

Arithmetic on a Parallel Computer: Perception Versus Logic*

JAMES A. ANDERSON

Department of Cognitive and Linguistic Sciences, Brown University, Providence, RI 02912, U.S.A., e-mail: james_anderson@brown.edu

**Some parts of this article were first described in a paper presented at the Proceedings of the First IEEE International Conference on Cognitive Informatics, Los Alamitos, CA, 2002.*

(Received: 24 January, 2003; in final form: 7 April, 2003)

Abstract. This article discusses the properties of a controllable, flexible, hybrid parallel computing architecture that potentially merges pattern recognition and arithmetic. Humans perform integer arithmetic in a fundamentally different way than logic-based computers. Even though the human approach to arithmetic is both slow and inaccurate it can have substantial advantages when useful approximations (“intuition”) are more valuable than high precision. Such a computational strategy may be particularly useful when computers based on nanocomponents become feasible because it offers a way to make use of the potential power of these massively parallel systems. Because the system architecture is inspired by neuroscience and is applied to cognitive problems, occasional mention is made of experimental data from both fields when appropriate.

Key words: arithmetic, attractor, cognitive science, computation, data representation, network of networks, neural network.

1. Introduction

I have been interested for a while in the problem of “hybrid” computation, that is, building a system that uses both logic based and associative computing. An ultimate goal of such a project might be a physically hybrid computer architecture combining the best properties of the classic, logic-based digital computer, and a largely associative, perceptually-based neural net computer. The human brain seems to have evolved the early, crude, but intermittently successful, beginnings of such a system. Artificial systems built along these lines might be of considerable practical interest. A version of this architecture as applied to elementary arithmetic operations is available in a working prototype computer program, the Neural Network Program Program (NNPP), available, with documentation, from the author (james_anderson@brown.edu).

The nervous system treads a delicate but productive balance between continuous, “analog,” perception-based computation and discrete, logic-based computation. Integer arithmetic is the quintessential abstract computational function. A logic-based computational device, for example, the familiar digital computer, does integer arithmetic quickly and accurately. From extensive experimentation over many years a great deal is known about how humans perform integer arithmetic: humans perform arithmetic slowly and badly (see Anderson, 1998, for a review and further discussion.) Evidence strongly suggests that

humans perform integer arithmetic in a fundamentally different way than do logic-based computers, but the way humans perform arithmetic can have substantial advantages when useful approximations (“intuition”) are more valuable than high precision.

These observations led to the development of a more formal functional computer model of simple human arithmetic computation based on a nonlinear neural network. Such a model is not merely an academic curiosity from cognitive science. When used for a more general parallel computing architecture, it may have practical applications, for example, building systems with artificial mathematical intuition.

In summary

- human arithmetic computation can be modeled by using a neural net-based dynamical system with discrete attractors corresponding to integers.
- the proposed technique used allows flexible, simple, and accurate programming of certain classes of arithmetic operations.
- this technique is slow and inaccurate for purely arithmetic computation. However, when such a controllable network is combined with the pattern recognition abilities of a neural network, a powerful fully parallel hybrid computational architecture may emerge.
- such a computational architecture may be realizable in hardware when computers based on nanocomponents or using massive parallelism becomes feasible.

2. Discrete vs. Continuous

Warren McCulloch and Walter Pitts in their famous 1943 paper, *A logical calculus of the ideas immanent in nervous activity*, suggested that “neural events and the relations among them can be treated by means of propositional logic” (McCulloch and Pitts, 1943/1965, p. 19). They were led to this conclusion from their model of the neuron that approximated it as a binary device that computed logic functions based on its inputs. However, we now know that neurons act much more like small, sophisticated, and powerful analog computers. As Warren McCulloch himself put it a few years later in his essay, *Why the mind is in the head*, (p. 73. McCulloch, 1951/1965)

sense organs and effectors are analogical. For example, the eye and the ear report the continuous variable of intensity by . . . impulses the logarithm of whose frequency approximates the intensity. But this process is carried to extremes in our appreciation of the world in pairs of opposites. As Alcmaeon, the first of experimental neurophysiologists, so well observed, “the majority of things human are two – white black, sweet–bitter, good–bad, great–small.”

If the ‘logic,’ that is, the discrete states, is not present in the neurons themselves, then where is it? A recent popular suggestion is that it is in discrete attractor states arising from an underlying continuous dynamical system. Such a system forms a hybrid computational architecture that combines valuable aspects of both discrete and continuous systems. In the next sections, we will describe how such a system is consistent with human performance on one set of elementary arithmetic tasks and how abstractions of it can be generalized to wider domains.

3. Arithmetic in Humans

When a computer multiplies two single digit integers, it performs a sequence of formal operations based on logic, the definitions of integers and the rules of binary arithmetic. Humans seem to do multiplication quite differently, using a combination of estimation and memory. The human algorithm might be formalized as something like ‘Choose the product number (that is, a number that is the answer to *some* multiplication problem) that is about the right size.’ For example, the most common error for 9×6 is 56. More careful inspection of the experimental data indicates also a complex associative structure working along with estimation, for example, errors are frequently ‘off by one’ multiples of 6 or of 9, for example, 48 or 63. These effects do not seem to arise because of errors in the application of formal logic to the computation.

The error rate for elementary multiplication facts in college-educated adults is high, as high as 8% in some studies (Graham, 1987). Such a high rate is remarkable considering that several years are devoted to practicing arithmetic facts in elementary school, at a time when learning of other complex material – language, for example – is fast and accurate. However, there are also some positive aspects to such an error prone algorithm. For example, errors in elementary arithmetic are usually “close” to the correct answer. Sometimes being ‘close’ is good enough to do the job and is often much better than the kind of gross errors that malfunctioning computers are notorious for.

Attempts by cognitive scientists to model human number performance have virtually always assumed the presence of an important ‘perceptual’ part to the internal human number representation. In many experiments, numbers act more like ‘weights’ or ‘light intensities’ than abstract quantities. There is also evidence from numerous sources (see Hadamard, 1949 for a classic description) that higher mathematics as actually done by mathematicians or physicists is often more perceptual than abstract. The classic theorem–proof process is primarily used to check for errors and as a way of convincing others that results are correct. These ill-defined but widely used sensory and perceptual aspects of mathematics as it is actually practiced often go by the name ‘mathematical intuition.’

