Automatic Learning of Symbol Descriptions Avoiding Topological Ambiguities

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

Automatic Learning of Symbol Descriptions Avoiding Topological Ambiguities*

J. Mas [1] ,B. Lamiroy[2], G. Sanchez[1] and J. Llados[1]
[1]Computer Vision Center,Computer Science Dept. UAB, Spain
[2]INPL-LORIA,Ecole des Mines, Nancy CEDEX, France

July 27, 2006

Abstract

In this paper we address both automatic recognition of sketched symbols and the construction of the corresponding models from user drawn examples. Our approach is based on a two stage process. In a first phase we use an Adjacency Grammar to express topological properties of the symbol. In order to be able to further disambiguate topologically similar configurations on the rules of the grammar that are triggered by the recognition process produce a set of local geometric invariants is defined. The combination of both steps results in an efficient recognition method for user drawn sketches. Furthermore, we show that the same approach can easily be adapted for the generation of Adjacency Grammars from user provided and hand drawn examples.

1 Introduction

Shape description is one of the important steps in symbol recognition. Depending on the primitives used to represent the shape, we may distinguish between two major categories on shape recognition, Contour-Based and Region-Based. The former bases the description on the contour or the edges of the image, while the latter is based on the whole image or on closed regions of the image. Inside these two categories we may distinguish between two methodologies: Structural and Global descriptors. Global descriptors calculate global features on the images. Inside this category we may find basic descriptors as: Area, Perimeter, Compactness, etc. and some other descriptors are based on moments and signal processing approaches like Zernike, Legendre, Fourier, *etc.* or based on grids as in Zoning.

On the contrary, Structural methods use features based on attributed primitives and the relations among them. As Structural methods we may find several

*This paper was presented at the 3rd Eurographics Workshop on Sketch-Based Interfaces and Modeling (Sep. 3-4, 2006) in Vienna. The definitive version is available at digilib.eg.org.

different approaches as string-based methods, chain codes, polygonal approximation. Within this methodology we may further distinguish (among others) Syntactic methods. These methods are based on the representation of shapes applying techniques of formal language definition. In this paper, we focus more precisely on this last category of shape description. A review on shape description techniques is presented in [ZL04].

Although this techniques are widely used and deeply studied, one of the main problems and recurring difficulties of shape description is to cope with ambiguity. There always exist situations in which symbols cannot be disambiguated without addition of extra knowledge. This is most often due to the fact that all approaches need to be robust to noise and distortions on one hand, and need to be sufficiently discriminant on the other.

Some works presented in the literature try to describe sketches using an structural approach. Veselova in [VD04], captures the relevant characteristics describing a shape and formulate three heuristics to apply to this characteristics based on human perception. Using them they extract the constraints to create the final shape description based on a global threshold. Mankoff in [MHA00] presents a shape descriptor based on sketched GUI. It solve the problem of ambiguity by asking the user to chose among a set of possible models.

The work presented in this paper is based on a structural contour based descriptor, and more precisely on a syntactic approach defined by an Adjacency Grammar. An Adjacency grammar describes a symbol as a set of primitives and the relations among them. The description of each model is obtained by a learning process based on a set sketched instances. This allows us to infer the constraints forming the shape based on an adaptative learning instead of based on a global threshold as Veselova in [VD04].

The method therefore is able to cope with high distortion representations of the reality. Working with sketches introduces distortions on the relations among the primitives and also generates the difficulty of having to handle different chronological drawing of the strokes composing a symbol. Further more, as we shall further show in this paper, there are some intrinsic ambiguity problems related to the description method itself.

The purpose of the work is to improve the method presented in [MLSL06] with the capability to disambiguate between shapes. This method solve the ambiguity problem computing some invariants related to the constraints defining the shape, instead of asking the user as in the method presented by Mankoff in [MHA00]. On this paper we define two ambiguous shapes as topological identical. We consider two shapes being topological identical if they can be expressed with the same constraint set. Trying to disambiguate among shapes we we create a specific vector of invariants associated to any relation. This description contrary to the work presented by Veselova in [VD04], is rotated and scaled invariant.

