No. 44 November-December 1988 $3.95

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Object-Oriented Programming

Objects are the latest buzz word. For a long time there was only Smalltalk, now there's a whole troup of players including Actor, Objective C and C++. So we start out this issue with a four-piece trilogy on objects: complete with theory, examples, and projects. (Space is no object.)

Thinking Objectively

Computers are dumb. Right? They only do what you tell them. Right? Even expert systems boil down to a list of rules, and that's about as fancy as computers can get. Right?

Wrong. (You knew I'd say that.)

If Doug and Arthur are correct, we'll just teach computers how to learn and then turn them loose. In no time at all, they'll be the experts. All by themselves. Mankind can go back to manual labor.

Our minds have the (as yet) machine-unsimulatable ability to make sense out of noise. We can distinguish individual (and often subtle) meaning out of a cacophony of sounds.

Imagine a symphony. You not only hear the orchestra (as a whole), but also distinguish many of the individual instruments. You can even hear people whispering in the audience or the siren of an emergency vehicle outside.

You differentiate among sounds although they occur simultaneously, have overlapping frequencies, and are input through only two analog channels — your ears. In your mind, different sounds represent different "objects."

The ability to recognize objects is essential for high-level intelligence and must precede, to some degree, any high-level association of meaning.

If a squirrel, for example, repeatedly sees a cat and learns that the cat is dangerous, it *first* recognizes the cat as a separate object in the real world. Then it associates the object, cat, with danger.

The recognition step is essential. Without it the squirrel would have no quick way of associating danger — danger would be entwined with an infinite number of real world stimuli. The learning of objects is fundamental to intelligent organisms. And it's of equal importance in the design of intelligent machines. Despite this, we don't have a good understanding of the idea of object.

For the past three years, we've been building systems that use conventional and electronic technology to explore (in depth) the idea of the object. The results are encouraging.

We believe that a better understanding of what the object is will remove a longstanding roadblock in the transition from fixed design methods (e.g., expert systems) to the development of truly automatic learning technologies.

In this article we'll present some of the fundamental concepts involved with object recognition and some general approaches to implementation.

Who Teaches?

Many traditional neural networks can learn to recognize stimuli — but these networks must be told what they're learning. We do this by telling the network to learn when the desired stimuli are present. In other words, we teach the network.

We decide what the network will learn by placing chosen stimuli (objects) in nonconflicting scenes, and directing the network to respond when it recognizes a particular set of stimuli. Is this how animals learn?

We don't think so. Although you might argue that animals learn about danger from their parents or from their genes, it seems more likely that the world, itself, contains the clues needed by animals to group information. So far, neural nets have not tried to automatically extract this information.

Seen It Before; See It Again

Most of us make the erroneous assumption that neural tissue "takes in" data, puts it somewhere, remembers where to retrieve it, and reproduces the data intact. Conventional computers operate this way, but neural tissue doesn't. It neither stores input data nor needs to!

How can tissue remember (recognize) objects if it hasn't stored data?

Good question. When remembered stimuli appear at the input of neural tissue, the tissue responds, that is, it determines whether it knows (has experienced) the stimulus.

In other words, if the tissue has experienced a stimulus or group of stimuli before, it can experience it again. See Figure 1.

Animal Learning

There are many animals, some very complex, that appear to learn little (or nothing) in their short lifetimes. The highly stereotyped responses of many insects fall into this category. How do these animals acquire their useful behaviors?

When we realize that life, in general, is a learning system, we realize that the design of these insects (their genetic makeup) was learned through repeated testing and gradual improvement. The result is similar to a well-tested expert system that's been given the necessary complexity to be successful.

But many animals (like people) seem to learn throughout their lifetimes. They absorb knowledge of an extremely complex real world and automatically adjust to deal with it. It's known that we have over 10,000,000,000,000 neural synapses and only about 100,000 genes, or developmental instructions, to encode these connections.

It's obvious that these are far too few genes to supply enough information for the design of such high-level order. The genes, nevertheless, provide the necessary framework to get learning under way, although most of the complexity of life is acquired (learned) thereafter through experience.

