Sketch-Based Interfaces: Exploiting Spatio-temporal Context for Automatic Stroke Grouping

Lutz Dickmann[1], Tobias Lensing[1],
Robert Porzel[1], Rainer Malaka[1], and Christoph Lischka[2]

[1] University of Bremen,
Bibliothekstr. 1, D-28359 Bremen, Germany
{dickmann,tlensing,porzel,malaka}@tzi.de
[2] s.l.
c.lischka@khora.de

Abstract. In this paper, we investigate how discourse context in the form of short-term memory can be exploited to automatically group consecutive strokes in digital freehand sketching. With this machine learning approach, no database of explicit object representations is used for template matching on a complete scene—instead, grouping decisions are based on limited spatio-temporal context. We employ two different classifier formalisms for this time series analysis task, namely Echo State Networks (ESNs) and Support Vector Machines (SVMs). ESNs present internal-state classifiers with inherent memory capabilities. For the conventional static SVM, short-term memory is supplied externally via fixed-length feature vector expansion. We compare the respective setup heuristics and conduct experiments with two exemplary problems. Promising results are achieved with both formalisms. Yet, our experiments indicate that using ESNs for variable-length memory tasks alleviates the risk of overfitting due to non-expressive features or improperly determined temporal embedding dimensions.

Keywords: Sketch-based Interfaces, Stroke Grouping, Contextual Computing, Reservoir Computing.

1 Introduction

In freehand sketch drawing, sketched objects often consist of multiple strokes. Manipulating these objects within a sketch-based interface—e.g., during agile collaborative meetings or presentations—requires that all their constituent strokes be selected and grouped. As of today, this step is usually performed manually with lasso or box selection tools. This can be time consuming, especially when strokes are distributed sparsely over a large canvas or placed densely within a small area. In the sketch understanding domain, automatic stroke grouping is typically addressed as part of the overall goal to recognize sketched objects. A common consensus here is to perform stroke grouping by matching template objects from a prerequisite database. Such an approach requires explicit *a priori*

R. Taylor et al. (Eds.): SG 2010, LNCS 6133, pp. 139–151, 2010.

knowledge of all the objects that may occur. What is more, it can pose a combinatorial problem.

In this work we present a possible alternative in the form of an intelligent assistant that automatically groups strokes without matching explicitly represented objects. We show that when using predictions that pertain to consecutive strokes in a limited spatio-temporal context of a sketched scene's creation history, no explicit object recognition is required to form meaningful stroke groups in an experimental setting. The view that the spatio-temporal patterns in a sketch creation process represent an unfolding discourse context in a certain domain context—using the terminology originally suggested by Gurevych et al.[4] for natural language processing (NLP) tasks—governs our approach. In the case at hand, we seek to capture discourse context via short-term memory (STM).

In the following section, we will shortly outline related work in the handwriting and sketch processing domain. After that, we state our fundamental ideas and continue with methodical and technical details. Then, experimental results are reported and consequently discussed.

2 Related Work

In the sketch understanding research domain, scene segmentation is often a necessary prerequisite of sketch interpretation. Typically, this is done by performing template matching algorithms. Sezgin & Davis, for instance, first perform primitive fitting to approximate freehand strokes as a series of distinct line and arc segments, then build Hidden Markov Models (HMMs) for all possible objects to be recognized in a scene, and align these on the temporal vector of a scene's creation history [15]. This approach implies pre-computing log-likelihood scores for all possible object classes, all possible per-class stroke counts, and all possible positions on the observation vector, then at the analysis stage to solve the problem of determining the combination with the highest score.

In the field of mathematical handwriting recognition, Wan & Watt employ both temporal and spatial proximity thresholds in order to determine when an object segmentation decision is appropriate [17]. This is seconded by Xie et al. who use an object grouping parser based on bounding box proximity thresholds in the circuit diagramming domain [21]. Both techniques rely on fixed heuristics or parser rules and capture only a specific sketching domain. Nataneli & Faloutsos [12] and Zhou et al. [22] present SVM-based workflows for stroke classification and grouping and letter segmentation in handwritten Japanese text, respectively. Except for the spatial structural approach of Nataneli & Faloutsos [12], all of the aforementioned works employ the creation history of a sketched scene for data organization and analysis. Sezgin & Davis provide empirical evidence for per-user predictability of stroke orderings for recurring known objects [15]. The *part-by-part drawing principle* as postulated by Avola et al. supports the exploitability of the creation history for grouping decisions by stating that "[d]uring the hand-drawing process of one or more than one objects whose parts are clearly identifiable, the user generally ends one part or object before drawing another" [1].

