Pattern Recognition 44 (2011) 1225–1234

Contents lists available at ScienceDirect

Pattern Recognition

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

Sketch recognition by fusion of temporal and image-based features

Relja Arandjelović [a,1], Tevfik Metin Sezgin [b,*,1]

[a] University of Oxford, Department of Engineering Science, Oxford, UK
[b] Koç University, College of Engineering, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 27 May 2010
Received in revised form
9 November 2010
Accepted 11 November 2010
Available online 20 November 2010

Keywords:
User interfaces
Sketch recognition
Graphics recognition

ABSTRACT

The increasing availability of pen-based hardware has recently resulted in a parallel growth in sketch-based user interfaces. Sketch-based user interfaces aim to combine the expressive power of free-hand sketching with the processing power of computers. Most sketch-based systems require intelligent ink processing capabilities, which makes the development of robust sketch recognition algorithms a primary concern in the field. So far, the research in sketch recognition has produced various independent approaches to recognition, each of which uses a particular kind of information (e.g., geometric and spatial constraints, image-based features, temporal stroke-ordering patterns). These methods were designed in isolation as stand-alone algorithms, and there has been little work treating various recognition methods as alternative sources of information that can be combined to increase sketch recognition accuracy. In this paper, we focus on two such methods and fuse an *image-based* method with a *time-based* method in an attempt to combine the knowledge of *how objects look* (image data) with the knowledge of *how they are drawn* (temporal data). In the course of combining spatial and temporal information, we also introduce a mathematically well founded fusion method for combining recognizers. Our combination method can be used for isolated sketch recognition as well as full diagram recognition. Our evaluation with two databases shows that fusing image-based and temporal features yields higher recognition rates. These results are the first to confirm the complementary nature of image-based and temporal recognition methods for full sketch recognition, which has long been suggested, but never supported by data.

1. Introduction

Sketching is a natural way of expressing and sharing ideas. It allows us to succinctly convey concepts on paper. These qualities of sketching has caught the attention of many graphics application designers who have started exploring graphics applications that can take advantage of intelligent sketch-based interfaces. In addition, the increasing availability of Tablet PCs and other hardware that support pen-based interaction has led to increased interest in interactive graphics applications that can interpret hand-drawn sketches.

At the core of these interactive sketch-based graphics applications lies the sketch recognition technology. Given a hand-drawn sketch, sketch recognition can informally be defined as the task of finding groups of ink in the sketch that represent individual objects (*segmentation*), and then determining the class of the object represented by each ink group (*object recognition*). So far, researchers have attempted to address both issues within recognition frameworks that mainly differ by the particular kind of information used.

For example, some authors assumed simple definition of drawings and treated icons as gestures. This group of work use distinguishing global features extracted from single or multiple strokes for object recognition [1,2].

Others preferred to define objects using geometric and spatial constraints [3–7]. These *constraint-based approaches* are founded on cognitive science studies which suggest that, when shown a symbol, people attend preferentially to certain geometric features (e.g., a rectangle is formed by two pairs of lines of equal length, and the lines meet with a 90° angle).

Other authors have taken a more computer-vision-like approach to recognition and formulated *image-based algorithms* that use image features such as pixel intensities, and intensity histograms [8–10].

A fourth class of recognition algorithms are based on the temporal stroke-ordering patterns that are naturally used while drawing diagrams [11–14]. The motivation for these *time-based approaches* is based on the observation that when people sketch objects, they use highly characteristic drawing orders (e.g., when drawing a stick figure, most people draw the head first, and then respectively draw the body, the legs and the arms). Hence the stroke-ordering patterns in sketches can be used for sketch recognition.

So far, research efforts have mostly focused on getting the best recognition accuracy with any one of the approaches listed above (gesture, constraint, image, and time-based approaches). Relatively little effort has been spent to explore how various recognition

* Corresponding author.
E-mail addresses: relja@robots.ox.ac.uk (R. Arandjelović), mtsezgin@ku.edu.tr (T.M. Sezgin).
URL: http://iui.ku.edu.tr (T.M. Sezgin).
[1] Part of this work was completed when the authors were at the University of Cambridge.

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.11.006

methods can be used as individual sources of information, and combined to boost sketch recognition accuracy. Specifically, the issue of how temporal recognition methods can be combined with others for segmenting and recognizing complete sketches has not been studied.

This paper is a step in this direction. We focus on combining image-based and time-based recognition methods. We have three main contributions:

- Drawing upon results from combining classifiers, we choose a set of combination methods and evaluate them for combining image-based and time-based recognizers.[2]
- We describe a mathematically well-founded classifier combination method for full sketch recognition (i.e., continuous sketch recognition).
- Using two databases, we show that fusing image-based and temporal features yields better recognition rates compared to using either method alone. These results not only show the virtues of combining multiple recognition methods, but are also the first to show the complementary nature of image and time-based methods for full sketch recognition, which has long been suggested, but never supported by data.

In the rest of this paper, we first describe an image-based recognition algorithm that uses Zernike moments, and a time-based sketch recognition algorithm that uses Hidden Markov Models. In Section 4, we describe five methods for classifier fusion that are subsequently used for fusing image-based and time-based features for isolated symbol recognition. In Section 5, we describe how isolated symbol recognizers can be combined using dynamic programming to simultaneously segment and recognize entire sketches with many symbols. In the evaluation section, first we evaluate the performance of the five classifier fusion methods for isolated symbol recognition using two different databases. Then, we report recognition accuracies of image-based, time-based, and combined recognition methods for recognizing full sketches. We also report the runtime for our recognition and preprocessing algorithms. We conclude with related work and a summary of future research directions.

2. Image-based recognition method: Zernike moments

Although there are many image-based recognition methods, we adopt one based on Zernike moments, which was demonstrated as a simple and effective method for sketch recognition. Our use of Zernike moment features for sketch recognition is based on work by Hse et al. [9], and we refer the reader to this work for the details of feature extraction using Zernike features.

Zernike moments work with bitmap image representations, where the input is represented by a function $f(x,y)$, which is equal to 1 if there is a point at position (x,y), and 0 otherwise. The moments $A_{n,m}$ of $f(x,y)$, are defined over a unit circle as

$$A_{n,m} = \frac{n+1}{\pi}\sum_x\sum_y f(x,y)V_{n,m}(x,y), \quad x^2+y^2 \le 1$$

and $V_{n,m}$ and $R_{n,m}$ are defined as

$$V_{n,m}(x,y) = V_{n,m}(\rho,\theta) = R_{n,m}(\rho)e^{jm\theta}$$

$$R_{n,m}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!}\rho^{n-2s}$$

[2] Although we focus on fusing image-based and time-based recognizers, the fusion methods that we propose generalize to multiple sources of information.

where n (the order of the moment) is a positive integer, m is an integer such that $n-|m|$ is even and $|m| \le n$, $\rho = \sqrt{x^2+y^2}$, and $\theta = \tan^{-1} y/x$. Here the moments $|A_{n,m}|$ normalized by $A_{0,0}$ make good rotation and scale-invariant features. We feed these features into linear SVMs for classification.

3. Time-based recognition method: hidden Markov model (HMM)

Existing time-based methods use either HMMs or Dynamic Bayesian Networks, which generalize HMMs. For our purposes, both approaches are essentially equivalent, hence we use an HMM-based approach.

