Application of an Associative Memory to the Analysis of Document Fax Images

S. E. M. O'Keefe*, J. Austin†
Advanced Computer Architecture Group
Department of Computer Science
University of York
York YO1 5DD

Abstract

An essential part of image analysis is the identification of objects within the image. This paper describes the results of applying a binary associative memory neural network (ADAM), to the complex task of identifying shapes in document images. The associative memory is used to implement the generalised Hough Transform, exploiting the fast look-up ability of the associative memory to give a high-speed image analysis tool.

1 Introduction

This paper describes the application of an associative memory neural network to the task of identification of 2-D shapes in document facsimile images. The general approach is similar in some respects to the generalised Hough Transform.

The main aim of the project as a whole is to develop an architecture for the identification of the principle structural and contextual features of a document facsimile image (see figure 1 for a typical example), to enable a broad classification of the fax to be made, and to facilitate the analysis of the content of the document, in the light of the context derived at this first stage. This problem is made difficult by the wide variety of possible faxes with which any recognition system must expect to be presented, containing any of large number of textual and non-textual objects, the noise inherent in the facsimile production and transmission process, and the large size of the images being dealt with. Neural networks are an attractive proposition for this analysis task, because of their ability to deal with the noise introduced by the fax system, and because of their ability to generalise from examples. The potential for the implementation of the neural network directly in hardware, allowing the very rapid processing of data in parallel, is another attractive feature.

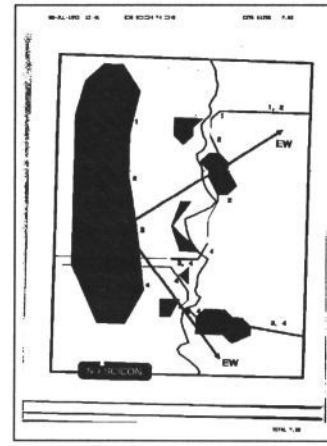


Figure 1: Document Fax Image

A primary sub-goal of the project is the *location* of structures within the image, and the *identification* of these structures. Typical examples of structures in technical documents might be paragraphs, abstracts and title blocks.

*email: sok@minster.york.ac.uk
†email: austin@minster.york.ac.uk

BMVC 1994 doi:10.5244/C.8.31

The following section describes briefly the generalised Hough Transform, to allow similarities to be drawn between it and the neural network approach described later on. Section 3 describes the method we have used. Section 4 highlights the benefit of the approach, and section 5 describes its application to a typical problem.

2 The Generalised Hough Transform

The generalised Hough Transform (GHT) [5] is an image processing technique which abstracts away from the raw input data, and attempts to provide an image representation which is both more concise and more meaningful than the raw data from which it is derived. The GHT achieves this through image parameterisation, in which a feature-detected representation of the image is mapped into its corresponding parameter space representation. The GHT does not rely on explicit boundary detection and image segmentation in order to locate objects within an image. Rather, information about parameterised objects is generated directly from the feature-detected data.

The GHT is essentially a template matching paradigm. The template used is a table containing information characterising the object to be detected, generally referred to as the R-table. The R-table is built up from an analysis of the required object. Suitable features characterising the object are selected, and for each distinct instance of these features an entry is made in the R-table. For example, a shape may be characterised by the position of edge pixels relative to some arbitrary object centre. In this case, the features are the edge elements in the edge-detected representation of the object, and the table entries are the positions of the edge elements relative to the object "centre". The "centre" is some arbitrary reference point against which the position of the object boundary points might be measured. The selection of the centre does not have any significance in terms of the object, it is purely for reference. Thus, the R-table will contain a row for each distinct edge-detected feature on the boundary, and each row will contain the offsets (from the object centre) of all the instances of this feature on the object boundary.