The paper is organized as follows: In the next section we present the sketch based interface that has been used as environment for our experiments. Section 3 refers to the basis of the methodology used to learn the description of a symbol. Section 4 presents the method used to cope with the ambiguities,

section 5 presents the experimental results and finally conclusions are presented on section 6.

2 Framework

The experiments described in this paper are developed under a sketch based environment. The application is named PVPC (Virtual Prototyping of Projects under Construction) and it is a sketch environment to design architectural floor plans. The user interacts with the system by means of drawings on a Wacom Tablet or a Tablet PC or using a digital pen & paper protocol (*e.g.* as the Digital IO Pen from Logitech [Log04]).

The application tries to recognize the different structural, furniture and services symbols that appears on it. More details on the application itself can be found in [SVL+04]. The method described in this paper is based on grammatical rules, expressed with respect to image primitives. However, before being able to manipulate the primitives that form a drawing, we need to extract them from the strokes of the user. Furthermore, when drawing, one single stroke may represent more than one primitive. We therefore pre-process the strokes with polygonal approximation as the presented in [TASD+00]. This approximation divides them by using the high curvature points and also checks if the primitive is an arc or a segment. In this paper we restrict ourselves to segments although the method may be readily extended to arcs as well.

The method presented in this paper may be easily integrated in applications allowing users to define their own set of symbols to recognize. This may be a valuable add-on for applications in the architectural world, since there exists no standard in this area. The environment is also suitable for other kinds of applications like design of electrical circuits, physical blueprints, *etc.*

3 Adjacency Grammars

Adjacency grammars allow to describe 2D-shapes on a linear way, describing it in terms of a set of primitives and the relations among these primitives. Adjacency grammars were first introduced in [JG95]. The reader can also refer to [MSL05] for further details on the use of Adjacency Grammars for sketched symbol recognition.

3.1 Symbol Description Model

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

The main idea is to segment the sketch into primitives (*e.g.* line segments and curved arcs). For each pair of primitives (A, B), a number of constraints (as depicted in Figure 1) are evaluated using a normalized associated uncertainty degree $\delta = [0 \dots 1]$ which measures the degree of distortion with regard to an ideal shape. Values close to 0 indicate that the constraint is satisfied, while values close to 1 mean that the constraint makes no sense. We currently associate the following constraints and functions:

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

This may then be used to describe symbols within the Adjacency Grammar. For instance, a rectangular triangle, made out of three line segments A, B and C, is described as follows:

$$Triangle(A, B, C) \begin{cases} Adjacent(A, B) \\ Adjacent(B, C) \\ Adjacent(A, C) \\ Perpendicular(A, C) \end{cases}$$

3.2 Automatical Model Generation

This approach is not only well suited for description and recognition, it can also be easily adapted to automatically infer and construct the models from sample sketches provided by the user.

Simply taking into account the constraints with the lowest associated cost may be sufficient to describe a symbol from a given sketch. However, notice that we work with sketched instances of symbols, which are rough expressions of the reality, containing a high degree of distortion with respect to an ideal model. Therefore, we need more than one single instance to correctly and automatically infer the ruleset of our adjacency grammar that describes the symbol. Not doing so would invariably lead to over- or underconstrained rulesets.

To correctly infer a sketched symbol we need several instances to cope with the possible distortions that may appear. To reach this description we build on the method presented in [MLSL06]. This method is divided into three steps:

- Ruleset Generation
- Ruleset Normalization and Primitives Alignment
- Factorization

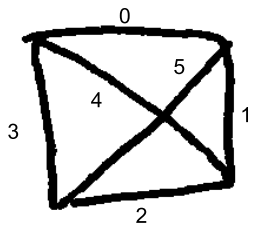

Figure 2: Instance sample.

The first step tries to keep those constraints which are sufficiently realistic sorting them by value. Fitting a normal distribution over the data allows to easily find the most appropriate and statistically salient features that compose the model.

