Robust Classification of Strokes with SVM and Grouping

Gabriele Nataneli and Petros Faloutsos

University of California Los Angeles
nataneli@cs.ucla.edu,
pfal@cs.ucla.edu

Abstract. The ability to recognize the strokes drawn by the user, is central to most sketch-based interfaces. However, very few solutions that rely on recognition are robust enough to make sketching a definitive alternative to traditional WIMP user interfaces. In this paper, we propose an approach based on classification that given an unconstrained sketch, can robustly assign a label to each stroke that comprises the sketch. A key contribution of our approach is a technique for grouping strokes that eliminates outliers and enhances the robustness of the classification. We also propose a set of features that capture important attributes of the shape and mutual relationship of strokes. These features are statistically well-behaved and enable robust classification with Support Vector Machines (SVM). We conclude by presenting a concrete implementation of these techniques in an interface for driving facial expressions.

1 Introduction

In recent years there has been a proliferation of sketch-based solutions designed for a variety of different applications. Some approaches rely on a procedural mapping between two-dimensional strokes and possibly higher-dimensional geometries [1] or models [2]. The problem is generally well-defined and well-constrained in these cases. As a result, these solutions tend to be very usable and effective, although they may be limited to a specific domain. Other approaches rely on the detection of a well-defined dictionary of symbols. In this context, there is a large literature on interfaces based on handwriting recognition and the recognition of other symbols that belong to the vernacular of a particular application. Alternatively, some solutions define a new set of symbols that is easy to learn and can be employed to simplify certain interactive tasks. Motion Doodles [3] is an example of the latter, which enables users to define character animation through a fluid set of cursive motions.

The most challenging task is that of recognition-based sketching. A general approach for recognition-based solutions , such as [4], is to analyze the sketch by identifying and recognizing components that capture the semantics of the sketch. We call these components simply as strokes.

In this paper, we explore the problem of recognizing strokes robustly with particular emphasis on the problem of classification. We use a support vector

G. Bebis et al. (Eds.): ISVC 2007, Part I, LNCS 4841, pp. 76–87, 2007.

machine that is trained on quantifiable aspects of the sketch to perform the classification. We then use stroke grouping to augment the capability of the classifier to work with complex sketches in the presence of outliers.

The contributions of this work can be summarized as follows:

- We propose a machine learning approach to classify the elements of unconstrained input sketches based on support vector machines
- We propose a set of attributes that capture both individual shape features and spatial relationships of groups of strokes.
- We propose a grouping technique that makes our approach robust to outliers.

2 Related Work

There is a large body of work concerned with the related problem of handwriting recognition. [5] presents an approach for recognizing and grouping handwritten text. This work groups strokes by minimizing a cost function efficiently using a dynamic programming algorithm. This paper is similar in spirit to our approach, but is tailored around the problem of symbol recognition, while our approach strives to be more general. [6] is related to [5] and uses stroke groupings to aid in the recognition of numeral strings. Their work selects the best grouping based on a *grouping-hypothesis* and their focus is restricted to the recognition of numeral strings. This paper also puts particular emphasis on segmentation which is absent both in [5] and our paper. The work by Saunds on perceptual organizations [7,8] is one of the main sources of inspiration for our approach based on groupings. These papers show how to organize strokes into groups that are perceptually sound. Sharon in the paper [4] presents an approach for sketch-recognition based on a computer vision technique referred to as *constellation model*, which is related to our work. The classic work by Arnheim [9] presents an enlightening study of visual perception in the context of art and psychology. This work is the basis and main inspiration for the shape attributes that we use to train the SVM. We briefly mention a few additional papers that present interesting ideas that are related to our work. The paper [2] describes a similar application for posing 3D faces via sketches, but it does not rely on sketch recognition. The work by Yang [10] presents an approach for the analysis of sketches based on their connectivity. Some of the ideas introduced by this paper, such as the use of templates and the parametrization, resemble some aspects of our application described in section 9.

3 Overview

As the user draws a sketch interactively, we record every continuous motion of the mouse or stylus as a separate stroke. In order to recognize the sketch, a classifier assigns a label to each stroke that comprises the sketch based on the content of a user-generated training set. The training set is based on a compact and statistically well-behaved set of shape descriptors, called shape attributes,

that are discussed in section 7. We augment the capabilities and robustness of the classifier by grouping strokes [7,8]. What is novel in our approach is that our mechanism for selecting the groupings is largely data driven. As a result we do not simply rely on spacial groupings [5] or groupings based on the geometry of strokes [5,8], but we also enable groupings based on the semantics of the sketch that can be inferred from the training set. In the following sections, we describe our classification mechanism and then show how it can be associated with groupings to robustly handle complex sketches containing outliers. Our discussion is followed by a detailed description of shape attributes, which lie at the heart of the classification. A detailed understanding of shape attributes is not required to understand the following sections, but the interested reader may want to skip ahead to section 7 before reading sections 4 and 5.