As the designers of artificial intelligence systems, we might consider a similar approach. A fixed design would work in many simpler cases, but we'd have trouble deliberately designing-in the

By Doug Gaffin & Arthur Gaffin
Dept. of Zoology
Oregon State U.
Corvallis, OR 97331

Neuro Dynamics
1514 Canna Court
Mt. View, CA 9404

complexity of a high-level intelligent system. A modified form of mammalian intelligence may actually be the best approach. If we design and build the framework, the rest of the system can be self-learned.

The Forces Of Weak Association

Within real world experiences, there are relationships between events that provide hooks for learning. We call these relationships "the forces of weak association."

The term weak is used because these relationships are useful only in a statistical sense — they don't provide immediate absolute associations. These observations are simple and very important. They give intelligent animals the ability to gradually develop a correct and complex model (not a copy) of nature.

The forces of weak association are —

(1) If a stimulus is similar to another stimulus, not necessarily occurring at the same time, then the associated events in nature are likely to be related.

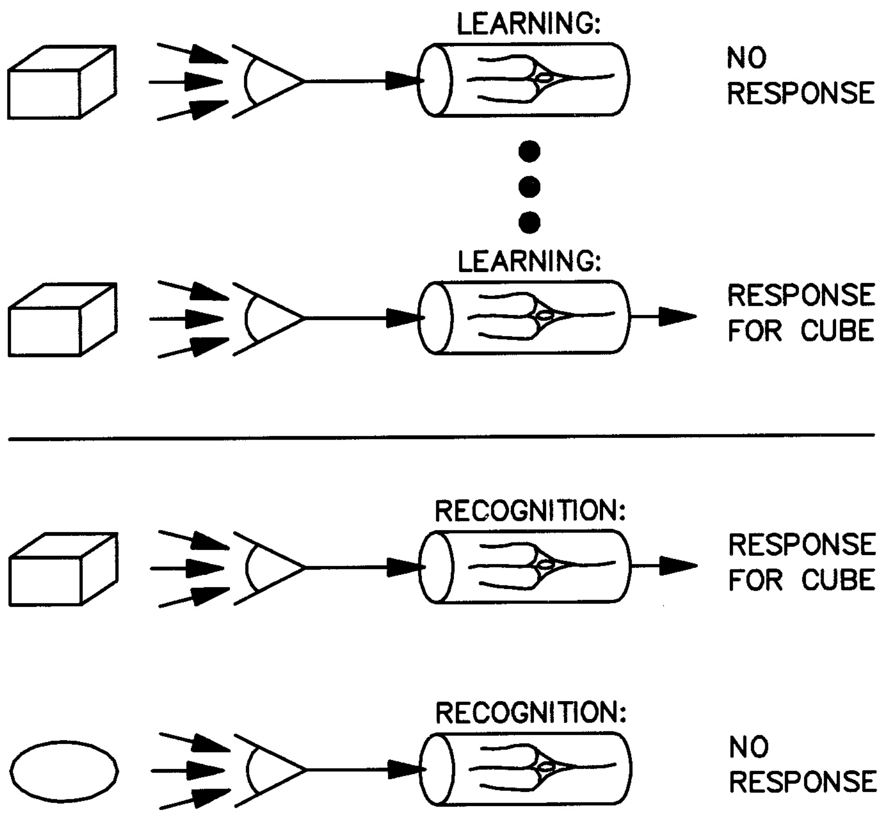
(2) If two or more stimuli are repeatedly encountered proximate with each other, in terms of time and/or space, the events are likely to be related.

(3) The more proximate the observation, the more important the relationship.

(4) The more frequent the observation, the more important the relationship.

(5) Most events in nature are continuous. For example: If an object is detected at one moment, it is likely to remain present for an extended period.

Figure 1 — Simple Learning



The All-Important Object

Let's look at the concept of an object in more detail. Suppose we show Rob, our robot, a coffee cup that's resting on a table. We want Rob to learn to recognize coffee cups. To make this task more interesting, assume that the cup casts a shadow.

