VISUAL LANGUAGE RECOGNITION WITH A FEED-FORWARD NETWORK OF SPIKING NEURONS

Craig Rasmussen and Garrett Kenyon
Los Alamos National Laboratory
CCS-7, B287, Los Alamos, NM 87545

Matthew Sottile
Galois, INC
421 SW 6th Ave. Suite 300, Portland, OR 97204

ABSTRACT

An analogy is made and exploited between the recognition of visual objects and language parsing. A subset of regular languages is used to define a one-dimensional 'visual' language, in which the words are translational and scale invariant. This allows an exploration of the viewpoint invariant languages that can be solved by a network of concurrent, hierarchically connected processors. A language family is defined that is hierarchically tiling system recognizable *(HREC)*. As inspired by nature, an algorithm is presented that constructs a cellular automaton that recognizes strings from a language in the *HREC* family. It is demonstrated how a language recognizer can be implemented from the cellular automaton using a feed-forward network of spiking neurons. This parser recognizes fixed-length strings from the language in *parallel* and as the computation is pipelined, a new string can be parsed in each new interval of time. The analogy with formal language theory allows inferences to be drawn regarding what class of objects can be recognized by visual cortex operating in purely feed-forward fashion and what class of objects requires a more complicated network architecture.

KEYWORDS

pattern recognition, neural networks, parallel parsing algorithms, parallel computation, machine learning

1. INTRODUCTION

We make an analogy between the recognition of objects by visual cortex and the recognition of strings by language parsers. Each performs a recognition task; one on two-dimensional patterns of bits in an image and the other on one-dimensional patterns of characters in a string. This analogy is made more precise by research on two-dimensional picture languages. Blum and Hewitt (1967) developed the first automaton model for recognizing picture languages followed later by the development of tiling systems that can also recognize these languages (Giammarresi and Restivo 1992, 1997).

We draw attention to an ubiquitous feature of biological, visual object recognition; visual objects remain recognizable at any location and scale in the visual scene. For example, an oncoming car is still recognized as a car, even as the relative size of the car increases as it draws closer. In order to make progress, we reduce the complexity of a two-dimensional visual system by considering only one-dimensional strings. As suggested by the visual system, we present a family of 'visual' languages representing fixed length strings that are viewpoint (scale and translation) invariant.

The formalism for defining visual languages is adapted from one-dimensional versions of *tiling recognizable picture languages (REC)* and is presented in Section 2. Strings in this language family are finite, can be defined by regular expressions, and are thus recognizable by finite automata. We present an algorithm for constructing a cellular automaton (CA) that will recognize a visual language in parallel, each automaton in the two-dimensional grid operating on separate substrings concurrently. We note that the CA so constructed has similarities with the hierarchical structure in visual cortex. Ambiguities arising from tiling systems and viewpoint invariance are discussed in Section 3 and we conclude the paper by briefly speculating on the relevance of formal language theory to computation in visual cortex and by suggesting further work.

2. METHODOLOGY

The visual language family is perhaps best understood in terms of two-dimensional *tiling recognizable picture languages,* denoted by *REC* (see Anselmo et al 2009 for details). The *REC* language family is defined by a tiling system (Σ, Γ, Θ, π) where Σ and Γ are finite alphabets, Θ is a finite set of tiles over Γ, and π : $\Gamma \rightarrow \Sigma$ is a projection. A tile is a picture of size (2,2) (2 rows and 2 columns) and $B_{2,2}(p)$ is the set of all sub-pictures of size (2,2) that produce a picture *p*.

A key point is the concept of a *local* language $\mathscr{L}_i \subseteq \Gamma_i^{**}$ where Γ_i^{**} is the set of all possible pictures over the *local* alphabet Γ_i. A language $\mathscr{L} \subseteq \Sigma^{**}$ is tiling recognizable if there exists a tiling system projecting $\mathscr{L}_i$ to $\mathscr{L}$. We consider only the one-dimensional analog of *REC*, the class of regular string languages. In one dimension, a tile as described in Giammarresi and Restivo (1997) reduces to $B_{1,2}$. However, we extend this slightly to consider tiles of size (1,*n*). In particular, we examine local languages expressed in terms of *overlapping* tiles of size (1,3). The tiles are made overlapping in order to more closely conform to the visual system as explained below.

Visual processing in the brain takes place in layers of neurons. In a feed-forward model, as neurons in one layer spike they pass on this information via synapses to neurons in the successive layer. A two-dimensional representation of this layering is shown in Figure 1, where each neuron in layer L_1 receives afferent input from a neighborhood of neurons in layer L_0. For all the cases explored here, the size of the neighborhood is 3, although in visual cortex the neighborhood can be larger (Alonso et al 2001).

