Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Sketched symbol recognition with auto-completion

Caglar Tirkaz [a,*], Berrin Yanikoglu [a], T. Metin Sezgin [b]

[a] *Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul 34956, Turkey*
[b] *College of Engineering, Koç University, Istanbul 34450, Turkey*

ARTICLE INFO

Article history:
Received 5 July 2011
Received in revised form
19 March 2012
Accepted 25 April 2012
Available online 4 May 2012

Keywords:
Sketch recognition
Auto-completion
Constrained semi-supervised clustering
Partial sketched symbol recognition

ABSTRACT

Sketching is a natural mode of communication that can be used to support communication among humans. Recently there has been a growing interest in sketch recognition technologies for facilitating human–computer interaction in a variety of settings, including design, art, and teaching. Automatic sketch recognition is a challenging problem due to the variability in hand drawings, the variation in the order of strokes, and the similarity of symbol classes. In this paper, we focus on a more difficult task, namely the task of classifying sketched symbols before they are fully completed. There are two main challenges in recognizing partially drawn symbols. The first is deciding when a partial drawing contains sufficient information for recognizing it unambiguously among other visually similar classes in the domain. The second challenge is classifying the partial drawings correctly with this partial information. We describe a *sketch auto-completion* framework that addresses these challenges by learning visual appearances of partial drawings through semi-supervised clustering, followed by a supervised classification step that determines object classes. Our evaluation results show that, despite the inherent ambiguity in classifying partially drawn symbols, we achieve promising auto-completion accuracies for partial drawings. Furthermore, our results for full symbols match/surpass existing methods on full object recognition accuracies reported in the literature. Finally, our design allows real-time symbol classification, making our system applicable in real world applications.

1. Introduction

Sketching is the freehand drawing of shapes and is a natural modality for describing ideas. Sketching is of high utility, because some phenomena can be explained much better using graphical diagrams especially in the fields of education, engineering and design. Sketch recognition refers to recognition of pre-defined symbols (e.g., a resistor, transistor) 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. There are many approaches in the literature for sketched symbol recognition. These include gesture-based approaches that treat the input as a time-evolving trajectory [1–3], image-based approaches that rely only on image statistics (e.g., intensities, edges) [4–6], or geometry-based approaches that attempt to describe objects as geometric primitives satisfying certain geometric and spatial constraints [7–9]. However, these methods mostly focus on recognizing fully completed symbols. In contrast, here we focus on the recognition of partially drawn symbols using image-based features.

The term auto-completion refers to predicting the sketched symbol before the drawing is completed, whenever possible. Auto-completion during sketching is desirable since it eliminates the need for the user to draw symbols in their entirety if they can be recognized while they are partially drawn. It can thus be used to increase the sketching throughput; to facilitate sketching by offering possible alternatives to the user; and to reduce user-originated errors by providing continuous feedback [10]. Despite these advantages, providing continuous feedback might also distract the user if premature recognition results are displayed [11,4].

Auto-completion requires continuously monitoring the user's drawing and deciding whether the input given thus far can be recognized unambiguously. In order to formalize the terms ambiguity and confidence, consider the task of auto-completion in SMS applications where the task is to try to guess the intended word before it is completely typed, so as to increase typing throughput. For this problem, suppose the language consists of three words: *cat*, *car*, and *apple*. If the first input character is ‘a’, then the word auto-completion system can infer the intended word (apple) unambiguously. On the other hand, if the first character is ‘c’ and no other information is available about the language, the intended word is ambiguous (either “cat” or “car”) and a text-based auto-completion system can be only 50% confident. However, suppose that the same auto-completion system is allowed to make two guesses on the word the user intends to type. Then, the system can guess the top two choices as “car” and “cat” with 100% confidence as no ambiguity is present.

* Corresponding author. Tel.: +90 555 479 84 00.
E-mail address: caglartirkaz@gmail.com (C. Tirkaz).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.04.026

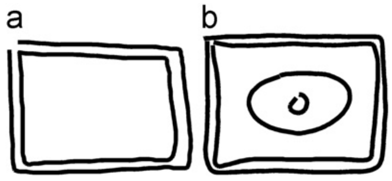


Fig. 1. Two sample sketched symbols from the COAD database.

Sketch recognition is a difficult problem due to the variability of user's hand drawing, the variability in the stroke order and the similarity of sketch classes to be recognized. Sketch recognition with auto-completion is further complicated since the system is faced with the problem of computing a confidence during the recognition process. A hand-drawn symbol is ambiguous if it appears as a sub-symbol of more than one symbol class. This is often the case with partial symbols and occasionally even with fully completed symbols.

Note that in the auto-completion framework, the system is not told when the drawing of a symbol ends. This introduces additional difficulty in classifying *full* symbols as well. For example, although the symbol shown in Fig. 1a is a fully completed symbol, it appears as a sub-symbol of another symbol shown in Fig. 1b. Hence, without knowing that the drawing of a symbol ended, a symbol such as the one shown in Fig. 1a would be classified as ambiguous. The issue of the ambiguity of fully completed symbols is discussed further in Section 3.2.

Supplying the user with predictive feedback is an important problem that has been previously studied (in terms of its effects, desirable extent, *etc.*) [12,13]. Most of the previous work has focused on giving this feedback in the form of beautification. In the context of sketch recognition, the word 'beautification' has been used in two different senses. First, it refers to recognizing and replacing a fully completed symbol with its cleaned-up version [14–16]. Second, it is used in the context of partial drawings to refer to converting the strokes of a symbol to vectorized primitives such as line segments, arcs, and ellipses [17–19]. Sometimes these primitives are further processed to adhere to Gestalt principles (e.g., lines that look roughly parallel/equal-length are made parallel/equal-length) [20–22]. Approaches of the first kind are not directly comparable to our work, as they only deal with fully completed symbols. Approaches of the second kind are also not very relevant in the context of our work, because in these systems the primitives are recognized and post-processed using Gestalt principles, however the object class is not predicted.

An implementation of beautification that couples with the idea of auto-completion has been proposed by Arvo and Novins [23]. They introduce the concept of *fluid sketching* for predicting the shape of a partially drawn primitive (e.g., a circle or a square), as it is being drawn. However, they focus on primitives only, and do not generalize their system to recognize complex objects. Li et al. [24] use the term *incremental intention extraction* to describe a system that can assist the user with continuous visual feedback. This method also has the ability to update existing decisions based on continuous user input. They focus on recognizing multi-lines and elliptic arcs. Mas et al. [25] present a syntactic approach to on-line recognition of sketched symbols. The symbols are defined by an adjacency grammar whose rules are generated automatically given the small set of seven symbols. The system can recognize partial sketches in arbitrary drawing order, using the grammar to check the validity of its hypotheses. The main shortcoming of this system is its syntactic approach, consisting of rigid rules for rule application and primitive recognition. In comparison, we use image features to describe individual symbols to handle different drawing orders and our framework is fully probabilistic.

An auto-completion application similar to ours deals with the auto-completion of complex Chinese characters in handwriting recognition, in which the auto-completion is used to facilitate the input by providing possible endings for a given partial drawing. For instance, Liu et al. [26] use a multi-path HMM to model different stroke orders that may be seen in the drawing of a character. They report accuracies with respect to the percentage of the whole character trajectory written. They obtain accuracies of 82% and 57% when 90% and 70% of the whole character is drawn, respectively.

In this paper, we present a general auto-completion application that is capable of auto-completing sketched symbols without making any assumptions about the complexity of symbols or the drawing style of users or the domain. The system classifies sketched symbols into a set of pre-defined categories while providing auto-completion whenever it is confident about its decision. The steps of the proposed method for auto-completion are explained in detail in Section 2; the experiments on databases using the method are described in Section 3; the results of the experiments are discussed in Section 4; and future directions for research are presented in Section 5.

2. Proposed method

In order to realize auto-completion, our system monitors the user's drawing and determines probable class labels and assigns a probability to each class as soon as new strokes are drawn. If the drawn (partial or full) symbol can be recognized with a sufficiently high confidence, the system makes a prediction and displays its classification result to the user. Otherwise, classification decision is delayed until further strokes are added to the input symbol.

In order to deal with the ambiguity of partial symbols, a constrained semi-supervised clustering method is applied to create clusters in the sketch space. The sketch space is acquired by extracting features from the *extended* training data, which consists of only full symbols and their corresponding partial symbols. Specifically, each full symbol in the training data and all partial symbols that appear during the course of drawing that symbol are added to extended training data (see Section 2.1). The goal of the clustering stage is to identify symbols that are similar based on the extracted features, but may belong to different classes (see Section 2.3). At the end of clustering, a cluster may contain partial/full symbols from only one class (homogeneous cluster) or from multiple classes (heterogeneous cluster). Hence, in the last step of training, we use supervised learning where one classifier per heterogeneous cluster is trained to separate the symbols falling into that cluster (see Section 2.4). If a cluster is homogeneous, then a classifier is not needed for that cluster.

During recognition, the system first finds the distance of a symbol to each of the clusters and then computes the posterior probability of each sketch class given the input, by marginalizing over clusters (see Section 2.5). This is done so as to take into account the ambiguity in assessing the correct cluster for a given query. Dealing with probabilities allows us to compute a confidence in the classification decision during the test phase. If the class label cannot be deduced with a confidence higher than a pre-determined threshold, as in the case of a partial symbol shared by many classes, the classification decision is postponed until more information becomes available. The described steps are displayed in the form of a flowchart in Fig. 2.

2.1. Extending training data with partial symbols

In standard sketched symbol databases, there are only instances of fully completed symbols rather than partial symbols.

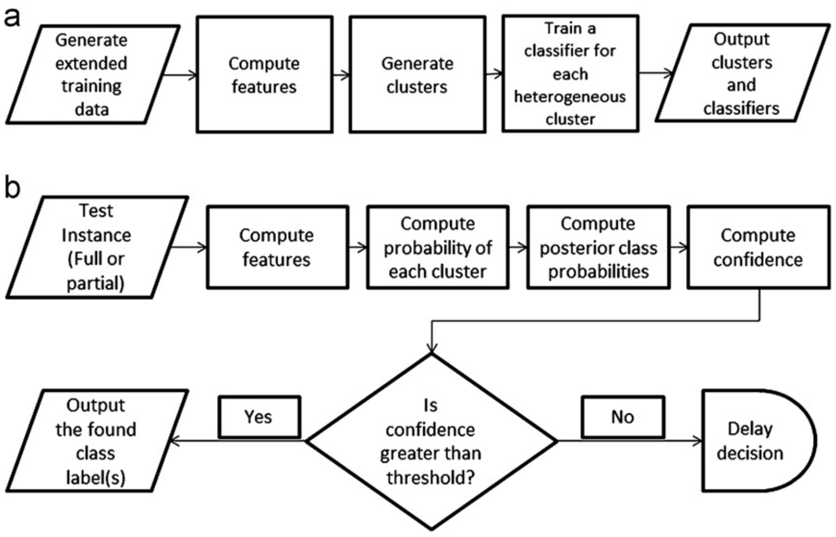

a
Generate extended training data
Compute features
Generate clusters
Train a classifier for each heterogeneous cluster
Output clusters and classifiers
b
Test Instance (Full or partial)
Compute features
Compute probability of each cluster
Compute posterior class probabilities
Compute confidence
Is confidence greater than threshold?
Yes
Output the found class label(s)
No
Delay decision

Fig. 2. The flowchart of the proposed algorithm. (a) The training steps. (b) The testing steps.

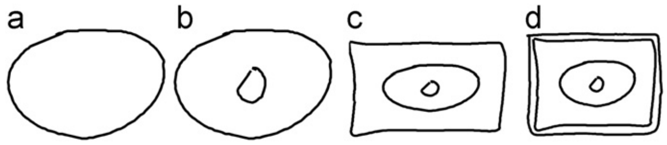


Fig. 3. A sample of extending an instance with four strokes. The original symbol shown in (d) is used to generate the three other symbol instances. (a) After first stroke. (b) After second stroke. (c) After third stroke. (d) Fully completed symbol.

In our approach, the training and test data are *automatically* extended by adding all the partial symbols that occur during the drawing of fully completed symbols. More specifically, if a particular symbol consists of three strokes (s_1,s_2,s_3), two partial symbols are extracted $\{(s_1),(s_1,s_2)\}$ and added to the database. For another user who draws the same symbol using the order (s_2,s_1,s_3), the partials $\{(s_2),(s_2,s_1)\}$ are extracted. In this fashion, for a symbol that consists of S strokes, $S-1$ partial symbols are extracted and added to the extended database, in addition to the original symbol. This process is illustrated in Fig. 3.

The number of all partial symbols that can be generated using S strokes is exponential in the number of strokes if all combinations of strokes are used. In other words, if we disregard the order between the strokes, we get 2^S-1 possible stroke subsets for a symbol with S strokes. However, since we extend the database with only those partials that actually appear in the drawing of the symbols, the number of partial symbols added to the database is much smaller. This issue can be illustrated with an example of drawing of a stick figure. If no one draws a stick figure starting with the head which is then followed by the left leg, the system would not add a partial symbol consisting of the head and the left leg into the database.

Hence, even though a pre-specified drawing order is *not* required by our system, the system takes advantage of preferred drawing orders, when they exist. Indeed, it is true that when people sketch, they generally prefer a certain order. This is based on observations from previous work in sketch recognition and psychology, which show that people do tend to prefer certain orderings over others [27–29]. So, our approach puts the focus on learning the drawing orders that are present in the training data, so as to reduce the complexity of the sketch space and improve accuracy. However, a partial symbol that results from a different drawing order may still be recognized by the system depending on its similarity to the instances in the sketch space.

2.2. Feature extraction

In order to represent a symbol, which may be a partial or a full symbol, the Image Deformation Model (*IDM*) features are used as proposed in [30]. The IDM features consist of pen orientation maps in four orientations and an end-point map indicating the end points of the pen trajectory. In order to extract the IDM features for a symbol, firstly, the orientation of the pen trajectory at each sampled point in the symbol is computed. Next, five maps are created to represent the IDM features. The first four maps correspond to orientation angles of 0°, 45°, 90° and 135°, where each map gives a higher response at locations in which the pen orientation coincides with the map orientation. The last map gives a higher response at end-points where a pen-down or pen-up movement occurs. These operations are carried out using a down sampled version of the symbol. The major advantage of the IDM feature representation is that it is independent of stroke direction and ordering.

2.3. Clustering

There is an inherent ambiguity in decision making during auto-completion. In order to address this ambiguity, we cluster partial and full symbols based on their feature representation. Clusters which contain drawings mostly from a single class indicate less ambiguity, whereas clusters that contain drawings from many distinct classes indicate high ambiguity.

In order to cluster training instances, we first experimented with the unsupervised *Expectation Maximization* (EM) algorithm [31]. We used the implementation of EM available in *WEKA* [32]. The results with the EM algorithm showed a low performance for full symbols. Since our goal is to provide auto-completion without sacrificing full symbol recognition performance, we switched to the semi-supervised constrained k-means clustering algorithm (*CKMeans*) [33]. Our motivation when using CKMeans is to enforce the separation of full symbols of different classes into different clusters, while grouping the full symbols of the same class in one cluster through constraints. With this approach, we aim to reduce errors in classifying full symbols, since errors done in full symbols may distract the user more than errors done in partial symbols. The effect of the clustering algorithm on the recognition accuracies is further discussed in Section 3.6.

The CKMeans algorithm employs background knowledge about the given instances and uses constraints of the form *must-link* and

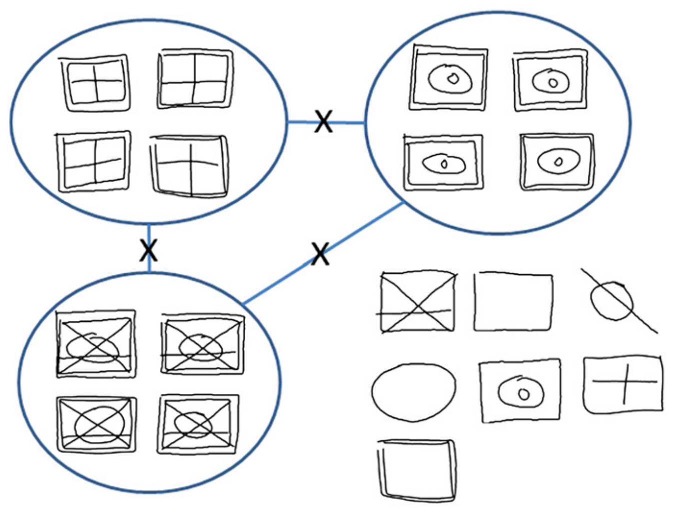


Fig. 4. A visual depiction of defined constraints used in the CKMeans algorithm. Must-link and cannot-link constraints involving full shapes are represented as circles and crossed lines, respectively.

cannot-link between individual instances while clustering the data. The must-link constraint between two instances specifies that the two instances should be clustered together, whereas the cannot-link constraint specifies that the two instances must not be clustered together. In this work, we generate

- Must-link constraints between full sketches of a class since we want them to be clustered together.
- Cannot-link constraints between full sketches of different classes since we do not want them to be clustered together.

A visual depiction of the must-link and cannot-link constraints between the fully drawn symbols is given in Fig. 4. The must-link constraints are shown as circles, indicating that circled instances should be clustered together; while the cannot-link constraints are shown as crossed lines between circles, indicating that full-shape instances in different classes should not be clustered together. No constraints are generated for partial symbols. We allow partial sketches of different classes to be clustered together because partial sketches of different classes can be visually similar and have similar feature representations.

The constraints are specified using an $N \times N$ symmetric matrix, where N is the number of instances to be clustered in the extended training set and the matrix elements can be -1, 0, 1 denoting cannot-link, no constraint and must-link constraints, respectively. The process of generating constraints is handled fully automatically using the class labels present in the original training data.

2.4. Posterior class probabilities

In order to make a prediction given a test symbol, x, we compute the posterior probability of each symbol class s_i, by marginalizing over clusters

$$P(s_i|x) = \sum_{k=1}^{K} P(s_i,c_k|x) = \sum_{k=1}^{K} P(s_i|c_k,x)P(c_k|x) \tag{1}$$

where x represents the input symbol; K is the total number of clusters; $P(s_i|c_k,x)$ is the probability of symbol class s_i given cluster c_k and input x; and $P(c_k|x)$ denotes the posterior probability of cluster c_k given x. Notice that rather than finding the most likely cluster, we take a Bayesian approach and consider $P(c_k|x)$ in order to reflect the ambiguity in cluster selection.

Given the distance from x to each cluster center, an exponentially decreasing density function and Bayes' formula, we estimate $P(c_k|x)$ as

$$P(c_k|x) = P(x|c_k)P(c_k)/P(x) \simeq e^{-\|x-\mu_k\|^2} P(c_k)/P(x) \tag{2}$$

where μ_k is the mean of the kth cluster c_k and $P(c_k)$ is the prior probability of c_k estimated by dividing the number of instances that fall into the kth cluster by the total number of clustered instances. $P(x)$ denotes the probability of occurrence of the input x, which is omitted in the calculations since it is the same for each cluster.

2.4.1. Supervised classification within a cluster

In order to compute $P(s_i|c_k,x)$, a support vector machine (SVM) [34] is trained for each *heterogeneous* cluster, which is defined as a cluster that contains instances of more than one class. If an instance is clustered into a *homogeneous* cluster, which is defined as a cluster containing instances of only a single class, then we simply assign a probability of 1 for the class that forms the cluster and 0 for the other classes.

Note that supervised classification step can help in cases where the symbols falling into one cluster can actually be classified unambiguously. For instance, consider the synthetic cluster given in Fig. 5a containing six partial symbols. Furthermore, assume that the corresponding fully completed drawings are given in Fig. 5b. Hence, the partial symbols in the cluster belong to two distinct classes, either the class with an upside 'T' or a downside 'T' inside a square. While the partial symbols in the cluster look similar enough to be clustered together, the position of the line in the square can be used to separate them apart. This is the motivation for training a classifier to separate the instances falling in heterogeneous clusters. If all the symbols that fall in a cluster look very similar, the supervised classification may not bring any contribution and the shapes falling in that cluster would be labeled as ambiguous by the system, since multiple classes would have similar posterior probabilities.

The semi-supervised clustering step used before the supervised classification performed for each cluster aims to divide the big problem into smaller problems that are hopefully easier to solve.

In order to show the contribution of the supervised classification step, we conducted an experiment in which we assumed $P(s_i|c_k,x) = P(s_i|c_k)$ and modified Eq. (1) accordingly. Specifically, if it is assumed that the probability $P(s_i|c_k,x)$ is independent of the input instance x, then

$$P(s_i|c_k,x) = P(s_i|c_k) \tag{3}$$

and $P(s_i|c_k)$ can be estimated during training by dividing the number of instances from symbol class i that fall into cluster k by the total number of instances in that specific cluster. Of course,

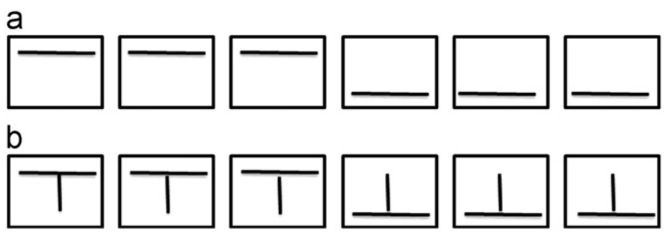


Fig. 5. The visual representation of a synthetic cluster containing six symbols is given in (a). The completed drawings are given in (b).

with this assumption there is a loss of information and we see a decrease in the accuracies, as explained in Section 3.7.

2.5. Confidence calculation

Having computed the posterior class probabilities for an input symbol, the system either rejects the symbol (delays making a decision) or shows the inferred class label(s) to the user. In an auto-completion scenario, the user may be interested in seeing the Top-N guesses of the system and choose from among those to quickly finish drawing his/her partial symbol. For example, if $N=2$, the system shows the user two alternative guesses.

Naturally, as N increases, the accuracy increases, though too many alternatives would also clutter the user interface. Keeping N as a variable that can be set by a user, the confidence in prediction is calculated by summing the estimated posterior probabilities of the most probable Top-N classes. The classification decision is delayed until there is enough information to unambiguously classify the symbol if the computed confidence is lower than a threshold. We refer to the proportion of symbols that are not classified due to low confidence as "reject rate". If the confidence is above the threshold, the N most probable classes are displayed to the user. In the experiments, the system performance is measured for $N=[1,2,3]$.

3. Experimental results

The proposed system is evaluated on two databases from different domains, in terms of the Top-N classification accuracy in full and partial symbols separately, for varying values of N. For each database, the system parameters (the number of clusters, K, and the confidence threshold, C) are optimized using cross-validation. Parameter optimization is done as follows: for each parameter value pair (e.g., $K=40$ and $C=0.0$), we record the validation set accuracy using eight-fold cross-validation. Cross-validation is done by splitting the training data randomly, selecting 80% of the full symbols and all of their partials as training examples and the remaining 20% of the full symbols and all of their partials as validation examples. This is repeated eight times with randomly shuffled data and the median system performance on the validation set is recorded, for that particular parameter combination. The selected parameter pair is then fixed and used in testing the system on a separate test set.

3.1. Databases

The first database we use to test our system is the Course of Action Diagrams (COADs) database. The COAD symbols are used by military in order to plan field operations [35]. The symbols in this database represent military shapes such as a friendly or enemy units, obstacles, supply units, *etc*. Some samples of the hand-drawn symbols from this domain are displayed in Fig. 6. As mentioned before, some symbols have distinctive shapes whereas others appear as partial symbols of one or more symbols. For example, Fig. 6n is a sub-shape of Fig. 6m. In total this database contains 620 samples from 20 symbols drawn by eight users.

Since no separate test set is available for the COAD database, a randomly selected 20% of all the available data is reserved for testing, prior to parameter optimization done with cross-validation. The parameter optimization for the COAD database aims to find (C,K) pairs at which the system performs close to human recognition rates as will be described in Section 3.2. We report the system performance on this test set in detail in Sections 3.3 and 3.4.

The second database we used in our experiments is the NicIcon [36] symbol database used in the domain of crisis management. The database contains 26 163 symbols representing 14 classes collected from 32 individuals. The symbols represent events and objects such as accident, car, fire, *etc*. Some of the sketched symbols from the database are displayed in Fig. 7. The NicIcon database defines the training and test sets and in the experiments we used these sets accordingly.

3.2. Auto-completion performance benchmark

There are no reported auto-completion accuracies for the COAD and the NicIcon databases, in the literature. However, both databases have been used before in testing sketched symbol recognition algorithms designed for classifying full symbols. While presenting the results of our experiments on the databases, we give both partial and full symbol recognition performances and compare them to full symbol recognition rates from the literature.

In order to test the accuracy decrease due to the auto-completion scenario (since without knowing that the drawing ended, we cannot be sure of the class), we measured a human expert's performance on the COAD database, assuming an auto-completion framework. Specifically, we showed all partial and full symbols in the COAD database to a human expert without telling

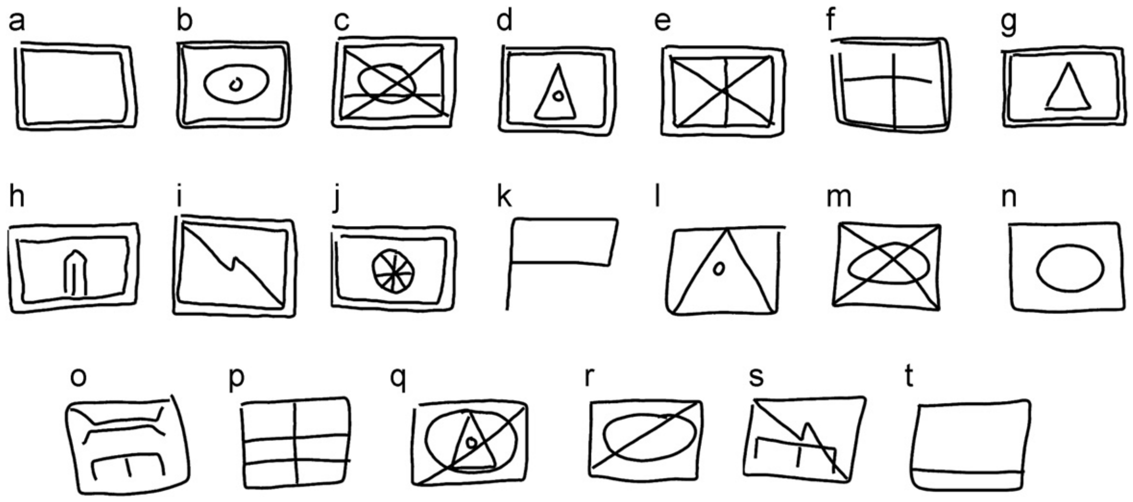


Fig. 6. A sample symbol from each class in the COAD database.

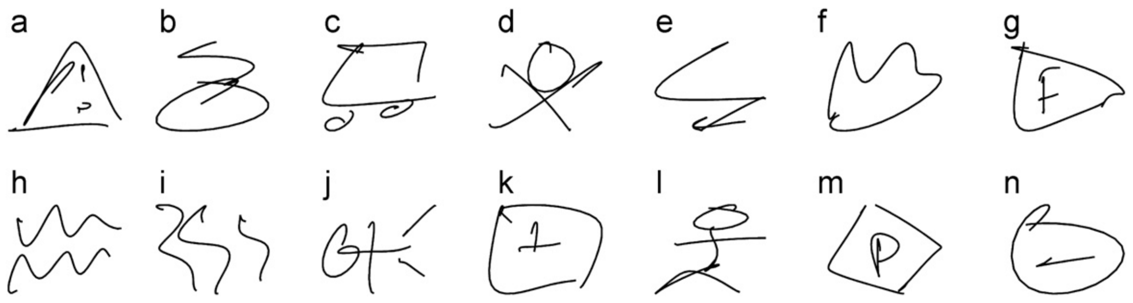


Fig. 7. A sample symbol from each class in the NicIcon database.

Table 1
Human accuracy showing the proportion of partial and full symbols that need to be rejected in order to achieve 100% accuracy, for varying values of $N=[1-3]$ on the COAD database. The reject rate indicates percentage of the cases where the human expert decided that there is not sufficient information for classification, hence declined prediction.

Top-N policy	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
$N=1$	100	100	75.36	33.58
$N=2$	100	100	61.74	18.25
$N=3$	100	100	55.07	12.41

whether the drawing was finished or not. The expert was then asked to choose the correct class if the sample could be classified unambiguously for varying values of N, and reject it otherwise.

The first row of Table 1 indicates that 75.36% of the partial symbols and 33.58% of full symbols are found to be ambiguous when $N=1$; that is when the expert is asked to identify the correct class. The symbols that were not rejected were classified with 100% accuracy. As mentioned before, both partial and full symbols in a database may be ambiguous in an auto-completion scenario, without knowing that the user has finished drawing. In particular, full symbols that are found ambiguous are those that can be partial drawings of other symbols.

For $N=2$ and $N=3$, the task is to decide whether the symbol can be placed with certainty in one of the N possible classes, hence, the reject rates decrease as N increases. Human performance for the NicIcon database was not calculated due to the large size of the database and the amount of manual work involved.

Human recognition rates may be used as a point of reference for assessing an automatic recognition system's performance. In particular, we can compare the proposed system's accuracy to that of the human expert's, at the reject rates close to the expert's.

3.3. Accuracy on the COAD database with EM clustering

As described in Section 2.3, we first used the EM method for clustering. We summarize the validation and test performances on the COAD database using the EM algorithm, so as to motivate the use of the CKMeans algorithm.

Table 2 shows a summary of the cross-validation accuracies obtained with different values for the system parameters (cluster count parameter K and confidence threshold C) that give reject rates close to human reject rates, for comparability. The best parameter pair giving the highest validation set accuracy is indicated with an asterisk. We then used the chosen parameters ($K=40$, $C=0.84$) to evaluate the test set performance, obtaining the results shown in Table 3. The human accuracies and reject rates measured on the whole COAD database are also listed for easy comparison.

While the results shown in Table 3 are already good (not only lower reject rates compared to human reject rate, but also

Table 2
Validation accuracies for the COAD database for $N=1$ using the EM algorithm.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
80/0.87	88.40	96.90	71.68	33.87
60/0.84	91.50	98.31	73.26	33.87
40/0.84*	91.37	98.44	73.83	33.33

Table 3
Test accuracy for the COAD database for $N=1$ using the EM algorithm.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate partial (%)	Reject rate for full (%)
40/0.84	94.05	97.98	63.95	27.74
Human	100.00	100.00	75.36	33.58

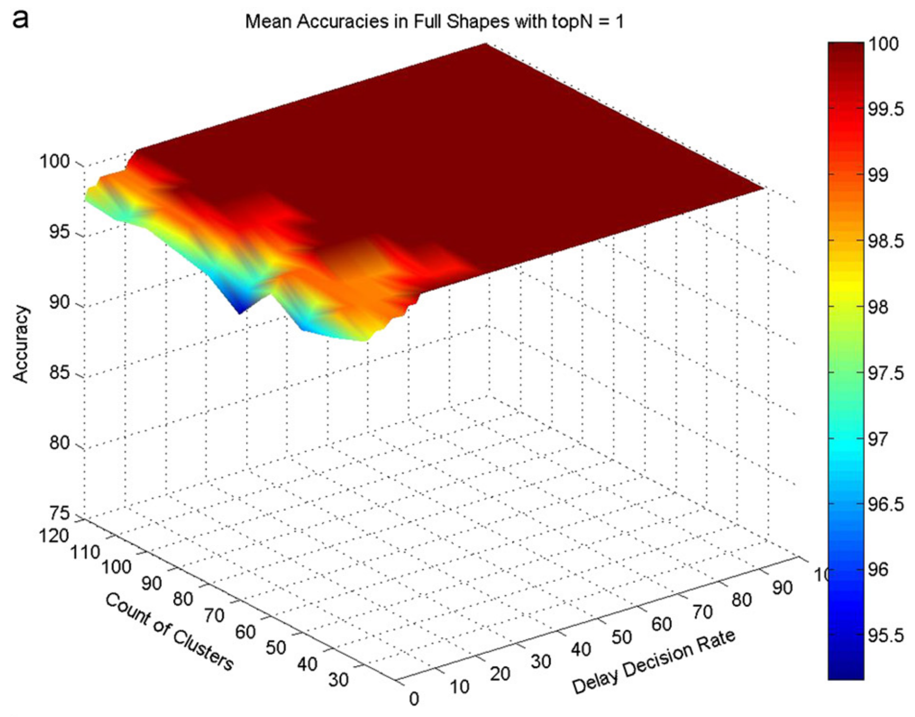
somewhat lower accuracies compared to human accuracies), we next evaluated the semi-supervised CKMeans algorithm that was expected to do better with the fully drawn symbols, due to the imposed constraints. The results obtained using the CKMeans for clustering while keeping the other parts of the system unchanged are given in the next sections.

3.4. Accuracy on the COAD database using CKMeans clustering

The performance surface of the system during validation with respect to the system parameters for the COAD database is shown in Fig. 8, while representative points on this performance surface are listed in Table 4. The table is organized such that the first three rows present accuracies where the system is forced to make a decision ($C=0$); while the last three rows present accuracies where the reject rates are close to human expert rates. Note that the accuracy result for full shapes at zero reject rate is comparable to recognition results without auto-completion, while the accuracies at human reject rates can be compared to human accuracies.

The best results are obtained with $K=40$, $C=0.74$ and $K=40$, $C=0.00$, depending on whether the system has the reject option or not, respectively. The corresponding test performance, obtained with these parameters, is shown in Table 5. One can see that the system achieves 100% accuracy for full symbols and 92.65% for partials, when the reject rates are even lower than the human reject rates. This counter intuitive result is explained in Section 3.4.1.

When $N=2$ and $N=3$, the accuracies increase since the system can make two/three guesses as to what the class of the object is. We again choose the best parameters in terms of validation set accuracy at close to human reject rates, which are found to be $K=40$, $C=0.88$ for $N=2$ and $K=40$, $C=0.95$ for $N=3$. At these settings, the test accuracies are given in Tables 6 and 7.



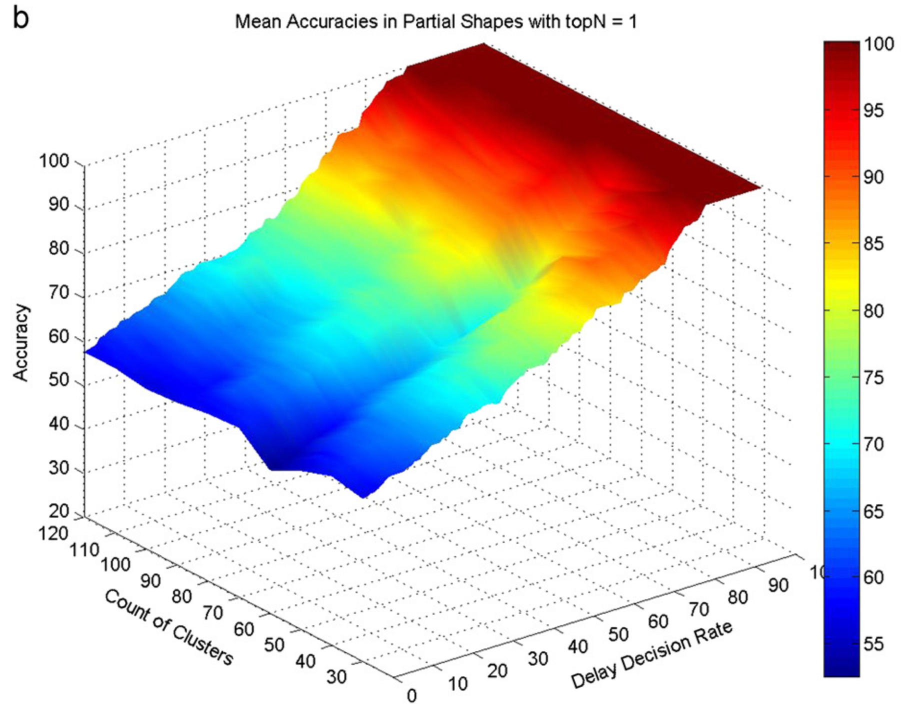


Fig. 8. Validation performance for $N=1$, on COAD database, using the CKMeans algorithm. (a) The performance surface for full symbols. (b) The performance surface for partial symbols.

As mentioned before, the COAD database does not have explicitly defined training and test sets. So, in order to strengthen our results, we repeated the experiments using five-fold cross validation where for each fold, 20% of the instances are separated for testing and the remaining instances are used for training. In each of these five experiments, the system parameters are optimized in a separate cross-validation as explained above, using only the training set allocated in that fold. For brevity, the results obtained with different test sets are shown in Table 8 for only $N=1$, along with the selected optimal parameter values. The row labeled *Exp*1 contains the results that are presented before. As illustrated in the table, the test results with different folds show low variance for the proposed classification method.

3.4.1. *Discussion*

For the COAD database, Tumen et al. [37] report a recognition accuracy around 96% for full symbols. This can be compared to the 97.08% accuracy obtained by our system during testing of full symbols, when reject was not an option (first row of Table 5). So, our system not only achieves better accuracy, but also does so while providing auto-completion.

Table 4
Validation performance for $N=1$ using the CKMeans algorithm on the COAD database.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
80/0.00	58.03	96.77	0.00	0.00
60/0.00	52.48	97.85	0.00	0.00
40/0.00*	58.49	96.77	0.00	0.00
80/0.83	90.46	100.00	75.00	30.11
60/0.79	88.71	100.00	75.76	23.12
40/0.74*	95.48	99.31	75.47	24.19

The rows with * indicate the parameters giving the best results.

Table 5
Test performance for $N=1$ using the CKMeans algorithm on the COAD database.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
40/0.00	54.94	97.08	0.00	0.00
40/0.74	92.65	100.00	70.82	17.52
Human	100.00	100.00	75.36	33.58

Table 6
Test performance for $N=2$ using the CKMeans algorithm on the COAD database.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
40/0.00	72.10	99.27	0.00	0.00
40/0.88	95.00	100.00	65.67	18.25
Human	100.00	100.00	61.74	18.25

Table 7
Test performance for $N=3$ using the CKMeans algorithm on the COAD database.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
40/0.00	79.83	99.27	0.00	0.00
40/0.95	97.53	100.00	65.24	17.52
Human	100.00	100.00	55.07	12.41

More importantly, our system obtains 100% recognition accuracy in recognizing full symbols (second row, Table 5) at *lower* reject rates compared to humans. This may seem unintuitive at first, but it can be explained by two factors. First of all, human experts reject a full symbol F and tag it as ambiguous if it is a partial symbol of some other symbol S. However, if F has not occurred in the partial symbols of S in the training data, that information is exploited in the presented system. For instance, if the outer squares in Fig. 6b–e are always drawn last, then Fig. 6a is not a partial symbol of any of these symbols in practice. This information is captured by the system, as explained in Section 2.1.

Secondly, our system is biased toward performing better in full symbol recognition, when CKMeans is used with the constraints of not mixing full shape clusters. The algorithm is designed this way because, as mentioned earlier, an error in classifying a full symbol might cause more of a distraction to the user, than an error in classifying partial symbols.

3.5. Accuracies on the NicIcon database using CKMeans clustering

The validation set performance of the system with respect to the system parameters for the NicIcon database is shown in Fig. 9, while representative points on this performance surface are listed in Table 9. The first three rows of the table present accuracies on the performance surface at zero reject rate. Since no human expert labeling is done for this database, the last three rows in the table present the points at which less than 10% of the partials are rejected. The best results are obtained with $K=20$, $C=0.48$ and $K=20$, $C=0.00$, depending on whether the system has the reject option or not, respectively.

For $N=2$ and $N=3$, we do not change C and K. At these settings, the test accuracies are given in Tables 11 and 12.

Table 8
Experiment results obtained using different test sets. *Exp*1 refers to the results given in Tables 4 and 5. The last row shows the mean of the accuracies and reject rates for the five-folds.

ID	K, C	Test Type	Validation accuracy (%)	Validation reject (%)	Test accuracy (%)	Test reject (%)
*Exp*1	40, 0.74	Full	99.31	24.19	100	17.52
		Partial	95.48	75.47	92.65	70.82
Fold 1	100, 0.78	Full	100.00	28.19	100.00	24.63
		Partial	87.69	77.19	90.00	76.53
Fold 2	80, 0.82	Full	100.00	24.91	100.00	18.58
		Partial	94.97	73.83	95.92	68.70
Fold 3	80, 0.82	Full	100.00	25.51	98.91	22.34
		Partial	91.62	75.33	88.00	70.48
Fold 4	60, 0.77	Full	100.00	22.11	100.00	16.10
		Partial	94.87	71.24	89.39	69.59
Fold 5	40, 0.75	Full	100.00	25.05	100.00	20.53
		Partial	94.19	76.28	98.28	73.52
Mean		Full	100.00	25.15	99.78	20.44
		Partial	92.67	74.77	92.32	71.76

3.5.1. Discussion

As mentioned earlier, the NicIcon database is an easier database from the perspective of the auto-completion problem because the symbols in this database have more discriminative sub-symbols and fewer number of strokes. Even when the reject rate in partial symbols is 0%, partial symbol recognition accuracy is quite high (87.63%). As a comparison, the partial symbol recognition accuracy for the COAD database is only 54.94% as presented in Table 5.

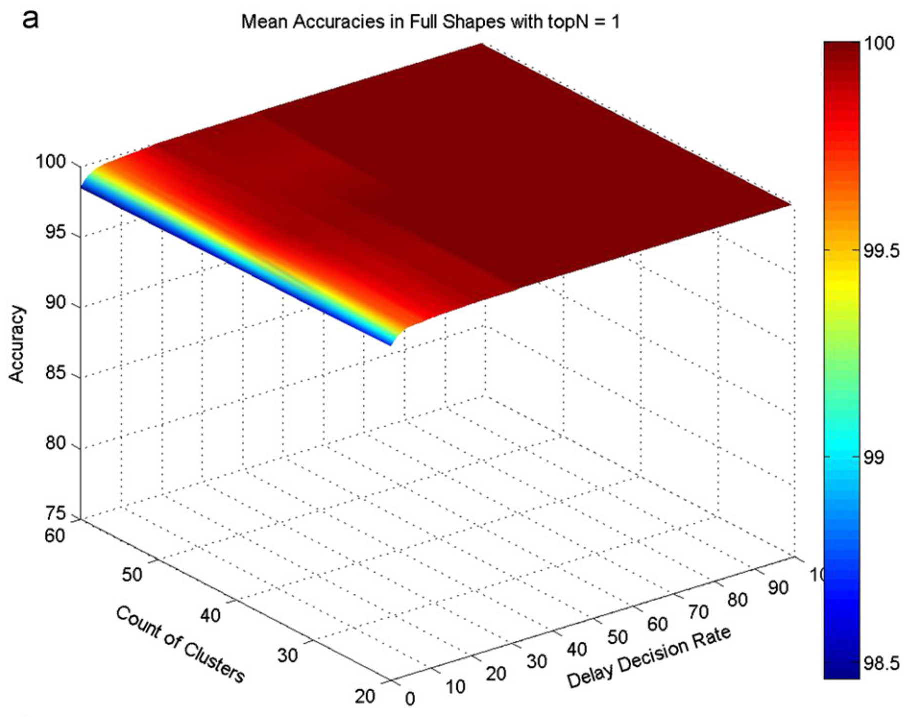
In [3], the authors report a recognition accuracy of 99.2% for the NicIcon database. Our system achieves a recognition rate of 93.26% for full symbols, with 0% reject rate as displayed in Table 10. Our recognition accuracy in full symbols is lower than the reported recognition rate on this database. However, our system is capable of performing auto-completion which is a valuable feature for sketch recognition applications.

The last experiment result in the NicIcon database for $N=3$ is interesting. When $N=3$, our system produces a higher recognition accuracy for partials than for fully completed symbols (97.47% vs. 96.75%). This result also supports the claim that auto-completion is well suited to the symbols in this database.

3.6. Comparison of clustering algorithms

In order to better observe the effect of semi-supervision on performance, we compared the accuracies obtained using each of the two clustering algorithms, for varying reject rates. In Fig. 10, we present the comparison of EM and CKMeans in full symbol recognition using 80 clusters.[1] We can observe that the CKMeans

[1] Similar patterns are observed for different cluster sizes as well.



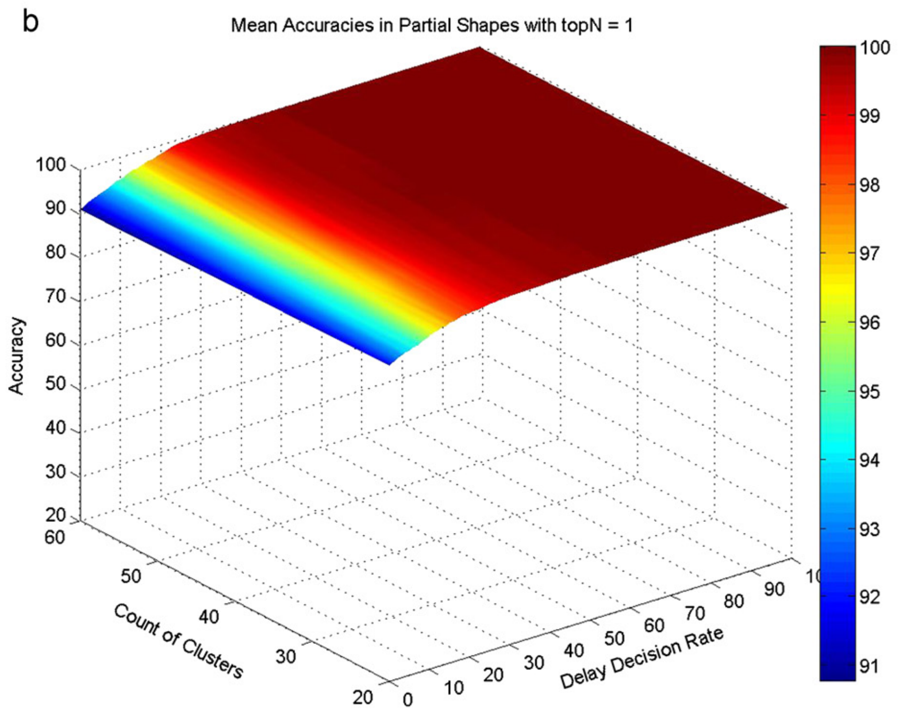


Fig. 9. Validation performance surface for $N=1$ using the CKMeans algorithm on the NicIcon database. (a) The performance surface for full symbols. (b) The performance surface for partial symbols.

performs better for all reject rates and achieves a high accuracy even for low reject rates.

Similarly, Fig. 11 compares the partial symbol recognition accuracies, using the two clustering algorithms. We see that in the presence of semi-supervision, the accuracies increase when CKMeans is used not only for full symbols but also for partial symbols.

3.7. Effect of supervised classification

As mentioned earlier in Section 2.4.1, we also conducted an experiment in order to observe the effect of supervised classification on system accuracy. We repeated the same experiments as above, using the NicIcon database, but removing the supervised classification component completely and using Eq. (3).

When the supervised learning step was eliminated, the best validation result at zero reject rate was obtained for 40 clusters. When the test performance was measured at this setting ($K=40$, $C=0.00$), we obtained the results given in Table 13. In this table, the first row shows the test performance on the NicIcon database using supervised classification (from Table 10) whereas the second row shows the test performance without the supervised classification. The contribution of the supervised classification is clear according to these results: through the supervised classification step, the accuracy increases not only for full symbols, but also for partials.

3.8. Implementation and runtime performance

We used LibSVM for the implementation of support vector machines [38], while the code for testing is written in Matlab. Classification of a single test instance takes roughly 0.07 s on a 2.16 GHz laptop. So, the system runs in real time—as required for an auto-completion application.

Table 9
Validation performance for $N=1$ using the CKMeans algorithm on the NicIcon database.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
60/0.00	90.75	98.43	0.00	0.00
40/0.00	91.39	98.52	0.00	0.00
20/0.00*	91.88	98.64	0.00	0.00
60/0.42	94.79	99.10	9.33	1.80
40/0.42	95.00	99.37	8.94	1.96
20/0.48*	95.92	99.40	9.74	2.36

The rows with * indicate the parameters giving the best results.

Table 10
Test performance for $N=1$ using the CKMeans algorithm on the NicIcon database.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
20/0.00	87.63	93.26	0.00	0.00
20/0.48	93.06	96.97	14.33	7.34

Table 11
Test performance for $N=2$ using the CKMeans algorithm on the NicIcon database.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
20/0.00	94.81	95.71	0.00	0.00
20/0.48	95.66	96.51	2.77	1.80

Table 12
Test performance for $N=3$ using the CKMeans algorithm on the NicIcon database.

K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
20/0.00	97.47	96.75	0.00	0.00
20/0.48	97.57	96.86	0.30	0.19

4. Summary and discussions

We describe a system that uses semi-supervised clustering followed by supervised classification for building a sketch recognition system that provides auto-completion. Our system approaches the auto-completion problem probabilistically and, although we have used a fixed confidence threshold during our tests, the confidence parameter can be modified by the user to specify the desired level of prediction/suggestion from the system. Experimental results show that predictions can be made for auto-completion purposes with high accuracies when the reject rates are close to that of a human expert. As described in the experiments, our system achieves 100.00% and 92.65% accuracies in the COAD database at human expert reject rates for full and partial symbols, respectively. For the NicIcon database, 93.26% and 87.63% accuracies are obtained without rejecting any instances for full and partial symbols, respectively. The system works in real time.

Few points are worth noting. First of all, there is a trade-off between accuracy and the ability to make predictions. For all values of N and K, increasing the confidence threshold improves accuracy, but it also increases the reject rate. It is important to locate points at which both reject rates and accuracies are acceptable. These points are found in this work using a validation set, while in the actual application, the confidence threshold for a similarly selected K could be adjusted by the user.

Another point is that the system does not discriminate between full and partial symbols in its rejections. When the confidence threshold increases, more and more full symbol instances are rejected. However, what we would really like is to recognize full symbols well, at the cost of rejecting more partials if necessary. As we discussed in Section 3.6, integrating knowledge about the full symbols and using a semi-supervised clustering algorithm achieve this to some degree and also increase partial symbol recognition accuracy.

5. Future work

In this work, although we addressed on-line sketch recognition, we assumed that the scene contained only one object (either partial or full). In other words, we have not addressed issues that come up in the context of continuous sketch recognition where the scene may contain multiple objects. As shown the in previous work [39], continuous sketch recognition has its own challenges. In particular, the issue of how segmentation and auto-completion

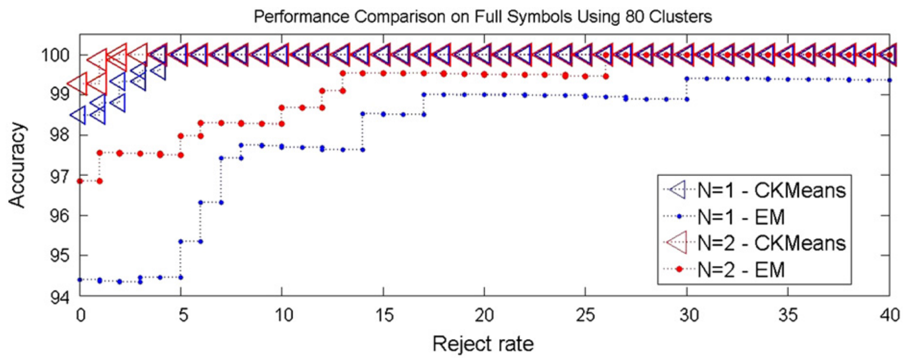


Fig. 10. Comparison of full symbol accuracies on the COAD database, using EM and CKMeans with 80 clusters.

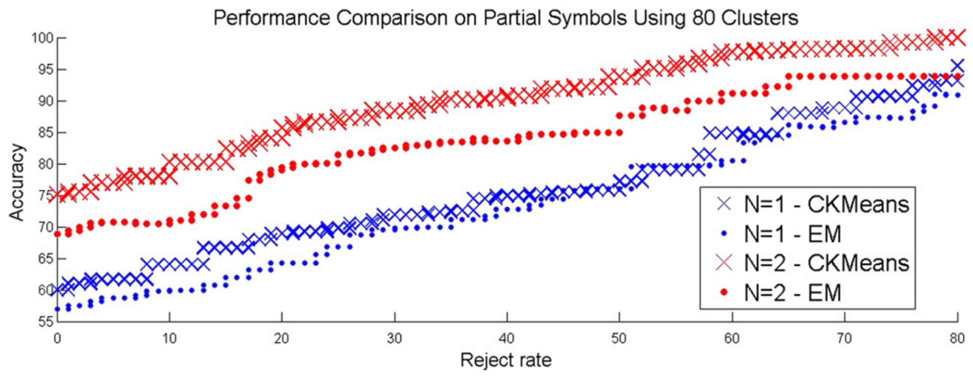


Fig. 11. Comparison of partial symbol accuracies on the COAD database, using EM and CKMeans with 80 clusters.

Table 13
The effect of removing the supervised classification step on the accuracies.

Method	K/C	Partial accuracy (%)	Full accuracy (%)	Reject rate for partial (%)	Reject rate for full (%)
Proposed	20/0.00	87.63	93.26	0.00	0.00
No supervision	40/0.00	74.96	89.37	0.00	0.00

can be addressed simultaneously requires further research. It may be the case that an approach that is based on dynamic programming may suffice and can be adapted to a scenario where the most recent object is partially drawn. It may also be the case that introducing the option for auto-completion may require modifications to the segmentation framework. More specifically, one has to make sure that the segmentation hypotheses generated by the recognition system allows only the latest object to be partial; all other groups computed by the segmentation step will have to correspond to fully completed objects.

Another question that arises naturally is how humans react to an interface which offers auto-completion. In order to figure out how and when auto-completion should be offered, user experiments need to be carried out. During those experiments, the parameters that we used throughout the paper, such as confidence threshold C and the number of choices to be offered, N, can be studied to find optimum parameter values.

Integrating machine learning methods for classifier combination into the system is also a future direction of research. During experiments, we observed that certain values of K do a better job at predicting full symbols, whereas others are better at predicting partials. Exploring a system that employs an ensemble of different K values can further boost the accuracies.

References

[1] D. Rubine, Specifying gestures by example, in: Computer Graphics and Interactive Techniques, vol. 25, 1991, pp. 329–337.

[2] A.C. Long, Jr., J.A. Landay, L.A. Rowe, J. Michiels, Visual similarity of pen gestures, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '00, ACM, New York, NY, USA, 2000, pp. 360–367.

[3] D. Willems, R. Niels, M. van Gerven, L. Vuurpijl, Iconic and multi-stroke gesture recognition, Pattern Recognition 42 (12) (2009) 3303–3312.

