Rezvanifar *et al. IPSJ Transactions on Computer Vision and Applications* (2019) 11:2
https://doi.org/10.1186/s41074-019-0055-1

IPSJ Transactions on Computer Vision and Applications

REVIEW PAPER Open Access

Symbol spotting for architectural drawings: state-of-the-art and new industry-driven developments

Alireza Rezvanifar, Melissa Cote and Alexandra Branzan Albu*

Abstract

This review paper offers a contemporary literature survey on symbol spotting in architectural drawing images. Research on isolated symbol recognition is quite mature; the same cannot be said for recognizing a symbol in context. One important challenge is the segmentation/recognition paradox: a system should segment symbols before recognizing them, but some kind of recognition may be necessary to obtain a correct segmentation. Research has thus been recently directed toward symbol spotting, a way of locating possible symbol instances without using full recognition methods. In this paper, we thoroughly review symbol spotting methods with a focus on architectural drawings, an application domain providing the document image analysis and graphic recognition communities with an interesting set of challenges linked to the sheer complexity and density of embedded information, that have yet to be resolved. While most existing methods perform well in terms of recall, their performance is rather poor in terms of precision and false positives. In light of the review, we also propose a simple yet effective symbol spotting method based on template matching and a novel clutter-tolerant cross-correlation function that achieves state-of-the-art results with high precision, high recall, and few false positives, able to cope with "real-life clutter" found in industry-standard architectural drawings.

Keywords: Architectural drawing, Document image analysis, Graphic recognition, Symbol spotting

1 Introduction

Symbol recognition is a particular application of the general problem of pattern recognition, in which unknown input patterns are classified as belonging to one of many classes (i.e., predefined symbol types) in the application domain. According to [1], symbols can be defined as the graphical entities which hold a semantic meaning in a specific domain and which are the minimum constituent that conveys the information. Logos, silhouettes, musical notes, and simple line segment groups with an engineering, electronics, or architectural flair constitute some examples of symbols that have been investigated by the document image analysis (DIA) community.

The research on isolated symbol recognition is quite mature. Several comprehensive papers summarizing the state of the art in symbol recognition can be found in the literature (see Table 1). However, the research on symbol recognition in context, i.e., the retrieval of symbols that are embedded in larger images as part of complex drawings, is still lacking in maturity. Recognizing a symbol in context has many more practical applications than recognizing an isolated symbol, but also entails many more challenges. In the earlier algorithms of symbol recognition in context, the retrieval process is mostly done by first using a segmentation step to extract regions of interest and then using a recognition step to validate these regions. However, the main challenge here is the segmentation/recognition dilemma (known as Sayre's paradox [2], originally formulated in the context of an automated handwriting recognition system): a system should segment the symbols before recognizing them but, at the same time, some kind of recognition might be necessary to obtain a correct segmentation [3]. This dilemma may explain why the older survey papers in Table 1 did not cover retrieval methods or only talked about possible segmentation approaches. The second challenge is that due to the ever-growing size of document image repositories,

*Correspondence: aalbu@uvic.ca
University of Victoria, Victoria, BC, Canada

Table 1 Key review papers on symbol recognition and spotting methods

Survey	Recognition	Spotting	Categorization approach
Chhabra [81], 1997	✓	×	Application domains (circuit diagrams, engineering drawings, architectural drawings, etc.)
Kasturi and Luo [82], 1997	✓	×	Application domains
Cordella and Vento [9], 2000	✓	Segmentation only	Stages of a general recognition method
Lladós et al. [83], 2001	✓	Segmentation only	Application domains and pattern recognition methods (structural vs. statistical)
Tombre et al. [4], 2005	✓	✓	Challenges and research directions
Rusiñol and Lladós [1], 2010	✓	✓	Stages of a general spotting method
Tabbone and Terrades [84], 2014	✓	✓	Stages of a general spotting method
Santosh et al. [85–87], 2015, 2016, and 2018	✓	✓	Statistical, structural, and syntactic approaches

fast(er) methods are necessary to handle large databases. To overcome these bottlenecks, the research trend has gradually shifted in recent years from the traditional concept of symbol recognition toward the new concept of symbol spotting, a way of locating a given query symbol within a graphical document image without using full recognition methods [4], while limiting the computational complexity. As a consequence, symbol spotting has experienced a growing interest in the Graphics Recognition community in such a way that the evaluation of symbol spotting approaches was added for the first time in 2011 to the series of IAPR Workshops on Graphics Recognition (GREC) symbol recognition contests [5]. Even though the research on symbol spotting is quite young, several review papers can be found in the literature that focus on the topic and are included in Table 1.

In this paper, we propose a review of the recent progress on symbol spotting with a specific focus on digital architectural floor plans as an application, along with a simple yet effective approach to architectural symbol spotting that addresses the issue of clutter, an important industrial requirement. An architectural floor plan (or architectural drawing) is a scaled two-dimensional diagram of one level of a building, consisting of lines, symbols, and textual markings. Created by an architect, an architectural floor plan shows the relationships between rooms and physical features from above and typically includes walls, doors, windows, staircases, fixtures, dimensions, labels of various kinds, and sometimes furniture. Figure 1 shows an example of the floor plan of a residential dwelling unit's foyer and kitchen areas.

Digital architectural drawings constitute a very challenging dataset for testing symbol spotting methods. This application was also noticed as being of particular interest in DIA communities, as can be seen in the Graphics Recognition workshops and related conferences. For example, [5–8] report on the four editions of the International Symbol Recognition Contest, which were held at GREC'03, GREC'05, GREC'07, and GREC'11, respectively. The main purpose of these contests was to provide some standard evaluation tools (datasets, ground truth data, evaluation metrics, and evaluation protocols) in order to compare the performance of different symbol recognition methods (and symbol spotting methods in the latest edition). As a result, one of the most popular databases for symbol recognition and spotting, Systems Evaluation SYnthetic Documents (SESYD), was proposed in [3], focusing on synthetic architectural drawings and electrical diagrams, part of which was used to create the dataset for the latest contest edition [5].

Symbol spotting methods are usually based on a query-by-example approach (QBE) [3]: an input symbol model is used as a query to locate similar symbols in architectural drawings. The output of a spotting method is a ranked list of similar symbols along with their localization data. This process can be more complex when no library of models is available at first and the input query is selected interactively. In this case, which is known as on-the-fly symbol recognition [4], learning-based methods cannot be used, and no initial assumption can be made about the shape of the input model. Finding and locating architectural symbols in context, i.e., as embedded elements of a realistic and complex complete architectural drawing, brings substantial difficulties to the analysis process due to the presence of clutter, connecting lines with other parts of the design, and overlaps with text and/or other symbols.

In the literature, symbol recognition and spotting methods have been categorized differently based on the authors' viewpoint. Categorizations follow, for instance, the application domains, the various stages involved in the overall process, and the challenges in the field (see Table 1). In this paper, we review papers according to their contributions with respect to the different steps involved in symbol spotting. The general framework of a symbol spotting method can be considered as consisting of three main levels [1]. First, the query symbol and input document images are described via feature descriptors. The descriptors can either extract features based

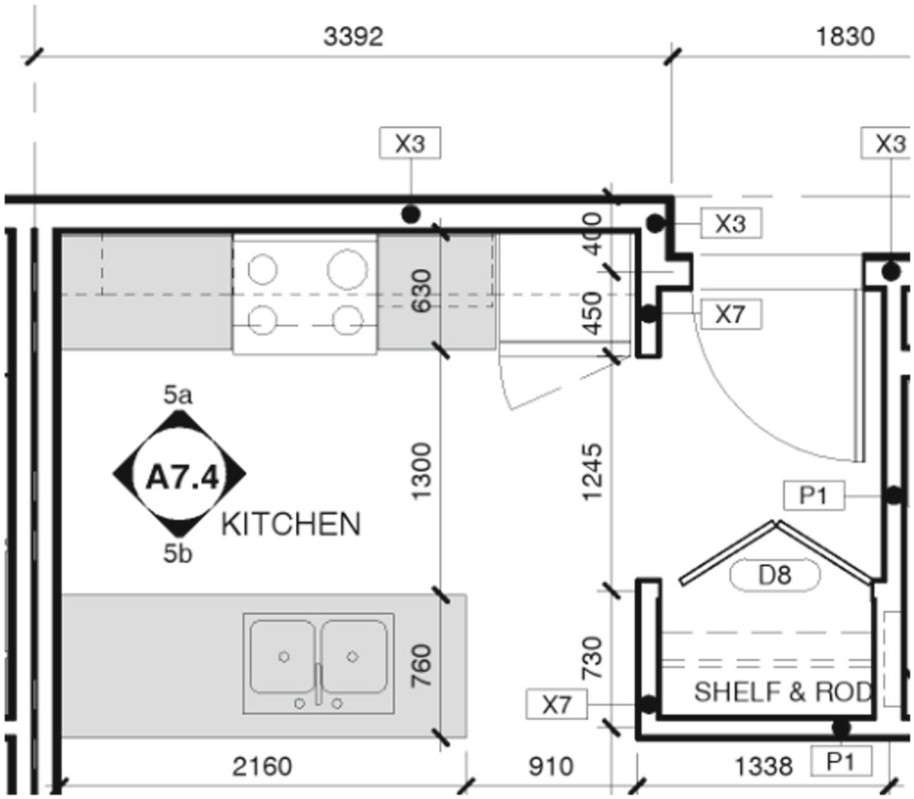

Fig. 1 Architectural floor plan example. Part of an architectural floor plan (kitchen and foyer areas of a residential dwelling unit)

on image pixels or on some vectorial primitives such as arcs, lines, and segments. At the second level, descriptors are organized in such a way that they can be efficiently employed for the purpose of matching. In some methods, this organization requires the extraction of regions of interest which are more likely to contain symbols. Finally, at the third level, hypotheses arising from the matching between models and document images are validated and a ranked list of localized symbols is obtained. Recent symbol spotting methods mostly bring novelty to the first two steps since the validation step is usually subjective and application-dependent. Figure 2 gives an overview of the categorization of the reviewed methods and corresponding sections in the paper.

1.1 Contributions

Our contributions are two-fold. From a theoretical viewpoint, we offer a thorough review of the literature on symbol spotting advances with a specific focus on architectural drawings, a particularly challenging application domain. Considering the performance of existing symbol spotting methods and industry-driven requirements, we propose, from a practical viewpoint, a simple symbol spotting method based on template matching and a clutter-tolerant cross-correlation function that is able to cope with overlapping elements and achieves state-of-the-art results on the dataset of the latest edition of the International Symbol Recognition and Spotting contest [5], strongly outperforming the contest participant.

The paper is structured as follows: Sections 2 and 3 review the literature on symbol spotting methods applicable to architectural drawings according to the proposed two-step categorization. In particular, Section 2 targets papers contributing to the description level, and Section 3 to the matching/locating (spotting) level. Section 4 focuses on the performance of symbol spotting methods on architectural drawings. Section 5 discusses key findings from the literature review. Section 6 describes our proposed symbol spotting method and discusses its experimental results. Section 7 presents concluding remarks.

2 Symbol spotting: description phase

As a first step of any symbol spotting method, much like for any other vision algorithm, the most significant and relevant information should be extracted from the document images for further processing. In this section, we review symbol information representations

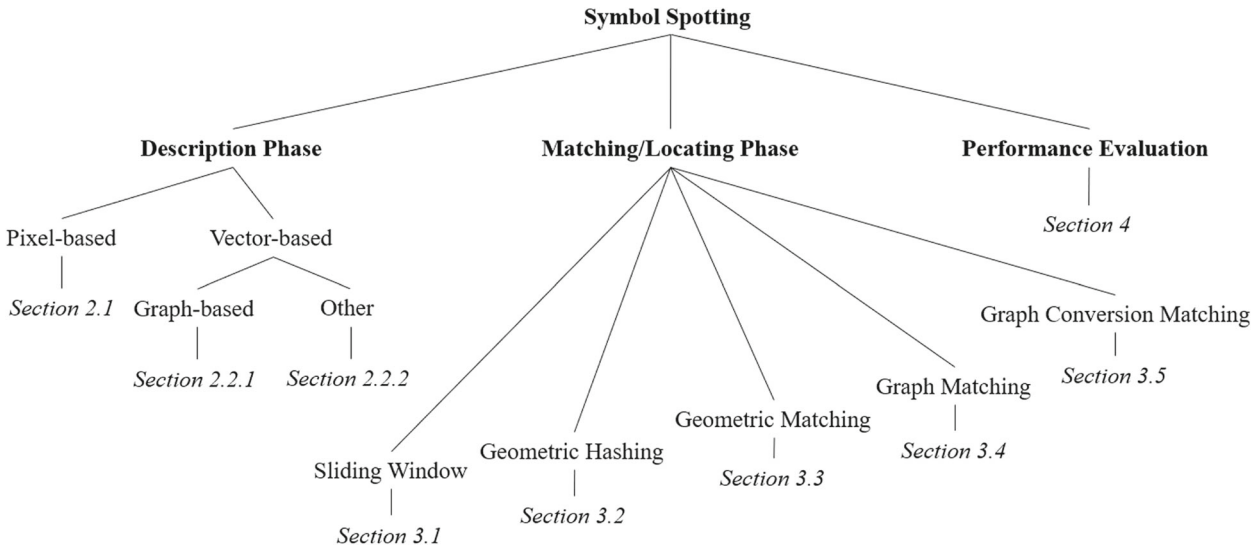

Fig. 2 Categorization of methods. Hierarchical overview of processes and concepts in the reviewed methods and corresponding sections in the paper

and descriptors which are used for architectural floor plan analysis. In [9], a distinction is made between the representation and the description of symbols as follows:

1. Representation phase: aims to reduce noise and extract only the most significant information from the images. Some examples are image skeletonization, polygonal approximation, and Hough or other transforms.
2. Description phase: tries to use preliminary information provided by the representation phase to build a new descriptor.

