A syntactic approach based on distortion-tolerant Adjacency Grammars and a spatial-directed parser to interpret sketched diagrams

J. Mas \textsuperscript{a,b},*, J. Llados \textsuperscript{a,b}, G. Sanchez \textsuperscript{a,b}, J.A.P. Jorge \textsuperscript{c}

\textsuperscript{a} Computer Vision Center, Edifici O, Bellaterra 08193, Spain
\textsuperscript{b} Computer Science Department, Universitat Autonoma de Barcelona, Edifici Q, Bellaterra 08193, Spain
\textsuperscript{c} INESC-ID/IST/UTL, Rua Alves Redol, 9, 1000 Lisboa, Portugal

\textbf{A B S T R A C T}

This paper presents a syntactic approach based on Adjacency Grammars (AG) for sketch diagram modeling and understanding. Diagrams are a combination of graphical symbols arranged according to a set of spatial rules defined by a visual language. AG describe visual shapes by productions defined in terms of terminal and non-terminal symbols (graphical primitives and subshapes), and a set functions describing the spatial arrangements between symbols. Our approach to sketch diagram understanding provides three main contributions. First, since AG are linear grammars, there is a need to define shapes and relations inherently bidimensional using a sequential formalism. Second, our parsing approach uses an indexing structure based on a spatial tessellation. This serves to reduce the search space when finding candidates to produce a valid reduction. This allows order-free parsing of 2D visual sentences while keeping combinatorial explosion in check. Third, working with sketches requires a distortion model to cope with the natural variations of hand drawn strokes. To this end we extended the basic grammar with a distortion measure modeled on the allowable variation on spatial constraints associated with grammar productions. Finally, the paper reports on an experimental framework an interactive system for sketch analysis. User tests performed on two real scenarios show that our approach is usable in interactive settings.

\section{Introduction}

Diagrammatic notations provide excellent means to expressing concepts due to the descriptive power of graphical symbols and spatial arrangements. Graphics Recognition (GR) is the subfield of Document Image Analysis and Recognition (DIAR) concerned with interpreting non-textual information present in document images. This field is important because many documents use a diagrammatic notation: i.e., architectural floor plans, mechanical drawings, electronic circuit diagrams, musical scores, flow charts, etc. In particular sketches which roughly express abstract concepts are a kind of diagrams consisting of freehand line drawings. Because of their concise and expressive nature, sketches are a very effective communication mechanism. Recently, emergent pen-based devices, such as Tablet PCs, Smartphones or Digital Pen and Paper settings, have turned sketches into an interesting human–machine communication modality to explore. Thus, on-line recognition for sketching interfaces has elicited a growing interest among the Human Computer Interaction (HCI) community \cite{35}.

The interpretation of a diagram usually requires two components. First, the recognition of compound objects, i.e., symbols which are instances of an alphabet. Second, recognizing relations between symbols instances provides meaning to the diagram proper. Thus, symbols and relations can be recognized according to a syntax (domain dependent or not), which prescribes an alphabet of valid symbols and a set of rules that define the diagrammatic notation. Valid diagrams can therefore be parsed once symbols are recognized and satisfy valid notation rules. Cleverly, a visual language consists of a graphical alphabet and grammar productions describing relationships between the elements of this alphabet. Since its early formulation by Fu \cite{17}, Syntactic Pattern Recognition, a subfield of Structural Pattern Recognition, has focused on describing and recognizing 2D patterns, both for DIAR but also to understand scenes and images as we review in Section 2.

Due to the coarse expressiveness of sketches, there has been a steady interest in sketch-based human–machine interfaces. This interest has brought together the Graphics Recognition and Human Computer Interaction communities. However, sketch recognition remains a hard problem to tackle, despite the steady
increases in computing power and ever more capable pen-based hardware. Many interesting approaches have been devised to achieve graphical symbol recognition, an overview of which can be found in [30]. However, most of the techniques surveyed concentrate on classifying isolated symbols. On the other hand, the research on diagram recognition has mostly yielded applications that process particular classes of diagrams. Furthermore, these approaches assume that compound symbols were previously segmented and recognized [25], or mostly work with elementary geometrical shapes such as arrows, squares or circles [20,28,29]. Most focus on engineering diagrams where the difficult problem of distortion is neglected or handled using empirical thresholds in angles and distances or reduce the alphabet to very basic objects. While these approaches are adequate to parsing engineering diagrams, they do not cope with the expressive ability of freehand drawings. Several HCI applications have applied classical Pattern Recognition Techniques including Bayesian Networks [1], Hidden Markov Models [38] and 2D Dynamic Programming [16] to recognize more complex shapes. Still, these methods do not scale up very well when dealing with complete diagrams. We are interested in syntactic approaches which offer robust methods to interpret visual languages and provide mechanisms to validate whole diagrams in terms of their symbols and the relations between them. Early syntactic approaches such as Sketch Grammars [10] or more recently Context-Driven Constraint Multiset Grammars [32] are examples of frameworks amenable to sketch understanding.

One of the key requirements to a successful sketch-understanding system is being able to handle the variability, imprecision and coarseness which are typical of hand made strokes. Another important feature to consider is the order-free-nature of human drawings in contrast to the sequential nature of textual languages. While in some applications the order of strokes can be important (e.g. in forensics when recognizing signatures or in Chinese calligraphy), the same diagram can be drawn using many different stroke sequences by different people and still be correctly recognized. Furthermore, sketches can be captured in on-line by means of a pen-device, or off-line by means of a scanner or a camera. In the latter case any “natural order” is lost in the capture. A good syntactic approach to describe sketched diagrams should combine a high descriptive power, tolerant to the distortion inherent to sketches with the ability to cope with the freedom in the drawing order of the input primitives.

The main objective of this work is to propose a robust syntactical framework to describe and recognize sketches. Our work proposes a grammatical formalism which explicitly handles a distortion model. A spatially aware on-line chart-based parsing mechanism can analyze complex visual languages, from the recognition of sketched graphical symbols to the interpretation of diagrammatic visual sentences. The work described in this paper contributes a syntactic approach to describe sketched diagrams. To allow freedom in the input sentence we propose a grammatical formalism derived from Adjacency Grammars [23,24], whereby productions are defined over multisets of terminal and non-terminal symbols instead of a sequence of symbols typical of linear languages. The distortion model is defined in terms of attributes of the symbols appearing in grammar productions. When evaluating productions we synthesize a distortion attribute, measured over the structural properties of the graphical symbols, which represents a quantitative estimation of how far the instance of the left-hand-side deviates from the ideal model embodied in the production.

Another key contribution of this work relies on the parsing model. To allow for on-line interpretation we propose an incremental parser which analyzes the input as strokes are drawn, i.e., it does not wait until the diagram is completed to yield its analysis. As we have seen above, in most freehand drawings the parser cannot assume that the primitives follow a sequential order. As a rule, the most recent symbol input may or may not yield a valid reduction with the immediately preceding symbol recognized. Then a search among all previously input symbols is needed to find candidates which may form valid reductions. Left unchecked this mechanism would entail exponential time complexity of the parser. A key observation is that most visual languages rely on spatial proximity and adjacency relations. Thus, to reduce the complexity while retaining expressiveness we propose an indexing mechanism based on spatial enumeration. Given an input symbol, the parser works bottom-up using a spatial grid as a look up table to find candidate symbols, matching appropriate spatial relations, to find valid reductions.

The rest of the paper describes our work as follows: Section 2 surveys related syntactic approaches to interpreting diagrams. Section 3 describes our grammatical formalism based on Adjacency Grammars. Section 4 introduces the grid directed parsing algorithm. Section 5 presents an application scenario based on sketches where our work is applied. Section 6 presents the experimental evaluation and user tests. Finally, Section 7 is devoted to conclusions and future work.

