriate angular change number as shown in Figure 5. We note again that the higher the definition of the image, the more grid cells, the longer the chain code and hence the more accurate the chain code. The advantage of the A8 representation compared to the C8 representation is that A8 is rotationally invariant but C8 is not. Rotational invariance is important when matching shapes as an inverted triangle will match a standard triangle with rotational invariance but the two shapes would not match otherwise.

By using our Advanced Uncertain Reasoning Architecture (AURA) [4] to match the chain codes we are able to efficiently match chain codes of long lengths. If the shape edges are complex then the edge pixels may be mapped directly to the edge representation. If the shapes are simple and regular polygons such as triangles, rectangles, rhombi etc then the shapes may be simplified to the number of sides and each side represented in the edge representation (see Gonzalez and Woods [12] for details of shape simplification).

Johnson Counter codes (also known as switch-tail ring counter codes) are binary reflected cyclic codes. They are used in the electronics industry for controlling operations in digital systems [24]. The number of bits in the code is arbitrary. Johnson Counter codes assign a code in which the Hamming distance between two adjacent codes in a contiguous list is 1 and the Hamming distance increases monotonically as the distance between the number codes increases as shown in Table 2. The cyclic code effectively "wraps around" so the Hamming distance between 0 and 7 in a 4-bit code is also 1. This is particularly relevant for chain codes which are also cyclic. The C8 and A8 8-directional numbers of the chain code correlate to the same 8 decimal numbers of the Johnson code which in turn map to the 4-bit Johnson Counter code as shown in Table 2. For both the chain code directions and the Johnson Counter code, 7 is most similar to 0 and 6; and, 0 is most similar to 7 and 1.

Decimal	4-bit Johnson codes
0	0000
1	0001
2	0011
3	0111
4	1111
5	1110
6	1100
7	1000

Table 2 List of 4-bit Johnson codes

In the next section, we describe a binary neural shape matcher that represents shapes using a fusion of chain codes and Johnson Counter codes to allow distance-based matching. The fusion produces binary vectors which are integrated with a binary Random Access Memory-based (RAM-based) neural network (AURA) to allow rapid, efficient and accurate indexing and matching of shapes in shape databases.

4 RAM-based Neural Networks

RAM-based networks were first developed by Bledsoe & Browning [6] and Aleksander & Albrow [1] for pattern recognition and led to the WISARD pattern recognition machine [2]. RAM neural networks are founded on the twin principles of matrices (usually called Correlation Matrix Memories (CMMs)) and n-tupling. For CMMs, each matrix accepts *m* inputs as a vector or tuple addressing *m* rows and *n* outputs as a vector addressing *n* columns of the matrix. During the training phase, the matrix weights $\mathbf{M}^{lk}$ are set to 1 if both the input row $\mathbf{I_j}^{l}$ and output column $\mathbf{O_j}^{k}$ are set. Therefore in CMMs, training is a single epoch process with one training step for each input-output association preserving the high speed. During recall from a CMM, the presentation of vector $\mathbf{I}_j$ elicits the recall of vector $\mathbf{O}_j$ as vector $\mathbf{I}_j$ contains all of the addressing information necessary to access and retrieve vector $\mathbf{O}_j$. This training and recall makes RAMs computationally simple and transparent with well-understood properties. RAMs are also able to partially match records during retrieval. Therefore, they can rapidly match records that are close to the input but do not match exactly.

4.1 AURA

AURA may be implemented using a C++ software library or using proprietary hardware coupled with a dedicated embedded C++ library [33]. Both the software and hardware AURA libraries provide a range of classes and methods for rapid partial matching of large data sets [4]. AURA software and hardware has been used in an information retrieval system [13], high-speed rule matching systems [5], 3-D structure matching [32] and trademark searching [3]. In this paper, we focus on an AURA software chain code matcher though the approach could be easily mapped to a hardware implementation. AURA software techniques have demonstrated superior performance with respect to speed compared to conventional data indexing approaches [14] such as hashing and inverted file lists which may be used for indexing. AURA software trains 20 times faster than an inverted file list and 16 times faster than a hashing algorithm. It is up to 24 times faster than the inverted file list for recall and up to 14 times faster than the hashing algorithm. We further demonstrate that AURA chain coding is up to 15 times faster than a standard chain code approach in this paper.

The rapid training and partial match capability of AURA coupled with encoding numeric chain codes as distance-based binary vectors for training and recall make AURA ideal to use as the basis of an efficient implementation. A more detailed definition of AURA, its components and methods now follows.

4.1.1 CMM Training

Correlation Matrix Memories (CMMs) are the building blocks for AURA systems. AURA uses binary input **I** and output **O** vectors to train records in to the CMM as in Equation 1 and Figure 7.

Equation 1 $\mathbf{CMM} = \bigvee_{\text{all } j} \mathbf{I}_j \times \mathbf{O}_j^T$ where $\bigvee$ is logical OR.

Training is a single epoch process with one training step for each input-output association (each $\mathbf{I}_j$ x $\mathbf{O}^T_j$ in Equation 1) which equates to one step for each record $\mathbf{I}_j$ which is associated with a unique identifier vector $\mathbf{O_j}$. Each $\mathbf{O_j}$ has a single bit set. The first bit is set in the identifier vector $\mathbf{O_0}$ for the first record to store; the second bit is set in the identifier vector $\mathbf{O_1}$ for the second record to store and so on.

FIGURE 7 HERE

4.1.2 CMM Recall

To retrieve the matching stored records for a particular query record, AURA effectively calculates the dot product of the input vector $\mathbf{I}_k$ and the CMM, computing a positive integer-valued output vector $\mathbf{O}_k$ as in Equation 2. $\mathbf{O}_k$ is the same dimensionality as the number of columns in the CMM and represents the sum of the 1s (set bits) in each CMM column that coincide (are in the same row) as a 1 in $\mathbf{I}_k$.

Equation 2 $\mathbf{O}_k^T = \mathbf{I}_k \bullet \mathbf{CMM}$

The integer-valued output vector $\mathbf{O}_k$ is thresholded to produce a superimposed binary output vector $\mathbf{S}_k$ as in Equation 3 which has a bit set for each selected chain code.