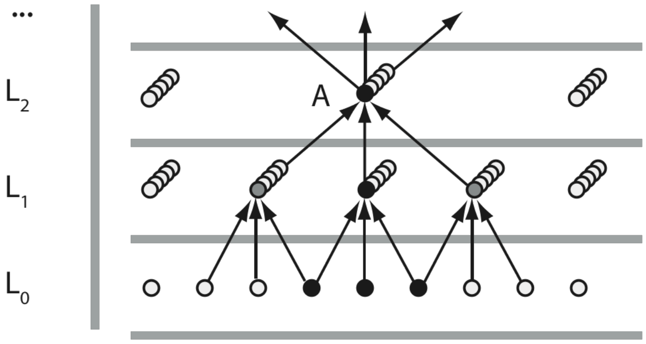

Fig. 1. A two-dimensional network of spiking neurons connected by modifiable synapses.

In Figure 1, three neurons in layer L_0 are drawn as solid black circles indicating they are spiking at some frequency. This information is transmitted upward causing three neurons in layer L_1 to spike in turn, although the middle neuron is more active as all of the neurons in its afferent neighborhood are spiking strongly. This causes neuron A in layer L_2 to also begin spiking. It is important to note the geometry in Figure 1. While there are two spatial dimensions (*x,L*), the neurons are actually distributed in a three-dimensional lattice with neurons in L_1 and above stacked in the third dimension. This stacking allows neurons to be tuned to respond to different combinations of inputs. If neural activity in L_0 is encoded as a string, with *spiking* encoded as a 1 and *not spiking* as a 0, then a neuron of tuning 0_1 in L_1 might respond to the input string '000' while another neuron at the same spatial location might respond to '111'.

Also note that the number of spatial sites in *x* drops by a factor of 2 in each successive layer. This *reduction* in *x* allows an *increase* in the number of different tunings that can be detected at a layer without the total number *N* of neurons in a layer growing appreciably. In effect, specific spatial information input to L_0 is transformed into another representation by the computation being done within each layer. More on this important concept will be provided later.

2.1 Finite-State Machine Construction

As seen in Figure 1, visual cortex can be viewed as a pipeline with two-dimensional visual information flowing in at the bottom and working its way up the visual hierarchy. By contrast language processing (parsing) by compilers takes input as a string of characters and processing takes place one character at a time. Thus visual processing is inherently parallel while string parsing (recognition) is most commonly performed serially. We explore the possibility of recognizing an input string in parallel by analogy with visual cortex. Assuming overlapping tiles $B_{1,3}$ of size 3, the tiles can be arranged spatially to provide the connectivity pattern shown in Figure 1. We also assume a finite input string and that the number of processing elements (neurons), N_0, in the first layer L_0 is greater than or equal to the length l of the input string. Furthermore, additional processing elements are distributed in multiple layers L_i, with the total number of layers (system height) M such that $M \geq log_2 (N_0)$. Each processing element $n_{i,j}$ receives input from 3 elements in the preceding layer, $L_{i\text{-}1}$.

A finite input string from language $\mathscr{L}$ is over the alphabet $\Sigma=\{0,1\}$ where $\mathscr{L} \subset \Sigma^*$. A local language $\mathscr{L}_1$ is defined over a local alphabet Γ_1 such that $\mathscr{L}$ is a projection of $\mathscr{L}_1$. In general Γ_1 is constructed from the set of tiles {'000', '001', '010', '011', '100', '101', '110', '111'} of all possible combination of inputs from L_0. Likewise, additional local languages $\mathscr{L}_i$ are defined over alphabets Γ_i at each layer in the hierarchy. The local alphabets are subsets of the tiles B in the preceding layer. The length of strings processed by a layer satisfies the recurrence relationship $l_i = l_{i\text{-}1}/2$. At the top of the hierarchy there is one final local alphabet Γ_h. This alphabet corresponds to a set of neurons $n_{h,1}$, at a single spatial site, whose receptive fields span the entire input space in L_0. This set is able to recognize strings in $\mathscr{L}$.

We define a language $\mathscr{L} \subset \Sigma^*$ to be *hierarchically tiling system recognizable (HREC)*, if it can be obtained from successive projections of local languages $\mathscr{L}_i$. This system (local tiling sets plus projections) is effectively a description of the state-graph of a cellular automaton that recognizes a visual language. Because of the close correspondence between the construction of the cellular automata recognizing *HREC* languages and visual cortex, we assume that visual cortex is capable of recognizing this family of languages (though not limited to this family).