3.1. Brief introduction to hidden Markov models

Hidden Markov models are used extensively for modeling time-varying signals and processes. Here, we adopt the terminology and notation used in [14]. An HMM is defined by $\lambda(A,B,\pi)$, and specified by three parameters A,B,π. A is the transition probability matrix $a_{ij} = P(q_{t+1} = j | q_t = i)$, B is the observation probability distribution $B_j(v) = P(O_t = v | q_t = j)$, and π is the initial state distribution. $Q = \{q_1, q_2, \ldots, q_N\}$ is the set of HMM states and $V = \{v_1, v_2, \ldots, v_M\}$ is the set of observations symbols.

3.2. Model topology

Model topology defines the overall constraints that are imposed over the connections between the states in an HMM. We would like the HMM states in our models to mimic partially drawn versions of the given symbol. Hence, as more of a symbol is drawn in time, we would like to move on to states corresponding to partial drawings with more strokes. The left-to-right topology framework [15] allows us to achieve this, therefore as in [14], we use a Bakis (left-to-right) topology to model the incremental nature of sketching. This is achieved by setting $a_{i,j}=0$ for each pair of states $j < i$.

To prevent the HMMs from producing high matching scores for partial symbols, a dummy *end-observation* is used to mark symbol completion. During training, the HMMs are trained with sequences corresponding to complete objects, and these sequences are appended with the end-observation before they are passed on to training. We also designate an *end-state* to act as the only state that can generate the end-observation, thus we force all state sequences corresponding to complete objects to finish with the end-observation.

During recognition, all observation sequences passed to the HMMs for scoring are also appended with the end-observation. Naturally, if the observation sequence does not correspond to a complete object, this would force an unlikely Viterbi path with a low probability to emit the end-observation. Hence, state sequences for incomplete (i.e., partially drawn) objects receive very small probabilities if the last state in the sequence is the end-state.

3.3. Observations

We encode sketches to obtain sequences of observations for training HMMs and for classification. Our setup supports discrete as well as continuous observations. This is unlike the general practice in previous work on HMM-based sketch recognition, which has focused on either discrete features or continuous features only. The observations are computed from temporally ordered primitives extracted by fragmenting the input strokes into ellipses, arcs and lines using a stroke fragmentation algorithm.

For all primitives, we compute a feature that captures the length (circumference for ellipses) of the primitive, normalized by the bounding box of the entire symbol to make the system scale-invariant. We also take the angle between the bounding boxes of consecutive primitives to capture rotation. In addition, for ellipses, we store the ratio of the minor and major axes, and a measure of extent for arcs. The observations are modeled using joint density functions of continuous and discrete feature values as described in [16].

3.4. Classification

For each symbol class, we trained a single HMM using the standard EM algorithm. One way of classifying a given observation sequence is to use the likelihood values obtained from each HMM and assign the class of the HMM with maximum response, under the assumption of uniform priors.

This approach has the downside that HMMs are generative models trained only on positive examples, and they are not discriminative. They are trained to respond maximally to a target symbol, but they are not trained to discriminate between different symbols. Therefore, we perform classification by learning a decision rule that maps all HMM responses to object classes using an SVM with an RBF kernel.

4. Fusion of the Zernike moments and HMM methods

The goal of combining multiple information sources is to achieve a superior performance by exploiting the redundant and complementary nature of the information provided by the different sources of information.

The ways in which the information sources can be combined vary depending on the context of the work, and various communities have come up with different taxonomies that emphasize different aspects of the combination operation. For example, in the context of machine learning, one can talk about combining information sources at the *feature level*, or at the *classifier level* [17,18]. In the context of multimodal interaction, one can talk about *early-fusion*, *intermediate-fusion*, and *late-integration* techniques [19]. Our focus in this work is on combining the outputs of multiple classifiers, hence our work fits in the classifier-level fusion category.

One way to achieve classifier fusion is to perform recognition separately with time-based and image-based methods, and then combine the obtained probabilities using a combination method. Based on previous work in classifier fusion [17,18], we chose the following combination methods for experimentation: Mean rule, Dempster–Shafer combination rule and Naïve Bayes.[3]

Another way to combine the methods is to perform the classification using all the HMM outputs and Zernike moments as input features to another classifier (in our case an SVM classifier). Two SVM-based schemes were tested: *one-against-all* and *one-against-one* classification, as explained below. In all cases where we used an SVM, we used facilities of LIBSVM to obtain probability estimates [21].

4.1. Mean combination rule

Mean of estimates obtained from the two methods is used to estimate symbol probabilities. Although this method is simple, it was shown to be resilient to probability estimation errors of the methods to be combined [17].

[3] Also see [20] for a detailed taxonomy of multiple classifier decision combination strategies for character recognition. Our work fits in the "analytical combination methods" and "horizontal decision combination" categories in that taxonomy. The combination methods we study here also include applicable strategies listed in [20].

4.2. Dempster–Shafer combination rule

The Dempster–Shafer theory is used to determine the belief in certain propositions based on evidence that supports them. The belief is expressed as a numeric value between 0 and 1 and the Dempster–Shafer combination rule [18] is used to combine two belief functions:

$$B(S) = \frac{P_{hmm}(S)P_{zer}(S)}{1-\prod_{S_1 \neq S_2} P_{hmm}(S_1)P_{zer}(S_2)}$$

where S is a symbol class, B is the combined belief function, and P_{hmm} and P_{zer} are the methods' probability distributions, which for the purposes of the Dempster–Shafer combination are considered to be the belief functions. Since in general $\sum_S B(S) \neq 1$, the combined belief function is normalized to obtain a proper probability distribution.

4.3. Combination with Naïve Bayes

Naïve Bayes is a commonly used combination method that assumes the used classifiers are independent, which we will assume to be the case here. Two methods that we use have different classification frameworks, and they operate on features that are very different in nature (appearance vs. ordering). Therefore, we believe the independence assumption is reasonable.

The conditional probabilities $P_{method}(S|D)$, where S is a symbol class and D is the decision symbol of the method in question, are estimated from the training data for each method. They are then used to estimate the posterior probability of the symbol using the independence assumption:

$$P(S) = \frac{1}{C} \cdot P_{hmm}(S|D_{hmm}) \cdot P_{zer}(S|D_{zer})$$

where S is a symbol class, P is the combined probability distribution, D_{hmm} and D_{zer} are the methods' symbol class decisions, P_{hmm} and P_{zer} are the methods' conditional probability distributions, and C is a normalizing constant. C is computed such that the marginal of $P(S)$ over the symbol classes sums up to 1. This normalization is standard in applications of Naïve Bayes.

4.4. SVM: one-against-all (OAA)

For each symbol, an SVM was trained to discriminate it from all others, as proposed in [22]. The maximally responding SVM gives the symbol class.

4.5. SVM: one-against-one (OAO)

An SVM was trained to discriminate each pair of symbol classes. This was done using LIBSVM [21], which combines the outputs of all the pairwise classification results. The specifics of the algorithm that combines the decisions from pairwise classifiers is described in [23].

5. Segmentation and recognition of full diagrams

A major problem in sketch recognition is segmentation: partitioning a sketch into groups of ink that represent individual symbols. Knowing the correct segmentation of a sketch immensely simplifies recognition. Therefore, some methods force the users to explicitly specify when they finish drawing each symbol making the system less usable, which defeats the main motivation behind sketch-based interfaces.