2. State of the art of syntactic approaches for 2D diagram understanding

The use of syntactic approaches to interpret diagrams has been widely studied in the literature. In this section we present some of the relevant works. A syntactic approach consists of two different but interrelated parts. First, the grammatical formalism that is used to describe valid diagrams. Second, the parsing methodology that checks whether a given input pattern belongs to the language as defined by the grammar or not. In the case of visual languages the parser recognizes pictorial sentences.

Definition 1. Formally, a Grammar is defined as a 4-tuple $G = (V_T, V_N, S, P)$ where:
- $V_T$ is a finite set of terminal symbols,
- $V_N$ is a finite set of non-terminal symbols,
- $S$ is the initial symbol of $G$,
- $P$ is the set of productions or rewriting rules of $G$ with the form: $\alpha \rightarrow \beta$ with $\alpha, \beta \in (V_T \cup V_N)^*$.

Given a production of a grammar of the form $\alpha \rightarrow \beta$, we refer to $\alpha$ as the left hand side (LHS), and to $\beta$ as the right hand side (RHS). The process of substituting the symbol in the LHS by the symbol in the RHS, is known as derivation. The substitution in the backward direction is known as reduction.

Definition 2. Let $\Sigma$ be the alphabet of a grammar $G$ and $\Sigma^*$ the set of all the sentences generated with the combination of the elements of $\Sigma$. We define the language generated by a grammar $G$, $L(G)$, as the subset of sentences in $\Sigma^*$, $L(G) \subseteq \Sigma^*$, generated by $G$, as a derivation from $S$.

A grammatical formalism can be characterized by different attributes. We propose the taxonomy presented in Table 1. In this table we define two criteria to characterize the different grammatical formalisms, namely dimensionality and relationship mechanisms. The former indicates the dimension of the structure used to describe a visual concept. Grammars are usually divided into linear (e.g. string structures) and $n$-dimensional grammars (arrays or graph structures). The second criterion describes the way in which the grammar expresses the relationships between...
the symbols in its productions. The relation can be expressed as a key word denoting a relational operator or can be defined as a function. Finally, the last two columns in Table 1 describe particularities of the grammar and whether the grammar derives from another type of grammar.

2.1. Linear grammatical formalisms

Linear Grammars can be divided into three categories taking into account the second criterion of Table 1. Each category represents the relations between the symbols of a production with operators, connection points or functions, respectively. The first group uses a set of reserved words as operators that represent different relations. The productions of the grammars are expressed as a linear concatenation of elements and operators between element pairs. A simple example of production is $a + b$ where the symbol $c$ is formed by the symbols $a$ and $b$ and the operator $+$ indicates a particular relation between $a$ and $b$ (e.g. incidence, concatenation, etc.). Picture Description Languages (PDL) [5,39] describe symbols with two end points (head and tail). The operators describe the way the symbols are concatenated between their head and tail points. Pictures should be connected, by adding a blank symbol defined with head and tail to allow the description of not joined symbols. Enhanced Positional Formalisms (EPF) [11] define a positional operator describing, given a symbol, at which position in the sequence the parser can find the next one. The inclusion of a factorization operator allows these formalisms to express a relation between a symbol and two or more symbols at a time. Positional Grammars (PG) [7] and Extended Positional Grammars (XPG) [8] can express joining or positional relations between symbols. Moreover, they describe explicitly in the operator which of the previous symbols in the production are related with the next one. Sketch Grammars (SG) [10] are an improvement of PG used to describe and interpret sketches. This kind of grammars not only use positional relations but also temporal relations. The distortion is modeled by a set of thresholds in the relations between the symbols. These thresholds describe the tolerance of the relation.

On the positive side, the grammars in this category have a simple formulation. On the negative side, they have a limited description ability, e.g. PDL only can express concatenations between symbols. Moreover, different symbols can be described with the same production. The ordering of the input is conditioned to the order of appearance of the symbols in the production. Concerning distortion, the majority of grammars do not define a distortion model.

The second group of linear grammars consists of grammars that define a set of connection points for each element. Relations are therefore defined in terms of links between connection points. Thus, having the symbols $a$ and $b$ defined with two attaching points, a production in this category could be represented as $c \rightarrow (ab)21$ where the symbol $c$ is composed by the symbols $a$ and $b$ with the connection of the second attaching point of $a$ to the first attaching point of $b$. Plex Grammars [5,15] belong to this group. These grammars propose a join strategy defining symbols with $n$-attaching points called napes. The construction of the grammar productions is as simple as it is in the case of operators based grammars. Moreover, their descriptive power is also limited to the connection points defined in the napes. The ordering of the input is implicitly determined by the symbols in the production and no distortion model is presented.

The last category of linear grammars consists of grammars that use functions to define the relations between the elements on the grammar. These formalisms are based on Attributed Multiset Grammars. It means that a production is defined as a multiset, a set that allows repeated instances of the same symbol and a set of functions representing the relation between the elements. A production of this category can be represented as $c \rightarrow \text{Incident}(a,b)$ where the function Incident evaluates whether one of the extremes of $a$ touches the primitive $b$. Different approaches of this category are determined depending on whether the context is considered or not in the definition of grammatical productions. While Relation Grammars (RG) [12] do not define context in the productions of the grammar, Constraint
Attributed Programmed Graph Grammars (APGG) [3] define a subgraph matching technique to carry out the direct embedding. The attributes of the new graph are computed with the attributes in the analyzed production. Similarly, Fahmy and Blostein [14] define a graph transformation paradigm able to cope with some noise. n-Dimensional grammars are the most descriptive among all the grammatical formalisms reviewed in this section. However, they are complex to construct and the addition of a distortion model further increase their complexity in terms of grammar description and parsing.

2.3. Parsing techniques associated to each grammatical formalism

Parsers paradigms can be characterized by two main features. First, at the time that a parser analyzes the input it constructs a tree where the leaves represent instances of symbols in Vf and at each level the different non-terminal symbols forming VN are found. The way this tree is constructed characterizes a parser paradigm. Thus, depending on whether the parse tree is constructed from the root to the leaves or the other way round, the parser paradigm is known as Top-down or Bottom-up, respectively. The second feature is Directionality, which indicates the way that an input is analyzed. A parser would be directional if the input is analyzed from a left to right or right to left order. Non-directional parsers analyze the input in a non-predefined direction, i.e., they group or divide the input following the productions of the grammar. Directional parsers can manage the input token by token. This allows them to start the parsing process of a given input before it is completed. On the other hand, non-directional parsers need all the input before starting to analyze it. This is because the parser constructs an exploration space with all the possible partitions of the tokens forming the input. Inside directional parser there is a family parsing an input in linear time. This family of parsers is known as Linear parsers. There are a wide variety of parsers used with textual languages that inspired the techniques used to describe graphical sentences. Among all the parser paradigms, outstanding ones are non-directional methods like Unger [42] and CVK [26] parsers, directional methods like Earley [13] and Recursive Descent Parsers [19], and linear methods such as LL(k) [19] and LR(k) [27] algorithms. See Table 2 for a summary of the different categories.

According to the grammatical formalisms presented before, a number of parsing methodologies have been developed. Shaw [39] presents a top-down parser based on Unger's method for PDL. Couasnon [11] presents a parsing algorithm based on first order logic to parse sentences with an EPL grammatical formalism. LR techniques have been developed to work with the PG, XPG and SKG formalisms in [7,8,10], respectively. Bunke and Haller present [4] a parsing algorithm derived from the Earley algorithm for context-free PDL grammars. The works of Marriot [34] and Mace and Anquetil [31] present an incremental parsing algorithm that copes with CMG. This kind of algorithms may be seen as directional bottom-up. Crimi et al. [12] present a top-down parsing algorithm corresponding to directional parsers. The parser

![Fig. 1. A 2D sentence and its representation using different categories of linear grammar: (a) the original sentence, (b) using operators, (c) using connection points and (d) using functions.](image-url)
Bunke [6] presented three criteria to define the suitability of a set of graphical elements that follow a spatial distribution. A diagram, as mentioned before, can be seen as a representation of a graph. An Adjacency Grammar to describe diagrams has been developed in order to adapt it to work in a graphical domain. More precisely a formalism to describe graphical sentences and the different grammar and the relations between those elements. The tokens of the grammar follow this structure: A grammar production of the form $g \rightarrow a$ represents a constraint defined in the symbol $a$.