Here, we opted to review methods tackling the description phase, since representation phase methods are typically low-level, have been well-studied before and are not specific to architectural floor plan images nor symbol spotting.

In architectural floor plan analysis, images are typically bi-level or grayscale, and thus, *shape* [10], as opposed to color for instance, is the most important visual cue for describing them. The main challenges present in this step are scale and rotation changes, occlusions, elastic deformations, and intra- and inter-class variations for symbols. Ideally, images should be coarse to decrease the computation load of the matching/locating phase but at the same time accurate enough to yield a reasonable recognition rate. Several ad hoc descriptors have been introduced in the literature to address those challenges. They can be divided into two main categories based on the type of primitives that are used for representing a query symbol: pixel-based vs. vector-based. Pixel-based descriptors work directly on the raster image format. They are usually associated with statistical approaches and are typically (more) robust to noise. On the other hand, vector-based descriptors require the raster image format to be converted to vectorial primitives such as segments, polylines, and arcs. They typically allow for a more compact description, are usually associated with structural approaches (e.g., graphs), and typically lead to more false alarms and are more sensitive to noise. Table 2 summarizes descriptors according to this pixel-vector paradigm, which we review in the following subsections. Descriptors for both symbol recognition and symbol spotting are included as some relevant descriptors were proposed prior to the advent of symbol spotting. The table also mentions the low-level representation methods (corresponding to the representation phase in [9]) that are used in conjunction with each descriptor.

2.1 Pixel-based descriptors

One of the earliest pixel-based signatures (compact representations) of line drawing images was proposed in [11] based on the idea of the $\mathcal{F}$-signature, which is a particular histogram of forces. This signature calculates exerted attraction forces, based on an attraction function, between different segments of a symbol in each direction θ. This description is invariant to fundamental geometric transformations such as scaling, translation, symmetry, and rotation and is also able to handle symbol degradation to some extent. Although this descriptor is fast to implement with a computational complexity equal to $\mathcal{O}(p.n)$, where n is the number of image pixels and p the number of digitalization steps of angles in the histogram,

Table 2 Reviewed representations and descriptors for symbol recognition and spotting

Paradigm*	Descriptor	Robustness and invariance	Features used
P	$\mathcal{F}$-signature [11]	Robust to geometric transformations and noise	Image pixels
P	Pixel-Level Constraint [12]	Scale and rotation invariant and robust to degradation	Image skeleton
P	Blurred Shape Model (BSM) [13, 14]	Robust to soft, rigid, and elastic deformations	Skeleton points
P	Circular Blurred Shape Model (CBSM) [15, 16],	BSM properties + rotation invariant	Contour map (flexible for other representations)
P	Shape Context Descriptor (SCIP) and extensions [20, 22]	Scale and rotation invariant	Interest points
V	Perceptual grouping, Fully Visibility Graph (FVG) [24]	Rotation and scale invariant	Vectorial primitives
V	Structural representation and Attributed Relational Graph (ARG) [25, 26, 29, 30]	Scale and rotation invariant, robust to small variations	Vectors and quadrilateral primitives
V	Hierarchical Plausibility Graph (HPG) [32, 33]	Robust to various distortions	Critical points and lines
V	Shape, topology, and Region Adjacency Graph (RAG) [35, 36]	Rotation and scale invariant	Image regions
V	Boundary and Region Adjacency Graph (RAG) [37]	Rotation and scale invariant	Image regions
V	Convexity and Near Convex Region Adjacency Graph (NCRAG) [38]	Rotation and scale invariant	Oriented line segments
V	Bag-of-GraphPaths (BoGP) [39]	Rotation invariant	Critical points
V	Jacobs' statistical grouping [45]	Scale and rotation invariant	Contour map
V	Bag-of-Relations (BoR) [31]	Scale and rotation invariant and robust to irregularities	Thick (solid) components, circles, corners, and extremities
V	Cassinian ovals [47]	Not invariant to scaling and rotation	Polylines of closed region contours

*P = pixel-based, V = vector-based

the descriptor does not lend itself easily to recognizing symbols in context, embedded in a floor plan.

In [12], geometric constraints such as length ratios or angles between pair of pixels considering a third pixel as the reference point are summarized via histograms. These histograms are used to determine the similarity between symbols. Considering the constraints makes the descriptor rotation- and scale-invariant and also robust to degradation. One issue is that the complexity of the resulting algorithm is roughly equal to $\mathcal{O}(n^3)$, and like $\mathcal{F}$-signature, it is not easily applicable to recognition in context.

The Blurred Shape Model (BSM) descriptor was first introduced in [13] for recognizing handwritten symbols and further developed in [14]. To define BSM using skeleton points, the images are partitioned into a grid of $n \times n$ equal-sized sub-regions (where $n \times n$ identifies the blurring level allowed for the shapes). Each bin of the grid receives votes from the shape points it contains and also from the shape points in the neighboring bins. The most interesting property of this descriptor is its robustness to elastic and non-uniform distortions. However, it cannot cope with geometric transformations like rotation and scale changes. According to the experiments in [13], BSM outperformed several other descriptors (e.g., Zernike descriptors) in terms of accuracy and scalability. In terms of computational complexity, for a region of $n \times n$ pixels, $k \leq n \times n$ skeleton points are considered to obtain the BSM with a cost of $\mathcal{O}(k)$ simple operations, which is faster than the compared descriptors. To make BSM rotational invariant, an extension named Circular Blurred Shape Model (CBSM) was proposed in [15, 16]. CBSM is also able to handle irregular deformations and is fast to compute ($\mathcal{O}(k)$ for k contour points). Using correlograms (radial distribution of sub-regions of the image) and shape features derived from the Canny edge detector, CBSM is defined based on a spatial arrangement of pixels (contour map). Figure 3 shows the grid structures being used for BSM and CBSM. The correlogram grid of CBSM makes its rotation invariant when the final result is rotated and aligned around the main diagonal of the correlogram with the highest density. Results show that CBSM can outperform many other descriptors, including the original BSM, for multi-class object categorization problems. Although

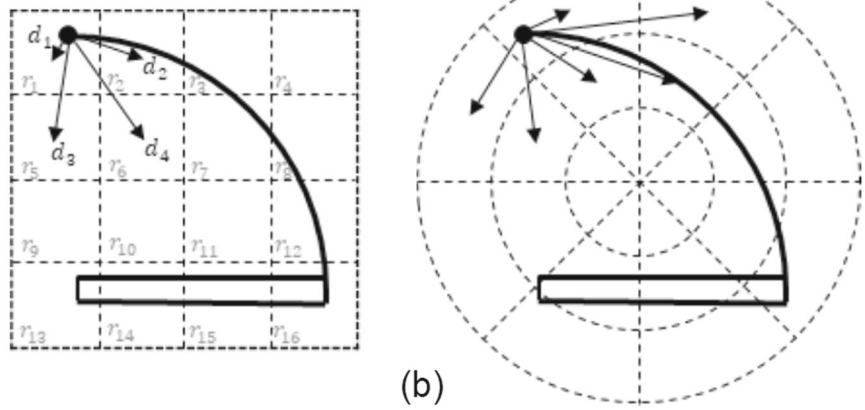

Fig. 3 Pixel-based descriptors. Grid structures of BSM (**a**) and CBSM (correlogram) (**b**) for describing a door symbol. Each boundary pixel votes based on its distance to the adjacent points

CBSM was successful in making BSM invariant to rotation, occlusion and noise can highly affect the location of the main diagonal of the correlogram. Moreover, the calculation of the scale change remains a challenge.

Some symbol descriptors are inspired from the concept of visual words in text indexing/retrieval approaches. This concept has been studied in many works related to video/image retrieval (see for instance [17–19]). In [20], a new descriptor, known as Shape Context for Interest Points (SCIP), is proposed in order to describe a graphic symbol using visual words. The shape context of a contour point is defined based on the bivariate histogram (distance and angle) of the relative coordinates of neighboring contour points. Since the number of contour points may be huge in real documents, shape context is only computed on key points (for instance obtained via the difference of Gaussian operator, or DoG). SCIP can adaptively reflect the local geometry of symbols based on their complexity and details and also guarantee invariance to scaling and rotation for isolated symbols. Unfortunately, the rotation and scale invariance is accomplished via a normalization step at the symbol level, and it is therefore difficult to apply the descriptor at the document level, i.e., for recognition in context. For this reason, an extension of SCIP (sometimes called ESCIP in some papers, such as in [21]) for the document level was introduced in [22]. In that paper, ESCIP is used to describe a document with a library of visual words. This allows for floor plans to be processed as text documents; thus, text retrieval/indexing techniques can be applied. Although the experimental results of this method are promising, false detections are a major issue, due to spatial relations between visual words being ignored and to an instability of the key point detection in the case of symbols composed of curves.

2.2 Vectorial descriptors

Considering the fact that architectural floor plans are man-made and digitally born, symbols are usually composed of regular primitives such as lines, arcs, and polylines. For this reason, a vectorial representation of symbols may seem appropriate to describe their shape. Typically, a vectorial representation of the meaningful parts of document images and symbols is first obtained via primitives, which are then further grouped via a suitable vector-based descriptor. We further categorize vectorial descriptors into graph-based and other, due to the prominent role of graph structures.

2.2.1 Graph-based

Graph structures constitute one popular way of utilizing vector-based descriptors. Although they cannot be considered themselves as descriptors, they constitute a flexible and powerful tool for describing the relational information found in architectural symbols. For that reason, we introduce here some related background information on graph structures typically used in the description phase for symbol spotting in architectural floor plan images. The main advantage of using graphs is their capability to represent individual symbols with variable size and complexity. This is because a graph size (number of nodes and edges) does not have to be set a priori and can be adjusted depending on the data. Moreover, graphs can simultaneously encode both the numeric properties and the structural data of a symbol. This latter property is what makes graphs popular in structural pattern recognition. To define nodes and edge attributes, scale and rotation invariant descriptors, in addition to relational information, are typically used in a way that makes the final description scale and rotation invariant, to avoid putting an extra computational load on the graph matching step.

A drawback of using a graph structures for describing objects (or symbols) is that pattern recognition (or symbol spotting) is converted to a subgraph matching (or isomorphism) problem which is NP-complete. Another disadvantage of graphs is their sensitivity to noise, as noise can affect the adjacency and Laplacian matrices of the graphs, and thus may negatively affect the final performance [23]. Nonetheless, graph structures remain highly popular for vector-based symbol description and spotting. Figure 4 shows several graph-based representations of a bed symbol.