To detect this object in some input image, the template is matched against the image data. This matching process involves taking each point in the image, evaluating its feature-value (in this case which edge-feature is matched), and then using this feature to perform a look up in the R-table which contains the template. From the row in the R-table which matches the feature, are retrieved the set of offsets corresponding to that feature in the original object. The GHT asks the question, "Assuming the feature at some location in the image matches one of the R-table entries, what is the implied position of the object reference point in the input image?" The implied origin is calculated as the vector sum of the feature location in the image, and the recalled offset, for each offset recalled. A "vote" is accumulated at each of the resulting locations. Assuming the feature does indeed lie on the boundary of the object, then one of these votes will be cast for the correct object centre. The other votes will be "randomly" spread around the accumulator. As the image is scanned, the correctly cast votes will reinforce to form a recognisable peak in the accumulator. Detection of these peaks will indicate the location of objects within the original image. Incorrectly cast votes should not reinforce as much as the correctly cast votes, so the peaks should be distinguishable from this background noise.

This description has outlined the basic operation of the GHT. There are many enhancements and extensions possible, to deal with rotation and scaling of objects, multiple shapes, compound shapes, and so on.

The main benefits of the GHTs over most other object-detcting paradigms are

the simplicity of the algorithm, the resiliance to noise, and its ability to identify incomplete or occluded objects.

The next section describes the application of an associative memory to the object detection problem, using a paradigm which is similar in a number of respects to the GHT. However, our approach improves on the Hough Transform in a number of areas, and these will be discussed later.

3 Application of the associative memory

3.1 Overview

Section 3.1 is a general overview of the system, to guide you through the main components and their interaction. Section 3.2 discusses the operation of the system, and the associative memory used to implement the system, in more detail. The training of the system and the detection of objects are discussed separately, although both processes make use of the same associative memories.

3.1.1 Training

The purpose of the training cycle is to train an associative memory (ADAM) to associate features of an object with a label for that class of objects, and with the position of that feature in the object.

The first part of ADAM is a binary correlation matrix which learns the associations between features (small sections of the image) presented to it, and labels for each feature (i.e. it recognises arbitrary features). Each distinct feature has a unique label. Features which are similar to within some threshold are assigned the same label. No account is taken at this stage of the objects of which a feature may be a part. The effort in concentrated on the reliable training and recognition of features and recall of their associated labels.

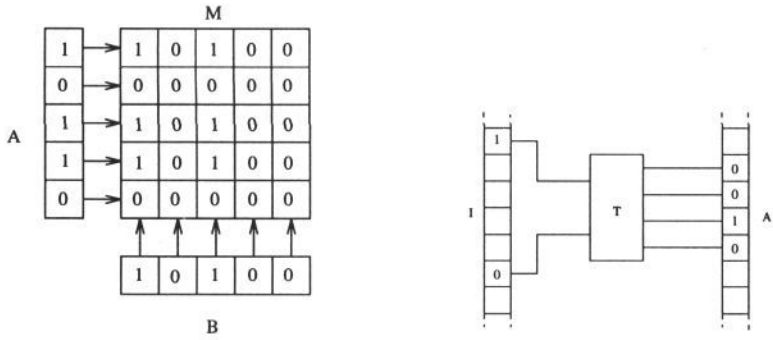
The second part of ADAM is another binary correlation matrix which takes these feature labels and associates them with the label for the object of which they are a part, and with the offset of the feature from the object's notional "centre" (the feature's position in the object). A single feature may be associated with any number of object label / object centre offset pairs, dependng on the objects which contain that feature. Simple features (e.g. line segments) will occur in many different classes of objects, more complex features will be specific to perhaps a single class. Thus this section of the memory operates as the GHT R-table.

3.1.2 Object detection

The purpose of the object detection cycle is to search the image for features which the system recognises, associate them with objects at specific locations, and accumulate the evidence for objects of different classes at locations within the image.

The first part of the system is used to recognise features. A window is scanned over the image, and if a feature is recognised by ADAM in the window, the feature label recalled by the first stage associative memory is passed to the second stage.