2.2 Language of Line Segments

Consider the language of finite line segments, $\mathscr{L}(line)$ over the character set $\Sigma = \{0,1\}$. This language can be represented by $\mathscr{L}(line) = \mathbf{0}^+ + (\mathbf{1})^+ + \mathbf{0}^+$. $\mathscr{L}(line)$ consists of a background of zeros and a foreground of any number $0 < n < N\text{-}1$ of ones. Notice that the language can also be written as $\mathscr{L}(line) = \mathbf{0}^+ + (\mathbf{1})^{\mathrm{n}} + \mathbf{0}^+$ as this form can be transformed to regular form by concatenating strings of different scales; writing it the latter way emphasizes the scale invariance. Translational invariance occurs via the padding of the foreground pattern on either side with zeros. This allows the line segment to be translated side to side and still be recognized as a line segment.

A cellular automaton can be constructed that will recognize $\mathscr{L}(line)$. We now provide an overview of this construction; more complicated examples have been examined, *e.g.,* a dashed-line language, but because of space considerations these examples are not shown here. At layer L_1 the local character set is $\Gamma_1 = \{0_1, s_{001}, s_{011}, e_{110}, e_{100}, 1_1\}$ where the characters are constructed from tiles over combinations of characters in L_0 that are present in the language. These character combinations are 0_1 = '000', s_{001}='001', s_{011}='011', e_{110}= '110', e_{100}='100', and 1_1='111'. Note that the two combinations '010' and '101' are not members of Γ_1 because they never occur in strings from $\mathscr{L}(line)$ (except for the trivial string of only one '1'). The symbols s and e are used because character combinations making these L_1 characters signal the start and end of the line segment.

The process of building local character sets at each layer continues until the top of the hierarchy is reached. These local sets Γ_i are created from combinations of characters in the previous layer. The local sets all have symbols representing zeros 0_i, ones 1_i, and the start and end symbols, s_i and e_i respectively. At the top of the hierarchy, there will be one set of neurons whose receptive fields span the entire input space in L_0 and this set is able to make a decision as to whether there is a line segment.

Each local language $\mathscr{L}_i$ is a description of the original language, but constructed from the local character set Γ_i representing the enhanced scale. The scale is enhanced because any character in Γ_i is based on three characters from $\Gamma_{i\text{-}1}$ (recall that the number of spatial nodes decreases by a factor of 2 at each layer). An exception to this rule is that sometimes there are *pass through* characters that simply carry information from one layer to another. For example, a line segment of scale $n = 8$ is recognized within the receptive field of some node at layer L_3 ($3 = log_2(8)$) and can be recognized at this layer. Thus in layer L_3 the character c_{yes} will be activated and this character is passed up the hierarchy indicating a string from $\mathscr{L}(line)$ has been recognized. Illegal combinations of characters can also be detected, so in general, a local character set consists of characters of the form $\Gamma_i = \{0_i, s_i, 1_i, e_i, c_{no}, c_{yes}\}$.

3. AMBIGUITY

As described in Giammarresi and Restivo (2008), the definition of languages in terms of recognizability by tiling systems (local languages and projections) is implicitly non-deterministic (ambiguous). Informally, a tiling system is unambiguous if every picture in a language has a unique counter-image in its corresponding local language. Indeed, Anselmo et al (2006) have shown that it is *undecidable* whether a given tiling system is unambiguous or not. This may seem like a severe restriction, however in this section we make the case that visual cortex *must* exploit ambiguity in some form in recognizing visual languages.

3.1 Constraints

At this point it is time to step back and consider what we are trying to accomplish. The goal is to construct cellular automata that recognize a viewpoint invariant subset $\mathcal{V}$ of the hierarchically-tile-able languages. $\mathcal{V}$ is a subset of regular languages and we know it is possible to construct an automaton that will recognize *any* regular language, even a specific random string. However, this finite automaton would have a number of specific states (nodes) and would be very brittle in that it could only recognize *one* very specific pattern at one scale in the family of regular languages.

On the other hand, a finite automaton could be constructed that would recognize the set of all strings $S \subset \Sigma^*$ whose length is $l < N_0$. A layer's local character set Θ_i would then contain all three-way combinations of characters in the previous set Θ_{i-1}. The size of the character set grows as the cube, $size(\Theta_i) = size^3(\Theta_{i-1})$ where $size(\Theta_0) = 2$. Thus the last recognizing layer would contain an exponentially large number of characters.

