Remarks on the Notation of Coordinate Grammars

Eckart Michaelsen and Uwe Stilla

Research Institute for Information Processing and Pattern Recognition (FGAN-FIM)
Eisenstockstr. 12, 76275 Ettlingen, Germany
Ph.: +49 7243 99249, Fax: + 49 7243 99229
E-mail: {mich,usti}@gate.fim.fgan.de

Abstract. Coordinate grammars are discussed and alternative notation variants are given. The lack of generality and feasibility of Rosenfeld's later notation leads back to the original notation of productions. The interaction of the productions is depicted by production nets. From the structure of these nets a hierarchy of computational complexities for coordinate grammars can be inferred. As an example a grammar is given for the recognition of certain vehicles in ground-based visual spectrum domain picture sequences. It uses coordinates in 2D picture space and 3D scene space. The analysis process runs purely bottom-up and data-driven. Relations to some non-syntactic paradigms of pattern recognition like production systems or semantic nets are mentioned. Emphasis is more on the practical use of such structures for complicated pattern recognition tasks and less on the theoretical survey.

1 Introduction

Based on Anderson's work [1] Milgram and Rosenfeld [7] proposed a new type of generic grammar in 1972 suitable for picture analysis. A generation or reduction works on a *set* of symbols instead of a string or a graph. The symbols are additionally attributed by coordinates and therefore grammars like this are called coordinate grammars [7]. Since then different notations have been used for the specification of such grammars [7],[11],[9],[6]. Similar syntactic methods have been applied in image analysis. Some early examples are chromosome analysis with attributed grammars [17], bubble chamber data with PDL [12] and chemical formulas using NAPE-Productions [3]. Recently many applications can be found in the field of document analysis [16]. All these systems work on some structure such as strings, trees, arrays or graphs. In this paper we present applied coordinate grammars working on unordered sets.

We have previously discussed similar systems of productions using the term *production nets* as applied to tasks of automated 3D-photogrammetry and remote sensing [13],[14],[15]. Particularly in model based recognition of complex objects in uncontrollable environment and lighting conditions, 3D coordinate grammars help to circumvent problems of perspective invariance, self occlusion etc. Viewing such systems as coordinate grammars opens the way for the definition of hierarchies of languages and precise semantics for them.

2 Notations for Coordinate Grammars

We briefly review definitions from [7] with some useful extensions [5]: An *attribute domain* D is an n-tuple of sets. Although no further restrictions are made, simple structures like intervals in Z, Z^2, Z^3 are preferred. An *element* e is a symbol out of a finite lexicon of terminals T and non-terminals N ($T \cap N = \emptyset$) associated with a vector in the attribute domain D. One non-terminal g is called *goal*. A *configuration* κ is a a k-tuple of such elements. A *production* consists of an input word Σ and an output word Λ of symbols of the lexicon. Λ contains at least one non-terminal. Additionally a relation π and a function ϕ are defined on the attributes associated with Σ. The function calculates the attribute values to be associated to Λ. A configuration is called *input configuration*, if its symbolic part equals the input word and the relation holds on its attributes. The calculated output is called *output configuration*. A *coordinate grammar* defines a finite set of such productions P on the same lexicon and attribute domain. A *direct reduction* $\rightarrow$ using $p \in P$ transforms a set S of elements into another set S' of elements. It is permitted, if some subset of the set S can be listed as input configuration of the production. Then this subset is removed from S and replaced by the output configuration to form the new set S'. A *reduction* using such a coordinate grammar is a finite chain of direct reductions using its productions. The arrow is written in the reducing direction. A set S belongs to the *language* of a coordinate grammar, if a set S' may be reduced from it, that contains a goal element. Robustness against noise and background objects is gained by permitting arbitrary additional elements in S'.