One could argue that sketch segmentation should not be considered separate from individual symbol recognition. Instead of segmenting the sketch first, and then recognizing the individual symbols, it makes more sense to interleave segmentation and recognition, since after all, the segmentation step should "know" which set of strokes look like a symbol, which is tightly related to its recognition. Also, proper segmentation should generate groups of ink, each of which represents a valid complete symbol, and we need recognizers for the verification. We approach the problem of recognition and segmentation by generating many recognition hypotheses for small fragments of the input sketch, and then combining compatible hypotheses to obtain a globally optimal fragmentation and recognition hypothesis.

5.1. Proposed method for segmentation and recognition

The proposed method is a modification of the method described in [14]. We construct a graph $G(V,E)$ in which vertices V correspond to the fitted primitives indexed by the order in which they were drawn. The weight $w(i,j)$ associated with an edge from v_i to v_j in G corresponds to the probability that the set of primitives between i inclusive and j exclusive correspond to a valid and fully drawn symbol, or more formally:

$$w(i,j) = \max_S P_{i,j}(S) \tag{1}$$

where $P_{i,j}(S)$ corresponds to the probability that primitives with indices between i inclusive and j exclusive correspond to the symbol S. This formulation sets the constraint that a symbol needs to be fully drawn before starting another symbol (in other words, no *interspersing* is allowed). The optimal segmentation is computed through dynamic programming:

$$S(i,j) = \max \begin{cases} w(i,j) \\ \max\limits_{i \le k < j} (S(i,k) \cdot S(k,j)) \end{cases} \tag{2}$$

where $S(i,j)$ is the probability of the optimal segmentation of a subsketch made of primitives with indices between i inclusive and j exclusive. Thus, the optimal segmentation of the entire sketch has the probability $S(1,|V|+1)$, and the segmentation can be reconstructed in a way that is common for many algorithms that use dynamic programming: during the calculation of $S(i,j)$ the choices made along the way (i.e., the value of k or a value indicating $w(i,j)$) are stored in a separate matrix, and this matrix is sufficient to completely determine the optimal segmentation.

The above setup effectively combines smaller recognition hypotheses into larger ones, and eventually finds the optimal segmentation of the entire sketch. The initial hypotheses are formed in accordance with Eq. (1), and then combined into larger hypotheses in accordance with Eq. (2). Given the observation sequence corresponding to the entire sketch, for each class, initial hypotheses can be generated using Eq. (1) for all subsequences of length l, where $l_{min} \le l \le l_{max}$, l_{min}, and l_{max} are the assumed length of the shortest and longest observation sequences in the entire set of training examples for that class. All subsequences with lengths outside these bounds are given 0 weight by setting $P_{i,j}(S)=0$ if $j-i < l_{min}$ or $j-i > l_{max}$.

5.2. Probability estimates $P_{i,j}(S)$

The posterior probability estimates returned by the time-based and image-based classifiers described in Sections 2 and 3.4, as well as the values returned by the fusion method in Section 4 model the probability that the data represents a particular symbol given that it represents a valid symbol. Therefore these are in fact conditional probabilities of the form $P_{i,j}(S|\text{valid symbol})$, and should not be directly substituted for $P_{i,j}(S)$ in Eq. (1). If we knew $P_{i,j}(\textit{valid symbol})$ we could calculate the required probabilities using

$$P_{i,j}(S,\text{valid symbol}) = P_{i,j}(S|\text{valid symbol}) \cdot P_{i,j}(\text{valid symbol})$$

We do not calculate $P_{i,j}(\textit{valid symbol})$ directly, but train a one-class SVM [24] based on all examples of valid symbols to obtain a binary approximation. We set $P_{i,j}(\textit{valid symbol})$ to 1 if the one-class SVM decides the data represents a valid symbol, otherwise it is set to 0. This can also be thought of as a filtering stage, where invalid symbols are filtered out using the one-class SVM and assigned a probability of 0, while the valid ones that go past the filtering are recognized using any of the considered methods.

5.3. Choice of the ν parameter

In order to train the one-class SVM, one must set a parameter ν that controls how different an example has to be from a known valid symbol in order to be considered invalid [24]. At the same time it corresponds to the proportion of valid symbols which are deemed invalid by this method [24]. This parameter takes a value in the range [0,1]. A small value means that few valid symbols will be marked invalid but many invalid symbols will be marked valid, whereas a large value means the converse. Value of the ν parameter is crucial to the performance of the recognition and segmentation algorithm. We preferred not to reject symbols aggressively during filtering, hence assigned a small value to ν. This is because there are at least two more opportunities for filtering out invalid symbols (e.g., they can receive low score from the time-based method or the image-based method).

6. Evaluation of the proposed methods

Our evaluation included measuring the accuracy of isolated object and complete sketch recognition. We also measured the time required for processing each additional stroke, including the time required to fragment the stroke, and the amount of time required for updating the recognition and segmentation hypotheses for each added stroke.

6.1. Evaluation data

We have evaluated our recognition system with two databases. The first database includes symbols from our Course of Action Diagrams database, and the other database is the publicly available NicIcon database [25].

6.1.1. Course of Action Diagrams database

Symbols in this database come from the domain of military *Coarse of Action Diagrams* [26] shown in Fig. 2. A complete list of objects in this domain is listed in the US Army Field Manual 101-5-1. Among hundreds of symbols in this domain, we focus on a subset of 20 for practical reasons.

Fig. 1 shows examples of computerized versions of Course of Action Diagrams. As seen in these examples, the figures consist of a map, which defines the background, and the foreground symbols. The symbols often appear very close to one another, and also overlap with drawings that define landmarks such as boundaries, frontiers, etc. This makes it difficult to bypass the problem of segmentation by context free preprocessing. Also as reported in [28], in general simple spatial and temporal grouping approaches do not work in sketch recognition. Our methods with the non-interspersed drawing assumption allows us to deal with these complications.

Some symbols in this domain are quite distinct, while some others look similar. For example, an *Enemy Artillery Observation Unit* (Fig. 2(d)) is the same as Fig. 2(g) with an added small circle in the middle.

Eight different users were shown symbol images picked randomly and asked to sketch examples from each of the 20 symbol classes. In total 620 examples of different symbols were collected. The number of examples per symbol varied between 27 and 45, with a median of 30.

The recognition and segmentation of full diagrams were tested on diagrams consisting of individual objects following a common practice for gesture recognition and handwriting recognition [1]. In total, there are 200 diagrams, each with 3–6 symbols (20 of the diagrams had 3 symbols, 80 had 4, 60 had 5 and 40 had 6). Each symbol was randomly chosen from the entire testing set and positioned randomly in the diagram. Hand drawn symbol examples are shown in Fig. 3 for a subset of the symbols.

6.1.2. The NicIcon database

The NicIcon database includes multi-stroke symbols used in the context of an emergency management application (e.g., symbols for fire, injury [1]). This database contains a total of 23 641 symbols distributed into 14 classes (Fig. 4). The symbols in the database consisted of an average of 5.2 strokes, and the average number of strokes for the individual categories ranged from 3.1 to 7.5.