If Rob is to learn what a coffee cup looks like, he must learn where the cup ends and the table begins. And he must recognize that the shadow isn't part of the table.

How do we recognize shadows, cups, and tables? By learning their intrinsic qualities during many experiences of cup, table, shadow.

We see the cup in different locations and circumstances on the table. We observe the stimuli that relate to the cup, say the handle, to be more frequently with the rest of the cup. And we observe stimuli that more frequently relate to the table, cup or no cup. The shadow is really no different; we observe it most frequently with the cup. Therefore, the shadow eventually becomes part of the cup object.

With this in mind, let's define an object —

An object occurs when certain patterns of stimuli repeat themselves, in a manner that's similar, in terms of time and/or space, with a frequency of repetition that is, on the average, relatively high.

Or, more simply —

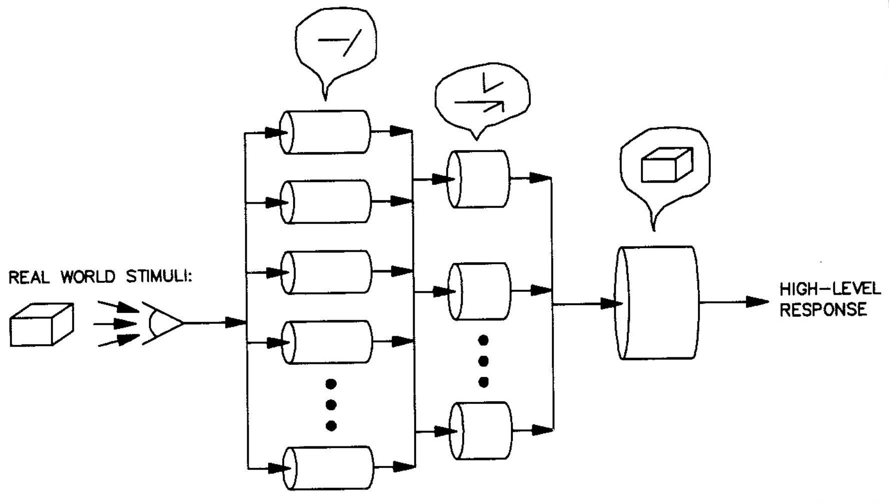
An object occurs when two or more stimuli are observed repeatedly together. The real world is ordered (we order it through a multitude of associations. Objects represent the embodiment of this order — forming the basis of fundamental associations.

Objects Yield Hierarchies of Knowledge

Suppose we have a piece of neural tissue and we program it (give it the ability) to learn and recognize frequent combinations of stimuli. We also program it to respond uniquely to each of these learned patterns of stimuli.

This is a high frequency filter — emerging responses correspond only with frequently occurring events at the input. Infrequent events aren't remembered and, therefore, aren't passed to the output as responses.

Figure 2 — Hierarchies



The events that are learned by this piece of tissue are objects. The organism takes the world for what it is and extracts information from it to build its model.

An infant in its first days of visual experiences can't physically alter its visible world, except for the direction and quality of view. The initial flow of data is largely one-way, from the real world to the infant.

The most frequent of these patterns of stimuli will be learned first and these will consist of small relationships that repeat themselves frequently — long edges, corners, basic shading patterns, types of blobs, etc.

The next level of learning will be in terms of these low-level features. Since the input stimuli of the next level are the responses from the first level, the next level sees its real world through the filters of the first level.

A third level may be added in the same manner, and so on. With many levels, very general and complex objects can be learned and recognized. See Figure 2.

Tricks Of Tissue

How can we design a simple, fast, and self-adjusting mechanism that automatically learns to sort out the most frequent stimuli? Let's look at a piece of tissue at one level in the hierarchy. How can it remember to recognize only stimuli that are frequently observed and therefore important?

Studies suggest that neural tissue employs temporary memory in the synapses (points of connection between neurons). These synapses are eventually converted to permanent memory if repeatedly reinforced. Some studies also suggest that neural growth may be affected by the level of activity the neuron.

