A NOVEL APPROACH TO BINARY NEIGHBOURHOOD OPERATIONS UTILISING A HARDWARE BASED RBF NETWORK

Peter Gaughran
Dept. of Computer Science
National University of Ireland, Maynooth
Ireland
peterg@cs.may.ie

ABSTRACT
In this paper we present a novel approach employing a radial basis function (RBF) network to implement binary neighbourhood operations such as dilation, erosion, opening and closing. The network used is a parallel electronic hardware device (ZISC Neuroboard 576) designed by IBM Microelectronics Division. Despite increases in serial processor speed, neighbourhood operations have proven to be computationally intensive, but by utilising a multi-purpose, generalised parallel device computation overheads can be significantly reduced.

KEY WORDS
Machine learning, neighbourhood operations, RBFs, parallel.

1. Introduction
Prior to the 1980s, image processing relied heavily on specifically produced hardware to implement the neighbourhood operations, and in circumstances where none were available, well-known algorithms for standard serial execution computers were used [1]. Subsequently, algorithms involving ‘a region of interest per line technique’ or pointer-sorting were used, as well as run-length representation of images to achieve some of the standard binary neighbourhood operations (dilation, erosion, medial axis transform, propagation, etc.)[2].

Interval coding has also been used for other neighbourhood operations such as skeletonisation[3], however none of these algorithms have made use of parallel techniques. In 1992, recursive neighbourhood operations were efficiently implemented on a 1-D parallel computer known as SYMPATI-2 [4], and also a parallel CPU based implementation by van Boomguard et al [5]. Recently, a model in 2003 using an extreme vertices model has been realised [6]. Given that the nature of the problem and its solution involves input image pixels versus output image pixels and the correspondence of this operation to pattern recognition, the problem is ideally suited to artificial neural networks. The network used is a parallel electronic hardware board.

2. The ZISC Neuroboard 576
The ZISC (zero instruction set computer) is a digital chip with 64 8-bit inputs and 36 radial basis function neurons. Many of these chips can be connected together to create networks of (theoretically) arbitrary size, and the particular model used in our implementation consisting of 16 chips in total on-board. An input vector V is compared to a stored prototype vector P for each neuron. Each neuron then gives a 14-bit output, resulting in 16,384 possible categories. The classification is calculated based on a distance value from the input vector V to the stored prototype P using one of two methods:

\[ \text{(1) Distance} = \text{sum of abs}(V_i - P_i), \ i = 1..64 \]
\[ \text{(2) Distance} = \text{Max}((V_i - P_i)), \ i=1..64. \]

Despite the fact that the board itself operates on a 16Mhz ISA bus, this is not a limiting factor to the overall performance of the card; an input vector to the card is an 8-bit serial input composed of 64 elements, with loading of the elements taking 3.5 microseconds, and another 0.5 microseconds for classification. The neurons may store a prototype each, and evaluate their own particular input via one of the two distance calculations described above. The influence field (and therefore other neurons) is also consulted, and the neuron may then fire, see Fig. 1. Each neuron may communicate with another via an on board bus, and their collective input is sent to a learning and decision logic centre before the final decision output is given [7].
2.1 Description of operations

In order to describe the operations, it is necessary to define the neighbourhoods first. Traditionally, the neighbourhoods or masks used to alter the image are of size 3x3, see Fig. 2. Only masks as large as 7x7 may be used by a single Neuroboard, however, given its maximum category classification of 16384 prototypes. Even though the maximum allowable input is a vector size of 64, it is not possible to allow for larger sized masks with only one of these networks, as for 9x9 masks, for example, 262144 categories would be needed. As it is possible to daisychain the ZISC Neuroboards, larger masks may then be used. An output pixel is generated on the basis of the pixel at the corresponding position in the input image and on the basis of its neighbouring pixels, affected by means of convolution.

\[ A \odot B = \{ x | \forall b \in B, x + b \in A \} \]

As critical connectivity is not a factor when using erosion, it is not a consideration if the object in the image breaks into two or more parts. Dilation, then, may be described as \( A \bigodot B \), and affects the reverse of the erosion operation. If \( f(A) \) is the resultant image when \( A \) has been eroded by \( B \), generally speaking it is not possible to recover \( A \)'s initial pixel values by dilating the eroded values of \( f(A) \). This new, dilated version of \( f(A) \) is only partly reconstructed; it has fewer details and is simpler but has details that might be deemed essential to the structure of \( A \). This operation is known as opening. The closing of an image \( A \) occurs when \( A \) is dilated by \( B \), and then eroded.

**Pseudocode algorithms for each of the operations implemented**

- Erosion
  
  FOR \( p_i, i = 0 \) to \( n \), where \( p \) is a pixel in the image
  IF \( p_i \) is an element of the object and all of its neighbours are elements of the object,
  COPY to the output image

- Dilation
  
  FOR \( p_i, i = 0 \) to \( n \), where \( p \) is a pixel in the image
  IF \( p_i \) is an element of the object,
  CHANGE all of its neighbours to be elements of the object.