6.2. Training and testing methodology

For the Course of Action Diagrams, the symbol examples were split randomly into two sets: the training set with 80% of the examples and the testing set with the remaining 20%. The training was done only using the data in the training set while the testing was done only using the testing set. All the recognition rates quoted refer to the recognition rates for the test set. The random splitting of the data, and training/testing using this data were repeated 8 times and the recognition rates were averaged in order to account for particularly lucky or unlucky training/testing splits of the data.

For the NicIcon database, we followed the writer-independent data split proposed by Vuurpijl, where 40% of the data was used for validation [25]. Thus all results on this database are for the writer-independent setting.

The recognition accuracy is measured as the percentage of correctly classified examples of each object class, averaged across all the symbols in order to account for data imbalance (i.e., the recognition rate for the symbol with 45 examples contributes as much to the overall recognition rate as the one for the symbol with 27 examples).

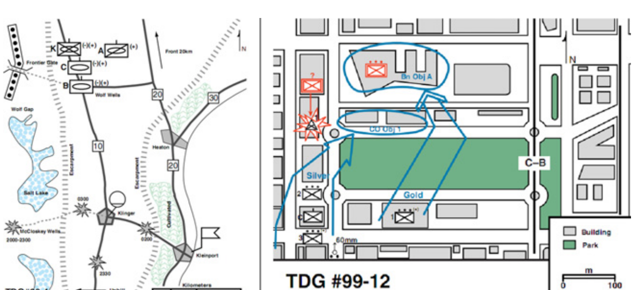

Fig. 1. Two examples of Course of Action Diagrams from the tactical games published in the Marine Corps Gazette [27].

6.3. Isolated symbol recognition results

Isolated symbol recognition assumes the scene contains only a single object.

6.3.1. HMM-based methods

The test results in Fig. 5a and b show the recognition accuracy of the HMM-based methods for our databases. They confirm that people do tend to sketch symbols in certain ways and that a time-based approach can yield good recognition rates for isolated object recognition.

Notably, these results show that HMM-based methods yield good performance irrespective of the writer-dependency of the data, because the results for the NicIcon database are for a writer-independent data split. In fact, as we later present in Table 2, for this writer-independent setup, the HMM-based methods can outperform the Zernike-based methods.

Fig. 5a and b also show that the SVM classification method with the HMM outputs as its input features substantially and consistently outperforms the common method which just assigns the symbol class by simply considering the HMM with the largest likelihood, as predicted in Section 3.4.

6.3.2. Zernike method

The results in Fig. 5 confirm that Zernike moments can be used successfully in sketch recognition. For low orders, Zernike moments result in accuracies comparable to those obtained using HMMs. Higher orders generally yield superior performance. Nevertheless, as we show below the performance is further boosted when the two methods are combined.

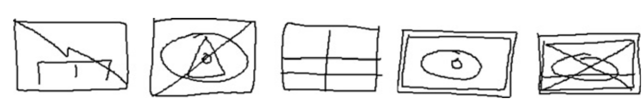

Fig. 3. Examples of hand drawn Course of Action symbols used in our evaluation.

Fig. 4. Examples of hand drawn icons from the NicIcon database [1]. From top-to-bottom and left-to-right, the symbols represent *fire brigade*, *gas*, *roadblock*, *injury*, *paramedics*, *police*, *accident*, *bomb*, *fire*, *car*, *person* and *flood*.

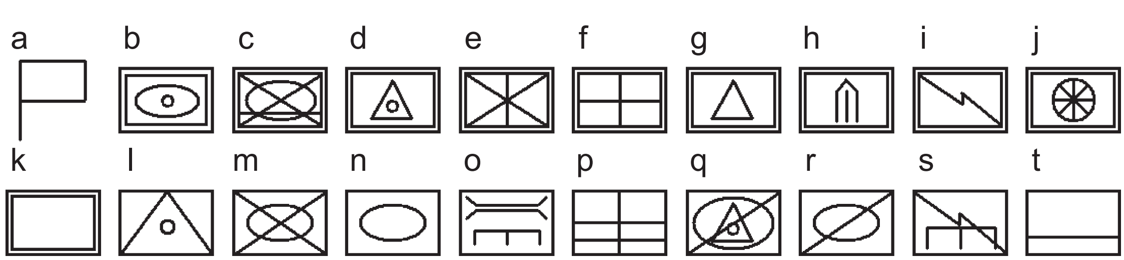

Fig. 2. The subset of Course of Action Diagram symbols used in our evaluation.

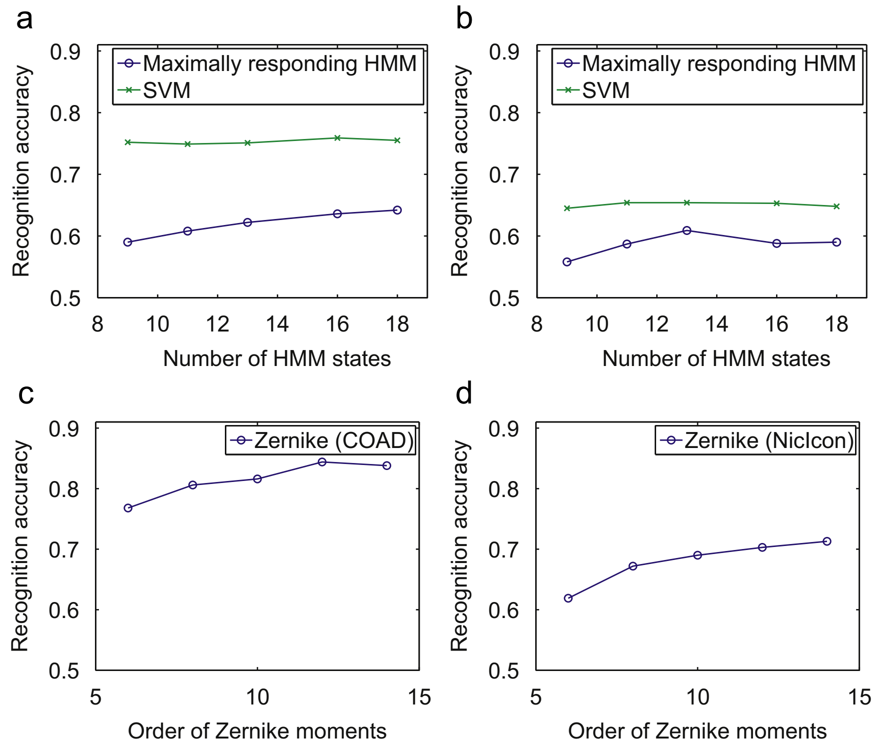

Fig. 5. The recognition rates achieved using the HMM-based and Zernike-based methods for the Course of Action (a, c) and the NicIcon databases (b, d). Values are listed for various choices of the relevant free parameter for each method—namely the *number of states* for HMMs, and the *order* for the Zernike-based method. See [14,9] for a detailed discussion of the choice of free parameters.

Table 1
Recognition rates obtained using different recognition methods. The number of HMM states is set to 16, and order of Zernike moments is 12.

	Method name	Database	
		COAD	NicIcon
Individual methods	HMM	0.636	0.588
	HMM with SVMs	0.759	0.653
	Zernike moments	0.844	0.703
Combined methods	Mean	0.872	0.786
	Dempster–Shafer	0.880	0.831
	Bayesian	0.864	0.777
	OAA-SVM	0.843	0.760
	OAO-SVM	0.864	0.814

Table 2
The recognition rate of the OAO-SVM combination method (Combination) is good even with small orders of Zernike moments (z) and number of HMM states (h).