$g \rightarrow a$ for $g \in G$, $a \in VT$, and $G$ is a set of attributes defined over the symbols of the productions. The relations $R$ are described by a set of non-terminal symbols. With $VT$, $VN$, and $S$ are the set of terminal symbols, non-terminal symbols, and the start symbol of $G$. $P$ is the set of predefined constraints between the symbols in the productions. $A$ is a finite set of attributes defined over the symbols of $G$.

### 3.1. Adjacency Grammars: theoretical framework

Starting from the definitions introduced in Section 2, let us define the set of concepts needed to understand the syntactic approach proposed in this work.

**Definition 3.** An Adjacency Grammar is defined as a 6-tuple $G = \{VT, VN, S, P, R, A\}$ where:

- $VT$ is a finite set of terminal symbols.
- $VN$ is a finite set of non-terminal symbols. With $VT \cap VN = \emptyset$.
- $S$ is the start symbol of $G$.
- $P$ is the set of production rules of $G$.
- $R$ is the set of predefined constraints between the symbols in the productions.
- $A$ is a finite set of attributes defined over the symbols of $G$.

The productions of the grammar follow this structure:

$$\alpha \rightarrow \{\beta_1, \ldots, \beta_n\} \text{ if } r_1(\Gamma_1), \ldots, r_k(\Gamma_k)$$

where $\alpha \in VN$, $\beta_i \in \{VT \cup VN\}$. The relations $r_i \in R$ for $I = 1, \ldots, k$ represent a constraint defined in $R$ that is necessary in order to validate the rule. Each constraint is defined over a set of objects present in the right hand side of the production, $\Gamma_i \subset \{\beta_1, \ldots, \beta_n\}$ for $i = 1, \ldots, k$.

**Definition 4.** Given a grammar production of the form $\alpha \rightarrow \{\beta_1, \ldots, \beta_n\}$, we define as a constraint $r$ the function that describes a relation between the elements of a subset $\{\beta_1, \ldots, \beta_n\}$. It is defined as: $r : \{\beta_1, \ldots, \beta_n\} \rightarrow \{true, false\}$ where $\beta_i$ are the symbols of the grammar that belong to $\Gamma$ and $r$ is computed from such symbol attributes. Let $|\Gamma|$ be the cardinality of $\Gamma$, we can have unary, binary, ..., $n$-ary constraints depending on $|\Gamma|$.

The symbols of the grammar $G$ are described by a set of attributes $A = \{a_1, \ldots, a_n\}$ in terms of domain dependent features. Then we define the attributing function $\gamma$, the function that given a symbol $\beta \in \{VT \cup VN\}$ returns the attribute’s values of $\beta$ denoted as: $\gamma(\beta) = (a_1, \ldots, a_n)$ where $a_i \in A$.

**Definition 5.** Given a grammar production of the form $\alpha \rightarrow \{\beta_1, \ldots, \beta_n\}$, we define the synthesizing function $f$ as the function that computes the attributes of the symbol in the LHS from the terms of domain dependent features. Then we define the attributing function $\gamma$, the function that given a symbol $\beta \in \{VT \cup VN\}$ returns the attribute’s values of $\beta$ denoted as: $\gamma(\beta) = (a_1, \ldots, a_n)$ where $a_i \in A$.
attributes of the symbols in the RHS. Formally $f$ is denoted as

$$f : A_{\beta_1} \times \cdots \times A_{\beta_n} \rightarrow A_x$$

Fig. 2 shows an example of a description using Adjacency Grammars. Fig. 2(a) presents the formulation of the Adjacency Grammar. Fig. 2(b) shows the primitives or terminal alphabet of the grammar: a triangle, a quad and a circle. Fig. 2(c) shows a valid visual sentence. This sentence consists of three terminals validating the productions RECTANGLE and RECTRI. The rectangle symbol is composed by the quad with attributes $(x_0,y_0) = (0,0)$ and $bb = \{(0,0),(10,10)\}$ and the circle with attributes $(x_c,y_c,r) = (5,5,2)$ and $bb = \{(3,3),(7,7)\}$. In this case the synthesizing functions compute the attributes as follows:

$$f(\text{RECTANGLE} \lVert (x_0,y_0),bb) = \{(0,0),(0,0),(10,10)\}.$$

The attributes for the RECTRI symbol are calculated in the same way.

3.2. Sketched diagram description by means of Adjacency Grammars

Two main properties of sketches are: the freedom of their drawing order and their inherent distortion. A syntactic approach to describe sketches must accomplish these two properties and a third one consisting on having a high descriptive power. Our approach is based on Adjacency Grammars, it belongs to the function based formalism that, as we reviewed in Section 2, is the most expressive among linear grammars. The way the productions are defined makes inherent the first feature to this kind of grammars. The use of multisets instead of list of elements allows not to predefine an order in the input of the symbols. Before defining the distortion model which is an important feature of our approach, we have adapted Adjacency Grammars (cf. Definition 3) to cope with a graphical domain. The terminal symbols represent the basic elements of diagrams to be parsed. In our case, the set of terminal symbols $V_T$ is defined by segments and arcs, $V_T = \{\text{segments,arcs}\}$. The set of non-terminal symbols includes the graphical elements with specific meaning. The elements of this set are those defined by the productions of the grammar.

A set of attributes, $A$, has been defined according to the domain. Depending on whether the symbol is terminal or non-terminal the attributes are different. In the case of terminal symbols, segments are attributed by the tuples $(x_0,y_0)$ and $(x_f,y_f)$ denoting the end points of the segment. Arcs are described by the tuple $(x_c,y_c,r)$ where $x_c$ and $y_c$ represent the $x$ and $y$ coordinates of the arc center and $r$ is the radius. Non-terminal symbols are described by a 4-tuple $(x,y,\theta,ch)$ where $x,y$ is the position, $\theta$ is the orientation, and $ch$ is the convex-hull. These attributes serve to compute the constraints between them and to synthesize the attributes of the new symbol reduced when a production is valid.

To add new attributes in the primitives such as the end points of the arc does not mean to reformulate the grammatical formalism. In addition, it increases the expressive power of the grammar since new constraints can be associated to the new added attributes, e.g., the incidence between a segment and an arc.

The constraints of our grammar represent spatial and structural relations and differ depending on the kind of symbol appearing in the production. Constraints for terminal symbols are parallelism, incidence and perpendicularity. Other constraints such as neighbourhood and inclusion refer to relations between terminal and non-terminal symbols or between non-terminal symbols. The intersection constraint belongs to both categories. Hence, $R=\{\text{parallelism, incidence, perpendicularity, neighbourhood, inclusion}\}$.

Since the aim of this work is to interpret sketches, independently if they are acquired off-line or drawn by the user in an online framework, distortion is a key issue in our grammatical formalism. It is present in both graphical elements and relational constraints between them. Fig. 3 illustrates different types of distortion to model. Fig. 3(a) and (b) illustrate distortions in the drawing of symbols. In the case of Fig. 3(c), the constraint intersection is represented but with a high distortion. The distortion model is an attribute passed through the symbols in the grammatical productions. It is estimated as the variation of the relational constraints between the symbols and their ideal model. For example, the perpendicular constraint between two segments is computed in terms of the normalized angle between them. Its formulation is as follows:

$$\text{perp}(\beta_1,\beta_2) = \frac{\langle \beta_1,\beta_2 \rangle}{90}$$

Fig. 2. Example of an Adjacency Grammar and a visual sentence that belongs to its language: (a) the Grammar, (b) the terminal alphabet and (c) the visual sentence.

