Generalised Syntactic Pattern Recognition as a Unifying Approach in Image Analysis

Mark Venguerov

POINT Information Systems Ltd.
E-mail: Mark.Venguerov@pointinfo.com

Pádraig Cunningham

AIG, Department of Computer Science
Trinity College, University of Dublin
E-mail: Padraig.Cunningham@cs.tcd.ie

Abstract. In this paper we analyse theoretical foundations of syntactic pattern recognition and its relationships with mathematical linguistics, structural pattern recognition, and statistical pattern recognition. We point out that all the above mentioned methods are based on two fundamental operations: the *combination* of parts into wholes and *abstraction*. We provide examples of those operations in different contexts. Finally, we note that at this level of generalisation, all three main approaches to pattern recognition (structural, statistical and syntactic) can be merged into one powerful method. We outline a system based on these principles and intended to be used for the recognition of unconstrained handwriting.

1 Introduction

There exists a huge variety of methods for pattern recognition. All those methods are usually classified as either statistical (decision-theoretical) or structural. Within the realm of structural methods there exists a special area called syntactic pattern recognition (SPR) dealing with the application of methods developed in mathematical linguistics to pattern recognition.

If we start to consider the theoretical foundations of SPR, a natural question arises: why can (and should) methods developed in mathematical linguistics be applied in so different an area as pattern recognition? In this paper we try to answer this question as well as show that certain generalisations of SPR can naturally include other structural and statistical methods thus providing a framework for the unification of all different strategies in pattern recognition. We hope that this unified approach will bring new power and will eventually achieve human level performance in computer vision and speech recognition.

2 Syntactic Pattern Recognition: State-of-the-art

As it is defined in [1,2], syntactic pattern recognition is the application of methods of mathematical linguistics to pattern recognition. The standard procedure is as follows: first, some pre-processing is done in order to reduce noise, improve contrast, etc. During the next stage certain predefined features are extracted from the input. Finally, the system parses the set of features discovered in the previous step using some type of grammar. If this parsing can be successfully finished (i.e. the whole input can be reduced to the grammar start symbol S) then the system has recognised a pattern (an object, a set of objects, etc.). Syntactic and semantic information calculated at this stage can be used for understanding and reasoning about the input.

Depending on the type of grammar used in this approach concrete implementations can be further classified as string, tree and graph grammar based. The last ones are the most general and are perceived to be most suitable for generic pattern recognition. Unfortunately, graph grammar based syntactic pattern recognition is generally speaking intractable.

3 Similarities and Differences Between Pattern Recognition and Linguistics

Why can methods developed in one area of computer science be used in another? A naïve answer to this question is: because they work (when they work). Interestingly enough, this answer is considered often to be a satisfactory one. A more enlightened approach to the problem suggests that there must be something common between the two areas, some abstract entity which realises itself in different ways in those areas, but facilitates the transfer of methods between them.

In the case of pattern recognition and mathematical linguistics such a key common abstract entity is the notion of *structure*. The structure of an object is the way its constituent parts are put together to produce the object. There are two important parts in this definition: a) an object consists of *parts* and b) in order to constitute the object its parts have to *relate* to each other in a certain way, i.e. they should satisfy certain *relations*.

In linguistics the objects to be recognised or generated are sentences. Sentences consist of words. So, words and word groups are those *parts* that constitute wholes - sentences. The relations that the words have to satisfy in order to produce valid sentences depend on the type of a particular language. In non-order free languages (e.g. English) this relation is *adjacency*. In word order free languages (e.g. Slavic languages) this relation is *grammatical conjugation* through suffixes, auxiliary particles, articles, etc. In any case, the application of the notion of *structure* in linguistics is quite straightforward.

The concrete realisation of the *structure* notion in pattern recognition depends on the specific area. In computer vision for instance, the elements are usually basic features like edges, blobs, etc. They are combined into more and more complex objects, e.g. edges can be combined into straight line segments, which in turn are com-

bined into rectangles and so on. The relations between the parts are 2-D (and deduced 3-D) spatial relations.

This universality of the notion of *structure* made some philosophers and cognitive scientists believe that any representation in any cognitive system is 'language' like.