4. A Nonlinear Neural Network

This paper is not the place to review neural network theory. Many introductions are available including my own text, *An Introduction to Neural Networks* (Anderson, 1995). The basic idea is that there are a large number of interconnected elementary computing units that operate in parallel. The computing units themselves are loosely based on the properties of neurons, and act as integrators of their inputs. Their connections are abstractions of synapses. Figure 1 shows the generic two stage **neural network computing unit**. The unit receives continuous valued inputs from other units over a potentially large number of input lines. The connections between ‘presynaptic’ and ‘postsynaptic’ units typically have continuous values. The first stage takes the inner product between the pattern of input activations, represented as a state vector, \mathbf{f} , and the set of weights, \mathbf{w}_i , that is, $[\mathbf{f}, \mathbf{w}_i]$. The second stage performs some nonlinear operation on the output of the first stage, most often

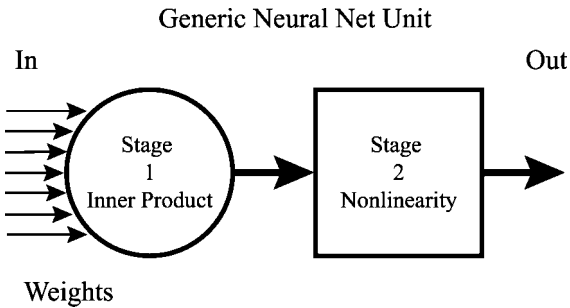
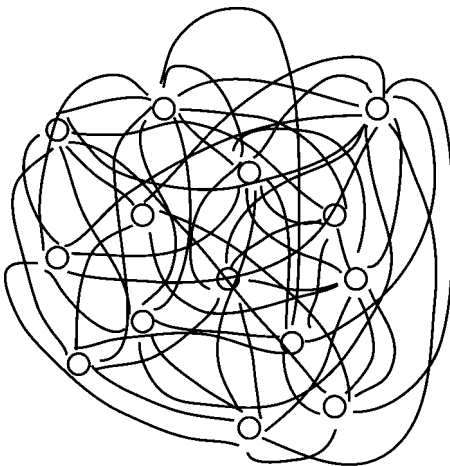


Figure 1. Generic neural network computing unit.

using a monotonic sigmoidal function. Figure 1, showing the generic neural net computing unit can be compared with Figure 9, showing the basic computing unit of the ‘Network of Networks’ model discussed in Section 7.

Neural networks form natural pattern recognizers. A complex low-level input pattern can become associated with a particular class name, for example a particular shape represented as an input pattern of unit activations can become associated with a letter. However, pattern recognition involves an implicit nonlinearity. A whole range of items – different handwritten digits, for example – is given the same output classification. It is possible to perform this classification operation several ways. The way of most interest to us involves the use of what are called **attractor networks**.

Typical attractor networks are the well-known Hopfield networks and the Brain State in a Box (BSB) network (Anderson, 1993). These are both single layer **recurrent networks**, where a set of units connects to itself, as shown in Figure 2. In operation, the system



**Recurrent Network:
A Layer Connects to Itself**

Figure 2. A single layer nonlinear recurrent network.

state is started at a particular point in state space and as time progresses the system state evolves in time. It can be shown that attractor networks as they evolve are minimizing a system energy function (Lyapunov function). The system state will eventually reach a stable state in an energy minimum. The long-term behavior of these networks is characterized by these stable **attractor** states. All the points in state space that end in the same attractor are referred to as the **basin of attraction** for that attractor. Both Hopfield and BSB networks have single points as stable attractor states, that is, **point attractors**, but more complex attractor behaviors such as limit cycles are possible for other classes of recurrent networks.

Learning algorithms allow the stable states to become associated with categories or ‘memories.’ One mode of operation of such a network would be to have many different examples of the same handwritten letter, say, end in a common stable point which could then serve as the prototype pattern for an entire category. Several models for human concept formation in the cognitive literature have taken this form.

The network used in the NNPP program is the BSB network, a one layer recurrent net with a simple limiting (i.e. clipping) nonlinearity in the second stage of the units. A detailed discussion of BSB can be found in Ch. 15 of Anderson (1995). Briefly, the BSB algorithm can be analyzed as a feedback system. If the connection matrix connecting the set of units to itself is \mathbf{A} , the current state vector at time step t is $\mathbf{x}(t)$, the original input to the network is $\mathbf{x}(0)$, and the state of the system decays with a decay parameter γ (between 0 and 1), then the next state of the system, $\mathbf{x}(t + 1)$ is given by

$$\mathbf{x}(t + 1) = \text{LIMIT}(\alpha\mathbf{A}\mathbf{x}(t) + \gamma\mathbf{x}(t)\delta\mathbf{x}(0)).$$

Weighting constants are α , δ , and γ . The qualitative behavior of the system is insensitive to the exact values of these parameters. The LIMIT function clips the elements of the state vector so if an element value exceeds in magnitude the upper or lower limit, the value of the element is replaced by the value of the limit.

This system can be analyzed as a feedback system, with dynamics related to the ‘power method’ used for numerical extraction of eigenvectors. The low energy points (attractors) of this system can be shown to be generally located at the ‘corners’ of the box of limits. Standard neural network learning algorithms work nicely with this network to set the connection strengths; in the NNPP simulations we used the LMS (Least Mean Squares) supervised error correction algorithm.

We can define **network computation** with an attractor network as the following sequence of steps:

- The network has built an attractor structure through previous learning.
- Input data combined with the program for the computation gives rise to a starting point in state space. For a neural network, the conversion of input data into a state vector is called the **data representation**.
- The network state evolves from this starting point.
- The final network stable state gives the answer to the computation.

5. The Data Representation for Number

The most difficult problem in the application of neural networks usually is converting the input data into the state vectors that can be manipulated by network dynamics. Data representation usually has more influence on system performance than the type of learning algorithm or the form of the network. There are few explicit rules that determine what the best data representations are for a particular problem. For example, there is a constant tension between accuracy and generalization since generalization to be useful requires an ‘appropriate’ response to an unlearned pattern. ‘Good’ generalization is, therefore, critically dependent on the details of the specific application and the data representation. To be useful in practice, generalization should also be controllable so the same network can generalize one way at one time and in a different way with different task requirements

A combination of computer simulations and inference from experimental data has suggested one useful data representation for number. The starting point for our formal system will, therefore, be a suggestion for the representation of number in a neural network. Topographic representations of parameters are very common in the nervous system. For example, in vertebrates, the early stages of the visual cortical representation are topographic in that the image on the retina is roughly mapped onto the two-dimensional cortex. A topographic map of the visual field is used in vision in animals as diverse as humans and the horseshoe crab, *Limulus*. There are many other topographic maps in vertebrate cortex, for example, somatosensory maps of the body surface and frequency of sound in audition.