Fig. 3. Distortions: (a) junctions (b) angles and (c) constraints (intersection in this case).
Then, the formal description of constraints given in Definition 4, is modified as follows: \( r : O_1 \times \cdots \times O_n \rightarrow \{0 \ldots 1\} \), indicating the variation regarding the ideal constraint. The new definition of the constraints forces to reformulate the productions of the grammar. Then, the productions are defined as

\[ \alpha \rightarrow \{\beta_1, \ldots, \beta_j\} \quad \text{if} \quad r_1(\Gamma_1, c_1), \ldots, r_k(\Gamma_k, c_k) \]

where each \( c_i \in \{0, \ldots, 1\} \) represents the distortion measure.

An example of production representing a right-angled triangle is as follows:

\[
RT \rightarrow \{\text{Seg}_1, \text{Seg}_2, \text{Seg}_3\} \text{Incident}(\text{Seg}_1, \text{Seg}_2, c_1) \}
\]

\[
\text{Incident}(\text{Seg}_1, \text{Seg}_2, c_2) \} \text{& Incident}(\text{Seg}_1, \text{Seg}_3, c_3) \} \text{& Perpendicular}(\text{Seg}_1, \text{Seg}_2, c_4)
\]

The attributes of the constraints are synthesized in the distortion attribute of the symbol \( \alpha \) when a reduction is achieved. The function \( f \) is defined by the following formulation:

\[
f(\alpha)_{\text{dist}} = \frac{1}{2} \left( \frac{1}{k} \sum_{i=1}^{k} \gamma(\beta_i)_{\text{dist}} + \frac{1}{k} \sum_{i=1}^{k} c_i \right)
\]

In particular in the above equation \( f(\alpha)_{\text{dist}} \) refers to the distortion attribute synthesized when \( f \) is applied.

Given an input representing a compound symbol of the grammar, the distortion of the symbol is defined by the variations on the constraints performing it, but also in terms of the distortions of the possible sub-shapes forming it. To estimate the distortion of \( \alpha \) we assign the same weight to both distortions of sub-shapes and constraint variations. Then, given the production representing a right angled triangle, defined above, the distortion is measured as follows:

\[
f(\text{RT})_{\text{dist}} = \frac{1}{2} \left( \frac{\gamma(\text{Seg}_1)_{\text{dist}} + \gamma(\text{Seg}_2)_{\text{dist}} + \gamma(\text{Seg}_3)_{\text{dist}}}{3} + c_1 + c_2 + c_3 + c_4 \right)
\]

The formulation of the constraints has to take into account the ideal case and parameterize it to estimate the variation measure. For example, if we want to describe the constraint inside we may look at the area of an element that overlaps with the other element. The ratio between the total overlapped area and the area of the object gives us an idea of which is the distortion value of this constraint. We have decided to implement this distortion measure in order to avoid to train the system with a huge quantity of samples. The use of other possibilities such as modeling a probability function for each constraint requires a previous learning step and an approximation based on a Bayesian model, such as the work of Shilman et al. [40].

From a general point of view, our distortion model may look similar to the one proposed by Costagliola et al. [10] with Sketch Grammars. The authors propose a set of manually defined thresholds in the relations between the elements. Fig. 4(a) shows a SkG production describing an Arrow symbol. As we can see, different thresholds have been defined in the relations, e.g. \( t_1 \) between the junctions of \( \text{LINE}_1 \) and \( \text{LINE}_2 \). Given an Arrow symbol instance such as presented in Fig. 5, the SkG formalism rejects the input since the threshold \( t_1 \) is not accomplished, while our formalism obtains an instance of the symbol Arrow with a distortion of \( \text{Arrow}_{\text{dist}} = 0.5(c_1 + c_2 + c_3)/3 \).

Another kind of distortion that occurs while working with sketches is the phenomenon of spurious strokes. As mentioned by Mahoney and Fromherz [33], this factor is common when drawing. The authors propose a solution based on creating a new junction between two segments and eliminating the possible segment between them. In our case, we propose a grammatical production that pre-processes the basic primitives and grouped together those that can be candidates to be spurious strokes. Other phenomena such as overtracing, overlapping or gaps are treated in the same way, see Fig. 6 for a description of these productions where colinear represents two lines that are near and follow the same direction.

Once we have described the grammatical formalism used in our work, let us explain the parser approach used to recognize diagrams.

4. A parsing technique to recognize diagrams

This section presents the second contribution of our work: a parsing technique to recognize diagrams. First, we introduce our parsing methodology and the problems of parsing 2D structures. Second, a mechanism to reduce complexity based on an indexing structure is presented. Finally, an analysis of the complexity is reported.

Before analyzing the parsing engine in detail, let us briefly outline the pre-process of extracting the basic primitives from the input. This process is known as lexical analyzer or scanner. In a graphical domain, the lexical analyzer recognizes the input from graphical elements to primitives carrying geometrical and topological information. We propose two types of lexical analyzers depending on whether the input is captured on-line,
pen or mouse events, or off-line, raster images acquired by a scanner or a camera. In on-line modes the input consists of strokes. A stroke is a set of points captured between two consecutive pen-down and pen-up actions. Each stroke may represent one or more basic primitives. For each stroke the lexical analyzer applies a polygonal approximation [37]. In the case of off-line input modes the lexical analyzer consists on a classical vectorization step [41].

4.1. A directional incremental parser

Among all the parser categories presented in Section 2 our parser methodology belongs to the directional class. In particular, our directional parser scans the input in a time sequential order, analyzing it symbol by symbol. As we need to cope with on-line input modes, the selected parser does not need the completed input to start analyzing it. According to the way the parse tree is constructed, our methodology is a bottom-up parser, i.e., it applies reductions from the input to the start symbol of the grammar. Then, given a new terminal symbol the parser has to search for all the possible productions that this symbol can reduce. Two different search techniques could be applied: Breadth First Search and Depth First Search. The first one, opens all the possible productions at the same time while the last one gives one production at a time and if no possible reduction is found the parser backtracks and selects another production. This entails more complexity, since each time the production selected does not produce a valid reduction, the parse tree must be reconstructed from a previous step. For that reason the proposed methodology uses a Breadth First Search. This technique leaves all the candidate productions open until they are not valid or the input is finished.

Given a non-terminal symbol \( x \) of the grammar defined by a production \( x \rightarrow \{ \beta_1, \ldots, \beta_j \} \), we denote \( F_x = \{ \beta_1, \ldots, \beta_j \} \) as the set of symbols forming \( x \) which have been already reduced or analyzed; and \( M_x = \{ \beta_1, \ldots, \beta_j \} \setminus F_x \) as the set of missing symbols in \( x \) to be completely reduced. For a given symbol \( x \), \( F_x \cup M_x = \{ \beta_1, \ldots, \beta_j \} \) and \( M_x \cap F_x = \emptyset \). From now on we define under reduction symbols as the non-terminal symbols where \( M_x \neq \emptyset \), and reduced symbols as the non-terminal symbols with \( M_x = \emptyset \).

When scanning a symbol \( x \in V_T \) the parser has to take into account the under reduction symbols and the new possible symbols as:

- For each under reduction symbol \( x \), the parser has to check:
  - The symbol \( x \) invalidates an under reduction symbol: \( x \in M_x \) but not satisfies the constraints in the production, or \( x \notin M_x \) and invalidates possible successive relations between the symbol \( x \) and other possible terminals. Then, \( x \) is eliminated as under reduction symbol.
  - For each new symbol that \( x \) can generate:
    - The symbol \( x \) and its neighbours, \( NB \), produce a valid partial reduction of a production creating an under reduction symbol \( x \). I.e., \( \{ x \} \cup NB \subset F_x \). If this does not occur:
    - The symbol \( x \) and its neighbours, \( NB \), produce a valid reduction of a production creating a reduced symbol \( x \). I.e., \( \{ x \} \cup NB = F_x \).

In the case of a completely reduced non-terminal symbol, the parser takes into account also the under reduction symbols and the new possible ones. The parser proceeds in the same way when the symbol being analyzed is a non-terminal symbol.

