Automatic Adjacency Grammar Generation from User Drawn Sketches

Joan Mas Romeu, Bart Lamiroy, Gemma Sánchez, Josep Lladós

Automatic Adjacency Grammar Generation from User Drawn Sketches

Joan Mas Romeu[1], Bart Lamiroy, Gemma Sanchez[1], Josep Llados[1]
[1]Computer Vision Center, Dept. Informàtica, Universitat Autònoma de Barcelona
Edifici O - Campus UAB, 08193 Bellaterra, Spain
INPL – LORIA
École des Mines de Nancy, Parc de Saurupt – CS 14234
54042 Nancy CEDEX – France

Abstract

In this paper we present an innovative approach to automatically generate adjacency grammars describing graphical symbols. A grammar production is formulated in terms of rulesets of geometrical constraints among symbol primitives. Given a set of symbol instances sketched by a user using a digital pen, our approach infers the grammar productions consisting of the ruleset most likely to occur. The performance of our work is evaluated using a comprehensive benchmarking database of on-line symbols.

1. Introduction

This work concerns Adjacency Grammars in a sketchy symbol recognition framework. Adjacency Grammars represent visual syntax in terms of adjacency relations between primitives [2]. The productions or grammatical rules of these grammars are expressed as a set of primitives and a set of adjacency relations among them. These are referred to as *constraints* and express relations as incidence, intersection, *etc.* Readers are invited to consult [4] for a formal definition and further details of Adjacency Grammars.

Since productions are formed by constraints specifying relations among the primitives that form it. Adjacency Grammars directly incorporate the ability to represent bidimensional shapes. Likewise these grammars consist of sets of primitives instead of sequences. This is an advantage with regard to traditional string grammars, even though some variations as PDL or Plex grammars allow to represent bidimensional entities with string-like productions. The orderless representation of adjacency productions is also noteworthy in an on-line input mode where different people tend to draw identical symbols in different way (especially where the order of primitives is concerned).

Model learning is a crucial task in any Pattern Recognition problem. Syntactic Pattern Recognition often requires a declarative process where the user is responsible of explicitly defining and compiling the grammar rules that identify a model shape, into the system. This requires the person creating and identifying the rules to be aware of the syntax, the geometric constraints and the pitfalls of the grammar. Within an interactive and large userbase environment, this is a limiting factor. Our goal is therefore to completely avoid explicit creation of rulesets. The main idea is to allow the user to simply draw the symbol he wants to further exploit (for recognition, editing, ...), and automatically generate the corresponding ruleset.

Grammatical inference is a classical research field in formal language learning. A few works exist on the inference of grammars representing visual patterns. Miclet presents in [5] a survey on grammatical inference applied to pattern recognition. A general presentation may be found in the work of de la Higuera [1].

The next section presents the basis of the inference method. In order to be robust to hand drawn distortions, over and under-constraining, *etc.*, section 3 details the required enhancements that allow the method to be fully exploited. The experimental part is presented in section 4. The final section presents conclusions and further work.

2. Constraint Alphabet for Representing Graphical Symbols

We have to preprocess the sketch made by the user using a digital pen [3]. This process is a polygonal approximation using methods described in [6]. Hence, a shape is described as a combination of basic primitives (segments) according to an alphabet of relational constraints.

For each pair of primitives (A, B), a number of constraints (as depicted in Figure 1) are evaluated using a normalized associated uncertainty degree $\delta = [0 \dots 1]$ which measures the degree of distortion with regard to an ideal

The 18th International Conference on Pattern Recognition (ICPR'06)
0-7695-2521-0/06 $20.00

shape. The following constraints were currently associated:

1. $Parallel\,(A,B) \rightarrow \frac{2}{\pi}\left|\widehat{A,B}\right|_{\left[0\ldots\frac{\pi}{2}\right]}$
2. $Perpendicular\,(A,B) \rightarrow 1-\frac{2}{\pi}\left|\widehat{A,B}\right|_{\left[0\ldots\frac{\pi}{2}\right]}$
3. $Incident\,(A,B) \rightarrow$ min. distance between segments and endpoints
4. $Adjacent\,(A,B) \rightarrow$ min. distance between endpoints
5. $Intersects\,(A,B) \rightarrow$ min. distance between midpoints

Figure 1. From left to right: *incidence*, *adjacency*, *intersection*

The set of constraints with the lowest associated uncertainty degrees correctly describes the symbol. This set determines the final grammatical ruleset.