This general recognizer would allow visual objects to be constructed out of many smaller recognizable objects. For example, a dog has a body, fours legs, a face, and a tail. However, the brain simply does not have enough resources to retain all possible character combinations. Each distinct cortical area (*e.g.,* V1, V2, and V4) in visual cortex has approximately the same number of neurons. As we have assumed that spatial resolution only drops off by a factor of two at each layer, the number of local characters can only increase by a roughly a factor of two at each layer. This is a *very* tight constraint as the growth rate in the size of the local character set is linear, $size(\Theta_i) = k$, $size(\Theta_{i-1}) << size^3(\Theta_{i-1})$ with $k \sim 2$. The important conclusion to make from this simple analysis is that we must ruthlessly reduce (constrain) the number of local characters in whatever way possible.

3.2 Scale and Position Information

From the example of the language $\mathscr{L}(line)$, it was shown that a line segment of scale $n = 8$ is recognized at layer L_3 and that the character c_{yes} is activated and passed on to higher layers. Note that a local character $c_{3,yes}$ could be created that carries scale information as well as recognition. However, because this would lead to an increase in the number of total states (because size information is associated with each possible object recognized), we assume that size information is discarded in favor of conserving the total number of states. It may be possible that *some* relative size information is retained in visual cortex, but not that all *complete* size information is retained. It is likely that a more complex network topology is employed rather than a purely feed-forward network to compute size.

It may appear surprising to see that size can be described with a regular language subset and not require a more complex language such as a context-free language. This apparent contradiction occurs because we are considering *only* strings of fixed length. Thus a finite automaton *can* be constructed that measures the length of the line segment. Likewise, position information could be recorded at the point a string is recognized, perhaps with the character $c_{i,j,yes}$. However, allocating resources to record position information would also explode the number of characters (states) in the recognizer.

Thus, at the top of the hierarchy, we argue that relatively little spatial and size information about an object is kept by visual cortex in order to conserve resources. Retaining only the fact that an object is present allows many more objects (patterns of strings) to be represented. In effect, spatial and position information has been traded for the ability to recognize more objects. It is thought that position information is obtained from the separate dorsal pathway in visual cortex. We *predict* that some other mechanism may also be present to measure size information.

3.3 Equivalence Classes

Ignoring all size and location information in the hierarchy helps reduce the proliferation of states, although this alone may not meet the resource constraint on the number of actual states. We are forced to eliminate the combinatorial explosion of possible states (enumerated by characters) by combining states into equivalence classes. For example, for the line-segment language we combine the local characters s_{001} and s_{011} in Γ_1 into one character represented by the regular expression $\boldsymbol{s}_1 = \boldsymbol{s}_{001} + \boldsymbol{s}_{011}$. Likewise, we combine the line end characters so that $\boldsymbol{e}_1 = \boldsymbol{e}_{110} + \boldsymbol{e}_{100}$.

Support for combining characters into equivalence classes comes from considering the translation of an input pattern by one character position either to the left or to the right. A string of zeros '00000' isn't modified by a translation so the zero state can't be reduced further and a tile covering this string is labeled by the local character 0_1. Likewise the tile covering a string of ones is labeled 1_1. However, consider a “dashed” string '01010101'. If the string is shifted to the left or right by one character, it will either appear as the tile '010' or '101' to a detector node. Surely this level of detail (relative position in a dashed line) can't be important — especially since the language family being considered is translationally invariant — so the two characters are combined into an equivalence class (a single character) representing that a portion of a dashed line, $\boldsymbol{d}_1$ = '010' + '101', has been detected. A similar argument holds for the start and end characters s_1 and e_1.

After taking into consideration translational invariance, the *general* local character set in L_1 is $\Gamma_1 = \{0_1, s_1, d_1, 1_1, e_1, c_{yes}, c_{no}\}$. Since the recognition characters c_{yes} and c_{no} are not combinable with other characters in later layers, the effective size of Γ_1 is reduced from eight to five characters. Combining like characters under translation is but one way to form equivalence classes. This may not even lead to ambiguity as it may be just used as a way to reduce redundant information (recall the use of overlapping tiles). In general, equivalence classes add ambiguity. However, equivalence classes are likely necessary from a resource point of view and if so visual cortex must be tolerant of a certain level of ambiguity.