When analyzing 2D structures in a sequence of primitives, each time the parser analyzes a new input symbol it has to search among the previous input symbols to find which ones can be combined with the new one to produce a valid reduction. The input needs to be re-analyzed and could reduce a valid production from symbol that are far one from the other. Fig. 7 shows a graphical description of this. Given the production:

\[ QC \rightarrow \{ \text{Quad}, \text{Circle} \} \ Above(\text{Quad}, \text{Circle}) \]

and the input of Fig. 7(a) where the numbers represent the input order of the symbols. When the symbol Circle is scanned, Fig. 7(b), the parser looks for the previous analyzed input and tries to apply the production QC. Since the previous analyzed input contains two instances of the symbol Quad, the production QC is applied two times. From the production description we can see that the production is accomplished by the Circle and the two Quad symbols. There, the parser not only has re-analyzed all the previous input but it has created two QC symbols. One of these symbols is composed by a Quad and a Circle that are sufficiently far from each other to belong to different symbols. This fact becomes one of the main drawbacks when parsing 2D structures. Then, when the third symbol Quad is analyzed, the parser re-analyzes the previous input and a new production QC is reduced. An indexing structure based on a grid and a neighbourhood pattern to search candidates are proposed to solve the problems of re-analysis and redundancy. Let us further describe the indexing structure.

4.2. An indexing structure to reduce the parser complexity

To avoid the combinatorial search among the primitives of the input candidates that produce a valid reduction, we have defined

![Fig. 7. Analysis of visual sentences: (a) a visual sentence and (b) its corresponding input analysis.](image-url)
an indexing mechanism in the parsing algorithm to reduce the search space. This mechanism is based on a spatial division. A spatial grid is constructed where each cell contains the symbols appearing in these positions of the space. Then, a symbol in the input is indexed in the spatial division. To search candidates in the previously scanned input, an influence zone for the symbols is defined. This zone is defined by the indexing cells and the set of cells obtained by applying a neighbouring pattern to the previous ones.

**Definition 6.** Being $C = \{C_1, ..., C_n\}$ the set of cells obtained by dividing the space into $n$ parts and being $O$ be a symbol representing a graphical element in a diagram. We define $CO(O), NC(O) \subseteq C$ as the set of cells containing the symbol and $NC(O), NC(O) \subseteq C$ represents the set of neighbouring cells for each of the cells in $CO$. We define the influence zone $\Psi$ of a symbol $O$ as the union of the sets $CO$ and $NC$, $\Psi(O) = CO(O) \cup NC(O)$.

Being $SI = \{Y_1, ..., Y_m\}$ the set of previous parsed symbols and $P = \{P_1, ..., P_j\}$ the set of grammatical productions then, when a new terminal symbol $x$ is scanned from the input, the parser selects a set of previously parsed symbols $SI \subseteq SI$ so that for all $Y_i \in SI$, $\Psi(x) \cap CO(Y_i) \neq \emptyset$. If $Y_i \in VT$ or $Y_i \in VN$ and it is a reduced symbol, the parser applies the set of productions $P \subseteq P$ containing $Y_i$ and $x$ as parts. Otherwise, if $Y_i \in VN$ is under reduction, the parser tries to add the symbol $x$ to $Y_i$. If $x$ is part of $Y_i$ and no more symbols are missing in $Y_i$, $Y_i$ becomes a reduced symbol, otherwise $x$ is added to $Y_i$.

When the parser finalizes the analysis of the input, the elements composing the sketch are placed in the nodes of the parse tree. As the parser also allows under reduction entries it is possible that when it finalizes, the drawing is not completely parsed and the initial symbol of the grammar is not reached. Then, the parser returns the finished elements that have been found during the process.

Fig. 8 outlines the parsing paradigm. As we can see, there are four levels: the input, the lexical level, the grid level and the parse tree.

For each new symbol in the input the parser indexes it into the grid level and obtains the set of its neighbouring symbols. Then the parser applies the corresponding productions and constructs the corresponding parse tree. This algorithm is shown in Fig. 9. This algorithm uses two additional functions, namely Validate and Finalize. The Validate function, given a symbol $x$ and a reduced symbol $Y_j$, checks if the symbol $x$ and $Y_j$ belongs to production $p$, i.e., the symbols $x$ and $Y_j$ belongs to the missing symbols of $p$ and accomplishes the constraints belonging to them. Then, a new symbol $O$ is constructed. The Finalize function given a symbol $x$ and an under reduction symbol $Y_j$, checks if the symbol $x$ is one of the symbols of the missing set of $Y_j$ and checks the corresponding constraints. The algorithm is shown for a symbol $x \in V_T$. The algorithms proceeds in the same way if $x \in V_N$, and it is a reduced symbol.

For the sake of understandability Fig. 10 shows an example on how the parser works for a sketch drawn on-line. Given the following productions of a sample grammar:

\[
S \rightarrow SYM1|SYM2|SYM3
\]

\[
SYM1 \rightarrow \langle Rectangle, Triangle \rangle \text{Inside}(Rectangle, Triangle)
\]

\[
SYM2 \rightarrow \langle Rectangle, Triangle \rangle \text{LeftOf}(Rectangle, Triangle)
\]

\[
SYM3 \rightarrow \langle Rectangle, Circle \rangle \text{Inside}(Rectangle, Circle)
\]

being at the situation of Fig. 10(a) where a circle has been drawn. Then a symbol Rectangle is drawn and is placed into the grid and the parser tries to find a symbol in its neighbourhood, see Fig. 10(b). The quads in dark grey represents the influence zone of the Rectangle. As no symbol is found the parser continues scanning the input. Then the parser scans the third graphical element, in this case a symbol Triangle, see Fig. 10(c), the parser indexes it into the grid and searches for neighbouring symbols, see Fig. 10(d). Afterwards the parser finds the previous symbols and tries to apply the productions of the grammar containing at
least one instance of each symbol. First, the parser tries to apply SYM 1 but the constraint Inside is not valid for these symbols. Then the parser tries production SYM 2. This production is valid and the parser reduces the symbol SYM 2. There is no production with the Triangle and Circle as parts. Thus, any production is applied using them.

4.3. A study of the parsing complexity

To analyze the complexity problem of our parsing algorithm we focus on the membership problem of a visual sentence to the language generated by a grammar $G$. Given a sentence $w$, with $\tau = |w|$, the parser algorithm analyzes each terminal symbol $x_i$ and tries to reach the initial symbol of the grammar by applying reductions that depend on the grammatical productions.

Being $x_i$ a terminal symbol of the grammar, we denote $P$ as the set of productions that have the symbol $x_i \in \{\beta_1, \ldots, \beta_t\}$ and $\rho = |P|$. We also define $NB$ as the set of neighbouring symbols of $x_i$, i.e., $Y_j \in SI$ with $CO(Y_j) \cap \Psi(x_i) \neq \emptyset$, and $v = |NB|$. Following Algorithm 1 the complexity of the parser for each terminal input is $O(\rho v)$ being in the worst case $O(\rho v)$ for the complete sentence.

To conclude, the higher bound of the parser complexity is $O(m^2)$, for each input primitive, being $m$ the maximum between $\rho$ and $v$. This bound corresponds to grammars that define several productions with the same symbols and with a high number of neighbours for each symbol in a sentence. The size of the cells of the indexation structure affects the parameter $v$. High values of cell size increases the number of neighbours that may produce a valid reduction since the portion of the space influenced by a symbol increases.

Fig. 11. Sample instances of both languages: (a) and (b) sketch floor-plans and (c) and (d) user interfaces.

5. An application scenario: the use of Adjacency Grammars for sketch understanding

In this section we present a scenario belonging to the domain of sketch diagrams understanding to illustrate the applicability of our approach. We show two grammar examples for constructing graphical sentences using graphical alphabets of two different domains. The first one describes architectural drawings in terms of building symbols like doors, walls, windows, etc. The second one is designed to describe valid designs of user interface described in terms of graphical widgets. Some examples of valid graphical sentences corresponding to both grammars are seen in Fig. 11.