Equation 3 $\mathbf{S_k} = \boldsymbol{L\text{-}Max}(\mathbf{O_k})$

We use the *L-Max* thresholding technique (detailed in [4]) to threshold $\mathbf{S_k}$. L-Max thresholding essentially retrieves *at least L* top matches by setting a bit in the superimposed output vector $\mathbf{S}_k$ for every location in the integer-valued output vector that has a value higher than a threshold value. *L* is set to a value equivalent to the number of chain codes to retrieve. The threshold value is set to the highest integer value that will retrieve at least *L* matches. For example, to retrieve the top 2 matches from figure 10, L-Max thresholding would threshold the output vector at 44 to retrieve a superimposed binary vector. This superimposed binary vector would have a bit set in the first two positions (`1100`) as the two leftmost columns in the CMM sum to 44 or more indicating that the chain codes stored in the two leftmost columns of the CMM are the two best matches (chain codes A and B from figure 10).

5 Chain Codes

In this section, we focus on how to represent chain codes in AURA. For the methodology described in this paper, we:

- Train the binary distance-based chain codes and their identifier vectors into the CMM allowing them to be matched.
- Match unseen binary chain codes using the trained CMM.

For C8 chain codes, Lu [23] introduced procedures for normalising the chain codes to ensure invariance to rotation and scaling. He proposes orienting shapes along the principal (major) axis of the shape. The minor axis lies perpendicular to this major axis. This forms a superimposed rectangle which may be subdivided into a grid of cells. To ensure scale normalisation we fix the perimeter-size of this grid so that it is equal for all shapes, for example a 4x4, 3x5, 2x6 or 1x7 grids all have equivalent perimeter length of 16. We thus superimpose the most appropriate of these onto the shape. If we fix the starting point to the top right cell then we have normalised for rotation and scale.

For an A8 chain code, we do not need to establish the major axis as the method is rotationally invariant already. Hence, to produce A8 chain codes, the user needs to superimpose the normalisation grid to ensure the chain codes is normalised for scale as for the C8 method.

For both A8 and C8, we further enhance Lu's normalisation procedures to ensure that all stored and query chain codes are equivalent length. We propose counting diagonals twice.

With an 8-directional chain code there are 3 routes from point (1, 1) to point (2, 2). They are: east then north (2 steps and thus 2 codes in a chain); north then east (2 steps and thus 2 codes in a chain); or, north-east which is 1 step. By ensuring that the diagonal counts twice, all routes are 2 steps and 2 codes. The new chain code (with double diagonals) for Figure 4 is given in Figure 8.

In the remainder of this section, we employ C8 chain codes which have been normalised using Lu's recommendations and our double diagonal augmentation to illustrate how to process chain codes in AURA. Switching to the alternative A8 chain codes method requires revising the representation from figure 4 to the shape representation for figure 6, i.e., replacing the direction codes with the angular direction codes. The processing of the chain codes within AURA is equivalent for both C8 and A8. The differences are all contained in the pre-processing which generates the chain code representations. Both A8 and C8 use the chain code numbers 0-7 and are scale, rotation and translation invariant following our normalisation procedures.

5.1 Chain Codes in AURA

5.1.1 Input Vectors

AURA requires binary input vectors for training and storage. Thus, we convert the numeric chain code to a binary Johnson code equivalent. If we take the chain code for Figure 4 then we must convert each number in turn to a Johnson code as in Figure 8.

FIGURE 8 HERE

We concatenate all Johnson codes together to form **I**. We then take **I** and negate all vector elements to form $\hat{\mathbf{I}}$ as shown in Figure 9.

FIGURE 9 HERE

By concatenating **I** and $\hat{\mathbf{I}}$ we produce **I'** the distance-based binary chain code (the input vector for the CMM training) as in Equation 4.

Equation 4 $I'=I \oplus \hat{I}$ where $\oplus$ represents concatenation.

During recall, matching the first half (**I**) of **I'** will count the number of matching 1s between the query and stored vectors and matching the second half ($\hat{\imath}$) will count the number of matching 0s between the query and stored vectors. We note that exactly 50% of the elements in I' will be set (active) using this approach. Each Johnson code element in I is inverted in $\hat{\imath}$ so each element will appear twice in I' as both a 0 (inactive) and a 1 (active) thus 50% of the elements are active. Figure 10 shows a trained CMM where the rows are Johnson code bits and each column represents a distance-based binary chain code $\mathbf{I'_j}$ (image shape) stored for matching.

5.1.2 Chain Code Matching (Recall)

To find the best matches for a particular chain code we initially create an input vector **Q** as per the training input vector so the chain code 112334556770 would be mapped to a Johnson code input vector as in Figure 8. This chain code is 12 elements long and each element comprises 4 binary bits so the perfect match would score 48.

FIGURE 10 HERE

We then logically negate this binary input vector to produce $\hat{\mathbf{Q}}$ and append $\hat{\mathbf{Q}}$ to the end of the query vector **Q**. The resultant query input vector is thus given by Equation 5.

Equation 5 $\mathbf{Q'=Q \oplus \hat{Q}}$

We note that 50% of the elements in **Q'** will be set so 50% of the CMM rows will be activated during recall thus recall is predictable and constant for equivalent length chain codes (ignoring extraneous factors).

We input **Q'** to the CMM to elicit a summed output vector, **O** as per Equation 2, which effectively counts the number of matching 1s AND 0s between the input and each, stored chain code.

The proposed method then calculates the score for each stored chain code using the summed output vectors (counting matching 1s and 0s, **O**).

For each stored chain code (each position **j** in the summed vectors) we calculate the score as given by Equation 6.

Equation 6 $\mathbf{Score=O_j}$

This is essentially the negated XOR (**~XOR** or inverse Hamming Distance) of the input with each stored binary chain code and counts the number of matching 1s AND 0s between the query and the stored code. This score represents the degree of match and will decrease monotonically as the similarity between the input and a stored code decreases. For best matching, we L-Max threshold with *L* set to 1 and thus store the index (position **j** in the summed output vector) of the chain code(s) with the highest score.

5.1.3 Worked Example

FIGURE 11 HERE.

Figure 11 shows four example shapes A, B, C & D to be compared to a query shape Q (shown in Figure 4). In the following, we compare the retrieval process of the standard technique to the retrieval process of the AURA technique when matching the query against the four stored shapes to identify the best matching shape. Figure 12 shows the standard chain code for the query Q and the four stored shapes A, B, C & D respectively; the binary codes for the query Q and the four stored shapes A, B, C & D respectively and Figure 13 shows the inverted binary codes for the query $\hat{Q}$ and the four stored shapes $\hat{A}, \hat{B}, \hat{C} \& \hat{D}$ respectively.