Along with the similarities described above there exist a number of differences, which preclude direct application of linguistic methods in pattern recognition. The main obstacle here is the higher dimensionality of the input, which in turn leads to the absence of linear ordering. Although this may seem a minor issue, it results in exponential growth of the computational resources with the growth of input. Similar problems arise in linguistics when algorithms developed for non-free order languages are directly applied to a free word order language.

The size of the input is another substantial difference. The size of a typical sentence (a unit of recognition in linguistics) is usually tens of words, whereas the size of a typical (realistic) image is at the level of hundreds and even thousands of features. This, combined with absence of ordering of the input, makes direct application of linguistic methods computationally intractable.

Some researchers made an attempt to overcome this intractability introducing an 'artificial' ordering of the input. This approach allows the application of efficient parsing algorithms from linguistics. Unfortunately, imposing an arbitrary ordering on the input leads in the general case to very unstable recognition, i.e. presence or absence of a single small feature can dramatically change the result.

4 Two Basic Operations in Recognition

If we analyse the recognition process from a very abstract and general point of view, we discover that basically all of the different approaches to recognition are based on two fundamental operations - the *combination of parts into wholes* and *abstraction*. These two operations have corresponding fundamental relations - the *part-whole* relation and the *is-a* relation.

The *abstraction* operation in recognition is the process of either finding a class to which an instance, detected during the recognition process, belongs, or finding a super-class for a given sub-class and further using this class in the recognition process. Such a definition of *abstraction* corresponds to the usage of this term in logic. We don't specifically distinguish the two forms of the operation - the instance-to-class and the sub-class-to-super-class.

Certain areas in pattern recognition emphasise one of those operations more than the other, e.g. structural pattern recognition is explicitly based on the *part-whole* relation, whereas statistical pattern recognition pushes this relation almost to oblivion. Nevertheless, thorough analysis shows that either a specific method of recognition has a rather restricted area of applicability, or it implicitly uses both operations/relations.

Let us consider, for instance, structural image analysis. The knowledge the system uses for recognition is expressed here as a set of models, each of which contains

a set of features (parts) and a set of relations those features have to satisfy in order to constitute an instance of what the model represents. Sometimes those models are organised into a hierarchy, so that one model is a part of another one. This scheme works well for the recognition of simple objects and scenes. If it is applied directly to the recognition of more realistic images, the number of models grows very quickly. What can help here is the organisation of the knowledge base into a structure similar to semantic networks, with explicit use of the *is-a* relation and the *abstraction* operation. We discuss this structure in more detail in the next section.

In linguistics, the *combination* operation corresponds to string concatenation. The *abstraction* operation is in this case more subtle. Let us consider a (context-free) grammar, which has exactly one rule with a given non-terminal in its left-hand side. If such a grammar contains recursion it generates infinite length strings. Otherwise, it generates exactly one string! So, for a grammar to generate more than one string, it has to contain multiple rules with the same non-terminal in their left-hand side. We can consider therefore such a non-terminal as a class, whose sub-classes are constituted by strings generated by different rules. Based on this observation, we consider this combination of different rules for the same non-terminal a special case of *abstraction.*

The notion of *abstraction* in linguistics becomes clearer if we convert a grammar into its Wolff's normal form [3,4]. In this form there are two types of rules, OR-rules and AND-rules. OR-rules are of the form:

A = B | C | D,

which is a short representation of a set of 'normal' rules A=B, A=C, A=D. AND-rules are just normal rules of the form:

W = X Y Z,

which means that W is a concatenation of terminals/non-terminals X, Y, and Z. Each non-terminal can be used in the left-hand side of exactly one rule, so the set of all non-terminals has two subsets: OR-symbols and AND-symbols. Any context-free grammar can be transformed into Wolff's normal form.

Now, each non-terminal is a class label for the class of all strings, which can be generated from it. Hence, the OR-rule A = B | C | D means that class A contains sub-classes B, C, and D and is an *extensional abstraction* of those classes.