The second stage of the ADAM recalls all the object label / object centre offset pairs with which the feature is associated. These pairs are used to indicate which locations on the image are candidate centres of objects having that label. The possible centres are located at the vector sum of the current feature location and the recalled offset. Every time a label / offset pair is recalled, the object label is added into an accumulator grid over the image at the calculated object centre.



(a) Binary correlation matrix M, with input A and output B

(b) Tuple decoder (T). The tuple has two inputs from the original image I, giving four outputs into the array A.

Figure 2: Correlation matrices and input tupling

If several locations are recalled for a particular feature, all these locations are incremented.

The end result is an image, in the accumulator, of the likelihood that objects of particular types are at each location, expressed as the sum of the number of labels each location has received from features recognised in the image. Thresholding the accumulator returns the most likely location or locations for objects of different classes, just like in the GHT.

3.2 Operation of the associative memory

We now describe the operation of ADAM within this system in detail.

The associative memory uses binary correlation matrices to learn and recall associations between arrays of binary data. An example of such a matrix is shown in figure 2(a). A is the input array and B is the output array. The matrix M represents the outer product of A and B. All elements of M are set to 0 initially. During training, the elements of M record the intersections of the rows corresponding to the bits set to 1 in A, with the columns corresponding to the bits set to 1 in B. At these intersections the elements of the arrays are set to 1. Each subsequent pattern is written over existing patterns, forming the logical OR of the patterns.

To recall an associated pattern from the memory, the test pattern is placed in A and a matrix multiplication is performed, with the result placed in B. The result will be a non-binary array, which must be thresholded to give a binary result. The thresholding method used is that developed by Austin [3], and is referred to as N-point thresholding (sometimes termed L-max [7]). In this method, every original pattern in B has N bits set. On recall, the N elements of the array which have the highest value are set to 1, and all others are set to 0.

This basic correlation matrix can be used to recognise patterns on the input,

by associating them with a particular label. However, the number of patterns which can be taught before interference occurs between them is relatively small, particularly where patterns are similar. To overcome this capacity problem, the input to the correlation matrix is preprocessed with an orthogonalising preprocessor. This increases the hamming distance between patterns to increase the overall reliability of recall.

The preprocessing method used is tupling [6] of the input. Elements of the input pattern are mapped into (fixed size) tuples (see figure 2(b)). A simple function is applied to these tuples which returns a value, indicating the state implied by the values of the tuple elements. The tuple has a separate output for each possible state, only one of which will be at logical 1 at any time. The function used in this instance is a simple binary decoder, which interprets the pattern of bits in the tuple as a binary number, and assigns the state accordingly. The outputs of the tuple decoder forms the input to the correlation matrix. This preprocessing also has the advantage of making the input pattern more sparse than the original image, so fewer 1's are ORed into the matrix, and the input pattern has a fixed number of bits set to 1, making the behaviour of the system more predicatable.

As was noted above, ADAM is in two stages, each stage having a correlation matrix, input and output buffers, and a thresholding mechanism. The input to the second stage is the thresholded output from the first stage. No tupling of this input takes place. Training of this second stage takes place in exactly the same way as for the first stage. On recall, the output from the second stage is not N point thresholded, but simply 'Wilshaw' thresholded [9].

3.3 Training the system

To train a new 2-D pattern, an object is selected and an object centre is selected. At points around the object, where the salient features are, small windows are selected (see figure 6). At each window position ADAM is trained as follows. The training takes place in two stages. First we describe how the first correlation matrix is trained to recognise features. Then we show how the second correlation matrix is trained to recall the object label / object centre offsets.