In a neural network, one useful form of such a topographic representation is called a **bar code** (see Figure 3). The value of the parameter represented depends on the location of a group of active units on a linear array of units. The price paid for this useful data representation is low precision and inefficiency in use of units. If a single parameter is represented only as a location on a surface, then many units are required to represent a value that a single unit could represent by an activity level. Such a physical mapping is also inherently low precision, with precision determined by the number of units devoted to representing the parameter. Such a representation technique and its variants are most naturally implemented in a system that is composed of many relatively inexpensive units and that is performing a function that is only secondarily concerned with high accuracy. Some

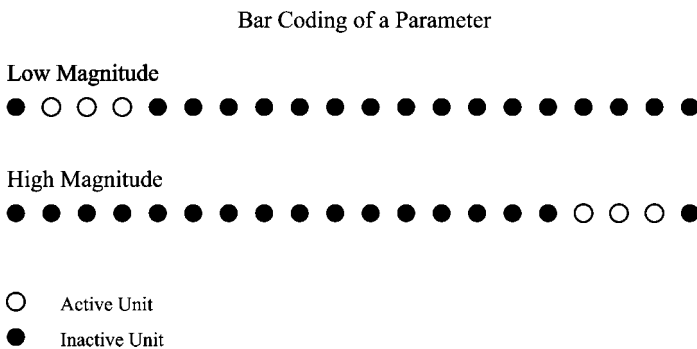


Figure 3. Topographic neural net data representation of a magnitude.

nanocomponent-based computing devices may fall into this category, as does, of course, the nervous system. When used to represent data in artificial neural networks, topographic maps can be very effective for many purposes. One example is the representation of radar signal parameters used in a system to detect and analyze the structure of complex electromagnetic environments (Anderson *et al.*, 1990).

Such a representation can represent an essentially continuous range of values. However, often in cognition discrete states – words, concepts, numbers – are used to represent many different but somehow related examples. Much of the computational power of human cognition arises from its ability to keep only enough information about the world to be useful, and no more. For example, every physical example of a table is different in detail from every other one, but the word ‘table’ often captures the essence of the commonalities of the group and the individual details can be discarded. Perhaps we could consider this process a cognitive example of lossy data compression like MP3. MP3 is not a perfect copy of the music, but still sounds fine. With our network model, we can couple a continuous underlying data representation with a nonlinear attractor neural network with discrete attractors. Figure 4 shows a number of possible bar coded magnitudes that are mapped into two stable output attractors.

In the initial versions of a network model for arithmetic fact learning (see Anderson, 1998) we used a hybrid data representation. The state vector contained two parts, one a bar code and one an arbitrary pattern that could be used to build abstract associations, for example, between the names of the numbers. For the NNPP simulations we only used the bar code part of the data representation, as the ability to perform random name associations is not needed. NNPP used bar codes for the 10 digits, one through zero. We assumed that 0 corresponded to 10. The bar codes overlapped, that is, the patterns were not orthogonal. This overlap turns out to be an important coding assumption since it means that adjacent digit representations

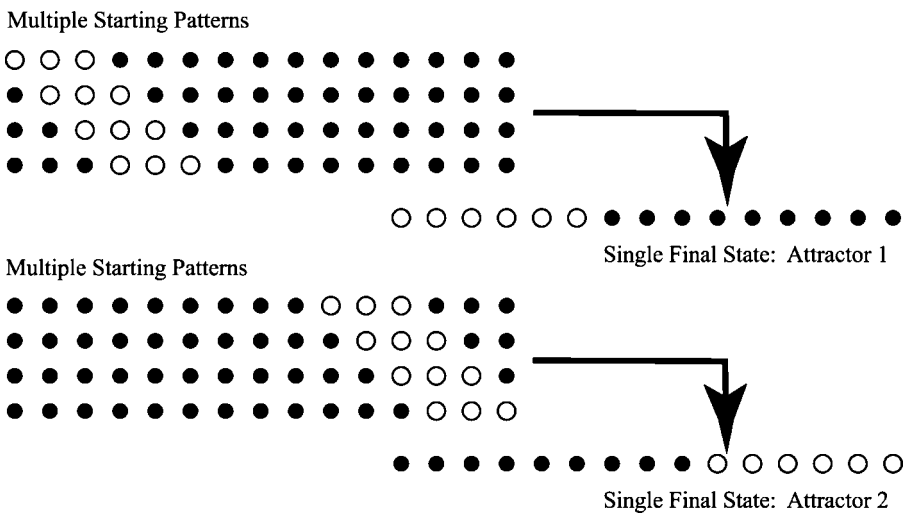


Figure 4. Multiple input patterns associated with a smaller number of attractor states.

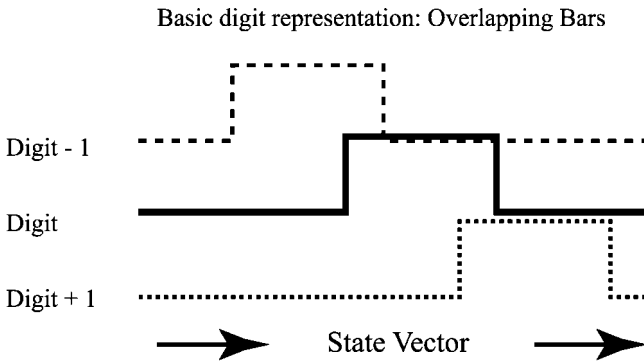


Figure 5. Bar coded data representation for number magnitude. Bars are arranged topographically in order of size.

are correlated. For example, the pattern representing 6 is similar to both 5 and 7 because their patterns overlap but is not similar to 4 and 8. Correlations between neighboring integers give rise to what acts in practice like the successor–predecessor relationships often assumed in logical axiomatizations of the integers (see Figure 5).

The goal of learning is, therefore, to construct a network that has 10 attractors with this correlational structure corresponding to the 10 digits. This problem is not hard for standard learning algorithms. In NNPP, the LMS algorithm learns the 10 digits to a satisfactory degree of accuracy in about 10 epochs, that is, 10 passes through the entire set of input patterns.

6. Programming Patterns: Controlling the Computation

Learning numbers is only the beginning of arithmetic. If we want to build a useful computational system, we have to be able to direct the computation to give answers to specific problems without further learning. Therefore, our first job is to specify the operations that we would reasonably expect an ‘arithmetic’ network to perform. Let us suggest several primitive candidate operations for our computing system.