Concerning the proposed grammatical formalism, the formulation in Section 3.2 is here instantiated to the current cases. As a reminder, $V_G = \{\text{Segments, Arcs}\}$ and $R = \{\text{Incident, Intersects, Perpendicular, Parallel, Inside, Neighbour}\}$. Fig. 12 presents a graphical description of these constraints.

A key step in the parsing approach described in Section 4 is the configuration of the cells $C$ for the spatial tessellation. Two division strategies can be considered: fixed or adaptive. The first one divides the space into a regular grid, having all the cells of the same size. The second one consists in an adaptive division of the space using techniques such as X–Y trees, Voronoi Regions, etc. A fixed division requires to set the cell size to the optimum value depending on each application. On the other hand, an adaptive tessellation is more accurate but time consuming because it requires a space redefinition once a new symbol is drawn. This redefinition could cause to refer the parse tree with the new configuration. A tessellation based on a fixed division requires low temporal and memory resources.

Once the space division is defined it is required the setting up of the influence zone of a given symbol. As mentioned in Section 4 the influence zone of a symbol is composed by the union of two sets of cells. First a set with the cells containing the symbol and second a set with the cells obtained by applying a neighbouring pattern over the cells in the first set. To obtain the cells of the first set we start from the bounding-box of a symbol, see Fig. 13(a). This is suitable when the symbols are compound but the methods fails when we have single primitives. In the case of a diagonal line, see Fig. 13(b), using its bounding-box a number of cells will be selected as belonging to the symbol when they do not really belong to it. This is why we improved the selection mechanism dividing into two algorithms depending on whether we have a compound symbol or a primitive. In the case of primitives we use an adaptation of Bresenham’s algorithm [2], see Fig. 13(b) for a graphical description.

To obtain the neighbouring cells for a given symbol, we have decided to apply a neighbouring pattern based on a 8-connectivity. Fig. 14 shows a graphical description. This pattern is applied to each of the cells where a symbol is indexed. In the case of Fig. 14, the pattern will be applied to the four cells containing the triangle. Candidates to possible reductions will be found in the union of the indexed cells and the neighbouring cells.

For the sake of understandability some productions of the grammar are presented in a graphical way, see Fig. 15. The first two productions belong to the grammatical formalism to describe

![Diagram](image-url)
floorplans and the other two to the design of user Interfaces. The use of subscripts in the production Button help us to differentiate between the two symbols of the same category.

6. Experimental evaluation

In this section we present the experimental part of the work. It consists of four experiments in the application scenarios defined above. In the first experiment we present a comparison between our parser and two different parser algorithms, a Golin parser [18] and a G-XpLR parser [9]. The second experiment is devoted to determine the optimum cell size in the space division. The third experiment shows how our parser is able to tackle with some inherent distortions of sketches. Finally the fourth experiment illustrates qualitatively and quantitatively the performance of the recognition algorithm with different sketches.

6.1. Experiment 1: a comparison with other parser

This experiment compares our parsing algorithm with two literature approaches, namely a Golin Parser and a G-XpLR parser.

6.1.1. Golin parser

The first comparison is performed between our parsing algorithm and a Golin parser [18]. According to the state of the art of Section 2 this algorithm works in two steps. First, a bottom-up step constructs all the possible parse trees and afterwards a top-down step rejects those productions that are not valid. The parser has been selected because it works in a similar way as our algorithm. The rest of parsers in the literature construct a parsing table to determine which is the next action to the previous scanned symbols and the next symbol.

Fig. 16 shows a graphical illustration of the samples used in this experiment. The results of the experiment are shown in Table 3. The values in the cells correspond to the number of reduced symbols and under reduction symbols created in the parsing process. As seen, except from the first row, the number of reduced symbols is equivalent but they differ in the number of unreduced symbols. This difference is due to the reduction of the

<table>
<thead>
<tr>
<th>Sample</th>
<th>Grid reduced/under reduction</th>
<th>Golin based parsing reduced/under reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 16(a)</td>
<td>15/1</td>
<td>17/125</td>
</tr>
<tr>
<td>Fig. 16(b)</td>
<td>1/2</td>
<td>1/65</td>
</tr>
<tr>
<td>Fig. 16(c)</td>
<td>8/4</td>
<td>8/86</td>
</tr>
<tr>
<td>Fig. 16(d)</td>
<td>4/0</td>
<td>4/118</td>
</tr>
</tbody>
</table>

Fig. 16. Examples used in the first experiment: (a) Example1, (b) Example2, (c) Example3, and (d) Example4.

Fig. 15. Graphical samples of grammatical productions.

Room :: \{ , , \} Intersects( , , ) &\& Intersects( , )

Bed-Room :: \{ Room, \} Inside( , Room)

Checkbox :: \{ , \} Intersects( , )

Button :: \{ , \} Inside( , )
search space by means of the spatial division and the neighbourhood criteria when a new graphical element is drawn. In the case of the first row the Golin-based parser applies twice the production formed by the two circles and the production formed by the triangle and the circle. While our parsing methodology only applies the production once.

The use of a neighbouring mechanism reduces the number of productions to be checked. This fact results in a most optimum complexity, i.e., a reduction in the parsing time. A real time parsing is one of the main requirements of on-line applications. Using the grid-based lookahead strategy allows our approach to be suited for on-line applications.

6.1.2. LR parser

The second parsing paradigm is a derivation of an LR parser (XpLR) applied to sketch recognition [9]. This parser analyzes the input in a non-sequential way driven by the relations defined in the grammar. This fact entails an increase of the occurrences of parsing conflicts and makes difficult to establish the symbol of sentence with which the parser has to start the analysis. To solve these problems three main features are defined for this parser: incrementality, bi-directionality and non-determinism. The first is essential for the construction of visual interactive frameworks since it allows the provision of immediate feedback to the user. The second allows us to scan the input in opposite directions from an arbitrary starting symbol. The third allows a syntax specification that naturally corresponds to abstract syntax.

Let us analyze in detail the bi-directionality feature. First, given a grammar $G$ two parsing tables are constructed, one for the grammar $G$ and the other for its reverse form $rev(G)$. Then for each state reachable after the occurrence of the starting symbol the algorithm starts an incremental parsing, $forward$ and $backward$. Both parsers interact only when a parser tries to reduce a production. In this case, the parser waits for an opposite parser attempting to apply the reverse version of the same reduction.

Table 4 summarizes the comparative of the G-XpLR formalism and our proposed approach. As can be seen, both paradigms share some properties such as how the scanning of the input is done or the way the input is analyzed. The differences of both paradigms are referred to the properties of bi-directionality, input primitives and distortion measures. The first is defined in our case by a unique parser that analyzes the input in a unique direction. Concerning to the second, our proposal works at segment and arc level. Distortion has a model assigning a distortion rate instead of setting up a threshold associated to each production's constraint. This implies the acceptance of an instance independently of its distortion rather than accepting only distorted instances using a binary decision under the threshold accomplishment.

The most costly step in the G-XpLR algorithm is the construction of both tables. This has no influence in the complexity of the parsing process since it can be done off-line before the analysis starts. Nevertheless, more parsing conflicts occur in the construction of both tables. On the other hand, the way our parser is defined does not imply the appearance of parsing conflicts.

6.2. Experiment 2: choosing an optimum value for the cell size

The second experiment shows a comparative in terms of memory accesses and recognition rates, when the size of the cells in the grid is changed. This comparative allows us to set up a first approximate value for the size of the grid. Given the grammar of Fig. 17(a) and the sample sentence of Fig. 17(b) we have estimated the number of memory accesses and computed the different parse trees for cell sizes of 96, 48, 24, and 12 pixels.