In [7] these definitions where regarded as being 'too powerful'. Even with additional strong restrictions on the relations π and functions ϕ and on the form of the productions coordinate grammars can still simulate Turing machines. Therefore Rosenfeld restricts his investigations to *bounded* productions [11]. The attribute domain D has to be a *metric* and *discrete* space and the coordinates of all elements in the input and output configuration have to be within a certain predefined bound d_{max} of each other. A further necessary restriction is that to *shift-invariant* productions. If one adds a constant value $c \in D$ to all coordinates in the input configuration, then the relation π should still give the same truth value and the function ϕ should produce the same coordinates as before plus c. Under these restrictions the additional demand, that only one specific arrangement is replaced by another, yields no loss in generality, since only a finite set of arrangements can fulfill a relation π under these restrictions. If the productions are thus restricted and D is of dimension one, then they can be written down in the following form:

$$(\Sigma_1, ..., \Sigma_m | i_1, ..., i_m) \longrightarrow (\Lambda_1, ..., \Lambda_k | j_1, ..., j_k),$$

where $0 = i_1 < ... < i_m \leq d_{max}$ codes the arrangement (i. e. π) and $-d_{max} \leq j_1 < ... < j_k \leq d_{max}$ codes the function value (i. e. ϕ). The extension of this notation into higher dimensions is straight forward. Through

standard forms (where $m = k$ etc.) Rosenfeld shows equivalence to isometric string- and array-grammars. Thus he ends up with the desired hierarchy of grammars and languages, because isometric grammars are equivalent to classic generative grammars of the Chomsky types [11]. Nakamura follows this branch of investigation [8],[9].

For practical reasons we prefer notations that follow the original definition and demand no restrictions or special forms. Examples are $(\Sigma, \Lambda, \pi, \phi)$ or $\Sigma|\pi \xrightarrow{\phi} \Lambda$. A typical example production putting together lines may be of the form

$$line\ line \mid adj(head_1, tail_2) \overset{(tail_1, head_2)}{\longrightarrow} poly,$$

where the attribute domain is $[1..N_x]^2 \times [1..N_y]^2$ for the positions of head and tail of an element in an image, the symbol *line* stands for straight line segments, *poly* for polygons, and *adj* means adjacency. In PDL it would be written as $line + line \longrightarrow poly$. It might be necessary to be fairly liberal in the choice of the parameter inherent in such an adjacency relation. If we take Euclidean distance and set a fixed maximum d_{max} for it, then we get approximately $d_{max}^2 \cdot \pi$ possible arrangements. For $d_{max} = 10$ we would have to write down some 300 productions in Rosenfeld's notation instead of one.

A generative coordinate grammar may be defined using the same productions left to right. Its language does however not equal the language defined by the reductions, because the same element may be inserted in a set many times, but it can only be removed once. This may be circumvented using constrained multiset grammars [4]. For pattern recognition applications mostly the reducing direction is examined.

3 Production Nets

We defined a bipartite graph notation for systems of productions of the form used above [15]. The nodes of such a *production net* are given by the symbols and productions used in the system. An edge is drawn from a symbol a to a production $p = (\Sigma, \Lambda, \pi, \phi)$, whenever a appears in Σ. An edge is drawn from a production $p = (\Sigma, \Lambda, \pi, \phi)$ to a symbol a, whenever a appears in Λ. If the symbol appears i-times in the word, then the edge is drawn i-times. Thus a coordinate grammar can be notated as a production net.

Some properties of the grammar can be concluded from the graph structure of it's production net (see [6]). For example, if the net is *cycle free*, then the depth of a reduction is limited by the number of productions. Therefore we can give a polynomial worst case border $\mathcal{O}(|T|^l)$ for the computational effort required (with some limited power l and T being the input set of terminal elements). We call reduction cycles in the net *monotone*, if any reduction $S \rightarrow S'$ running through the cycle once implies $|S| > |S'|$. Under this restriction still a worst case border (though higher than polynomial) can be calculated from

the cardinality of the input set [6]. On the other hand unrestricted coordinate grammars can simulate Turing machines and thus pose a word problem. So we can give a hierarchy for coordinate grammars (unrestricted > monotone > cycle free) with corresponding complexity assertions without using isometric grammars or the notation of [11].