The first correlation matrix in ADAM is trained to associate the features we wish it to recognise with a unique label, X. This recognition of features is independent of the objects of which the features may be a part. The training is carried out as described above, with a slight modification — before training the feature into the system, the first correlation matrix memory is tested to determine whether the feature (or a sufficiently similar feature) has already been taught. If it has, we proceed straight to the second part of the training cycle. If the feature is not recognised, the system generates a (random) unique label for it, teaches the feature to the recogniser, and proceeds to the second stage. The algorithm for training X into the first stage associative memory is thus:

```
For each feature you wish to train
   Position the window over the feature
   Calculate the state of each tuple, and put these in the input array
   Perform a matrix multiplication to get the raw output
   N-point threshold this output
   If a valid label is produced by thresholding (feature recognised)
      Train second stage
   Else (feature not recognised)
      Generate a new unique label, and place this in the output array
      Train correlation matrix to associate input and output arrays
```

Train second stage

The second stage of the system associates a particular feature, which has been trained into the first correlation matrix, with an object label / object centre offset pair. The object label is a unique label for the class of objects which the training object belongs to, and is assigned by the operator. The object label is N-point coded, that is, exactly N bits in the label are set to 1. Using this N-point coding scheme, we can represent a very large number of classes very compactly. For example, if K (the size of the label) is 40 bits, and N (the number of bits set) is 3, we can represent ${}^{K}C_{N}$, or 9,880 different classes. We need to learn the object class label so that we can recall which classes of objects the feature belongs to.

The object centre offset is the vector displacement of the object's notional centre from the feature we have trained into the first correlation matrix. To calculate the offset vector, for each feature we assume a rectangular grid of points overlays the image, centred on the current feature and extending over most of the image (we do not cover the whole image for reasons of economy). The fineness of the grid is set before training the system. Given the positions of feature and object centre, we approximate the position of the centre with the closest grid point. We need to learn the centre offset so that we know where the feature occurs in the object, relative to its centre.

Because the same feature may occur a number of times in the same object (particularly simple features such as line elements), and may occur in many different classes of objects, we need to be able to correctly associate together the object labels and object centre offsets. Because of the way in which the correlation matrices are taught, we can recall many such pairs in parallel, but since the labels and offsets have been ORed together into the memory, we may find it difficult to determine which label to associate with which offset. We overcome this problem by binding the two variables together, by forming their tensor product [8]. The second stage of the system associates the feature label with this tensor product, and on recall, we can successfully unbind the variables to give the object label / object centre offset pairs.

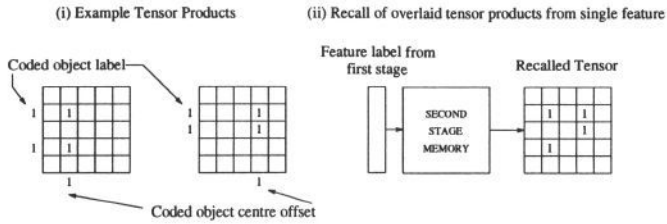


Figure 3: Result of training more than one tensor product for a feature. (i) Two example tensor products, for different object label / object centre offset pairs. (ii) Recall of the two tensor products overlaid

The result of this recall of multiple tensor products is illustrated in figure 3. Figure 3(i) shows two tensor products formed from different pairs of object labels and centre offsets. If both these tensor products are associated in the second stage memory with the same feature (indicating that the feature could be a part of either of two different objects), then when the second stage is tested, we recall the two tensor products overlaid (figure 3(ii)). From this representation, we can extract the appropriate pairs of values for the variables.

3.4 Detecting objects in images

The algorithm for detecting objects in images is straightforward. A small window is scanned over the image, at some small spatial interval. At each position, the first stage matrix is tested with the part of the image in the window. If the feature is succesfully recognised (the thresholding returns a valid feature label), the recalled feature label is used to recall the object label / object centre offset pairs from the second correlation matrix. (In a truly parallel implementation, the image would be processed in parallel, rather than serially.) The output from the second correlation matrix consists of one or more tensor products ORed together (as in figure 3) corresponding to the different positions of the feature within the object or objects.

To record the results of the processing as the window is scanned over the image, we create an accumulator grid. This grid covers the whole image, at the same spatial interval as the window is scanned, i.e. there is a grid point for each location tested, to record the results of the test at that location. Each point in the grid can hold the object labels in their N-point encoded form.

