Journal of Network and Innovative Computing
ISSN 2160-2174 Volume 2 (2014) pp. 010-021
© MIR Labs, www.mirlabs.net/jias/index.html

On Multichannel Neurons, with An Application to Template Search

Helio Perroni Filho[1], **Alberto Ferreira De Souza**[2]

[1]Universidade Federal do Espírito Santo, Laboratório de Computação de Alto Desempenho,
Av. Fernando Ferrari s/n, Vitória, Brazil
xperroni@gmail.com

[2]Universidade Federal do Espírito Santo, Laboratório de Computação de Alto Desempenho,
Av. Fernando Ferrari s/n, Vitória, Brazil
albertodesouza@lcad.inf.ufes.br

Abstract: **The multichannel model is a complete reassessment of how neurons work at the biochemical level. Its results can be extended into an overarching theory of how vision, memory and cognition come to be in the living brain. This article documents a first attempt at testing the model's validity, by applying its principles to the construction of an image template-matching framework, which is then used to solve a Graphical User Interface (GUI) automation problem. It was found that the template-matching function thus implemented can consistently locate required visual controls, even when template and match differ by color palette or (to a slighter degree) feature proportion. The article concludes by discussing the significance of these results and directions for further research.**

Keywords: cognitive models, computer vision, fourier transform, GUI automation, machine learning, template matching.

I. Introduction

Today, neuroscience is besieged by two pressing questions. On the one hand, a general theory of brain function remains elusive, as most research so far have focused on specific structures or groups of cells – disconnected efforts out of which a coherent picture has yet to emerge [1]; on the other, not only we don't have a clear idea of how a whole brain works – we may have failed to grasp what individual neurons do, as well.

Classical neurophysiology states that a neuron's immediate output – a "spike" – is a 1-bit datum, by itself unfit to express virtually any meaning. This is a direct consequence of the much verified fact that all spikes produce an electrical potential of about the same amplitude: without evidence of any other kind of signal being transmitted, we are led to conclude that a single neuron cannot produce any response more elaborate than whether it's "on" or "off". Therefore, only by combining multiple spikes in some form of *code* can any meaningful message be communicated across the nervous system. In particular, *frequency-coding,* where information is thought to be encoded on the number of spikes a neuron gives off over a fixed length of time, was for the longest time the dominant neural code theory [2].

The problem with frequency-coding – any coding theory, actually – is that neurons are *slow.* A nerve impulse's speed was measured for the first time by Prussian scientist Hermann von Helmholtz in 1849, who recorded an average speed of $27m/s$ [3]. This value eventually turned to be on the low end of the scale – myelinated nerves can conduct signals at speeds approaching $100m/s$ [4] – but that's hardly lightning speed either. Furthermore, after a neuron discharges, it undergoes a *refractory* period of about $1ms$ during which it cannot fire again [5]; so, not only do nerve signals travel slowly, there is a strict limit on how quickly they can be produced. Both findings place heavy restrictions on how efficient a nervous system communicating on encoded messages could be.

One might argue that the lack of speed is circumvented in the brain by a highly parallel architecture. However, no processing problem is composed entirely of parallelizable parts; some sequential parts must exist, if the parallel computations are to interact in any way. This sequential part will place a restriction on how fast a task can be made to run on any number of processing units. According to Amdahl's Law, for P processing units and a sequential part of length α, the maximum speed-up S that can be achieved is:

$$S = \lim_{P \to \infty} \frac{1}{\frac{1-\alpha}{P} + \alpha} = \frac{1}{\alpha} \qquad (1)$$

In the brain, the "sequential part" is the exchange of spikes among neurons. Therefore, far from making it unimportant, the brain's highly parallel architecture turns the speed of spikes into the *dominating* performance factor: given how fast spikes travel along the axon, the time a neuron requires from one spike to the next, and how far nerve impulses have to travel from sensory organs to the brain and then back to the motor system, a limit can be set on an animal's optimal reaction time.

From the early 1990's onwards, experiments on bats and flying insects managed to establish how fast those creatures can change course to avoid an obstacle after they notice it [6]. It was found that this time window – from the moment an obstacle comes into view to that of the motor response – is about as short as the time a single spike would take to traverse the involved neural pathways. It is not, obviously, that a single spike on a single neuron would somehow recognize the obstacle, plot and execute an evasion; the point is that no matter how many neurons are involved in the operation, each one has only enough time to produce a single spike.

One way around this experimental result is to remark that there is *apparently* only a single spike present – but somewhere, in some other pathway, a cycling or otherwise baseline signal exists which could provide a reference for decoding that lone impulse. It's a clever workaround that preserves the concept of a neural code – though it's hard to imagine how an impulse produced at some random moment, when measured against a predictable reference, could produce *just* the right meaning that was required at that given instant.

A simpler, if more heretical alternative, is to consider the possibility that no code exists – a single spike is somehow able to transmit meaningful information from one neuron to the next. The *multichannel neuron model* proposed by John Harris [7] takes this route and, by working out the theoretical consequences, brings together a number of overlooked experimental results into a surprisingly feasible theory of how the brain works.

Unfortunately, for all the sense it makes, the multichannel model stills lacks an experimental body that would unequivocally validate its claims; in fact, at least where its basic biochemical propositions are concerned, the technology to study neurons at the required scale may be years ahead. Meanwhile, however, there is nothing preventing us from looking into other ways to investigate its validity – for example, through computer experiments.

Computers are often employed as substitutes when direct experiments would prove unwieldy [8] – software tools such as simulation suites can be used to test the viability of biological models, helping advance our understanding of living beings. This also works the other way around, as Computer Science researchers often take a page from biology when designing solutions for problems in their own domain. Indeed, over the years, this has proved a very productive partnership, particularly in fields such as neuroscience and studies on vision, both biological and mechanical.

This article (an expanded version of our original paper [9]) presents a first attempt at evaluating the multichannel neuron model and related theories through computer experimentation. It starts by summarizing an overall model of brain function founded on the concept of multichannel neurons; then it introduces **Skeye,** a programming framework implementing a limited version of that model. The framework is used to solve a problem of desktop automation involving *template matching,* the computer equivalent of the *visual search* skill found in sighted life forms. The article concludes by discussing the significance of the experiment's results, its shortcomings, and directions for further research.

II. Related Work

The dialogue between neuroscience and computing, with the aim of explaining and perhaps even reproducing human cognitive skills, goes back to the invention of the perceptron algorithm by Frank Rosenblatt in the late 1950's [10]. But it was only in the late 80's, with the development of back-propagation [11] that the related *connectionist* approach flourished as an AI field. However, much as its source material, neuroscience-inspired computing research has focused more on solving specific problems (such as pattern recognition) than on developing comprehensive cognition models; and conversely, research on such models is often done outside of connectionism, and doesn't often concern itself with the question of biological plausibility.

Deep learning [12], connectionism's most promising development in recent time, illustrates this trend. While its techniques incorporate a number of recent results from neuroscience (such as the "convolutional" nature of neuron layers in the primate visual pathway), it is for the most part concerned with constructing better pattern recognition machines – doubtless an important goal, but rather narrow when compared to the full range of human cognition. On the other hand, general problem-solvers such as Goedel Machines [13] could theoretically exhibit many of our cognitive abilities – but are too departed from the biological realm to constitute a plausible model of biological cognition.