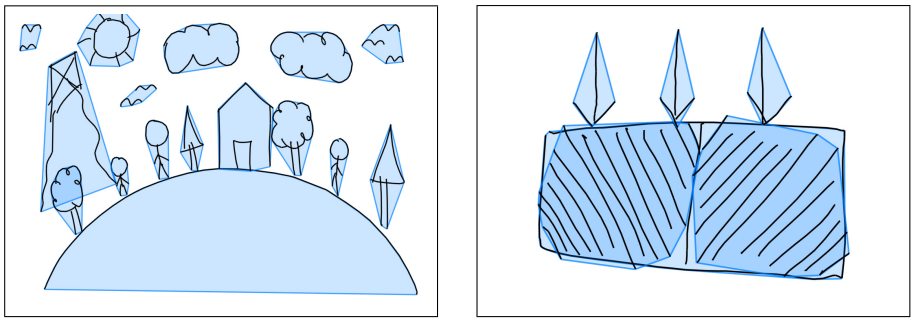
3 Concept, Methods, and Implementation

3.1 Rationale and Overview

We present an intelligent stroke grouping assistant which uses only few spatio-temporal characteristics of strokes and does not employ feature selection algorithms. With reference to the part-by-part drawing principle [15,1], we assume that stroke interspersion does not occur between consecutively drawn objects. Grouping decisions are irrevocable and pertain to a current event, taking into account a series of consecutive past events, but neither is there an anticipation of future events, nor will new insights have an effect on past decisions. In order to analyze to what extent meaningful grouping decisions can be made based on STM, we utilize and compare two different state-of-the-art machine learning techniques that appear sufficient to cope with the given problem. The first one is a support vector machine (SVM) [2] that serves as our baseline and operates on explicitly represented contextual features which pertain to a fixed number of past strokes. The second is a relatively recent approach to recurrent neural networks, namely the echo state network (ESN) technique as introduced by Jaeger [7]. In contrast to the SVM approach, this classifier has an internal state, i.e., it does not only statically operate on the currently supplied input. Both techniques work with subsymbolic numerical input—as opposed to HMMs, for instance, which require prior vector quantization and codebook creation. We follow established recommendations for commonly used basic setups wherever applicable, cf. Hsu et al. [6] and Jaeger [7]. Both concurrently used assistant variants are supplied with input from two exemplary "toy problems". We assess the suitability of both candidate techniques for the given problem and compare the results in terms of precision and recall measures.

3.2 Sketching "Toy Problems"

Naïve Landscape Scenes (NLS). The first problem—naïve landscape sketches—is designed to reflect a rather general problem in freehand sketching. There are numerous perceived objects in each scene. There is left room for arbitrary drawing strategies in terms of stroke dynamics, per-object stroke ordering, object detail, and object creation order. NLS as we consider them contain outline style objects as follows, drawn in arbitrary order: 1 hill, 1 house with 1 door, 1 sun with rays, at least 2 trees, at least 1 cloud, 2 stick figures, at least 1 flock of birds, 1 kite with a cord that one of the stick figures holds. Figure 1 shows an example from the NLS test set. The overall number of strokes in the train set is 1061, with 813 group decisions (76.63%) and 248 segmentation decisions (23.37%). The number of strokes in the test set is 131, with 99 group decisions (75.57%) and 32 segmentation decisions (24.43%). Of the total 1192 decisions in the overall dataset, 912 are group decisions (76.51%) and 280 are segmentation decisions (23.49%), therefore the prior segmentation probability is $\omega = 0.2349$.



(a) (b)

Fig. 1. Excerpt scenes from the employed toy problems. (a): Naïve landscape sketch scene. (b): Hatchings and Arrows Scene. Correct groups (ground truth) are highlighted with transparent blue convex hulls.