Counting seems to be present in all human societies of which we have knowledge and evidence for it goes far back into prehistory. There is good evidence that nonhuman primates and many higher mammals can count up to small integers. Formally, we can represent the counting function as two related operations: starting with a digit, add one to it, that is, **increment**, and, the symmetrical operation, subtract one, that is, **decrement**. Another valuable arithmetic related function is comparison of two numbers, that is, the equivalent of the formal operations **greater-than** or **lesser-than**. Another useful operation is **round-off**, that is, two and a bit more can be reported as ‘about two.’

Therefore, we suggest that these five operations form a useful starting point:

1. **increment** (add 1)
2. **decrement** (subtract 1)

3. **greater-than** (given two numbers, choose the larger)
4. **lesser-than** (given two numbers, choose the smaller)
5. **round-off** to the nearest integer

The digits in order are represented as bar codes in adjacent locations in the state vector, that is, the spatial locations of the bars follow the order of magnitudes: 1 next to 2, 2 next to 3, 3 next to 4, and so forth. If we use the bar coded data representation we have discussed (Figure 5), it is surprisingly easy to find programming patterns that work to give the proper answers to the operations. This robustness arises because we are dealing with qualitative properties of the geometry of representation, that is, **representational topology**.

In a previous paper (Anderson, 1998), I suggested a way to control a network using a **vector programming pattern** that multiplied term by term the state vector derived from the input data. The data representation we are using—the overlapping bar codes suggested earlier—contains enough information about the relations between digits to perform these operations easily.

The overall architecture of the system we have been describing is presented in Figure 6. The system has two branches. One is connected to the physical world through the sensory

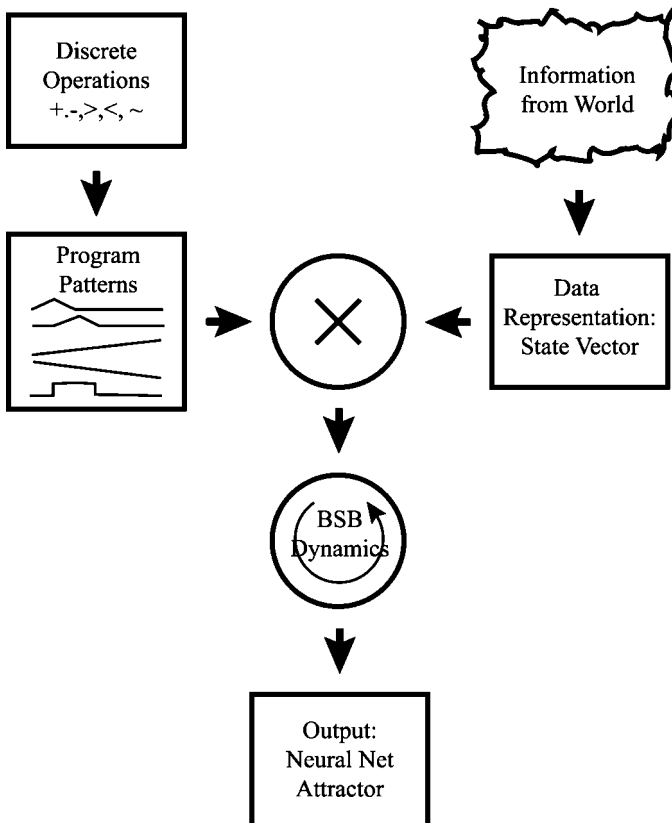


Figure 6. Numerical computation involves merging two components. The left branch contains the program patterns and the right branch contains input quantities. The attractor network generates the output.

'Increment' programming pattern:

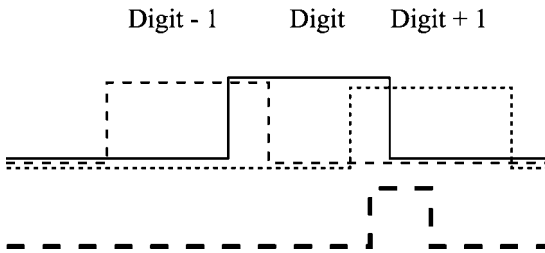


Figure 7. The programming pattern for **Increment** weights the input digit and shifts it to the next attractor to the right.

systems. The other forms the 'abstract' branch and chooses the desired arithmetic manipulation. An operation is chosen. This operation is associated with a programming pattern. In the other branch of the computation, information from the world is represented as a bar code. The two vectors are multiplied term by term. BSB attractor dynamics are then applied. The state vector then evolves to an attractor that contains the answer to the problem.

We can present intuitive arguments to support the particular programming patterns used. Consider counting, that is, the **Increment** operation. If we start from a particular location on the topographic map, we know that one direction on the map corresponds to larger numbers, the other to smaller. Therefore, if we differentially weight the map so that nearby larger numbers are weighted more strongly, the system state can be encouraged to move toward the attractor corresponding to the next largest digit (see the heavy dashed line in Figure 7).

The **greater-than** operation can also be done with differential weighting of the input data, so large digits reach attractors before small digits as shown in Figure 8.

Round-off is performed slightly differently. A bar for an intermediate value is generated at a location between the two digits. The network then moves to the attractor with the largest overlap with the input data bar. The two other operations—**less-than** and **decrement**—are reflections of the patterns for greater-than and increment.

'Greater Than' programming pattern:

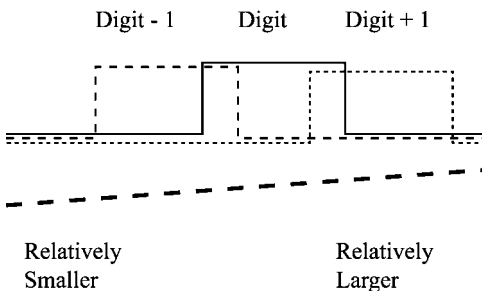


Figure 8. The programming pattern for **greater-than** weights the larger digit more heavily and drives the state vector to that attractor.

There are some important properties of the operation of the program that are worth mentioning. For example, when the 'greater-than' program runs, it displays what is called a **symbolic distance** effect in cognitive psychology. The time to an attractor is a function of the difference in magnitude between the two numbers being compared, that is, the larger of the pair [9,1] reaches its termination state faster than does the larger of [5,4]. In our model, this effect is driven by the magnitude representation in the data representation. The intrusion of statistical and perceptual components into what appears at first to be a purely abstract computation is to be expected from our approach, though perhaps a result not welcomed by purists since it would seem to have no place in logic. Notice also that the topographic map is effectively a realization of the number line analogy for integers.