In [24], vectorial primitives are considered as the nodes of a graph called Fully Visibility Graph (FVG), and edges represent the visibility between every pair of primitives. Figure 4c shows an example of FVG for the bed symbol of Fig. 4a. Visibility here refers to the existence of at least one straight line between two points on the primitives such that the line does not touch any other primitive in the drawing. Perceptual grouping of the primitives can then be done by applying clique detection to FVG. In [25], a structural representation of line drawing images is proposed based on polygonal and quadrilateral approximations of the symbol contours. After this vectorization step, a structural graph is constructed based on different types of interactions between vectors in order to organize the extracted vectors more appropriately. More specifically, the graph is an Attributed Relational Graph (ARG) as it extracts topological and geometric features. Figure 4d

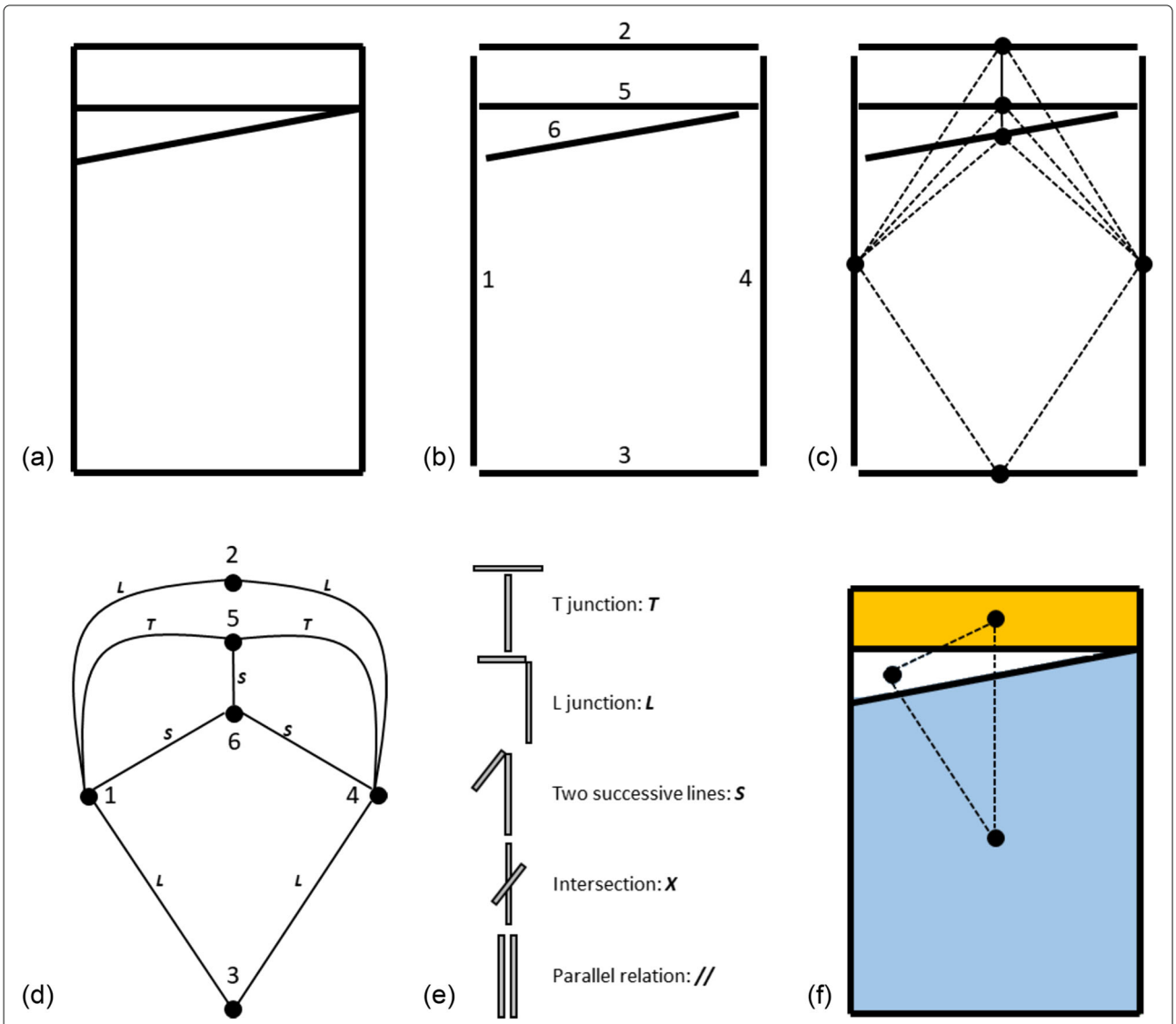

Fig. 4 Graph-based descriptions of a bed symbol. Bed symbol from SESYD (**a**), vectorized image (**b**), Fully Visibility Graph (FVG) (**c**), Attributed Relational Graph (ARG) (**d**), intersection types used for ARG (**e**), and Region Adjacency Graph (RAG) (**f**)

shows an example of ARG for the bed symbol considering the attributes defined in Fig. 4e. In this figure, *L*, *T*, and *S* show an *L* junction, a *T* junction, and two successive lines, respectively. The relative geometric features considered in ARG allow for invariance to rotation and scale changes and also make the graph robust to small variations in the symbols.

ARGs have also been utilized in hybrid symbol recognition approaches in order to integrate structural and statistical methods. In [26], a symbol is described by a set of spatio-structural descriptors (vocabulary) associated with visual primitives such as corners, circles, thick primitives, and extremities. This description is mainly inspired from [27, 28], where both complete symbols and symbols with missing parts are recognized using their vocabulary and grammar. The ARG-based spatio-structural description in [26] combines both topological and directional information specific to the symbol. The label assigned to each ARG vertex is the class of the corresponding extracted vocabulary, while the edge label is defined based on the spatial relation between two endpoint primitives. Two extensions of this method are proposed in [29] and [30] to include global shape signatures of each vocabulary class in the constructed ARG. Given the fact that graph nodes have unique labels and all instances of one specific vocabulary type are merged into one single node, the graph matching step for symbol recognition is not NP-hard. However, the underlying graph matching problem requires the symbols to have at least two different types of visual primitives in order to compute the spatial relations. Thus, the method cannot handle symbols containing only one primitive type [31].

One of the difficulties when working with graphs is their sensitivity to the vectorization step. Structural errors in this step can easily result in spurious nodes and edges, and in some disconnections between different parts of the graph, yielding to poor results. To cope with this problem, a hierarchical graph representation known as Hierarchical Plausibility Graphs (HPG) is introduced in [32, 33] to cover different possible vectorizations. HPG is able to solve three kinds of distortions: split nodes, dispensable nodes, and gaps. Although HPG can address the distortion issue under these circumstances, heavy distortion is still a major concern. In addition, building an HPG is time-consuming and the hierarchical matching algorithm sometimes fails to find local optima, yielding erroneous solutions.

Another type of graph representation called Region Adjacency Graph (RAG) can be created using regions of the floor plan images (e.g., based on connected components) as graph nodes and connecting the adjacent regions via graph edges (Fig. 4f). Typically, the characteristics of the region boundaries and the relational information between regions are used to label graph nodes and edges. Examples include boundary strings [34], Zernike moments for nodes and relative scale and distance for edges [35, 36], and histogram of distances between border points and centroids [37]. The main drawbacks of RAG is that it only considers closed regions as components of symbols and can fail to represent a symbol well in case of discontinuities in the boundaries. This problem is especially important when working with real-word architectural floor plan documents.

To tackle the problem of RAG with closed regions, the Near Convex Region Adjacency Graph (NCRAG) is proposed in [38], in which regions do not need to be clearly and continuously bounded. NCRAG rather focuses on the convexity of the different parts of a symbol, as convexity is one key symbol property. Even though NCRAG improves upon RAG, it still has limitations with respect to small variations of symbol instances in the architectural drawing, which can cause erroneous splits or merges between neighboring regions.

Since graph matching is a NP-complete problem, [39] proposed a Bag-of-GraphPaths (BoGP) descriptor that finds all acyclic paths between any two connected nodes of a graph representation of symbols and then used Zernike moments to describe these paths. The BoGP descriptor inherits the rotation invariance properties of graph paths. It is also unique in the sense that the descriptor converts the graph matching step to a statistical problem, which can be solved via hashing methods.

2.2.2 Other

While graph-based representations are popular for vectorial descriptors, other representations have also been proposed in the literature. One of the earliest methods for representing models in architectural plans is to consider a set of constraints between model segments obtained from a vectorization step (geometrical features) [40–42]. Taking inspiration from the work in [43, 44], these constraints are passed to a network of constraints and symbols are detected by testing constraints on different nodes of the network. In [45], the authors apply Jacobs' statistical grouping algorithm [46] to find meaningful (or salient) convex groups of geometric primitives, as convexity is considered as one of the non-accidental properties of line segments. Likewise, the approach in [31] avoids the use of graphs by extracting a meaningful vocabulary of visual primitives such as thick (solid) components, circles, corners, and extremities. The vocabulary is then used to build a Bag-of-Relations (BoR) based on pairwise topological and directional relations between visual primitives. Since the description is built from information on the spatial relations of symbols, it is robust to scaling and rotation changes and also to irregularities. In addition, the use of BoR indexing reduces the computation time during the recognition process, which is the main problem of

graph-based representations. The authors in [47] extract polylines of closed region contours of each symbol as primitives, which are then described with Cassinian ovals. Cassinian ovals were originally introduced in [48] to model the orbit of the Sun around the Earth. The use of this descriptor is an attempt to encode the eccentricity and non-circularity of an object. Using this descriptor, each polyline is encoded in terms of a tuple (a, b). This few-digit descriptor is then employed to build a hash table for indexing the input document. Unfortunately, Cassinian ovals are not invariant to scaling and rotation changes and are only suitable for a subset of architectural symbols, i.e., those comprised of only closed regions.

3 Symbol spotting: matching/locating phase

Once symbols are described by an appropriate descriptor, the most difficult challenge of a symbol spotting method is finding the embedded symbols in the documents in a way that avoids a segmentation step. To accomplish this matching step, one has a range of options, from a trivial brute force method, such as a sliding window, to more complicated mechanisms like graph matching. The remainder of this section reviews matching methods for locating symbols in context, which we have categorized as follows: sliding windows and correlation matching, geometric hashing, geometric matching, graph matching, and graph conversion matching.

3.1 Sliding windows and correlation matching

Searching the entire floor plan image via a sliding window (with or without overlap) performs an exhaustive search but is very time-consuming. In addition, depending on the robustness of the symbol descriptor, it may be necessary to search the image using windows with varying scales and rotations. Correlation-based methods like [49] which proposed a variation of the Hit and Miss transform for template matching, and some descriptors like CBSM [15] (see Section 2.1) and vectorial signatures [50] use this method to locate regions of interest (ROIs) in input images.

One way of avoiding the unreasonable computational complexity of exhaustive searches is to limit the search area to the most relevant parts of image. For instance, [11] applies a text string separation technique and implements the matching step on connected components. Like other matching methods relying on some form of pre-segmentation, such as connected components, or trying to detect ROIs first, there is an assumption that the symbols can be found as one entity within the components or ROIs, which is not necessary true.

3.2 Geometric hashing

Indexing methods constitute another popular approach for spotting symbols within architectural drawings; indexation usually involves encoding the symbols' and input documents' primitives via hashing functions in the form of lookup tables. The existing literature on primitive indexing methods usually follows the idea of geometric hashing introduced by [51]. In [52], the contours of the closed regions composing a symbol are described by a chain of adjacent polylines, then transformed into attributed cyclic strings. Two polylines are thus compared and matched by taking into account the different string edit operations needed to transform a string into another. To build an indexing lookup table for locating symbols, representatives of similar strings are used as indexing keys. Another example of hashing approaches consists in choosing a digit descriptor obtained from Cassinian ovals to describe symbols and building an indexing hash table [47].

In [53], the authors propose a structural approach for indexing vectorial drawings. In their method, a proximity graph, which is built based on extracted primitives from the image, is used to save not only the spatial relationship between primitives both also their numerical description in a 2D hash table. This approach, called relational indexation, allows for the consideration of spatial relationships between primitives in the retrieval process. Considering spatial relationships makes this approach invariant to scale and rotation transforms. One downside is that spatial information is only limited to the proximity (adjacency) between primitives, which might yield the same representations for different symbols.

3.3 Geometric matching

Nayef and Breuel in [54] use a geometric matching technique [55, 56] known as RAST (Recognition by Adaptive Subdivision of Transformation space) to search the space of the transformations for the most likely transformation parameters, and thus, the location of symbols in the document. In order to find matches of the features of a symbol model within a floor plan image, RAST performs a branch-and-bound search which yields a globally optimal solution. The authors employ either sample points along all lines in the drawings and symbol images, or segments and their orientation as feature points. They improved their geometric matching framework in [57] so that it is able to deal with vectorial primitives such as lines and arcs in addition to pixels. As an additional improvement, non-matched features are also penalized in order to decrease the number of incorrect matches. This approach has a relatively low precision when dealing with noisy or older documents, so a solution is presented in [58], which adds a post-spotting module in order to refine results. In this module, candidate regions obtained from the geometric matching algorithm are classified as true and false matches using a support vector machine (SVM) classifier.

3.4 Graph matching

As mentioned in Section 2.2, graph-based representations convert the problem of symbol spotting to a subgraph-matching problem; however, finding the optimal solution is not feasible since it is an NP-complete problem. Accordingly, several solutions have been proposed in the literature to find a reasonable solution. In [59], a new scoring function is proposed to evaluate the similarity of two different graphs. This scoring function employs a greedy graph matching algorithm to avoid the implementation of an exhaustive search.

In [60], an ARG representation is integrated with the graph matching technique proposed in [61] to build a complete symbol spotting framework. To decrease the computational complexity of the spotting step, a set of hypotheses are considered by the authors for detecting ROIs in the input architectural drawing images (or equivalently, detecting the parts of the corresponding graph that may include a symbol). This way, the graph matching only compares the model with ROIs, instead of the entire document. Although the extraction of ROIs can considerably decrease the computational complexity, the final performance of the algorithm is limited by the quality of this localization step. Indeed, the more comprehensive the hypotheses are, the better precision and recall can be expected, so the hypotheses should be properly defined in a way that no symbol will be missed, while being able to cope with occlusion and noise.

In [36], the authors propose a substitution-tolerant subgraph isomorphism to solve symbol spotting in technical drawings. Here, substitution tolerance means that the matching can cope with attribute differences such as vertex and edge labels, but unfortunately, a one-to-one mapping has to exist between each vertex and each edge of the template graph (symbol) and the target graph (floor plan); thus, structural distortions are not handled. In their approach, architectural floor plan images are described with a RAG, and subgraph isomorphism is modeled via an integer linear program (ILP) optimization problem. To generalize this approach for handling structural distortions, [62] proposes a binary linear program (BLP) which allows for the deletion of vertices and/or edges in the template graph. Unfortunately, there is no polynomial-time algorithm for solving ILP and BLP. The authors utilize Mathematical Programming (MP), which provides tools for solving optimization problems. In order to reduce the search space, these programs use a branch-and-bound algorithm along with some heuristics. Experimental results show better matching performance for BLP as well as faster solving time. The main drawback of these subgraph isomorphism approaches is that they remain unsuitable for large databases, since subgraph matching requires the investigation of the entire graph corresponding to the floor plan image for each symbol. Indexation techniques have thus the advantage over subgraph matching to be able to describe the input drawing just once in an offline process, allowing for a faster online querying. Another disadvantage of subgraph isomorphism approaches is that solving the optimization problem only yields the optimal solution in each implementation, which means that only the best match can be found in each iteration. The entire procedure must therefore be repeated several times for each symbol, ignoring the previous optimal solutions.

3.5 Graph conversion matching

To reduce the computational cost of graph matching, several works focus on translating graph characteristics into a statistical space and then use a statistical method for locating symbols, as opposed to structural approaches. We refer to this group of techniques as graph conversion matching.

Fuzzy Graph Embedding (FGE) is one of the conversion methods used for symbol spotting in [63, 64]. Graph embedding is a dimensionality reduction method which aims to exploit significant structural and statistical details of an attributed graph for embedding it into a feature vector (point) in a vector space. Feature vectors should reflect the similarity of corresponding graphs, i.e., the more similar two graphs are, the smaller the distance between their corresponding feature vectors should be. Graph embedding has the interesting advantage of enabling graph-based representations to access the whole range of statistical classifiers and machine learning tools. The resulting feature vector in the FGE approach is termed as Fuzzy Structural Feature Vector (FSFV), containing graph, node, and edge level features. Graph level features include graph order and size; node level features include fuzzy histograms of node degrees and of values taken by node attributes; edge level features include a fuzzy histogram of values taken by edge attributes. FGE employs fuzzy overlapping intervals for minimizing the information loss while mapping from continuous graph space to discrete vector space. The main drawbacks of this method are twofold: first, the presence of some imperfections in the representation of crossing lines or lines with angular points and second, a potential unwanted decomposition of a line into several quadrilaterals. The interested reader can refer to [23] for more information about graph embedding approaches.

Another alternative to graph matching is converting the geometric information carried by a graph to one-dimensional structures (serialization), which are less computationally expensive [65–67]. In these works, graph nodes are the critical points detected in the vectorized graphical documents and the lines joining them are considered as the edges. Serialization is accomplished by

factorizing the graph into a set of all the acyclic paths between each pair of connected nodes. After this factorization, the paths are described by a proper descriptor and symbol spotting can be done through hashing the shape descriptors of the graph paths. The factorization step introduces a lot of extra paths which can help the algorithm in handling distortion and noise, but on the other hand, it creates a lot of redundant information. In addition, since critical points are highly sensitive to noise, the serialization process can affect the final performance.

4 Symbol spotting: performance evaluation

As mentioned in Section 1, the symbol recognition and spotting contests of the GREC workshops have provided the research community with standard evaluation tools including ground truth data, evaluation metrics, and evaluation protocols for comparing the performance of different symbol recognition and spotting methods in architectural floor plan applications. The general framework of these three key tools was published in [68]. Only two datasets of architectural floor plans are publicly available: SESYD and FPLAN-POLY. The interested reader can find a comprehensive list of datasets for document analysis and recognition in [69]; however, these datasets are out of the scope of this paper. In FPLAN-POLY [53], the floor plans are provided as vectorized graphic documents, which differs from our focus on document images (although one might argue that they could be rasterized). The SESYD dataset[1] [3], related to the GREC symbol recognition and spotting contests, remains the most popular image dataset for evaluating symbol recognition and spotting algorithms. Figure 5 shows a typical floor plan, along with all 16 architectural symbol models, from SESYD. An overview of the use of SESYD can be found in [70]. As SESYD is a synthetic dataset, its authors proposed a generation tool to build synthetic documents that sets several constraints to determine various positions and locations of a given vocabulary in different ways over the same set of architectural backgrounds (building structure). Reference [71] proposed different evaluation metrics that should also be considered, introduced in terms of recognition abilities, location accuracy, and scalability, as well as a tool for manually annotating the location of symbols and their labels. In [72], a semi-automatic framework was proposed for ground-truthing real-word floor plan images. A top-down matching algorithm was used to minimize user interaction for defining the boundary of ROIs. The output of the tool includes graphics primitives composing the symbols as well as the location and class of symbols. Finally, [71, 73] introduced characterization metrics that take into account not only the retrieval ability of symbol spotting methods in real images, but also their localization strength.

Table 3 summarizes the results of symbol spotting algorithms that have been evaluated using architectural floor plan images from SESYD. In this table, the evaluation metrics are precision (P), recall (R), f-score (F), the average precision (AveP), and the retrieval time of each symbol in each document (T). Since spotting methods usually yield a ranked list of results, the quality of the ranking system cannot be assessed through standard precision and recall values, so AveP is used for this purpose. A detailed definition of all those parameters can be found in [71].

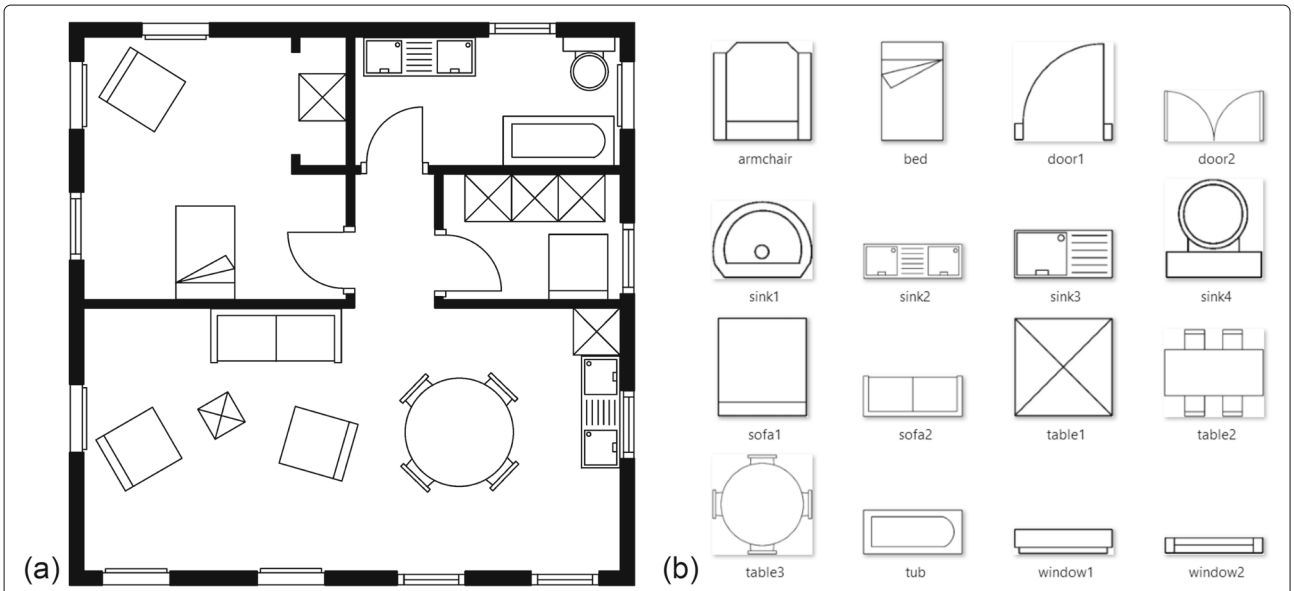

Fig. 5 Sample SESYD data. Sample synthetic architectural floor plan (**a**) and all architectural symbol models (**b**) from the SESYD dataset

Table 3 Performance evaluation of symbol spotting approaches on SESYD

Method	*P* (%)	*R* (%)	*F*	AveP (%)	*T* (s)	Subset
Nguyen et al. [22]	70.00	88.00	79.50	–	–	6 queries in 15 images from SESYD
Broelemann et al. [33]	75.17	93.17	83.21	–	–	SESYD (floorplan16-01, floorplan16-05, floorplan16-06)
Le Bodic et al. [36]	90.00	81.00	85.30	–	–	16 queries in 200 architectural plans from SESYD
Nayef and Breuel [57]	98.90	98.10	98.50	–	–	12 queries in 20 architectural plan from SESYD
Dutta et al. [38]	62.33	95.67	75.50	70.66	0.57	SESYD (floorplans16-01)
Dutta et al. [65]	56.92	83.96	67.85	60.87	0.07	SESYD
Dutta et al. [67]	50.32	83.06	62.67	60.87	0.07	SESYD

It can be seen from Table 3 that, while SESYD is the most popular performance evaluation dataset for symbol spotting, only a handful of authors in the literature used it for evaluation purposes. Moreover, different subsets are typically used (see last column), which makes it difficult to provide fair comparisons between the spotting methods. Also, one can notice that the precision is typically fairly low. False positives thus remain an open issue. In the next section, we expand on the remaining problems in the field.

5 Takeaways from literature review

Our literature review attempts to include the most important trends in the field of symbol spotting as they pertain to the application of architectural floor plan analysis. Section 1 mentions some of the most important challenges, such as the lack of a priori information about the shape of symbol models, the presence of noise, clutter, occlusion, and the variability of notations for a given class of models. Other problems also arise when working with real-word images found in industry (i.e., architectural floor plans that are actually used for building construction projects). The most popular dataset of document images for symbol spotting, SESYD, contains a set of synthetic images that are far from reflecting all the difficulties of real-word images. Thus, issues like occlusion, boundary discontinuities, distortions, and symbol irregularities, which are linked to real-word images, cannot be fully tested with SESYD. Since these problems are the main open challenges in real-word images used in industry, a considerable number of papers in the literature have tried to address these problems. For example, graphs are the main way to address irregularities and distortions of the images because of their flexibility in reflecting the structural relation between geometric primitives (although they are sensitive to noise that can throw off the vectorization process), despite a high computational complexity for the graph matching step. Although some authors have tested their approaches on their own datasets of real-word images, there is not a unique framework in the literature that would allow for fair comparisons of the ability of different methods to address the real-word architectural floor plan images problem.

In the definition of a symbol spotting problem, it is generally assumed that the model image is a piece of input document which is either cropped by the user or is already available in a library. Symbol instances, embedded in a document, do look like the query symbol from a topological viewpoint. However, when considering real-world floor plans actually used in industry, we are confronted with a large variability in the graphical notation used for a given architectural symbol class, that can affect even the most basic features of a symbol, such as its topology. This difficulty of clustering different designs of architectural symbols was noted in [71] in the context of ground-truthing graphic documents. As an example, a kitchen sink can have as many different representations as there are architectural firms, and even more (Fig. 6). The variability in graphical notation is an open problem which is almost impossible to solve with a symbol spotting method in its regular sense, i.e., without using query symbols for all potential graphical notations. Methods that do tolerate some inexact matching between a symbol model and a symbol instance in a floor plan cannot reliably solve

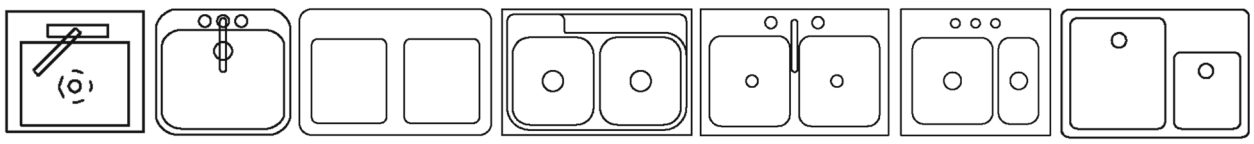

Fig. 6 Graphical notation variability. Examples of graphical notation variability for kitchen sink symbols

the problem due to potential changes in topology and the unbound variability. This issue seems more suitable for training-based methods which, however, bring their own issues such as a lack of sufficient and reliable training sets of symbols.

Deep learning techniques have recently enjoyed a spectacular success in detection, recognition, and classification tasks related to natural patterns [74, 75], but are not yet employed for symbol spotting. Such techniques (e.g., YOLO [76], r-CNNs [77]) are data hungry; thanks to large training datasets of natural images made publicly available by the computer vision community (e.g., ImageNet [78], MS-COCO [79]), these methods perform well for natural scenes. However, there is no equivalent of training datasets obtained from real-life architectural floor plans. Moreover, symbolic images such as floor plans have a high level of abstraction, conveying information via combinations of linear and curved segments, and lack the rich textural and colour appearance of natural images. Selecting semantically meaningful patches of symbols and non-symbols for training purposes would be very difficult, mostly due to clutter and symbol variability.

Finally, the existence of very similar geometric substructures in architectural symbols constitutes another important issue which can mislead spotting algorithms and decrease their scalability. This issue makes the different classes of symbols less distinguishable and causes low precision when working on large datasets (see Table 3).

In light of the surveyed literature, and especially of the performance of existing symbol spotting approaches presented in Section 4, we propose, in the following section, an industry-driven simple yet effective approach to architectural symbol spotting that yields a high precision and recall and is highly scalable.

6 Proposed symbol spotting approach

Our proposed approach utilizes template matching along with a novel clutter-tolerant cross-correlation function. According to the categorization of Fig. 2, the description phase of our approach falls into the pixel-based descriptors category. The features used are image pixels, after some minimal pre-processing (part of the representation) on the input architectural floor plan and symbol images. The matching/locating phase relies on template matching to spot symbol instances in an architectural floor plan image. In our opinion, this simple matching technique offers the best trade-off between the ability to precisely spot any symbol versus a robustness to clutter. The remainder of this section presents some theoretical background on template matching and cross-correlation, followed by our proposed clutter-tolerant cross-correlation function and the corresponding algorithm, then discusses our experimental results.

6.1 Theoretical background

The purpose of template matching is to find regions of a source image that are similar to a template image. In our case, this translates to finding regions within an architectural floor plan that are similar to an architectural symbol image. Cross-correlation measures the similarity between two series of data; it is thus commonly used as a similarity metric to indicate how well the template matches the source. The cross-correlation between the source image (f) and the template image (t) is typically computed from the following equation, with $\gamma(u,v)$ a cross-correlation coefficient at location (u,v):

$$\gamma(u,v) = \sum_{x,y} f(x,y) - t(x-u, y-v) \tag{1}$$

where $f(x,y)$ is the pixel intensity value at coordinates (x,y) of the source image, $t(x-u,y-v)$ is the pixel intensity value at coordinates $(x-u, y-v)$ of the template image, and (x,y) are incremented to cover all pixel coordinates of the region of the source image where the template image is superimposed.

One advantage of cross-correlation is that it can be computed at once for all locations over the source. However, it is not rotation—nor scale-invariant, which means that different angulations or scales of the template image compared to the source image will typically yield different levels of similarity. An architectural symbol can appear at various angles and scales on a floor plan. It is therefore necessary to compute the cross-correlation between the template image and the source image several times, each time with a modified template image (scaled and/or rotated), to cover all possibilities. Those modifications are considered as geometric transformations and are similar to those used in the geometric matching proposed by Nayef and Breuel [57], with the difference that translation transformations are not required.

While standard cross-correlation as in (1) allows for finding locations in a source image where matching with a template image is maximal, it does not allow for making a decision as to whether or not an object corresponding to the template has been located. As we cannot assume that there will be one and only one instance of the template in the source image, we cannot simply detect symbols based on the maximal cross-correlation coefficient. Moreover, cross-correlation coefficient values can range anywhere from 0 to an undetermined number, depending on the contents and the size of the template image. As template images can vary largely in contents and size, from one type of architectural symbol to another, standard cross-correlation with no known maximal coefficient value is of little use. Normalizing the coefficient values, between 0 and 1, would allow to make a decision based on a set threshold: for instance, we could consider that all coefficients that are larger than 0.75 indicate the location of a

match. If a source image would yield coefficients that are all below this threshold, then no match would be found.

The normalized cross-correlation normalizes the images by subtracting the mean pixel value from all pixel values and by dividing by the standard deviation of pixel values. This is usually done to remove variations in brightness. In Lewis' popular implementation [80], all correlation coefficients are brought back to the range [-1,1] as follows:

$$\gamma(u,v) = \left(\sum_{x,y}\left[f(x,y) - \bar{f}_{u,v}\right]\left[t(x-u,y-v) - \bar{t}\right]\right) / \left(\sum_{x,y}\left[f(x,y) - \bar{f}_{u,v}\right]^2 \sum_{x,y}\left[t(x-u,y-v) - \bar{t}\right]^2\right)^{1/2} \quad (2)$$

where $\bar{t}$ is the mean value of the template image, and $\bar{f}_{u,v}$ is the mean value of the source image in the region covered by the template.

One issue that might arise from (2) is that an instance of an architectural symbol on a floor plan might be connected to other elements of the floor plan, or might be overlapping other elements of the floor plan. In this scenario, to which we can refer as "real-life clutter," the extraneous items that are part of the region of the source image (floor plan) that is covered by the template image will influence the normalization process as they will be taken into account in the mean values. Therefore, an instance of a symbol on a floor plan that is isolated will yield a different (higher) coefficient than an instance of the same symbol that is not isolated. Selecting a proper threshold to make a decision as to whether or not an instance is considered as matched might be difficult in this case, as the more clutter present, the lower the coefficient. Too low a threshold would allow for the detection of non-isolated or cluttered instances but would also probably cause many false detections, while a higher threshold might work better in term of not detecting incorrect instances but miss non-isolated or cluttered correct instances.

6.2 Proposed clutter-tolerant cross-correlation

We propose a novel clutter-tolerant cross-correlation function to address real-life clutter, in which the correlation coefficients are brought back to the range [0,1] by dividing them by a normalizing factor w_t. We work with binary source and template images as the originals are typically binary or can be easily binarized. In the case of binary images, w_t simply corresponds to the number of "ON" (foreground) pixels in the template image, i.e., pixels composing the symbol:

$$\gamma'(u,v) = \frac{\sum_{x,y} f(x,y)\, t(x-u,y-v)}{w_t}. \quad (3)$$

With this formulation, only the "ON" pixels of the template count towards the computation of the correlation, which makes it clutter-insensitive. The computational complexity of (3) is the same as that of the standard cross-correlation (1). One drawback of this approach is that "ON" pixels of the template image can match "ON" pixels of the source image, whatever the shape of the elements contained on the source image. For instance, if the source image contains a solid region of "ON" pixels that is larger than the template—which rarely happens in architectural floor plans, then the template will yield matches over the solid region. This drawback can be addressed through pre-processing if necessary.

Our clutter-tolerant cross-correlation function bears similarities with the morphological-based Hit-or-Miss Transform extension proposed in [49] to deal with overlapping elements, but is faster and simpler in completely discarding template background information. Another important difference is that in [49], the authors change the threshold value for every single symbol query, while our method allows us to use the same correlation threshold value not only for all symbol queries, but also for all images (see Section 6.4 and Figs. 7 and 8).

6.3 Algorithm

Utilizing the proposed cross-correlation function (3), architectural symbols can be spotted in digital architectural floor plan images according to Algorithm 1. Steps 1 to 6 can be viewed as a representation/description phase, steps 7 to 11 as a spotting phase, and steps 12 to 15 as simple post-processing.

6.4 Experimental results

We evaluated our proposed method, implemented in MATLAB, on the architectural dataset used in the most recent edition of the GREC Symbol Recognition and Spotting Contest [5] (see Sections 1 and 4). The rationale for using the contest dataset as opposed to SESYD as some other methods do (see Table 3) is that the contest dataset is finite and well-defined. The test set in the architectural domain is comprised of 20 different synthetic floor plan images that contain symbol instances from 16 symbol models. Symbol instances can appear in the floor plans at any orientation across various scales. Various noise levels have been added to the 20 images to generate four subsets of 20 images each with varying degradation (ideal, level 1, level 2, and level 3).

We present the performance of our approach considering the correlation threshold TH as its main tunable parameter. Table 4 shows our results on the GREC contest dataset [5] for various TH values for the ideal case and compares them with Nayef and Breuel's method [57], the sole participant in the contest. The rationale for focusing

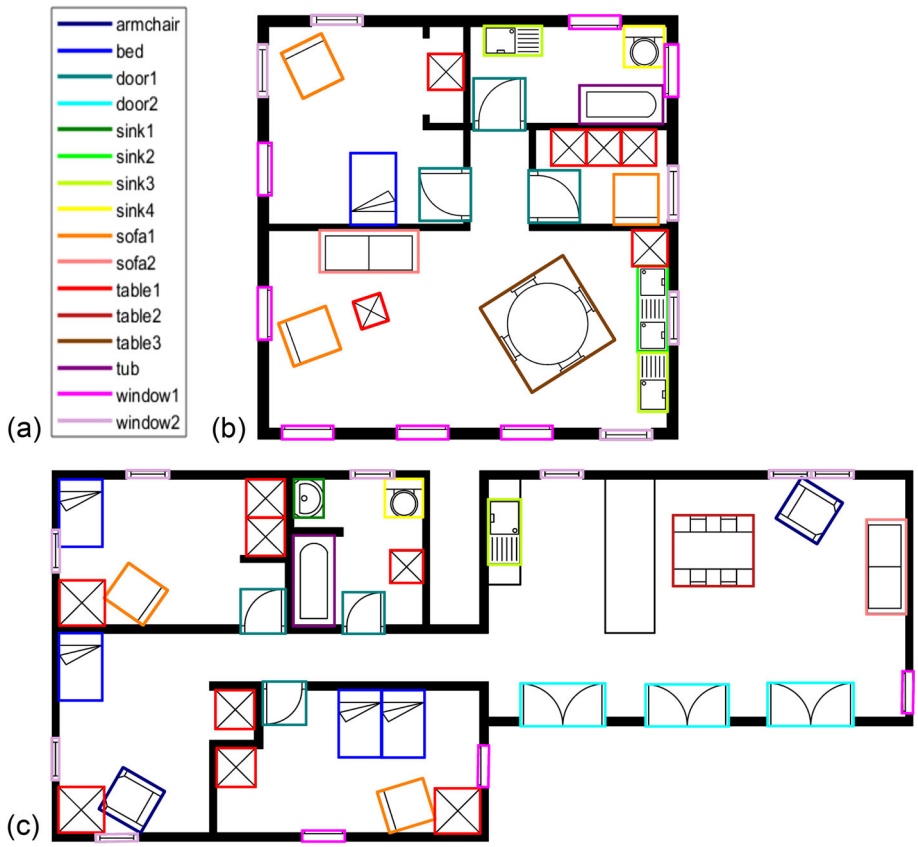

Fig. 7 Sample visual results on GREC contest dataset, part 1. Sample visual results of the proposed symbol spotting approach on the GREC contest dataset [5], ideal case, for a correlation threshold TH = 0.875. Color legend (**a**) and spotted symbols on images 6 (**b**) and 20 (**c**). In both examples, all symbols were correctly detected with no false positives

here on the ideal case for evaluation purposes is that it is representative of digitally born architectural floor plan documents, the industry standard. In Table 4, the precision (*P*), computed pixel-wise, represents the proportion of the detected symbol pixels that are part of actual symbols. The recall (*R*), also computed pixel-wise, represents the proportion of the actual symbol pixels that are detected as such. The F-score (*F*) is a combination of the precision and the recall (harmonic mean) into one single measure. The recognition rate (RecR), computed symbol-wise, is the percentage of symbols in the ground truth that are correctly identified. As per the contest instructions, a detected symbol is considered as recognized if there is an overlap of at least 75% between its area and that of the actual symbol in the ground truth. The average false positives (AveP), computed symbol-wise, is the average number of detected symbols with no correspondence in the ground truth per image over the test set. Figure 9 plots the precision-recall curve, showing the trade-off between a high recall and a high precision, with the best combination (highest F-score) highlighted.

From Table 4, we can see that our proposed approach yields excellent results (high precision, high recall, high F-score, high recognition rate and low average false positives) and significantly outperforms the contest participant [57] in terms of precision, F-score, and average false positives, by 36 p.p., 22 p.p., and 15 fewer false positives on average, respectively (with similarly high recall and recognition rate). Interestingly, as the correlation threshold TH decreases from 0.95 to 0.85, the precision monotonically decreases and the recall monotonically increases, which adequately reflects the "strictness" of the threshold. Indeed, a higher, stricter threshold of 0.95 yields fewer albeit better detections, while a lower, looser threshold of 0.85 allows us to detect more symbols at the price of more false positives. The F-score is highest for a correlation threshold TH of 0.875, as can be seen in Fig. 9. The recognition rate monotonically increases as the threshold decreases, reaching the theoretical maximal value of 100% at TH = 0.875, also corresponding to the best F-score. The average false positives monotonically

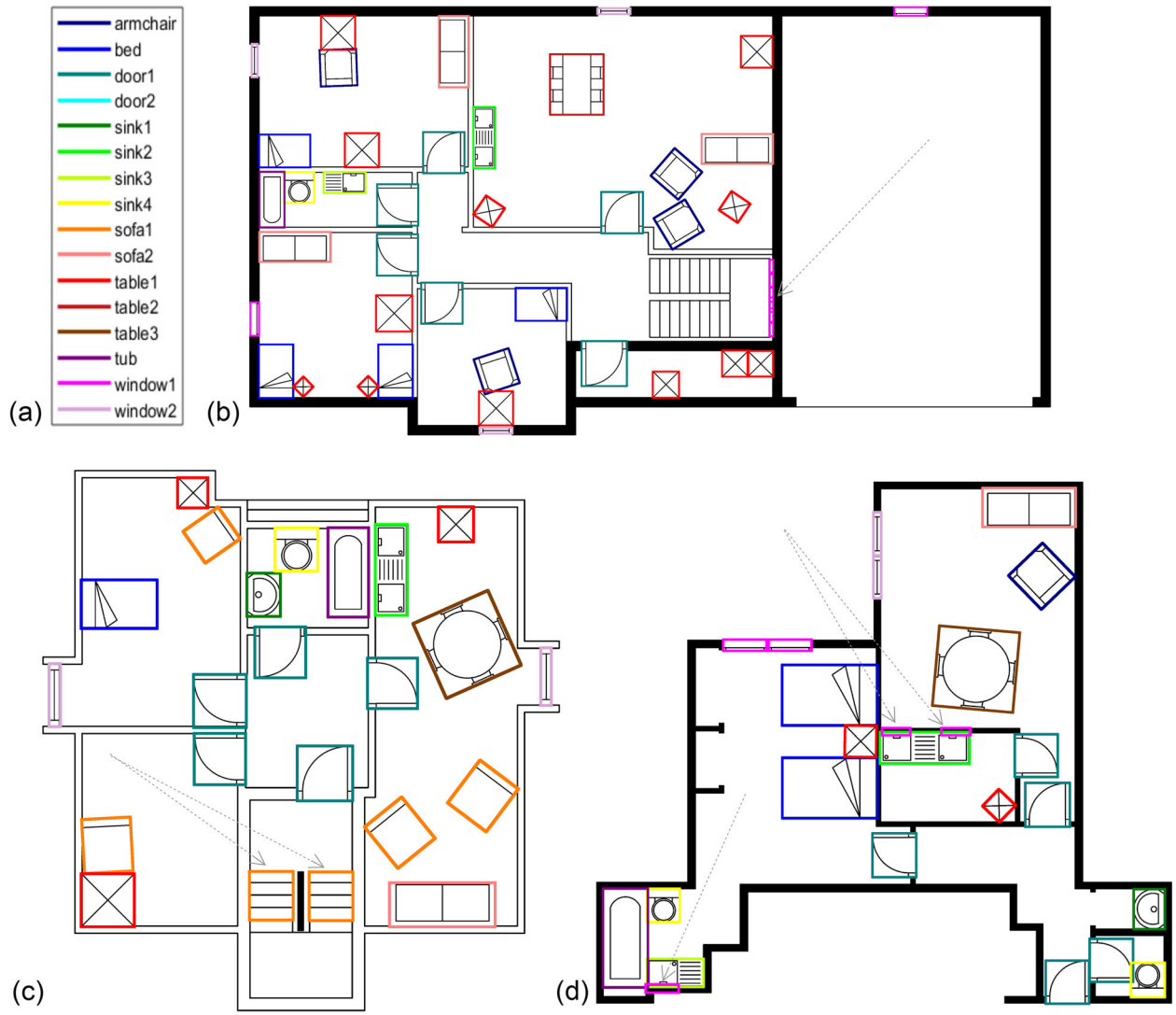

Fig. 8 Sample visual results on GREC contest dataset, part 2. Sample visual results of the proposed symbol spotting approach on the GREC contest dataset [5], ideal case, for a correlation threshold TH = 0.875. Color legend (**a**) and spotted symbols on images 1 (**b**), 8 (**c**), and 19 (**d**). In all examples, all symbols were correctly detected, along with six (**b**), two (**c**), and three (**d**) false positives, shown by dashed gray arrows

increases as the threshold decreases and is minimal (0) only at TH = 0.950, i.e., for the strictest threshold considered in the experiments. The monotonic behavior of the proposed algorithm, with respect to the correlation threshold value, is a highly desirable feature.

Figures 7 and 8 show sample visual results obtained using the best correlation threshold TH of 0.875, covering all five different layouts available in the dataset. Figure 7 illustrates ideal cases in which all symbol instances were correctly detected with no false positives, while Fig. 8 illustrates less-than-ideal cases where all true symbol instances are detected along with a few false positives. In the latter, all false positives (six in Fig. 8b, two in Fig. 8c, and three in Fig. 8d) reflect the downside of the clutter tolerance feature. The positioning of the symbols within the floor plans, as well as the floor plan structures, sometimes create regions that share similar features with a specific symbol to be spotted. For instance, in Fig. 8d, a kitchen sink symbol (sink2 or sink3 instances) is next to a wall, which creates a region that shares similar features with a specific window symbol (window1). A simple post-processing step looking at the proportion of ON pixels in the floor plan region covered by spotted symbols, i.e., its solidity, could be envisioned to discard such false positives.

Figures 10 and 11 show examples of symbol spotting on real-life architectural drawings used in industry to illustrate the claimed tolerance to clutter of our proposed

Algorithm 1: Clutter-tolerant symbol spotting

Input : Template (symbol) image t, source (floor plan) image f
Output: Spotted instances sy

Binarize and invert t and f;
// [Note: The original images, once binarized, will typically have their background as white pixels, which may be interpreted as "ON" pixels, thus the need for inversion.]
Optional: Replace thick lines by their edges in f;
// [Note: This optional step may be carried out to remove solid regions that could cause over-detections, such as the presence of thick solid walls combined with a search for very small symbol instances that could fit within the walls.]
for *all scales s tested* **do**
    Resize t at scale s: t_s;
    for *all rotations r tested* **do**
        Rotate t_s: t_{sr};
        Compute correlation coefficients between f and t_{sr} using (3): γ';
        // [Note: γ' is a correlation map about the size of f in which higher values indicate higher correlation.]
        Find peaks p in $\gamma' >$ TH;
        // [Note: TH is a correlation threshold used to consider a symbol instance as spotted, and should be $[0, 1]$. For instance, if TH = 1, a perfect correspondence of "ON" pixels between t_{sr} and f is required.]
        for *each peak* p_i *in* p **do**
            Extract spotted symbol coordinates within f using location of p_i and dimensions of t_{sr}: sy_i;
            // [Note: sy_i can be viewed as a bounding box containing a spotted instance.]
        end
        If large overlap between 2 spotted symbols in sy, discard sy_i with lower correlation value;
        // [Note: This is to prevent finding the same instance multiple times with small shifts.]
    end
    If large overlap between 2 spotted symbols in sy at different rotations, discard sy_i with lower correlation value;
    // [Note: This is to prevent finding the same instance multiple times with small differences in rotation.]
end
If large overlap between 2 spotted symbols in sy at different scales, discard sy_i with lower correlation value;
// [Note: This is to prevent finding the same instance multiple times with small differences in scale.]

Table 4 Evaluation of the proposed symbol spotting approach on the GREC contest dataset [5], ideal case (best overall performance shown in italics)

Method		P	R	F	RecR	AveP
Proposed	TH = 0.950*	0.989	0.947	0.968	97.00%	0.00
	TH = 0.925	0.986	0.962	0.974	97.60%	2.40
	TH = 0.900	0.985	0.977	0.981	98.30%	2.60
	TH = 0.875	*0.983*	*0.992*	*0.987*	*100.00%*	*3.60*
	TH = 0.850	0.966	0.992	0.979	100.00%	4.30
Nayef and Breuel [57], as reported in [5]		0.620	0.990	0.760	99.31%	18.75

*TH = correlation threshold

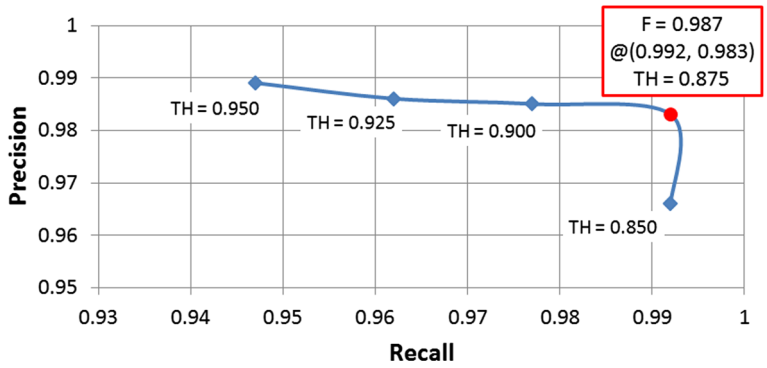

Fig. 9 Precision-recall curve. Precision-recall curve of the proposed symbol spotting approach on the GREC contest dataset [5], ideal case. The best performance, shown with a red circle, is for a correlation threshold TH = 0.875, with a corresponding F-score of 0.987, a precision of 0.983 and a recall of 0.992

approach. In Fig. 10, both examples are taken from a new construction project of a residential dwelling and contain an "intermediate" level of clutter. In the example on the left, the refrigerator (a) is correctly identified in (b) despite the additional layered information related to measurements and cupboards. In the example on the right, the two instances of the door symbol (c) are correctly identified in (d), despite the presence of layered text over one of the instances. In Fig. 11, both examples are taken from a renovation project of an old heritage building. Renovation project drawings typically contain a lot more layered data than new construction projects, making them ideal to showcase the clutter tolerance of our algorithm. Those additional data (notes, hatch lines, etc.) are necessary to indicate to the builder what to do with the existing conditions of the site. For instance, one pattern of hatch lines represents flooring infill, which indicates to the builder to patch/repair/level the floor in this area. Another pattern of hatch lines represents an area of a dropped ceiling to conceal piping from the unit above. In the example at the top of Fig. 11, bi-fold closet doors (a) are correctly identified in (b) in spite of the clutter that includes hatch lines at the same angle as one side of the closet doors. In the example at the bottom, two nightstands (lamp on top of a stand) (c) are correctly detected in (d) even though many different types of inscriptions and objects are in partial overlap with symbol instances. All symbol queries in Figs. 10 and 11 (a, c) are very different in terms of shape, topology, and symmetry, which also illustrates the genericness of the proposed method.

The proposed method has a number of advantages in addition to its monotonic behavior and simplicity. It is scalable both in terms of the number of symbol models and the number of symbol instances that it is able to

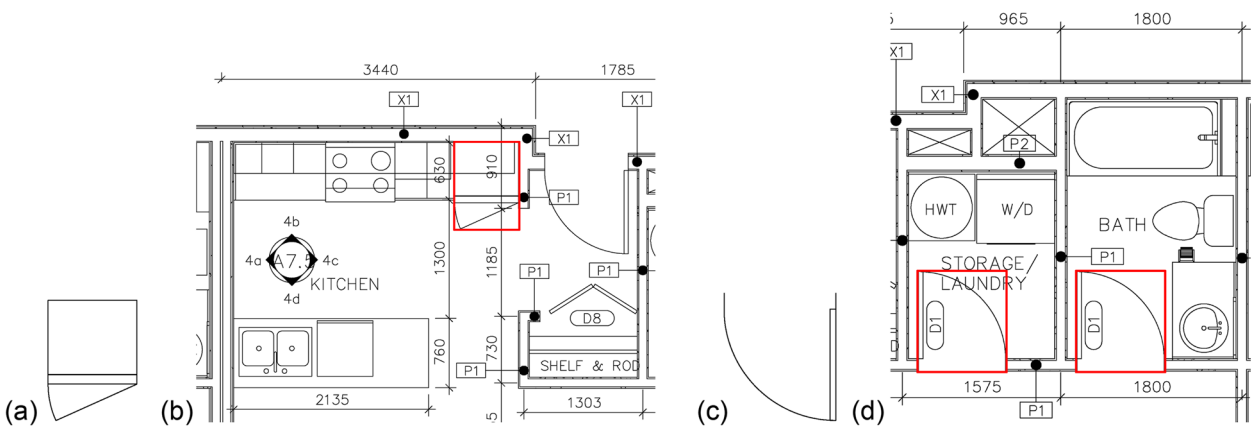

Fig. 10 Sample visual results on real-life architectural drawings, part 1. Sample visual results of the proposed symbol spotting approach on real-life, cluttered architectural drawings from a new construction project: refrigerator (**a**) and door (**c**) query symbols, and corresponding architectural drawings with spotted instances (**b**, **d**)