The single selection of the constraints with the lowest associated values is not sufficient. In addition, we have to determine what it is the threshold for selecting such set of constraints. The next section details what further precautions need to be considered in order to obtain a fully usable algorithm.

3. Grammatical Ruleset Inference

Ruleset Generation. In order to generate an optimal ruleset, it is necessary to retain only the constraints that are sufficiently realistic with respect to the drawn figure. In a perfect world, it would be sufficient to only retain constraints with a null cost. In reality, however, hand drawn samples are very approximate, the segmentation process may introduce artefacts, and computational or sampling side-effects introduce supplementary noise.

In order to select the most significant constraints the solution consists in sorting all constraints by increasing order of their associated uncertainty degree. Considering that constraints should either be labelled as *valid*, *uncertain* or *spurious* (with associated values $\{0, 0.5, 1\}$ by construction of the normalized uncertainty degrees), the corresponding sorted histogram represents a progressive step function, as shown in Figure 2.

Since the progressive step function is the accumulative distribution function of a normal distribution, it is fairly straightforward to derive this function, and fit a Gaussian over it. The resulting parameters μ and σ (mean and standard deviation) give an excellent estimate of the boundaries of the previously cited *valid*, *spurious* or *uncertain* uncertainty degree values. In other terms, we only retain the constraints for which the associated uncertainty degree falls below $\mu - \sigma$. Figure 2 gives an idea of what kind of selection this results in.

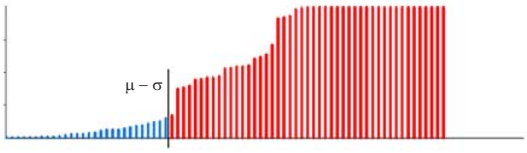

Figure 2. Ordered Constraint Histogram with $\mu - \sigma$ Cutt-off Position

Another important issue of our approach lies in the fact that human drawings are far from perfect (as shown for instance in Figure 3), lines that are supposed to join do not really so, perpendicularity is not respected, *etc.*. Even with the most sophisticated selection method, there is no way of inferring that the user intended to actually draw joined, perpendicular lines. In that case the generated model will tend to be under-constrained. On the opposite, a user may intend to draw a generic example, but the drawn lines happen to be perpendicular, by pure coincidence. Here the generated model will tend to become over-constrained. The only solution to this problem is to refer to multiple samples of the same symbol. This adds, however, a whole lot of difficulties.

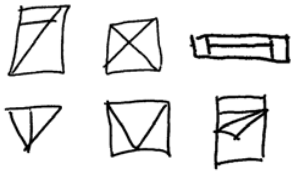

Figure 3. User Drawn Samples

Alignment of Primitives and Ruleset Normalization. The main issue is that, given two raw rulesets, inferred from two samples, there is only a very low probability that both will have the same number of primitives, will use the same names for the same primitives (drawing order is not guaranteed), do not apply symmetric rules (due to symmetry of certain rules: $\mathsf{Perpendicular}\,(A,B) \Leftrightarrow \mathsf{Perpendicular}\,(B,A)$), have filtered out the same constraints, *etc.*

For instance, two similar drawings may give equivalent, but different rulesets (Figure 4).

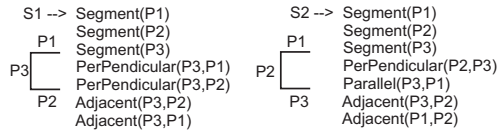

Figure 4. Different but equivalent rulesets.

To guarantee that all models generated from the samples are represented in a normalized way we proceed in three steps.