A conspicuous exception to this divide is the Hierarchical Temporal Memory (HTM) model [14], which intends to be at once a biologically-plausible general theory of intelligence. Its progress over the years has been encouraging, however HTM is still very computationally expensive, particularly as problem sizes increase [15]. This impact both the scale of experiments it allows, and its applicability as a software tool.

III. A Model of Visual Cognition

In this section we construct a model of visual cognition founded on the concept of a multichannel neuron. We start from a short review of eye optics, working our way up through conjectures on neurobiology until we arrive at an abstract model of the brain as a data processing device. As we go, we draw theoretical support and inspiration from a variety of sources; though familiarity with them all is not required, the reader is encouraged to check the reference material [2, 6, 7, 16–18] for a more complete discussion.

A. Images and Diffraction Patterns

In this age of ubiquitous digital cameras and display devices, we might be forgiven for taking visual records for granted. Yet, for the longest time one's own eyes were virtually the only reliable source of visual accounts: though it was known since Antiquity that light passing through a pinhole projects an image of its source on an opposing surface, it was not until the 19th century that the first photochemical pictures were recorded. Even then, the contraptions used to capture such images differed little from the *camera obscura* employed by artists and scientists alike for centuries before, a drawing of which is shown on Figure 1.

This kind of drawing is a typical example of *Geometrical Optics*, where light is modeled as particle rays traveling, unless deflected, along straight lines. This is a valid abstraction for most scenarios in classroom Optics; however, its relatively simple mathematics is ultimately too limited to truly describe the process of image formation, where the phenomenon of *diffraction* takes center stage.

The phenomenon of diffraction was first described by Thomas Young in the early 19th century. In his classical

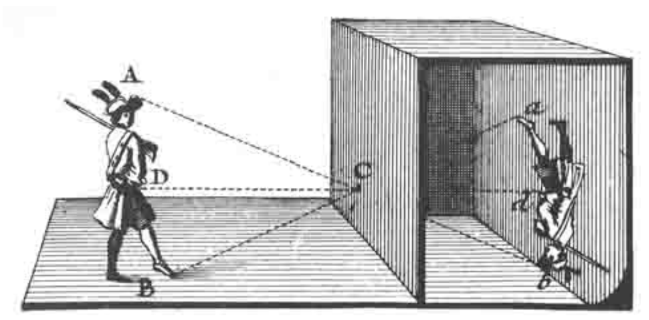

Figure. 1: A model of image formation according to Geometrical Optics. Light rays coming from an infinitude of points at a source object (such as sample points A, B and D) pass through the aperture of a *camera obscura* (C) and are projected on inverted locations at the opposite wall (such as sample locations a, b and d). Image retrieved from [19].

double-slit experiment – as seen in Figure 2 – he demonstrated how light, passing through apertures of size comparable to its own wavelength, exhibits interference patterns characteristic of wave dynamics. This is an experiment of historical significance, as it demonstrated for the first time that light, despite its many instances of particle-like behavior, could also be made to behave as a wave.

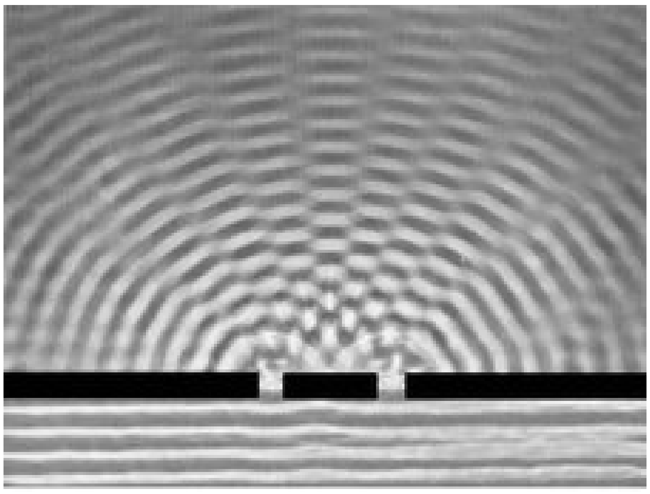

Figure. 2: In Young's double-slit experiment, a light beam is projected against an obstacle through which two minuscule orifices (the "slits") were pierced. As it comes out on the other side, interference patterns characteristic of wave dynamics are observed. The pattern is very complex at first, but quickly settles down to expanding "beams" intervened by bands of darkness. Image retrieved from [20].

In the early 1870's Ernst Abbe, while working for microscope manufacturer Carl Zeiss, realized that just as in Young's experiment, light coming from many periodic structures found in nature (e.g. insect scales) could be modeled as spreading in waves, which at first interact in intricate ways, but then settle down into a stable pattern – the object's image. In this view, the object itself could be abstracted as not a light source, but a *diffraction grating* – an opaque surface full of tiny orifices, blocking and diffracting light as it passes through, warping it in just the right way to produce an image. Abbe's original model was restricted to periodic structures under coherent light conditions, but it has since been generalized into an overall theory of imaging formation comprehending non-periodic objects under semi-coherent light – that is, the objects and light of everyday life.

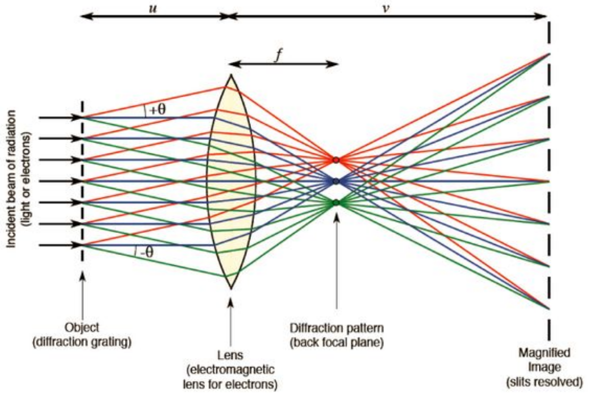
u
v
f
+θ
-θ
Incident beam of radiation (light or electrons)
Object (diffraction grating)
Lens (electromagnetic lens for electrons)
Diffraction pattern (back focal plane)
Magnified image (slits resolved)

Figure. 3: A simplified model of image formation according to Wave Optics. Light coming from a "diffraction grating" (an abstract model for an illuminated object) passes through a lens, producing two interference patterns: a *diffraction pattern* at the back focal plane, and an image of the object at the image plane. Image retrieved from [21].

The resulting theory of *Wave Optics* describes the process of image formation in quite a different way from the ray-tracing account of Geometrical Optics. As seen in Figure 3, light coming from an object (the "diffraction grating") passes through a lens or small aperture, and on its way out produces two distinct interference patterns: a *diffraction pattern* at the back focal plane, and an image of the object at the image plane.

It turns out that the diffraction pattern and the image are related by the mathematics of the Fourier series – more specifically, the diffraction pattern is the Fourier transform of the image. This is a remarkable feature of Wave Optics: many operations on images, such as edge sharpening and detection, smoothing etc. can be implemented by calculating the Fourier transform of an image, performing some simple operation on the transform (e.g. applying a low-pass filter) and then calculating the inverse Fourier transform on the result (see Figure 4 for a typical example). A device meant to enhance or otherwise manipulate captured images could shortcut a lot of processing work by, instead of calculating the Fourier transforms of captured images, getting them "for free" from the back focal plane of its imaging apparatus.