FIGURE 12 HERE

FIGURE 13 HERE

5.2 Standard Chain Code Technique

To compare two chain codes where chain code X is 1234 and chain code Y is 7654 then the standard approach performs an element by element cumulative distance calculation. The standard technique firstly compares chain code X element 1 against chain code Y element 1,

For these two elements which are 1 and 7 respectively, the difference is 2 due to the cyclic code used in chain coding as shown in figure 3. The standard technique then compares chain code X element 2 against chain code Y element 2. For these two elements which are 2 and 6 respectively, the difference is 4 and the cumulative difference after two element comparisons between chain codes X and Y is now 4+2=6. The overall cumulative difference of the two chain codes can then be calculated by summing all of the individual element differences.

5.3 Comparison of AURA versus Standard

For the standard technique the scores for the four shapes A, B, C & D in Figures 10 –12 compared to the query shape are:

Score_A = 2
Score_B = 4
Score_C = 6
Score_D = 8

For the standard technique, the lowest score indicates the best match so A is the best match (as we would expect) and differs to the query by 2 (i.e. two 1/8 directions are different between the query and shape A). The differences are in the first two elements of the chain code where the query begins "11" and A begins "20" giving a cumulative difference of 2. Shape D is least similar to the query as it scores highest and thus differs in eight 1/8 directions. It differs from the query in elements 1, 2, 4, 5, 7, 8, 10 & 11.

For the AURA approach, the scores for the 4 stored chain codes in Figures 10 –12 compared to the query shape are:

Score_A = 46/48
Score_B = 44/48
Score_C = 42/48
Score_D = 40/48

For the AURA technique, we are counting matching 1s in Q compared to A, B, C & D and matching 1s in $\hat{Q}$ compared to $\hat{A}, \hat{B}, \hat{C} \& \hat{D}$. In contrast to the standard approach, the shape with the highest score is the best match to the query. For the standard technique we are counting differences (hence the lowest score represents the best match). The standard score is the maximum score permissible in AURA minus the actual score, i.e. for shape A, the maximum permissible score is 48 and the actual score is 46 so the standard score is 2 (as seen above). Shape A is the most similar to the query as it scores highest. Shape D is least similar to the query as it scores lowest. Shape A differs from the query by 2 which indicates that it differs in two 1/8 directions – those shown by dotted lines in Figure 11. Shape D differs in eight 1/8 directions – those shown by dotted lines in Figure 11.

If we **~XOR** the query vector with the best matching vector(s), where the resultant vector is 0 is where the stored vector differs from the query. This facility is useful for pinpointing shape variations.

6 Evaluation

We compare the AURA-based method against a standard chain code representation for speed of training and recall. We also verify that both representations retrieve the same matches. All analyses were performed using a 3.4 GHz Pentium 4 PC with 2GB RAM running

Slackware Linux. All code was written in C++ and compiled with the G++ compiler using identical optimisations

The standard method is implemented using a lookup table to allow comparison. Chain codes are stored as STL vectors of integers. Chain code elements (vector elements) are then compared using the lookup table.

```
int lookupTable[8][8] =
      {
      {0,1,2,3,4,3,2,1},
      {1,0,1,2,3,4,3,2},
      {2,1,0,1,2,3,4,3},
      {3,2,1,0,1,2,3,4},
      {4,3,2,1,0,1,2,3},
      {3,4,3,2,1,0,1,2},
      {2,3,4,3,2,1,0,1},
      {1,2,3,4,3,2,1,0} };
```

To compare two chain codes where chain code 1 is 1234 and chain code two is 7654 then the C++ code indexes the lookup table. To compare chain code 1 element 1 against chain code 2 element 1, the C++ indexes the row and column representing the respective chain code element values. For these two chain codes, that is row 1 (the value of chain code 1 element 1 where rows are numbered from 0-7) and column 7 (the value of chain code 2 element 1 where columns are numbered from 0-7). The difference is therefore 2. The cumulative difference of the two chain codes can then be calculated by summing the individual element differences.

For the evaluations here, we require a large dataset to enable a thorough test and to demonstrate the practicality of the proposed method for trademark retrieval where potentially millions of images need to be indexed and matched. The available image datasets are too small. The UKPTO data set [10] used in various evaluations comprises 10,745 images but many are noisy and textured and there is no shape ground truth information with the dataset so this set would not be suitable for our chain code comparisons. The de facto standard shape retrieval dataset is the MPEG-7 shape dataset [21] used widely in the literature but this only contains 70 classes of shapes with each class containing 20 shape variants giving 1400 shapes in total which is insufficient for our evaluation. [29] use a set of shapes containing shape similarity information for their evaluation but this set only contains 3 queries and 22 shapes. Therefore, to allow comparisons, we generate synthetic sets of chain codes using a random number generator in Java. The chain codes are formed from the numbers 0-7 and could be C8 or A8 as they are both generated from equivalent number sets {0-7}. The differences between C8 and A8 all lie in the pre-processing as described in section 5. For our first analysis, the routine outputs 1.5 million chain codes each with 100 dimensions (100 numbers). For the second analysis, the routine outputs 100,000 codes each with 100 dimensions, 100,000 codes each with 200 dimensions, 100,000 codes each with 300 dimensions and finally 100,000 codes each with 400 dimensions

In the first part of our evaluation, we compare the training, recall and overall processing times of the AURA and the standard method using varying numbers of stored chain codes but with each chain code of fixed dimensionality. We train 250K, 500K, 1.0M and 1.5M chain codes of dimensionality 100 into the index for the AURA method and the standard method respectively. We then record the time to retrieve the top match for the first 1000 chain codes in each data set for the AURA and standard methods respectively. The graph in Figure 14

shows the overall time which includes the time to train the respective methods with the full database (250K, 500K, 1.0M and 1.5M 100-D chain codes) along with the retrieval time to find the best match for the first 10 chain codes and the recall times for the AURA and standard methods.

FIGURE 14 HERE

In the second part, we compare the training, recall and overall processing times of the AURA and the standard method using a fixed number of stored chain codes but with each chain code of varying dimensionality. We train 100K chain codes of dimensionalities 100, 200, 300, 400 into the index for the AURA method and the standard method respectively. We then record the time to retrieve the top match for the first 1000 chain codes in each data set for the AURA and standard methods respectively. The graph in Figure 15 shows the recall times for the AURA and standard methods.

FIGURE 15 HERE

From both graphs, the AURA method is between 8 and 15 times faster than the standard method. Both methods show linear growth as the number of chain codes increases (while the dimensionality remains static) and linear growth as the dimensionality increases (while the number of chain codes remains static). However, the gradient of the standard method is much steeper than the gradient of the AURA plot so the total time for the standard method increases much more quickly then the AURA method.

