Computers & Graphics 39 (2014) 101–116

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Recognition and autocompletion of partially drawn symbols by using polar histograms as spatial relation descriptors

Gennaro Costagliola, Mattia De Rosa, Vittorio Fuccella*

Dipartimento di Informatica, University of Salerno, Via Giovanni Paolo II, 84084 Fisciano (SA), Italy

ARTICLE INFO

Article history:
Received 9 August 2013
Received in revised form
16 December 2013
Accepted 17 December 2013
Available online 31 December 2013

Keywords:
Symbol recognition
Partial symbol recognition
Symbol autocompletion
Sketch recognition
Interface

ABSTRACT

We present an approach for recognizing multi-stroke hand-drawn symbols. The main feature of the approach is its capacity of recognizing partially drawn symbols. Furthermore, it is invariant with respect to scale, and supports symbol recognition independently from the number and order of strokes. The recognition technique is based on subgraph isomorphism and exploits a novel spatial descriptor, based on polar histograms, to represent relations between two stroke primitives. Using different symbol sets, both hand-drawn and artificially deformed, we evaluated the effectiveness of the approach in recognizing the symbols as a function of the number of primitives already drawn by the users. The results show that the approach gives a satisfactory recognition rate with partially drawn symbols, also with a very low level of drawing completion, and outperforms the existing approaches proposed in the literature. We also report the results of a user study aimed at evaluating whether the users can efficiently exploit symbol autocompletion.

1. Introduction

The spread of systems equipped with touch screens has recently attracted the interest of researchers on hand-drawn symbol and diagram recognition technology. At the same time, the availability of increased computing power has enabled the development of systems able to predict the user's input to improve the user experience.

Autocompletion has proven effective or appreciated by the users in various text-based applications [1–4], and, recently, researchers in [5] demonstrated that computer assistance in the completion of graphical symbols can be exploited by users with significant advantages in terms of drawing efficiency and accuracy. To date, only a few systems [6–9] have been introduced supporting such a feature. Nevertheless, symbol autocompletion can be advantageous for the users in several applications. For instance, it can be useful to accelerate symbol retrieval in icon-driven user interfaces [10] or the handwriting of oriental characters, replacing the current complex systems, such as the *pinyin* coding system [11].

The main contribution described in this paper regards automatic completion of hand-drawn symbols. To this aim we propose an approach for the recognition of partially drawn symbols and the design of a system for the autocompletion. The approach we describe uses an *Attributed Relational Graph* (ARG) [12] to represent symbols and is invariant with respect to scale. Furthermore, the user can plan to draw the symbol with the desired stroke number and order. The approach can work with a single *perfect* template for each class, without the need of a training phase to extract features or to select multiple templates. Being based on subgraph matching, the recognition can be performed on partially drawn symbols, i.e., when only a part of the primitives composing the symbol is available. An innovation of the presented approach is the use of a single spatial descriptor to represent relations between symbol components. The descriptor is an adaptation of the *shape context* defined in [13] and its use makes the approach free from the identification of the type of the *primitives* and from the check of fuzzy relations. The symbol matching is performed through an approximate graph matching procedure which incrementally produces new results as soon as more input strokes are available.

Different sets of symbols have been used to test our approach: a large set of symbols used to evaluate a previous method [8] and two differently sized sets of hand-drawn symbols extracted from the real domain of *Military Course of Action* diagrams [14]. The symbols in the former set have also been artificially perturbed to test the invariance of the approach with respect to scale and its tolerance to random drawing errors. The results show that the proposed approach can recognize a reasonably high percentage of symbols even with a small number of available primitives. Furthermore, improvements in the recognition rate of partially drawn symbols are obtained against the existing approaches.

* Corresponding author. Tel.: +39 089963390; fax: +39 089796438.
E-mail addresses: gencos@unisa.it (G. Costagliola), matderosa@unisa.it (M. De Rosa), vfuccella@unisa.it (V. Fuccella).
URL: http://weblab.di.unisa.it.

0097-8493/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.12.003

The rest of the paper is organized as follows: the next section contains a brief survey on the main approaches for hand-drawn symbol recognition; in Section 3 we describe the approach for the recognition of partially drawn symbols; Section 4 is devoted to outline the design of the interactive system for symbol autocompletion; Section 5 presents the evaluation of the performance of our approach in comparison to those of existing approaches; lastly, some final remarks and a brief discussion on future work conclude the paper.

2. Related work

Autocompletion is a functionality which involves the program in outputting the result desired by the user without the user actually entering the input data completely. It is commonly used with textual input and studies in the context of text autocompletion have already been carried out. For instance, it has proven effective or appreciated by the users in various text-based applications, such as text entry on mobile devices [1], search engine interfaces (e.g., Google Instant [2]), source code editors [3], database query tools [4], and so on.

In the present paper, we focus on autocompletion of graphical symbols. To achieve autocompletion, it is necessary that the recognizer is able to recognize partially drawn symbols. Only a few methods [6–9] have been introduced supporting this feature. In general, symbol recognition is a classification process in which the unknown input symbol is compared to a set of templates in order to find the best matching class. Most approaches require a time-consuming training phase in order to correctly define the characteristics of each class of symbols. Furthermore, the invariance of the recognition with respect to scale, stroke number and order are desirable characteristics. The invariance with respect to rotation and not uniform scale could also be required when necessary.