B. The Eye As A Fourier Optics Device

The Chinese philosopher Mozi described the working principles of the camera obscura in the 5th century BCE, yet it wasn't until the 17th century CE that Johannes Kepler first conjectured that they might also apply to the human eye. This is a common trend in the history of science: often, theories developed to describe man-made devices are later found to be applicable to structures or processes that already existed in Nature. Likewise, the usefulness of the Fourier transform as an image manipulation tool has been known to us for

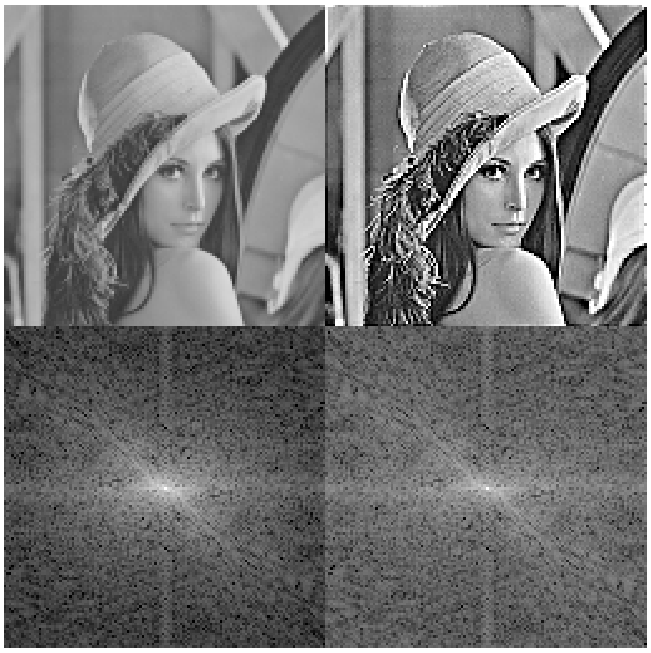

Figure. 4: Image enhancement via Fourier transform. The Fourier transform (bottom left) of the original image (upper left) is changed by increasing the values of all components past a certain frequency threshold (i.e. distance from the center); the result (bottom right) is then passed through the inverse Fourier transform, producing the enhanced image (upper right). Image retrieved from [22].

scarcely a century; how much of a stretch is it to imagine Nature might have got to it before us?

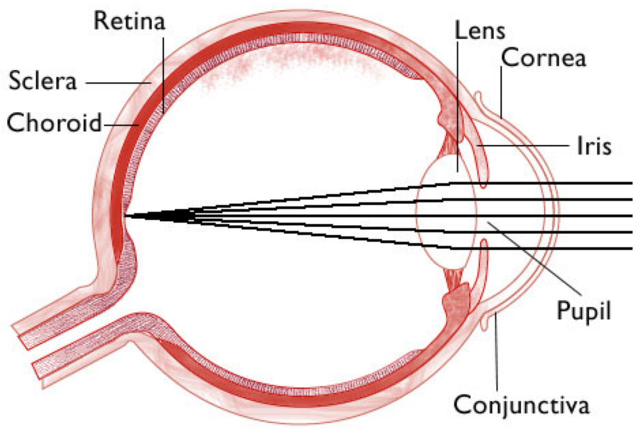

Figure. 5: Longitudinal section diagram of the human eye. Light rays are deflected by the lens and converge on the image plane over the retina. The back focal plane is too close to the image plane to be drawn separately.

Figure 5 shows a longitudinal section diagram of the human eye. Light rays are shown to be deflected by the eye's lens and converge directly on the image plane. The back focal plane is not depicted, but this omission is not accidental. The thin lens equation:

$$\frac{1}{o} + \frac{1}{i} = \frac{1}{f} \tag{2}$$

Where *o, i* and *f* are respectively the object's, the image plane's and the back focal plane's distances from the lens, reveals that, as objects get farther from the imaging apparatus, the back focal plane gets ever closer to the image plane. In the human eye, the retina is on average about $22mm$ from the lens; in practice, virtually the whole world is "in infinity" from our eyes. As a result, both the back focal plane and image plane get stacked over the retina – not actually merged, but too close to be drawn apart in the scale of the illustration above – and accommodation ensures they will be kept that way. The wavelength of visible light is in the range of 390 to 700 nanometers $(10^{-9}m)$, while the outer segment of the retina is about 25 micrometers $(10^{-6}m)$ thick; if we take this for the thickness of photosensors, then there is sufficient space for both the back focal plane and image plane to be recorded.

So it is at least physically possible for the eye to capture the diffraction patterns (that is, the Fourier transforms) of the spatial images projected onto it. But what for? Or in other words, is there any pressing problem of vertebrate vision, that evolution would have to recourse to a tool as sophisticate as Fourier optics in order to solve it? In fact there is, and it's one of the deepest mysteries in the narrative of vertebrate evolution.

It is a well-known fact that the vertebrate retina is "wired backwards" – a relatively thick tangled mass of nerve and glial cells stands between projected images and the photoreceptors, which point away from the eye aperture instead of towards it (see Figure 6 for a SEM image of the human retina, with the photoreceptor layer near the bottom). Over the years this apparently "sub-optimal" (or at any rate, counter-intuitive) arrangement has been explained in various ways – see [7] for a particularly intriguing treatment – as well as fueled much polemic on the mechanics of evolution. One thing is certain, though: while the inner retinal layers are fairly transparent overall, its inhomogeneity causes it to scatter incoming light, producing a blurring effect that should affect our ability to see the world in detail.

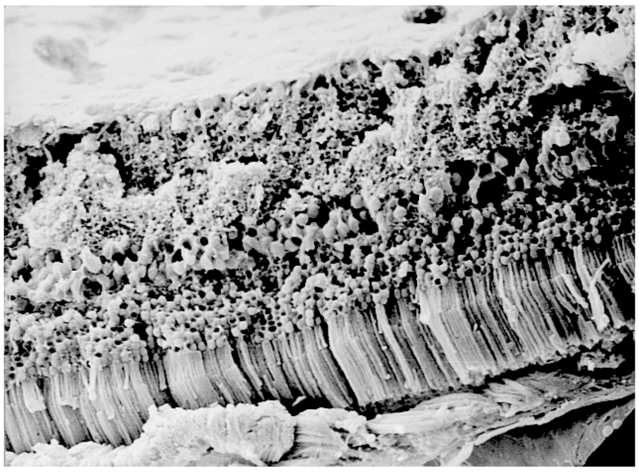

Figure. 6: Scanning Electron Microscope (SEM) image of a longitudinal section of the human retina. The vertical structures at the bottom of the image are photorecptors, while the above layer is formed by nerve and glial cells. Image retrieved from [23].

Obviously we *don't* forever see the world from behind a window of frosted glass, so the blurring effect caused by the inner layers of the retina must be somehow accounted for. This problem is sometimes explained away by the *fovea,* the small spot in the retina that is mostly clear of intervening tissue; however, while the fovea's role of ensuring clearer vision is indisputable, many sharp-eyed vertebrates (including all non-primate mammals) have no fovea as such. It seems unlikely that nearly all vertebrate species would possess functional eyes, yet only a few could picture the world in any level of detail – especially as the sense of vision imparts so much essential information to predator and prey alike.

Could it be that before the evolution of the fovea, vertebrates had already arrived at a way to compensate for the scattering effects of the retina's backward wiring? As far as the optics of the eye are concerned, there is no reason why it couldn't be done. Processing power shouldn't be much of a problem either: since the optical characteristics of the blurring layers are fairly fixed (or at worst, change very gradually over time), it is well within the ability of the vertebrate nervous system to "hard-wire" a Fourier transform into its base visual processing structures. We know that vision is "learned" by the brain during its early stages of development – kittens who have had one eye sewn shut for a number of weeks since birth become "blind" from that eye, even though the eye itself is physically perfect [5] – so that "learning" process could well include the development of a "deblurring" operation.