Fig. 18 shows the number of memory accesses when changing the size of the cells. As can be seen, the lower the size of the cells the higher the number of memory accesses. This is because while the size of the cell becomes smaller, a symbol is indexed in a large number of cells. Then, the neighbouring pattern is applied to a larger number of cells. Fig. 19 shows the parse trees for the sample sentence of Fig. 17 when varying the size of the cells. Decreasing the size of the cell results in a worse recognition rate. In the case of Fig. 19(d) none of the two Captioned-TextBox symbols is detected. For the biggest cell size, see Fig. 19(a), the sentence contains an additional reduced symbol. A Captioned-TextBox symbol is created with the TextBox in the upper part of Fig. 17(b) and the segment in the lower part.

To establish the optimum value of cell size, the best way to proceed is to set it as proportional to the size of the different

### Table 4

<table>
<thead>
<tr>
<th>Property</th>
<th>Parser</th>
<th>Our parsing approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning of the input</td>
<td>Non-sequential</td>
<td>Non-sequential</td>
</tr>
<tr>
<td>Incremental</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-determinism</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis</td>
<td>Bi-directional</td>
<td>Directional</td>
</tr>
<tr>
<td>Input primitives</td>
<td>Basic shapes</td>
<td>Segments and arcs</td>
</tr>
<tr>
<td>Distortion</td>
<td>Thresholds</td>
<td>Distortion value</td>
</tr>
</tbody>
</table>

Fig. 17. Representation of a captioned textbox symbol: (a) example grammar and (b) sample sentence.
drawing elements. This process implies a learning step which is beyond the aim of this work. In this work, we have set the cell size to $48 \times 48$ pixels according to the test data collected from different users. It gives a good ratio between high recognition rate and low rate of false positive reduced symbols.

6.3. Experiment 3: distortions affecting parser

This experiment is designed to show how our parser paradigm is able to cope with inherent distortions appearing on sketches such as overtracing, gaps and spurious strokes. Fig. 20 describes the grammar used in this experiment. We use the GapSegments and the OvertracingSegments productions to define the possible distortions inherent to sketches. The former defines two lines composing a GapSegment symbol if the lines are colinear, i.e., they are close to each other and follow the same direction. The latter describes an OvertracingSegment symbol as two lines that are incident and parallel at the same time. Fig. 21 shows different samples of the grammar presented in Fig. 20. As we can see, the samples contain different levels of distortions. While (a) and (b) only contain one instance of distortion, (c) two and (d) all the segments containing distortions. The corresponding parse trees are presented in Fig. 22. The parser has achieved the symbol QUAD in the four cases. In the parse tree corresponding to Fig. 21(d), see Fig. 22(d), the phenomenon of spurious strokes has been

![Fig. 18. Comparative of memory accesses vs different sizes of cells.](image1)

![Fig. 19. Different parse trees for the instance of Fig. 17(b) with different size of cell values: (a) 96 x 96 pixels, (b) 48 x 48 pixels, (c) 24 x 24 pixels and (d) 12 x 12 pixels.](image2)

![Fig. 20. Adjacency Grammar formalism used in the distortion experiment.](image3)
efficiently tackled by the parser applying the production \textit{GapSegments} between the pairs of lines (Line 3, Line 5), (Line 6, Line 8), (Line 9, Line 11) and (Line 2, Line 12).

6.4. Experiment 4: qualitative and quantitative evaluation

The fourth experiment describes how the parsing algorithm works. To perform the experimental evaluation of the parser we have used two different grammar examples (see Section 5). The first grammar describes Graphical User Interfaces (GUI). This grammar is characterized by having a big intersection set in the productions. A big intersection set determines that a large number of symbols of the grammar are defined in a similar way. The second grammar describes architectural floor plans. This grammar describes the productions in a hierarchical way. It involves that the intersection set among the productions is smaller than the previous one.

The evaluation set consists of eight sketches, three of which belong to the GUI set and the other five are architectural floor plans. Each sketch is acquired at a resolution of 96 DPI. The sketches were drawn by nine people each, without constraining the order in which the graphical elements where drawn, nor the way to draw it. It resulted in a total of 72 samples used to give the recognition performance of the parsing methodology. The population of the experiment consists of six men and three women from different disciplines such as Mathematics or Computer Science. The users were not experts in sketch based interfaces. Fig. 23 shows the different samples that were used for the experiment.

Although an input belongs to a grammar if the initial symbol is reached, to perform a quantitative evaluation of the parsing algorithm we propose a recognition rate metric. This metric compares the number of nodes in the parse tree that appears in the annotated parse tree of the ground truth. This metric can be seen as a recall measure with the equation:

$$r = \frac{\text{Correct NonTerminals}}{\text{Correct NonTerminals} + \text{Missing NonTerminals}}$$  \hspace{1cm} (5)

Fig. 24 plots the average and the standard deviation of the recall metric of Eq. (5) for the experiment. As we can see the average recall is over 80% for each of the sketches and the values of the standard deviation are under 0.16 which implies that the recognition rates for all the users are closed to each other. The average recall metric in each of the grammars is of 89% and 95%.

![Fig. 21. Samples of the grammar of Fig. 20 containing different levels of distortion: (a) gap segments, (b) overtracing segments, (c) overtracing and gap segments and (d) spurious strokes.](image1)

![Fig. 22. Parse trees corresponding to the samples of Fig. 20 respectively.](image2)
respectively. The overall recall is of 93%. Moreover, 63.9% of the sketched instances are completely analyzed by the parser, i.e., the initial symbol of the grammar is reached. Table 5 presents the detailed results obtained in this experiment. With a configuration of $48 \times 48$ pixels in the indexing mechanism and a neighbouring pattern of 8-connectivity the results lead us to describe our methodology as domain and user independent.

The errors produced during the parser analysis could be classified into two categories. First, the drawing errors made by the user. Typical errors in this category are missing primitives and a high distortion level in the strokes or the constraints compounding a symbol. The second category contains the errors produced by the parser algorithm. Here, errors such as not reducing a production or the reduction of a wrong production are found. The former situation refers to symbols that are far away one to the other according to the cell size. The latter refers to the selection of productions sharing the same symbols in its RHS. Looking at the results of Table 5, the errors for the UI1 sample of the users User 1 and User 2 belong to parse errors, in these instances the configuration of the cell size results in a misrecognition of the entire sentence. Another kind of error occurs in the UI3 from User 9, where the user has not included to draw a primitive from the original, this fact means that two reductions are missing.

7. Conclusions

In this work we have presented a syntactic approach to describe and interpret sketched diagrams. Three main features are desirable in a syntactic approach in order to describe them. These features are: to have a high expressive power, to be able to cope with the freedom in the drawing order of the input primitives and to cope with the distortion inherent of sketches. The proposed syntactic approach derives from Adjacency Grammars. Following the taxonomy of Table 1, our methodology is a function based approach based on attributed multiset grammars. Attributed multiset grammars have a descriptive power similar to $n$-dimensional grammars but they do not need an embedding mechanism since they are linear formalisms. The freedom in the ordering of the primitives is also inherent to the way that the productions are defined. We have proposed a distortion model based on attribute synthesized from the symbols in the productions. The attribute is estimated as the variation of a given constraint to its ideal model rather than being a set of thresholds as in the case of Sketch Grammars [10] and LADDER [21] or driven by a primitive extractor as in the case of CDCMG [31].

To interpret a sketched instance of a diagram, we have proposed an incremental parser. This kind of parsers can start

Fig. 23. Samples used for the experimental evaluation: (a) UI1, (b) UI2, (c) UI3, (d) FP1, (e) FP2, (f) FP3, (g) FP4 and (h) FP5.

Fig. 24. Average recall metric and standard deviation for each of the sketches used in experiment 3.