4 Classification

Our approach for classification relies on a support vector machine (SVM). The robustness of SVM depends primarily on the amount and choice of training data as well as the choice of a kernel. If the training data makes the structure of the data obvious, we can expect machine learning to perform robustly even for a fairly small training set. Otherwise, even a large training set may perform poorly. For the specific problem of classification the quality of a training set can be evaluated quantitatively by studying the variance of the data. Items that belong to the same class should exhibit relatively low variance, while items that belong to different classes should be well-separated and therefore exhibit large variance. A classifier trained on a training set that possesses these properties is expected to perform robustly. We represent shape through a set of features that we call *shape attributes*. In this paper, we use the term shape attribute interchangeably to express both the features of strokes and the functions used to quantify these features.

We use the following formalism to clarify our approach. When the user draws a stroke, we store it in memory as a vector of points s_i. When the drawing is complete the entire sketch is merely a set of strokes $S = \{s_1, s_2, \ldots, s_n\}$. A shape attribute is a function $A(s_i)$ that analyzes the stroke s_i and produces a real number $a \in \Re$. During training the user also needs to manually associate a label l_i to each stroke s_i in the sketch. We construct a training vector t_i by combing the label l_i with a selection of shape attributes. Hence, a training vector t_i has the form

$$t_i = [l_i, A_1(s_i), A_2(s_i), \ldots, A_m(s_i)]$$

A training example T for the SVM corresponding to the sketch S is therefore represented by the set of training vectors $T = \{t_1, t_2, \ldots, t_n\}$. During performance, the user draws a new sketch $S' = \{s'_1, s'_2, \ldots, s'_k\}$. For each stroke s'_i we construct a query vector

$$q_i = [A_1(s'_i), A_2(s'_i), \ldots, A_m(s'_i)]$$

The query vector q_i is then used to query the classifier C and obtain a class label l'_i for each corresponding input stroke

$$l'_i = C(q_i)$$

We initially developed 12 shape attributes that capture the notions described in [9]. However, for the classifier we restricted the number of shape attributes to a stable subset that results in robust classifications. To find this stable subset, we generated many alternative training sets using different selections of shape attributes and we studied their variance. We chose a selection of shape attributes that exhibits small variance for strokes that belong to the same class and large variance for strokes that belong to distinct classes. We repeated this process for various categories of sketches corresponding to faces, houses, and cars to make sure that our selection is independent of the content of the sketch. Table 1 shows the stable selection of shape attributes that we chose for the classification. Surprisingly all these shape attributes correspond to high level features of strokes. This finding agrees with our intuition. In fact, a few generic strokes are generally sufficient for a human to recognize the content of a sketch and a robust classifier is expected to do the same.

Bounding Box Width
Bounding Box Height
Bounding Box Aspect Ratio
Centroid X
Centroid Y
Horizontal Ordering
Vertical Ordering
Overall Stroke Count
Depth

Fig. 1. The stable selection of shapes attributes used for the classification

We verified the robustness of the classification based on this selection of shape attributes by training the SVM and running cross-correlation tests. Subsequently, we tested the classification manually by drawing many sample sketches and observing classification results. The average accuracy of our classifier with different training sets is about 93%. Our sample sketches did not contain either outliers or duplicate strokes - that is multiple strokes of the same class. We refer to sketches that satisfy these two properties as *clean* sketches.

Our experiments show that in general a fairly small training set that contains approximately ten training vectors per class is sufficient to obtain a robust classification of clean sketches using our method. The actual size of the training set that is needed to achieve a robust classification is correlated to the variance of training vectors. If the training vectors of two distinct classes exhibit a small variance, we need more examples for those two classes.

A SVM is sensitive to the scaling of data, so we always have to normalize shape attributes before using them in the training. However, we observed a 20% improvement in the classification by exploiting a non-uniform scaling of the data. Specifically, the ordering attributes, the overall stroke count, and the depth attribute are left unchanged as integer values. This improvement follows from the fact that the different scaling makes the SVM weigh these shape attributes differently. This choice also impacts how the choice of kernel for the SVM affects the performance of the classifier.

Support Vector Machines rely on a choice of kernel for their operation. The only two kernels that provide acceptable performance for our task are linear and polynomial. A linear kernel works best when we want to classify sketches containing a single stroke or sketches in which the relationship of strokes is not relevant. In these cases, the SVM prioritizes the shape attributes that are properly scaled, which for us are the ones that relate to individual features of a stroke. On the other hand, A polynomial kernel is less influenced by the non-linearity of the training set, so it is the best choice in all common sketch recognition tasks. This property of our classifier offers an advantage over solutions that rely solely on the relation of strokes [4].