Another argument in favor of the presence of Fourier processing in the visual pathway is the apparent and very curious correlation between foveated eyes and intelligence. It's not just that primates happen to be the only mammals with foveae. Birds, whose large eyes often feature two or even three foveae, have been found to be remarkably smart – some species can count, use tools or recall specific memories [24] – yet their brains don't seem correspondingly developed. If, however, the bird brain evolved at first mainly to perform the computations required to produce cleared-up retinal images from their Fourier transforms, and one day it was partly relieved from this toil by the evolution of foveae – it seems natural to imagine the surplus processing power would be redirected to other tasks, such as improved cognition.

Unfortunately, this fledging model of visual cognition falls short of one insurmountable obstacle: the so-called "neural bottleneck" constraining information flow from the retina into the brain. The retinal ganglion cells are outnumbered by photoreceptors at a rate of $1/100$; if, as is the current understanding of neuroscience, each neuron can only transmit a single binary signal (either a 0 or a 1) per spike, there is simply not enough bandwidth to relay images – either spatial or Fourier-transformed, and certainly not both – directly to the brain. Of course we could imagine some coding strategy to transmit the data; however, the gap between retinal projection and conscious realization of an image is known to be of $100ms$ in average for humans. With axon transmission speeds no faster than $100m/s$ and an obligatory recovery time of about $1ms$ between spikes, it simply cannot be done in anything approaching the immediacy with which we see the world.

C. *The Brain As A Fourier Processing Machine*

Computer Scientists have long divined that an effective neural architecture cannot be built on single-bit units. The earliest Artificial Neural Network model, the single-layer perceptron, was based on abstract "neurons" that could only output a 0 or 1 for a value; however, despite early promise, it was eventually shown to not work for many classes of problems. When the original model was reworked into the more capable multi-layer perceptron, output range was expanded to any real value between 0 and 1. The same is true for the more recent convolutional networks [25]. Weightless neural networks [26] do work on binary data, but each "neuron" handles whole arrays of bits at a time, not just a single value.

If software-simulated neurons operating at the speed of light could not be made practical before its domain was extended beyond single-bit values, there's seldom reason to believe biological neurons, operating at frequencies as low as $1KHz$, could manage. We may have not found conclusive evidence of it yet – but a multi-valued output might nevertheless lurk inside biological neurons, working in ways outside the reach of our current research instruments. In the multichannel neuron model [7], adjacent sodium channels [16] assemble to form multiple 1-bit "wires" along an axon's length. Neurons then become multichannel cables. Signals of identical measured amplitude can convey quite different numerical values, depending upon which wire they're coming through. Figure 7 illustrates the concept.

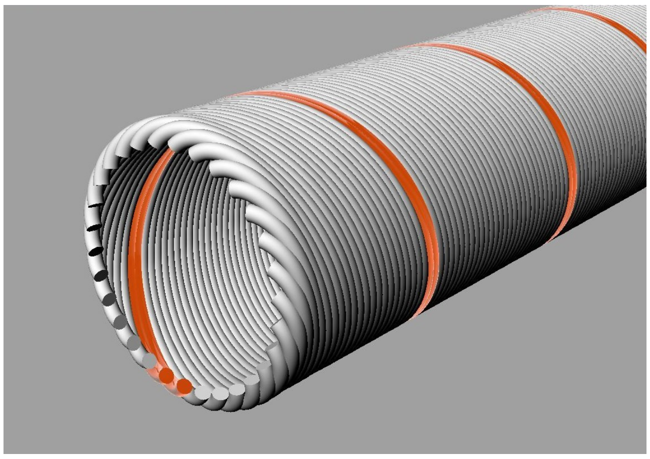

Figure. 7: Model of a multichannel axon membrane. Individual wires are formed by an assembly of adjacent sodium channels. A single marked channel is firing. Image retrieved from [7].

In the multichannel model neurons are both digital, in that output values (i.e. channel numbers) are discrete, and analog, in that channel numbers mirror (are *analogue* to) some physical value. There is no need for any encoding, in frequency or otherwise: a channel number (e.g. "channel 27") can correspond to an environment quantity ("elbow bent 27 degrees", "brightness level 27", etc) directly. If true, this model would have remarkable consequences for our understanding of how brain functions – in particular, vision and cognition – work.

Under the multichannel neuron model, the aforementioned "neural bottleneck" between the eye and the brain doesn't exist. If each neuron has a number (e.g. 300) of distinct channels, and each channel is assigned to different input signals from visual cells, then raw visual information can travel to the brain unimpeded. This data comes in two forms:

- Spatial domain: recorded from the eye's image plane, which, because photoreceptors are tuned to specific ranges within the visible color spectrum [17], takes the form of a mosaic much like a Bayer filter – which was, incidentally, inspired by this very property of visual receptors;
- Frequency domain (i.e. the image's Fourier Transform): recorded from the eye's back focal plane.

If the brain's processing power developed primarily to work on frequency data from the retina, before being partially freed to do other tasks by the evolution of the fovea, it seems reasonable to assume visual memories could be stored as Fourier interference patterns. For a two-dimensional image $a_{m,n}(x, y)$, its discrete Fourier transform $A_{m,n}(u, v)$ can be defined as:

$$A_{m,n}(u, v) = \sum_{y=0}^{n-1} \sum_{x=0}^{m-1} a_{m,n}(x, y) e^{-2\pi i(\frac{xu}{m} + \frac{yv}{n})} \quad (3)$$

And the inverse transform from $A_{m,n}(u, v)$ to $a_{m,n}(x, y)$ can be defined as:

$$a_{m,n}(x, y) = \frac{1}{mn} \sum_{v=0}^{n-1} \sum_{u=0}^{m-1} A_{m,n}(u, v) e^{2\pi i(\frac{xu}{m} + \frac{yv}{n})} \quad (4)$$

On both cases, each output term can be computed independently of all others from the input alone; with sufficient resources, the whole transform can be almost instantly computed as the juxtaposition of a great number of parallel, single-step computations. Likewise, representing memories as interference patterns allows them to be stored in a fail-resistant form, and processed both incrementally and in parallel. In fact, so efficient is this storage system that it doesn't require any particularly clever indexing strategy – memories are stored linearly, indexed by its position in the time stream. Information is stored wholesale, and when specific fragments are required, they can be extracted by operating on the interference patterns.

The brain model enabled by the multichannel neuron can be thus summarized as a computing machine where:

- Input is provided as pairs of matrices – a two-dimensional mosaic in the spatial domain and its Fourier transform in the frequency domain;
- Successive inputs are stored linearly, indexed by time of acquisition;
- Specific constructs within inputs are specified as extracting operations;
- Information is processed mainly in the frequency domain, by manipulating the Fourier transform, with occasional transforms back to the spatial domain in order to relate the computations to the physical world.

Even though its validity as a model of brain function rests on the concept of a multichannel neuron, clearly there is nothing about this architecture that actually *requires* a neural processing substrate to work. This provides a convenient path for experimentation: first implement the model without regarding its neurological roots (which at this point are just conjecture anyway), and later, if results are encouraging, work in the direction of more closely matching the biologic domain.

In the next section we begin to devise that first implementation.

