On the Use of Geometric Matching for Both: Isolated Symbol Recognition and Symbol Spotting

Nibal Nayef and Thomas M. Breuel

Technical University Kaiserslautern, Germany
nnayef@iupr.com, tmb@informatik.uni-kl.de

Abstract. Symbol recognition is important in many applications such as the automated interpretation of line drawings and retrieval-by-content search engines. This paper presents the use of geometric matching for symbol recognition under similarity transformations. We incorporate this matching approach in a complete symbol recognition/spotting system, which consists of denoising, symbol representation and recognition. The proposed system works for both isolated recognition and spotting symbols in context. For denoising, we use an adaptive preprocessing algorithm. For symbol representation, pixels and/or vectorial primitives can be used, then the recognition is done via geometric matching. When applied on the datasets of GREC'05 and GREC'11 symbol recognition contests, the system has performed significantly better than other statistical or structural methods.

Keywords: Symbol Recognition, Symbol Spotting, Geometric Matching.

1 Introduction

Graphics recognition is a challenging problem in the field of document image analysis. It deals with graphical entities that appear in line drawings such as architectural and electrical symbols. The localization and recognition of such symbols are important for retrieval applications among others. Much valuable work has been done on the recognition of isolated symbols, and the number of methods for recognizing symbols in context is increasing. However, the solutions proposed so far have not reached to the point where they can be reliably used in real world applications like automatic analysis of technical drawings or query based retrieval in a digital library.

In this paper, we present the usage of geometric matching techniques for recognizing symbols both isolated and in context. The system starts by applying an adaptive preprocessing algorithm inspired by the work in [14] for denoising. After that, for symbol or line drawing representation, either pixels or vectorial primitives can be used as features. For matching a pair of images – a query with a database image –, the geometric matching framework of [1] is used.

The authors have used similar techniques in a previous preliminary work [7] for symbol spotting. In this paper, we largely extend our previous work as follows.

– The geometric matching framework is generalized to deal with vectorial primitives such as lines and arcs, not only pixels. This gives the flexibility to use either a statistical or a structural representation for symbols. It also speeds up the matching since the number of features as lines and arcs is much smaller than in the pixel representation.
– The geometric matching framework is improved to deal with very similar shapes. This is done by assigning penalties to the non-matched features, which helps – for example – in recognizing symbols whose shapes are subsets of other symbols' shapes.
– An adaptive preprocessing module is added to the system to deal with different noise types including clean images.
– Large-scale experiments and benchmarks have been carried out for both isolated symbol recognition and symbol spotting.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 describes the proposed recognition system. Section 4 presents the performance evaluation, and Section 5 concludes the paper.

2 Related Work

Isolated symbol recognition approaches can be classified into structural [2], [5], [15], statistical [6], [12], [13], [16] and hybrid [3], [11] approaches. We review here the ones that use geometric techniques, whether for feature extraction or for matching and recognition.

Coustaty et. al. [2] used an adapted Hough transform to extract the segments of the symbol, and arrange them in a topological graph as a structural signature of the symbols, and then, Galois Lattice classifier is used for recognition. Min et. al. [6] presented a pixel distribution based method, and a similarity measure called bipartite transformation distance. In their system, the symbols are aligned by their angular distributions to achieve rotation invariance. They also presented a denoising algorithm to deal with different noise types before applying the recognition system. Wong et. al. [12] computed descriptors of symbols from a modified Hough transform, which makes them invariant to scale and rotation, after that, those descriptors were compared to find matching symbols.

In general, the methods in the structural approach rely on vectorial signatures of the symbols, so the preprocessing and primitives extraction greatly affect the matching results. The statistical approaches are more popular, but they are sensitive to noise. More importantly, most of the methods from both approaches are designed to recognize segmented (isolated) symbols, they cannot recognize symbols in context.

As for spotting approaches, they usually find regions of interest in the drawings and describe them using various descriptors, and then index them based on the query symbol. The regions could be described in various ways: based on vectorial primitives as in Rusinol et. al. [10], or on graphs as in Qureshi et. al. [9] and Locteau et. al. [4] or on local shape contexts [8]. Different indexing techniques were used in those works as hashing and inverted file structures.