Parameters	HMM (%)	HMM with SVM (%)	Zernike (%)	Combination (%)
Course of Action database				
$z=6$, $h=9$	59.0	75.2	76.8	85.3
$z=8$, $h=11$	60.8	74.9	80.6	85.0
$z=10$, $h=13$	62.2	75.1	81.6	86.0
$z=12$, $h=16$	63.6	75.9	84.4	86.4
$z=14$, $h=18$	64.2	75.5	83.8	87.0
NicIcon database				
$z=6$, $h=9$	55.8	64.5	61.9	78.6
$z=8$, $h=11$	58.7	65.4	67.2	80.1
$z=10$, $h=13$	60.9	65.4	69.0	81.1
$z=12$, $h=16$	58.8	65.3	70.3	81.4
$z=14$, $h=18$	59.0	64.8	71.3	81.2

6.3.3. *Fusion methods*

Further improvement in recognition accuracies is possible by combining the Zernike moment and HMM-based methods. Test results in Table 1 show that with the exception of the OAA-SVM method, all ways of combining time-based and image-based algorithms gives us better results than what we can achieve with each method taken individually.[4] Of the SVM-based methods, OAO-SVM has superior performance, though this comes with a cost. In particular, in the OAO the number of classifiers that need to be trained increases quadratically with respect to the number of object classes, while the increase is linear in the OAA case.

The improvement in recognition accuracies may appear to be incremental, nevertheless the corresponding reductions in the error rates are notable. In sketch based interfaces, correcting each misclassification requires effort on the part of the user and gets in the way of completing the main task. The relative error reduction rates for the Course-of-Action database lie around the 20–25% mark for the Mean and Dempster–Shafer combination methods, and above 37% for the NicIcon database. Hence, the improvements due to combining temporal and image-based features are substantial when considered in the context of a sketch-based user interface.

More importantly, as seen in Table 2, combining the two methods provides very good results even with considerably smaller order of Zernike moments and fewer HMM states. The relative error reduction rates for these parameter settings are listed in Table 3. As shown in this table, the combination method provides roughly 13–44% improvement over Zernike moments alone, 24–47% improvement over using HMMs with SVMs, and 52–64% improvement over using

[4] Note that the OAA method has been found to yield inferior performance for a variety of other practical multiclass classification tasks as well (e.g., digit recognition, image classification, and agricultural applications [29]).

Table 3
The relative error reduction rates for pairs of recognition methods under various choices of Zernike moments (z) and number of HMM states (h). Combining time- and image-based methods results in substantial reductions in relative error rates.

Parameters	HMM vs. HMM with SVM (%)	HMM vs. combination (%)	HMM with SVM vs. combination (%)	Zernike vs. combination (%)
Course of Action database				
$z=6$, $h=9$	39.51	64.15	40.73	36.64
$z=8$, $h=11$	35.97	61.73	40.24	22.68
$z=10$, $h=13$	34.13	62.96	43.78	23.91
$z=12$, $h=16$	33.79	62.64	43.57	12.82
$z=14$, $h=18$	31.56	63.69	46.94	19.75
NicIcon database				
$z=6$, $h=9$	19.68	51.58	39.72	43.83
$z=8$, $h=11$	16.22	51.82	42.49	39.33
$z=10$, $h=13$	11.51	51.66	45.38	39.03
$z=12$, $h=16$	15.78	54.85	46.40	37.37
$z=14$, $h=18$	14.15	54.15	46.59	34.49

HMMs alone for the Course of Action and NicIcon databases. All these results suggest that even though one can achieve better recognition rates by improving individual classifiers and features used by these classifiers, when the methods are combined, the combination always outperforms the individual methods. Furthermore, these results show that when taken individually, image-based and time-based methods provide quite distinct kinds of information about the drawn symbol, and their combination successfully uses all the available information resulting in a more accurate recognition method.

Another notable observation based on the results in Table 2 is that, combining image-based and time-based methods improves performance irrespective of the writer-dependency of the data. In particular, the results for the NicIcon database are for a writer-independent setup, and yet the performance is substantially improved by combining the two methods.

Numbers in Table 3 also illustrate the contribution of feeding HMM probabilities to the SVM classifier. As seen in the first columns in Table 3, up to 40% reductions are obtained in the error rates when the HMM scores are collectively sent to a classifier, as opposed to taking the classification suggested by the HMM with the best score.

6.4. Segmentation and recognition of full diagrams

For the full-diagram recognition tests, we used the Course of Action Diagram dataset, and employed the OAO-SVM combination algorithm to estimate the fragment probabilities $P_{i,j}(S|\text{valid symbol})$ (see Section 5.2), because it is analogous to the estimation of $P_{i,j}(\textit{valid symbol})$ (Section 5.2).

Before our segmentation algorithm can be run, the ν parameter should be set. In Sections 5.2 and 5.3, we introduced a single class classification scheme for filtering out invalid symbols. This filtering had two goals: (1) keep the number of cases where invalid symbols are marked as valid low, (2) keep the number of cases where valid symbols are marked as invalid low. Therefore it is important to set the ν parameter such that a good tradeoff is achieved between the two goals. As discussed earlier, a small ν value means that few valid symbols will be marked invalid but many invalid ones will be marked valid, whereas a large value means the converse. Fig. 6 shows the recognition accuracy for full-diagrams, for various choices of ν in the Course of Action Diagrams dataset. As seen in the graph, the optimal value for ν lies between the two extremes, and favors a larger false positive rate to a large false negative rate. This supports earlier discussion in Section 5.3.

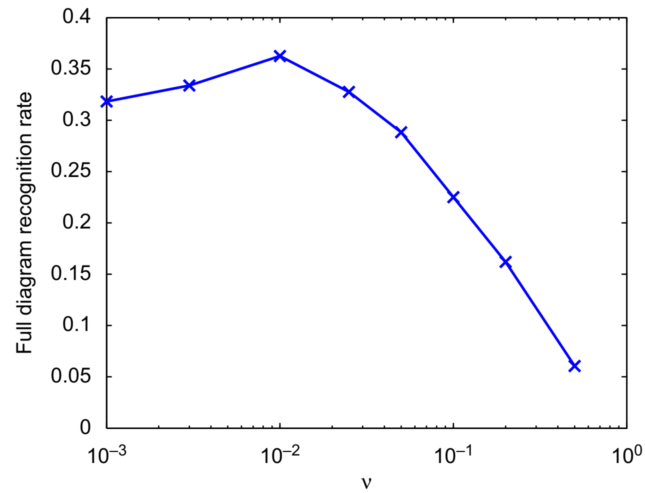

Fig. 6. Segmentation and recognition of full diagrams, with single symbol recognition rate of $r_S=0.8640$ (HMM states = 16, order of Zernike moments = 12).

The full diagram accuracy for our system is $r_D=0.363$ for 12 orders of Zernike moments, 16 HMM states and $\nu=0.01$. When judging the adequacy of full diagram recognition rates, it is critical to remember that in full diagram recognition, a recognition hypothesis is counted as a misrecognition even if all but one of the several symbols in the diagram are correctly recognized. In addition, errors can occur due to misrecognition of individual objects, as well as due to segmentation errors.