[4] L.B. Kara, T.F. Stahovich, Hierarchical parsing and recognition of hand-sketched diagrams, in: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology, ACM, New York, NY, USA, 2004, pp. 13–22.

[5] H. Hse, A.R. Newton, Sketched symbol recognition using zernike moments, in: Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04), IEEE Computer Society, Washington, DC, USA, vol. 1, 2004, pp. 367–370.

[6] M. Oltmans, Envisioning Sketch Recognition: A Local Feature Based Approach to Recognizing Informal Sketches, Ph.D. Thesis, Cambridge, MA, USA, 2007.

[7] T. Hammond, R. Davis, Automatically transforming symbolic shape descriptions for use in sketch recognition, in: Proceedings of the 19th National Conference on Artificial Intelligence, AAAI'04, AAAI Press, 2004, pp. 450–456.

[8] T. Hammond, R. Davis, Ladder, a sketching language for user interface developers, Computers & Graphics 29 (2005) 518–532.

[9] F. Brieler, M. Minas, A model-based recognition engine for sketched diagrams, Journal of Visual Languages and Computing 21 (2010) 81–97.

[10] A. Blessing, T.M. Sezgin, R. Arandjelovic, P. Robinson, A multimodal interface for road design, in: International Conference on Intelligent User Interfaces Workshop on Sketch Recognition, 2009.

[11] G. Feng, C. Viard-Gaudin, Z. Sun, On-line hand-drawn electric circuit diagram recognition using 2D dynamic programming, Pattern Recognition 42 (2009) 3215–3223.

[12] C. Alvarado, Sketch recognition user interfaces: guidelines for design and development, in: AAAI 2004 Symposium on Making Pen-Based Interaction Intelligent and Natural, AAAI Fall Symposium, Menlo Park, CA, 2004, pp. 8–14.

[13] P. Wais, A. Wolin, C. Alvarado, Designing a sketch recognition front-end: user perception of interface elements, in: SBIM '07: Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling, ACM, New York, NY, USA, 2007, pp. 99–106.

[14] B. Plimmer, J. Grundy, Beautifying sketching-based design tool content: issues and experiences, in: Proceedings of the Sixth Australasian User Interface Conference (AUIC 2005), vol. 40, 2005, pp. 31–38.

[15] C. Alvarado, R. Davis, Resolving ambiguities to create a natural sketch based interface, in: Proceedings of IJCAI, 2001, pp. 1365–1371.

[16] H.H. Hse, A. Richard Newton, Recognition and beautification of multi-stroke symbols in digital ink, Computers & Graphics 29 (2005) 533–546.

[17] B. Paulson, T. Hammond, A system for recognizing and beautifying low-level sketch shapes using NDDE and DCR, in: 20th Annual ACM Symposium on User Interface Software and Technology Posters, October 2007.

[18] B. Paulson, T. Hammond, Paleosketch: accurate primitive sketch recognition and beautification, in: Proceedings of the 13th International Conference on Intelligent User Interfaces, IUI '08, ACM, New York, NY, USA, 2008, pp. 1–10.

[19] J.A. Landay, B.A. Myers, Interactive sketching for the early stages of user interface design, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995, pp. 43–50.

[20] T.M. Sezgin, Sketch based interfaces: early processing for sketch understanding, in: Proceedings of PUI-2001, NY, ACM Press, 2001.

[21] T. Pavlidis, C.J. Van Wyk, An automatic beautifier for drawings and illustrations, SIGGRAPH Computer Graphics 19 (3) (1985) 225–234.

[22] T. Igarashi, S. Matsuoka, S. Kawachiya, H. Tanaka, Interactive beautification: a technique for rapid geometric design, in: Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology, UIST '97, ACM, New York, NY, USA, 1997, pp. 105–114.

[23] J. Arvo, K. Novins, Fluid sketches: continuous recognition and morphing of simple hand-drawn shapes, in: Proceedings of the ACM Symposium on User Interface Software and Technology, ACM, 2000, pp. 73–80.

[24] J. Li, X. Zhang, X. Ao, G. Dai, Sketch recognition with continuous feedback based on incremental intention extraction, in: IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces, ACM, New York, NY, USA, 2005, pp. 145–150.

[25] J. Mas, G. Sanchez, J. Llados, B. Lamiroy, An incremental on-line parsing algorithm for recognizing sketching diagrams, in: ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition, IEEE Computer Society, Washington, DC, USA, 2007, pp. 452–456.

[26] P. Liu, L. Ma, F.K. Soong, Prefix tree based auto-completion for convenient bi-modal Chinese character input, in: ICASSP, 2008, pp. 4465–4468.

[27] T.M. Sezgin, R. Davis, Sketch recognition in interspersed drawings using time-based graphical models, Computers & Graphics 2008, 32 (5).

[28] P. van Sommers, Drawing and Cognition: Descriptive and Experimental Studies of Graphic Production Processes, Cambridge University Press, 1984.

[29] S. Simhon, G. Dudek, Sketch interpretation and refinement using statistical models, in: A. Keller, H.W. Jensen (Eds.), Rendering Techniques, Eurographics Association, 2004, pp. 23–32.
[30] T.Y. Ouyang, R. Davis, A visual approach to sketched symbol recognition, in: Proceedings of the International Joint Conferences on Artificial Intelligence, 2009, pp. 1463–1468.
[31] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B 39 (1) (1977) 1–38.
[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, in: SIGKDD Explorations, vol. 11, 2009.
[33] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl, Constrained k-means clustering with background knowledge, in: ICML, Morgan Kaufmann, 2001, pp. 577–584.
[34] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning (1995) 273–297.
[35] 101-5-1, Operational Terms and Graphics, Department of the Army 30, Washington, DC, 1997.
[36] R. Niels, D. Willems, L. Vuurpijl, The NicIcon database of handwritten icons, in: 11th International Conference on the Frontiers of Handwriting Recognition, Montreal, Canada, 2008.
[37] R.S. Tumen, M.E. Acer, T.M. Sezgin, Feature extraction and classifier combination for image-based sketch recognition, in: ACM Symposium on Sketch Based Interfaces and Modeling, vol. 23, 2010.
[38] C.-C. Chang, C.-J. Lin, LIBSVM: A Library for Support Vector Machines, 2001.
[39] R. Arandjelović, T.M. Sezgin, Sketch recognition by fusion of temporal and image-based features, Pattern Recognition 44 (2011) 1225–1234.

Caglar Tirkaz graduated from Sabanci University in 2005 with a major in Computer Science and a minor in Mathematics. He then completed his master's at Yildiz Technical University in 2008. He is currently pursuing a PhD degree and is a member of computer vision and pattern analysis laboratory at Sabanci University. His research interests are computer vision and pattern recognition.

Berrin Yanikoglu graduated from Bogazici University in 1988 with a double major in Computer Science and Mathematics. She then got her PhD in 1993 from Dartmouth College in the area of handwriting recognition. After a year of postdoctoral research at Rockefeller University, she worked for Xerox Imaging Systems and IBM Almaden Research Center on various aspects of document recognition. She then joined Sabanci University as an Assistant Professor in 2000, where she is currently working. Her research areas are machine learning and pattern recognition with applications in computer vision, especially handwriting recognition and biometrics.

Sezgin graduated summa cum laude with Honors from Syracuse University in 1999. He completed his MS in the Artificial Intelligence Laboratory at Massachusetts Institute of Technology in 2001. He received his PhD in 2006 from Massachusetts Institute of Technology. He subsequently moved to University of Cambridge, and joined the Rainbow group at the University of Cambridge Computer Laboratory as a Postdoctoral Research Associate. Dr. Sezgin is currently an Assistant Professor in the College of Engineering at Koç University, Istanbul. His research interests include intelligent human–computer interfaces, multimodal sensor fusion, and HCI applications of machine learning. Dr. Sezgin is particularly interested in applications of these technologies in building intelligent pen-based interfaces. He is a recipient of the Turkish Scientific and Research Council's Career grant, and currently leads the Intelligent User Interfaces Laboratory at Koç University.