A GENERAL PURPOSE IMAGE PROCESSING PACKAGE

Michel Caplain , Claudine Metral, Christian Pellegrini

Centre Universitaire d'Informatique, Université II
24, rue du Général Dufour, 1211 GENEVE 4, SWITZERLAND

ABSTRACT

Optical image processing provides an elegant solution to a limited class of applications. Instead of using optical filters and lenses, we propose to propagate discrete images through massively parallel *"Cellular Networks"*.
The free choice of cells and connections - which may be endowed with memory - appears sufficient to implement any conceivable function.

A sequential simulator is available under UNIX BSD 4.2 to help design and validate such image processing networks. Higher resolution implementation would normally require special hardware.

This architectural model is also a candidate for other areas of Artificial Intelligence.

RESUME

Les processeurs optiques apportent une solution élégante à certains problèmes de traitements d'images. Pour les autres cas, nous suggérons de remplacer les lentilles et filtres optiques par des *"Réseaux Cellulaires"* propageant en parallèle de vastes images discrètes. Les cellules et connections - éventuellement douées de mémoire - sont arbitraires, permettant de composer une variété illimitée de traitements.

Le montage et l'expérimentation de réseaux s'effectuent à l'aide d'un simulateur séquentiel, utilisable sous UNIX BSD 4.2. La construction finale, en haute résolution, des réseaux ainsi validés, nécessite du matériel particulier.

D'autres secteurs de l'Intelligence Artificielle pourraient bénéficier de ce type d'architecture.

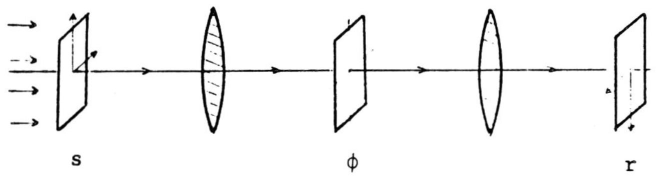
KEYWORDS: Optical Processing,Digital Signal Processing,Filtering,Artificial Intelligence

I - OPTICAL PROCESSING

Many computer professionals are not aware that optical processors exist (Casasent 78) which are capable of rather intelligent tasks such as image restoration, feature extraction or pattern recognition. Typical examples of optical processing are pictured in figure 1, next page.

1 - Optical filtering

Most optical processors are variations around the following *linear filtering* set-up:



A source image (s) transmits coherent light due to laser illumination. A first lens focuses its complex *Fourier Transform* $F(s)$ which is transformed back by a second lens. The normal flow of light is intercepted by a filter image ϕ, located in the Fourier plane, so that the middle image is $\phi.F(s)$ instead of just $F(s)$, and the result image is $r = F'(\phi.Fs)$ instead of $r = F'Fs = s$.

Figure 1a displays an example of *Pattern Recognition* , where every occurrence of the template "Optics" is to be signalled by a proportionately bright dot. The filter $\phi=1/F(\text{template})$ is shown above the template. A centered template would then be transformed into

$$F'\left(\frac{1}{F(\text{template})}.F(\text{template})\right) = F'(1) = \delta_0$$

(a centered dot)

Since filtering is linear and shift-invariant, every occurrence of the template will yield a dot in the corresponding position.
Because of energy conservation, the rest of the image does not disappear, it is randomly scattered into a noise-like background.

Filtering is also continuous, so that approximate occurrences of the template (e.g. "Optical" instead of "Optics") yield a slightly blurred dot (with a loupe, this should be visible on the photograph).

Similarly, image *refocusing* (fig. 1b) can be attempted by searching for the estimated blurred impulse response template.

2 - Advantages

This type of information processing has many interesting properties hardly found in digital processors.

a/ Simplicity

Transforms can be performed *globally* with just a few filters. Better yet, a sum (or a sequence) of transforms can be performed by a single filter which is the sum (respectively the product) of their filters.