Hatchings and Arrows (H+A). The second problem—a hatching scene with additional arrows—is much more constrained and further specialized. It is designed to pose a specific problem to the classifier, which can only be disambiguated using discourse context in the form of short term memory. Here, strict constraints as to a spatio-temporal drawing order, object constitution, and direction are employed. In the particular case exhibited in cf. Fig. 1 (b), the local ambigious measurement is implied by the similarity of the arrow heads and the transition between the two different kinds of hatchings. Both are composed of exactly the same strokes. However, the arrows' strokes should be grouped while the last stroke of the first hatching and the first stroke of the second hatching should be segmented. In the following, we describe the creation of a H+A scene by using compass-style pointers to coarsely indicate stroke directions. The first shape drawn in the scene is a rectangle that is constructed with one single stroke, counter-clockwise, starting in the NW quadrant of the canvas: N–S; W–E; S–N; E–W. The side aspect ratio is approximately 2:1. Next, a division element is added (approximately in the center of the already present box) by drawing a straight downward stroke N–S. This virtually segments the box into two parts which subsequently are both filled with different hatching patterns. The left hatching fills the box segment SW–NE, the filling direction on the right is NW–SE. The constituting individual strokes are directed NW–SE for the left hatching pattern, SW–NE for the right one. Three arrows are finally drawn above the box with loosely equidistant spacing so they point down towards the box. Each arrow is constructed from three separate strokes, the first one in N–S direction. The second (NW–SE) and third (SW–NE) one form the arrow head. The overall number of strokes in the train set is 660, with 549 group decisions (83.18%) and 111 segmentation decisions (16.82%). The number of strokes in the test set is 77, with 65 group decisions (84.42%) and 12 segmentation decisions (15.58%). Of the total 737 decisions in the overall dataset, 621 are group decisions (83.31%) and 123 are segmentation decisions (16.69%), therefore the prior segmentation probability is $\omega = 0.1669$.

Table 1. Symbol look-up table w.r.t. the feature sets employed for the NLS task and for the H+A task. Note that for the event-coded cases, stroke-level mean ∅vel and σvel are used instead of vertex-level horizontal and vertical velocity.

Symbol	$\Delta_P(pause)$	$\Delta_S(duration)$	$d_{AABB}(distance)$	$\text{vel}_x(hor.)$	$\text{vel}_y(vert.)$
¶ (NLS)	•	•	•	○	○
† (NLS)	•	•	•	•	•
£ (NLS, H+A)	○	•	○	•	•
$ (H+A)	○	•	•	•	•

3.3 Unfolding the Scene History

Feature Extraction. Among the feature-oriented taxonomic approaches to contextual computing in the machine learning domain, we highlight the works of Katz et al., Turney, or Widdows [10,16,19,18]. With our focus on STM exploitation, we differentiate between discourse-contextual features that already encode (spatio-)temporal inter-stroke measurements and stroke-centric features that only contain local intra-stroke characteristics measured on the vertex level. We extract horizontal stroke velocity (vel_x), vertical stroke velocity (vel_y), proximity as the absolute axis aligned bounding box (AABB) distance between a stroke and its predecessor stroke (d_{AABB}), absolute constrained time delay—i.e., pause—between a stroke and its predecessor stroke (Δ_P) with a timeout of 3000 milliseconds, and the absolute time duration of a stroke (Δ_S).

For event-timed scene encoding, inter-stroke features are calculated per stroke (e.g., distance between two bounding boxes or horizontal and vertical velocity mean and standard deviation). In contrast, for discrete-time approaches such inter-stroke features are calculated per vertex (e.g., distance between the current vertex of the current stroke to the bounding box of the predecessor stroke). We will use different constellations of the extracted features within the following problem-specific evaluations with labels as given in Table 1.

Data Set Construction. The SVM is supplied with an event-timed (ET) representation of the scene history. In this representation, each stroke represents an event. That is, short-term memory is provided to the SVM externally by employing a sliding window method. As explained above, in the ET approach vertex level features are summarized to numerical values that represent these aspects only on the stroke level. Because of the transformation of vertex level dynamics to mean and standard deviation as stroke level features, potentially indicative dynamics on the vertex level are only rudimentarily accounted for.

In contrast to the ET case introduced for the SVM, we supply an ESN with a discrete time signal (DTS) sampled over the scene history so that it can capture actual vertex level dynamics. Feature vector expansion is not necessary here since the temporal dynamics of the scene history can be encoded and supplied in the temporal domain—as a signal representing a time series. We expect the discrete

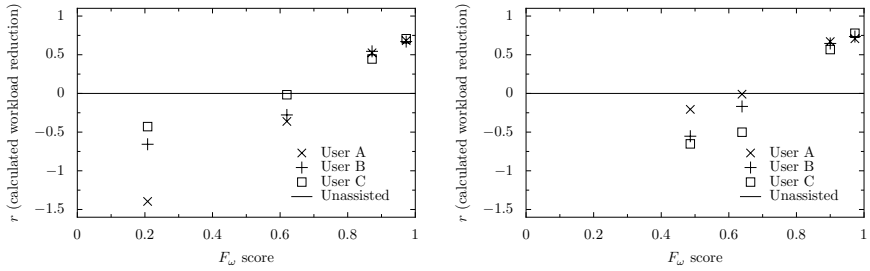


Fig. 2. User test result plots for the experimental assessment of workload reduction based on two scenes from the NLS set (left/right). r is the workload reduction ratio, the horizontal straight line labeled 'Unassisted' expresses the underlying assumption that there is no workload reduction when no assistant is employed (with F_ω unspecified)

time representation of the scene history to be most suitable for the ESN because of its nature as a classifier designed to deal with spatio-temporal dynamics.