7. Lateral Spread: The Network of Networks

Although the programming techniques suggested in the previous section work reliably for the digits 1 through 10, they are appallingly slow, limited in precision and dynamic range, and are clearly of no value for constructing a practical computing system that works with precise numbers. But neural networks are excellent pattern recognizers. Even if our proposed system does not do arithmetic very well, it should be able to combine simple arithmetic operations with pattern recognition to give an example of a hybrid computing device, perhaps a useful proof of concept for more advanced systems.

We would like to develop and investigate the properties of a controllable, flexible hybrid parallel computing architecture that merges pattern recognition, integer arithmetic, and, perhaps eventually, logic.

The computational architecture to do this computation involves both the numerical network previously presented and an additional component. A few years ago Jeff Sutton (Harvard Medical School, now NSBRI) and I proposed an *intermediate scale* neural network model we called the **network of networks**. (Anderson and Sutton, 1997; Guan *et al.*, 1997) We called it intermediate scale because its basic level of organization fell between single neurons (1 unit) and brain regions (10^6 units.) Very little is known experimentally about what happens in this region of neural scale except that it is almost certainly both interesting and important. The Network of Networks model has been tested extensively in computer simulation but is very difficult to test experimentally because it works at a nearly inaccessible scale of nervous system organization. We hope this situation will change with new experimental techniques in neuroscience.

The network of networks architecture was based on the assumption that the elementary computing units in the nervous system are not single units as in traditional neural networks, but are modules composed of many interconnected single units (see Figure 9).

The modules themselves are locally interconnected with other modules and repeat in a regular structure, as shown in Figure 10. Modules themselves are assumed to be complex neural networks, constructed from elementary units but to some degree removed from them since they are organized at a different level. In the brain model these elementary networks were assumed to be attractor networks, that is, neural networks realizing a nonlinear dynamical system whose behavior is dominated by attractor states. A state in one module

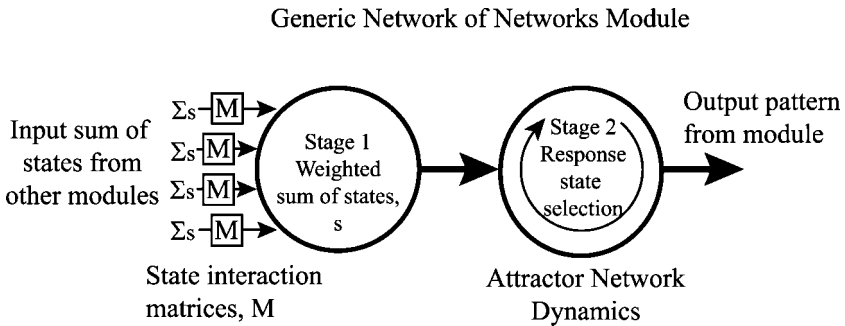
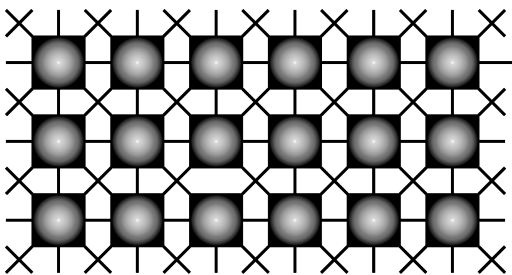


Figure 9. The network of networks elementary processing module. Connections between modules are matrices of state interactions and the nonlinear second stage is an attractor network.

influences the state of an adjacent module by influencing the weights of the activities of the attractor states. In the approximation we used, the modules had a **linear region**, where inputs from different modules added linearly forming a superposition of the module's attractor states and a **nonlinear region** where the module chose a single attractor state from the set of attractors. The temporal evolution of a module is from a sum of potential attractors initially to a single attractor finally. Notice the vague similarity to quantum computing in this property, though the number of considered alternatives is, of course, vastly less.

The use of intermediate scale structure has a number of advantages from the point of view of building computational devices based on nanocomponents. Intermediate level modules may be big enough to work with and control, even in a system built from nanostructures. This structure may also offer some intriguing computational approaches to difficult problems.

In the original description of the network of networks model, an attractor neural network, for example a Hopfield net or the BSB model, was used to form the elementary module. This meant that the resulting network had multiple stable states in the form of point attractors. Therefore, the behavior of a system with many elementary units was dominated by a much smaller number of attractor states. The network behavior was largely buffered from the specific properties of single units. Connections between the modules are assumed



Network of Networks Modular Architecture

Figure 10. The network of networks is composed of a number of locally connected interacting elementary processing units (see Fig. 9).

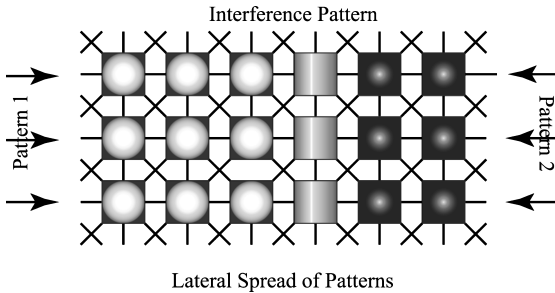


Figure 11. Activation due to particular patterns moves laterally across the network of networks. When two patterns collide, nonlinearity and learning may produce features that are combinations of more elementary features.

to be primarily local, between nearby modules. Local connections allow the number of connections to scale roughly linearly with the number of modules rather than as the square.

Because connections are local, much information processing takes place by movement of information laterally from module to module as shown in Figure 11. This lateral information flow requires time and some important assumptions about the initial wiring of the modules. It is not well understood yet by us though there is currently considerable experimental data supporting the idea of lateral information transfer in cerebral cortex over significant distances. The lateral information flow allows the potential for the formation of the feature combinations in the interference patterns, very useful for pattern recognition.

The network of networks model is a general computational model. However, there have been a number of related computational models specifically designed for vision that have assumed that image processing involves lateral spread of information. An early example is Pitts and McCulloch (1947/1965) who suggested, "A square in the visual field, as it moved in and out in successive constrictions and dilations in Area 17, would trace out four spokes radiating from a common center upon the recipient mosaic. This four-spoked form, not at all like a square, would be the size-invariant figure of a square (p. 55)." In the 1970's Blum (1973) proposed the 'grassfire' model where visual contours ignited metaphorical 'grassfires' and where the flame fronts intersected produced a somewhat size invariant representation of an object.