Also, the growth of some neurons seems to be affected by the activity of adjacent neural processes. It's therefore possible that neurons may be selectively seeking other neurons with high levels of firing activity with which to form connections (synapses).

Uninvolved neural processes can be allowed to forget by remaining unconnected, being routed around (overwritten) and, in certain cases, atrophying. Biological systems have developed other tricks as well to selectively filter only the most frequent of stimuli.

Trick 1: Crowding

By simply "crowding" the tissue, we can create a competition for synapses. Less frequent stimuli will be reinforced

DBASE™ for Norton Guides™

INTRO PRICE $69.00

DBASE *ON LINE* is a "pop-up" DBASE language reference system which includes over 2 million bytes of complete reference with clear concise descriptions, and detailed examples to every command function and feature for:

CLIPPER™ (Summer '87)	**QUICKSILVER**™ (Diamond Release)
dBASE III Plus™	**dBXL**™ (Diamond Release)
dBASE IV™	**FoxBASE+**™

DBASE *ON LINE* is powered by the Norton Guides "reference engine":

- Memory requirements: maximum 65k.
- Can run in resident or pass-through memory mode.
- Can be "popped-up" any time, inside any program.
- Automatically looks up a keyword read from the screen.
- Full or half-screen display.
- All available Norton language guides will run under the Norton "reference engine".
- Also included is a compiler and linker, allowing the creation of your own reference guides.

Additional Features:

- Tables - keyboard return codes, line drawing characters, color codes, error codes, ASCII chart and much more.
- The Clipper and Quicksilver guides also include commands and switches for the compile/link cycle, a table of reserved words and complete reference to the extend system.

ORDER: DBASE *ON LINE* and get the above language guides, a reference engine, a reference guide compiler/linker and manual.

30 Day Money Back Guarantee • Satisfaction Guaranteed
To order, call or write:

SofSolutions

440 Quentin Dr. • San Antonio, TX 78201 • (512) 735-0746

Trademarks: Norton Guides/Peter Norton Computing, dBASE/Ashton Tate, Clipper/Nantucket, dBXL Quicksilver/Wordtech Systems, FoxBASE+/Fox Software

Reader Service Number 108

less often and tend to become overpowered. See Figure 3.

Crowding gives neural tissue a simple, automatic mechanism to reduce the complexity of real world data — it reduces the data to the most important (frequent) events and responds only to these. The next level can repeat the process using the output of the previous level, thereby automatically creating orderly hierarchies.

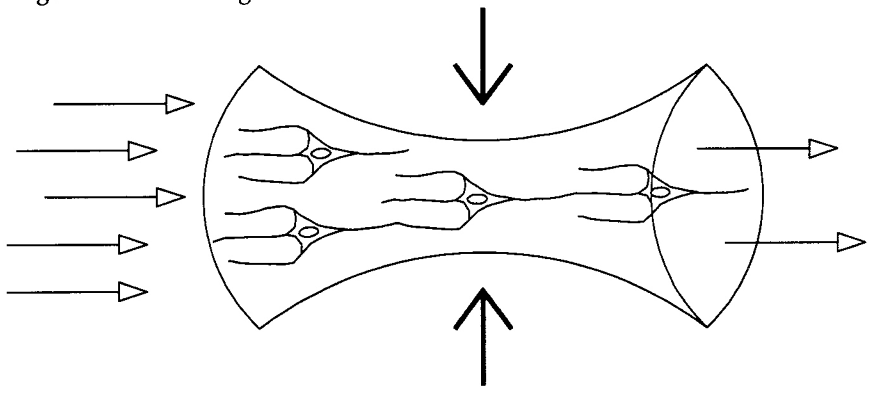
Trick 2: Inhibitory Synapses

Studies have shown that the connections (synapses) between neurons are inhibitory as well as stimulating by nature. When neural activity of inhibitory neurons reaches one of the places where two neurons touch, the pre-synaptic neuron inhibits activity in the post-synaptic neuron.