To allow for direct commensurability of classifier setups with the SVM and ESN formalisms, we also provide the ET signal representation of the scene history to an ESN in another line of experiments. With this approach to composing an ESN input signal, each time step represents information about one stroke. The internal memory of the recurrent network can thus be seen as an alternative to the explicitly represented fixed window that is necessary in the SVM case.

3.4 Evaluation Metrics

Precision & Recall Analysis. Building on the common precision and recall analysis [14], we introduce the F_ω score as a weighted mean of two class-specific macro-average F_1 scores determined for grouping and for segmentation decisions. With ω as the prior segmentation probability, we simply calculate $F_\omega = \omega * F_{1_{\text{group}}} + (1-\omega) * F_{1_{\text{segment}}}$. In what follows, F_ω denotes the biased mean with ω pertaining to the respective prior probabilities of the used datasets.

User Utility Value vs. F_ω. In order to coarsely estimate the actual utility value of the proposed assistant for different achieved F_ω scores, we perform a small-scale experiment as a "sanity check." Here, three test subjects have to manually repair two exemplary scenes from the NLS set, each at different levels of degradation. We consider as workload reduction the difference between the cost of manually grouping a flattened scene with standard tools as known from *Adobe Illustrator*, for instance, and the cost of establishing grouping ground truth from a scene with potentially erroneous groups as yielded by the grouping assistant. Our understanding of measuring cost here is to log the number of clicks and keystrokes and the task completion time. We normalize these quantities per scene with respect to the manual grouping results and calculate a savings ratio r_q per measured cost class q with this ground truth baseline.

The resulting r_q is negative if the repair process cost exceeds that of unassisted manual grouping. To obtain a unified value for the three measured quantities

duration, *clicks*, and *keystrokes*, the overall workload reduction ratio r is heuristically calculated with doubled weights for *duration*. This weighting schemes does not involve coefficients depending on the workstation or device topology, it can thus only suffice for an exemplary assessment in a constrained setting. The user test results as shown in Fig. 2 suggest that higher F_ω scores indeed imply better assistance, and that the utility value variance between different users appears to decrease with increasing scores. However, we must also note that assistive technologies with performance levels below a certain threshold according to this metric (≈ 0.75) can *add* substantial workload (rather than diminish it as intended). This will guide our interpretation of F_ω scores.

3.5 Classifier Setups

ESN Setup Procedure. Echo state networks as proposed by Jaeger [7] denote an approach from the domain of reservoir computing (RC) where the hidden layer of a recurrent neural network (RNN) is treated as a dynamical reservoir (DR). Training is performed by linearly combining the desired output signal from a variety of nonlinear dynamical transformations of the input and/or output signals inside the DR. In contrast to conventional RNN training methods, the internal connection weights of the DR are not trained. The main advantages of this approach are computational cheapness and guaranteed convergence. In order for the approach to work, a DR used in an ESN should exhibit a special type of damped dynamics. A DR with such dynamics is said to have the echo state property, which basically ensures that the current network state is uniquely defined by its input and output. As a consequence of the echo state property, ESNs exhibit a form of short term memory—cf. Jaeger [9].

ESN setup parameters are estimated manually first following the experience and intuition of the experimenter as described by Jaeger [8]. Once a functioning standard configuration is found it is consequently reused and adapted only if needed. An assumption here is that the ESN should be applicable to deal with different toy problems whose solutions require different amounts of discourse contextual information. In contrast to the SVM, the ESN's performance should neither be significantly decreased by unnecessary additional contextual information nor should it be necessary to make exhaustive adaptations to the classifier's setup parameters for different toy problems from the same domain.

Our general ESN configuration for the scope of this work is as follows: a standard ESN is set up with multiple input signals $u_i(n)$ and one output signal $y(n)$. Internal units use the *tanh* activation function while output units use the logistic sigmoid activation function. The number of internal units is set to $N = 800$ for the discrete time signal case and to $N = 80$ for the event-timed case. The DR spectral radius is set to $\alpha = 0.999$ as to maximize short-term memory capacity [8]. Input weights are sampled from a uniform distribution within range $[-3.0, 3.0]$. The sampling rate is set to $\Delta t = 0.1sec$ (10Hz). At this sampling rate input signals obtained from the discrete time signal construction approach on the NLS train set unfold to 9692 samples, of which 200 samples are initially discarded for a washout phase during training. The resulting network's

memory capacity is approximately $MC \approx 27$ time steps in simulated time or ~ 2.7 seconds in real time. Statistical data extracted from the NLS train and test set indicates that this enables the network to roughly capture two to four strokes on average, determined based on the arithmetic mean of stroke duration and stroke delay times. This estimation of memory capacity is based on similar approaches by Hertzberg et al. and by Jaeger [5,8].

SVM Regularization and Kernel Parameter Estimation. The underlying idea of the SVM approach is to find a maximum margin hyperplane that separates data points from two classes. The set of support vectors closest to this plane comprises all points with non-zero weights after the training procedure, which presents a convex optimization problem. With a soft margins SVM as proposed by Cortes & Vapnik [2], the constant C needs to be determined as a penalty parameter for regularizition (it is also referred to as complexity parameter). In line with the recommendations of Hsu et al. [6] for basic SVM setups, we choose a radial basis function (RBF) kernel, for which another parameter (γ) has to be estimated. Our heuristic to employ a cross-validation-based grid search procedure is directly adopted from Hsu et al. We first perform one coarse search with $C = \{2^{-5}, 2^{-5+\Delta_C}, 2^{-5+2\Delta_C}, ..., 2^{15}\}$ and $\gamma = \{2^{-15}, 2^{-15+\Delta_\gamma}, 2^{-15+2\Delta_\gamma}, ..., 2^{3}\}$, where $\Delta_{C\,coarse} = \Delta_{\gamma_{coarse}} = 1.0$. This result is then refined with a more fine-grained search (step size $\Delta_{C\,fine} = \Delta_{\gamma_{fine}} = 0.25$) within the respective neighborhood (within a ± 2.0 range for the exponents yielded by the coarse search).

We iteratively expand the feature vector with additional memory steps to which we will also refer as the embedding dimension with $m = 0$ for no additional past events, cf. the terminology of Mukherjee et al. [11]. This procedure is followed in two variants concurrently: to keep all other parameters except the embedding dimension m constant, we run each of the experiments once with an "out-of-the-box" SVM setup that only uses default values $C = 1.0$ for the complexity parameter and $\gamma = 0.01$ for the RBF kernel parameter. Additionally, we conduct the rather time-consuming parameter estimation with the previously outlined grid search heuristic for each experiment—that is, for each expansion step of the feature vector (embedding dimension). Consequently, we refer to classifiers with estimated parameters (EP) and such with unestimated parameters (UEP) to distinguish these setups. All classifier setups use feature scaling with factors determined prior to training by normalizing the training set features to the range $[0.0, 1.0]$. The same scaling factors are then applied during testing. We employ the sequential minimal optimization SVM training implementation included in Weka [13,20].

4 Evaluation

As is common practice in machine learning research, we use $\sim 90\%$ of each set for training and the remainder for testing. Each set contains 20 sketched scenes annotated with ground truth, of which 18 are initially randomly chosen as train set, and the remaining two are used as a test set. These data sets are

Table 2. Comparison of classifier results. Best test results per feature constellation (Fts.) and task are highlighted. m denotes (stroke) events for the SVM cases, for the ESN setups m expresses MC in simulation steps.