The scheme followed in these spotting methods is unrelated to our proposed matching framework, since our proposed algorithm searches for matching symbols in an image whether it contains an isolated symbol or a complete line drawing, without identifying or locating regions of interest first.

3 Description of the Recognition System

3.1 Adaptive Preprocessing and Feature Points

Although independent of the actual matching; preprocessing has an important effect on matching results, specially when the problem has extremely different noise types. As a preprocessing module, we use a largely modified version of the adaptive noise reduction algorithm by Zhang et. al. [14].

Zhang's algorithm [14] is based on the assessment of noise distributions and line widths, and then, according to this automatic assessment, different median and morphological filters are applied on an image. Inspired by the idea of the assesment of the image noise to automatically infer the noise type, we develop our adaptive preprocessing algorithm. It is called "adaptive" since it automatically adapts to the noise type and deals with it accordingly.

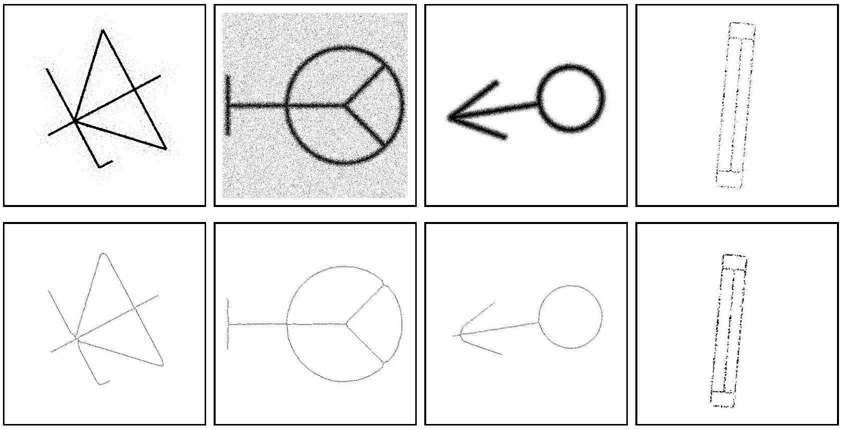

Fig. 1. Preprocessing; input: images in the first row, output: corresponding images in the second row

In many cases, the noise components are large, and its density distribution is higher than foreground symbols. Hence, instead of using actual median filters, we simply remove the small connected components that are smaller than a certain size. This size is adaptively and automatically determined based on noise distribution. This operation has the same effect as a median filter, but it removes noise components of different sizes with the same filter template.

Moreover, a smoothing step is needed to get better results in the subsequent feature extraction step, so we apply a Gaussian smoothing step followed by binarization. This operation is also adaptive based on the previously mentioned criteria. Fig. 1 shows the results of applying the preprocessing algorithm on different cases from the GREC'05 dataset.

For getting the feature points; equidistant points or line segments are sampled along the edges. Those feature points are the geometric primitives used as input for matching.

3.2 Recognition Using Geometric Matching

Geometric matching is concerned about simultaneously finding the pose and the correspondences between two sets of feature points using geometric relations. The two sets of features are usually extracted from two images. An example of such matching is shown in Fig. 2. In this section, we show how we use geometric matching to solve the problems of symbol recognition and symbol spotting.