<table>
<thead>
<tr>
<th>User</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI1</td>
<td>0.78</td>
</tr>
<tr>
<td>UI2</td>
<td>0.67</td>
</tr>
<tr>
<td>UI3</td>
<td>1.00</td>
</tr>
<tr>
<td>FP1</td>
<td>1.00</td>
</tr>
<tr>
<td>FP2</td>
<td>1.00</td>
</tr>
<tr>
<td>FP3</td>
<td>1.00</td>
</tr>
<tr>
<td>FP4</td>
<td>1.00</td>
</tr>
<tr>
<td>FP5</td>
<td>0.67</td>
</tr>
</tbody>
</table>

This metric is estimated by the number of symbols reduced by our parser that appear in the model parse tree.
the analysis of an input before it is finished. This makes them suitable to work in on-line frameworks. To choose the production of the grammar to be reduced, our parser methodology uses a Breadth First Search. This technique entails less time complexity than Depth First Search in terms of not redoing the parse tree when a production is not valid. Moreover, when analyzing a 2D sentence a parser cannot assume that the symbol being analyzed produces a valid reduction with the previous one. This results in the addition of an indexing structure in the proposed parser. This structure is based on a spatial tessellation. Each time a new primitive is analyzed the parser indexes it into the grid structure and find candidates to produce a valid reduction in terms of a neighbouring pattern. Some decisions have been taken in terms of the way the symbols are indexed and the size of the cells of the indexing structure. Two indexing methods were proposed and the size of the cells has been experimentally set up according to the size of the drawn.

Two different application scenarios have been developed to evaluate our syntactic approach. One approach is related to the design of User Interfaces and the other is devoted to the design of sketched architectural floor-plans. Although a preliminary evaluation of the methodology has been done. Experiments have shown the suitability of our method to describe and recognize sketched diagrams. Moreover, the use of our approach is successful in on-line applications due to the reduction of time given by the neighbouring pattern. In addition, the method is able to cope with diagrams containing user variability in the drawing order as well as variability in the number of graphical entities present in each diagram. Although the syntactic approach presented in this article is able to cope with distortion, it is not able to cope with errors produced by the misrecognition of a segment by an arc or vice versa. The addition of an error recovery method increases the performance of our methodology. Furthermore, changing the distortion model to a statistical model will increase the performance of our syntactic approach. Nevertheless, a learning step would be necessary. When searching for a production, the use of fixed cell size may introduce the appearance of errors. This is because the symbols forming that production are sufficiently separated, so they are not considered neighbours by the parser. In order to establish an optimum value of cell size it would be necessary to collect several instances of the different symbols and study an average of the size to setup the most adequate value. In addition, the study of space division techniques to create the grid thus avoiding the use of a predefined value of cell size, could help to increase the performance of the parser.

Acknowledgements
This work has been partially supported by the Spanish projects TIN2008-04998, TIN2009-14633-C03-03 and CONSOLIDER-INGENIO 2010 (CSD2007-00018). Joaquim Jorge was partially supported by the Portuguese Foundation for Science and Technology grant PTDC/EIAEIA/104031/2008.

References
Joan Mas received the degree in Computer Science in 2003 from the Universitat Autonoma de Barcelona. He joined the computer vision center in 2003. Currently he is an Assistant Professor of the Universitat Autonoma de Barcelona and he is doing his PhD at the Computer Vision Center under the supervision of Gemma Sanchez and Josep Lladós. His current research fields are document analysis, graphics recognition, syntactical and structural pattern recognition and sketch recognition. He has participated in different R+D projects and has published in different international conferences.

Josep Lladós received the degree in Computer Sciences in 1991 from the Universitat Politècnica de Catalunya and the PhD degree in Computer Sciences in 1997 from the Universitat Autònoma de Barcelona (Spain) and the Université Paris R (France). Currently he is an Associate Professor at the Computer Sciences Department of the Universitat Autònoma de Barcelona and a staff researcher of the Computer Vision Center, where he is also the director. He is the head of the Pattern Recognition and Document Analysis Group (2005SGR-00472). His current research fields are document analysis, graphics recognition and structural and syntactic pattern recognition. He has been the head of a number of Computer Vision R+D projects and published several papers in national and international conferences and journals. J. Lladós is an active member of the Image Analysis and Pattern Recognition Spanish Association (AERFAI), a member society of the IAPR. He is currently the chairman of the IAPR-ILC (Industrial Liaison Committee). Formerly he served as chairman of the IAPR TC-10, the Technical Committee on Graphics Recognition, and also he is a member of the IAPR TC-11 (reading Systems) and IAPR TC-15 (Graph based Representations). He serves on the Editorial Board of the ELCVIA (Electronic Letters on Computer Vision and Image Analysis) and the JDAR (International Journal in Document Analysis and Recognition), and also a PC member of a number of international conferences. He was the recipient of the IAPR-ICDAR Young Investigator Award in 2007. Josep Lladós has also experience in technological transfer and in 2002 he created the company ICAR Vision Systems, a spin-off of the Computer Vision Center working on Document Image Analysis, after win the entrepreneurs award from the Catalonina Government on business projects on Information Society Technologies in 2000.

Gemma Sánchez received the degree in Computer Sciences in 1994 from the Universitat Politècnica de Catalunya and the PhD degree in Computer Sciences in 2001 from the Universitat Autònoma de Barcelona (Spain) and the Université Henry Poincaré (France). Currently she is a lecturer at the Computer Sciences Department of the Universitat Autònoma de Barcelona and a staff researcher of the Computer Vision Center where she has participated in a number of industrial and research projects. Her current research interests are document analysis, graphics recognition and structural and syntactic pattern recognition. She is now working in the domain of visual languages for graphics representation and recognition and document image analysis. She has published several papers in national and international conferences and journals. She is a member of the IAPR TC-10 (Technical Committee on Graphics Recognition). She serves as editorial assistant in the ELCVIA (Electronic Letters on Computer Vision and Image Analysis).

Joaquim Jorge is Full Professor of Computer Graphics at Instituto Superior Técnico (IST/UTL) Lisboa, Portugal. He received PhD and MSc degrees in Computer Science from Rensselaer Polytechnic Institute, Troy, NY, in 1995 and a BEE from IST/UTL. He was co-chair of EUROGRAPHICS’98, which took place in Lisboa, Portugal. Recently he co-chaired the EG SBM’05 (Dublin, Ireland), SBM’04 (Grenoble, France, September) and EGMM’04 (Nanjing, PRC, Oct 04) Workshops. A long time practitioner of Computer Graphics, he first joined the Eurographics Association in 1986, helped organize the first Portuguese Computer Graphics Meeting in 1988 and was a founding member of the Eurographics Portuguese Chapter. He has served on the EG Promotions Board and on the Publications Board from 1997 to 2000, leads the EG Working Group on Sketch-Based Interfaces and Modeling, was EG Conference Monitoring Officer for EG2001 and served as elected member of the Associations’ Executive Committee from 2000-2006. He was invited as proposal evaluator for the ITR program of the National Science Foundation in 2000, 2001 and 2003 and EU’s IST Fifth framework program, EUREKA and related consultation meetings on different occasions, the latest in 2007. He has been a member of over 12 research projects both at the National and European Level. He was Principal Investigator (PI) of the SmartSketches (IST-2000-28169) project, and led the Alfa INETGAM II-0072-A network of excellence, both started in 2001. He also leads the CEGIS international project, co-funded by the EC Eurographics and SIGGRAPH Associations since September 2002. He serves or has served on the program committees of over 120 international conferences. He is Editor-in-Chief of Computers and Graphics Journal (Elsevier) served on the board of several journals including Computer Graphics Forum, International Journal of Virtual Reality, ITSE and IJMBL journals. He is affiliated with ACM/SIGGRAPH since 1989 and a member of the Educational Comittee since 2004, IEEE Computer Society (Senior Member since 2000), IFIP TC13 (Human Computer Interaction—National Contact Person), IAPR and EACE, the European Association of Cognitive Ergonomics. Joaquim Jorge’s interests are in Calligraphic and Multimodal User Interfaces, Visual Languages and Pattern Recognition techniques applied to Human–Computer Interaction. He is author or co-author of over 153 communications published in peer-reviewed international conferences, journals and books.