The second step considers that drawing two instances of the same symbol not necessarily results in an identical numbering of primitives and that the resulting ruleset may contain different, but geometrically equivalent rules. This requires an explicit primitive alignment such that we obtain a correspondence among the primitives of different instances.

The last step is dedicated to levelling out differences between generated rulesets from different examples, occurring from drawing deformations. It constructs the final constraint set based on a majority voting scheme between all the instances of a model.

Table 1 represents the constraint set that was automatically inferred from the instance presented on Figure 2. Note that perpendicularity was deduced for primitives 4 and 5 (the crossing diagonals) for this particular instance, although the lines themselves aren't really perpendicular. Further samples may contribute to enforcing or rejecting this rule.

This method works well for several kinds of symbols. But, as we shall show in the next section, there exist some configurations where the generated rulesets are not precise enough to distinguish between configurations that are geometrically different, but share the same topology. Two of those configurations are shown in Figures 3 and 4. The next section presents a way to cope with this kind of ambiguities.

Figure 3: A Plug, an Arrow and a Triangle, all three sharing the same ruleset

Adjacent(primitive0, primitive5)
Adjacent(primitive0, primitive4)
Adjacent(primitive0, primitive3)
Adjacent(primitive0, primitive1)
Adjacent(primitive1, primitive2)
Adjacent(primitive1, primitive4)
Adjacent(primitive1, primitive5)
Adjacent(primitive2, primitive3)
Adjacent(primitive2, primitive5)
Adjacent(primitive2, primitive4)
Adjacent(primitive3, primitive4)
Adjacent(primitive3, primitive5)
Intersects(primitive4, primitive5)
Parallelism(primitive0, primitive2)
Parallelism(primitive1, primitive3)
Perpendicular(primitive0, primitive3)
Perpendicular(primitive0, primitive1)
Perpendicular(primitive1, primitive2)
Perpendicular(primitive2, primitive3)
Perpendicular(primitive4, primitive5)

Table 1: Inferred Constraint Set for the instance of Figure 2

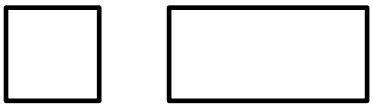

Figure 4: A square and a rectangle: topologically identical but geometrically different.

4 Overcoming Topological Ambiguities

The previously described method fails to distinguish between topological similar configurations, simply because of the fact that the rules only embed very poor geometrical information (mainly parallelism and perpendicularity). On the other hand, these very loose geometric constraints make the method very well suited for recognizing and capturing hand written distortions. The main challenge therefore is to correctly distinguish between sufficiently different configurations, whilst maintaining robustness to deformations that are proper to sketch based interfaces. We use geometric invariants [GBB98] to solve this dilemma.

4.1 Geometric Invariants

Two forms are considered topologically equivalent if there exists a continuous deformation (possibly non-rigid) that projects the first in the second. Invariants such as connexity, incidence and holes are the only properties that are preserved in this case. Since purely topological descriptions are too poor to account for more or less subtle differences between forms (*e.g.* a square and a circle are topologically equivalent) it is imperative to embed more geometrical information into the symbol descriptions. Two forms are considered geometrically or rigidly equivalent, if there exists a rigid transform projecting the first into the second. The advantage of geometric transforms (translation, rotation, similarity, affinity, projective, ...) is that they very neatly enter in a completely computationally controlled mathematical framework. On the other hand, their great drawback is that they do not capture all deformations that occur in image analysis, and more particularly in sketched based environments.

Rather than searching for full rigid equivalence between forms, we propose to only use geometrical invariants on very local configurations. Furthermore, we restrict them to similarity transform invariant values (*i.e.* scale, rotation and translation invariant). Skew is rather considered as an artefact related to the hand drawn distortions.

4.2 Associating Invariants with Grammar Rules

We proceed by associating a vector of invariant measures to each rule in the ruleset of a symbol, based on either of the following:

- the length ratio between the primitives triggering the rule,
- the angle between the primitives,
- the relative normalized distance between the primitives.

Figure 5: A T and a L Shapes.

This classification lead us to define a set of specific invariants for each rule to cope with different configurations. For instance, as we see in Figure 5 the represented shapes differ by the position of their incidence point of one of the segments to the other, while the constraint set obtained from these two samples are the exposed on table 2.

It is noteworthy to mention here that we slightly differ from the approach in [MLSL06] and which we presented in the introduction, in the sense that we do not use the rule of *Adjacency*. Adjacency is just seen as a particular case of incidence. Furthermore, not all invariants are computed for all kinds of rules. Some make more sense than others, and in some cases, the confidence measure of a rule already uses one of the mentioned invariants.

Incident(P1,P2)	Incident(P1,P2)
Perpendicular(P1,P2)	Perpendicular(P1,P2)

Table 2: Constraint Set corresponding the samples on Figure 5.

4.3 Length Ratio

The length ratio invariant is defined as the difference of length between the primitives forming the rule. Calculation is based on the following equation:

$$ratio = \min\left(\frac{length(P1)}{length(P2)}, \frac{length(P2)}{length(P1)}\right) \quad (1)$$

The obtained value is guaranteed to be 0 and 1. This invariant is associated with all rules, and allows to distinguish between topological equivalent shapes as in Figure 4 : the square has a ratio near 1 among all its primitives ; on contrary, the rectangle has ratio near to 1 among the primitives that are parallel one to another, and it is different than 1 among the perpendicular ones.

4.4 Angle

The angle invariant only associated with the intersection and incidence rules. Since it is defined as the minimum angle between the primitives forming the rule, it is fairly useless to use it in the case of parallelism or perpendicularity. This angle is calculated taking one of the segments as reference and based on the cosines theorem such that the angle is always less or equal to π. This invariant allows to distinguish between topological equivalent shapes as in Figure 3.

4.5 Normalized Distances

According to the considered rule, two kinds of distances are measured. In the case of *Incidence*, we compute the relative position of the virtual incidence point with respect to the extreme of the segment, as shown in Figure 6. The relative distance of the two primitives is already accounted for in the uncertainty degree that is associated with the rule itself.

On the contrary, in case of a *Parallelism* rule, the previous measurement makes no sense, since there is no virtual incidence point between both primitives (or at best it's a badly conditioned one, prone to noise). Figure 6 shows how the distances are calculated in this case : DX represents distance between the two lines supporting the segments, and DY represents the relative distance between their respective midpoints, measured following the direction of the segments.

All computed distances are normalized with respect to the length of the longest primitive, in order to remain invariant to scale.

Two special cases for this invariant are described in Figure 7(a) and (b). The particularity of Figure 7(a) is that there is a collinearity between the two segments. In this case the DX distance will be near 0. On the contrary, in

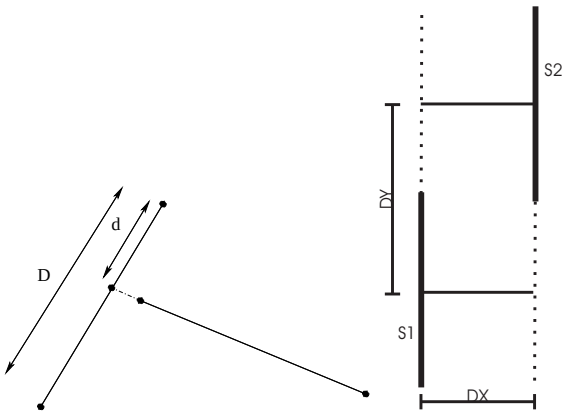

Figure 6: Distances calculated between incident segments (left) and parallel segments (right)

Figure 7(b) the two segments are parallel and are aligned by their midpoints, the DY distance will be near 0 for this case.

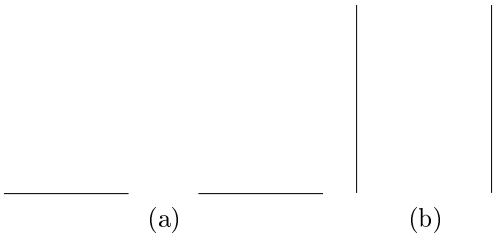

Figure 7: Particularities on axis distances.