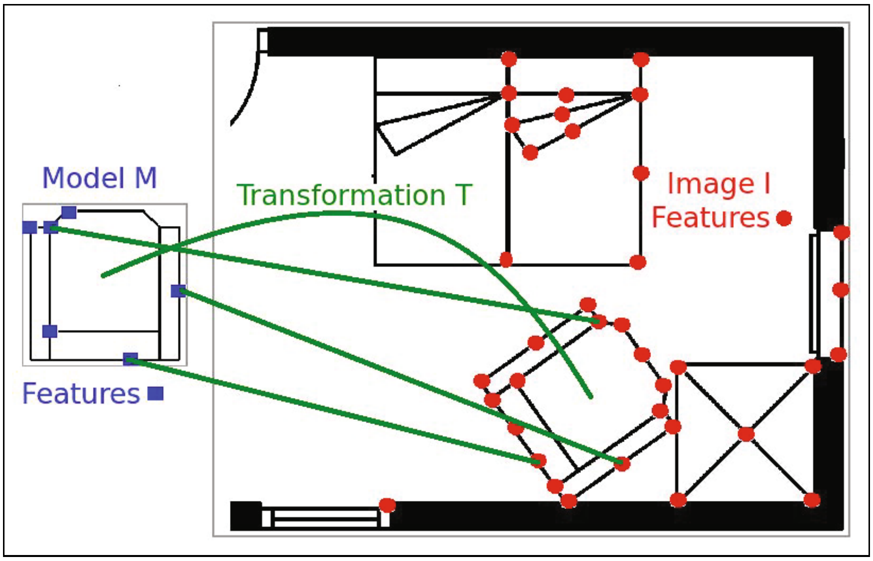

Fig. 2. Geometric matching: The model features (blue square points) are transformed by a certain transformation T (green curve) to match a subset of image features (red circle points). This matching gives us the correspondences between the two feature sets. For clarity of showing, we show only few feature points along the edges, and how some of them match.

Geometric Matching. First we define the problem of matching a pair of images: the model which is an isolated query symbol, and the image (an image of an isolated symbol or an image of a complete line drawing). This definition is adapted from [1].

Assume that the images have been preprocessed, so, the model M and the image I are defined as two sets of features. The features could be pixels, line segments or arcs. The matching solution is defined for any kind of geometrically parameterized features. The model $M = p_1, p_2, ..., p_m$ has m features, and the image $I = f_1, f_2, ..., f_n$ has n features.

A certain instance of the query symbol M can be inside the image I, but in that image, this symbol instance can be transformed with a certain transformation T which belongs to similarity transformations, so the symbol in the image can be a translated, rotated and/or scaled version of the query symbol M. Further more, the symbol instance inside image I could be surrounded by lots of clutter, for example, in non-isolated symbols, or complete line drawings, a certain symbol occupies just a small region within an image, the clutter in this case is the connecting lines and the other symbols in the image.

The goal is to find the transformation T_{max} that maps a **maximal subset** of the model features to **maximal subset(s)** of image features, with minimal error as defined by some error function.

First, the quality of a mapping two sets of features using a certain transformation T is calculated as:

$$Q(T) = \sum_{p \in M} \underset{f \in I}{max} \lfloor \|T(p) - f\| < \epsilon \rfloor \ . \tag{1}$$

where $\lfloor$*boolean expression*$\rfloor$ gives the value 1 if the *boolean expression* is true, and 0 otherwise, and ϵ is a user-defined error threshold that defines the maximum distance between two features to be considered matching to each other, we have experimentally chosen the ϵ parameter to be set to 6.0. Eq. (1) means that the quality is the count of the model feature points that match a subset of image features using a certain transformation.

As mentioned previously, the transformation T belongs to similarity transformations, so we calculate $T(p)$ in Eq. (1) as follows. Assume the feature p is a pixel feature defined by its coordinates (x, y) in the image, then:

$$T(p) = s * \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix} . \tag{2}$$

where s is the scale parameter, α is the rotation angle, and dx, dy are the translation parameters in the horizontal and vertical directions respectively.

Similarly, assume that the feature p is a line segment feature defined by the coordinates of its two end points (x_1, y_1) and (x_2, y_2), in order to transform this line segment with a certain transformation T, we simply apply Eq. (2) on each of the two end points.

We want to find T_{max} among all transformations T_{all} that maximizes this quality of match function for a certain matching problem:

$$T_{max}(M, I, \epsilon) = arg \max_{T \in T_{all}} Q(T; M, I, \epsilon) \ . \tag{3}$$

Finding T_{max} in Eq. (3) can be achieved using a search algorithm. The search is done in the 4D transformation space (2D translations, rotation and scale). We have set the transformation space parameters as follows: $dx \in [-dm_x, di_x]$, $dy \in [-dm_y, di_y]$, $\alpha \in [0, 2\pi]$ and $s \in [0.5, 3.0]$, where dm_x is the horizontal dimension of the model and dm_y is the vertical dimension of the model, and similarly di_x and di_y are the horizontal and vertical dimensions of the image respectively.