Also notice that, instead of producing numbers, formulae or listings, optical processors display qualitative results, akin to *nuance* in human mind.

b/ Fault-tolerance

Fig. 1c illustrates that a local destruction of 15% of the Fourier transform results in a barely visible loss of overall definition.
This is due to an immediate dilution of information by light, so that "everything goes everywhere", and no feature can be completely lost.

Unlike computers, where each transistor must be perfect, optical equipment may be dusty, scratched or greasy with no more consequence than a slight degradation of overall precision.

c/ Throughput

A spatial resolution of 2000x2000 in 250 shade levels is common, so that 3.10^7 bits of information are processed instantaneously.
Assuming a time resolution of 10 wavelengths (i.e. 6.10^{13} images/second), the inherent information processing capacity exceeds 10^{21} bauds.
Technologically, we are still unable to effectively use such a reserve of processing capacity.

———

It should be noted that some efforts in computer hardware (e.g. systolic arrays) tend to reproduce, on a small scale, these most interesting properties of optics.

3 - Limitations

Optical processing is still far from providing a perfect solution.

a/ Functional limits

Non-linear and a fortiori non-continuous transforms can generally not be implemented with present day optical processors.
Even linear shift-invariant applications may be difficult to perform, because the appropriate filter must be invented, a task whose subtlety may be compared with that of computer programming.

b/ Memory

To our knowledge, film recording remains the only form of memory in use. Optical processors do not learn. The recording of complex filters requires the use of holographic techniques, at some loss in precision.

c/ Operational difficulties

Unlike computer programs which can be typed, reproduced and teletransmitted, each copy of an optical processor must be carefully aligned by well trained specialists, on heavy benches, in soundproof laboratories. Additionally, the use of lasers implies safety requirements.

———

Given this appraisal of *current* optical processing, our goal is to find an information processing architecture that would cumulate the advantages of both optical and digital processors.

II - CELLULAR NETWORKS

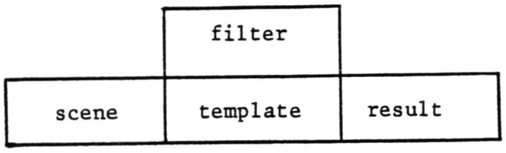
Cellular networks, as defined here, result from an attempt to generalize optical processors. They can be seen as machines performing *General Filtering* of images.

Their technological feasibility will be briefly discussed later; at this point, they should be considered as a purely *mathematical model*.

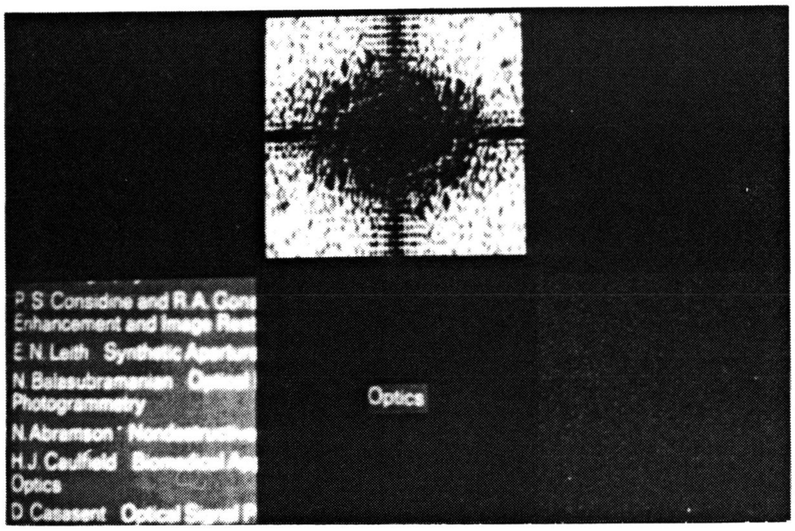
Why networks ?

Computers can (slowly) simulate optical processors of any resolution. (The reader may have guessed that we simulated the "optical" examples of fig.1).