The training time is negligible compared to the recall time for the standard method. For example, for 500K 100-D chain codes, the training time for the standard method is 2 seconds and the recall time is 6399 seconds giving a total train and recall time of 6401 seconds. For AURA, the recall time forms the bulk of the total time as the inputs have to be processed into the AURA binary vector representation which increases the training time compared to the standard method which does not process the inputs. However, this extra pre-processing required by AURA is easily offset by the much reduced retrieval time and thus much reduced overall time compared to the standard method. For the same 500K 100-D chain codes, the training time for AURA is 82 seconds and the recall time is 436 seconds giving a total train and recall time of 518 seconds (compared to 6401 seconds for the standard method)..

The AURA method and the standard method both retrieve the same best matches 100% of the time so the AURA method is a faithful implementation of the standard method.

Sajjanhar & Lu [23, 28, 29] introduced a grid-based shape representation based on chain codes which uses a binary vector to represent each shape analogous to the binary vector we use for our AURA implementation. During their evaluations Sajjanhar & Lu found that: "*the efficiency in terms of the storage requirements and the computation costs of the grid-based method is comparable to that of the Fourier descriptors method and the moment invariants method*" [28]. Next, we compare the efficiency of the AURA method to the grid-based method.

If we use an A8 chain code representation in AURA then it is rotation invariant. If we use the C8 scheme then we need to normalise to introduce rotational invariance as described. For both A8 and C8 we then need to normalise for scale invariance as described. The proposed grid-based approach [28] is normalised in the same manner as C8 to normalise for rotation

and scale. The grid-based shape representation is produced by mapping each shape onto a fixed-size grid. The grid is then scanned and each cell is labelled 0 or 1 depending on whether the number of shape pixels contained in that cell is below or above a pre-specified threshold. The resultant grid of 0 and 1 is then scanned from left to right and top to bottom to produce a binary number to represent the enclosed shape. The difference between the query shape and the stored shapes is calculated by the difference of the binary vectors (logical XOR).

The AURA approach uses 3 bits per chain code element, the vector **I'** is double length $I'=I \oplus \hat{I}$ so the representation requires *3*2*perimeter* binary bits. The grid-based approach scans the contents (area) of the bounding box and uses one bit per pixel so the representation requires *height*width* bits. For a square shape with perimeter $4l$ and area l^2, our proposed approach for either C8 or A8 will have a smaller bit representation if $3*2*4l < l^2$. If $l < 24$ then the grid-based will have a shorter bit vector but if $l > 24$ then the AURA-based approach will have a shorter bit vector. A shorter bit vector will process more quickly and we envisage that most shapes in trademark images will have sides longer than 24 pixels and hence, the AURA method will be the most efficient overall.

7 Multi-Resolution Chain Coding

We demonstrated that our method can match 1.5 million chain codes of 100 dimensions with a training time of 2225 seconds and a recall (match) time of 119 seconds. These times are generally practical for such large datasets but the speed may be increased further by introducing different matching granularities if a specific application required faster processing. A low resolution coding may be used for rapid, low precision matching to, perhaps, narrow the search space in a very large database of chain codes down to a subset of candidate matches. A more precise coding, which is slower to match against, may then be used on the reduced set of candidate matches. To support multi-resolution search, we introduce multi-resolution chain codes. By using a point-region quad-tree[2] (PRQ) reduction, we can produce a boundary at various levels of resolution. Each different resolution requires a separate CMM for training and matching due to the variation in length of the vectors.

The lowest resolution chain coding uses a grid with few cells as shown in Figure 16. As the resolution increases down the PRQ (as the number of cells in the grid increases monotonically by a factor of 4) then the size and precision of the chain code increases by a factor of 2 (the perimeter is twice as long) so the number of rows in the CMM would increase by a factor of 2 for each layer in the PRQ. The matching will obviously be slower due to the increase in size of the vectors but this will be offset by the increase in precision.

FIGURE 16 HERE

If the lowest resolution storage requires **x** amount of memory and the memory requirement doubles for each step increase in precision, then the total memory for **k** steps of increase in precision would be given by equation 7.

Equation 7 $\mathbf{x + 2x + 2^2x + 2^3x + \ldots + 2^kx = x(2^{k+1}-1)}$

[2] A point region quad-tree is a quad-tree where each node must have exactly four children, or have no children (leaf).

Assuming that the search time of memory is proportional to the memory size (**time ∝ size**) and assuming that on average **k/2** steps are required before the target is found, then the average memory searched is given by equation 8.

Equation 8 $\mathbf{x + 2x + \ldots + 2^{\frac{k}{2}}x = x(2^{\frac{k}{2}+1} - 1)}$

If only the high precision memory was searched rather than a multi-resolution search, this high precision search would be $\mathbf{2^k x}$.

Therefore, the speedup due to the multi-resolution approach is:

$$\frac{2^k}{2^{\frac{k}{2}+1} - 1}$$

This increases as **k** increases, i.e., there is more gain when greater precision is required.

Thus, if the database is relatively small, train a single CMM with chain codes at high precision (the largest chain code length permissible within the storage available). This may then be used for matching query shapes.

If the database is very large then we propose training the shapes into a low precision CMM which will minimise the storage requirements. If we query this low precision CMM to retrieve a set of candidate shapes then these candidates may be trained into a subsequent higher precision CMM. Again, querying this higher precision CMM will retrieve a set of better matches. This cycle of retrieving matches, increasing the precision, training a higher precision CMM and then querying can be repeated until the database of shapes has been reduced to the requisite number of matches.

8 Conclusion

In this paper we have introduced a binary neural shape matcher. The technique derives the edges in an image using Laplacian edge detection, tidies the edges using edge thinning, traces the closed contours within the edges and maps these edge traces onto chain codes. The shape matcher framework uses the AURA binary neural network framework to produce chain code storage and matching system allowing us to calculate shape similarity from chain code similarity. The AURA implementation is shown to be much faster than a standard chain code technique. The approach allows us to combine the simplicity, translation, rotation and scale invariance and flexibility of chain codes with the speed of AURA to produce fast flexible approach.

While some authors have stated that chain codes have high discriminatory power for matching shapes [9] other authors have found slightly lower recall for chain code measures [31] compared to other shape measures such as Zernike moments. However, chain codes are frequently used in conjunction with other shape matching methods, for example [35] and [10] found that combinations of shape measures outperform individual shape measures for recall and precision. Chain codes are boundary-based and capture detailed information about shape boundaries. They form a companion measures to region-based measures [25] such as Zernike moments which capture information about the overall shape. It is also worth noting that when chain codes are used in conjunction with other methods, speed is even more important

as the overall speed of a multi-feature system is dependent on the speed of the individual measures used. Therefore, a high speed approach is desirable. Some other shape measures may actually be derived from the chain code representation itself such as first and second order moments [22].