4 An Example 3D Coordinate Grammar

Figure 1 sketches the production net corresponding to an example coordinate grammar [5],[6]. This net is cycle free. The attribute domain contains sets of dimensions one, two and three, and of different topologies (bounded intervals, pixel spaces, voxel spaces, 2D- and 3D-orientations of finite resolution with closed topologies etc.). The task is here the detection and localization of small VW trucks in image sequences containing also large portions of unpredictable clutter and structure. A wide variety of aspects and lighting conditions should be covered.

The productions act as follows:

P1 implements a line prolongation process. Unusual is here that the right side contains arbitrary numbers of elements of the symbol L.

P2 is a version of the already mentioned PDL-like production putting together two non parallel adjacent prolonged lines LL. All four possibilities (head to head, tail to head, head to tail and tail to tail) are present. The result is one element of the symbolic type A.

P3 resembles P2. But here the symbol of one element in the right side is A. The result is one U-shaped structure of the symbolic type U.

P4 achieves depth reconstruction through inverting the projection functions (whose parameters are assumed to be known). Thus a point in the image becomes a ray in the scene. If these rays resulting from two U-elements intersect closely, an open quadrangle structure is formed in the spatial scene, whose symbolic type is O.

P5 regards any such open quadrangle just as a quadrangle, so that the missing edge is assumed to be present. Productions like this cope for missing contours and are of great importance in real world visual data.

P6 puts together two open quadrangles to form a spatial E-shaped structure of the symbolic type E. Here already model knowledge (angle constraints and Euclidean distance measures) of the vehicle is included. In 3D-scene analysis this is much more straight forward than in 2D-picture analysis, because there are no problems of perspective invariance.

P7 implements the final 3D model match. Mean square error sum minimization is performed. Here the right side has up to four elements, which only becomes computationally tractable because of the strong model knowledge incorporated.