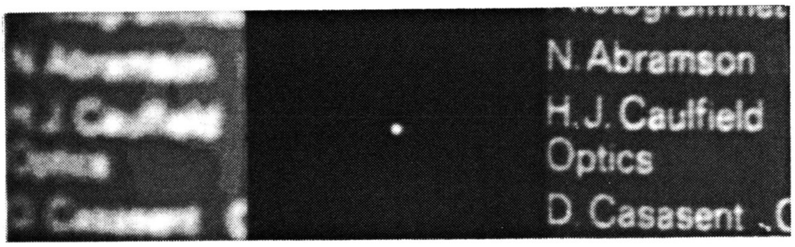
1a



Pattern Recognition


P.S. Considine and R.A. Gon
Enhancement and Image Res
E.N. Leith Synthetic Apertur
N. Balasubramanian Optical
Photogrammetry
N. Abramson Nondestructive
H.J. Caulfield Biomedical Ap
Optics
D. Casasent Optical Signal P
Optics

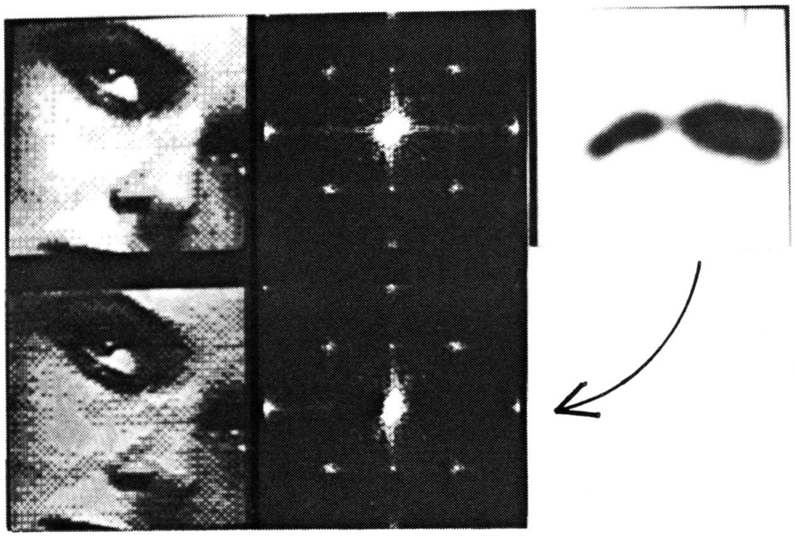
1b



1c

$$\begin{tikzcd} \boxed{x} \arrow[r] & \boxed{F(x)} \arrow[r] & \boxed{\text{alteration}} \arrow[dl] \\ \boxed{\sim x} & \boxed{\sim F(x)} \arrow[l] & \end{tikzcd}$$

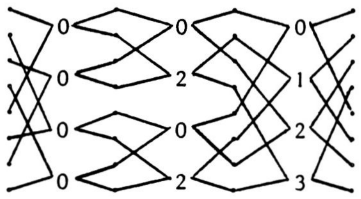
Fault Tolerance



- FIGURE 1 -

This is based on the Fast Fourier Transform algorithm (Cooley & Tukey 65) which is best described as a data flow graph: the Fourier transform is performed by *propagation* through successive "layers" of "cells", each of which linearly processes two "pixels":

8 point FFT :



Therefore, all optical processing can be performed by cellular networks. But networks are functionally more powerful, if we allow non linear cells or any form of memory.

The following is a short definition of the model, in accordance with the simulator which has been implemented.

1 - Data

Under this model, all data are discrete images, restricted for convenience to rectangular sets of pixels.

Various pixel types have been implemented, including some that cannot be displayed (complex numbers). Types are partially ordered, allowing upward conversion when necessary:

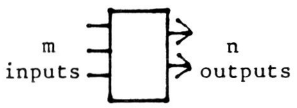
BIN < INT < REAL < COMPLEX ...