The main drawback of the chain code representation stems from the ability of chain codes to capture fine details of shape boundaries in that they are susceptible to noise particularly in the shape boundary where extra black pixels or missing boundary pixels due to salt and pepper noise will affect the chain code number produced and thus affect the match score. Therefore, to use chain code matching successfully, the images need to be clean or cleaned prior to processing and noise removed using suitable filters such as median filters or dilation/erosion filters [12] or using a preprocessing such as [16].

The proposed approach will be used for shape matching in trademark retrieval system. We feel the technique is flexible and easily extended to other domains/systems where distance-based numeric code matching is required and also to other application areas where chain codes are used such as face recognition, letter comparisons etc.

References

[1] Aleksander, I., & Albrow, R.C. Pattern recognition with Adaptive Logic Elements. IEE Conference on Pattern Recognition, pp 68-74, 1968.

[2] Aleksander, I., Thomas, W.V., & Bowden, P.A. Wisard: A radical step forward in image recognition. Sensor Review, pp 120-124, 1984.

[3] Alwis T.P.G.L.S. “Content-Based Retrieval of Trademark Images”. Ph.D. Thesis, University of York, February 2000.

[4] Austin, J. Distributed Associative Memories for High Speed Symbolic Reasoning. In, IJCAI Working Notes of Workshop on Connectionist-Symbolic Integration: From Unified to Hybrid Approaches, pp 87-93, 1995.

[5] Austin, J., Kennedy, J., & Lees, K. A Neural Architecture for Fast Rule Matching. In, Artificial Neural Networks and Expert Systems Conference (ANNES’95), Dunedin, New Zealand, 1995.

[6] Bledsoe, W.W., & Browning, I. Pattern recognition and Reading by Machine. In, Proceedings of Eastern Joint Computer Conference, pp 225-231, 1959.

[7] Biederman, I., Subramaniam, S., Bar, M., Kalocsai, P, & Fiser, J. Subordinate-Level Object Classification Re-examined. Psychological Research, 62:131-153, 1999.

[8] Canny, J. F. A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-698, 1986.

[9] Eakins, J.P. Trademark image retrieval - a survey, in: Multimedia Storage and Retrieval Techniques - State of the Art (Lew, M, ed). Springer-Verlag, Berlin (2000).

[10] Eakins, J. P., Riley, K. J. & Edwards, J. D. Shape Feature Matching for Trademark Image Retrieval, In, Image and Video Retrieval: Second International Conference, CIVR 2003. LNCS, vol. 2728, Jan 2003, Pages 28 – 38.

[11] Foley, J., van Dam, A., Feiner, S. & Hughes, J. Computer Graphics: Principles and Practice, 2nd edition in C: Addison-Wesley, ISBN no: 0-201-84840-6, 1995

[12] Gonzalez, R.C. & Woods, R.E. Digital Image Processing, 2nd Edition, Prentice-Hall, 2002

[13] Hodge, V., Integrating Information Retrieval & Neural Networks, PhD Thesis, Department of Computer Science, The University of York, 2001.

[14] Hodge, V. & Austin, J. An Evaluation of Standard Retrieval Algorithms and a Binary Neural Approach. Neural Networks 14(3), pp. 287-303, Elsevier, 2001.

[15] Hodge, V. & Austin J. **A** Binary Neural k-Nearest Neighbour Technique. Knowledge and Information Systems*,* 8(3): pp. 276–292, Springer-Verlag London Ltd, 2005.

[16] Hodge, V., Hollier, G., Austin J. & Eakins J. Identifying Perceptual Structures In Trademark Images. In, Fifth IASTED International Conference on Signal Processing, Pattern Recognition, and Applications (SPPRA 2008)*,* February 13 – 15, 2008. Innsbruck, Austria.

[17] Hu, M-K. Visual pattern recognition by moment invariants. IRE Trans. on Information Theory, IT-8:pp. 179-187, 1962.

[18] Jain, A. K. *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc. 1989

[19] Kato, Y., Hirano, T., & Nakamura, O. Fast template matching algorithm for contour images based on its chain coded description applied for human face identification. *Pattern Recognition.* 40(6) pp. 1646-1659, June 2007.

[20] Koschan, A. Comparative Study On Color Edge Detection. In, Proceedings 2nd Asian Conference on Computer Vision ACCV´95, Singapore, December 1995, Vol. III, pp. 574-578

[21] Latecki, L., Lakamper, R. & Eckhardt, U. Shape descriptors for nonrigid shapes with a single closed contour. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000, pp. 424–429.

[22] Levine M.D. Vision in Man and Machine. New York: McGraw-Hill, 1985.

[23] Lu, G. An approach to image retrieval based on shape. Journal of Information Science, 23(2): pp. 119-127, 1997.

[24] Mano, M. Digital design. 3rd Edition. Prentice-Hall, NJ, 2002. ISBN: 0130621218.

[25] Mehtre, B., Kankanhalli, M., & Lee, W., *Shape Measures for Content Based Image Retrieval: A Comparison*, Information Processing & Management, Volume 33, Number 3 (June 1997), pp. 319-337.

[26] Mojsilovic, A., Gomes, J. & Rogowitz, B. ISee: Perceptual features for image library navigation, Proc. 2002 SPIE Human Vision and Electronic Imaging.

[27] Peng, H.L. & Chen, S.Y. Trademark shape recognition using closed contours. Pattern Recognition Letters, 18:791--803, 1997.

[28] Sajjanhar, A. & Lu, G. A grid based shape indexing and retrieval method, Special issue of Australian Computer Journal on Multimedia Storage and Archiving Systems, Vol. 29, No.4, November 1997, pp. 131-140.

[29] A Sajjanhar & G. Lu, A comparison of techniques for shape retrieval, International Conference on Computational Intelligence and Multimedia Applications, 9-11 Feb. 1998, pp. 854-859.

[30] Sánchez-Cruz, H., Bribiesca, E., & Rodríguez-Dagnino, R. M. 2007. Efficiency of chain codes to represent binary objects. *Pattern Recogn.* 40, 6 (Jun. 2007), 1660-1674.

[31] Shih, J.-L.& Chen, L.-H. A new system for trademark segmentation and retrieval. Image and Vision Computing, 19(13) pp 1011-1018, November 2001.