4.6 Rules and Invariant Vectors

For any inferred constraint we can now define a specific vector with some of the previously mentioned invariants :

- the *Incidence* relation is associated with ratio, angle and the relative distance of the incidence point with respect of the extreme of the intersected segment.
- the *Parallelism* relation is defined by the ratio and the distances used for parallelism : DX and DY.
- *Perpendicular* is associated with the ratio and the distance.
- *Intersects* is defined by the angle between the primitives that form the rule.

Constraint Plug	Ratio	Angle	Distance
Incident(0,1)	0.776	1.097	$1.229 * 10^{-13}$
Incident(1,2)	0.266	2.616	$3.714 * 10^{-14}$
Incident(0,2)	0.343	2.571	$4.771 * 10^{-13}$

Constraint Arrow	Ratio	Angle	Distance
Incident(0,1)	0.844	0.961	$2.942 * 10^{-15}$
Incident(1,2)	0.348	0.414	0.016
Incident(0,2)	0.412	0.546	0.017

Constraint Triangle	Ratio	Angle	Distance
Incident(0,1)	0.942	1.071	0.008
Incident(1,2)	0.939	1.010	0.036
Incident(0,2)	0.931	1.061	0.020

Table 3: Constraints Sets corresponding to shapes of Fig. 3

The constraint set presented in table 3 shows the constraints with the corresponding parameters on *ratio*, *angle* and *distance*. We may observe as we explained before that in this case, the angle parameter, allows to distinguish between the topological equivalent samples.

4.7 Model Generation by Learning Invariants

Once we have defined how to avoid ambiguities between symbols which are topologically identical, we need to define a method or a way to learn the invariants from user provided samples.

Symbols are inferred from several instances, and the variability of their associated invariants is simply qualified by their average value and standard deviation. Average and standard deviation are computed on *ad hoc* models. From the experience we notice that the the angle invariant follows a normal distribution and that the ratio and the distance follow a χ^2-law.

4.8 Sketch Recognition

Once we have trained the system with the instances of the different symbols we want to recognize, we need a recognition process that, given an unidentified input, tells us what symbol that we have learnt is more similar. The process works as follows: given an input we calculate the value of the cumulated uncertainty degree, obtained by parsing the rules that are associated to our models. And we sort the resulting models by this value. Not all the rules are evaluated since we evaluate those rules that have the same number of primitives as the input. We then cross-verify the validity of the invariants in order to disambiguate among the rules that give very similar results. We select the model that minimizes

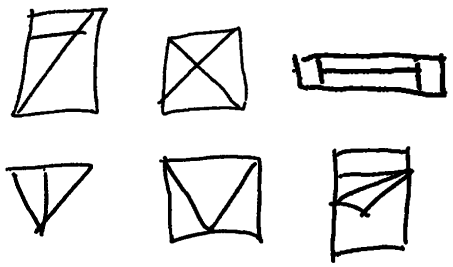

Figure 8: User Drawn Samples.

both the cumulated uncertainty value of the rules and respects the invariants constraints.

5 Experiments

Experiments show how our approach improves the method presented in [MLSL06] and they show how it works when trying to describe two symbols with the same topology but being geometrically different. We also show that it is able to differentiate between symbols that are made up out of disconnected parts.

As said in section 1, we consider that two symbols are topologically identical if they may be expressed with the same constraint set. *I.e.*, the shapes in Figure 3 are topologically identical and may be described with the constraint set presented on table 3. Figure 9 shows some other symbols that are difficult to differentiate due to this fact.

Figure 9: Some samples of our experiment set

For the first experiment we consider the three shapes of Figure 3 with 10 hand drawn instances for each shape. From the instances of the symbols we have calculated the expected value of the distribution and standard deviation for each invariant between the pairs of primitives. These expected and standard deviation values are showed on Figure 11 for each pair of primitives of the obtained ruleset (as shown also in Table 3). The big values on some of the standard deviations show that the user provided instances contained a high distortion level. Figure 10 shows some of the samples.