For each of the possible transformations in the search space, the algorithm transforms the model features M using that possible transformation T, and then calculates the quality of mapping those transformed model features to a subset(s) of the image features I using Eq. (1). The transformation that achieves the maximum quality of mapping is the transformation T_{max} that we are looking for according to Eq. (3).

Of course, a full search is not possible, so we use the branch and bound search algorithm described in our previous work [7]. The search algorithm can recognize/spot multiple instances of the query in an image as follows: once the algorithm finds one T_{max} result, it proceeds to find the next best mapping with the next T_{max}, until it stops according to a user-defined minimum quality of mapping. For example, if the model M has m features, the user can define the minimum quality of mapping to be 75% of the model features, this means at least 75% of the model M will always be matched to each of the model instances that exist in the image I. If the algorithm does not find any mapping that achieves the minimum quality, it will announce that there are no instances of the model inside the image. We have set the minimum quality parameter to 70% of the model features.

Penalizing Non-matched Features. In this subsection, we show how the recognition accuracy of our geometric matching framework can be improved by dealing with the problem of symbols that are very similar to each other.

Assume that we want to recognize the symbol in Fig. 3(a), so we will match it against the different available library symbols, among those symbols, there are some symbols that are very similar to each other like the two symbols in Fig. 3(b) and Fig. 3(d). Since the geometric matching aligns the points of the model on the points of the image, the test symbol in Fig. 3(a), will match both images in Fig. 3(b) and Fig. 3(d) with the same quality of match. The matching results are marked in red in Fig. 3(c) and Fig. 3(e). It is clear that the same number of model features were matched to both images, simply because the shape of symbol (b) is a subset of the shape of symbol (d). This might cause the problem of recognizing the test symbol in Fig. 3(a) to be the symbol in Fig. 3(d) instead of what it truely is (Fig. 3(b)).

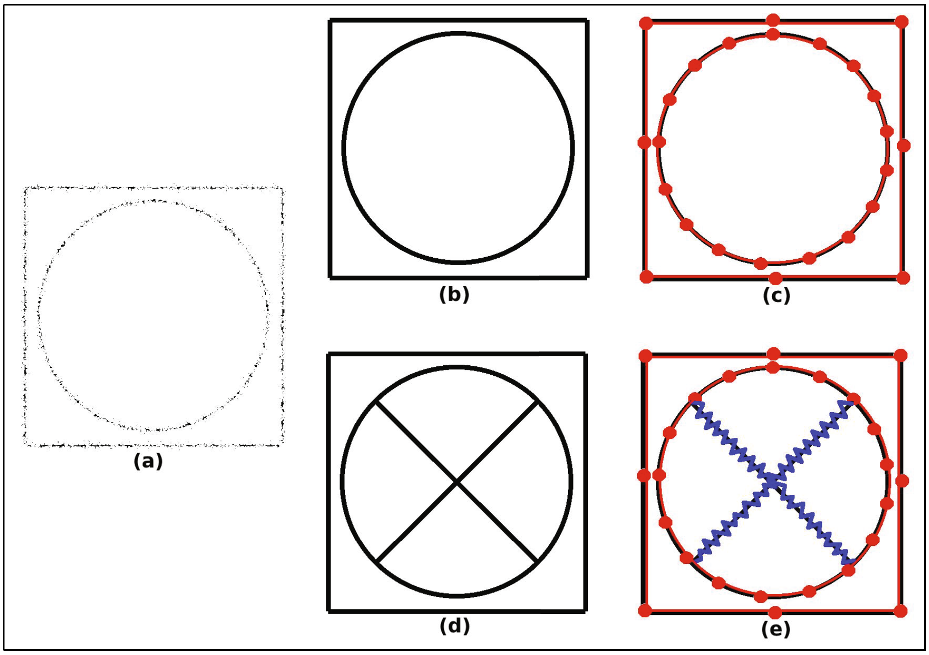

Fig. 3. Matching similar symbols shapes: (a) query test symbol (b) library symbol-1 (c) result of matching (red) the query symbol in "a" to the library symbol-1 (d) library symbol-2 (e) result of matching (red) the query symbol in "a" to the library symbol-2, some features (blue zig-zag) are not matched, because they do not exist in the query

We solve this problem by adding a quality value for each of the image features. This quality is calculated as the number of model points matched with this feature. This means, after the matching is done, we examine the model-image feature correspondences found by the algorithm. For each image feature, if it does not have a certain minimum number of model points matched with it, the quality of the overall match is penalized. This results in a higher quality of matching for the correct symbol. This is illustrated in Fig. 3(e), the image features marked with a zig-zag blue line, were not matched to any of the model points, we call those features the "non-matched features".

Assume the features marked with zig-zag blue in Fig. 3(e) are represented as 4 line segments, that means there are 4 non-matched features in the found match, the score of the whole match will be penalized by a certain amount. The penalty is calculated as follows. Each of the non-matched features has a feature size, in the case of line segments, the size is the length of the line segment, so for each non-matched feature we subtract half the feature size from the overall quality of match.

Note that the problem of similar shapes, does not arise in the following case: if the symbol we want to recognize is the same as the symbol in Fig. 3(d), then the quality of matching it to Fig. 3(d) will be higher than matching it to Fig. 3(b).

That is because each extra matching feature increases the overall quality of match.

The penalizing procedure easily generalizes to the non-isolated cases as follows. Having a number of candidate matching regions in a line drawing, we apply the same penalizing procedure by examining the features that are inside the region of a match.

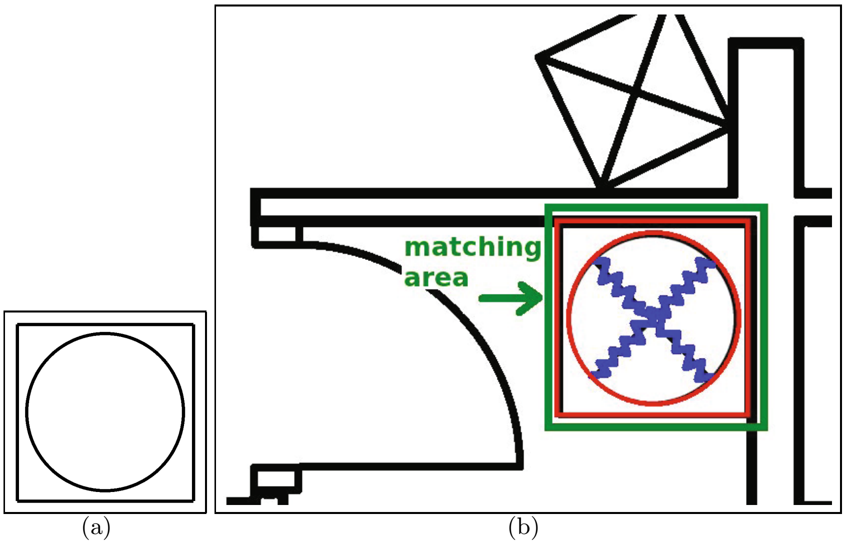

Fig. 4. Penalizing non-matched features for non-isolated cases (spotting): (a) query symbol (b) The algorithm finds the matching region (marked in green) with the feature correspondences, some of the features inside the region match to the query (marked in red), and some features do not match (marked in blue zig-zag lines), this match will be penalized

This is illustrated in Fig. 4. It is important to mention that we only examine the features that are inside the matching area (the bounding box that includes the matched features). That means we do not penalize the non-matched features that are outside this area. This is important for this procedure to work on non-isolated symbols, so the background and the lines connected to a symbol will not play any role in this penalization procedure.

3.3 Recognition

For the case of isolated recognition, the task is to recognize a certain query symbol. The query symbol is matched against a predefined set of library symbols. The query symbol is recognized to be the one that achieves the highest matching score among the library symbols. Fig. 5 shows a query symbol and its three best matches.

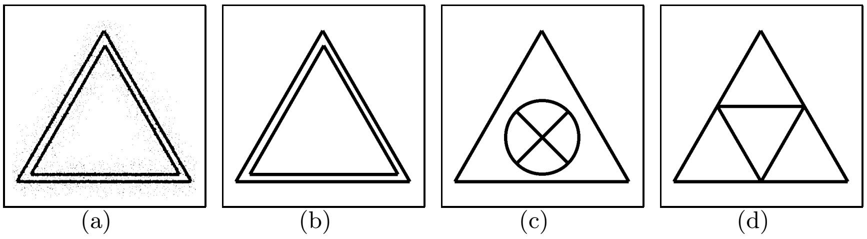

Fig. 5. Recognition results: (a) A test symbol (b) 1st best match (c) 2nd best match (d) 3rd best match

For the case of recognition in context –i.e. spotting–, the system spots all the regions that match the query symbol in all the drawings. Fig. 6 shows the spotted instances of two query symbols. In the figure, we draw shaded bounding boxes around the spotted instances of the query.

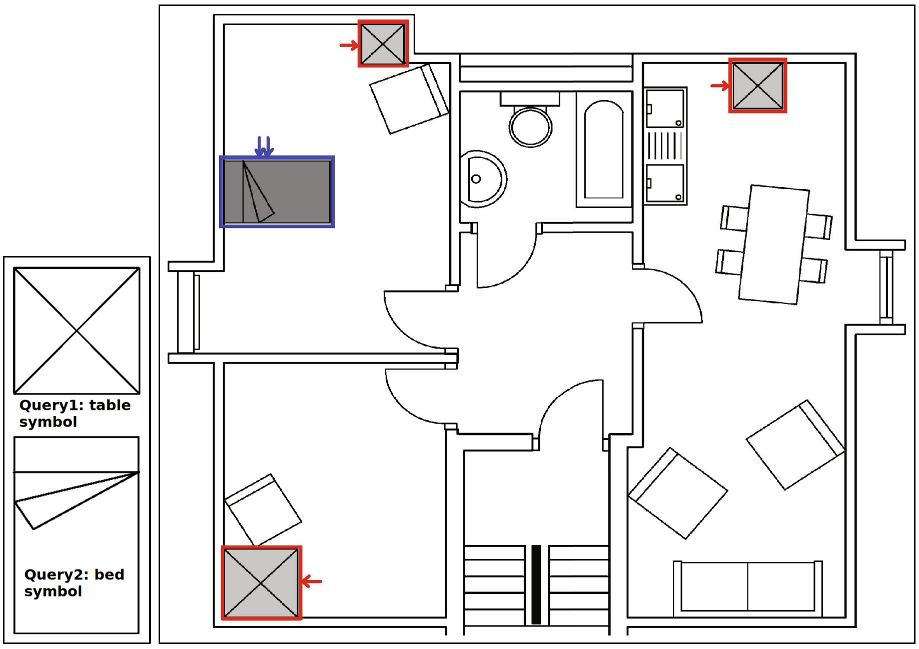

Fig. 6. Recognition results: (a) query symbol (b) spotting output

4 Performance Evaluation

4.1 The Datasets

We have used the publicly available GREC'05[1] and the GREC'11-contests datasets[2] for evaluation. Both datasets have a symbol library of 150 symbols segmented from architectural and electrical drawings. The symbols are composed of lines and circular arcs.

The test images in GREC'05 dataset are for isolated recognition, and they are degraded with 6 different noise types and different transformation (rotation and/or scaling).

The GREC'11 dataset has tests for both the isolated recognition and the spotting. In the isolated recognition tests, the test images are also transformed and degraded. There are 3 noise types with up to 25 levels of degradation in each type, the noise types simulate real world noise (due to scanning or otherwise).

The spotting training sets in GREC'11 dataset have 20 architectural drawings and 20 electrical drawings, also corrupted by 3 different noise types. The task is to spot query symbols in these drawings. There are 16 architectural and 21 electrical queries, we use only a subset of them.

4.2 Experimental Results

We present first the isolated recognition performance. We have evaluated our system on *all* of the tests of GREC'05 dataset, and also a random subset of the GREC'11 tests. Tables 1 and 2 show the results of testing with GREC'05 and GREC'11 datasets respectively. The recognition accuracy metric is used for evaluation.

Table 1. ISOLATED recognition results on all the tests of GREC'05 dataset (6000 images). The results are divided according to the degradation and transformation models used in the test images.

	noise1	noise2	noise3	noise4	noise5	noise6
no transformation	100	100	100	98.2	91.0	96.2
rotation	98.0	99.0	97.2	98.7	94.0	73.7
scaling	98.5	96.5	96.0	94.0	87.5	58.5
rotation+scaling	97.0	91.5	92.0	92.0	72.5	41.5

In general; the recognition accuracies decrease with scaling more than with rotation, because the scaled down versions of the models are severely corrupted even with simple degradation models, and they do not have much information about the symbol at the first place. Examples of such images are shown in Fig 7, those images are hard to recognize even by humans.