More modern versions of a lateral propagation ('shock wave') model can be found in Kimia *et al.* (1995) and Siddiqui *et al.* (1999). The approach involves construction of an intermediate **dynamic representation** from the contours of an image. The propagating waves are essentially computing the **medial axis representation**, that is, the point on the axis lying halfway between contours. Arguing against these models, Marr (1982) pointed out that these transforms were unrealistically sensitive to small perturbations in the image, for example, small bends and gaps. Marr's influential criticism caused loss of interest in the approach for a while. However later work has showed it is possible to avoid these problems, or even let them serve as a powerful image segmentation tool.

There is substantial experimental evidence for parts of the medial axis models. There are many examples of traveling waves of different conduction velocities in cortex. Bringuier *et al.* (1999) observed long range interactions in V1 with an inferred conduction velocity of

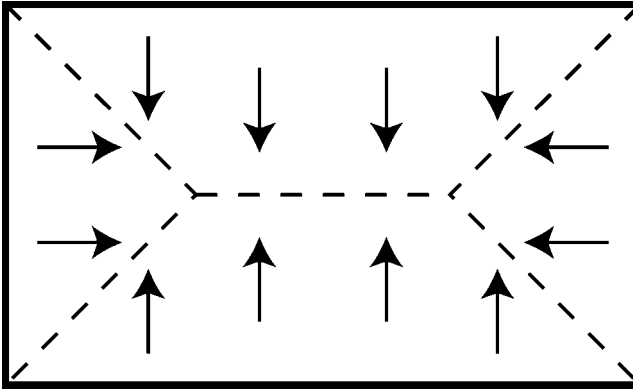


Figure 12. Information propagates from the bounding contours (the solid rectangle). Where two “wavefronts” collide, an interference pattern is formed (the dashed lines). This pattern can be a very useful re-representation of the original image.

approximately 0.1 m/s. Yen and Finkel (1998) developed a model for the perceptual salience of contours for perceptual popout using temporal synchronization of laterally connected visual system units forming a closed contour. Lee *et al.* (1998) discuss units in visual cortex that seem to respond to the medial axis. Particularly pertinent in this context is Lee (2002) who discusses medial axis representations in the light of the organization of V1. In psychophysics, Kovacs and Julesz (1994) and Kovacs *et al.* (1998) demonstrated threshold enhancement at the center of circle and at the foci of ellipses composed of oriented Gabor patches forming a closed contour. Wilson *et al.* (2001) have discussed cortical wave propagation as an explanation for binocular rivalry.

Various versions of medial axis representations have given rise to a useful set of object descriptors. Contours of an image propagate waves based on wave propagation equations (see Figure 12). The interference patterns when waves from different locations on the boundary become powerful representations of some invariant aspects of the object. This approach has led to considerable mathematical analysis and has been applied to figure-ground segmentation, object recognition, and prototype formation and categorization. The major difference between these models and the Network of Networks model is that they usually assume that an unspecified form of ‘activation’ is being spread whereas the Network of Networks assumes that selective pattern information, related to module attractor states, is what is being propagated.

It can be seen that a number of useful computer vision properties are an inherent part of the Network of Network representation:

First, the interference pattern representation itself is somewhat size invariant. If the rectangle expands, parts of the representation are untouched and other parts are changed relatively little, unlike the actual bounding contours.

Second, distortions of the object can leave the qualitative properties of the interference pattern essentially intact. Similarity in shape of the interference patterns suggests equivalences

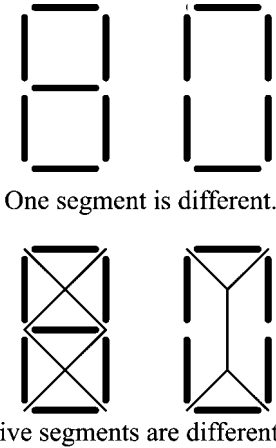


Figure 13. The medial axis re-representation of these digits makes them more orthogonal, hence more discriminable.

between classes of shapes that humans also tend to perceive as members of the same class, for example, rectangles have very different interference patterns than circles or hexagons.

Third, these representations can also act as orthogonalizing transformations when discrimination between similar shapes is important. Consider the two digits in Figure 13, similar to those found on digital displays. Clearly ‘8’ and ‘0’ are very similar as simple patterns with six of seven segments in common. However the medial axis representations are substantially less similar, with five segments different. The re-representation is capturing something of the essential topological differences between the figures.

Fourth, the medial axis representation often allows segmentation of an object into parts or to detect special features of the image. Discontinuities serve to mark important structure. Related variants of the model allow segmentation of a complex figure into parts.

An intriguing speculative application of medial axis models was suggested by Van Tonder *et al.* (2002). They proposed that the layout of rocks in the famous Zen garden in Ryoanji generates maximal Shannon information from the optimal viewing location if it is assumed that the viewer’s perception uses a medial axis representation for the rock locations. “The overall [medial axis] structure is a simple dichotomously branched tree that converges on the principal garden viewing area on the balcony. . . . We believe that the unconscious perception of this pattern contributes to the enigmatic appeal of the garden.” (p. 359)

8. Combining Pattern Recognition with Discrete Operations

Let us try to find a place where we can bring together the simple ‘abstract’ structures we have developed and a network architecture such as the network of networks and see what we can do with it.

One simple test vision problem might be the following. **Given a set of identical items presented in a visual field, count how many there are** (Figure 14 suggests the problem).

Examples of Counting Task

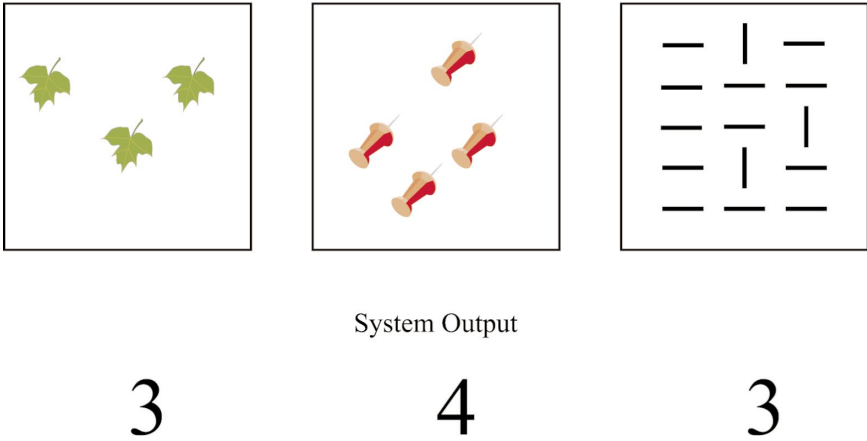


Figure 14. Determine the number of identical objects in a field. The rightmost task is the most difficult.