- Opening
  
  Perform DILATION operation, then
  Perform EROSION operation

- Closing
  
  FOR \( p_i, i = 0 \) to \( n \), where \( p \) is a pixel in the image
  Perform EROSION operation, then
  Perform DILATION operation

2.2 Learning

Given the binary nature of the images, this is a matter of training the neuroboard with all of the potential inputs for the relevant mask size. For example, in (i) 3x3 and (ii) 5x5 masks, respectively, there are

\[ i. 2^3 \times 2^3 = 2^9 \text{ or } 512 \]
\[ ii. 2^5 \times 2^5 = 2^{10} \text{ or } 33554432 \]

patterns to be recognised. It is possible to merely train the board on each input and its relevant output using the conventional well-known algorithms for the
neighbourhood operations to ensure a 100% classification result each time, and indeed, given the rapid loading time per vector even the permissible largest mask of 7x7 takes approximately 0.057344 seconds under optimal operating conditions. Once the board is configured for each particular mask size, it is possible to save its current state and re-load at a later date, so none of the training data is lost. The training images are stored as binary vectors and the output images are rendered from the resultant output binary vectors.

It is also possible to train the board for larger outputs than it was trained on given the similarity of many of the inputs and outputs by extending each neuron’s radius of generalisation, that is, to allow local inputs from the input space to map to the equivalent output category. As the neuroboard is composed of radial basis function neurons with digital Gaussian approximations, the radius may be set as small as 1, or as large as 16384. If there are only 2 categories, for example choosing to add/not add a pixel, or to remove/not remove a pixel, it is possible to generalise based on the inputs to produce the desired output, but the training time increases considerably. Previously, the operations known as opening and closing required two passes of different operations, as described in section 3. By training the Neuroboard in the appropriate manner, only one pass of the image data is necessary to effect the same result, thereby saving time and computation.

2.3 Experimental Results

The neighbourhood operations presented were implemented on the ZISC Neuroboard 576 using the latest API, version 1.51, in C. Also presented are serial implementations for means of comparison, also implemented in C. The images were loaded as binary bitmaps, and the input vector representing each neighbourhood was sent to the board. A table giving the time taken for various standard image sizes (640x480 up to 2048x1536) for each of the operations is presented, and an original binary image and its eroded/dilated counterparts are also presented here.

Fig. 3 Original image, left & eroded image, right.

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Time taken (in seconds) for each neighbourhood operation respectively</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>1.219848</td>
</tr>
<tr>
<td>800x600</td>
<td>1.908808</td>
</tr>
<tr>
<td>1024x768</td>
<td>3.1314</td>
</tr>
<tr>
<td>2048x1536</td>
<td>12.554248</td>
</tr>
</tbody>
</table>

Fig. 4 Dilated image, top left, opened image, top right, and closed image, bottom.

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Time taken (in seconds) for each neighbourhood operation respectively</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>0.64512</td>
</tr>
<tr>
<td>800x600</td>
<td>1.002788</td>
</tr>
<tr>
<td>1024x768</td>
<td>1.6515072</td>
</tr>
<tr>
<td>2048x1536</td>
<td>7.2060288</td>
</tr>
</tbody>
</table>

Fig. 5 Table showing results for 3x3 to 7x7 sized neighbourhoods on the ZISC Neuroboard

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Time taken (in seconds) for each neighbourhood operation respectively</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>2.332871</td>
</tr>
<tr>
<td>800x600</td>
<td>3.694364</td>
</tr>
<tr>
<td>1024x768</td>
<td>6.229548</td>
</tr>
<tr>
<td>2048x1536</td>
<td>24.56459</td>
</tr>
</tbody>
</table>

Fig. 6 Table showing results for 3x3 sized neighbourhoods on a standard desktop PC (2Ghz, 1024Mb RAM), using C code implementation

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Time taken (in seconds) for each neighbourhood operation respectively</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>0.64512</td>
</tr>
<tr>
<td>800x600</td>
<td>1.002788</td>
</tr>
<tr>
<td>1024x768</td>
<td>1.6515072</td>
</tr>
<tr>
<td>2048x1536</td>
<td>7.2060288</td>
</tr>
</tbody>
</table>

Fig. 7 Table showing results for 7x7 sized neighbourhoods on a standard desktop PC (2Ghz, 1024Mb RAM), using C code implementation
3. Conclusion

As the method used involves a parallel hardware board, direct comparisons with existing serial or recursive algorithms are difficult. While specialised parallel hardware exists to perform these operations, the device employed here is general purpose and may be used for many other applications [8,9]. As described in section 2.2, the initial training time is negligible under optimal operating conditions, and is guaranteed to give the correct solution. It is interesting to note that for 3x3 sized neighbourhoods, the ZISC board is outperformed by the modern desktop PC used in these experiments, yet for the larger neighbourhood size of 7x7 the ZISC outperforms the PC. Due to the default input vector size of 64, all neighbourhoods of size 7x7 or less take the same time to compute (system loads permitting); this is not true of a serial machine. The particular model of ZISC Neuroboard (the Zisc 576) was produced in 1998; given the improvements in silicon chip manufacture, it is conceivable that a similar board produced today would yield further computational boosts.

4. Acknowledgements

Special thanks to George Mitchell and the Department of Computer Science in NUI Maynooth.

References:


