Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 536192, 12 pages
http://dx.doi.org/10.1155/2015/536192

Research Article

An Efficient Primitive-Based Method to Recognize Online Sketched Symbols with Autocompletion

Wei Deng,[1] Lingda Wu,[1] Yougen Zhang,[2] and Chao Yang[1]

[1] *Equipment Academy, Beijing 101416, China*
[2] *Department of Information Systems, Academy of National Defense Information, Wuhan 430010, China*

Correspondence should be addressed to Wei Deng; dengweicn@126.com

Received 21 January 2015; Revised 27 March 2015; Accepted 9 April 2015

Academic Editor: Nazrul Islam

We present a new structural method of sketched symbol recognition, which aims to recognize a hand-drawn symbol before it is fully completed. It is invariant to scale, stroke number, and order. We also present two novel descriptors to represent the spatial distribution between two primitives. One is invariant to rotation and the other is not. Then a symbol is represented as a set of descriptors. The distance between the input symbol and the template one is calculated based on the assignment problem. Moreover, a fast nearest neighbor (NN) search algorithm is proposed for recognition. The method achieves a satisfactory recognition rate in real time.

1. Introduction

Sketch recognition is widely used in pen-based interaction, especially with the increasing popularity of devices with touch screens. It is a natural and efficient means of capturing information by automatically interpreting hand-drawn sketches and it can be the import part of the early design process, where it helps people explore rough ideas and solutions in an informal environment. Sketch recognition has been successfully applied in education [1, 2], engineering [2, 3], design [4], and so on.

Sketch recognition refers to recognition of predefined symbols or free-form drawings (e.g., an unconstrained circuit drawing); in the latter case, the recognition task is generally preceded by segmentation in order to locate individual symbols [5]. This paper focuses on the recognition of hand-drawn isolated symbols. However, it is a difficult problem due to the inherent imprecision and ambiguity of a freehand drawing [6]. Many challenges remain in terms of intraclass compactness and interclass separation due to the variability of sketching, because it is likely that different people have different drawing styles, such as the stroke order, stroke number, and nonuniform scaling, as well as complex local shifts. Moreover, the styles of the same individual may differ even at different times.

A practical application system should place few drawing constraints on users. So the invariance properties to scale, stroke number, and stroke order are desirable characteristics. In many applications, a graphical symbol can be drawn towards different orientations; hence, the recognition algorithm should also be rotation-free when necessary. A similar research is handwriting recognition, such as handwritten digit and Chinese character recognition, which has many effective algorithms.

The term autocompletion refers to predicting the sketched symbols before the drawing is completed [5]. It can be advantageous for the users in several applications [7]. Autocompletion during sketching is desirable since it can be used to facilitate sketching by offering possible user-intended symbol classes and to reduce user-originated errors by providing feedback immediately after receiving a new input stroke.

However, autocompletion is a more difficult problem, due to classifying with the partial information before the drawing is completed. Firstly, a hand-drawn symbol is ambiguous if it appears as a subsymbol of more than one symbol class. Secondly, the partially drawn symbol and the fully completed one differ in visual appearance. Finally, the similarity of them is changed with the process of sketching.

This paper focuses on the recognition of hand-drawn isolated symbols before they are fully completed and presents a structural framework to recognize online sketched symbols. The key contributions of our method are listed as follows:

(1) We present a framework to recognize sketched symbols with autocompletion. It has inherent invariance to stroke number and order. It can work with a single (or possibly more) representative template for each symbol class. So it provides fine extensibility to new shapes. It obtains high recognition accuracy in real time even when the hand-drawn symbols are highly incomplete.

(2) We also present two novel descriptors, to represent the spatial distribution between two primitives. One is called DAR (directional adaptive region descriptor), which is not invariant to rotation. The other is called DZM (directional Zernike moment descriptor), which inherits the rotation invariance from the traditional Zernike moments (ZM) descriptor. They are both statistical descriptors rather than topological relations (e.g., intersections, parallelism, etc.). So there is no need to recognize primitives. The approach is independent of the primitive types.

(3) A simple and fast NN search algorithm is proposed for recognition. It can reduce the runtime of structural matching. And it requires no burdensome mathematical procedures and complex data structures.

The rest of the paper is organized as follows. Section 2 contains a brief survey on the main approaches for sketched symbol recognition. In Section 3 we describe the proposed method for the recognition of partially drawn symbols. Section 4 evaluates the performance of our method. Lastly, we conclude the paper with some final remarks and a brief discussion on future work.

2. Related Work

According to a widely accepted taxonomy, sketched symbol recognition methods are classified into two main categories: structural and statistical [6].

Structural methods focus on building structural shape descriptions. The basic step is stroke segmentation using temporal and spatial features. Then a sketched symbol can be represented as a tree or graph, and the similarity between two sketched symbols can be calculated by structural matching. Hammond and Davis [8] developed a hierarchical language to describe how diagrams are drawn, displayed, and edited. Then they used the language to perform automatic symbol recognition. Attributed relational graph (ARG) is an excellent structural model to describe both geometry and topology of a symbol [9], and it is insensitive to orientation, scaling, and stroke order. The advantage of structural methods is distinguishing similar shapes. But the disadvantage is their sensitivity to the results of stroke segmentation. Furthermore, many approaches require the identification of the primitive type (e.g., line, arc, ellipse, etc.) and the spatial (topological) relations between two primitives (e.g., intersections, parallelism, etc.). And due to the high computational complexity, approximate algorithms for structural matching are often used, such as the approximate graph matching algorithms presented in [9].

Statistical approaches look at the visual appearance of symbols, without stroke segmentation and primitive recognition. Mostly, a number of features are extracted from the pixels of the unknown symbol, followed by a statistical classifier. Some shape descriptors, such as Zernike moment [10, 11] and shape context [12], can be used to represent a sketched symbol. Kara and Stahovich [13] proposed an image-based recognizer, using four template classifiers. In their method, polar coordinates were used to achieve rotation invariance. Ouyang and Davis [14] proposed a visual approach to sketched symbol recognition. It used a set of visual features that captured online stroke properties like orientation and endpoint location. Almazán et al. [15] described a framework to learn a model of shape variability based on the Active Appearance Model (AAM) and proposed two types of BSM (Blurred Shape Model [16]) descriptors. Willems et al. [17] explored a large number of online features, which were sorted in three feature sets due to different levels of details. Delaye and Anquetil [18] presented a set of 49 features, called HBF49, for the representation of hand-drawn symbols. And HBF49 can be used as a reference to evaluate a symbol recognition system. The advantage of statistical methods is the high robustness to noise and different drawing styles, such as stroke order and direction. It avoids the complex phase of primitive extraction.

Furthermore, most of current researches, such as the above cited methods in [8–18], concern the fully completed symbols. To date, only a few systems have been introduced supporting autocompletion [5, 7, 19–21]. In these works, a symbol is usually represented as a spatial relation graph (SRG) [20] or a spatial division tree (SDT) [19], and then the similarity or distance between the input symbol (probably incomplete) and the template can be calculated based on these representations. In [20] a syntactic approach is presented. Costagliola et al. proposed a graph-based method [6, 7]. It uses the ARG to represent a symbol, and the recognition is based on subgraph isomorphism. But it is not invariant to rotation. Unlike these structural methods, Tirkaz et al. [5] proposed an image-based method, whose framework was fully probabilistic. And it also has no inherent invariance to rotation. But the approach relies on the observation that people do tend to prefer certain stroke drawing orderings over others. Hence, it is not completely invariant to stroke order but relies on user's preferred order in the training data.

3. Our Approach

We designed our approach primarily using the primitive-based matching. An overview of the recognition process is shown in Figure 1. In particular, we mainly show how to represent the symbols and how to calculate the distance between the hand-drawn symbol and a template.

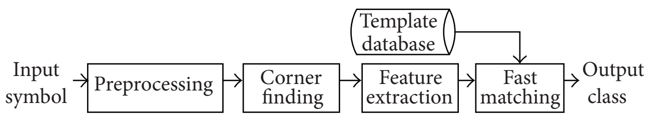


Figure 1: The flowchart of our approach.

This method is organized as follows. Firstly, all strokes of a symbol are preprocessed (Section 3.1) and segmented into primitives (Section 3.2). Then the descriptor is extracted for every biprimitive (Section 3.3) (a biprimitive means a pair of primitives). In this step, we propose two novel descriptors. Next, the distance (equivalently, dissimilarity) between the unknown symbol and a template is calculated by a bipartite graph matching (equivalently, optimal assignment) procedure (Section 3.4). Finally, a fast NN search algorithm is proposed for symbol recognition (Section 3.5).

3.1. Preprocessing. The preprocessing of the input sketch directly facilitates pattern description and affects the quality of description. Its tasks include noise elimination, shape scaling, and resampling. These operations are simple to perform and guarantee a better stability of extracted features, for any type of input sketch.

The noise in input trajectories is due to erratic hand motions and the inaccuracy of digitization. The noise reduction techniques include smoothing, filtering, and wild point correction. As the quality of input devices steadily advances, trajectory noise becomes less influential and simple smoothing operations will suffice.

To achieve invariance under scaling and translation, the coordinates of stroke points are simply shifted and linear scaled such that all points are enclosed in a standard box. In our experiments we set $x, y \in [0, 100]$. It means translating maximal dimension of a symbol to 100 with aspect ratio preserved [22].

Since online strokes are typically sampled at a constant temporal frequency, the distance between neighboring points varies based on the pen speed. This produces more sampled points where the pen is typically slower. In order to make feature extraction more reliable, we resample each stroke at a constant spatial distance. In our experiments the resampling interval is set to 1.0.

3.2. Corner Finding. Our method works with primitives and not directly with strokes, so corner finding is an essential step in order to extract the primitives, as well as most structural methods.

Primitives are regarded as simple graphical components, such as lines, arcs, and ellipse. The objective of corner finding is decomposing a stroke into primitives. There are many existing methods for corner finding, for example, IStraw [23], MergeCF [24], ClassSeg [25], SpeedSeg [26], QPBDP [27], DPFrag [28], and RankFrag [6, 7]. In fact, it is another well-studied problem in sketch-based interfaces. However, our main work is to represent the symbols and calculate the distance between two symbols after corner finding. We use the existing corner finding algorithm in [7]. It is the revisited version of Ouyang and Davis's work in [29]. It has been reported with satisfactory performance. Instead of immediately trying to decide which points are corners, it repeatedly removes the point that is least likely to be a corner. The details of the method are available in [7]. In a brief review, the method works as follows.

(1) Initially, a number of equally spaced points are extracted from the stroke and are all added to a list of possible corners.

(2) Then the points which are least likely to be corners are iteratively removed from the list. The likelihood is evaluated through a cost function. For each point p_i in the list, a cost value is computed as

$$\text{Cost}(p_i) = \sqrt{\text{mse}(S_i; p_{i-1}, p_{i+1})} \times \text{dist}(p_i; p_{i-1}, p_{i+1}), \quad (1)$$

where S_i is the subset of points in the resampled stroke between point p_{i-1} and point p_{i+1} and $\text{mse}(S_i; p_{i-1}, p_{i+1})$ is the mean squared error between the set S_i and the line segment formed by (p_{i-1}, p_{i+1}). The term $\text{dist}(p_i; p_{i-1}, p_{i+1})$ is the minimum distance between p_i and the line segment formed by (p_{i-1}, p_{i+1}).

For the point p with the smallest cost, it is iteratively removed from the list, and the cost in (1) is updated. At each iteration, the decision to remove a point is taken on a binary classifier, which is previously trained with data. The data include ten features, extracted from the strokes, points, or stroke fitting errors. Six of the features are described in [29], while the rest are advanced in [7]. The features are shown in Table 1.

(3) During classification, if the classifier decides that p is not a corner, it removes the vertex and continues to the next elimination candidate. Otherwise, if it decides that it is a corner, the process stops and all remaining vertices are returned as corners.

3.3. Feature Extraction. After corner finding, a sketched symbol is represented by a set of biprimitives. The main task of this step is to calculate the biprimitive descriptor. It is used to describe how two primitives are spatially related within a symbol.

We propose two descriptors. One is called DAR (directional adaptive region descriptor). It is inspired by the directional features, whose effectiveness in representing a character or symbol has been demonstrated in both handwritten character and sketch recognition [30]. But it is not invariant to rotation. The other is called DZM (directional Zernike moment descriptor), which incorporates local direction information into the ZMs. It inherits the rotation invariance from the traditional ZMs.

After feature extraction, each symbol is represented through a set of proposed descriptors where each element is associated with a pair of primitives.

3.3.1. DAR Descriptor. Firstly, for each resampled point p_i, there is a local line defined by two consecutive points (p_i, p_{i+1}). Each local line is decomposed into components in standard directions. We employ four chain code directions.

Table 1: List of features for corner finding.

Feature	Description	Reference
Cost	The cost of removing the vertex, from (1).	[29]
Diagonal	The diagonal length of the stroke's bounding box.	
Ink density	The length of the stroke divided by the diagonal length.	
Max distance	The distance to the farther of its two neighbors (p_{i-1} or p_{i+1}) normalized by the distance between the two neighbors.	
Min distance	The distance to the nearer of its two neighbors normalized by the distance between the two.	
Sum distance	The sum of the distances to the two neighbors normalized by the distance between the two.	
EllipseFit	A function calculated on the whole stroke, returning a real value between 0 and 1. The higher this value is, the more the stroke resembles an ellipse (or a circle).	[7]
PolyFit	A function calculated for the candidate corner point c_i on the substroke whose endpoints are the previous and the next candidate corner (c_{i-1} and c_{i+1}, resp.), returning a real value between 0 and 1. The higher this value is, the more the stroke resembles a polyline composed of the two segments $c_{i-1}c_i$ and c_ic_{i+1}.	
Angle	The magnitude of the angle with vertex in the candidate corner point, calculated with respect to the previous and the next sampled points of the stroke.	
SeqNumber	The sequence number of the iteration of the removal process.	

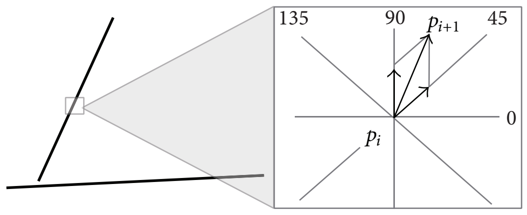


Figure 2: Directional decomposition of a local line segment.

The major advantage is the independence of local stroke direction; for example, the decomposition of (p_i, p_{i+1}) is the same as that of (p_{i+1}, p_i). If a local line lies between two neighboring standard directions, it is decomposed into two components in the two standard directions, as shown in Figure 2. Thus the local line is assigned to four directional planes, corresponding to four chain code directions. The length of the local line component is assigned to the corresponding pixel in the plane. An example of the whole process is shown in Figure 3.

Secondly, each directional plane is partitioned into several uniform zones. In our method, there are two kinds of partition for the high accuracy, such as Figure 3. In partition 1, the biprimitive region is partitioned into four subregions by two dashed lines, which both pass through the centroid (marked as a red point in the figure) of the two primitives. And the directions of the dashed lines are 0 and 90 degrees. Meanwhile in partition 2, the directions of the two dashed lines are 45 and 135 degrees, respectively.

Lastly, in each subregion, we accumulate the pixels. So for a biprimitive, we get two $4 \times 4 = 16$ dimensional vectors, named v_1 and v_2, corresponding to the two kinds of partition, respectively.

Given two biprimitives, a and b, denote their DARs as $(v_1(a), v_2(a))$ and $(v_1(b), v_2(b))$, respectively. Then we define the DAR distance $\text{dist}_{\text{DAR}}(a, b)$ as follows:

$$\text{dist}_{\text{DAR}}(a, b) = \min\left(\left\|v_1(a) - v_1(b)\right\|, \left\|v_2(a) - v_2(b)\right\|\right), \quad (2)$$

where $\|\cdot\|$ means Euclidean distance.

3.3.2. DZM Descriptor. DZM incorporates local direction information into the ZMs which represent only the spatial distribution of sample points. A shape is decomposed into several component channels and the DZM descriptor consists of the ZMs from all channels. Figure 4 shows an example.

Firstly, similar to DAR, for each resampled point p_i, there is a local line defined by two consecutive points (p_{i-1}, p_{i+1}). The local directional angle θ_i for p_i is defined as the angle between lines (p_i, o) and (p_{i-1}, p_{i+1}), where o is centroid (marked as a red point in Figure 4). Obviously, invariance to rotation is intrinsic to the angle θ_i.

Secondly, each local angle is decomposed into components in D uniformly spaced standard angles, such as $\{0, (1/D)\pi, (2/D)\pi, \ldots, \pi\}$; each of them would also be referred to as a channel later. If a local angle value lies between two standard angles, it is decomposed into two components in the two standard angles. It is similar to the process of directional decomposition in DAR, while the standard directions (angles) are different. Thus the biprimitive is decomposed into D directional planes (subimages), corresponding to D standard angles. The membership degree (component length) of θ_i is assigned to the corresponding pixel in the plane. And the planes are invariant to rotation.

Finally, extract a set of K ZMs on each plane (K is the order of ZMs). Eventually, the DZM descriptor consists of

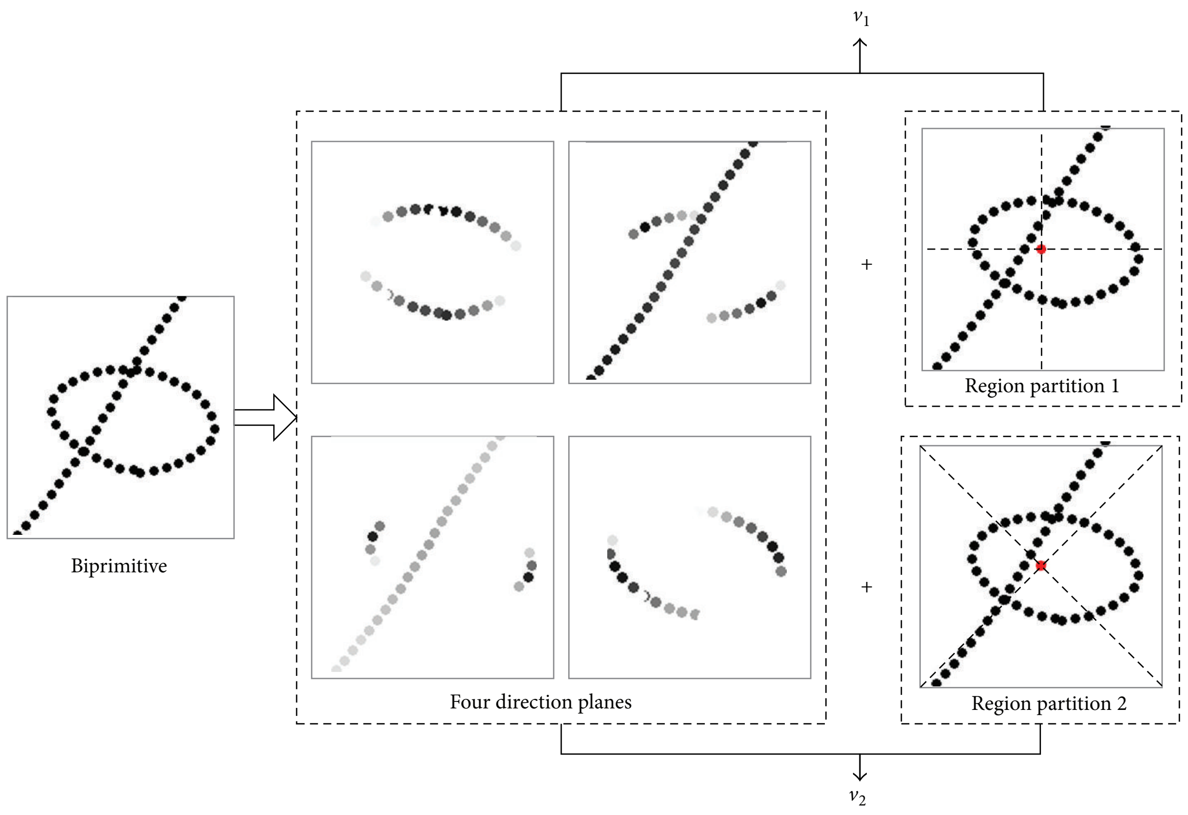
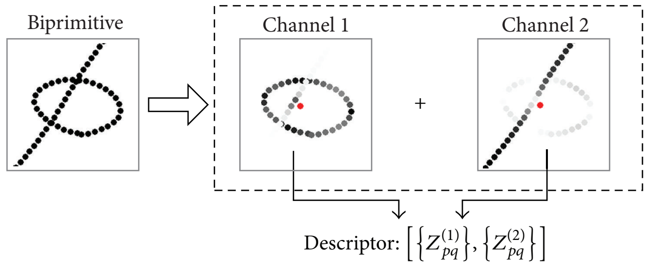


FIGURE 3: An example of DAR descriptor for a biprimitive.



Biprimitive
Channel 1
Channel 2
Descriptor: $\left[\left\{Z_{pq}^{(1)}\right\},\left\{Z_{pq}^{(2)}\right\}\right]$

FIGURE 4: An example of DZM descriptor for a biprimitive (two channels).

the $D \times K$ ZMs from all channels. In experiments we find that the recognition accuracy is well when D is set to 2 and K is set to 8. The DZM descriptor has better performance in symbol recognition than traditional ZM [11].

Given two biprimitives, a and b, denote their DZMs as $z(a)$ and $z(b)$, respectively. Then we define the DZM distance $\text{dist}_{\text{DZM}}(a,b)$ as follows:

$$\text{dist}_{\text{DZM}}(a,b) = \|z(a) - z(b)\|. \quad (3)$$

3.4. Symbol Matching. To facilitate the presentation, the input sketched symbol is recorded as U (meaning unknown, probably incomplete), and the template symbol is denoted as T. The purpose of symbol matching is to compute the distance between U and T.

After feature extraction, a symbol is represented as a set of descriptors where each element is associated with a biprimitive. So U and T can be denoted as

$$U = \left\{b_i^U\right\}, \quad i = 1, 2, \ldots, \left|b^U\right|;$$
$$T = \left\{b_j^T\right\}, \quad j = 1, 2, \ldots, \left|b^T\right|, \quad (4)$$

where b_i^U is the ith biprimitive of U and $|b^U|$ is the biprimitive number, and b_j^T is the jth biprimitive of T and $|b^T|$ is the biprimitive number of T.

Let $\mathbf{C}$ denote the *matching cost matrix* between U and T. Each element $C_{i,j}$ means the matching cost between b_i^U and b_j^T. Consider

$$\mathbf{C} = \left[C_{i,j}\right] = \left[C\left(b_i^U, b_j^T\right)\right]. \quad (5)$$

If the biprimitive numbers of U and T are not equal, the cost matrix can be made square by adding "dummy" biprimitives to the smaller set. If $|b^U| < |b^T|$, add dummy biprimitives to U (denoted as b_0^U). Else, if $|b^U| > |b^T|$, add another kind of dummy biprimitives to T (denoted as b_0^T). In order to recognize the incompletely hand-drawn symbols, the matching cost is defined as

$$C\left(b_i^U, b_j^T\right) = \begin{cases} 0, & i = 0 \\ \lambda \cdot \text{length}\left(b_i^U\right), & j = 0 \\ \text{dist}\left(b_i^U, b_j^T\right), & \text{else}, \end{cases} \tag{6}$$

where the term length( ) is the total length of the biprimitive and the variable λ is an empirical parameter. And dist( ) is DAR distance in (2) or DZM distance in (3). It depends on whether rotation invariance is required by the user or not. In our experiments λ can be set to 0.7. The matching cost of $C(b_i^U, b_0^T)$ is set to λ length(b_i^U) for "penalty," because if $|b^U| > |b^T|$, it is likely that U and T belong to different symbol classes. And the longer the biprimitive b_i^U is, the more the penalty is.

Given the matching cost matrix $\mathbf{C}$ between U and T, we want to minimize the total cost of matching

$$H(\pi) = \sum_i C_{i,\pi(i)} \tag{7}$$

subject to the constraint that the matching is one-to-one; that is, π is a permutation. In this case, a biprimitive will be matched to a dummy whenever there is no real match. This is an instance of the square assignment (or weighted bipartite matching) problem, which can be solved in $O(N^3)$ time using Hungarian algorithm. The minimum H in (7) is the distance between U and T.

3.5. Fast NN Search Algorithm for Symbol Recognition. By calculating the distances between U and each of the templates, the NN techniques can be used for symbol recognition. However, because the pattern is not described as a vector, the traditional strategies to speed up, such as KD-trees and M-trees, cannot be used. So the expensive cost of computation is a key issue which needed to be addressed. Inspired by the work in [31], we propose a simple and fast NN search algorithm for our framework of sketch recognition. The main idea is to reject a lot of candidates based on the lower bound of distances efficiently.

Denote D and d as

$$D = \sum_i \min_j \left(C_{i,j}\right), \qquad d = \min(H). \tag{8}$$

It means D is the sum of minimums in each row of the matching cost matrix $\mathbf{C}$ and d is the real distance between U and T using Hungarian algorithm. Obviously, the calculation of D is simpler and faster than d, and $D \leq d$ holds. So D can be seen as the lower bound of d. The fast NN search algorithm is described below with two steps.

Step 1. Sequentially scanning all the templates, the lower bound of the distance between U and the ith template (T_i, $i = 1, 2, \ldots$) is calculated (denoted as D_i). Meanwhile, the template with the minimized D_i is recorded as T_k where k is the subscript of the template. T_k is regarded as the initial candidate of nearest neighbour. Then the real distance between U and T_k is computed using Hungarian algorithm in (7) and regarded as the initial probably minimum distance (denoted as $d_{\min}$) in all templates.

Step 2. Scan each template again sequentially to compare its D_i with the probably minimum distance $d_{\min}$. If $D_i > d_{\min}$ holds, then $d_i \geq D_i > d_{\min}$. It means T_i is not the nearest neighbor and could be rejected immediately. Otherwise, the real distance d_i between T_i and U is computed using Hungarian algorithm. Then, if $d_i < d_{\min}$ holds, the candidate of nearest neighbor is updated as $k = i$ and $d_{\min} = d_i$; otherwise T_i would be rejected.

After scanning the templates twice in the above two steps, the final T_k is the nearest neighbor of U. The advantage of the search algorithm is that it reports the exact nearest neighbor, not an approximate one, and requires very simple implementing with no sophisticated data structures.

4. Evaluation and Discussion

The proposed method is tested on two datasets which have already been introduced in literatures. The symbols in COAD dataset [5] and COAD2 dataset [7] are two different subsets of the symbols used in domain of Military Course of Action Diagrams. In total the COAD dataset contains 620 samples from 20 classes of symbols drawn by eight users. Meanwhile the COAD2 dataset contains 4520 sketched symbols drawn by eight users, belonging to 113 classes. Some sketched samples of COAD are shown in Figure 5, and the template symbols of COAD2 are shown in Figure 6.

4.1. Accuracy of Recognition with Autocompletion. The accuracy in corner finding has been recorded in [7] as 99.65% for correct corners accuracy and 99.20% for all-or-nothing accuracy. Thus we mainly test the accuracy of sketch recognition with autocompletion.

Firstly, five perfect samples per class of symbols were chosen as the templates. Then the strokes of each original test symbol were reordered randomly in order to guarantee that the results were independent of stroke order. Additionally, when we use DZMs as biprimitive descriptors, the test symbols were rotated randomly to guarantee the rotation invariance. Next, for each symbol composed of n primitives, the recognizer was launched n times, each with the first $1, \ldots, n$ primitives, representing the symbol at different completion status. Furthermore, the top N recognition rate reports the percentage of times that the correctly matching template is in the top N positions of the candidate list. The results are shown in Figure 7. The recognition rate is calculated as a function of the number of primitives which have already been drawn.

4.2. Evaluation of the Proposed Descriptors. In the proposed method, a symbol is represented as a set of descriptors, where each element is associated with a biprimitive. Figure 8 shows

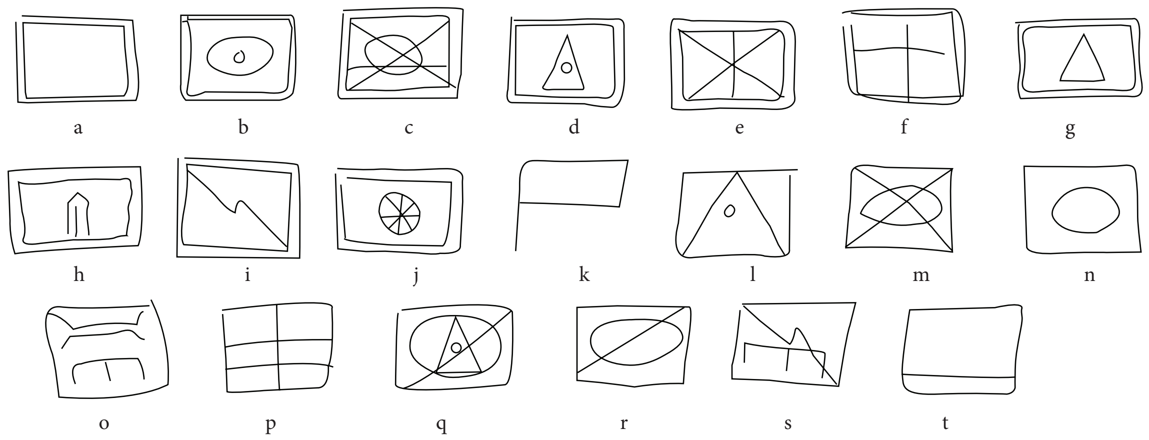


FIGURE 5: A sample symbol from each class in the COAD dataset [5].

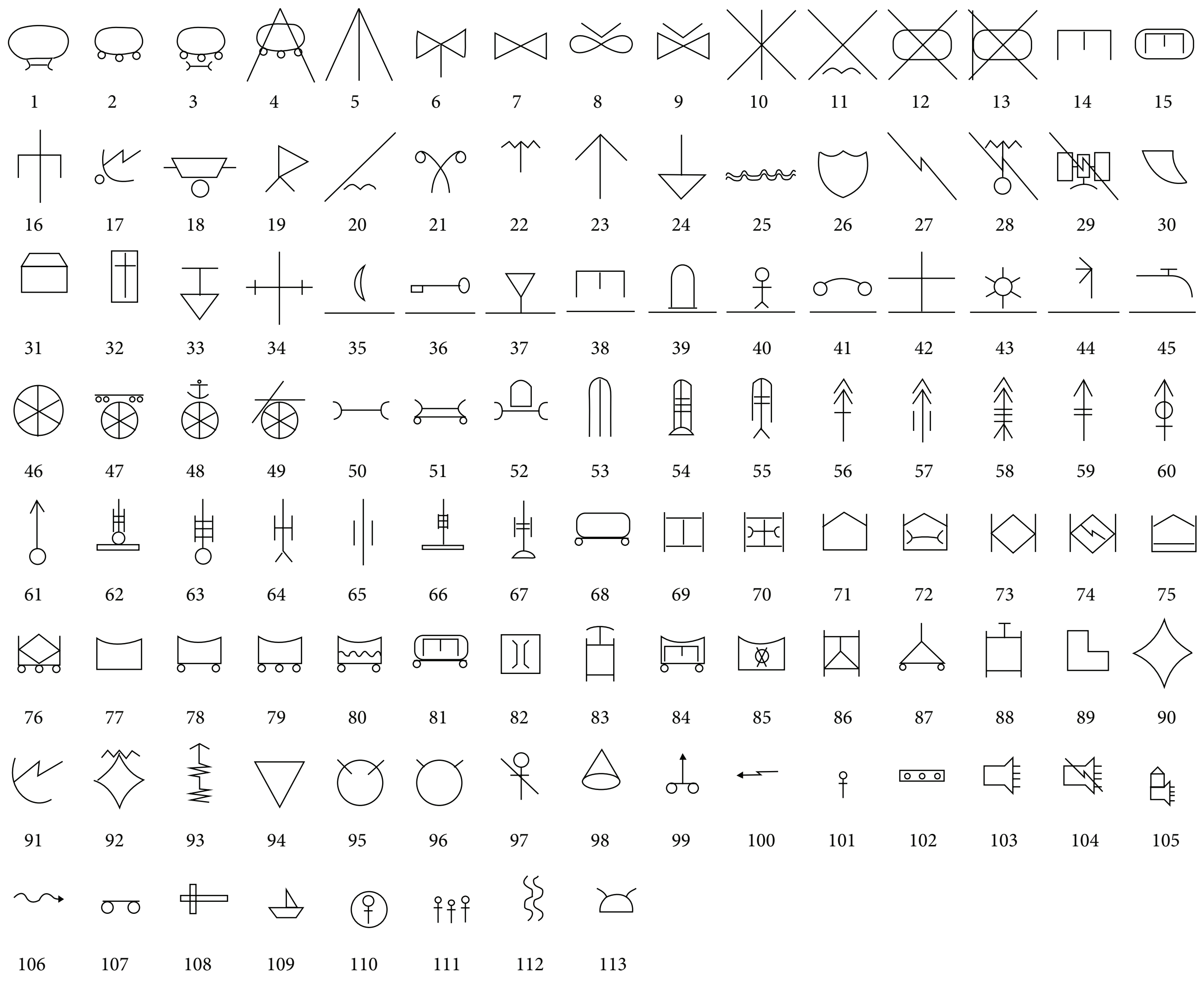
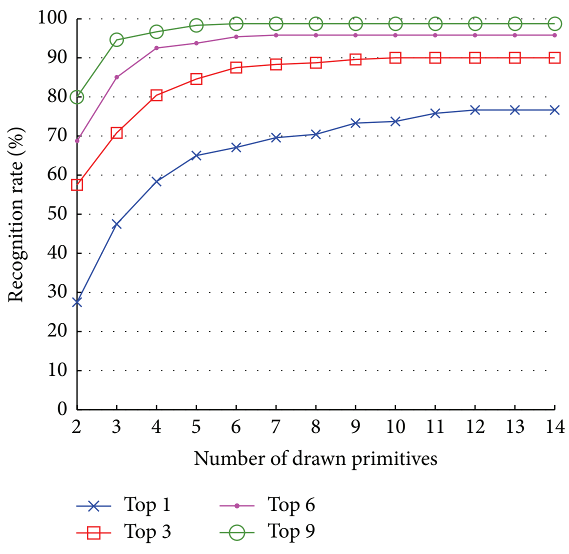
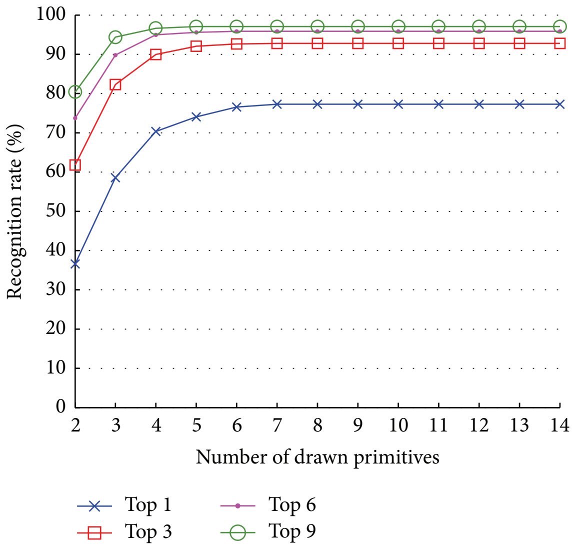


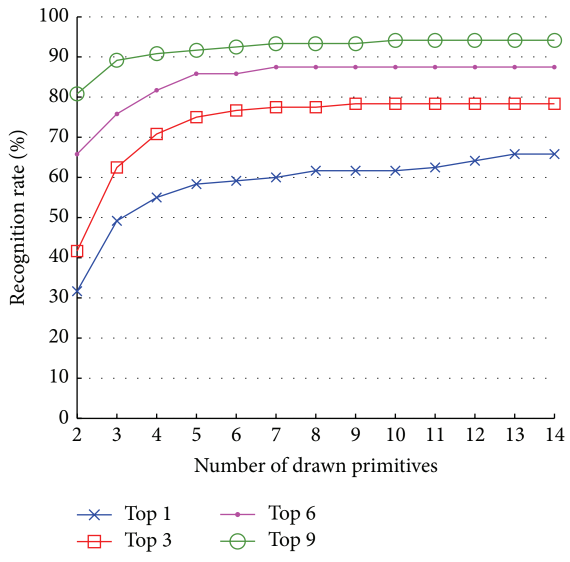
FIGURE 6: The 113 template symbols from the COAD2 dataset [7].



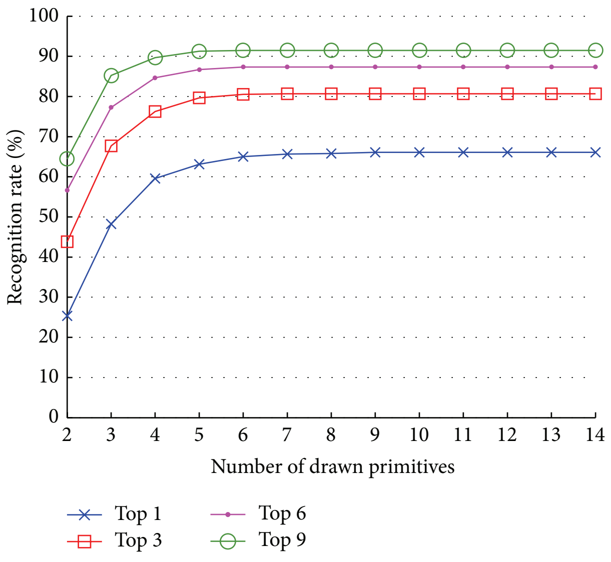
(a) Recognition using DAR in COAD dataset



(b) Recognition using DAR in COAD2 dataset



(c) Recognition using DZM in COAD dataset



(d) Recognition using DZM in COAD2 dataset

FIGURE 7: Recognition rate by the number of primitives drawn in two datasets.

an example. A sketched symbol is represented as six biprimitives. Obviously each sketched symbol (probably incomplete) belonging to this symbol class in Figure 8 will share one or more biprimitives in the figure. This idea is similar to the bag-of-features representation in the research of image retrieval [32].

In order to evaluate the proposed two descriptors quantitatively, we compared them with two other descriptors. One is called PSP (primitive spatial relation) presented in [7]. It is an adaptation of shape context descriptor defined in [33]. And PSP has no rotation invariance, as well as the proposed DAR. The other is the ZM descriptor, which is one of the best shape descriptors [34]. And ZM is invariant to rotation, as well as the proposed DZM.

Firstly, because the descriptors are calculated on biprimitives, we built two subdatasets. We chose 20 and 100 classes of biprimitives from COAD and COAD2 datasets randomly, respectively. They were used to evaluate the descriptors in different sizes of datasets. Figure 9 shows a set or subset of biprimitive samples.

Then the recognition rates were calculated based on 5-fold cross validation under the nearest neighbor rule. Figure 10 shows the results. The proposed descriptors have better recognition performances.

Besides, the PSP descriptor is an adaptation of shape context. So the PSP will be computed in $O(N^2)$ time. Meanwhile the proposed DAR captures the distribution of every point by its directional features. It can be solved in $O(N)$ time.

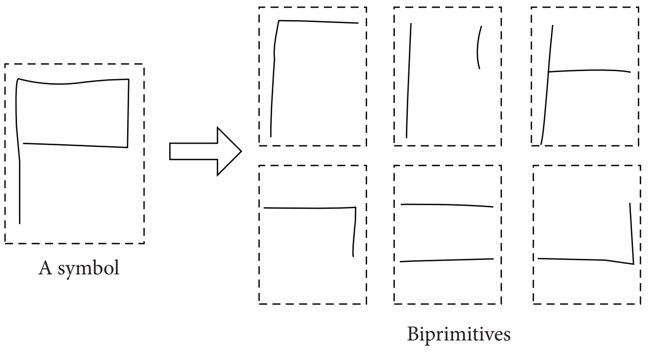


Figure 8: A sketched symbol is represented as a bag of biprimitives.

The DZM incorporates local directional information into the ZM. The computational cost of DZM is D times more than ZM, where D is the number of channels. In particular, ZM is the special version of DZM when $D = 1$. Although this leads to additional computational cost, more importantly the proposed DZM is more expressive and discriminative [11].

4.3. Comparison with Other Methods. Firstly, a summary about the related methods is briefly described. Then we compared our method with the state of the art in both recognition accuracy and response time.

(1) A Summary of the Methods for Autocompletion in References. Our method is free from the identification of the primitive types, unlike many other structural methods [19–21]. Table 2 briefly reviews the properties of the existing methods to recognize partially drawn symbols. This table also shows the comparison with other methods in the required knowledge for users. In the proposed method, the symbol matching just requires low-cost heuristic algorithm, which is simpler than others.

(2) Comparison with the ARG-Based Method in [7]. The ARG-based method presented in [7] employs the NP-complete subgraph isomorphism and gives an approximate solution. And it does not support rotation invariance. So we only compared it with our method using DAR descriptor. The results are shown in Figure 11. The performance of our recognizer (the solid lines) is better than the ARG-based method (the dashed lines), especially when the symbols are highly incomplete.

In addition, we also compared the response time of our approach with ARG-based method. The programming language was MATLAB and the CPU was Intel Core at 3.10 GHz. The average running time used in extracting DAR for biprimitive was 4.6 ms. The procedure of feature extraction can be proceeding incrementally with sketching. So the main proceeding time was in the NN search procedure (including the procedure of symbol matching). The response time in COAD2 dataset is shown in Figure 12. It is calculated as a function of the total number of templates. The proposed fast NN search algorithm makes our method nearly twice faster than the ARG-based method. And it is efficient to give real-time response for a dataset consisting of hundreds of symbols.

Table 2: The summary of methods for autocompletion.

Methods	Need to recognize primitive type?	Required knowledge
Image-based [5]	No	Statistical training and classifying
ARG [7]	No	Graph isomorphism
SDT [19]	Yes	Tree-based structural model
SRG [20]	Yes	Graph isomorphism
Grammar [21]	Yes	Syntactic recognition
Our method	No	Low-cost heuristic algorithm

Table 3: The comparison with the image-based method in COAD dataset.

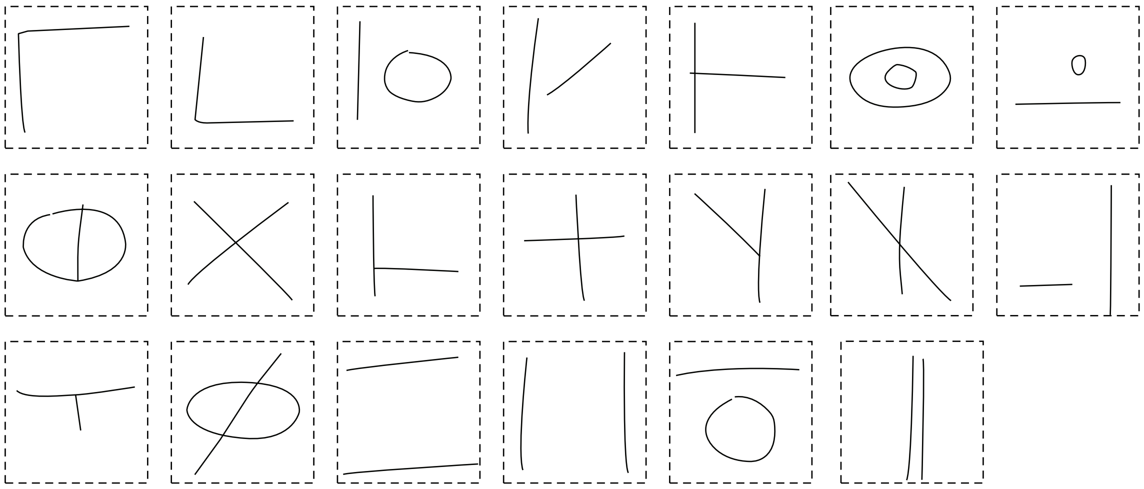
Methods	Symbols	Top 1 (%)	Top 3 (%)
Image-based [5]*	Partially drawn symbols	54.94	97.08
	Fully completed symbols	79.83	99.27
Proposed	Partially drawn symbols	83.57	98.90
	Fully completed symbols	75.60	85.53

*The accuracy of image-based method comes from [5] when confidence threshold is set to 0.

(3) Comparison with the Image-Based Method in [5]. Reference [5] proposed an image-based method to recognize sketched symbols with autocompletion. In fact, it adds partially drawn symbols into the training data and extracts the global statistical features of symbols. So it does not need to segment strokes into primitives. The main advantage of the image-based method is the robustness to the different drawing styles and noise.

However, there are some problems in the image-based method. Firstly, it clusters together the partial and full symbols based on their features. And it has two important parameters, the cluster number K and the confidence threshold C. The optimal parameters changed with the number of the symbol classes. So a lot of experiments are needed to train optimal parameters. Moreover, the accuracy of autocompletion relies on the number of partially drawn symbols in the training data. The autocompletion performance would fall when there are not enough partially drawn symbols. However, the training data are growing exponentially when the number of symbol classes grows [5]. So the image-based method is only tested in two small datasets in [5], which contain 20 and 14 classes of symbols, respectively. And if a new symbol class is added, the method should be trained again.

The recognition rates of the two methods are shown in Table 3. Although the accuracy for full completed symbols of the proposed method is lower than that of [5], our accuracy for partially drawn ones is better. The main reason is that, in COAD dataset, there are many symbols which are the subsymbols of other symbol classes. For instance, the symbol a is the subsymbol of b in Figure 5. So the symbol a is easily misrecognized as incomplete b. But in [5] the cluster procedure is beneficial to the recognition of fully completed symbols.


(a) A set of biprimitives in COAD dataset

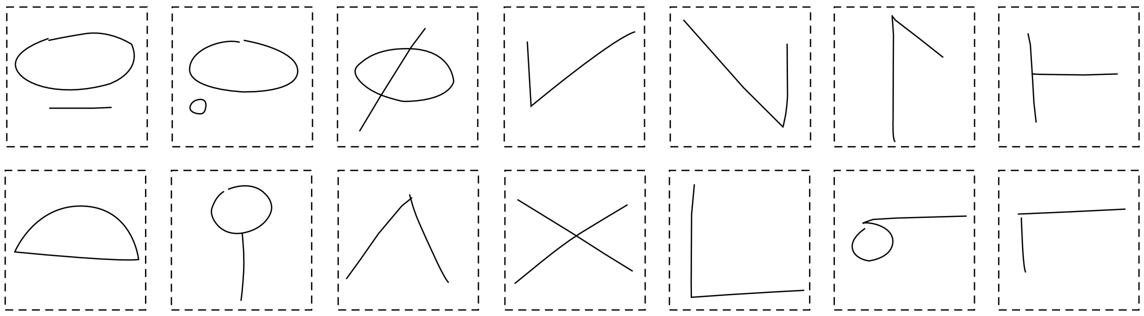
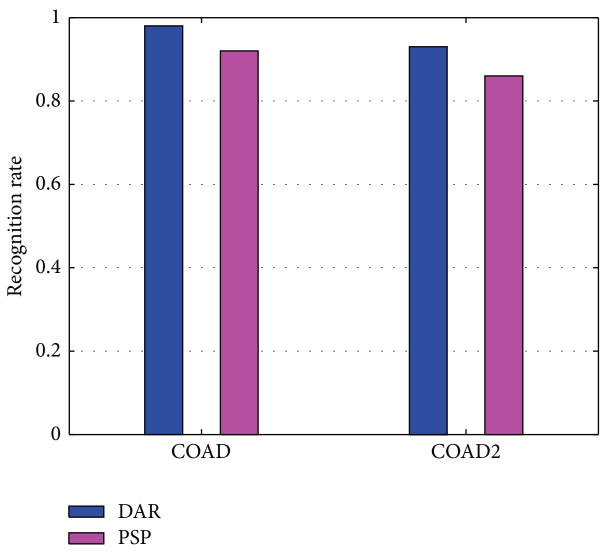
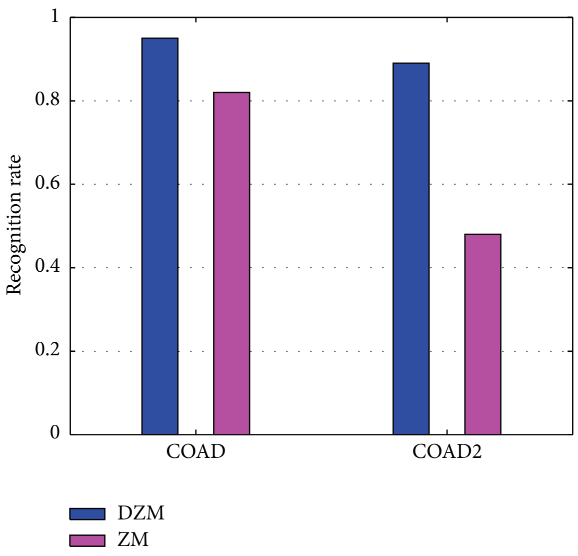

(b) A subset of biprimitives in COAD2 dataset

Figure 9: Examples of biprimitive samples.



(a) DAR and PSP without rotation invariance



(b) DZM and ZM with rotation invariance

Figure 10: Recognition rate of biprimitives using different descriptors.

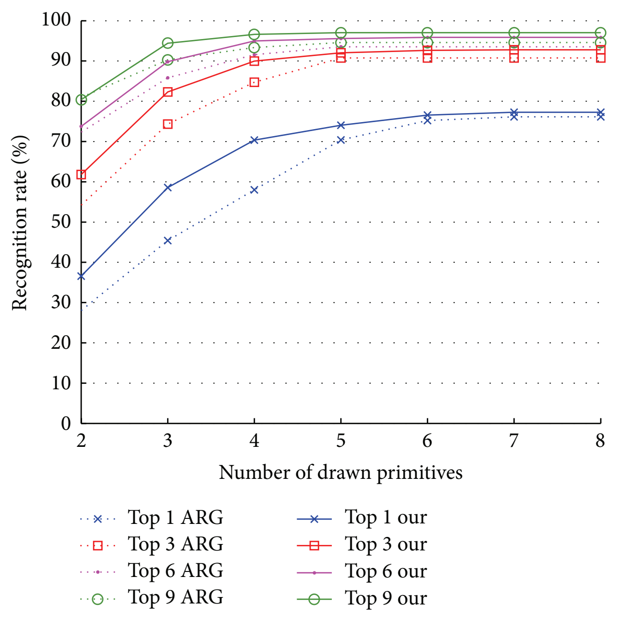


FIGURE 11: Comparing our method using DAR with the ARG-based method in COAD2 dataset.

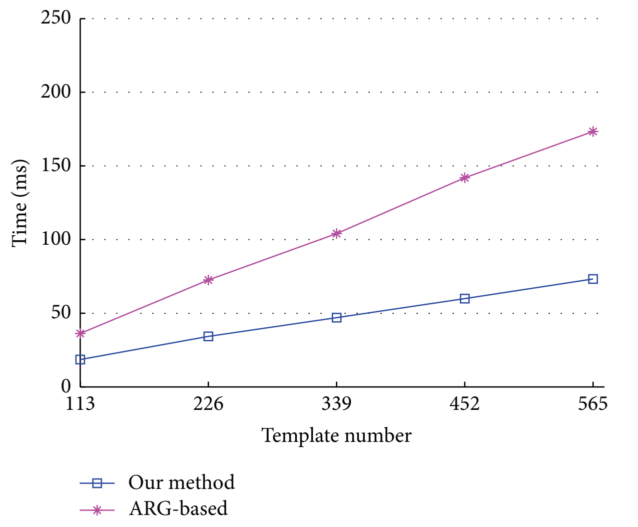


FIGURE 12: Response time of NN search in COAD2 dataset.

So the proposed method and the image-based method are suitable for different applications. When the size of symbol classes is small and the high accuracy for fully completed symbols is required, the image-based method is better for its high robustness. But when the recognition algorithm is used to support immediate feedback when users are sketching, the proposed method is better for its simple structural matching.

5. Conclusions

We have presented a new framework to recognize multistroke symbols with autocompletion. Firstly, strokes are segmented to primitives. Secondly, a symbol is represented as a set of biprimitives, each of which is represented as a shape descriptor. We propose two new descriptors, named DAR and DZM, respectively. Finally, the distance between an unknown symbol and a template one is calculated by biprimitive matching. Moreover, a fast NN search algorithm is also proposed, which significantly improves the search speed.

Our method is independent of stroke number and order. And there is no need to recognize primitives. Furthermore, invariance to rotation is achieved by using DZM descriptor. And it can work with few templates for each symbol class, easily extending to new symbols.