Experiment series:			**Naïve Landscape Scenes**				**Hatchings and Arrows**			
			Default setup		Est. parameters		Default setup		Est. parameters	
Classifier	Fts.	m	F_ω Train	F_ω Test	F_ω Train	F_ω Test	F_ω Train	F_ω Test	F_ω Train	F_ω Test
ESN DTS	¶	27	0.944	0.933	–	–	–	–	–	–
ESN ET	¶	14.3	0.973	0.960	–	–	–	–	–	–
SVM ET	¶	0	0.943	**0.973**	0.962	0.960	–	–	–	–
SVM ET	¶	1	0.957	**0.973**	0.985	0.960	–	–	–	–
SVM ET	¶	2	0.957	**0.973**	0.985	0.947	–	–	–	–
SVM ET	¶	3	0.957	**0.973**	0.988	0.947	–	–	–	–
ESN DTS	†	27	0.962	0.947	–	–	–	–	–	–
ESN ET	†	14.3	0.956	0.950	–	–	–	–	–	–
SVM ET	†	0	0.947	0.959	0.990	0.960	–	–	–	–
SVM ET	†	1	0.959	0.973	0.993	0.974	–	–	–	–
SVM ET	†	2	0.961	0.973	1.000	0.974	–	–	–	–
SVM ET	†	3	0.965	0.960	1.000	**0.987**	–	–	–	–
ESN DTS	£	27	0.875	0.774	–	–	0.980	0.856	–	–
ESN ET	£	14.3	0.773	0.865	–	–	0.988	**1.000**	–	–
SVM ET	£	0	0.422	0.314	0.771	0.673	0.679	0.657	0.826	0.743
SVM ET	£	1	0.617	0.611	0.926	0.842	0.819	0.828	0.948	0.790
SVM ET	£	2	0.608	0.620	0.949	0.826	0.907	0.922	1.000	0.903
SVM ET	£	3	0.619	0.629	0.947	**0.886**	0.884	0.922	1.000	0.965
SVM ET	£	4	0.629	0.629	0.916	0.838	0.883	0.922	1.000	0.965
SVM ET	£	5	0.636	0.629	0.929	0.800	0.889	0.922	1.000	0.933
ESN (DTS)	$	27	–	–	–	–	0.992	0.962	–	–
ESN (ET)	$	14.3	–	–	–	–	0.988	**1.000**	–	–
EP SVM (ET)	$	0	–	–	–	–	0.904	0.922	0.909	0.922
EP SVM (ET)	$	1	–	–	–	–	0.904	0.922	0.956	0.767
EP SVM (ET)	$	2	–	–	–	–	0.909	0.922	1.000	0.933
EP SVM (ET)	$	3	–	–	–	–	0.909	0.922	1.000	0.903
EP SVM (ET)	$	4	–	–	–	–	0.912	0.922	1.000	0.850
EP SVM (ET)	$	5	–	–	–	–	0.909	0.922	1.000	0.903

kept constant during all experiments to conserve the ability to conduct visual inspection of the results based on well-familiar scenes. We perform this basic sequence of experiments with each of the feature sets specified above—¶, †, and £ for the NLS example, £ and $ for the H+A problem—in their respective encoding (event-timed or discrete-time sampled). In the SVM case, we additionally exercise iterative expansion of the embedding dimension, with $m \in \{0, 1, 2, 3\}$ (NLS with ¶, †) and $m \in \{0, 1, 2, 3, 4, 5\}$ (H+A, NLS with £).

The overall results obtained from all experiments conducted in this series are listed in Table 2. The ET ESN and the DTS ESN perform similarly with all feature sets considered except feature set £, for which the event-timed ESN achieves considerably better scores. In all cases, the event-timed ESN reaches F_ω scores equal or better than those of the respective discrete time ESN setups. We therefore put emphasis on the comparison of the event-timed ESN with the event-coded SVM. For the memory-range tested with both EP and UEP SVMs, it can be observed that the ET ESN performance is always slightly below the optimum value reached by the best-performing SVM in the NLS case. This is the EP version (estimation per each classifier training) in two out of three cases. Recall that the parameters for the ESNs employed are manually adjusted only once, in contrast. For the ¶ feature set the ESN is initially *en par* with the

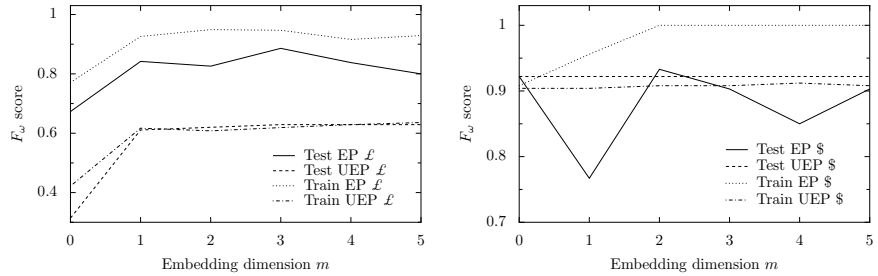


Fig. 3. SVM result plots for the NLS task with feature constellation £ (a) and for the H+A task using feature set $ (b)

EP SVM—however, while increasing memory can increase SVM classification performance, this may also lead to a decline as compared to smaller values of m due to overfitting. In the treatment of the H+A toy problem, only the ESN classifiers achieve an F_ω score of 1.0 on the test set (with $F_\omega = 0.988$ on the train set), while the EP SVM variants overfit on the train set with $F_\omega = 1.0$ and cannot achieve test performance higher than $F_\omega = 0.965$. The extended feature set $ has no impact on the ET ESN and increases the performance of the DTS ESN slightly but does not aid to let the EP or UEP SVM variants achieve higher test scores than with feature set £ only. The result plots in Fig. 3 shed further light on the SVM sensitivity to different feature constellations and embedding dimensions.