Since one neuron may contact as many as 10,000 other neurons, we can see that many neurons may be silenced (or at least partly silenced). This phenomenon can implement crowding even when there are more than enough synapses. It also causes active ideas in that tissue to choose addresses (synapses) that are as unique as possible for use as memory. Animal tissue, therefore, has implemented a clever, fault-tolerant addressing scheme that works even if some of the tissue is damaged.

Now that we understand why we need these inhibitory synapses, we can select whichever techniques work. It's important to note that we *don't need* to use the massively parallel approach once we understand what problem these tissue features are solving.

Figure 3 — Crowding



CROWDING is useful for learning and recognizing only the more important patterns of stimuli. Note that the pathwidth set of possible values of the input stimulus is greater than that of the response, thereby forcing a net reduction of the incoming data.

A Powerful Procedural, Database, and Declarative Language.

SPEED — Where Prolog must backtrack, Trilogy can often solve the problem logically. Trilogy takes advantage of logic constraints (they constrain the search to possible solutions) which either eliminate backtracking or reduce millions of backtracks to a very few.

SYNTAX — Trilogy uses an intuitive, Pascal-like, program structure.

INTEGRATION — Trilogy is complete. It's the only language you need for writing Pascal-style routines, database handlers, and Prolog-style programs.

MODULARITY — Trilogy is modular language, very similar to Modula-2.

ENVIRONMENT — A complete programming environment, you get editor, library, linker, loader, error handling, automatic make, and contextual help. Plus, you get modules for: math, string handling, file manipulation, windows . . .

A TRUE COMPILER — Trilogy is an interactive compiler which produces native code for the 8086/8087.

LOGICAL PURITY — Trilogy was designed from scratch as a simple language with a completely logical foundation. Trilogy's speed results from its design, not from added commands. (Prolog's assert, cut, var, and retract, are not logical parts of that language. They were added to improve performance.)

DATABASE SUPPORT — Trilogy supports: variable size records, records with arbitrary values (lists, recursive trees); plus record insertion, deletion, and modification. (Anywhere in the file.) Files are relations and can be queried from within the language.

PRICE — Only $99.95 postpaid, U.S. funds. Plus $5.00 shipping & handling. Or $12.00 shipping & handling outside North America. Check, money order or VISA accepted.

Order From:

741 Blueridge Ave.
North Vancouver BC Canada V7R 2J5
(604) 986-3234

ONLY $99.95

Trick 3: Winners Tire

If you look at Necker's cube for 30 seconds (see Figure 4), it appears to flip back-and-forth between two opposing interpretations (ideas). We can speculate that the winning idea eventually fatigues and, therefore, allows the second one to win.

Eventually that one fatigues, and the first idea wins again, and so on. Since the first idea is still partially fatigued from the first time, it will fatigue more quickly this time. The flip rate eventually settles at about two times per second which is directly related to our neural fatigue rates.

Fatigue is neural tissue's form of a do loop — it provides a foolproof way of trying many alternative ideas before repeating the original one. It implements a tissue-level checkoff list that automatically moves from one idea to another.

Given a multi-level hierarchy with feedback at different levels, we can see that a search through combinations of ideas can be very complex and would never exactly repeat itself. It's likely that the subconscious mind uses some of these mechanisms — it would help explain why it takes a long time for some answers to pop into our mind.

Fatigue adds a self-adjusting quality to the system and causes it to perform better in a complex world. It also keeps the system from getting stuck in infinite feedback loops.

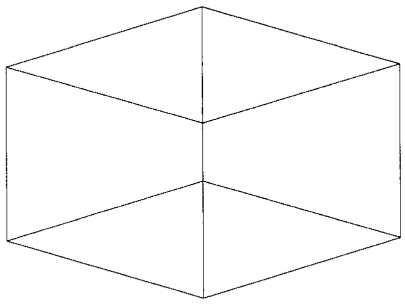
Since slowly changing stimuli tend to fatigue the associated paths, rapidly changing stimuli will have a competitive advantage — therefore fatigue created a bias towards changing stimuli. We can call this "tissue-level curiosity," the tissue actually gets bored with old information and responds better to dynamic situations.