2 - Composition laws

Networks perform functions on images and can be composed according to normal n-ary function composition rules.

a/ "arity"

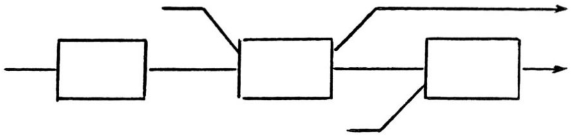
An (m,n)network reads m images and produces n images



Thus a (0,1)network is a "source" (e.g. a noise generator).
The Fourier transform is performed by a (1,1)network.
A multiplicator is a (2,1)network, etc...

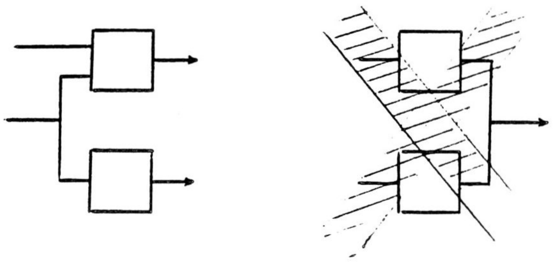
b/ Serial composition

Networks can be pipelined when an output of one is an input to the next:



c/ Parallel composition

Networks can share inputs.
Sharing outputs is meaningless.



Arbitrarily complex networks can be built by combining serial and parallel connections.
Loops are a special case of serial connection.

d/ Initial networks

Initial networks are those which are not composed of lower level networks. Without loss of generality, initial networks can be restricted to at most two input images and one output.

3 - Layers,cells,connections

An initial network is not an atomic object; it is a pipeline of "*layers*" (most often a single layer).

Layers, in turn, are made up of concurrent and independent "*cells*". Within a layer, there is no inter-cell communication; each cell processes a selection of input pixels specified by a left-hand side *connection*. Each cell also produces an exclusive subset of the layer output, specified by a right-hand side connection.

The various cells in a layer may perform unrelated processing, but they are usually at least similar.

A connection is a mathematical object expressing logical data dependency; it is not a set of "wires" and may not attributed physical properties such as length, weight, capacity or other such parameters which must be assigned to cells.

4 - Memory

We understand the term "memory" as the variability of a system; thus with this model, memory is one or both of the following:

a/ cell variation (e.g. status variable, hysteresis, fatigue ...);

b/ connection variation, known as "*plasticity*".

Notice that memory is dispersed everywhere and not confined to a dedicated physical unit as in ordinary computers.

III - COMPUTER ASSISTED PROBLEM SOLVING

Image processing problems are usually not amenable to clear cut specifications; they are rather -like cooking- informal matters of taste, best solved by *trial and error*.
But this is efficient only if one can attempt numerous approaches and variations in reasonable time and with moderate effort.

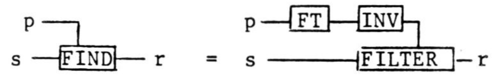
We propose a package built around the cellular network model. It accepts network descriptions and simulates them. Thanks to an increasing catalog of preprogrammed components, little or no programming is necessary.

The main difficulty for computer addicts may be to think "global" and visualize image processing as propagation through filters .

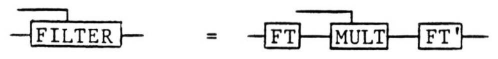
The package, available for distribution to UNIX users, provides the following features:

1 - High level description of networks

Admittedly, the most convenient syntax would allow the direct drawing of network interconnections. For example, the pattern recognition network would be defined as:



with



This will be specified by the statement:

```
= p; FT; INV; = s; FILTER pop
```

where the command "FILTER" is a previously defined executable file containing the description:

```
FT ; MULT $1 ; FT'
```

Network description is embedded in the UNIX shell language (csh), so that it is directly executable.