The information to take into account is the one contributed by the invariants. If we look to the *ratio* invariant we may see that the shape representing the

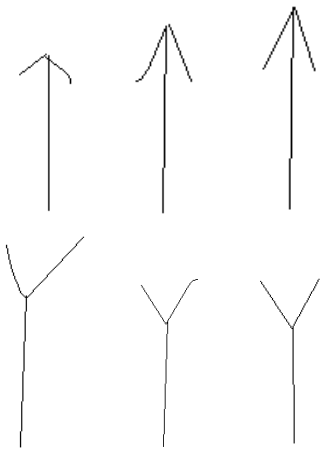

Figure 10: Distorted samples on the experiment image set.

Triangle is easily discriminated from the other two by using this value. The ratio is near 1 on all the pair of primitives, while the Arrow and the Plug have two primitives that have the same length but this length is approximatively 1/3 of the length of the other primitive.

On the other hand, if we look to the angle invariant we may see that it is the most discriminating invariant of the three shapes. The shape representing meanwhile the Arrow have the angles among the primitives that have not the same length lesser than 90 degrees the Plug has the angles greater than 90 degrees. Referring to the triangle the three angles are approximatively the same. We may conclude that the angle invariant disambiguates among the shapes of Figure 3. The shapes with the corresponding primitive numbering are showed on fig. 12

The second experiment tries to enforce the use of the invariant ratio. In order to enforce this invariant we have made the system to learn 2 shapes with 10 instances any representing and square and a rectangle.

The values obtained from the inference of the grammar are presented on Figure 13. As we may see we have presented the ratio and angle for any of the shapes. The values represented are the expected value of the distribution and standard deviation. Looking at the angle invariant we may observe that on both shapes are approximatively the same. We may observe two parallelism and four perpendiculars.

On contrary if we observe results on ratio invariant, we may distinguish between the two shapes in terms that the values on the shape representing the square, denoted as QUAD, are approximatively near 1 and the values on the other shape are near 0.4 between segments that are perpendicular and near 1 on segments that are parallel. Also if we take into account the ranks defined by the standard deviation we may seen that there not exists overlapping between the values accepted on by the ranks. This fact, take us to consider that the

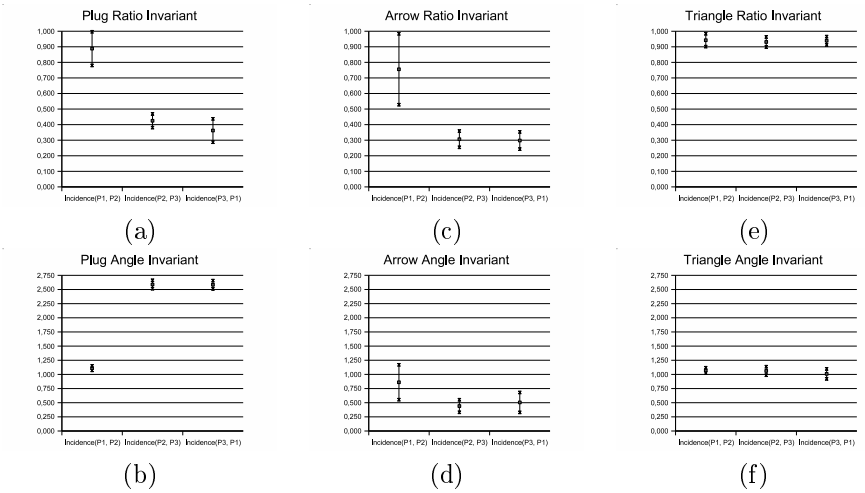

Figure 11: Comparative among the ratio and the angle invariant for shapes: Plug, Arrow and Triangle.

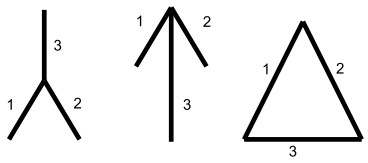

Figure 12: Numbering of primitives for shapes: Plug, Arrow and Triangle.

method will disambiguate between the two shapes.