Trick 4: The Axon

The real world provides an abundance of data, in fact so much data that the same exact set of stimuli or scene is never observed (experienced) more than once. We know that similar scenes are observed repeatedly, but the data in these similar scenes usually does *not have* many pixels of information that are the same. How, then, are these scenes similar?

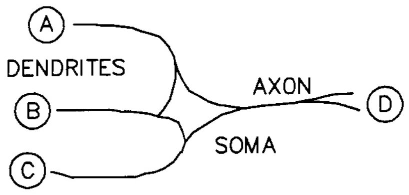
Let's consider a stimulus as a number which represents a point in a special highly multi-dimensioned space called, for lack of a better term, white space. This white space needs to be vast enough to represent a rich repertoire of stimuli, but not so vast that incoming stimuli cannot readdress previously learned points.

Figure 4 — Necker's Cube



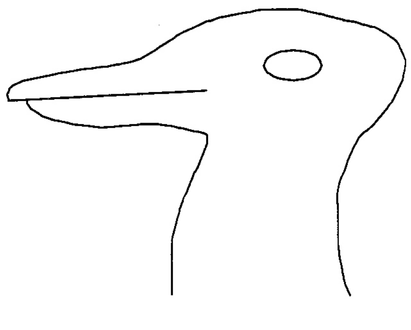
NECKER'S CUBE is an optical illusion that illustrates your neural tissue in action. When looked at for 30 seconds, the image will appear to flip back and forth between two different recognitions.

Figure 5 — Two out of Three Rule



The neuron acts as a digital reduction device. A simplified model could be a device that fires an action potential through the axon whenever at least 2 out of the 3 possible dendrites are active.

Figure 6 — Conflicting Images.



The drawing above should at first look like a DUCK. The drawing can also look like a RABBIT. Note that after the high-level idea of rabbit is thought of, all of the low-level features are immediately converted to those of a rabbit.

The neuron is ingeniously designed to provide an operational solution to this problem. A neuron consists of the following computational components:

- input processes - dendrite(s)
- decision mechanism - soma
- output processes - axon(s)
- inter-neuron connections - synapses

The neuron provides a very rich set of computational ingredients:

- COMPLEX INPUT system — inputs through the dendrites which provide a wide pathwidth of data.
- SUMMATION of the inputs as received by the central neuron body, the soma.
- DIGITAL DECISION system, in the soma, which acts somewhat like an analog to digital converter — it acts as a democratic element which fires only if enough inputs (votes) are present.
- DIGITAL OUTPUT system which fires and sends out an action potential through the axon.
- COMMUNICATION system in which the axon contacts many other dendrite processes of other neurons.
- MEMORY that is contained in the connection strengths of the synapses (the inter-neuron contacts).
- LEARNING which is the process by which the connection strengths and the locations of the synapses are formed.
- RECOGNITION when the system is excited by inputs and fires a digital output.

The neuron can receive many combinations of possible stimulus inputs which represent a fairly large white space. The output (axon), however, is binary (digital) — fire or no-fire.

Note that the neuron is functioning as a data REDUCTION device. By making decisions, the data is simplified and reduced by the action of the neuron. This process keeps the white-space of the data at any point in the neural tissue within operation bounds. See Figure 5.

Trick 5: Feedback

Some optical illusions show that the same picture (stimuli) can provide alternate interpretations. See Figure 6.

Some important points are listed below:

- Some illusions display alternate interpretations where only one can be visualized at a time — in other words, they compete.

- Many of the supporting low-level features are interpreted in concert with the overall interpretation.
- When the overall interpretation changes, these low-level features change also.

What is the usefulness of this mechanism?

- By the context of the overall interpretation, lower-level features can be more easily recognized, especially if the stimuli is difficult to interpret.
- If lower-level features were allowed to resolve totally on their own, they would seldom conclude a useful high-level recognition.
- Fatigue, ordinarily a very useful mechanism, would make matters difficult — the low-level recognitions would flip back-and-forth with little contribution to determining a valid high-level recognition. See Figure 7.