We carried out further analysis to gain an insight on the breakdown of the overall error into segmentation and recognition errors. For a mixture of full diagrams where 10% have 3 symbols, 40% have 4, 30% have 5 and 20% have 6 symbols, we computed the recognition rate with the assumption of perfect recognition to be 0.515. Using this estimate, the percentage of errors that could have been avoided with perfect segmentation was computed to be $(0.515-0.363)/(1-0.363)=0.239$. Therefore, roughly 24% of the diagram recognition errors can be avoided in interfaces where users explicitly or implicitly specify the perfect segmentation (e.g., by pausing or pressing a button to specify object boundaries).

6.5. Contribution of modeling object completions

As mentioned in Section 3.2, to prevent the HMMs from producing high matching scores for partially drawn symbols, we used a dummy *end-observation* to mark symbol completions. In order to assess the contribution of modeling object completions in the full diagram recognition rates, we ran a series of tests where we measured the full diagram recognition accuracies using HMMs that do not explicitly model object completions, using the same testing methodology described in the previous section.

Our tests showed a decrease in the full diagram recognition rates from 36.3% to 32.0%. This is a 11.8% reduction, which we believe is substantial. In other words, without end-state modeling, 11.8% of objects that would otherwise be correctly recognized are misrecognized.

6.6. Evaluation of runtime performance

We measured the time required to fragment each added stroke, and the amount of time required for updating the recognition and segmentation hypotheses after each added stroke.

Fig. 7a shows the amount of time taken for fragmenting each successive stroke computed for the full sketches used in

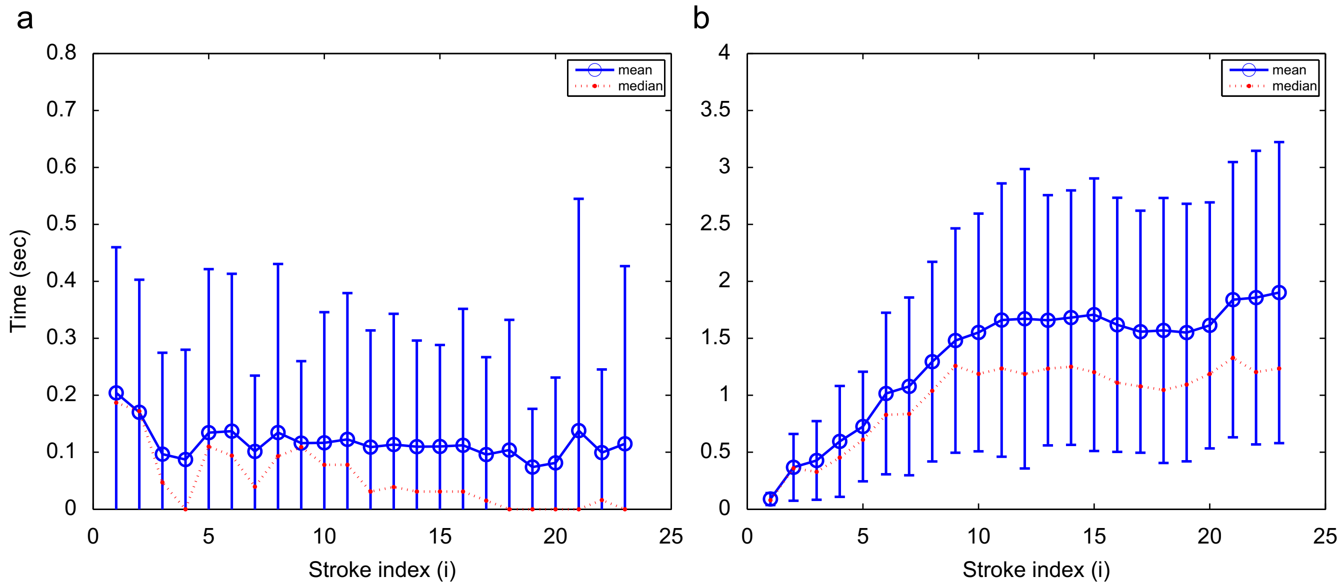

Fig. 7. The incremental time taken for stroke fragmentation and object recognition for increasing number of strokes.

the evaluation of the Course of Action database in Section 6.4. As seen in this figure, the average time required for processing each stroke is roughly constant, but varies substantially across strokes.

Fig. 7b shows the incremental time taken for updating the recognition and segmentation hypotheses after the ith stroke is added in each one of the 200 sketches used in Section 6.4, computed as in [4,6]. The horizontal axis shows the stroke index i. As seen in this figure, the average time required for processing each stroke hits a stable plateau after the first 10–12 strokes. This is because, the marginal time complexity of the dynamic programming operation used in recognition and segmentation is constant with respect to the number of strokes. In particular, for each new primitive, one new node, and at most $O(n \times k)$ arcs are added to the shortest path graph G, where n is the number of objects in the domain, and $k = l_{max} - l_{min}$. Computing the shortest path to the node corresponding to the new primitive takes $O(n \times k)$ operations. Since n and k are constant for a given domain, and database, the marginal cost of each added primitive is constant. This analysis as well as the numbers shown in Fig. 7b are good indicators of the practicality of our strategy for realtime recognition in online sketching, especially considering that the numbers are from our unoptimized Java implementation executed on a low-end PC.

In both time measurement figures (Fig. 7a and b), we have included the mean as well as the median processing times. This is because, as it was also reported in [6], processing times usually have high variances and the median values serve as better indicators of the processing time.

7. Related work

In this paper, we fused an image-based method with a time-based method in an attempt to combine the knowledge of how objects look (appearance) with the knowledge of how they are drawn (stroke orderings). We focused on appearance and stroke orderings because they not only represent conceptually different aspects of sketching, but they have also been shown to aid recognition individually. However, our combination method is general and any method producing probabilistic confidence values can be used in place of – or in addition to – the two methods presented here (e.g., fully probabilistic variants of constraint-based or structural methods as in [30–33]).

The image-based method that we adopted here was suggested by Hse et al. [9]. There are many other image-based methods producing probabilistic confidence values that could have been substituted in place (e.g., [10,30,34] or a fully probabilistic version of [8]).

Although there are a number of time-based methods for sketch and gesture recognition [11,13,14,35], the most relevant one is the HMM-based method described in [14]. Unlike [14] we use discrete continuous observations, while the original paper uses discrete observations only, and does not use length information. Also, we introduced the SVM classification stage which resulted in a 32–40% reduction in the error rates. Finally, the way we model "end states" differs from this line of work. Our method allows us to avoid the extra bookkeeping required for modeling object completions in [14]. Hence, our method is also easier to implement.