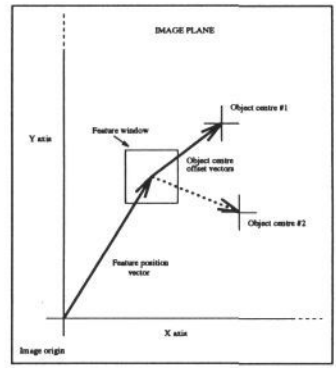


Figure 4: Calculation of the object centre positions from the feature position and the centre offsets, assuming two offsets have been recalled for this feature

The matrix recalled from the second stage corrrelation matrix contains object labels and their corresponding centre offsets. In the simple example in figure 3, it can be seen that each offset is encoded as a single bit. The labels corresponding to each offset (each bit set to 1) may therefore be recovered simply be taking the column in the matrix corresponding to that position. In other words, for each column which contains a label, the offset appropriate to that label is given by the column's position. (A more sophisticated coding of the offset would require a more sophisticated recovery of the object label / centre offset pairs, but the principle would be the same.) From these offsets (the positions of object centres relative to the position of the feature) the positions of the possible object centres in the image are calculated (figure 4).

At each of these positions, the corresponding label from the tensor product is added into the accumulator. (Because of the errors caused by approximating the object centre position on the grid, and the finite scan intervals, the labels are summed into a block of points rather than a single point. The centre of the block is weighted more heavily than the edges, according to the mask shown in figure 5.) Thus, as the window is scanned over the image, more and more labels are added into the accumulator. Note this essential difference between our system and the GHT. We are accumulating *labels*, corresponding to the object class. Labels for *any* class of object may be added into the accumulator, and thus many classes of objects may be searched for in parallel. Because the labels are N-point encoded, the accumulated evidence at each point may be N-point thresholded to reveal the dominant class at that point. The GHT only accumulates votes at each location, as the R-table

+1	+1	+1
+1	+2	+1
+1	+1	+1

Figure 5: Object label increment mask

for a specific shape is used as a template.

The end result is an accumulator whose contents describe the distribution of objects over the image. As with the GHT, the peaks in the accumulator indicate the most likely positions for objects of that class. However, our system provides a much richer description of the image, by accumulating labels for a large number of possible object classes in parallel. The results of N-point thresholding these accumulated labels is a measure of the *confidence* that a particular object is at a particular location.

In summary, the algorithm for object detection is as follows:

```
For each location on the image
   Take a window on the image
   Tuple the contents of the window
   Test the first stage correlation matrix
   If a feature is recognised
      Test the second stage with feature label, and recall tensor product
      For each object label / centre offset pair in the tensor product
         Calculate the object position in the image
      Add the label into the accumulator at this position
N-point threshold the accumulator to recover the object positions
```

3.5 Relationship to ADAM Associative Memory

The associative memory application described here has been developed using the the Advanced Disributed Associative Memory (ADAM) software library [3], with minor extensions to the basic ADAM mode of operation. This software was specifically designed for image analysis, and has since been extended as the result of further work on the analysis of infra-red ground images [1], [2]. Because the associative memory uses the ADAM library, it will be possible to port it to hardware (C-NNAP) specifically designed to implement vision problems using ADAM functions [4].

4 The benefits of using ADAM

Similarities exist between the basic mode of operation of the associative memory described above (and the ADAM software) and the GHT. In its simplest form, an ADAM accumulates evidence for the appropriate output, given a particular input, in the raw output from the correlation matrices. This accumulated evidence is then thresholded to determine what the appropriate response is.

Compare this with operation of the Hough Transform, which accumulates evidence for the existence of particular objects in an image. This accumulated evidence is then thresholded to determine the appropriate response.