IV. Architecture of Skeye

Skeye is a new programming framework developed specifically for the purpose of studying the multichannel neuron model. Currently in its early stages of development, it is an Open Source project available on the web [27]. Skeye is developed in Python, on top of third-party packages NumPy [28] and SciPy [29]. The combination of Python, NumPy and SciPy makes for agile development without sacrificing speed too much. Its architecture can be divided in two layers: the base package `skeye` implements the platform's core logic as a loosely-coupled library of processing functions, which in turn are used as building blocks to implement the `skeye.cogs` framework. The motivation for this division is that no one framework will be applicable to all problems: therefore, rather than try and shoehorn problems into a model that doesn't quite fit, it's best if the user can take a step back and tap into a set of building blocks from which she can build a more customized solution. It also provides a foundation for additional frameworks to be built into the platform in the future.

Functions in the base package can be divided in three categories:

- Data processing algorithms – these implement the biological- and cognitive-inspired algorithms that prompted the development of Skeye;
- Data processing facilities – these implement common operations that simplify the construction of more complex functions;
- Programming facilities – these implement common programming patterns (e.g. conversions between data types) that simplify basic programming tasks.

Data processing algorithms include functions such as `bayer(image)`, which converts a 3-dimensional color image to a 2-dimensional Bayer mosaic, and `templatesearch(image, template)`, which searches for a template within a larger image. Data processing facilities include `angle(a, b)`, which returns the cosine of the angle between two vectors `a` and `b`. Programming facilities include `snapshot(image)`, which converts an image in the Python Imaging Library (PIL) format to a NumPy equivalent, and `effectors.desktop(command, *arguments, **options)`, which implements a small set of desktop automation operations.

The `skeye.cogs` package is centered around the class `cogbot` (short for `cog`nitive ro`bot`). Cogbots are simple programmable agents that can be described in terms of a pre-defined *memory* that defines the agent's initial state, and a sequence of *commands* that implement its behavior, interacting with both the memory and surrounding environment as they run. When a Cogbot is activated, it executes its commands sequentially, returning a list of the commands' outputs after it finishes.

Any kind of object can be specified as the bot's memory, but since commands will often need to query it for data, logically the object must conform to their expectations. The default memory class in `skeye.cogs` is the `reelmemory`, which behaves just like a linked list. Individual memories must of course be represented by some kind of object; currently, the only pre-defined class fulfilling this role is `visualmap`, which holds together a visual snapshot (i.e. an image) and descriptions of objects found within it. Objects are described by sequences of operations `what` and `where`: `what` "differentiates" an object from the image, whereas `where` "integrates" a previously differentiated object into a larger one.

Commands are expected to be *callables* that accept a single argument `context`, which is set to the Cogbot itself. Both functions and callable objects can be used, therefore customizable "command classes" can be created – and are in fact an integral part of the `skeye.cogs` package. The special command classes `batch` and `latch` take sequences of commands as instantiation arguments: when executed, `batch` runs every command passing them the same arguments it was invoked with and returns the list of outputs, whereas `latch` passes the input arguments plus the output of the previous command in the sequence, and returns the output of the last command.

V. Template Matching in Skeye

The problem of *template matching* concerns locating, within a comparatively much larger scene, an object that (according to some pre-defined metric) "resembles" or *matches* a visual prototype, or *template*; the object may differ significantly from its template, or not be present at all. Efficient template matching is critical to applications such as automated surveillance, image categorization and robot navigation [30]; it is therefore a relevant test-ground for the theories embodied by Skeye.

The function `skeye.templatesearch(image, template)` searches for a template within a larger image. It returns a 3-tuple containing the matched image fragment, its top left corner's coordinates within the larger image, and the degree of similarity between the template and match as a real number between 0 and 1 (where 1 means a perfect match). It implements a normalized cross-correlation algorithm based on the convolution theorem [31].

For an image $f_{m,n}(x,y)$ and template $t_{p,q}(x,y)$, the algorithm works by first normalizing both image and template by their respective means:

$$F(x,y) = f(x,y) - \overline{f} \tag{5}$$

$$T(x,y) = t(x,y) - \overline{t} \tag{6}$$

Once image and template are normalized, their discrete Fourier transforms are calculated, the Hadamard (element-wise) product of the complex matrices is taken, and the result is transformed back to the spatial domain:

$$M(x,y) = \mathcal{F}^{-1}(\mathcal{F}(I) \circ \mathcal{F}(T)) \tag{7}$$

The resulting "match map" $M(x,y)$ – which describes the template's matching degree to each possible location within the image – is queried for the coordinates of the biggest value. These are taken as the match's top-left coordinates, and starting there a section the size of the template is taken from the image:

$$(u,v) = \underset{x,y}{\arg\max}\, M(x,y) \tag{8}$$

$$m = f[(u,v),\ldots,(u+p,v+q)] \tag{9}$$

The cosine of the angle between match and template is calculated:

$$\cos\theta = \frac{m \cdot t}{\|m\|\|t\|} \tag{10}$$

Finally, (u,v), m and $\cos\theta$ are returned as the results of the search procedure. Figure 8 illustrates the process.

It's possible to imagine this process being performed, in the context of the multichannel neuron model, by an interwoven collection of processing nodes. The Fourier transforms of the image and the template would have already been captured from the eye's back focal plane, and would be stored across neuron "sheets" of one neuron per pixel; a third layer would perform the multiplication, and a fourth, the inverse transform. Finally a fifth layer would use the similarity outputs to direct extraction of the match from the image's spatial representation, also captured previously.

Some machinery is still required to collect and manage templates. In conformance to Harris' visual memory model, `visualmap` objects store whole images, along with labeled `descript` objects which describe how templates are separated from recorded images in terms of "differentiation" (`what`) and "integration" (`where`) operations. A differentiation is just a cropping operation: the corresponding `what` object stores the row and column ranges which demarcate the template. Integration is more involved – it allows, given the coordinates of some template, to match a larger template to which it belongs. These larger templates are described as labeled row/column ranges in `visualmap` objects; `where` objects then look among these ranges to find which of them encompasses the smaller template.

In fact, a `descript` object containing a sequence of `what` and `where` operations doesn't just specify a template: since both classes also encapsulate the template-matching logic that implements differentiation and integration, it also constitutes an "executable" sequence of template-matching operations – or in other words, a template-matching program.

When a Cogbot is in execution, a command may recover a `visualmap` from the bot's memory, then query it passing an input image and template label. The image is then turned into a top-level `percept` object and passed along the `visualmap` to the first operation (likely a `what`) in the labeled `descript`: using information from its parent `visualmap`, the operation produces a match in the form of