[1] `http://iapr-tc10.univ-lr.fr/index.php/symbolrecognitionhome`

[2] `http://symbcontestgrec05.loria.fr/`

Table 2. ISOLATED recognition results on GREC'11 dataset. N is the number of test images in each different case, total=3150.

Noise type (no transformation)	noise A (N=750)	noise B (N=750)	noise C (N=750)
Recognition accuracy %	98.6	94.0	97.0

Transformation & random degradation	rotation (N=300)	scaling (N=300)	rotation & scaling (N=300)
Recognition accuracy %	97.0	96.4	97.0

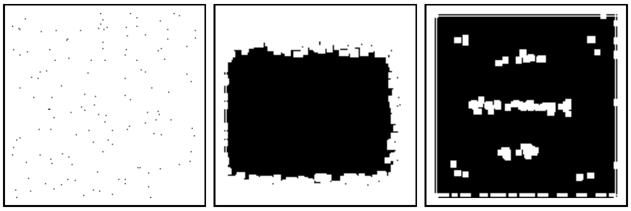

Fig. 7. Examples of scaled down corrupted images. The system fails to recognize those symbols.

Fig. 8 shows that the system scales well as the number of models increases. The scalability test is applied on the GREC'05 dataset.

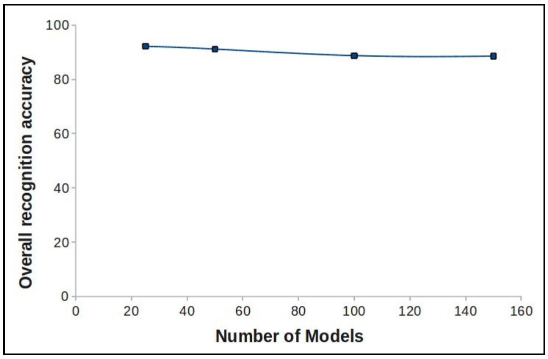

Fig. 8. Recognition system scalability

Table 3 shows the spotting performance of the system. Each of the recall and precision values is the average of spotting a number of different queries in all the drawings. As for the running time for matching a pair of images – both isolated and in context –, it ranges from a fraction of a second to few seconds.

Table 3. SPOTTING results on GREC'11 training datasets. Average recall and precision for 14 queries in 20 electrical drawings, and for 12 queries in 20 architectural drawings

Dataset	Noise type	# of instances of queried symbols	average Recall	average Precision
architectural	random	366	98.1	98.9
electrical	random	223	98.7	94.1

Table 4 shows a comparison with previous methods for both recognition and spotting. The comparison is shown for the methods that use the same datasets. Note that the methods that have higher overall accuracy than our method, have used a much smaller set of models and test images, also with simpler noise types and only a subset of the transformations.

Table 4. Comparison with previous methods. Note that some methods have used different numbers of models and test images, with different noise types and transformations.

Isolated Symbol Recognition - only for GREC'05 dataset					
Method	Accuracy	Models	Dataset - no. of tests	Noise	Transform
Our method	90.1	25 - 150	GREC'05 - 6000	Yes (all types)	Yes
Min et. al [6]	83.3	25 - 150	GREC'05 - 6000	Yes (all types)	No
Zhang et. al [16]	82.8	25 - 150	GREC'05 - 6000	Yes (all types)	Only rotation
Luqman et. al [5]	94.5	20 - 100	GREC'05 - 40	Yes (their own noise)	No
Wong et. al. [12]	94.0	25	GREC'05 - 100	yes (random)	Yes (random)
Symbol Spotting - for GREC'11 and similar datasets					
Method	Recall	Precision	Dataset (architectural)		Noise
Our method	98.1	98.9	12 queries, 20 drawings (8 backgrounds)		Yes
Nguyen et. al [8]	88.0	70.0	6 queries, 15 drawings		No

5 Conclusions and Future Work

The proposed system provides a practical and highly accurate solution for both isolated and non-isolated symbol recognition. The paper has shown that the use of geometric matching techniques can solve symbol recognition in context with interfering strokes, and can also perform competitively on isolated recognition tasks. The paper has also shown that adaptive preprocessing can be an important part of symbol recognition methods, specially in the case of extreme variability in images noise.