[32] Turner, A., & Austin, J. Performance Evaluation of a fast Chemical Structure Matching Method using Distributed Neural Relaxation. In, 4th International conference on Knowledge Based Intelligent Engineering Systems, 2000.

[33] Weeks, M., Freeman, M., Moulds A. & Austin, J. Developing Hardware-Based Applications Using PRESENCE-2 . Perspectives in Pervasive Computing 2005, 25th October 2005 at the IEE, Savoy Place, London

[34] Wu J. K., Mehtre B. M., Gao Y. J., Lam C. P. & Narasimhalu A. D. STAR – a Multimedia Database System for Trademark Registration. In, First Intl. Conf. on Applications of Databases (ADB-94), pp. 109-122, Sweden, June 1994.

[35] Yin, P. & Yeh, C. Content-based retrieval from trademark databases. *Pattern Recogn. Lett.* 23, 1-3 (Jan. 2002), 113-126.

Figures and Captions

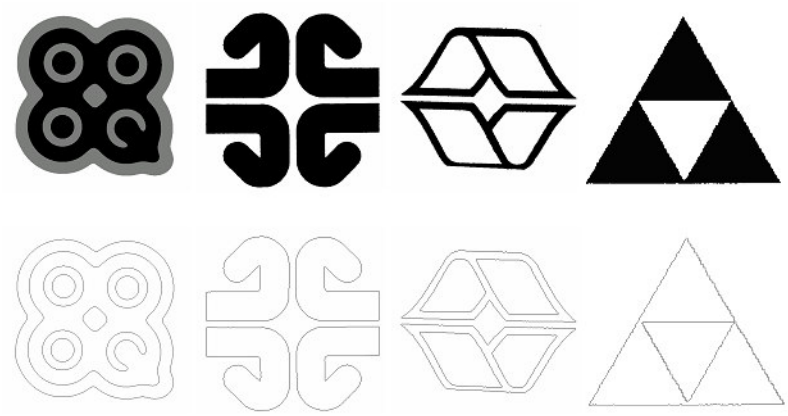

Figure 1. Showing four images (top row) and the edges detected by the Laplacian edge detector for each of the four images in the bottom row.

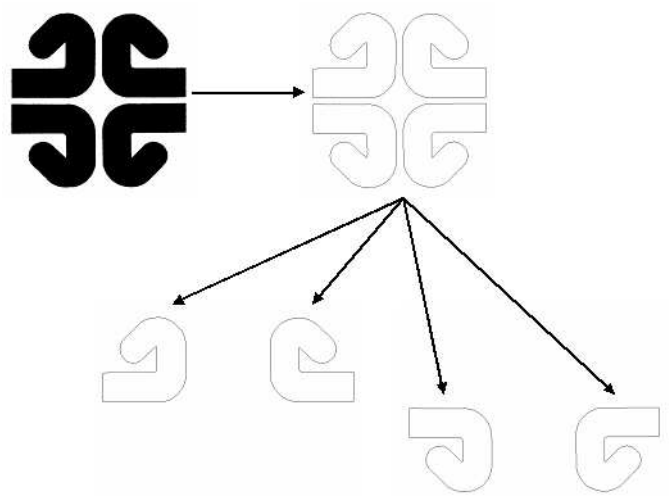

Figure 2. The original image is processed by the Laplacian edge detection to identify the edges. The closed contours (shapes) within the edges are then identified. Each of the closed contours (shapes) will be mapped to its respective chain code representation.

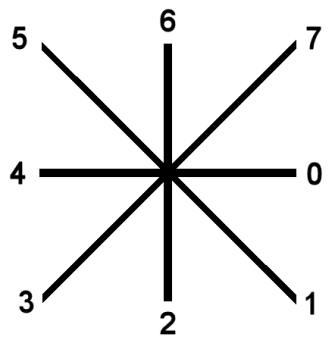

Figure 3. The 8-directions and associated numbers for C8 8-directional chain codes.

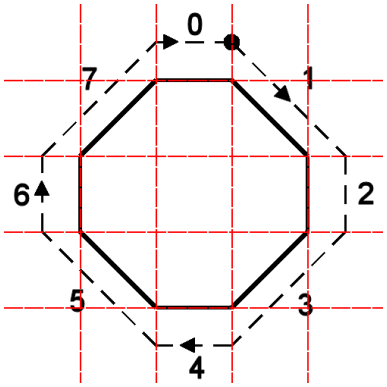

Figure 4. C8 Chain code for an octagon - starting from the dot, the chain code is 12345670.

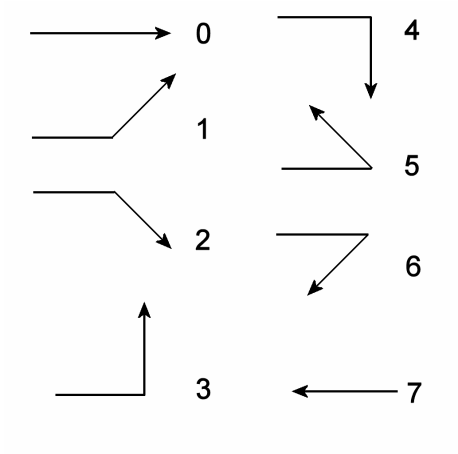

Figure 5. The 8-directions and associated numbers for A8 8-directional chain codes.

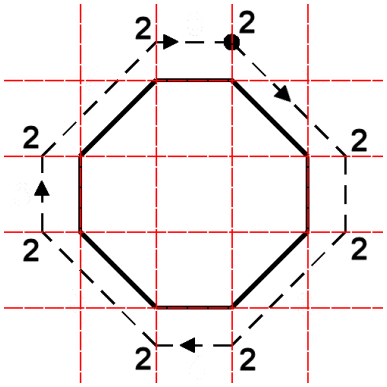

Figure 6. A8 Chain code for an octagon - starting from the dot, the chain code is 22222222 compared to 12345670 for the A8 chain code

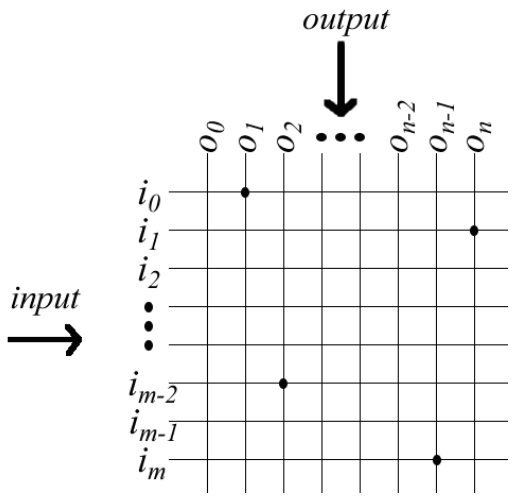

Figure 7. Showing a CMM with input vector I and output vector O. Four matrix locations are set following training I_0O_0, I_2O_{n-2}, $I_{m-1}O_0$ and I_mO_n.