Given this similarity in the basic function of the Hough Transform and ADAM, we have attempted to produce an image analysis system which performs the same basic task as the GHT, that of detecting objects in images. There are distinct advantages in using the ADAM approach to the problem. In image analysis, particularly in the analysis of large images such as those produced by facsimile machines, speed of processing is an important factor. The correlation matrices at the centre of ADAM, combined with the orthogonalising pre-processing and the N-point coding and thresholding, enable us to perform the template matching in a way which is inherently parallel. The binding of object labels to object centre offsets using a tensor product form means that these pairs of values can be trained

into the same memory, and then recalled in one step. The values of the variables are only unbound later, as needed.

Both our system and the GHT look for features in the image at each point, get the centre offsets associated with these features, and increment an accumulator. The GHT associates the features with the offsets by indexing into a table using the feature. The speed with which the GHT can do this look-up will depend on the number of features trained (and hance the size of the table). If n features are trained, then a binary-tree implementation of the table would take $O(log_2 n)$ to search. By comparison, ADAM does the association between features and centre offsets in a fixed time, independent of the number of features which have been taught.

The parallelism extends to the implementation of the ADAM. The ADAM library has been ported to specialised parallel hardware (C-NNAP). Hence, image processing using ADAM can be done using this hardware with the associated gain in speed.

One may ask why we don't just use a single ADAM to convolve the image of the whole shape, instead of using the complex method described here. The approach described here is less susceptible to interference from clutter, and can deal with objects which are very large. If a single ADAM is used and a large window is used, then there is a high likelihood of clutter within the image affecting recognition. Using this approach each feature ADAM sees is small, is less likely to be obscured by other structures, and has a better signal to noise ratio.

5 Initial results

As an example, to demonstrate the viability of the approach, the system described above has been applied to the image shown in figure 1. Figure 6 shows an object in the image. The system has been trained on this example of a single class of objects. The object is characterised by the features marked with the small squares on this object. This object is assigned the notional centre marked by the cross "+". The training window is 21 by 21 pixels, the offset of the object centre from the feature centre is recorded on a 21 by 21 grid, with a grid point spacing of ten pixels.

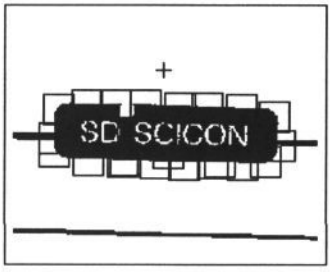


Figure 6: Training object. Object centre is marked with a "+", and trained features are marked with boxes. (Figure is not to scale)

After training, the image was scanned at intervals of five pixels (both horizontal and vertical intervals). At each point the first stage was tested to see if the window contained a recognisable feature. If a feature was recognised, the second stage was tested to recall the object label / object centre offset pairs. The labels were summed into the accumulator array at the positions indicated by the feature position and centre offset.

The results from scanning the image are shown in figure 7. Each small cross indicate a position which was recalled as a candidate object centre (there is no indication as to how many votes each location received). As can be seen, these candidate centres are spread widely over the image, indicating that features associated with this particular object also occur in a large number of other objects. Because only a single class has been taught, only a single class label has been recalled, and we are able therefore to do a simple thresholding of the image. On

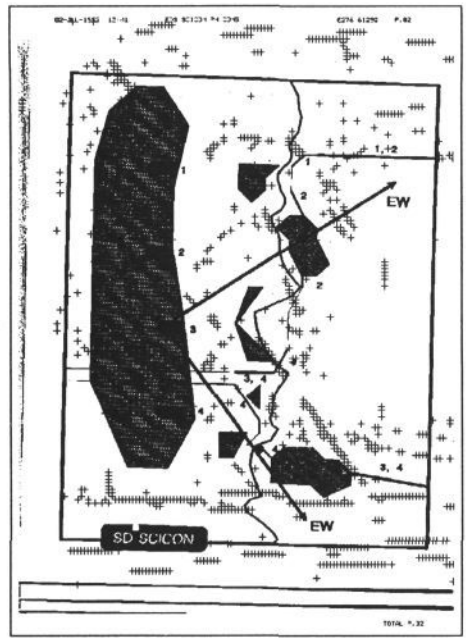


Figure 7: Fax image after processing. Candidate object centres are marked by crosses

thresholding of the accumulator, the correct object centre was returned.