In summary, we define a visual language $\mathcal{V}$ as a member of the *HREC* family constrained in two important ways. First, we assume that $\mathcal{V}$ is invariant under translation; the input string (padded by zeros at either end) may be presented at any position j in L_0. Second, we assume that $\mathcal{V}$ is scale invariant; the one-dimensional “image” represented by the input string may expand or shrink and still be a member of the language. This definition is analogous to the visual system in that it can recognize a predator, for instance, no matter where the predator appears in the visual frame. A predator will also be recognized both relatively near and far in the visual frame.

Manually constructing a CA recognizing a visual language is tedious because it requires constructing local character sets for separate strings at every scale n. It also requires constructing equivalence classes by considering string translations. This begs for machine construction of the recognizer. We have implemented a recognizer using spiking neurons with the PetaVision[1] neural simulator. In this implementation a separate neuron was assigned to each character in a local set with the network connectivity as shown in Figure 1.

The PetaVision recognizer was tested by generating strings from $\mathcal{L}(line)$. The recognizer accepted all strings from $\mathcal{L}(line)$ and none that were not in the language. The next step in this line of research is to attempt to learn synaptic weights for a complete CA hierarchy for a given language, using spike-timing-dependent synaptic plasticity (STDP, Song et al, 2000). This would allow a recognizer to be learned and not require tedious manual construction.

[1] http://sourceforge.net/projects/petavision/

CONCLUSIONS

We have begun to explore the question of which viewpoint invariant languages can be solved by a network of concurrent, hierarchically connected processors. This has led to the definition of a language that is hierarchically tiling system recognizable *HREC*. This language family is closely related to the *REC* family of languages.

A one-dimensional visual language was defined as an *HREC* language that is translational and scale (viewpoint) invariant. An algorithm was presented that constructs a recognizer for an *HREC* language and recognition of the line language was considered. We demonstrated that a recognizer can be implemented with a network of spiking neurons in PetaVision.

However, several problems were discovered while considering one-dimensional visual languages. Some of these problems are related to aspects of viewpoint invariance and others related to ambiguity. Regarding ambiguity, it is interesting to recall that it has been shown to be undecidable whether a particular *REC* is ambiguous or not. Perhaps ambiguity at some level is required for visual processing or is resolved by a higher level of computation than can be provided by a finite state automaton.

Because of an extremely tight resource constraint (linear growth rate in the size of a layer's local character set), only a small subset of possible local characters may be retained (in the form of equivalence classes). This implies that vision must be as much about what we don't (cannot) see as about what we can see. This can be seen as a positive, because as a consequence the brain must have learned how to replace specificity with generality. This generality must in some way decrease the brittleness of object recognition, as the loss of a few bits due to noise in an image doesn't significantly degrade performance.

We claim that formal language theory can provide guidance — especially concerning size and location information as discussed above — in understanding how computation is done in visual cortex. In the process of studying this relationship, we have discovered an algorithm (potentially new) for constructing a cellular automaton that will process a series of finite length strings in $log(N_0)$ time (wall clock) with N_0 nodes in L_0 processing concurrently.

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation grant 0749348 and by the Los Alamos National Laboratory LDRD program under grant 2009006DR.

REFERENCES

Alonso, J. et al, 2001. Rules of Connectivity Between Geniculate Cells and Simple Cells in Cat Primary Visual Cortex. *Journal of Neuroscience,* Vol. 21, pp 4002–4015.

Anselmo, M. et al, 2006. Unambiguous Recognizable Two-Dimensional Languages. *RAIRO – Inf. Theor. Appl.,* Vol. 40, pp 277–293.

Anselmo, M. et al, 2009. A Computational Model for Tiling Recognizable Two-Dimensional Languages. *Theoretical Computer Science,* Vol. 410, pp 3520–3529.

Blum, M. and Hewitt, C., 1967. Automata on a Two-Dimensional Tape. *IEEE Symposium on Switching and Automata Theory,* Vol. 6, pp 2–3.

Giammarresi, D., and Restivo, A., 1992. Recognizable Picture Languages. *International Journal of Pattern Recognition and Artificial Intelligence,* Vol. 6, pp 241–256.

Giammarresi, D., and Restivo, A., 1997. Two-Dimensional Languages. In *Handbook of Formal Languages,* Rozenberg, G. et al., Eds. Springer Verlag, London.

Giammarresi, D., and Restivo, A., 2008. Ambiguity and Complementation in Recognizable Two-Dimensional Languages. *In Fifth IFIP International Conference on Theoretical Computer Science,* G.A. et al., Eds., Vol. 273. Springer Verlag, London.

Song, S. et al, 2000. Competitive Hebbian Learning Through Spike-Timing-Dependent Synaptic Plasticity. *Nature Neuro-science,* Vol. 3, no 9, pp 919–926.