However, a limitation of our method is that a primitive cannot be drawn using more than one stroke. The future work is to alleviate the shortcomings, inspired by the work in [35].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors would like to gratefully acknowledge the "HGJ" National Major Technological Projects of China (Grant no. 2013ZX01045-004-002).

References

[1] J. J. LaViola Jr. and R. C. Zeleznik, "MathPad^2: a system for the creation and exploration of mathematical sketches," in *Proceedings of the ACM SIGGRAPH Course*, ACM, San Diego, Calif, USA, August 2007.

[2] T. F. Stahovich, "Pen-based interfaces for engineering and education," in *Proceedings of the Sketch-Based Interface and Modeling*, pp. 119–152, Springer, London, UK, 2011.

[3] L. Fu and L. B. Kara, "From engineering diagrams to engineering models: visual recognition and applications," *Computer-Aided Design*, vol. 43, no. 3, pp. 278–292, 2011.

[4] L. B. Kara, L. Gennari, and T. F. Stahovich, "A sketch-based tool for analyzing vibratory mechanical systems," *Transactions of the ASME—Journal of Mechanical Design*, vol. 130, no. 10, 2008.

[5] C. Tirkaz, B. Yanikoglu, and T. M. Sezgin, "Sketched symbol recognition with auto-completion," *Pattern Recognition*, vol. 45, no. 11, pp. 3926–3937, 2012.

[6] M. de Rosa, *New methods, techniques and applications for sketch recognition [Ph.D. thesis]*, University of Salerno, Fisciano, Italy, 2014.

[7] G. Costagliola, M. De Rosa, and V. Fuccella, "Recognition and autocompletion of partially drawn symbols by using polar histograms as spatial relation descriptors," *Computers & Graphics*, vol. 39, no. 1, pp. 101–116, 2014.

[8] T. Hammond and R. Davis, "LADDER, a sketching language for user interface developers," *Computers & Graphics*, vol. 29, no. 4, pp. 518–532, 2005.

[9] W. Lee, L. Burak Kara, and T. F. Stahovich, "An efficient graph-based recognizer for hand-drawn symbols," *Computers & Graphics*, vol. 31, no. 4, pp. 554–567, 2007.

[10] H. Hse and A. R. Newton, "Sketched symbol recognition using zernike moments," in *Proceedings of the 17th International*

Conference on Pattern Recognition (ICPR '04), pp. 367–370, IEEE, August 2004.

[11] Y. G. Zhang, L. D. Wu, and H. C. Song, "Directional Zernike moments for rotation-free recognition of online sketched symbols," *Electronics Letters*, vol. 49, no. 16, pp. 989–991, 2013.

[12] M. Oltmans, *Envisioning sketch recognition: a local feature based approach to recognizing informal sketches [Ph.D. thesis]*, Massachusetts Institute of Technology (MIT), Cambridge, Mass, USA, 2007.

[13] L. B. Kara and T. F. Stahovich, "An image-based, trainable symbol recognizer for hand-drawn sketches," *Computers & Graphics*, vol. 29, no. 4, pp. 501–517, 2005.

[14] T. Y. Ouyang and R. Davis, "A visual approach to sketched symbol recognition," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, pp. 1463–1468, July 2009.

[15] J. Almazán, A. Fornés, and E. Valveny, "A non-rigid appearance model for shape description and recognition," *Pattern Recognition*, vol. 45, no. 9, pp. 3105–3113, 2012.

[16] S. Escalera, A. Fornés, O. Pujol, P. Radeva, G. Sánchez, and J. Lladós, "Blurred Shape Model for binary and grey-level symbol recognition," *Pattern Recognition Letters*, vol. 30, no. 15, pp. 1424–1433, 2009.

[17] D. Willems, R. Niels, M. van Gerven, and L. Vuurpijl, "Iconic and multi-stroke gesture recognition," *Pattern Recognition*, vol. 42, no. 12, pp. 3303–3312, 2009.

[18] A. Delaye and E. Anquetil, "HBF49 feature set: a first unified baseline for online symbol recognition," *Pattern Recognition*, vol. 46, no. 1, pp. 117–130, 2013.

[19] Y. Liu, L. Wenyin, and C. J. Jiang, "A structural approach to recognizing incomplete graphic objects," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, pp. 371–375, Cambridge, UK, August 2004.

[20] X. Xu, Z. Sun, B. Peng, X. Jin, and W. Liu, "An online composite graphics recognition approach based on matching of spatial relation graphs," *International Journal on Document Analysis and Recognition*, vol. 7, no. 1, pp. 44–55, 2004.

[21] J. Mas, G. Sánchez, J. Lladós, and B. Lamiroy, "An incremental on-line parsing algorithm for recognizing sketching diagrams," in *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR '07)*, vol. 1, pp. 452–456, Paraná, Brazil, September 2007.

[22] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: investigation of normalization and feature extraction techniques," *Pattern Recognition*, vol. 37, no. 2, pp. 265–279, 2004.

[23] Y. Xiong and J. J. Laviola Jr., "A ShortStraw-based algorithm for corner finding in sketch-based interfaces," *Computers & Graphics*, vol. 34, no. 5, pp. 513–527, 2010.

[24] A. Wolin, B. Paulson, and T. Hammond, "Sort, merge, repeat: an algorithm for effectively finding corners in hand-sketched strokes," in *EUROGRAPHICS Symposium Sketch-Based Interfaces and Modeling*, pp. 93–100, August 2009.

[25] J. Herold and T. F. Stahovich, "A machine learning approach to automatic stroke segmentation," *Computers & Graphics*, vol. 38, no. 1, pp. 357–364, 2014.

[26] J. Herold and T. F. Stahovich, "SpeedSeg: a technique for segmenting pen strokes using pen speed," *Computers & Graphics*, vol. 35, no. 2, pp. 250–264, 2011.

[27] L. Wenyin, T. Lu, Y. Yajie, S. Liang, and R. Zhang, "Online stroke segmentation by quick penalty-based dynamic programming," *IET Computer Vision*, vol. 7, no. 5, pp. 311–319, 2013.

[28] R. S. Tumen and T. M. Sezgin, "DPFrag: trainable stroke fragmentation based on dynamic programming," *IEEE Computer Graphics and Applications*, vol. 33, no. 5, pp. 59–67, 2013.

[29] T. Y. Ouyang and R. Davis, "ChemInk: a natural real-time recognition system for chemical drawings," in *Proceedings of the 15th ACM International Conference on Intelligent User Interfaces (IUI '11)*, pp. 267–276, ACM, New York, NY, USA, February 2011.

[30] W. Deng, L. D. Wu, R. H. Yu, and J. Z. Lai, "On-line sketch recognition using direction feature," in *Human-Computer Interaction—INTERACT 2013: 14th IFIP TC 13 International Conference, Cape Town, South Africa, September 2–6, 2013, Proceedings, Part III*, vol. 8119 of *Lecture Notes in Computer Science*, pp. 259–266, Springer, Berlin, Germany, 2013.

[31] Y. Hwang, B. Han, and H.-K. Ahn, "A fast nearest neighbor search algorithm by nonlinear embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 3053–3060, June 2012.

[32] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa, "Sketch-based image retrieval: benchmark and bag-of-features descriptors," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 11, pp. 1624–1636, 2011.

[33] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[34] D. S. Zhang and G. J. Lu, "Review of shape representation and description techniques," *Pattern Recognition*, vol. 37, no. 1, pp. 1–19, 2004.

[35] T. Hammond and B. Paulson, "Recognizing sketched multistroke primitives," *Transactions on Interactive Intelligent Systems*, vol. 1, no. 1, article 4, 2011.