In order to complement the results from our main evaluation based on fixed test sets, we conduct examplary resampled paired t-tests—the applied procedure is detailed by Dietterich [3]. Such a "statistical sanity check" appears reasonable for situations where an ESN setup and an SVM setup both achieve identical F_ω scores for the same task. Note that ESNs are initialized randomly, therefore the validity of the reported scores requires confirmation. We pick the NLS case with feature set ¶, for which both the ET ESN and the EP SVM (with $m = 0$) score $F_\omega = 0.96$. 30 trials are conducted with uniformly random partitions of the overall available NLS dataset (with a train set proportion of $\sim 66\%$). In each trial, the instantiations of both classifier variants are trained and tested on the same data—the null hypothesis is that both algorithms will exhibit the same performance and error rate. The accuracy-based resampled paired t-test yields $t = -0.18706$, with the primary metric F_ω as employed in this work $t = 0.18868$. In both cases, $|t| < t_{29;0.975} = 2.04523$ and also $|t| < t_{29;0.9} = 1.3114$. Thus we cannot reject the null hypothesis: there is no significant difference in performance between the two considered classifier setups (with identical reported F_ω scores). In a second approach, we conduct two further t-tests with 10 trials each to check whether in the H+A case the ET ESN is also significantly better in comparison to an EP SVM with $m = 0$ and $m = 1$ for feature set $ if we diverge from the fixed test set. Also, we use a more fine-grained parameter estimation grid search

for these re-runs, which renders slightly altered SVM F_ω scores on the original fixed test set: $F_{\omega_{Train}} = 0.909$ and $F_{\omega_{Test}} = 0.922$ for $m = 0$; $F_{\omega_{Train}} = 0.959$ and $F_{\omega_{Test}} = 0.789$ for $m = 1$ (cf. Table 2). In all cases ($m \in \{0, 1\}$, accuracy-based and F_ω-based) we find that $|t| \approx 10 > t_{9;0.99} = 2.8214$. Here, the null hypothesis can be rejected: it appears to hold that the ESN indeed performs significantly better in the regarded exemplary cases, no matter whether we spend more cycles on SVM parameter estimation and/or whether we vary the train and test sets for the H+A task.

5 Discussion

We find that a discourse-contextual paradigm for exploiting STM can help to successfully form meaningful stroke groups. Feature choice along with domain/data knowledge is a primary prerequisite to proper configuration with both used formalisms. This can be seen for the NLS case, where both classifiers do not perform notably better with more features. The temporal delay and spatial proximity features appear to be most expressive here. An important issue w.r.t. the exploitation of discourse context by an SVM is that different problems may require different memory lengths to reach the best solutions. Mukherjee et al. note w.r.t. potential SVM sensitivity to embedding dimension expansion that overfitting problems may occur with non-linear kernels [11]. With regard to tackling different sketching problems, this may pose a problem when using the SVM. The ESN apparently deals with such issues more robustly. Inferring from the comparison results, we can state that while the SVM can reach slightly better classification results when we experimentally determine and optimize it for a particular memory length, the ESN appears to be relatively robust to overfitting issues and provides a more general solution with its inherent STM at similar performance. We were able to determine one adequate network setup from the first ESN experiment conducted and then transfer that setup to other signal codings, feature sets, and toy problems with little or no modification of setup parameters. However, it is to be noted that the determination of a suitable ESN setup is rather complex and laborious. As stated by Jaeger [9], it requires profound experience and intuition to determine adequate network setups. Then, however, a solution that works for one problem is likely to work for other related problems as well. This makes the ESN a more suitable technique when considering a transfer to other sketch domains and applications.