1. Alignment: first of all it is necessary to find the correspondence between the primitives of each ruleset. This is obtained by using the generated grammar as a recognition method [4] on the remaining models. Once the one-to-one mapping between primitives is computed, they are numbered from 1 to p.

2. Normalization: rule arguments are reordered such that the smallest numbered primitives are used as the first arguments (in case of symmetric rules).

3. Completion: in order to prevent incompatibilities between semantically equivalent rulesets, we proceed to generating a *closure* of the rulesets by inference of new rules based on geometric and topological knowledge (such as $Perpendicular(A, B) \wedge Perpendicular(B, C) \rightarrow Parallel(A, C)$).

Factorization. Factorization is the last step of the grammatical inference process. This step is needed to generate a final representative ruleset from all the different sample rulesets.

At this stage, each instance drawn by the user gives rise to a normalized ruleset. It may be over- or under-constrained as mentioned previously. It is clear from what precedes, that we cannot assume that all rulesets of all given instances are identical, due mainly to the fact that we are dealing with hand-drawn distorted symbols. The key point is now to deduce the common set of rules that defines the global model deriving from all given instances. To obtain this model, we proceed by using a voting scheme, as to extract the most frequently occurring rules.

Taking into account the values given by the cost functions we assign a weighted vote to each constraint of all samples selected by the above process. The weight is $1 - \alpha$ where α is the cost value of the constraint. Since $\alpha \in [0 \ldots 1]$, a highly satisfied constraint will have a weight close to 1 and a poorly satisfied one will have a vote closer to 0.

Constraints obtaining a cumulated weighted vote of at least $\frac{2}{3}$ the number of samples is considered belonging to the final model.

Taking this value has two major consequences. First, it implies that at least half of the samples vote for the constraint (even when having very few samples), and, second, the introduction of one erroneous or highly distorted sample does not provoke an alteration on the final ruleset. Results obtained with this method are exposed on the next section.

4. Experiments and Results

In order to validate the proposed method we have defined a set of examples extracted from a bench-marking database. We have used the CVC online database of symbols[1]. This database is formed by 50 models drawn by 21 persons each, divided into two groups, each person drawing an average of ten instances for 25 models. This results in a database of about 5000 sketched symbols. Since sketched symbols of the database are drawn with a digital pen & paper protocol [3], each instance consists of both on-line and off-line versions. Only the on-line ones have been selected.

We have selected the 12 symbols of the database whose primitive representations are compatible with our model (current version does not include circular arcs yet). The selected symbols have between 4 and 8 primitives. This selection allows to determine if there exists any relation between the samples needed to correctly infer a symbol and the number of primitives that form it. Another factor to take into account is the distortion of the symbols. Since different samples were drawn by different people, the personal drawing style inherently involves a distortion degree. The instances are selected randomly from the database without grouping by the degree of distortion unlike the work presented by Xiaogang in [7].

We have manually constructed the ground truth for each model. It consists in the ruleset corresponding to the ideal instance. We then randomly selected user sketches and incrementally incorporated the rulesets. At each stage the obtained inferred ruleset was compared with the ideal one. It is important to notice that we consider an inferred ruleset correct *iff* it is strictly identical to the ground thruth. *i.e.* no rules are missing (under-constrained ruleset) or no supplementary rules are present (over-constrained ruleset). We have performed two types of experiments. The first one aiming to compute a global performance ratio, and the second one to more specifically evaluate the performance of our method for each symbol class.

In the first series of experiments, we iteratively took random samples for each model until convergence of the inferred ruleset to the benchmark. This gave the following

[1]The database can be obtained by contacting the authors of this paper.