There are isolated symbol recognition systems that work with presegmented input (e.g., [9,36]), or systems that use domain knowledge for segmenting symbols in a preprocessing step [37]. Our fusion framework performs segmentation as well as recognition. Hence we limit our discussion here to systems that can do full diagram recognition (continuous sketch recognition). There are other pieces of work that offer solutions for full sketch recognition under different assumptions. For example, while we make the assumption that users draw objects one at a time, and stay within the time-efficient framework of dynamic programming, others have suggested systems combining constraint-based recognition schemes with indexing and constrained optimization, relaxing the assumption that objects are drawn without temporal interspersing of different strokes [31]. Similarly, there are image-based approaches for joint segmentation and recognition based on graphical models (e.g., [38,39]). Again, these approaches do not incorporate temporal information. A complete segmentation/recognition framework that incorporates temporal information and allows interspersed strokes is described in our previous work [35]. However that work is based on dynamic Bayesian networks, and uses temporal information only. Another method for segmentation and recognition has been suggested by Widmayer et al. [7]. They describe a combinatorial approach to recognition, and perform a shortest path search as described in our previous work [14] to obtain the best segmentation. However unlike ours, their system cannot be trained [7], they do not use temporal features, and they also impose stroke fragmentation unpractical requirements.[5]

To our knowledge, the method described in [40] is the first to use dynamic programming for simultaneous sketch segmentation and recognition. However it uses temporal information only. Similarly, there are pieces of later work that use dynamic programming with temporal features only [14] or image-based features only [41]. In this paper, we illustrate how multiple kinds of information can be fused together.

There are lines of work where the possibility of combining classifiers have been explored. For example, Kara et al. [8] combine four image-based classifiers, each of which defines a distance metric between a drawn symbol and learned templates. This is an example of combining multiple classifiers, though the authors use only the "mean-rule" to combine the outputs of the classifiers.[6] Here, we focus on combining classifiers that use information sources of substantially different character (i.e., temporal and image-based). We also present a comparison of many combination rules, including the Mean rule, Dempster–Shafer combination rule, Naïve Bayes in addition to two SVM-based combination rules for combining classifiers.

There are also pieces of work where various kinds of features have been combined together within the constraint-based recognition framework. For example, Widmayer et al. [7] and Cheriet et al. [5] describe recognition systems where constraints (as opposed to classifiers) based on geometric measurements are combined using various mean-based rules. Anquetil et al. describe a framework that makes it possible to incorporate "statistical recognition" in a constraint multiset grammar-based recognition framework [6]. Although it is not clear what kinds of statistical recognizers can be incorporated into this framework, and how they would be combined, the proposed method offers a way of coupling structural and statistical information. As it was the case for other pieces of work described above, our work differs by the virtue that we focus on classifier combination and evaluate the effectiveness of multiple combination methods for fusing temporal and image-based information sources. Also, as it is typical of other approaches that are based on grammars, languages, and constraints, this approach requires one to manually specify object definitions, whereas our models are fully trainable.

Although sketch recognition is intrinsically a different problem compared to handwriting recognition, some of the work is this area is also relevant. There are many handwriting recognition systems that combine structural and statistical approaches for recognition [42,43], but the most relevant ones are those that combine off-line and on-line information (e.g., work by Neskovic et al. [44], Nakagawa et al. [45,46]). Similar to our findings, these papers report an increase in the recognition rates. However, these systems often use only a subset of the combination rules studied here (e.g., [45,44]), or consider isolated character/word recognition (e.g., [45,46]), which is analogous to isolated gesture and symbol recognition, as opposed to the recognition and segmentation of complete sketches. The analog of recognizing complete sketches is handwritten text line recognition [47,48], hence this is the most relevant line of work for us. Text line recognition in the context of combining multiple classifiers differs from isolated character or word recognition, because the output of a text line recognizer is a sequence of word classes rather than just a single word, and the number of words hypothesized for a given line of text may differ across recognizers [49]. Similarly sketch recognizers output lists of recognized objects, and different methods may come up with recognition hypotheses with different numbers of classes. Two pieces of work for text line recognition are described in [50,49]. Both of these systems combine on-line and off-line information. On the other hand, unlike the dynamic programming framework that we use, they use an alignment procedure based on the so-called ROVER combination strategy, which uses a word transition network followed by a voting step. Because this strategy uses voting, it is more appropriate for domains with many independent classifiers (experts); hence it does not suit our case with only two experts.

[5] They assume perfect stroke fragmentation is available, which is rarely the case even in the simplest sketches.

[6] Also, the work in [8] uses a greedy approach for segmentation, while we use an approach based on dynamic programming, which is not susceptible to making locally plausible decisions that lead to globally poor recognition results.

8. Future work

The HMM-based method can be improved using sophisticated features computed from ink groups or image patches. Hence features do not necessarily have to be primitive-based. For example, carefully designed features based on shape contexts [51], congealing [52] or other local descriptors can be used. Furthermore *feature engineering* and *feature selection* techniques, which are outside the scope of our contribution here, can be used to boost accuracy.

As discussed in Section 3.4, one drawback of the HMM-based method is that HMMs are trained generatively using only the positive examples of the symbol. It might be better if the HMMs could be trained to discriminate between symbols directly, which might remove the need for SVM classification of the HMM outputs. One way of doing this could be to use maximum mutual information for HMM parameter estimation [53]. Another approach could be to use discriminatively trained HMMs as described in [54].

9. Summary

In this paper, we presented a framework for fusing an image-based method with a time-based method in an attempt to combine the knowledge of how objects look (image data) with the knowledge of how they are drawn (temporal data). This is unlike most existing approaches, which focus on one kind of feature only. We presented evaluation results for two databases illustrating that combining classifiers yields higher recognition accuracies, and confirmed the complementary nature of image-based and temporal recognition methods for full sketch recognition, which was suggested in the past, but never supported by data. We presented a mathematically well founded method for segmenting and recognizing entire sketches, and demonstrated how it can be used to combine different kinds of recognizers.

References

[1] D. Willems, R. Niels, M. van Gerven, L. Vuurpijl, Iconic and multi-stroke gesture recognition, Pattern Recognition 42 (12) (2009) 3303–3312.
[2] D. Rubine, Specifying gestures by example, SIGGRAPH Computer Graphics 25 (4) (1991) 329–337.
[3] T. Hammond, R. Davis, LADDER, a sketching language for user interface developers, Computers and Graphics 28 (2005) 518–532.
[4] C. Alvarado, R. Davis, SketchREAD: a multi-domain sketch recognition engine, in: UIST '04, ACM, New York, NY, USA, 2004, pp. 23–32.
[5] J.-P. Valois, M. Cote, M. Cheriet, Online recognition of sketched electrical diagrams, in: ICDAR '01, September 10–13, 2001, pp. 460–464.
[6] S. Mac, E. Anquetil, Eager interpretation of on-line hand-drawn structured documents: the DALI methodology, Pattern Recognition 42 (12) (2009) 3202–3214.
[7] A. Hall, C. Pomm, P. Widmayer, A combinatorial approach to multi-domain sketch recognition, in: Eurographics Workshop on Sketch-based Interfaces and Modeling, 2007, pp. 7–14.
[8] L. Kara, T. Stahovich, An image-based trainable symbol recognizer for sketch-based interfaces, in: AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural, 2004, pp. 501–517.
[9] H. Hse, A.R. Newton, Sketched symbol recognition using Zernike moments, in: Proceedings of International Conference on Pattern Recognition, vol. 1, 2004, pp. 367–370.
[10] M. Oltmans, Envisioning sketch recognition: a local feature based approach to recognizing informal sketches, Ph.D. Thesis, MIT, Cambridge, MA, May 2007.