6 Conclusion

Considering the rapidly developing field of surface computing and sketch-based interfaces, we have focused on the problem of automatically grouping consecutively drawn strokes. Posing that a database of explicit object representations for template matching on a complete scene is not desirable, we have investigated how grouping decisions can be made relying only on limited spatio-temporal context in the form of short-term memory. The formalisms of internal-state ESNs

(inherent memory capabilities) and conventional static RBF-SVMs (memory representation in Euclidian space via feature vectors) have been proposed for this time series analysis task and set up according to the respective requirements. An evaluation has been performed based on two different exemplary problems, "Naïve Landscape Scenes" and "Hatchings and Arrows." We have shown that the contextual computing approach as exercised here yields promising results in terms of error rates and workload reduction for both formalisms in the considered experimental settings. Moreover, our overall results suggest that ESNs are better suited for variable-length memory tasks as their use alleviates the risk of overfitting that may otherwise occur due to non-expressive features or improperly determined temporal embedding dimensions.

Acknowledgements. The research presented in this paper was supported by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 "Autonomous Cooperating Logistic Processes."

References

1. Avola, D., Caschera, M.C., Ferri, F., Grifoni, P.: Ambiguities in sketch-based interfaces. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS 2007), January 2007, p. 290b (2007)
2. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning, 273–297 (1995)
3. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10, 1895–1923 (1998)
4. Gurevych, I., Porzel, R., Malaka, R.: Context: Integrating Domain- and Situation-specific Knowledge. In: Wahlster, W. (ed.) SmartKom - Foundations of Multimodal Dialogue Systems, pp. 269–284. Springer, Berlin (2006)
5. Hertzberg, J., Jaeger, H., Schönherr, F.: Learning to ground fact symbols in behavior-based robots. In: ECAI, pp. 708–712 (2002)
6. Hsu, C.-W., Chang, C.-C., Lin, C.-J.: A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (2003)
7. Jaeger, H.: The echo state approach to analysing and training recurrent neural networks. Technical Report 148, GMD - German National Research Institute for Computer Science (December 2001)
8. Jaeger, H.: Short term memory in echo state networks. Technical Report 152, GMD - German National Research Institute for Computer Science, May 28 (2002)
9. Jaeger, H.: A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach (2nd revision). Technical Report 159, Fraunhofer Institute for Autonomous Intelligent Systems, AIS (2005)
10. Katz, A.J., Gately, M.T., Collins, D.R.: Robust classifiers without robust features. Neural Comput. 2(4), 472–479 (1990)
11. Mukherjee, S., Osuna, E., Girosi, F.: Nonlinear prediction of chaotic time series using support vector machines. In: IEEE Workshop on Neural Networks for Signal Processing VII, pp. 511–519. IEEE Press, Los Alamitos (1997)
12. Nataneli, G., Faloutsos, P.: Robust classification of strokes with svm and grouping. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Paragios, N., Tanveer, S.-M., Ju, T., Liu, Z., Coquillart, S., Cruz-Neira, C., Müller, T., Malzbender, T. (eds.) ISVC 2007, Part I. LNCS, vol. 4841, pp. 76–87. Springer, Heidelberg (2007)

13. Platt, J.: Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research (1998)
14. Rijsbergen, C.J.v.: Information retrieval, 2nd edn. Butterworths, London (1979)
15. Sezign, T.M., Davis, R.: HMM-based efficient sketch recognition. In: Proc. 10th Intl. Conf. Intelligent User Interfaces (IUI 2005), pp. 281–283 (2005)
16. Turney, P.D.: Robust classification with context-sensitive features. In: IEA/AIE 1993: Proc. 6th Intl. Conf. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, pp. 268–276 (1993)
17. Wan, B., Watt, S.: An interactive mathematical handwriting recognizer. In: MathML Conference (2002)
18. Widdows, D.: A mathematical model of context. In: Blackburn, P., Ghidini, C., Turner, R.M., Giunchiglia, F. (eds.) CONTEXT 2003. LNCS (LNAI), vol. 2680, pp. 369–382. Springer, Heidelberg (2003)
19. Widdows, D., Dorow, B.: A graph model for unsupervised lexical acquisition. In: Proceedings of the 19th international conference on Computational linguistics, pp. 1–7. Association for Computational Linguistics, Morristown (2002)
20. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
21. Xie, Q., Sun, Z.-X., Feng, G.-H.: Online diagramming recognition based on automatic stroke parsing and bayesian classifier. In: Proc. Machine Learning and Cybernetics 2006, August 2006, pp. 3190–3195 (2006)
22. Zhou, X.-D., Wang, D.-H., Liu, C.-L.: A robust approach to text line grouping in online handwritten japanese documents. Pattern Recogn. 42(9), 2077–2088 (2009)