While the earliest recognizers [15,16] were only able to recognize unistroke symbols, several specialized methods have been recently proposed for multi-stroke hand-drawn symbol recognition. According to a widely accepted taxonomy [17,7,18] the methods are classified into two main categories: *structural* and *statistical*.

In *structural* methods, the matching is performed by finding a correspondence between the structures, such as graphs [12,19,20] or trees [7], representing the input and the template symbols. The methods based on graph matching usually represent symbols through *Attributed Relational Graphs* (ARG). Such a representation gives a structural description of the symbol [12]: the nodes in the graph are associated to the *primitives* composing the symbol, while the edges are associated to spatial relations between the *primitives*. The relations are often based on the presence of conditions such as intersections, parallelism, etc. Furthermore many approaches require the identification of the type of the *primitives* (line, arc, ellipse, etc.) composing the symbol. Due to the imprecise nature of *sketchy* symbols, the detection of the above characteristics is far from being precise and tolerance thresholds must be set, e.g., to distinguish a line from an arc or to check parallelism etc. In most cases, the user strokes are pre-processed in order to smooth them and to extract the sequence of *primitives* from them. Due to the high computational complexity, approximate algorithms for structural matching are often used, as the approximate graph matching algorithms presented in [20].

Statistical methods offer the advantage of avoiding the complex pre-processing phase in which the primitives are extracted. In most methods [21–23] a given number of features are extracted from the pixels of the unknown symbol and compared to those of the models. The best match is chosen through a statistical classifier. While techniques as *Zernike moment descriptors* [24] enable a very natural drawing style and support the invariance with respect to many types of transformations, tools as *Hidden Markov Models* (*HMM*) [25] can only be used when a fixed stroke order is established. Other recognizers, such as [26], exploit common classifiers in image-based matching, such as *Hausdorff* distance (and an ad-hoc defined variant of it), *Tanimoto* and *Yule* coefficients. The *shape context* itself has already been brought in a sketch recognition system [27] to represent parts of a symbol. Here, instead, a variation of it is used to represent spatial relations between symbol primitives. The approach presented in [28], called $*P*, is an extension to the recognition of multi-stroke symbols of the $*1* approach proposed in [29]. It preserves the *minimalism* of its predecessor and relies on some of its unistroke recognition functionalities, even though it treats the symbols as point clouds.

Symbol recognition can be a functionality of general-purpose frameworks for sketch recognition [30–32]. *SketchREAD* [30] and *AgentSketch* [31] exploit the knowledge about the domain context for disambiguating the symbols recognized at a lower level. The former uses a structural description of the domain symbols, through the *LADDER* language [33], as a combination of lower level primitives meeting certain geometric constraints. *AgentSketch* [31] exploits an agent-based system for interpreting the sketched symbols. *CALI* [32] exploits a naive Bayesian classifier to recognize geometric shapes. In [34] a graph-based algorithm for recognizing multi-stroke primitives in complex diagrams is presented. The presented algorithm, based on *Paleosketch*, does not require any special drawing constraint to the user.

Only a few methods have been introduced which are able to assist the user in the completion of multi-stroke hand-drawn symbols. Some of them, such as *OctoPocus*, [35], *SimpleFlow* [36] and *GestureCommander* [37], only work for unistroke symbol completion: by exploiting different recognition and feedback techniques, they provide the user with a visual feedback on the recognition while the gesture is still being performed. Among those working for multi-stroke symbols, a grammar-based technique is presented in [9]. In particular, once a partial input has been processed, the parser is able to propose to the user a set of final acceptance states (valid symbols) that have as subshapes the current intermediate state. In [8] a *Spatial Relation Graph* (SRG) and its partial matching method are proposed for online composite graphics representation and recognition. The SRG structure is a variation of ARG in which an edge connects two nodes only if a spatial relation (interconnection, tangency, intersection, parallelism and concentricity) is present between the primitives associated to the nodes. A *Spatial Division Tree* (SDT) has been used in [7]. In this representation, a node in the tree contains a set of strokes. Furthermore, intersection relations among strokes are codified through links among nodes. The approach described in [6] is based on clustering. In order to assign a (possibly partial) symbol to a cluster, the set of features described in [22] is extracted from it. The features are extracted on the partially drawn symbols used in the training data. Hence, the approach relies on the observation that people do tend to prefer certain stroke drawing orderings over others. Other, domain-specific, systems supporting symbol autocompletion have been described in literature. For instance, [38] describes a system for the recognition of a set of 485 symbols from Course of Action Diagrams [14], which also supports autocompletion.

The first three of the above cited methods show poor performance with partially drawn symbols when only a few primitives are available. As a consequence, they might not lend themselves well for the realization of an interactive system for autocompletion. The one described in [6], instead, is not completely invariant with respect to stroke order, but relies on users' preferred order. The method introduced in this paper has been tested on a large set