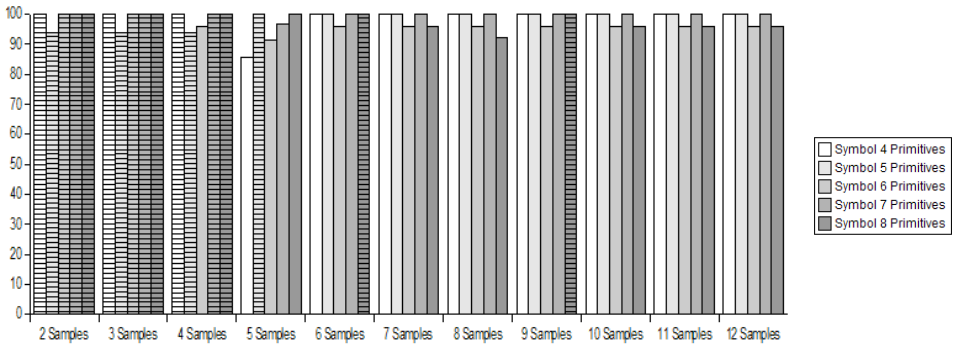

Figure 5. Learning curves for symbols with 4 to 8 primitives

indicators as of the number of samples needed to infer an exact ruleset.

Min(#Samples)	5 (model of 4 primitives)
Max(#Samples)	12 (model of 8 primitives)
Avg(#Samples)	6.62
Standard deviation	2.04

Again, these numbers refer to the complete convergence to the benchmark. In the case of partial convergence, for instance, 7 samples are sufficient for any model to retrieve at least 90.0% of the total constraints of the benchmark.

The second series of experiments takes a more detailed look at the different models. Again, we take random samples and progressively increase their number. At each iteration the inferred ruleset is compared to the benchmark. We are more particularly interested with the level of over- or under-*constrainedness* of the result.

Figure 5 shows the evolution of the inferred rulesets (one for each type of model, from 4 to 8 primitives) at each iteration. The y-value for each model shows the percentage of its correctly inferred constraints. We identify over-constrained rulesets by means of a hashed bar, while fully or under-constrained sets are plain filled bars. For instance, the model composed by 4 primitives gives rise to over-constrained rulesets for 2, 3, and 4 samples.

As can be seen on the diagram of Figure 5, all the constraints are inferred with three samples in almost all cases. However, since the rulesets are over-constrained, we cannot consider the symbol as fully inferred. For this reason, we need more samples, which implies a reduction of the over-constrained set and a loss of some constraints that are subsequently recovered.

5. Conclusions and Further Work

We have presented an innovative and very efficient approach of automatically generating adjacency grammar rulesets. The results presented in the previous section give an idea of the problems generated to work from noisy data.

The main flaw of our approach is its inherent computational complexity, mainly in its first stages, where the constraint evaluation complexity is $\mathcal{O}\left(c2^{p-1}\right)$ being c the number of constraints, and p the number of primitives (provided the constraints are binary functions – *i.e.* with two operands). There is an elegant way out of this flaw. Indeed, all constraints are not mutually independent (as shown, for instance, the Completion item presented on section 3 as the third step of Primitives Alignment). Future work will inevitably focus on the optimization of the constraint evaluation, on increasing the set of constraints and on treating over-segmenting due to vectorization or capturing devices.

References

[1] C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 2005.

[2] J. Jorge and E. Glinert. Online parsing of visual languages using adjacency grammars. In *Proceedings of the 11th International IEEE Symposium on Visual Languages*, pages 250–257, 1995.

[3] Logitech. IO digital pen, 2004. www.logitech.com.

[4] J. Mas, G. Sanchez, and J. Llados. An adjacency grammar to recognize symbols and gestures in a digital pen framework. In *Proceedings of Second IBPRIA*, pages 115–122, June 2005. Springer, Berlin.

[5] L. Miclet. Grammatical inference. In H. Bunke and A. Sanfeliu, editors, *SSPR: Theory and Applications*, chapter 9. World Scientific, Singapore, 1990.

[6] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini, and S. Tabbone. Stable and robust vectorization: How to make the right choices. In A. Chhabra and D. Dori, editors, *Graphics Recognition: Recent Advances*, pages 3–18. Springer-Verlag, Berlin, 2000. Vol. 1941 of LNCS.

[7] X. Xiaogang, Z. Sun, B. Peng, X. Jin, and W. Liu. An online composite graphics recognition approach based on matching of spatial relation graphs. *IJDAR*, 7(1):44–55, 2004.