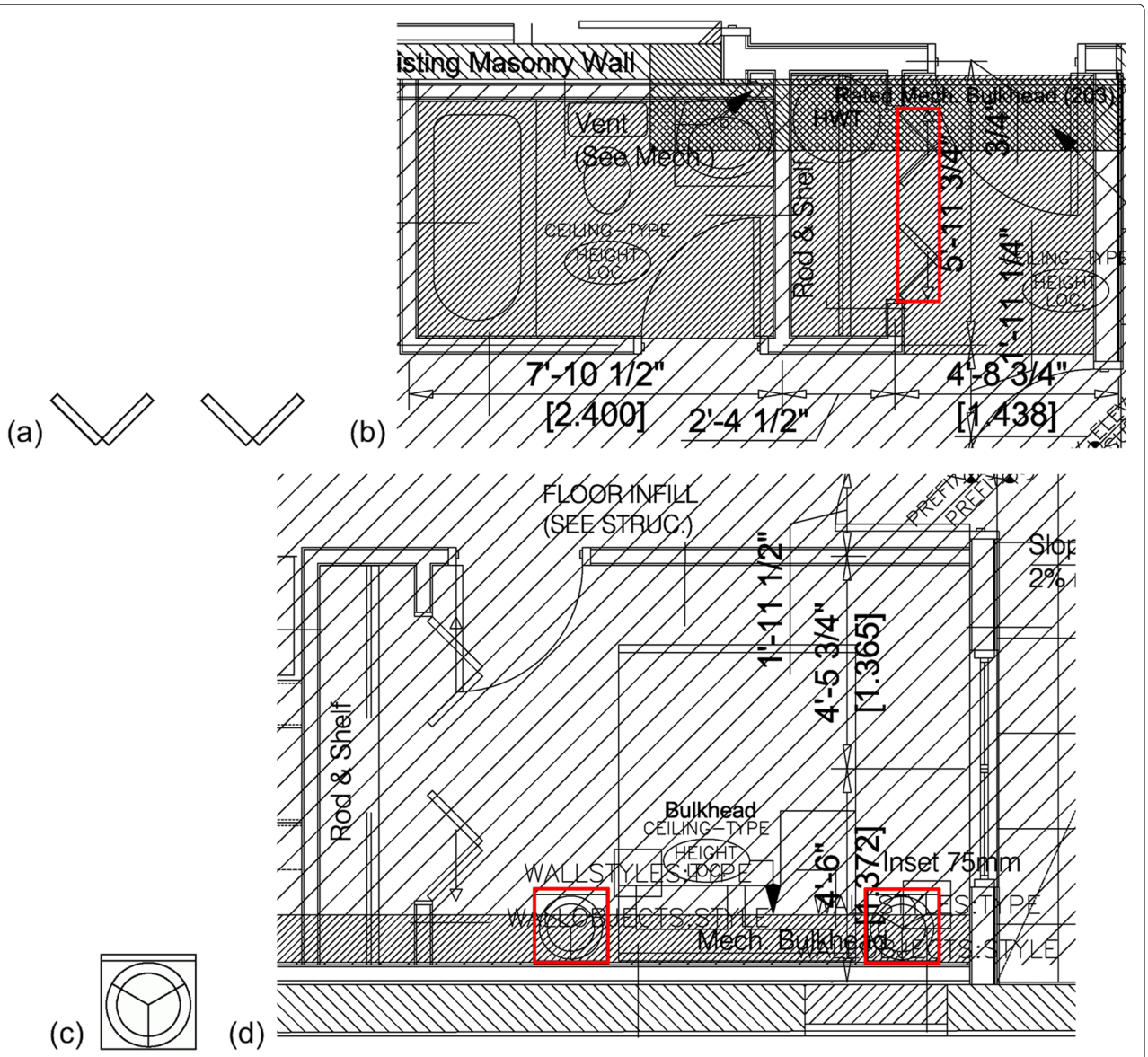

Fig. 11 Sample visual results on real-life architectural drawings, part 2. Sample visual results of the proposed symbol spotting approach on real-life, cluttered architectural drawings from a renovation project of an old heritage building: bi-fold closet doors (**a**) and nightstand (**c**) query symbols, and corresponding architectural drawings with spotted instances (**b**, **d**)

recognize in a single pass. It does not require any off-line pre-processing of the architectural floor plans as in graph-based approaches (see Section 2.2.1). It is generic enough to be applicable to any symbol design and was designed to cope with clutter, which is a particularly important issue in real-world architectural floor plan images found in industry. One can also tune the correlation threshold TH to make the method less strict in its matching and tolerate some level of noise. Since our method is not inherently scale- nor rotation-invariant, several geometric transformations are necessary to spot all potential symbol instances. However, these transformations are easily integrated into the proposed spotting algorithm (see Section 6.3).

7 Conclusion

This review paper presents a thorough survey of the literature on symbol spotting in graphical document images, with a specific focus on architectural floor plans as an application domain. Reviewed methods are categorized according to their contributions to each of two main phases in a symbol spotting framework: (1) symbol and floor plan image description and (2) symbol matching/locating within a floor plan. We also provide

a summary of the performance of spotting methods in the literature that were assessed with SESYD, the only standard public dataset of synthetic architectural drawing images. A key finding was that most existing methods perform rather poorly in terms of precision and false positives.

In light of the literature review and industry-driven needs, we propose a simple but effective spotting approach for architectural symbols that is based on template matching and a clutter-tolerant cross-correlation function that achieves state-of-the-art results, with excellent precision, recall, and recognition rate, and low false positives. The method has a desirable monotonic behavior with respect to the matching parameter value (correlation threshold), is scalable, and was designed especially to cope with clutter, a particularly important issue in real-world architectural floor plan images. Future research directions with respect to our proposed approach include looking into possible avenues for automatically determining the best correlation threshold value to use for a given dataset.

Architectural drawings, especially real-world drawings, provide the document image analysis and graphics recognition community with an interesting set of challenges that has yet to be resolved. Future work should definitely look at providing the community with a public image dataset and framework for assessing and comparing symbol spotting methods on real-world architectural floor plans as they are used in industry, with all the difficulties that they bring that are not replicated in synthetic document datasets. Such a real-world drawing dataset could be laborious to assemble due to potential intellectual property issues, but if successful, it would also pave the way for developing machine learning-based approaches that could tackle the variability in symbol graphical notation, an open issue quite problematic for the symbol spotting paradigm.

Endnotes

[1] http://mathieu.delalandre.free.fr/projects/sesyd/

Acknowledgements
This research was enabled in part by support provided by WestGrid (www.westgrid.ca) and Compute Canada Calcul Canada (www.computecanada.ca), as well as by the Natural Sciences and Engineering Research Council of Canada and Triumph Electrical Consulting Engineering Ltd. through the Collaborative Research and Development Grants Program. The authors would like to thank Steven Cooke at Triumph for his insights on how to interpret architectural drawings.

Funding
This research was supported by the Natural Sciences and Engineering Research Council of Canada and Triumph Electrical Consulting Engineering Ltd. through the Collaborative Research and Development Grants program. The funding body had no role in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

Availability of data and materials
This research used the publicly available GREC Symbol Recognition and Spotting Contest dataset [5].

Authors' contributions
AR collected and categorized publications for the literature review and wrote part of the manuscript. MC wrote part of the manuscript, helped with the literature review, and designed and carried out the experimentations for the proposed approach. ABA participated in the structural design of this manuscript and helped with the writing. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Received: 29 May 2018 Accepted: 23 April 2019
Published online: 10 May 2019

References

1. Rusiñol M, Lladós J (2010) Symbol spotting in digital libraries: focused retrieval over graphic-rich document collections. Springer, London
2. Sayre KM (1973) Machine recognition of handwritten words: a project report. Pattern Recognit 5(3):213–228
3. Delalandre M, Valveny E, Pridmore T, Karatzas D (2010) Generation of synthetic documents for performance evaluation of symbol recognition and spotting systems. Int J Doc Anal Recognit 13(3):187–207
4. Tombre K, Tabbone S, Dosch P (2005) Musings on symbol recognition. In: International Workshop on Graphics Recognition (GREC'05). Springer, Berlin. pp 23–34. https://doi.org/10.1007/11767978_3
5. Valveny E, Delalandre M, Raveaux R, Lamiroy B (2013) Report on the symbol recognition and spotting contest. In: International Workshop on Graphics Recognition (GREC'11). Springer, Berlin. pp 198–207
6. Valveny E, Dosch P (2003) Symbol recognition contest: a synthesis. In: International Workshop on Graphics Recognition (GREC'03). Springer, Berlin. pp 368–385. https://doi.org/10.1007/978-3-540-25977-0_34
7. Dosch P, Valveny E (2005) Report on the second symbol recognition contest. In: International Workshop on Graphics Recognition (GREC'05), vol. 3926. Springer, Berlin. pp 381–397. https://doi.org/10.1007/11767978_35
8. Valveny E, Dosch P, Fornes A, Escalera S (2007) Report on the third contest on symbol recognition. In: International Workshop on Graphics Recognition (GREC'07). Springer, Berlin. pp 321–328. https://doi.org/10.1007/978-3-540-88188-9_30
9. Cordella LP, Vento M (2000) Symbol recognition in documents: a collection of techniques? Int J Doc Anal Recognit 3(2):73–88
10. Zhang D, Lu G (2004) Review of shape representation and description techniques. Pattern Recognit 37(1):1–19
11. Tabbone S, Wendling L, Tombre K (2003) Matching of graphical symbols in line-drawing images using angular signature information. Int J Doc Anal Recognit 6(2):115–125
12. Yang S (2005) Symbol recognition via statistical integration of pixel-level constraint histograms: a new descriptor. IEEE Trans Pattern Anal Mach Intell 27(2):278–281
13. Fornés A, Escalera S, LLadós J, Sánchez G, Radeva P, Pujol O (2007) Handwritten Symbol Recognition by a Boosted Blurred Shape Model with Error Correction. In: Martí J, Benedí JM, Mendonça AM, Serrat J (eds). Pattern Recognition and Image Analysis. IbPRIA 2007. Lecture Notes in Computer Science, vol 4477. Springer, Berlin. https://doi.org/10.1007/978-3-540-72847-4_4
14. Escalera S, Fornés A, Pujol O, Radeva P, Sánchez G, Lladós J (2009) Blurred shape model for binary and grey-level symbol recognition. Pattern Recogn Lett 30(15):1424–1433
15. Escalera S, Fornés A, Pujol O, Escudero A, Radeva P (2009) Circular blurred shape model for symbol spotting in documents. In: 16th IEEE International Conference on Image Processing (ICIP'09). IEEE. pp 2005–2008. https://doi.org/10.1109/ICIP.2009.5413811
16. Escalera S, Fornés A, Pujol O, Lladós J, Radeva P (2011) Circular blurred shape model for multiclass symbol recognition. IEEE Trans Syst Man Cybern B (Cybern) 41(2):497–506
17. Agarwal S, Awan A, Roth D (2004) Learning to detect objects in images via a sparse, part-based representation. IEEE Trans Pattern Anal Mach Intell 26(11):1475–1490