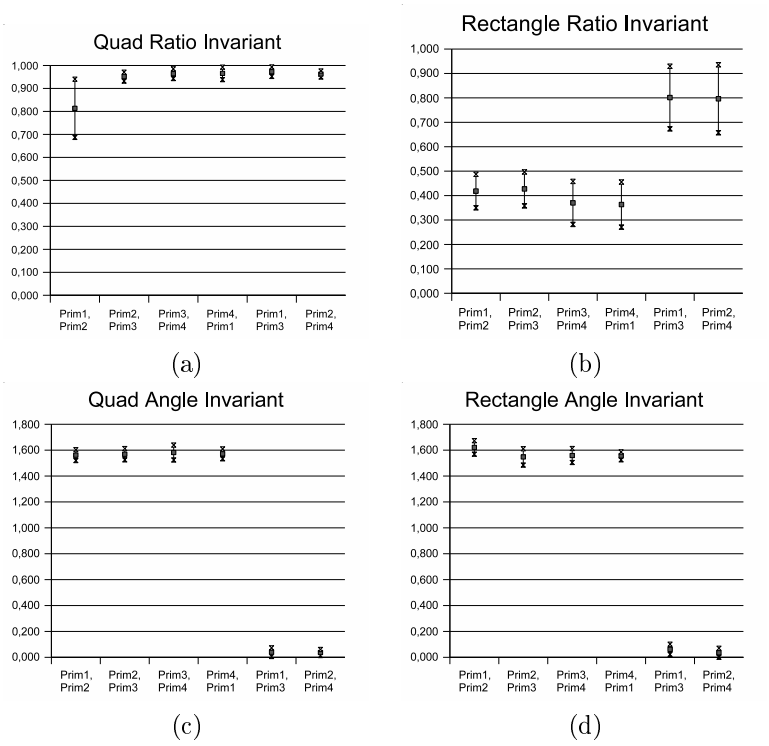

Figure 13: Comparative among the ratio and the angle invariant for shapes: Quad and Rectangle.

Finally, the last experiment allows us to show if the method presented is able to distinguish between two shapes with disconnected parts as are the shapes of fig. 14. On Table 4 we may seen how the system will disambiguate between the two shapes taking into account the invariant distance.

If we look to the values on the ratio and angle invariant we may see that they are similar for the two shapes. The distance invariant on parallelism constraint calculated as we explained before let us to disambiguate between the two shapes. If we look the values for *Parallelism* constraint we observe that for *Parallelism(P1,P2)* the values are 0.129 ± 0.083 and 3.607 ± 0.643 on table. 4(a), on contrary for the same constraint on table. 4(b) the values are 1.205 ± 0.264 and 3.809 ± 0.688. There exists a big difference between the two values what makes the system to reach its aim. The numbering of primitives follows the same configuration of fig. 14.

Constraint	Ratio	Angle	Distance1	Distance2
Incident(P0,P1)	0.813 ± 0.116	1.584 ± 0.045	$1.63 * 10^{-14} \pm 2.9 * 10^{-15}$	-
Incident(P2,P3)	0.905 ± 0.067	1.590 ± 0.079	$1.47 * 10^{-14} \pm 2.26 * 10^{-15}$	-
Parallelism(P1,P2)	0.887 ± 0.074	3.084 ± 0.050	0.129 ± 0.083	3.607 ± 0.643
Parallelism(P0,P3)	0.810 ± 0.106	0.030 ± 0.028	4.516 ± 0.545	0.186 ± 0.173

(a)

Constraint	Ratio	Angle	Distance1	Distance2
Incident(P0,P1)	0.920 ± 0.051	1.611 ± 0.067	0.0	-
Incident(P2,P3)	0.920 ± 0.041	1.604 ± 0.046	$4.98 * 10^{-15} \pm 1.47 * 10^{-15}$	-
Parallelism(P1,P2)	0.891 ± 0.066	3.090 ± 0.040	1.205 ± 0.264	3.809 ± 0.688
Parallelism(P0,P3)	0.838 ± 0.098	0.041 ± 0.028	4.705 ± 0.567	0.132 ± 0.081

(b)

Table 4: Results related to the shapes of fig. 14 (a) Corresponds to fig. 14(a) and fig. 14 (b)

(a) (b)

Figure 14: Special case: (a) Corners with collinear segments and (b) Corners without collinearity among segments.