1	1	2	3	3	4	5	5	6	7	7	0
0001	0001	0011	0111	0111	1111	1110	1110	1100	1000	1000	0000

Figure 8. Chain code and its Johnson Counter code representation.

1	1	2	3	3	4	5	5	6	7	7	0
1110	1110	1100	1000	1000	0000	0001	0001	0011	0111	0111	1111

Figure 9. Chain code and its logically inverted Johnson Counter code representation.

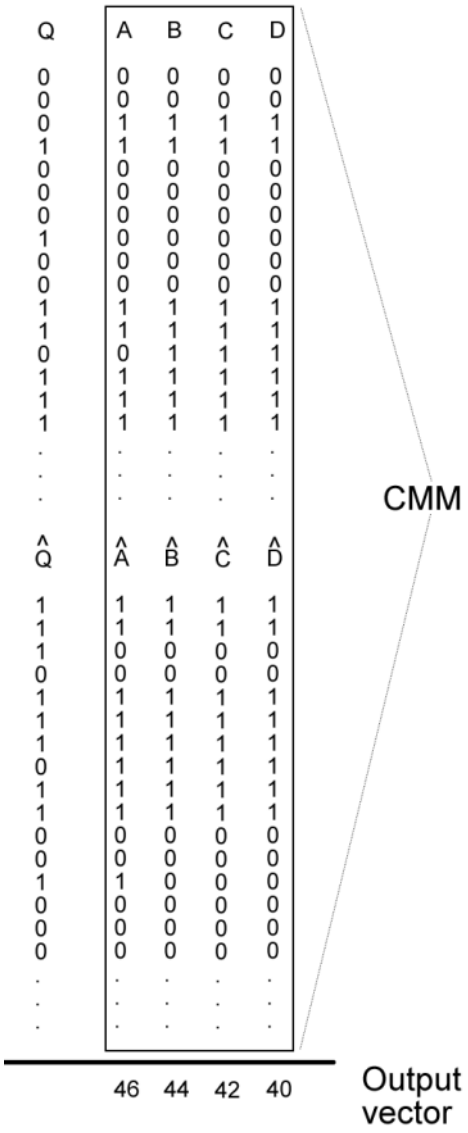

Figure 10. Diagram showing a subsection of the CMM recall for a best match. Each column in the CMM (four rightmost columns) is a stored binary chain code representing the shapes (A, B, C & D) shown in Figure 11. The leftmost column is the input vector representing the chain code of the octagon from Figure 4 and represents a concatenation of Q and $\hat{Q}$. AURA multiplies the input vector by the values in the CMM matrix columns, using the dot product and sums each column to produce the summed output vector O.

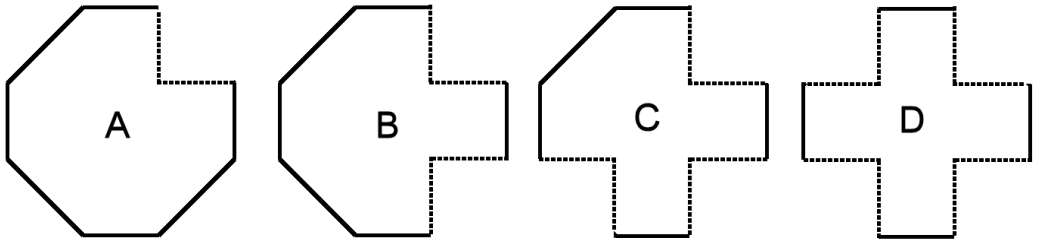

Figure 11. Showing the four shapes represented by the chain codes stored in the CMM in Figure 10. The dotted lines indicate where each of the shapes differs from the query shape in Figure 4.

Q=	1	1	2	3	3	4	5	5	6	7	7	0
	0001	0001	0011	0111	0111	1111	1110	1110	1100	1000	1000	0000
A=	2	0	2	3	3	4	5	5	6	7	7	0
	0011	0000	0011	0111	0111	1111	1110	1110	1100	1000	1000	0000
B=	2	0	2	4	0	4	5	5	6	7	7	0
	0011	0000	0011	1111	0000	1111	1110	1110	1100	1000	1000	0000
C=	2	0	2	4	0	4	6	4	6	7	7	0
	0011	0000	0011	1111	0000	1111	1100	1111	1100	1000	1000	0000
D=	2	0	2	4	0	4	6	4	6	0	6	0
	0011	0000	0011	1111	0000	1111	1100	1111	1100	0000	1100	0000

Figure 12. Showing the chain code & binary code for the query shape represented by Figure 4 (top) and the chain codes & binary codes for the four shapes in Figure 11 (A, B, C & D from top to bottom). The digits in the lighter shade indicate where each of the shapes (A, B, C & D from top to bottom) differs from Figure 4.

$\hat{Q}$=	1110	1110	1100	1000	1000	0000	0001	0001	0011	0111	0111	1111
$\hat{A}$=	1100	1111	1100	1000	1000	0000	0001	0001	0011	0111	0111	1111
$\hat{B}$=	1100	1111	1100	0000	1111	0000	0001	0001	0011	0111	0111	1111
$\hat{C}$=	1100	1111	1100	0000	1111	0000	0011	0000	0011	0111	0111	1111
$\hat{D}$=	1100	1111	1100	0000	1111	0000	0011	0000	0011	1111	0011	1111

Figure 13. Showing the inverted binary code $\hat{Q}$ for the query shape represented by Figure 4 (top) and the binary codes ($\hat{A}, \hat{B}, \hat{C}$&$\hat{D}$ from top to bottom) for the four shapes in Figure 11. The digits in the lighter shade indicate where each of the shapes (A, B, C & D from top to bottom) differs from Figure 4.