There is a surprisingly large amount of research on this specific problem and its variants in cognitive psychology and visual perception. There are several reasons for this interest. Experimental results suggest that determining the number of objects present in a visual field proceeds by what are conjectured to be fundamentally different means for different magnitudes. From one to about four items determination of number proceeds in what is called the **subitizing region**. In this region subjects ‘know’ quickly and effortlessly how many objects are present. Each additional item (up to 4) adds about 40 ms to the response time. In the **counting region** (beyond 4 objects in the field) each additional item adds around 300 ms per item, a figure consistent with other tasks where explicit counting is required. There is brain imaging evidence that different neural mechanisms are involved in subitizing and counting (Sathian *et al.*, 1999). Interestingly, this study suggests that there is a single subitizing brain region in the occipital extrastriate cortex whereas explicit counting involves widespread brain activation. Our model uses a single visual region and says little about counting, though that could be modeled as successive applications of the increment operation.

Our basic computational idea is very simple. The network of networks model propagates pattern information laterally. If a number of identical objects are present, they will all be propagating the same pattern information, that is, the same features and combinations of features. Our attractor networks are linear for small signals. Therefore, let us assume we are operating in the linear region of modular interactions, that is, the individual modules have not yet moved near attractor states. Let us assume that when two pattern waves from different sources arrive at the same location they add. Patterns from identical features will add amplitudes linearly; patterns from different features can interfere. After sufficient lateral spread has occurred, the ratio of the maximum activation of a given feature or set of features compared to the initial activation, will give the number of objects (Figure 15 shows this process).

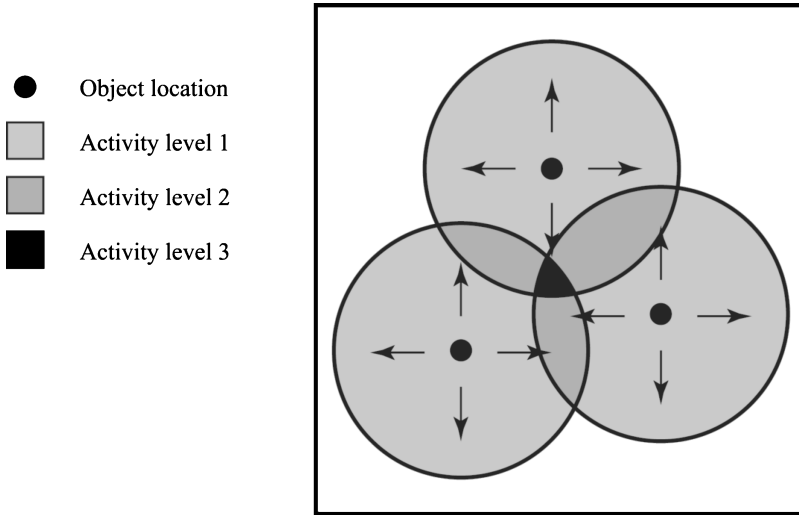


Figure 15. Spreading lateral patterns sum their activation linearly. Numerosity can be determined from the ratio of maximum activation to minimum activation.

A final integer output value can be produced when the output activity is processed by the round-off operator of the numerical network, giving the proper numerical answer. Because of the nonlinearity of the modules, several computational extensions are obvious. It is possible to segment the field by using modules in attractor states. There are a number of effects (metacontrast) suggesting lateral interactions can be halted by interposing lines or regions. For example, segmentation using a line lets the system answer questions like Which grouping has more components? A more homely version of this problem might be “**Which plate has the most cookies?**” a general problem of genuine cognitive and behavioral importance (see Figure 16).

We can analyze this problem as consisting of several steps, each of which we have discussed previously.

First, the image is **segmented** into pieces, with segmentation being accomplished by the line barriers just mentioned.

Second, the **numerosity** of objects in each segment is computed using activity-based lateral spread.

Third, the activity measure is cleaned up and converted into an integer by the **round-off** operation.

Fourth, the integers are compared using the **greater-than** operator with the largest integer is the output.

Why is human subitizing apparently limited to four items? From our point of view there are two interesting reasons. First, the numerical count network ceases to be sufficiently accurate in estimating the activity based numerosity measure. Second, nonlinear effects occur in the spread of lateral activity.

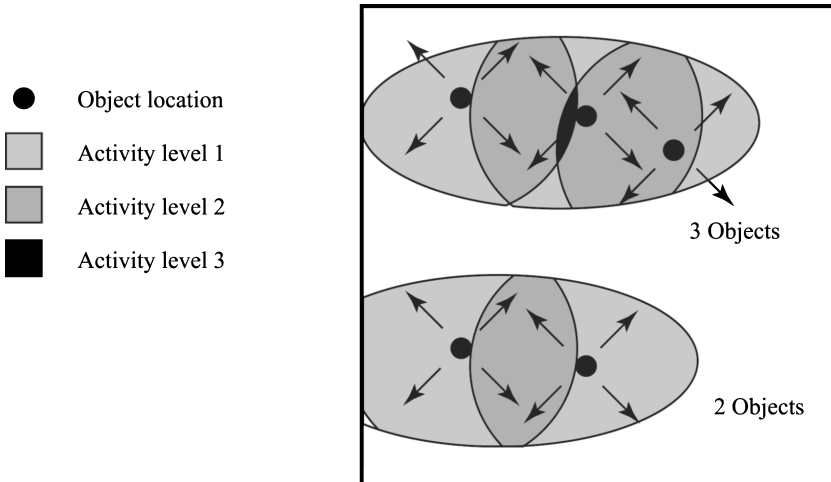


Figure 16. Which plate has the most cookies?

A recent paper by Nieder *et al.* (2002) has presented some fascinating single unit recordings from monkey prefrontal cortex. They used a variety of displays that were comprised of objects somewhat like the objects that we have been discussing. They found single units in prefrontal cortex that responded to number of objects in the field, independent of their shape, up to five. Their response was greatest to a particular number, but adjacent numbers also evoked a weaker response, as we would expect from the overlapping bars we assumed for our bar code. Discrimination between digits was worse as the number increased. They did not detect a topographic map in prefrontal cortex but it is possible a map might be located elsewhere if one is present, for example in a sensory area. The objects in their displays were not always identical, though they were similar to each other and were located in an otherwise uniform field. It would, of course, not be difficult to go far beyond human subitizing performance in an artificial system.