[11] S. Simhon, G. Dudek, Sketch interpretation and refinement using statistical models, in: Eurographics Symposium on Rendering, 2004, pp. 23–32.
[12] W. Jiang, Z.-X. Sun, HMM-based on-line multi-stroke sketch recognition, in: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 7, August 2005, pp. 4564–4570.
[13] D. Anderson, C. Bailey, M. Skubic, Hidden Markov model symbol recognition for sketch-based interfaces, in: AAAI Fall Symposium Series Making Pen-Based Interaction Intelligent and Natural, 2004, pp. 15–21.
[14] T.M. Sezgin, R. Davis, HMM-based efficient sketch recognition, in: Proceedings of the 10th International Conference on Intelligent User Interfaces, ACM, New York, NY, USA, 2005, pp. 281–283.
[15] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.
[16] L. Rabiner, B.-H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
[17] J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (3) (1998) 226–239.
[18] L. Xu, A. Krzyzak, C. Suen, Methods of combining multiple classifiers and their applications to handwriting recognition, IEEE Transactions on Systems, Man and Cybernetics 22 (3) (1992) 418–435.
[19] A. Jaimes, N. Sebe, Multimodal human–computer interaction: a survey, Computer Vision and Image Understanding 108 (1–2) (2007) 116–134.
[20] A. Rahman, M. Fairhurst, Multiple classifier decision combination strategies for character recognition: a review, International Journal on Document Analysis and Recognition 5 (July) (2003) 166–194.
[21] C.-C. Chang, C.-J. Lin, *LIBSVM*: A Library for Support Vector Machines, 2001. Software available at 〈http://www.csie.ntu.edu.tw/~cjlin/libsvm〉.
[22] V. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.
[23] H. Lin, C. Lin, R. Weng, A note on Platt's probabilistic outputs for support vector machines, Machine Learning 68 (3) (2007) 267–276.
[24] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. Smola, R. Williamson, Estimating the support of a high-dimensional distribution, Neural Computation 13 (7) (2001) 1443–1471.
[25] L. Vuurpijl, The NicIcon database 〈http://unipen.nici.ru.nl/NicIcon/〉.
[26] F. Manual, 101-5-1, Operational Terms and Graphics, Washington, DC, Department of the Army, vol. 30, 1997.
[27] O.E. The Marine Corps Gazette, Tactical decision games symbols 〈http://www.mca-marines.org/gazette/tdgsym.asp〉.
[28] C. Alvarado, M. Lazzareschi, Properties of real-world digital logic diagrams, in: Proceedings of the First International Workshop on Pen-Based Learning Technologies, 2007, pp. 12–24.
[29] J. Chen, C. Wang, R. Wang, Adaptive binary tree for fast SVM multiclass classification, Neurocomputing 72 (13–15) (2009) 3370–3375.
[30] M. Shilman, H. Pasula, S. Russel, R. Newton, Statistical visual language models for ink parsing, in: Proceedings of the AAAI Spring Symposium on Sketch Understanding, 2002, pp. 126–32.
[31] T.A. Hammond, R. Davis, Recognizing interspersed sketches quickly, in: Proceedings of Graphics Interface, Toronto, Ontario, Canada, Canadian Information Processing Society, 2009, pp. 157–166.
[32] J. Mas, G. Sanchez, J. Llados, B. Lamiroy, An incremental on-line parsing algorithm for recognizing sketching diagrams, in: International Conference on Document Analysis and Recognition, 2007, pp. 452–456.
[33] D. Sharon, M. van de Panne, Constellation models for sketch recognition, in: Eurographics Workshop on Sketch Based Interfaces and Modeling, 2006, pp. 19–26.
[34] M. Shilman, P. Viola, Spatial recognition and grouping of text and graphics, in: Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2004, pp. 91–95.
[35] T. Sezgin, R. Davis, Sketch recognition in interspersed drawings using time-based graphical models, Computers and Graphics 32 (5) (2008) 500–510.
[36] S. Cates, Combining representations for improved sketch recognition, Ph.D. Thesis, Massachusetts Institute of Technology, September 2009.
[37] L. Gennari, L.B. Kara, T.F. Stahovich, K. Shimada, Combining geometry and domain knowledge to interpret hand-drawn diagrams, Computers and Graphics 29 (4) (2005) 547–562.
[38] P.J. Cowans, M. Szummer, A graphical model for simultaneous partitioning and labeling, in: International Workshop on Artificial Intelligence and Statistics, 2005, pp. 73–80.
[39] Y. Qi, M. Szummer, T.P. Minka, Diagram structure recognition by Bayesian Conditional Random Fields, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 191–196.
[40] T.M. Sezgin, Generic and HMM based approaches to freehand sketch recognition, in: Proceedings of the MIT Student Oxygen Workshop, 2003.
[41] G. Feng, C. Viard-Gaudin, Z. Sun, On-line hand-drawn electric circuit diagram recognition using 2D dynamic programming, Pattern Recognition 42 (12) (2009) 3215–3223.
[42] D. Arrivault, N. Richard, C. Fernandez-Maloigne, P. Bouyer, Collaboration between statistical and structural approaches for old handwritten characters recognition, in: 5th IAPR Workshop, vol. 3434, Mars 2005, pp. 291–300.
[43] P. Foggia, C. Sansone, F. Tortorella, M. Vento, Combining statistical and structural approaches for handwritten character description, Image and Vision Computing 17 (9) (1999) 701–711.
[44] T. Steinherz, E. Rivlin, N. Intrator, P. Neskovic, An integration of online and pseudo-online information for cursive word recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (5) (2005) 669–683.
[45] H. Oda, B. Zhu, J. Tokuno, M. Onuma, A. Kitadai, M. Nakagawa, A compact on-line and off-line combined recognizer, in: 10th International Workshop on Frontiers in Handwriting Recognition, vol. 1, 2006, pp. 133–138.
[46] H. Tanaka, K. Nakajima, K. Ishigaki, K. Akiyama, M. Nakagawa, Hybrid pen-input character recognition system based on integration of online–offline recognition, in: International Conference on Document Analysis and Recognition, September 1999, pp. 209–212.
[47] M. Liwicki, H. Bunke, HMM-based on-line recognition of handwritten whiteboard notes, in: International Workshop on Frontiers in Handwriting Recognition, 2006, pp. 595–599.
[48] M. Liwicki, A. Graves, H. Bunke, J. Schmidhuber, A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks, in: International Conference on Document Analysis and Recognition, 2007, pp. 367–371.
[49] M. Liwicki, H. Bunke, Combining diverse on-line and off-line systems for handwritten text line recognition, Pattern Recognition 42 (12) (2009) 3254–3263.
[50] M. Liwicki, H. Bunke, J. Pittman, S. Knerr, Combining diverse systems for handwritten text line recognition, Machine Vision and Applications (2009) 1–13.
[51] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 509–522.
[52] E.G. Learned-Miller, Data driven image models through continuous joint alignment, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2) (2006) 236–250.
[53] L. Bahl, P. Brown, P. de Souza, R. Mercer, Maximum mutual information estimation of hidden Markov model parameters for speech recognition, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 11, 1986. pp. 49–52.
[54] M. Collins, Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms, in: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, Morristown, NJ, USA, Association for Computational Linguistics, 2002, pp. 1–8.

Tevfik Metin 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 currently leads the Intelligent User Interfaces at Koç University.