18. Jurie F, Triggs B (2005) Creating efficient codebooks for visual recognition. In: Tenth IEEE International Conference on Computer Vision (ICCV'05), vol. 1. IEEE. pp 604–610. https://doi.org/10.1109/ICCV.2005.66
19. Sivic J, Zisserman A (2006) Video google: efficient visual search of videos. In: Toward Category-level Object Recognition. Springer, Berlin. pp 127–144
20. Nguyen TO, Tabbone S, Terrades OR (2008) Symbol descriptor based on shape context and vector model of information retrieval. In: Eighth IAPR International Workshop on Document Analysis Systems (DAS'08). IEEE. pp 191–197. https://doi.org/10.1109/DAS.2008.58
21. Ha DT, Tabbone S, Terrades OR (2016) Spotting symbol over graphical documents via sparsity in visual vocabulary. In: International Conference on Recent Trends in Image Processing and Pattern Recognition (RTIP2R'16). Springer, Singapore. pp 59–70. https://doi.org/10.1007/978-981-10-4859-3_6
22. Nguyen T-O, Tabbone S, Boucher A (2009) A symbol spotting approach based on the vector model and a visual vocabulary. In: 10th International Conference on Document Analysis and Recognition (ICDAR'09). IEEE. pp 708–712. https://doi.org/10.1109/ICDAR.2009.207
23. Luqman MM, Ramel J-Y, Lladós J (2013) Multilevel analysis of attributed graphs for explicit graph embedding in vector spaces. In: Graph Embedding for Pattern Analysis. Springer, New York. pp 1–26
24. Locteau H, Adam S, Trupin E, Labiche J, Héroux P (2007) Symbol spotting using full visibility graph representation. In: Workshop on Graphics Recognition. IAPR. pp 49–50. https://hal.archives-ouvertes.fr/hal-00671265/
25. Ramel J-Y, Vincent N, Emptoz H (2000) A structural representation for understanding line-drawing images. Int J Doc Anal Recognit 3(2):58–66
26. Santosh K, Lamiroy B, Wendling L (2012) Symbol recognition using spatial relations. Pattern Recogn Lett 33(3):331–341
27. Tombre K, Lamiroy B (2008) Pattern recognition methods for querying and browsing technical documentation. In: Iberoamerican Congress on Pattern Recognition. Springer, Berlin. pp 504–518. https://doi.org/10.1007/978-3-540-85920-8_62
28. Santosh K, Lamiroy B, Ropers J-P (2009) Inductive logic programming for symbol recognition. In: 10th International Conference on Document Analysis and Recognition (ICDAR'09). IEEE. pp 1330–1334. https://doi.org/10.1109/ICDAR.2009.166
29. Santosh K, Lamiroy B, Wendling L (2013) Spatio-structural symbol description with statistical feature add-on. In: International Workshop on Graphics Recognition (GREC'11). Springer, Berlin. pp 228–237
30. Santosh K, Lamiroy B, Wendling L (2014) Integrating vocabulary clustering with spatial relations for symbol recognition. Int J Doc Anal Recognit (IJDAR) 17(1):61–78
31. Santosh K, Wendling L, Lamiroy B (2014) BoR: Bag-of-Relations for symbol retrieval. Int J Pattern Recognit Artif Intell 28(06):1450017
32. Broelemann K, Dutta A, Jiang X, Lladós J (2012) Hierarchical Graph Representation for Symbol Spotting in Graphical Document Images. In: Gimel'farb G, et al. (eds). Structural, Syntactic, and Statistical Pattern Recognition. SSPR /SPR 2012. Lecture Notes in Computer Science, vol 7626. Springer, Berlin. https://doi.org/10.1007/978-3-642-34166-3_58
33. Broelemann K, Dutta A, Jiang X, Lladós J (2013) Hierarchical plausibility-graphs for symbol spotting in graphical documents. In: International Workshop on Graphics Recognition (GREC'13). Springer, Berlin. pp 25–37. https://doi.org/10.1007/978-3-662-44854-0_3
34. Lladós J, Martí E, Villanueva JJ (2001) Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. IEEE Trans Pattern Anal Mach Intell 23(10):1137–1143
35. Le Bodic P, Locteau H, Adam S, Héroux P, Lecourtier Y, Knippel A (2009) Symbol detection using region adjacency graphs and integer linear programming. In: 10th International Conference on Document Analysis and Recognition (ICDAR'09). IEEE. pp 1320–1324. https://doi.org/10.1109/ICDAR.2009.202
36. Le Bodic P, Héroux P, Adam S, Lecourtier Y (2012) An integer linear program for substitution-tolerant subgraph isomorphism and its use for symbol spotting in technical drawings. Pattern Recognit 45(12):4214–4224
37. Barducci A, Marinai S (2012) Object recognition in floor plans by graphs of white connected components. In: 21st International Conference on Pattern Recognition (ICPR'12). IEEE. pp 298–301. https://ieeexplore.ieee.org/abstract/document/6460131
38. Dutta A, Lladós J, Bunke H, Pal U (2013) Near convex region adjacency graph and approximate neighborhood string matching for symbol spotting in graphical documents. In: 12th International Conference on Document Analysis and Recognition (ICDAR'13). IEEE. pp 1078–1082. https://doi.org/10.1109/ICDAR.2013.215
39. Dutta A, Lladós J, Pal U (2013) Bag-of-GraphPaths descriptors for symbol recognition and spotting in line drawings. In: International Workshop on Graphics Recognition (GREC'11). Springer, Berlin. pp 208–217
40. Ah-Soon C (1997) A constraint network for symbol detection in architectural drawings. In: International Workshop on Graphics Recognition (GREC'97). Springer, Berlin. pp 80–90. https://doi.org/10.1007/3-540-64381-8_41
41. Ah-Soon C, Tombre K (1998) Network-based recognition of architectural symbols. In: Advances in Pattern Recognition. SSPR /SPR 1998. Lecture Notes in Computer Science, vol 1451. Springer, Berlin. https://doi.org/10.1007/BFb0033243
42. Ah-Soon C, Tombre K (2001) Architectural symbol recognition using a network of constraints. Pattern Recogn Lett 22(2):231–248
43. Messmer BT, Bunke H (1995) Automatic learning and recognition of graphical symbols in engineering drawings. In: International Workshop on Graphics Recognition (GREC'95). Springer, Berlin. pp 123–134. https://doi.org/10.1007/3-540-61226-2_11
44. Messmer BT, Bunke H (1997) Fast error-correcting graph isomorphism based on model precompilation. In: International Conference on Image Analysis and Processing (ICIAP'97). Springer, Berlin. pp 693–700. https://doi.org/10.1007/3-540-63507-6_262
45. Nayef N, Breuel TM (2011) Statistical grouping for segmenting symbols parts from line drawings, with application to symbol spotting. In: International Conference on Document Analysis and Recognition (ICDAR'11). IEEE. pp 364–368. https://doi.org/10.1109/ICDAR.2011.81
46. Jacobs DW (1996) Robust and efficient detection of salient convex groups. IEEE Trans Pattern Anal Mach Intell 18(1):23–37
47. Rusiñol M, Lladós J (2007) A region-based hashing approach for symbol spotting in technical documents. In: International Workshop on Graphics Recognition (GREC'07). Springer. pp 104–113. https://doi.org/10.1007/978-3-540-88188-9_11
48. Lockwood EH (1967) A book of curves. Cambridge University Press, Cambridge
49. Weber J, Tabbone S (2012) Symbol spotting for technical documents: an efficient template-matching approach. In: 21st International Conference on Pattern Recognition (ICPR'12). IEEE. pp 669–672. https://ieeexplore.ieee.org/abstract/document/6460223
50. Dosch P, Lladós J (2003) Vectorial signatures for symbol discrimination. In: International Workshop on Graphics Recognition (GREC'03). Springer, Berlin. pp 154–165. https://doi.org/10.1007/978-3-540-25977-0_14
51. Lamdan Y, Wolfson HJ (1988) Geometric hashing: a general and efficient model-based recognition scheme. In: International Conference on Computer Vision (ICCV'88). IEEE. pp 238–249. https://doi.org/10.1109/CCV.1988.589995
52. Rusiñol M, Lladós J, Sánchez G (2010) Symbol spotting in vectorized technical drawings through a lookup table of region strings. Pattern Anal Applic 13(3):321–331
53. Rusiñol M, Borràs A, Lladós J (2010) Relational indexing of vectorial primitives for symbol spotting in line-drawing images. Pattern Recogn Lett 31(3):188–201
54. Nayef N, Breuel TM (2010) Graphical symbol retrieval using a branch and bound algorithm. In: 17th IEEE International Conference on Image Processing (ICIP'10). IEEE. pp 2153–2156. https://doi.org/10.1109/ICIP.2010.5651137
55. Breuel TM (1992) Fast recognition using adaptive subdivisions of transformation space. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92). IEEE. pp 445–451. https://doi.org/10.1109/CVPR.1992.223152
56. Breuel TM (2003) Implementation techniques for geometric branch-and-bound matching methods. Comp Vision Image Underst 90(3):258–294
57. Nayef N, Breuel TM (2013) On the use of geometric matching for both: isolated symbol recognition and symbol spotting. In: International Workshop on Graphics Recognition (GREC'11). Springer, Berlin. pp 36–48
58. Nayef N, Breuel TM (2013) Combining geometric matching with SVM to improve symbol spotting. In: Document Recognition and Retrieval XX

(DDR'13). International Society for Optics and Photonics. p 86580. https://doi.org/10.1117/12.2002795
59. Qureshi RJ, Ramel JY, Cardot H (2007) Graph Based Shapes Representation and Recognition. In: Escolano F, Vento M (eds). Graph-Based Representations in Pattern Recognition. GbRPR 2007. Lecture Notes in Computer Science, vol 4538. Springer, Berlin. https://doi.org/10.1007/978-3-540-72903-7_5
60. Qureshi RJ, Ramel J-Y, Barret D, Cardot H (2007) Spotting symbols in line drawing images using graph representations. In: International Workshop on Graphics Recognition (GREC'07). Springer. pp 91–103. https://doi.org/10.1007/978-3-540-88188-9_10
61. Qureshi R, Ramel J, Cardot H (2006) Graphic symbol recognition using flexible matching of attributed relational graphs. In: 6th IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP'06), ACTA Press. pp 383–388. https://www.actapress.com/Abstract.aspx?paperId=28040
62. Lerouge J, Hammami M, Héroux P, Adam S (2016) Minimum cost subgraph matching using a binary linear program. Pattern Recogn Lett 71:45–51
63. Luqman MM, Delalandre M, Brouard T, Ramel J-Y, Lladós J (2010) Employing fuzzy intervals and loop-based methodology for designing structural signature: an application to symbol recognition. arXiv preprint arXiv:1004.5427. https://arxiv.org/abs/1004.5427
64. Luqman MM, Ramel J-Y, Lladós J, Brouard T (2011) Subgraph spotting through explicit graph embedding: an application to content spotting in graphic document images. In: International Conference on Document Analysis and Recognition (ICDAR'11. IEEE. pp 870–874. https://doi.org/10.1109/ICDAR.2011.178
65. Dutta A, Lladós J, Pal U (2011) Symbol spotting in line drawings through graph paths hashing. In: International Conference on Document Analysis and Recognition (ICDAR'11). IEEE. pp 982–986. https://doi.org/10.1109/ICDAR.2011.199
66. Dutta A, Lladós J, Pal U (2011) A Bag-of-Paths Based Serialized Subgraph Matching for Symbol Spotting in Line Drawings. In: Vitrià J, Sanches JM, Hernández M (eds). Pattern Recognition and Image Analysis. IbPRIA 2011. Lecture Notes in Computer Science, vol 6669. Springer, Berlin. https://doi.org/10.1007/978-3-642-21257-4_77
67. Dutta A, Lladós J, Pal U (2013) A symbol spotting approach in graphical documents by hashing serialized graphs. Pattern Recognit 46(3):752–768
68. Valveny E, Dosch P, Winstanley A, Zhou Y, Yang S, Yan L, Wenyin L, Elliman D, Delalandre M, Trupin E, Adam S, Ogier J-M (2007) A general framework for the evaluation of symbol recognition methods. Int J Doc Anal Recognit 9(1):59–74
69. Valveny E (2014) Datasets and annotations for document analysis and recognition. In: Handbook of Document Image Processing and Recognition. Springer, London. pp 983–1009
70. Delalandre M, Valveny E, Ramel J-Y (2011) Recent contributions on the SESYD dataset for performance evaluation of symbol spotting systems. Retrieved from semanticscholar.org, Dec. 2017. https://www.semanticscholar.org/paper/Recent-contributions-on-the-SESYD-dataset-for-of-Delalandre-Valveny/42a3d89544393fe80acb6d6c4eae0239c9c96b99
71. Rusiñol M, Lladós J (2009) A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. Int J Doc Anal Recognit 12(2):83–96
72. Delalandre M, Ramel J-Y, Sidere N (2013) A semi-automatic groundtruthing framework for performance evaluation of symbol recognition and spotting systems. In: International Workshop on Graphics Recognition (GREC'11). Springer, Berlin. pp 163–172
73. Delalandre M, Ramel J-Y, Valveny E, Luqman MM (2009) A performance characterization algorithm for symbol localization. In: International Workshop on Graphics Recognition (GREC'09). Springer, Berlin. pp 260–271. https://doi.org/10.1007/978-3-642-13728-0_24
74. Felsberg M (2017) Five years after the Deep Learning revolution of computer vision: state of the art methods for online image and video analysis. Linköping University Electronic Press. http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Aliu%3Adiva-143676
75. Everingham M, Eslami SA, Van Gool L, Williams CK, Winn J, Zisserman A (2015) The pascal visual object classes challenge: a retrospective. Int J Comput Vis 111(1):98–136
76. Redmon J, Farhadi A (2017) Yolo9000: Better, faster, stronger. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17). IEEE. pp 6517–6525. https://doi.org/10.1109/CVPR.2017.690
77. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14). IEEE. pp 580–587. https://doi.org/10.1109/CVPR.2014.81
78. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vis 115(3):211–252
79. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context. In: European Conference on Computer Vision (ECCV'14). Springer, Cham. pp 740–755. https://doi.org/10.1007/978-3-319-10602-1_48
80. Lewis JP (1995) Fast normalized cross-correlation. In: Vision Interface, vol. 10. Canadian Image Processing and Pattern Recognition Society. pp 120–123. https://s3.amazonaws.com/academia.edu.documents/3607631/10.1.1.157.3888.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1556776300&Signature=5YTa8qZ0UlYl1ircWTCrxOaMq%2Fw%3D&response-content-disposition=inline%3B%20filename%3DFast_template_matching.pdf
81. Chhabra AK (1997) Graphic symbol recognition: an overview. In: International Workshop on Graphics Recognition (GREC'97). Berlin. pp 68–79. https://doi.org/10.1007/3-540-64381-8_40
82. Kasturi R, Luo H (1997) Research advances in graphics recognition: an update. In: Advances in Document Image Analysis. Springer, Berlin. pp 99–110
83. Lladós J, Valveny E, Sánchez G, Martí E (2001) Symbol recognition: current advances and perspectives. In: International Workshop on Graphics Recognition (GREC'01). Springer, Berlin. pp 104–128. https://doi.org/10.1007/3-540-45868-9_9
84. Tabbone S, Terrades OR (2014) An Overview of Symbol Recognition. In: Doermann D, Tombre K (eds). Handbook of Document Image Processing and Recognition. Springer, London. https://doi.org/10.1007/978-0-85729-859-1_17
85. Santosh K (2016) Complex and composite graphical symbol recognition and retrieval: a quick review. In: International Conference on Recent Trends in Image Processing and Pattern Recognition (RTIP2R'16). Springer, Singapore. pp 3–15. https://doi.org/10.1007/978-981-10-4859-3_1
86. Santosh K, Wendling L (2015) Graphical symbol recognition. Wiley Encyclopedia of Electrical and Electronics Engineering
87. Santosh K (2018) Document image analysis. In: Document Image Analysis. Springer, Singapore. pp 1–15