5 Abstraction

In order to understand better how the generic recognition process described in this paper works it is very important to analyse the *abstraction* operation. First of all, let us recall that in mathematical logic there exist two types of class definition - intentional and extensional. An extensional description of a class is an explicit listing of all its elements. An intentional definition is a predicate, which must produce TRUE when calculated for an object that belongs to the class. Corresponding to this distinction we define *extensional* and *intentional abstraction.*

An example of the extensional abstraction is a dictionary, which assigns a grammatical category to each word listed in it. Such a dictionary defines a gram-

matical category as the set of all words marked with this category. The only way to check that a certain word belongs to a certain category is to find this word in the dictionary and check if this word is marked with this category. It is impossible to derive or calculate this from the word itself.

In contrast to this, sets of numbers usually have intentional definitions. For instance, it is impossible to list all the prime numbers. Instead, having a number we can calculate the predicate 'prime' for this number (e.g. using Eratosthenes sieve algorithm). If the output of the algorithm is TRUE then the number is prime.

An important feature of the *abstraction* operation is that it always leads to a loss of information. An abstract concept contains less information than a specific instance of the concept. On the other hand, exactly this 'loss of information' allows cognitive systems to process information about the world. The trick here is that only the information irrelevant to the goals of perception or reasoning is stripped away. Which information is relevant and which is not is highly context dependent. It makes it very difficult to model this feature of cognitive systems algorithmically.

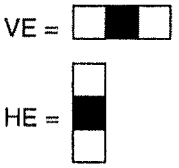
A major problem in image analysis is the huge variability of the input. By this we understand the fact that even a relatively simple object (e.g. a cube) can produce a huge number of different images depending on its position in the view field, its pose, illumination conditions, etc. In this case the ability of a recognition system to strip out this irrelevant (for the purposes of recognition) information becomes instrumental. Many well-known methods in image analysis are examples of the *abstraction* operation (e.g. thinning, edge detection, etc.) though it is rarely stated explicitly.

6 Knowledge Representation in Recognition Systems

All the systems that perform recognition contain two main types of information. The first type is the 'knowledge' of the system about the recognition task. In neural networks this knowledge is represented as connection weights, in case-based reasoning systems - as the case base, in model matching systems - as the database of models, in syntactical parsers - as the set of grammatical rules. The second type of information is the current input of the system, its state in the recognition process and the result (or set of results) of recognition. Some systems use one representation for both types of information, some use two different representations. For instance, in neural networks both the 'knowledge' of the system and the processing information are represented as vectors of real numbers; in model matching systems used in computer vision models and the input are usually represented as graphs. In syntactical parsers the knowledge of the system is represented as a set of rules, the input of the system is a string, and the output is a parse tree.

In our model the 'knowledge' of the systems is represented as generalised syntactic rules. One type of rule representation corresponds to Wolff's OR-rules. Rules of this type describe a class as a collection of its subclasses. Structural rules constitute the second type of rules. They describe what parts an object consists of and what relations these parts have to satisfy and correspond to Wolff's AND-rules.

This scheme is very similar to the graph grammar approach in SPR. The difference is the representation of relations. Let us recall that there exist two types of representations for sets - extensional and intentional. The same is true for relations. A relation can be specified as a table, which lists all the pairs (for a binary relation) of objects that satisfy this relation - this is an extensional representation. Alternatively, a relation can be specified as a (multi-place) predicate. Graph grammars are usually specified using extensional representation of relations. In contrast to this, in our approach we use intentional representation. This means that relations are specified as procedures taking object attributes as parameters and calculating truth values. The representation of relations allows us to express relations with different degree of abstraction, ranging from exact positional relations (e.g. one object is located 30 pixels below the other) to very abstract qualitative relations (e.g. one object is located to the left of the other). This set of relations with the increasing degree of abstraction constitutes the framework of the knowledge representation of the system. It facilitates the creation of more and more abstract representations of the input of the system leading to the efficient recognition.