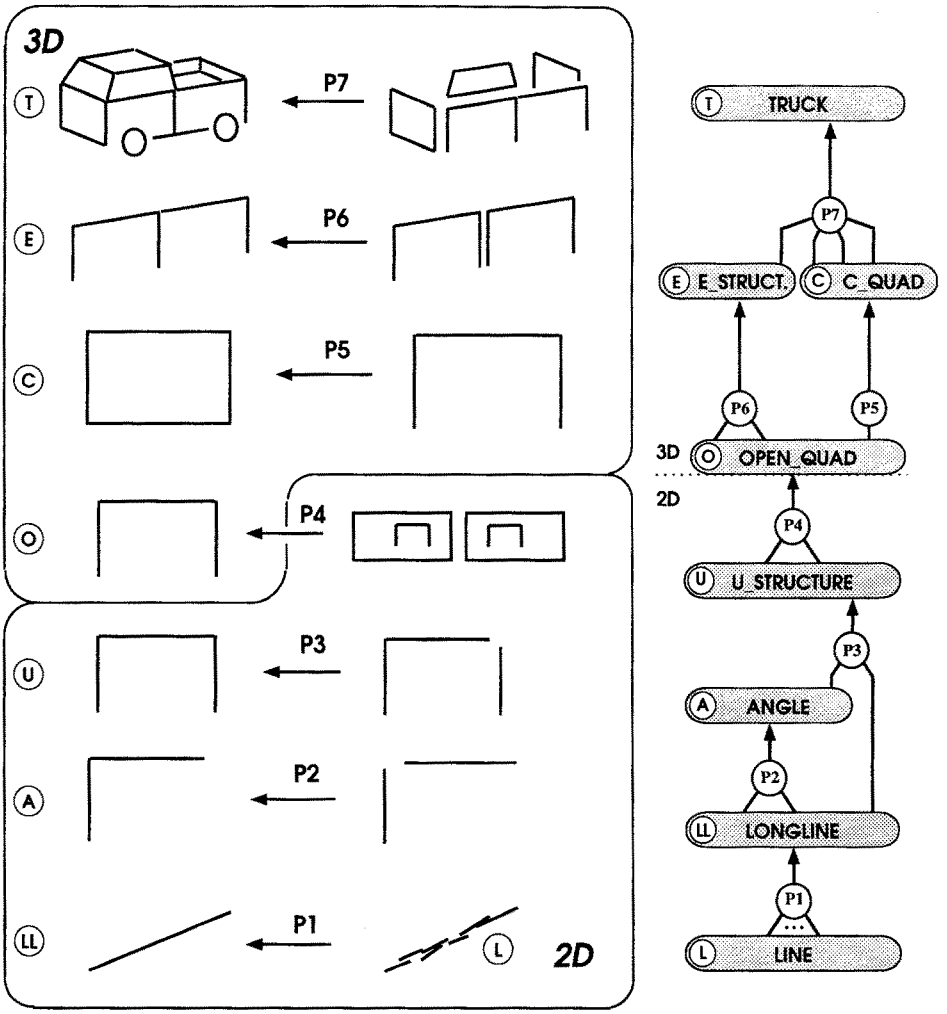

Fig. 1: Example grammar and corresponding production net

These seven productions form a typical selection of important types of productions. The goal element may be found several times in one location with slightly different attribute values for rotation and exact positioning. Then among these alternatives the best one is chosen with respect to some assessment and presented as the result. Other possible choices are the first instance, any instance or the mean calculated from all instances in one location.

Figure 2 shows the result of a parse overlayed as white wire frame model over some section of one frame of an analyzed image sequence. Below three narrow sections of images from the sequence containing only the vehicle are depicted to give the reader some idea of the perspectives used. Corresponding results of the feature extraction process (described in [14]) are shown above.

Fig. 2: Example of an image sequence, extracted terminals (L), and result (T)

5 Discussion

Coordinate grammars are one of many possibillities to bring syntactic views into pattern recognition theory and practice. Many of their properties resemble other structure based methods. For instance structural decomposition in *part-of* hierarchies is also used in semantic nets [10] and similar combinatorial search problems arise in aspect graph matching [2].

The production net notation gives some good overview over the information flow, the serial sequence in which certain computations are defined, the possible parallelism that is inherent in a coordinate grammar, and cycles where recursive computation occurs with a risk of infinite looping. In fact we emphasized common structure with petri nets [15].

One important difference between coordinate grammars and other structure based pattern recognition methods is the fact that the order of the primitives (terminals) given by the feature extracting process, be it a list, a string, a graph or a regular grid, is completely neglected. They are treated as an unordered set. This means, that coordinate grammars are to be recommended, whenever there is low confidence in this order. One reason for this lack of confidence in vision data lies in the projection function between a 3D-scene and

a 2D-retina. Adjacency for example is not invariant under this function. Using coordinate grammars, the system designer is thinking in a generative way coming from the objects he is looking for and then going down into more and more primitive concepts until a manifestation in the picture data is reached. The machine then is working in the opposite way, reducing instances of these concepts and neglecting everything else. Some of the combinations performed during the reduction may already be contained in the result of the feature extraction process. If for instance a list of primitive lines is given, then succesive lines are probably adjacent. Thus some of the computational effort might be wasted. Maybe because of this sub-optimality and the high computational effort in the search coordinate grammars have gained so little attention in the past decades. In the original definitions - which we still prefer to use - this point is quite evident. In Rosenfeld's later notation it is obscured.

We recommend coordinate grammars - with the productions in the proposed old notation and the system being displayed as a production net - because of their lucidity and simplicity. The gain of sound modular semantics, easy understanding and swift construction is much more important than the loss of some extra computation or wasted storage capacity.