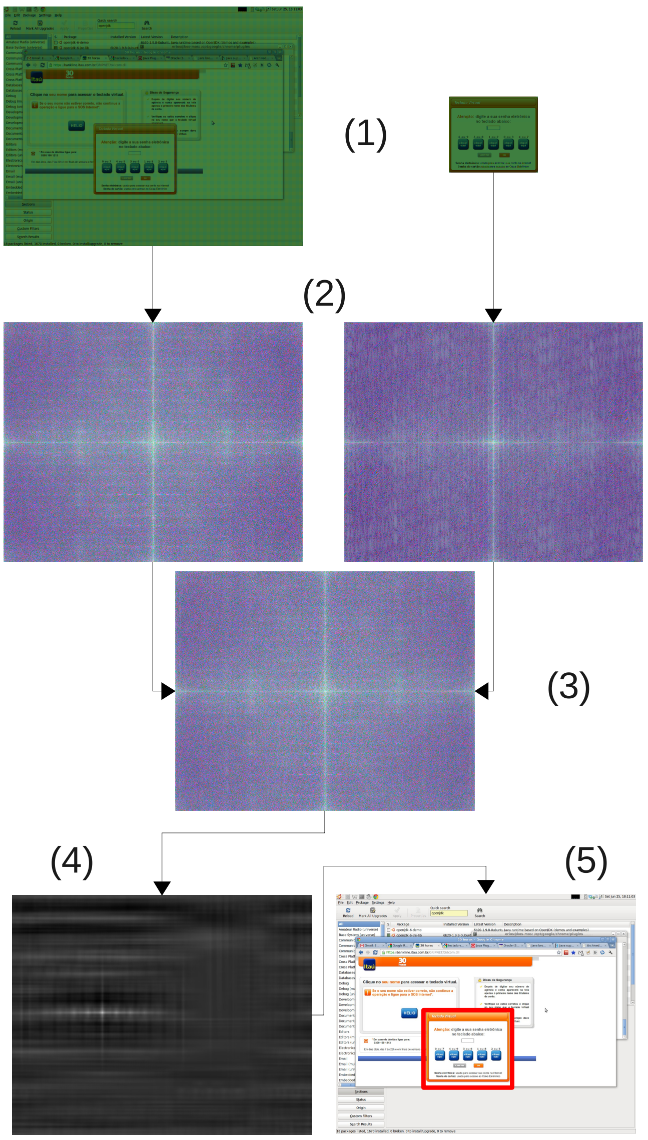

Figure. 8: Fourier template search process. Bayer-filtered image and template (1) are Fourier-transformed (2) and the resulting interference patterns are multiplied element-wise (3). The result is then inverse-transformed (4), producing a "match map" which is used to locate the template within the image (5).

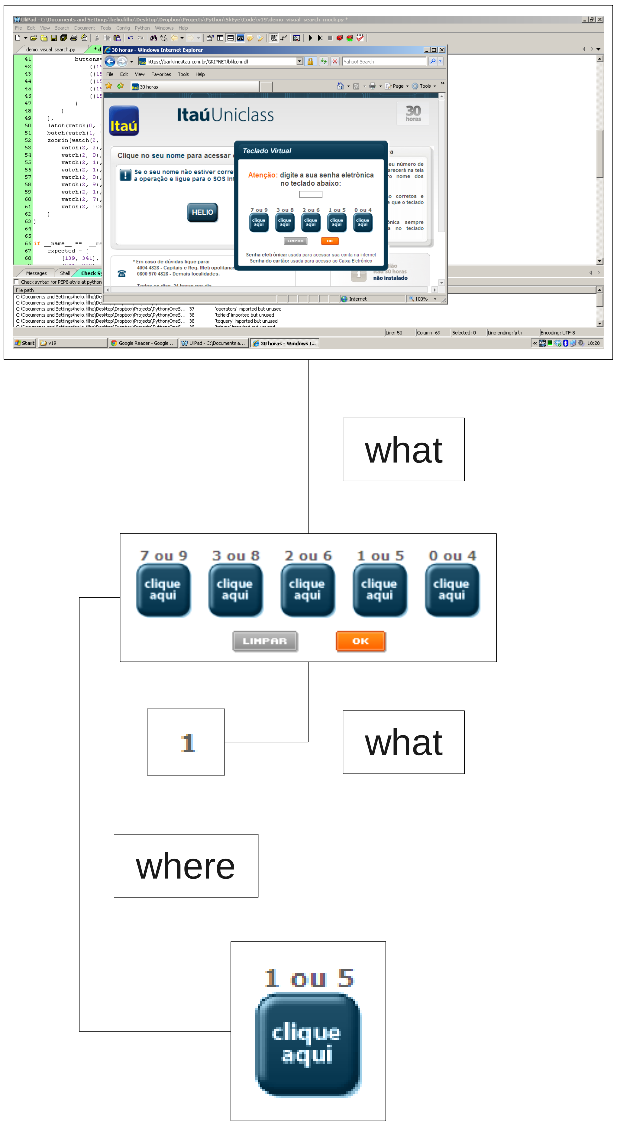

Figure. 9: Example of a template-matching program at work. The top-level percept reproduces Itau's home banking web site's login page with the virtual keyboard displayed: the program intends to recover the coordinates of the button corresponding to the digit "1". The first `what` operation differentiates the virtual keyboard from the overall image; the second `what` differentiates the label "1" from the keyboard; and the `where` operation integrates the label into its button. Full lines represent hierarchical relations between percepts.

a second-level `percept`, related to its parent by an *offset*. This second-level `percept` is passed to the next operation, producing a third-level `percept`, and so on to the end of the sequence; the last `percept` produced is returned as the sequence's output.

Figure 9 illustrates how a template-matching program works. The input image reproduces the Brazilian bank Itau's web site's login page, with its virtual keyboard displayed; the program intends to recover the coordinates of the button corresponding to the digit "1". The first `what` operation differentiates the virtual keyboard from the overall image; the second `what` differentiates the label "1" from the keyboard; and the `where` operation integrates the label into its button.

VI. Case Study: Home Banking Automation

Home banking websites often implement login procedures meant to be performed only by humans [32], thus providing a good benchmark to AI systems and, in particular, machine vision systems which intend to emulate human traits. Also, as actual deployed services routinely used by people, they provide an interesting balance between a controlled environment and real-world conditions. For practical reasons, this case study is based on the home banking site of the Brazilian bank Itau (`http://www.itau.com.br`). To login into the Itau website the following steps must be performed:

1. Locate the *AGÊNCIA* text box and click on it;

2. Enter the bank branch code and account number and click the *OK* button, or alternatively hit `<ENTER>`;

3. Click the button containing the account owner's name;

4. Enter the account's password on the virtual keypad that shows up and click the *OK* button.

Of all steps above, the last one is the most complex. The numeric keypad contains only five keys for the ten numeric digits; each key corresponds to two digits, which change position randomly on each login; pairings aren't fixed; and the keypad itself may appear at different locations within the login page. Therefore, not only no fixed button sequence can be established – even relying on a fixed position for the *buttons* themselves is temerarious, as they can be drawn somewhere else on subsequent logins.

```
bot = cogbot(
 reelmemory(
  visualmap(
   'itau00.png',
   descript('agencia',
    what((88, 110), (263, 533), precision=0.9),
    what((88, 110), (318, 363))
   )
  ),
  visualmap(
   'itau01.png',
   descript('name',
    what((148, 391), (15, 438), precision=0.98),
    what((200, 391), (15, 438)),
    what((279, 328), (190, 265))
   )
  ),
  visualmap(
   'itau02.png',
   descript('keyboard',
    what((499, 806), (385, 741), precision=0.94)
   ),
   descript(0, what((655, 675), (419, 430)), where('bt')),
   descript(1, what((655, 675), (599, 610)), where('bt')),
   descript(2, what((655, 675), (659, 670)), where('bt')),
   descript(3, what((655, 675), (539, 550)), where('bt')),
   descript(4, what((655, 675), (479, 490)), where('bt')),
   descript(5, what((655, 675), (691, 702)), where('bt')),
   descript(6, what((655, 675), (571, 582)), where('bt')),
   descript(7, what((655, 675), (451, 462)), where('bt')),
   descript(8, what((655, 675), (631, 642)), where('bt')),
   descript(9, what((655, 675), (511, 522)), where('bt')),
   descript('OK', what((723, 744), (585, 630))),
   bt=(
    ((156, 216), (34, 81)),   # Button "0 ou 7"
    ((156, 216), (94, 141)),  # Button "4 ou 9"
    ((156, 216), (154, 201)), # Button "3 ou 6"
    ((156, 216), (214, 261)), # Button "1 ou 8"
    ((156, 216), (274, 321))  # Button "2 ou 5"
   )
  )
 ),
 automate('run',
  r'"C:\Program Files\Internet Explorer\iexplore.exe" ' +
  '"http://www.itau.com.br"'
 ),
 latch(locate(0, 'agencia', delay=3), click(Left)),
 automate('write', '0863963445\n'),
 latch(locate(1, 'name', delay=3), click(Left)),
 zoomin(locate(2, 'keyboard', delay=3),
  latch(locate(2, 2), click(Left)),
  latch(locate(2, 0), click(Left)),
  latch(locate(2, 1), click(Left)),
  latch(locate(2, 1), click(Left)),
  latch(locate(2, 0), click(Left)),
  latch(locate(2, 9), click(Left)),
  latch(locate(2, 1), click(Left)),
  latch(locate(2, 7), click(Left)),
  latch(locate(2, 'OK'), click(Left))
 )
)
```