6 Conclusions

The aim of the method presented in this paper is to improve the description method presented in [MLSL06], allowing the differentiation among symbols which have the same topology although they are geometrically different. In order to achieve this, we have considered a set of invariants, based on the observation of the different confusions that we have observed. The Invariants that has been chosen are: *Ratio*, *Angle* and *Relative Distances*.

Results obtained from the conducted experiments show that the method is able to distinguish among topological identical but geometrically different symbols. Even when symbols contain disconnected parts the proposed method is able to distinguish them. This is the case for the symbols presented on Figure 14. The two corners in Figure 14(a) share collinear segments while in Figure 14(b) this collinearity does not exist.

We may observe that the description method based on two steps: a first step consisting in an *Automatic construction of a Constraint Set* [MLSL06], followed by a *Parametrization of the inferred constraints based on invariants* allows to describe forms, and construct models, avoiding the user to specify in a formal way the set of symbols she wants to use.

The use of additional attributes like *ratio*, *angle* and *distance* invariants also improves the alignment of primitives. This information helps to reduce the

intrinsic complexity of the method.

Furthermore, the work is integrated in a sketch based framework, allowing the user to define the set of symbols it wants to use. These symbols may have or have not a functional background, *i.e.* the user can define a set of symbols that allows to interact with the framework by means of selecting, moving, rotating *etc.*

Future work will be related to define an on the fly recognition method that is able to recognize symbols while drawing (and possibly before completion). It will take into account the constraints forming the symbol and the invariants defined between any pair of primitives.

Acknowledgements

This work has been partially supported by the Spanish project CICYT TIC2003-09291.

References

[GBB98] P. Gros, O. Bournez, and E. Boyer. Using local planar geometric invariants to match and model images of line segments. *Computer Vision and Image understanding*, 69(2):135–155, 1998.

[JG95] J.A.P. Jorge and E.P. Glinert. Online parsing of visual languages using adjacency grammars. In *Proceedings of the 11th International IEEE Symposium on Visual Languages*, pages 250–257, 1995.

[Log04] Logitech. IO digital pen, 2004. www.logitech.com.

[MHA00] Jennifer Mankoff, Scott E. Hudson, and Gregory D. Abowd. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In *CHI*, pages 368–375, 2000.

[MLSL06] J. Mas, B. Lamiroy, G. Sanchez, and J. Llados. Automatic adjacency grammar generator from user drawn sketches. In *Proceedings of 18th International Conference on Pattern Recognition*, august 2006. Hong-Kong.

[MSL05] J. Mas, G. Sanchez, and J. Llados. An adjacency grammar to recognize symbols and gestures in a digital pen framework. In *Proceedings of Second IBPRIA*, pages 115–122, June 2005. Springer, Berlin.

[SVL+04] G. Sánchez, E. Valveny, J. Lladós, J. Mas, and N. Lozano. A platform to extract knowledge from graphic documents. application to an architectural sketch understanding scenario. In S. Marinai and A. Dengel, editors, *Document Analysis Systems VI*, pages 349–365. World Scientific, 2004.

[TASD+00] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini, and S. Tabbone. Stable and robust vectorization: How to make the right choices. In A.K. Chhabra and D. Dori, editors, *Graphics Recognition: Recent Advances*, pages 3–18. Springer-Verlag, Berlin, 2000. Vol. 1941 of LNCS.

[VD04] Olya Veselova and Randall Davis. Perceptually based learning of shape descriptions for sketch recognition. In *AAAI*, pages 482–487, 2004.

[ZL04] Dengsheng Zhang and Guojun Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, January 2004.