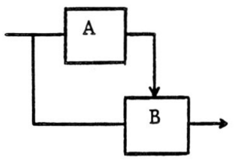
The rules are as follows:

a/ "= x" pushes the current image (if any) onto a background stack and brings image x into the foreground;

b/ unary functions (like FT) act on the foreground image. To produce $F(x)$, write "= x; FT". This is a postfix notation.

c/ binary functions (like MULT) need to be given the second argument. To produce image x×y, write "= x; MULT y". This is an infix notation.
In "subroutines" (like the FILTER example above), formal arguments are named $1,$2 ...

d/ special image names like pop and top, and special commands like SWAP,SPLIT,SAVE help describe parallel composition of networks:



$\Longleftrightarrow$ SPLIT;A;SWAP;B top

This may seem awkward, but there is no easy language to describe arbitrary connection diagrams. A scratchpad would help keep track of the stack !

2 - A Kit of frequently used networks performing arithmetics,geometric and all sorts of signal-processing tasks.
Arbitrary functions may be defined on-line and directly applied to every pixel of the foreground image.

3 - A Toolbox to construct new initial networks.

Standard connections, cells and declarations help build new networks:

```
BEGIN(REAL,NIL,REAL)
LAYER(PIXtoPIX,log)
END
```

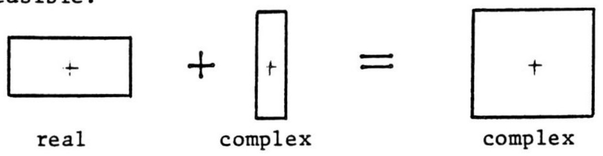
completely describes a unary network of type REAL → REAL. It is made of one layer of log cells in the pixel-to-pixel connection.

A compile command "ccc" creates the corresponding executable network.

Instructions are given to program new connections and cells,if necessary; only then, some knowledge of language C is required.

4 - Hands-off handling of image types and sizes

Automatic upward conversion applies, when feasible:



5 - Tracing facilities to remember sessions:

To save space, images can be regenerated from their histories.

6 - Display commands are obviously hardware-dependent.

IV - A CASE STUDY

This section reports an actual exercise in *Gel Electrophoresis* interpretation, illustrated fig. 2.

Gel Electrophoresis (Anderson 77) creates pictures showing protein accumulations around a few hundred specific locations, each of which characterizes a precise type of protein. Various "constellations" have been experimentally correlated with potential upcoming diseases. For prevention purposes, the problem is to help detect diseases whose signature is known.

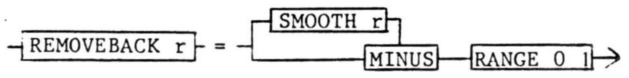
A 128x128 portion of a much larger electrophoretogram has been selected. It is obviously of poor quality, so that preliminary cleaning steps are necessary.

STEP 1 shows the original image.

The horizontal extent of its Fourier transform is related to the conspicuous vertical "dragging". By filtering out some of the horizontal Fourier extension, we would have reduced or eliminated this effect; but this has been omitted here.

STEP 2 removes the background.

This emphasizes the significant features.



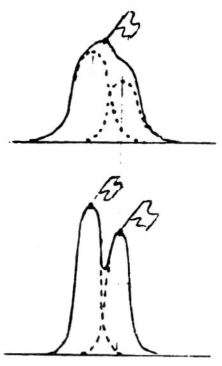
Several smoothing radii were attempted, radius 30 was selected.

The next image shows the peaks, most of them due to noise; two plateaus show that the exposure was saturated.

STEP 3 smoothes the image to remove false peaks.

A smoothing radius of 1 was experimentally found to remove enough noise without erasing significant peaks. The image seems worse, but the peak selection -shown next- is better.

STEP 4 is an efficient peak separation method.



When two hills are close together, they tend to merge into one hill with only one (displaced) peak instead of two.

Deblurring with a narrow Gaussian yields narrower hills while preserving their volumes, and restores actual peaks which the naked eye would have been unable to separate.

STEP 5 estimates the volume under each peak.

This represents the protein amount.

Theoretically, for a standard hill shape subjected to independent elongations along the three axes, the volume is proportional to:

$$\text{peak}^2/\text{curvature}$$

This gave unstable results because flat areas having near zero curvatures result in enormous volumes.
We applied the simpler volume model peak^2, knowing it should be improved.