Figure. 10: Final version of the automation script for login into the Itau home banking site. `import` directives were omitted to save space. Account data has been changed for obvious reasons.

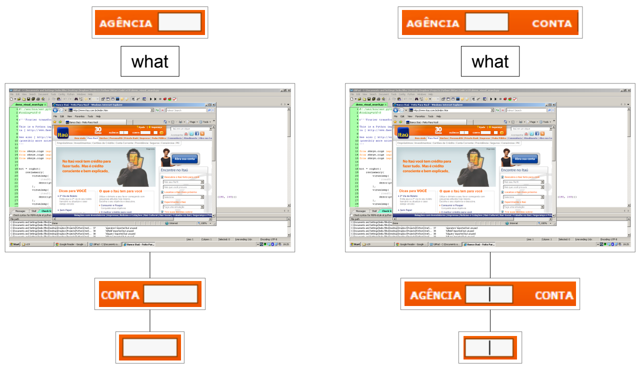

Figure. 11: The "AGÊNCIA" text box is too inexpressive on its own to guarantee a correct match; expanding the initial template to include both the "AGÊNCIA" and "CONTA" labels, and then narrowing into the "AGÊNCIA" text box, solves this problem. The second `what` operation has been omitted for brevity.

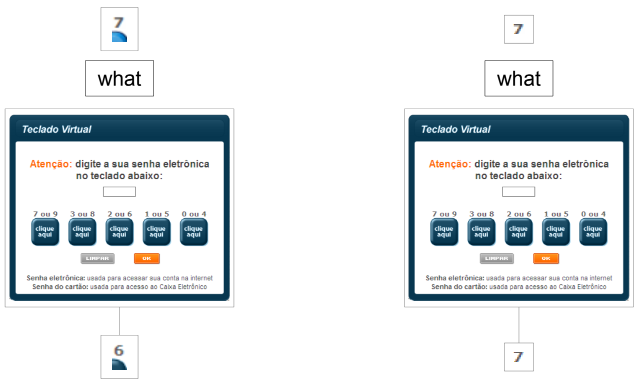
7
what
Teclado Virtual
Atenção: digite a sua senha eletrônica no teclado abaixo:
7 ou 9
3 ou 8
2 ou 6
1 ou 5
0 ou 4
clique aqui
LIMPAR
OK
Senha eletrônica: usada para acessar sua conta na internet
Senha do cartão: usada para acesso ao Caixa Eletrônico
6
7
what
7

Figure. 12: Including part of the double digit button to the template causes the digit "7" to be mismatched; narrowing the template to only include the digits solves the problem.

Clearly the main problem is how to determine the positions of all relevant visual controls within the login page. It turns out that `descript` sequences of `what` and `where` operations can be easily created to locate such controls; from there, it's just a matter of employing a desktop automation API to interact with the GUI and perform the necessary actions.

A second, deceptively prosaic problem, is the time it takes for the web page to load. Because loading times are hard to predict (depending on specifics of the client computer's specs, bandwidth and transient network load, etc), making the agent wait a fixed time length after requesting a web page would be impractical; a long waiting time would have to be set, lest the agent risked reacting too early. Therefore, some pooling mechanism is necessary for the agent to check whether the web page it requested has finished loading, before moving on.

On `skeye.cogs`, this pooling mechanism is implemented in two stages. The `what` class accepts an optional argument `precision`: if the similarity between the template and its match is less than this value, an exception `failure` is raised. By its turn, the command `locate` accepts a time delay in seconds, an index into the Cogbot's memory and a label: it sleeps for the duration of the delay, then try to execute the `descript` identified by the label, repeating the two steps each time a `failure` is raised.

The final version of the automation script to login into the Itau home banking site is given in Figure 10 (account data has been changed for obvious reasons). Training screenshots from the website were taken on a Chromium web browser running on Linux, but tests were performed on Internet Explorer running on Windows, thus allowing for some variation in the size and proportion of page elements. Also, on a happy coincidence, the site's color palette was changed shortly after the first batch of screenshots was collected; this new layout was also incorporated into the tests. On both cases the script consistently succeeded in finding the required UI controls and logging into the bank account. It also was remarkably fast: the full script takes an average $20s$ (plus page loading times) to run on a $2Ghz$ Intel Core i3 system, while an equivalent raster search on the same system took more than $1h$.

During development of the automation script there were several instances of mismatch between templates and the regions found in the test screenshots. There were essentially two reasons for that:

1. If the template is too small, slight feature differences may cause it to be mismatched to an inequivalent region in the test screenshot. This can be solved by starting with a larger template, and using successive `what` operations to narrow the found region to the required percept;

2. On the other hand, a mismatch may also happen if a template includes some "generic" feature that is much larger or otherwise stronger than other, more characteristic features. In this case, the solution is to shrink the template to leave the "generic" feature out, and use a `where` operation to include it back into the search result.

Figure 11 and Figure 12 illustrate these cases. On the first case, the "AGÊNCIA" text box is too inexpressive on its own to guarantee a correct match; expanding the initial template to include both the "AGÊNCIA" and "CONTA" labels, and then narrowing into the "AGÊNCIA" text box, solves this problem. On the second case, including part of the double digit button to the template causes the digit "7" to be mismatched; narrowing the template to only include the digit solves the problem.

VII. Conclusion

The Skeye framework represents a first step towards evaluating the multichannel neuron model in a practical context, with so far encouraging results. The architecture Skeye implements, which draws heavily from its supported model of brain function, was found to be satisfactory for the task at hand; its visual algorithms were robust on the face of changing environment conditions. Nevertheless it is just a start – by no means this experiment can be taken as definitive validation of the multichannel neuron model.

While the reported test case included various real-world variables concerning a Graphical User Interface (GUI) automation script, such as differences in rendering, irregular response times and unexpected layout changes, it's still a far controlled environment when compared to object search tasks in the physical world, where differences in lighting, scale and rotation are also possible. Therefore, research ways to abstract such differences in the search process would be a natural next step; experiments of a more quantitative nature are also in order, even in the context of computer environments.