The methods that the system uses can be a basis for future work on applications like symbol retrieval and automatic analysis of line drawings. For example, using a set of known models, the spotting method can be used to spot symbols off-line, and then for on-line retrieval, the previously spotted symbols can be quickly indexed and retrieved.

References

1. Breuel, T.M.: Implementation techniques for geometric branch-and-bound matching methods. CVIU 90(3), 258–294 (2003)
2. Coustaty, M., Guillas, S., Visani, M., Bertet, K., Ogier, J.-M.: On the Joint Use of a Structural Signature and a Galois Lattice Classifier for Symbol Recognition. In: Liu, W., Lladós, J., Ogier, J.-M. (eds.) GREC 2007. LNCS, vol. 5046, pp. 61–70. Springer, Heidelberg (2008)
3. Locteau, H., Adam, S., Trupin, É., Labiche, J., Héroux, P.: Symbol Recognition Combining Vectorial and Statistical Features. In: Liu, W., Lladós, J. (eds.) GREC 2005. LNCS, vol. 3926, pp. 76–87. Springer, Heidelberg (2006)
4. Locteau, H., Adam, S., Trupin, E., Labiche, J., Heroux, P.: Symbol spotting using full visibility graph representation. In: GREC, pp. 1–7 (2007)
5. Luqman, M.M., Brouard, T., Ramel, J.: Graphic symbol recognition using graph based signature and bayesian network classifier. In: ICDAR, pp. 1325–1329 (2009)
6. Min, F., Zhang, W., Wenyin, L.: Symbol Recognition Using Bipartite Transformation Distance and Angular Distribution Alignment. In: Liu, W., Lladós, J. (eds.) GREC 2005. LNCS, vol. 3926, pp. 398–407. Springer, Heidelberg (2006)
7. Nayef, N., Breuel, T.M.: Graphical symbol retrieval using a branch and bound algorithm. In: ICIP, pp. 2153–2156 (2010)
8. Nguyen, T., Tabbone, S., Boucher, A.: A symbol spotting approach based on the vector model and a visual vocabulary. In: ICDAR, pp. 708–712 (2009)
9. Qureshi, R.J., Ramel, J.-Y., Barret, D., Cardot, H.: Spotting Symbols in Line Drawing Images Using Graph Representations. In: Liu, W., Lladós, J., Ogier, J.-M. (eds.) GREC 2007. LNCS, vol. 5046, pp. 91–103. Springer, Heidelberg (2008)
10. Rusiñol, M., Lladós, J., Sánchez, G.: Symbol spotting in vectorized technical drawings through a lookup table of region strings. Pattern Analysis and Applications 13(3), 321–331 (2010)
11. Su, F., Lu, T., Yang, R.: Symbol recognition combining vectorial and pixel-level features for line drawings. In: 20th International Conference on Pattern Recognition (ICPR), pp. 1892–1895 (2010)
12. Wong, A., Bishop, W.: Robust invariant descriptor for symbol-based image recognition and retrieval. In: IEEE Int. Conf. on Semantic Computing, pp. 637–644 (2007)
13. Yang, S.: Symbol recognition via statistical integration of pixel-level constraint histograms: A new descriptor. PAMI 27(2), 278–281 (2005)
14. Zhang, J., Zhang, W., Wenyin, L.: Adaptive Noise Reduction for Engineering Drawings Based on Primitives and Noise Assessment. In: Liu, W., Lladós, J. (eds.) GREC 2005. LNCS, vol. 3926, pp. 140–150. Springer, Heidelberg (2006)
15. Zhang, W., Wenyin, L.: A new vectorial signature for quick symbol indexing, filtering and recognition. In: ICDAR, pp. 536–540 (2007)
16. Zhang, W., Wenyin, L., Zhang, K.: Symbol recognition with kernel density matching. PAMI 28(12), 2020–2024 (2006)