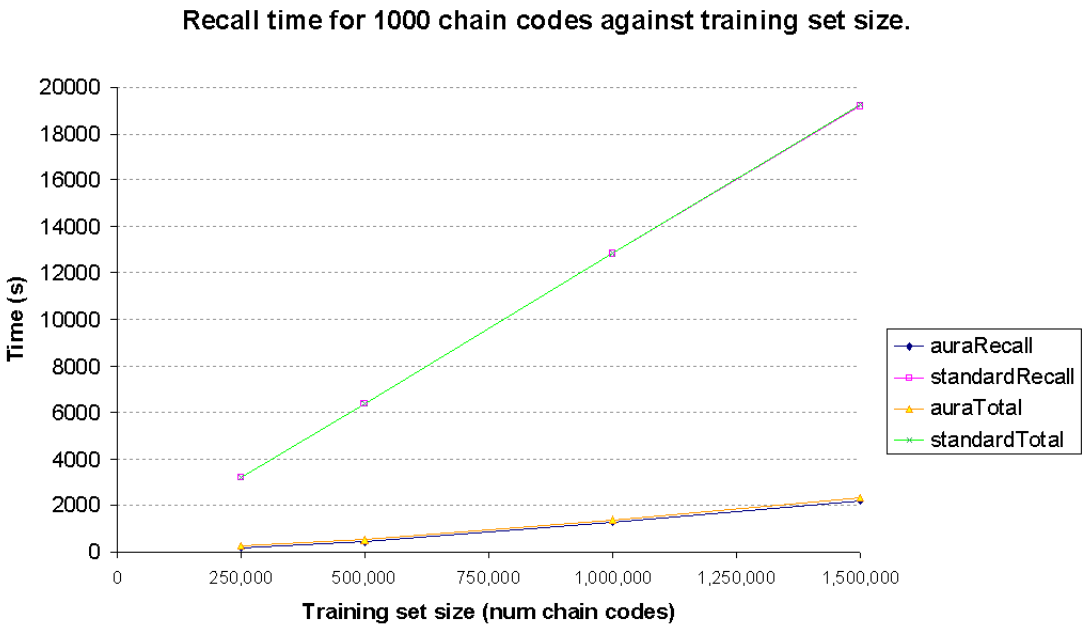

Figure 14. Graph shows the time (in seconds) to find the set of best matches (may be more than 1 with equivalent score) for the first 1000 chain codes using a stored database with between 250,000 and 1,500,000 chain codes. The chain codes are all 100 dimensional produced using a random integer generator where 0<= I <= 7. The standard approach uses integers and a lookup table to calculate the score between two chain codes. The AURA approach operates as described here. The total time includes the training time. The recall time is retrieval for 1000 chain codes only.

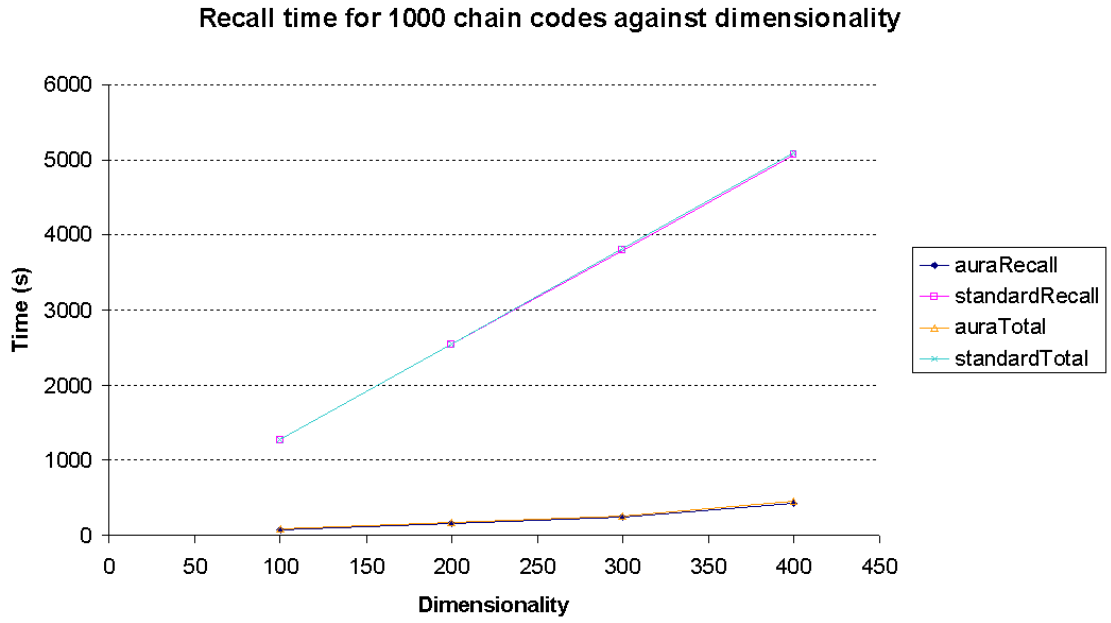

Figure 15. Graph shows the time (in seconds) to find the set of best matches (may be more than 1 with equivalent score) for the first 1000 chain codes using a stored database of 100,000 chain codes. The chain codes vary from 100 to 400 dimension and were produced using a random integer generator where 0<= I <= 7. The standard approach uses integers and a lookup table to calculate the score between two chain codes. The AURA approach operates as described here. The total time includes the training time. The recall time is retrieval for 1000 chain codes only.

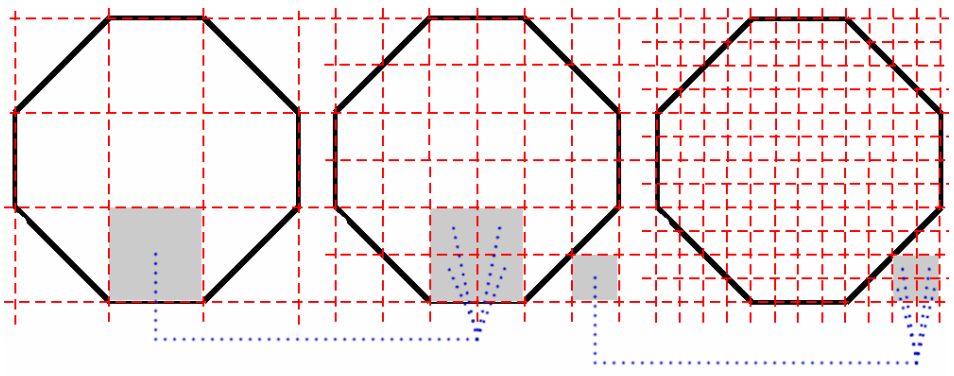

Figure 16. Figure showing a PRQ reduction (from left to right) of the grid in Figure 4. There are 36 cells in the centre shape compared to 9 cells in the leftmost shape. The chain code for the centre shape will be twice as long (the perimeter is now twice as long) as the leftmost shape. The rightmost shape has 144 cells and the perimeter is 4 times as long as the leftmost shape and twice as long as the centre shape.