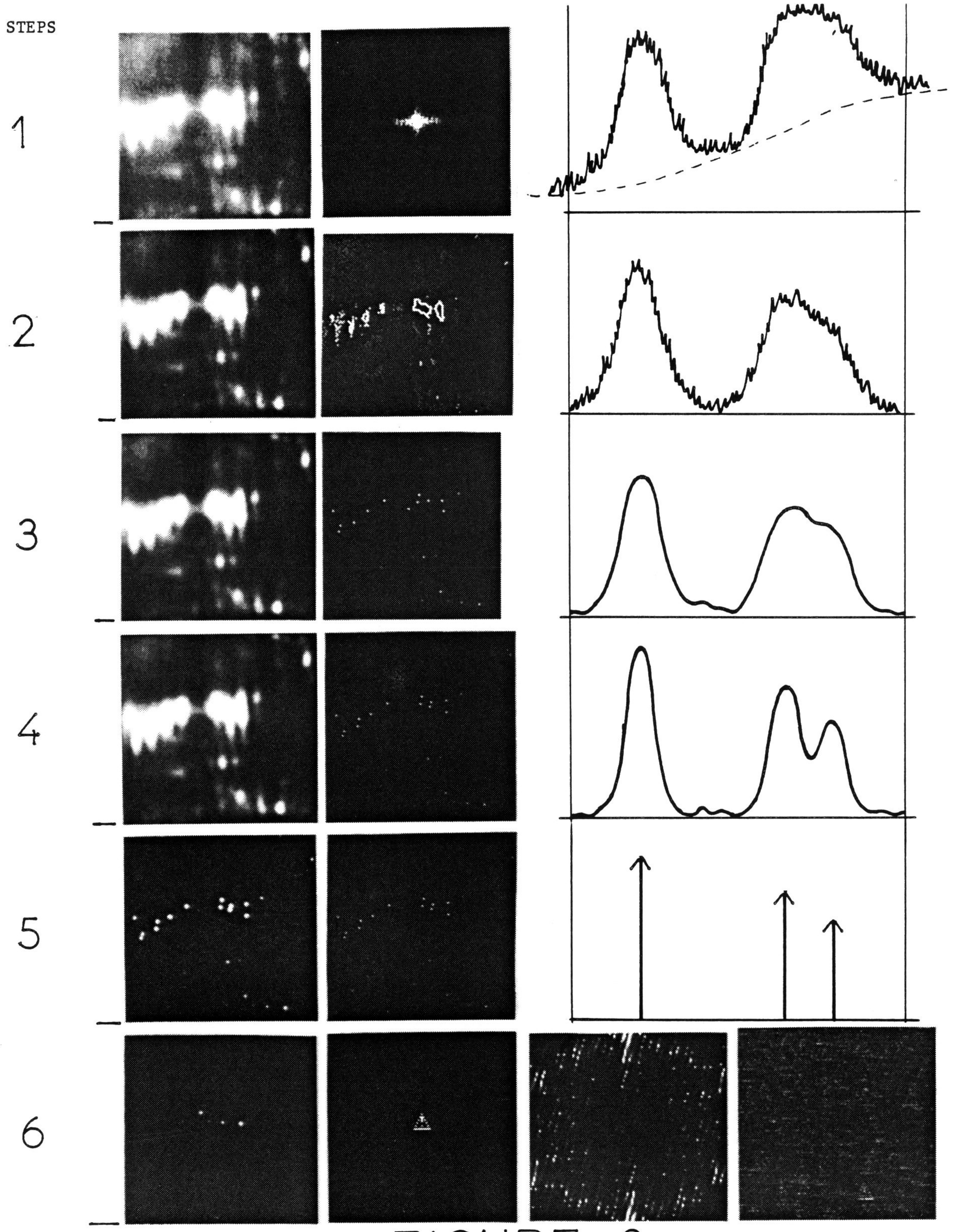
For visual aid, peaks were convoluted with e^{-r}/r to display magnitude in the form of log-scale discs.