VE - Vertical Element
(Element of a vertical line)

HE - Horizontal Element
(Element of a horizontal line)

VL1 = VE + VE : R1
VL2 = VL + VE : R1
VL = VL1 | VL2 - Vertical Line
GVL1 = VL + VL : R2
GVL = GVL1 | VL - Generalised Vertical Line

HL1 = HE + HE : R3
HL2 = HL + HE : R3
HL = HL1 | HL2 - Horizontal Line
GHL1 = HL + HL : R4
GHL = GHL1 | HL - Generalised Horizontal Line

LINE1 = GHL | GVL
LINE2 = LINE + LINE : R5
LINE = LINE1 | LINE2 - Curved Line

R1: $left_1 = left_2$ and ($bottom_1 = top_2$ or $bottom_2 = top_1$)
R2: $left_1 = left_2 \pm 1$ and ($bottom_1 = top_2$ or $bottom_2 = top_1$)

R3: $top_1 = top_2$ and ($right_1 = left_2$ or $right_2 = left_1$)
R4: $top_1 = top_2 \pm 1$ and ($right_1 = left_2$ or $right_2 = left_1$)
R5: $distance(beg_1, end_2) < 2$ and $abs(angle_1 - angle_2) < 10°$

Fig. 1 A fragment of a knowledge base

On the other hand, our representational scheme can be conceived as a generalised dictionary of models, where each AND-type rule constitutes a model. This view shows explicitly that the difference between syntactic and structural pattern recognition systems disappears at this level of generalisation.

Fig.1 shows a fragment of the knowledge base used to recognise (curved) lines starting at the pixel level in a binary image. Rules in this fragment have either AND-form (A=B+C:R) or OR-form (A=B|C|D). R denotes a set of spatial relations which parts in the right-hand side have to satisfy. Some relations (R1-R4) are very concrete as they specify exactly mutual positions of parts. The relation R5 is an example of a more abstract relation.

Note the recursive nature of some rules in this fragment. This recursion is not related to any 'fractalness'. It rather reflects the fact that if we combine two straight line segments with the same direction and beginning of one segment coinciding with the end of another we still get a straight line segment with the same direction, i.e. an instance of the same class of segments.

7 The Recognition Procedure

The recognition procedure contains two simultaneously working processes. The most fundamental one is similar to a bottom-up chart parser. The elementary steps of this process are checks that a certain configuration of parts satisfies the set of relations for some object and the generation of such configurations. These elementary steps are inserted with different priorities into a priority queue corresponding to the agenda in a chart parser.

The second process is a top-down modification of elementary recognition step priorities, which uses information about most probable routes of recognition and the previous experience of the system. This process effectively is a heuristic reducing the exponentially sized search tree of the bottom-up process, so that the overall recognition process becomes computationally tractable.

As the bottom-up process works with fuzzy relations and calculates a degree to which an instance satisfies a given model, it can be represented as an evidence-propagation approach, therefore facilitating application of statistical pattern recognition methods. This is just one more piece of evidence that all the different approaches in pattern recognition could (and should) be naturally combined together in order to produce a really powerful system.

A detailed description of the knowledge representation and the recognition procedure are given in [5], as well as more detailed analysis of their underlying principles.

8 Conclusion

In this paper we presented a unifying approach to image analysis, which is a generalisation of syntactic and structural recognition methods. We discussed similarities and differences existing between mathematical linguistics and pattern recognition. We emphasised the importance of the explicit and thorough analysis of the two fundamental operations - combination and abstraction. Then we presented a brief discussion of the abstraction operation and introduced the notions of extensional and intentional abstraction. Finally, we outlined knowledge representation and recognition algorithms used by the system.

It is important to note that there still exist a number of problems with the approach, but we hope that once those problems are solved (and the solution is in our sight) we will develop a new highly extensible and powerful system for image analysis.

A specific version of the system based on the principles discussed in the paper and designed for the off-line recognition of unconstrained handwriting is being currently implemented by one of the authors as a part of his Ph.D. thesis.

9 References

[1] M. Flasinski, Mathematical Linguistics Models for Computer Vision, Machine GRAPHICS & VISION, vol. 5, ½,1996, pp.87-97.

[2] K.S. Fu, Syntactic Pattern Recognition and Applications, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[3] J. G. Wolff, Language acquisition, data compression and generalization, Language and Communication, vol. 2, 1, pp. 57-89, 1982.

[4] J. G. Wolff, Simplicity and power - some unifying ideas in computing, Computer Journal, vol. 33, 6, pp. 518-534, 1990.

[5] M. Venguerov, Generalised syntactic pattern recognition and its application to the off-line handwriting analysis, Ph.D. Thesis, Department of Computer Science, Trinity College Dublin, October 1998.