6 Conclusion

A fairly old proposal of the field of syntactical pattern recognition - the coordinate grammar - has been adapted to suit complex 3D vision tasks. Coming from the application side we ended up with a different notation for the productions compared to the formalisms lately used by Rosenfeld, Nakamura and Marriott for their theoretical work. We state important properties like *simplicity, flexibility, robustness, modular* and *strict semantics* for our approach and hope to join theory and application. Problems with the computational effort or lack of robustness against noise and arbitrary deletion and insertion, from which the early syntactic approaches to pattern recognition suffered, have been overcome. We work with 2D- and 3D-data belonging to the most difficult type currently proposed for automatic vision methods. Tests with benchmark data from the photogrammetric community have been successfully carried out [14]. On the other hand theoretic investigations also proceed for our approach, and we can already give some sound results and some proper assessments on the computational complexity for our parsers and language hierarchies.

Current focus of attention is speed up by evaluation criteria, special hardware and inclusion of probability attributes to gain semantics with some statistic rigor.

References

[1] R. H. Anderson. Syntax-directed recognition of hand-printed two-dimensional mathematics. In M. Klerer and J. Reinfelds. *Interactive systems for experimental applied mathematics.* Academic Press, New York, 436–459, 1968.

[2] E. Gmür and H. Bunke. 3-D Object Recognition Based on Subgraph Matching in Polynomial Time. In: R. Mohr, Th. Pavlidis and A. Sanfeliu. *Structural Pattern Analysis*, World Scientific, Singapore, 131–147, 1989.

[3] J. Feder. Plex languages. *Inform. Sci.* 3: 225–241. 1971.

[4] K. Marriott. Constraint Multiset Grammars. *IEEE Symposium on Visual Languages*, 118–125. 1994.

[5] E. Michaelsen. 3D coordinate grammars. B. Girod, H. Niemann and H.-P. Seidel. *3D image analysis and synthesis '96* Infix, Sankt Augustin, 81–85, 1996.

[6] E. Michaelsen. Über Koordinaten Grammatiken zur Bildverarbeitung und Szenenanalyse. University of Erlangen, Thesis, (in preparation), 1998.

[7] D. L. Milgram and A. Rosenfeld. A note on 'grammars with coordinates'. In F. Nake and A. Rosenfeld. *Graphic Languages*, North Holland, 187–194, 1972.

[8] A. Nakamura and K. Aizawa. Relationships between coordinate grammars and path controlled graph grammars. *Int. J. of Pat. Rec. and A. I.*, 3: 445–458, 1989.

[9] A. Nakamura. Some notes on parallel coordinate grammars. *Int. J. of Pat. Rec. and A. I.*, 9: 753–761, 1995.

[10] H. Niemann. Pattern analysis and understanding. Springer, Berlin, 1990.

[11] A. Rosenfeld. Coordinate grammars revisited. *Int. J. of Pat. Rec. and A. I.*, 3: 435–444. 1989.

[12] A. C. Shaw. A formal picture description scheme as a basis for picture processing systems. *Information and Control*, 14: 9–52, 1969.

[13] U. Stilla. Map-aided structural analysis of aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 50(4): 3–10 1995.

[14] U. Stilla, E. Michaelsen and K. Lütjen. Automatic extraction of buildings from aerial images. In F. Leberl, R. Kalliany and M. Gruber. *Mapping buildings, roads and other man-made structures from images,* 229–244. Oldenburg, Wien, 1996.

[15] U. Stilla and E. Michaelsen. Semantic modelling of man-made objects by production nets. In A. Gruen, E.P. Baltsavias and O. Henricsson. *Automatic extraction of man-made objects from aerial and space images (II).* Birkhäuser, München, 1997.

[16] K. Tombre. Structural and syntactic methods in line drawing analysis: To which extent do they work? In P. Perner, P. Wang and A. Rosenfeld. *Advances in structural and syntactical pattern recognition.* 310–321. Springer, Berlin, 1996.

[17] W. H.Tsai and K. S. Fu. Attributed grammar - a tool for combining syntactic and statistical approaches to pattern recognition. *IEEE Trans. SMC*, 10: 873–885, 1980.