The subject of feedback extends far beyond the purposes discussed here — it's likely to be involved in nearly all neural processes in one form or another. The mechanism discussed here is fairly simple and easy to implement. In fact, we have implemented it in Turbo Pascal. The complete source code (and user-friendly .EXE) is available on the Micro C RBBS and from the authors.

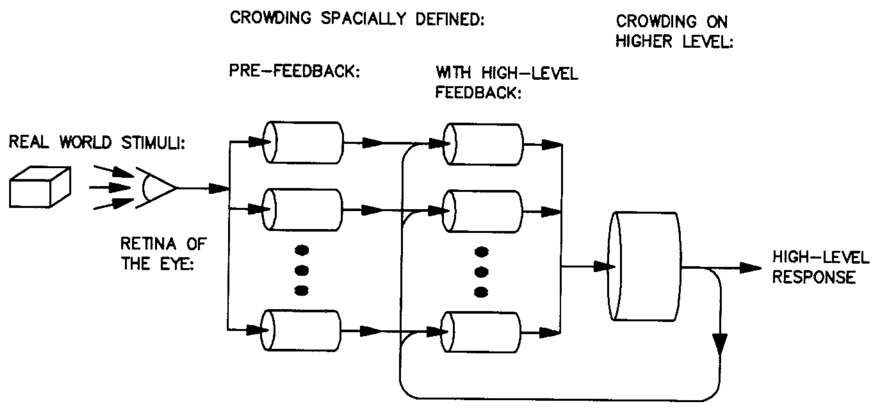
Summary

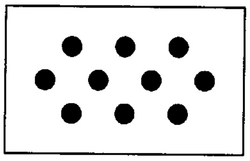
Electronic technology is moving forward at an extremely rapid rate, providing unprecedented computational power for the money. We now have more tools than ever for building neural network based systems. As expected, the level of public and private interest in neural intelligence is increasing dramatically.

With our new understanding of automatic learning, we are now ready to remove major roadblocks to our progress. By designing-in the ability to learn complex objects within complex hierarchies, we delegate most of the learning to the system — thereby making our job much easier.

♦ ♦ ♦

Figure 7 — Conflicting Images suggest Feedback


CROWDING SPACIALLY DEFINED:
CROWDING ON HIGHER LEVEL:
PRE-FEEDBACK:
WITH HIGH-LEVEL FEEDBACK:
REAL WORLD STIMULI:
RETINA OF THE EYE:
HIGH-LEVEL RESPONSE



CONFLICTING IMAGES suggest FEEDBACK: The set of black dots shows two different images of circular patterns—but both cannot be seen at the same time. The model shown above suggests one possible configuration of recognizers (neural tissue) that promotes similar conflicting recognitions. Small combinations of dots form subsections (arcs) of the circles, but, due to high-level feedback, the arcs are interpreted according to the context of the high-level idea (one circle or the other.

dBASE III+ 20 TIMES FASTER!!

DBDO

dBASE III+ Enhancment utility

"What a difference! No more waiting for output while I could have been processing other data. It's great!"
V. Kovacs
Penn Services

"I'm using it in every new system. I write. Super for creating test data from large files. It's fast, easy to use, and follows dBASE syntax."
W. H. Whitney
McGraw Hill, Inc.

FAST:
Up to 20 times faster than dBASE.
In one case, report generation on a 60,000 record file was reduced from 18 hours to 2 hours!

FLEXIBLE:
Call from a program file or DOS prompt.
Run on a stand alone PC or a network.

EASY:
dBASE-like syntax - No need to learn another language.

COMPATIBLE:
Recognizes and creates dBASE III+ files.
Transfer DBF data to DAT files for use with other languages (Basic, Pascal, etc.)

Commands: COPY, DELETE, APPEND, RECALL, REPLACE, COUNT, MANY MORE

Functions: LTRIM[], UPPER[], TRIM[], SUBSTR[]

Introductory Offer
$99.00
Offer ends 12/31/88

FOR ADDITIONAL INFORMATION CALL: (215) 536-5858

Computerized Processing Unlimited
Country Square Shopping Center
Quakertown, Pa. 18951

Reader Service Number 105