At this point, we could have stopped and sent a listing of weighted spots to some symbolic interpretation software ("expert system").

STEP 6 shows a perfectible attempt at diagnosis by pure image processing methods.

We depict the signature of a hypothetical disease identified by a specific little warning symbol shown next. From these, we compute the complex filter $\frac{F\text{symbol}}{F\text{disease}}$ which should replace any occurrence of the signature by the warning symbol.

The (rather poor) result signals the occurrence (bottom right) against an excessive background.



- FIGURE 2 -

The last step, not shown, would be an integrated diagnosis of several <disease,symbol> couples, by simply applying the sum of their filters.

The resulting "program" is the following:

```
REMOVEBACK 30; SMOOTH 1; FIND gauss1
PEAK; THRESHOLD 0.1; VOLUMEMODEL
REPLACE disease symbol
```

where most commands are themselves compositions of networks.

This could be optimized by collapsing linear filtering sequences, reducing the number of costly Fourier transforms to four.

DISCUSSION

Much qualitative improvement is needed and possible on this example. This is the result of a five hour session at the terminal, two thirds of which were in waiting for Fourier transforms to unfold. 128x128 FT takes 17 CPU seconds (VAX 780 UNIX BSD4.2), multiplied by 2 or 3 because of time sharing. A richer "kit" of preprogrammed networks and special purpose FFT hardware would considerably speed up the whole computer aided design process.

V - PERFORMANCE

In continuous operation, sequences of images propagate through a succession of layers. It is convenient to measure the time in units equal to the propagation delay through the slowest layer, which imposes its rhythm upon the whole network. Then, the response time of a network is simply equal to its longest sequence of layers, and the throughput is one image per unit of time.

Simulation or sequential computer implementation is another story, because it can process only one image at a time, which divides the throughput by the number of layers. Furthermore, with little or no parallelism, the required time is proportional to image resolution.

It can be shown that any global function, where each output pixel depends on most input pixels, costs at least K.N.logN operations, where N is the resolution. For example, a small 512x512 Fourier transform takes 5 minutes of a VAX780 when special hardware requires 1 second and optics 10^{-9} second.

Storage requirements in case of cell memory or connection variation are enormous.

The package should only be used to experiment small resolution versions. Actual implementation of high resolution products requires appropiate hardware. Array processors are a faster but costly possibility.

Special purpose VLSI is conceivable to implement common layers and variable connections (N.logN gates are enough to implement any N to N connection).

In some cases optical processors are a good solution and holographic filter plates can be synthesized by our system.

Analog devices that mimic the cellular network model are being considered.

VI - PROSPECTS

There is no reason to restrict the model to the processing of "visual" images. Some images like holograms or various transforms are totally meaningless to even the most experienced eye.

Likewise, unreadable images may have the potential for representing concepts which Artificial Intelligence vainly tries to capture in words, symbols and formulae.

There are reasons, neurobiological or other, to believe that language, ideas, knowledge, reasoning are mostly effects of image processing activity. For example, inhibitor filters could be a welcome substitute for present day heuristics.

A final obvious observation is that some of the model's features also make it suitable for neuromimetic experiments.

VII - REFERENCES

Anderson 77 : N.G. ANDERSON & N.L. ANDERSON
High Resolution Two-dimensional Electrophoresis of Human Plasma Proteins
Proc. Nat. Acad. Sc. U.S.A.
num 74, pp 5421-5425, 1977

Casasent 78 : D. CASASENT
Optical Data Processing
Lect Notes in Physics, Vol 23, Springer Verlag

Cooley & Tukey : COOLEY and TUKEY
An Algorithm for Machine Calculation of Complex Fourier Series
Math. of Comput. , vol 19, Apr. 1965

Ullmann 73 : J.R. ULLMANN
Pattern Recognition Techniques
Butterworths, London 1973