Figure 8 is a surface plot of the contents of the accumulator. The bottom of the original image corresponds to the near edge of the surface, labelled "X". On the z-axis is plotted the number of counts of the label for the object we are are searching for, at each point in the accumulator. It can be seen that there are many small peaks in the accumulator space, but that the principle peak is centred correctly at the orginal position of the object reference point. The spread of this principle peak is small, indicating that the location of the object has been well defined despite the relative coarsness with which object centre offsets were recorded, and with which the image was scanned.

6 Summary

This paper has described an initial implementation of an associative memory for the detection of complex objects in complex images, by implementing a version of the GHT using the ADAM Associative Memory library. These initial results show that the approach is valid for this difficult task. By exploiting the strengths of the ADAM system, in particular the parallel look-up made possible by the associative memories, we can expect substantial gains over serial implementations.

7 Acknowledgements

During this work Simon O'Keefe was supported by a PhD CASE award from the Government Communication HQ and the EPSRC. Thanks are due to Aaron

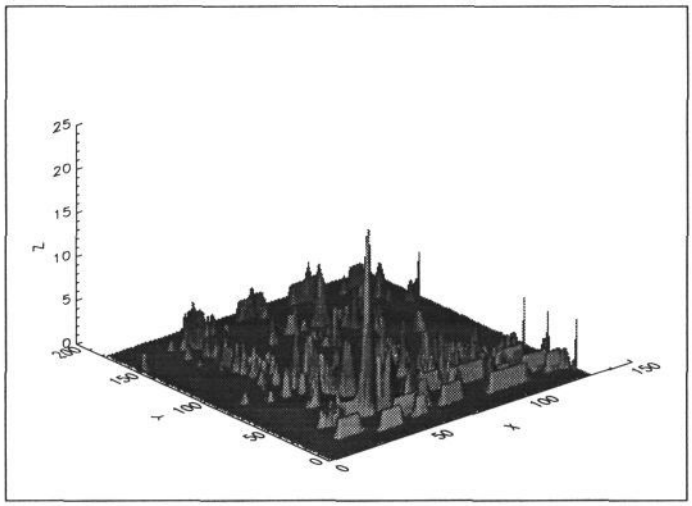


Figure 8: Surface plot of contents of accumulator. The X and Y axes are the X and Y axes of the original image. The Z axis shows the confidence is the existence of an object centre at each point (in this case, the number of object labels in the accumulator in at each point)

Turner and Steve Buckle for their help with the ADAM software.

References

[1] J Austin, M Brown, I Kelly, and S Buckle. Adam neural networks for parallel vision. In *JFIT Technical Conference 1993*, pages 173–180, 1993.

[2] J. Austin and S Buckle. The practical application of binary neural networks. In *Proc. of the UNICOM seminar on Adaptive computing and information processing.*, 1994. To be published.

[3] J. Austin and T.J. Stonham. The adam associative memory. Technical report, Dept. of Computer Science, University of York, 1987.

[4] J Austin, A Turner, S Buckle, M Brown, A Moulds, and Rick Pack. The cellular neural network associative processor, C-NNAP. *IEEE Computer*, 1994. Accepted for publication.

[5] D. H. Ballard. Generalising the hough transform to detect arbitrary shapes. *Pattern Recognition*, 12:111–122, 1981.

[6] W. W. Bledsoe and I. Browning. Pattern recognition and reading by machine. *Proc. Joint Comp. Conference*, pages 255–232, 1959.

[7] D. Casasent and B. Telfer. High capacity pattern recognition associative processors. *Neural Networks*, 4(5):687–698, 1992.

[8] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216, 1990. Artif. Intell. (Netherlands).

[9] D. J. Wilshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, page 222, June 1962.