9. Future Development: Linking Computation and Pattern Recognition

Current digital computer technology is exceedingly fast when performing logic and arithmetic. Massively parallel systems are fast and effective for some kinds of pattern recognition and vision problems because their intrinsic parallelism matches the problem. Note, however, that one architecture is bad at what the other does well. This fact presents opportunities for hybrid computer architectures that combine logic based and parallel systems.

The centrality of data representation for neural net operation has been known for a long time. However, there is little guidance beyond experience and intuition as to how to choose the most effective representation for a given problem. The hybrid system described here, where clear discrete computational functions are to be carried out in conjunction with a traditional network immediately suggests why some representations are effective. For example, the 'number line' analogy for the integers is directly realized in NNPP and forms

the basis of the elementary arithmetic operations performed because of the way it represents similarity between adjacent integers. Examples of some of the peculiarities that occur in arithmetic when the line analogy is not explicitly available, as it was not for much of human history, can be found in Yeldham (1926).

However, the most interesting potential extension of this approach is also the most ill formed. We have suggested that it is possible to develop a distributed, largely parallel computer architecture based on a system that harnesses the abilities of discrete and continuous systems that work together. The most important long-term result of this project will be to get some formal idea of the power of this approach as an alternate or, more likely, supplemental computer architecture.

By necessity of its massive parallelism combined with the unreliability and difficulty of controlling single computing elements, a nanocomponent based computer would be an efficient computer for a different set of computational operations than a traditional computer. A major point of this paper is to give one detailed example of what such software might look like, the kinds of problems to which it might be applied, and, where appropriate, to suggest these models may already have some connection to experimental data in both neuroscience and cognitive science.

Acknowledgment

This research was supported by DARPA Award MDA972-00-1-0026 to the Brown University Center for Advanced Materials Research.

References

- Anderson, J. A., 1993: The BSB network, in M. H. Hassoun (eds.), *Associative Neural Networks*, Oxford University Press, New York, pp. 77–103.
- Anderson, J. A., 1995: *An Introduction to Neural Networks*, MIT Press, Cambridge, MA.
- Anderson, J. A., 1998: Learning arithmetic with a neural network, in D. Scarborough and S. Sternberg (eds.), *An Invitation to Cognitive Science, Volume 4: Methods, Models and Conceptual Issues*, MIT Press, Cambridge, MA, pp. 255–300.
- Anderson, J. A., Gately, M. T., Penz, P. A. and Collins, D. R., 1990: Radar signal categorization using a neural network, *Proc. IEEE* **78**, 1646–1657.
- Anderson, J. A. and Sutton, J. P., 1997: If we compute faster, do we understand better? *Behav. Res. Methods, Instruments, Comput.* **29**, 67–77.
- Blum, H. J., 1973: Biological shape and visual science (Part I), *J. Theor. Bio.* **38**, 205–87.
- Binguier, V., Chavane, F., Glaeser, L. and Fregnac, Y., 1999: Horizontal propagation of visual activity in the synaptic integration field of area 17 neurons. *Science* **283**, 695–699.
- Graham, D. J., 1987: An associative retrieval model of arithmetic memory: How children learn to multiply, in J. A. Sloboda and D. Rogers (eds.), *Cognitive Processes in Mathematics*, Oxford: University Press, Oxford, pp. 123–141.
- Guan, L., Anderson, J. A. and Sutton, J. P., 1997: A network of networks processing model for image regularization. *IEEE Trans. Neural Networks* **8**, 1–6.
- Hadamard, J., 1949: *The Psychology of Invention in the Mathematical Field*, Dover, New York.
- Kimia, B., Tannenbaum, A. and Zucker, S. W., 1995: Shapes, shocks and deformations {I}: The components of shape and the reaction-diffusion space, *Int. J. Comp. Vision* **15**, 189–224.
- Kovacs, I., Feher, A. and Julesz, B., 1998: Medial point description of shape: a representation for action coding and its psychophysical correlates. *Vision Res.* **38**, 2323–2333.

- Kovacs, I. and Julesz, B., 1994: Perceptual sensitivity maps within globally defined shapes. *Nature* **370**, 644–646.
- Lee, T.-S., 2002: Analysis and synthesis of visual images in the brain, in P. Olver and A. Tannenbaum (eds.), *Image Analysis and the Brain*, Springer, Berlin.
- Lee, T. S., Mumford, D., Romero, R. and Lamme, V. A. F., 1998: The role of primary visual cortex in higher level vision. *Vision Res.* **38**, 2429–2454.
- Marr, D., 1982: *Vision*, Freeman, New York.
- McCulloch, W. S., 1951/1965: Why the mind is in the head, in W. S. McCulloch (ed.), *Embodiments of Mind*, MIT Press, MA, Cambridge, pp. 72–141.
- McCulloch, W. S. and Pitts, W., 1943/1965: A logical calculus of the ideas immanent in nervous activity, in W. S. McCulloch (ed.), *Embodiments of Mind*, Cambridge, MA, MIT Press, pp. 19–39.
- Nieder, A., Freedman, D. J., Miller, E. K., 2002: Representation of the quantity of visual items in the primate prefrontal cortex. *Science* **297**, 1708–1711.
- Pitts, W. and McCulloch, W. S., 1947/1965: How we know universals: The perception of auditory and visual forms, in W. S. McCulloch (ed.), *Embodiments of Mind*, MIT Press, Cambridge, MA, pp. 46–66.
- Sathian, K., Simon, T. J., Peterson, S., Patel, G. A., Hoffman, J. M. and Grafton, S. T., 1999: Neural evidence linking visual object enumeration and attention. *J. Cogn. Neurosci.* **11**, 36–51.
- Siddiqi, K., Kimia, B., Tannenbaum, A. and Zucker, S., 1999: Shocks, shapes, and wiggles. *Image Vision Comp.* **17**, 365–373.
- Van Tonder, G. J., Lyons, M. J. and Ejima, Y., 2002: Visual structure of a Japanese Zen garden. *Nature* **419**, 359.
- Wilson, R., Blake, R. and Lee, S.-H., 2001: Dynamics of traveling waves in visual perception *Nature* **412**, 907–910.
- Yeldham, F. A., 1926: *The Story of Reckoning in the Middle Ages*, Harrup, London.
- Yen, S.-C. and Finkel, L. H., 1998: Extraction of perceptually salient contours by striate cortical networks. *Vision Res.* **38**, 719–641.