In the long term, ways to reimplement the framework according to a more "connectionist" approach should be evaluated. Today Skeye deliberately disregards its neural roots, but clearly, if its architecture continues to evolve, encompassing ever more diverse and complex cognitive tasks, eventually the question of the degree to which it resembles real brains will arise. An implementation built upon a neural model would be a valuable asset at this point, serving as a guide to physiological experiments that would ultimately confirm or deny the validity of the multichannel neuron model.

On the other hand, a practical opportunity exists for a software framework based on the Fourier transform. The recent popularization of multimedia-enabled "smartphones" and "tablets" has led the processor industry to invest heavily on the development of low-power, multicore architectures matching the requirements of mobile consumer electronics. Increasing the number of instructions a processor can execute simultaneously seems like an elegant way to improve nominal computing capacity while keeping cycle frequency down (and thus conserving power), but it has the side-effect of passing to the software the responsibility of implementing parallelism – a task notoriously hard in the general case. The Fourier transform, however, can be made almost as parallel as the underlying hardware allows with relative ease. Even if the multichannel neuron and the cognitive architecture it supports are eventually found to not be valid biological models, Skeye may still become useful as a data processing tool fit to run on these increasingly ubiquitous computers.

Acknowledgment

We would like to thank Michael Cesare Gianturco for his editorial suggestions, as well as Conselho Nacional de Desenvolvimento Científico e Tecnológico-CNPq-Brasil (grants 552630/2011-0, 309831/2007-5, and 309172 /2009-8) and Fundação de Amparo à Pesquisa do Espírito Santo - FAPES-Brasil (grant 48511579/2009) for their support to this research work.

References

[1] J. Hawkings and S. Blakeslee, *On Intelligence*. Times Books, 2004.

[2] E. D. Adrian, "The impulses produced by sensory nerve endings," *J. Physiol*, vol. 61, pp. 49–72, 1926.

[3] H. v. Helmholtz, "Vorläufiger bericht über die fortpflanzungs-geschwindigkeit der nervenreizung," *Archiv für Anatomie, Physiologie und wissenschaftliche Medicin*, pp. 71–73, 1850.

[4] D. Hartline and D. Colman, "Rapid conduction and the evolution of giant axons and myelinated fibers," *Current biology : CB doi:10.1016/j.cub.2006.11.042*, vol. 17, no. 1, pp. R29–R35, 2007.

[5] D. e. a. Purves, *Neuroscience*, 3rd ed. Sinauer Associates, 2004.

[6] F. Rieke, D. Warland, R. de Ruyter van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code*, 1st ed. Cambridge, MA: MIT Press, 1997.

[7] J. Harris, "Rewiring neuroscience," 2010. [Online]. Available: http://www.rewiring-neuroscience.com/

[8] D. Gardner and G. M. Shepherd, "A gateway to the future of neuroinformatics," *Neuroinformatics*, vol. 2, pp. 271–274, 2004, 10.1385/NI:2:3:271. [Online]. Available: http://dx.doi.org/10.1385/NI:2:3:271

[9] H. Filho and A. Souza, "A biology-based template-matching framework," in *Intelligent Systems Design and Applications (ISDA), 2012 12th International Conference on*, 2012, pp. 326–331.

[10] F. Rosenblatt, "The Perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Learning internal representations by error propagation, pp. 673–695.

[12] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[13] J. Schmidhuber, "Goedel machines: Self-referential universal problem solvers making provably optimal self-improvements," *CoRR*, vol. cs.LO/0309048, 2003.

[14] G. Dileep, "How the brain might work: a hierarchical and temporal model for learning and recognition," Ph.D. dissertation, Stanford, CA, USA, 2008.

[15] R. W. Price, "Hierarchical temporal memory cortical learning algorithm for pattern recognition on multi-core architectures," Master's thesis, Portland State University, 2003.

[16] B. Hille, *Ion Channels of Excitable Membranes*. Sinauer, Sunderland, MA, 2001.

[17] G. Svaetichin, *Spectral response curves from single cones*, ser. Acta physiologica Scandinavica: Suppl. acta physiologica, 1956. [Online]. Available: http://books.google.com/books?id=7bYgGwAACAAJ

[18] H. Volkmann, "Ernst abbe and his work," *Appl. Opt.*, vol. 5, no. 11, pp. 1720–1731, Nov 1966. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-5-11-1720

[19] A. Bosco, "History and consequences: Photography," 2011. [Online]. Available: http://ambosco.edublogs.org/2011/02/20/history-consequences-photography/

[20] B. Crowell, "Light and matter," 2011. [Online]. Available: http://www.lightandmatter.com/html_books/lm/

[21] U. of Cambridge, "Image formation," 2012. [Online]. Available: http://www.doitpoms.ac.uk/tlplib/diffraction/image.php

[22] J. Brayer, "Introduction to fourier transforms for image processing." [Online]. Available: http://www.cs.unm.edu/ brayer/vision/fourier.html

[23] U. of Delaware, "Eye ultrastructure." [Online]. Available: http://www.udel.edu/biology/Wags/histopage/empage/eey/eey.htm

[24] R. G. Cook, "Avian visual cognition," 2001. [Online]. Available: http://www.pigeon.psy.tufts.edu/avc/toc.htm

[25] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 1995.

[26] H. Perroni Filho and A. F. De Souza, "Vg-ram wnn approach to monocular depth perception," in *Proceedings of the 17th international conference on Neural information processing: models and applications - Volume Part II*, ser. ICONIP'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 509–516. [Online]. Available: http://dl.acm.org/citation.cfm?id=1939751.1939821

[27] H. Perroni Filho, "Skeye," 2012. [Online]. Available: https://github.com/xperroni/Skeye

[28] T. E. Oliphant, "Python for scientific computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 10–20, 2007. [Online]. Available: http://link.aip.org/link/?CSX/9/10/1

[29] E. Jones, T. E. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001-. [Online]. Available: http://www.scipy.org/

[30] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley, 2009.

[31] R. Bracewell, *The Fourier Transform and Its Applications*, 3rd ed. McGraw-Hill Science/Engineering/Math, 1999.

[32] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: using hard ai problems for security," in *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, ser. EUROCRYPT'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 294–311. [Online]. Available: http://dl.acm.org/citation.cfm?id=1766171.1766196

Author Biographies

Helio Perroni Filho Alumnus of the Laboratório de Computação de Alto Desempenho (LCAD – High Performance Computing Laboratory) at the Universidade Federal do Espírito Santo (UFES), Brazil. Received B. Sc. in Computer Science in 2004 and M. Sc. in Computer Intelligence in 2010, both from Universidade Federal do Espírito Santo (UFES), Brazil. Has more than ten years of experience in the IT industry. His research is concerned with development and application of computer vision technologies, with a focus on biology-inspired techniques.

Alberto Ferreira De Souza Associated Professor of Computer Science and Coordinator of the Laboratório de Computação de Alto Desempenho (LCAD – High Performance Computing Laboratory) at the Universidade Federal do Espírito Santo (UFES), Brazil. Received B. Eng. (Cum Laude) in electronics engineering and M. Sc. in systems engineering and computer science from Universidade Federal do Rio de Janeiro (COPPE/UFRJ), Brazil, in 1988 and 1993, respectively; and Doctor of Philosophy (Ph.D.) in computer science from the University College London, United Kingdom, in 1999. He has authored/co-authored one USA patent and over 90 publications. He has edited proceedings of six international conferences (three IEEE sponsored conferences), is a Standing Member of the Steering Committee of the International Conference in Computer Architecture and High Performance Computing (SBAC-PAD), and Senior Member of the IEEE.