5 Grouping

In the previous section we highlighted the fact that our core classification mechanism is only expected to work robustly for clean sketches. Again, a clean sketch is one that satisfies the following two properties:

1. It does not contain outliers.
2. No two strokes belong to the same class (duplicate strokes).

We know that a clean sketch is classified successfully only if each query vector is assigned a distinct label. Therefore, if we find a duplicate label in the classification results, we are certain that a misclassification has occurred. This is a central property for searching the space of possible groupings as explained later. Furthermore, we also expect more reliable classifications for clean sketches. Clean sketches do not contain outliers, which are one of the major sources of errors in classification. An outlier is a stroke whose correct label is not present in the training set. Hence, outliers always lead to misclassifications, since the classifier is always expected to output a label for any given query vector. The reason why the second property of clean sketches is important is more subtle. Duplicate strokes are strokes whose corresponding query vectors exhibit low variance and thus are assigned the same label by the SVM during performance. Therefore, duplicate strokes are intrinsically ambiguous and they are generally not well-behaved statistically. Moreover, if we allowed duplicate strokes, we would not have any automated way for distinguishing a duplicate stroke from a misclassified one. In summary, clean sketches give us a better way to attest the robustness of the classifier.

Sketches of practical interest, however, do not satisfy the properties of clean sketches in most cases. Therefore we use groupings to accommodate more complex sketches within the existing framework for classification. Given a Sketch $S = \{s_1, s_2, \ldots, s_n\}$ a grouping $G = \{g_1, g_2, \ldots, g_n\}$ is a set of groups g_i where

1. $g_i \subset S$
2. $g_i \cap g_j = \emptyset$ with $i \neq j$

Our objective is to find a proper grouping G of sketch S that is guaranteed to be clean. Any sketch that is not clean can be reduced to a clean sketch by some grouping as follows:

- Group duplicate strokes with the same label into the same group
- Group every outlier with some other stroke

Groupings, however, present an additional challenge as well. After we group several strokes, we need to define the meaning of applying a shape attribute to the group - that is we have to define the meaning of $A(g_i)$. We do this by replacing the group g_i with a new stroke that characterizes the basic features of the group. This is explained more in detail later in this section.

The space of possible groupings is a power set over the number of strokes given as input and it is clearly intractably large to be searched exhaustively. In order to make the problem tractable, we need to prune the space. We consider three different kinds of groupings to accomplish this task: structural grouping, overlap grouping, and semantic grouping. Each kind of grouping corresponds to a specific level of abstraction of the properties of the sketch.

Structural Grouping. At the lower level we have structural groupings. A structural grouping is based on purely geometric notions of *proximity* and *continuity*. *Proximity* refers to strokes that are aligned (vertically or horizontally) and are near each other. In terms of shape attributes, a proximity grouping selects strokes that have the same ordering (refer to Section 7) and whose distance between the centroids is below a threshold. Figure 2a is an example of proximity group. We characterize the proximity group with a new stroke that connects the centroids of each stroke that participates in the group.

Continuity refers to strokes whose endpoints are near each other. Unlike [7] we are not considering the orientation of the strokes in order to establish continuity, since the purpose of groupings in our case is slightly different. Figure 2b is an example of continuity group.

A structural grouping corresponds to a perceptual organization as described in [7] and [8]. A structural grouping is also the simplest kind of grouping and does not depend on the training set. The strokes that are candidates for a structural grouping are generally very common and very numerous in a typical sketch, so we do not want to perform expensive calls to the SVM in order to establish these groups. At the same time, even though structural groupings are not tailored to the training set -that is the particular domain of the recognition task - this fact does not affect the ability of higher level groupings, such as overlap groupings and

semantic groupings, to analyze the semantics of the sketch successfully. Lastly, structural groupings are very effective at pruning the space of possible groupings and removing common outliers.

Overlap Grouping. At an intermediate level of abstraction there are groupings based on overlap. Stroke s_i overlaps stroke s_j if $A_{depth}(s_i) > A_{depth}(s_j)$ and $\|centroid(s_i) - centroid(s_j)\| < \alpha$, where A_{depth} is the depth shape attribute, *centroid* is a function that returns the position of the centroid for the given stroke, and α is a threshold. In this example, strokes s_i and s_j form an *overlap group candidate* (OGC) $g_{oc} = \{s_i, s_j\}$. We determine all OGCs efficiently by building an adjacency matrix of all strokes, while computing the shape attributes. The depth of an OGC g_{oc} is

$$depth(g_{oc}) = max(|A_{depth}(s_i) - A_{depth}(s_j)|)$$

where $s_i, s_j \in g_{oc}$. Figure 2c is an example of overlap group. The body of the house, the window, and the window grille form an OGC. The body has depth 0, the window has depth 1, and the window grille has depth 2. Hence, the depth of the OGC is 2.

We first reduce the number of strokes in each OGC by grouping the strokes in each OGC so that their depth falls below a given threshold β. The value of β depends on the amount of detail that we want to preserve. In Figure 2c if we set β to 2, we will discard the window grille. For each OGC we run an algorithm that can be summarized as follows:

1. Consider the power set of all OGCs. This will generate many subsets of OGCs of the form $\{g_{oc}^1, \ldots, g_{oc}^k\}$.
2. For each subset of OGCs group the strokes in each OGC. This will partition the original set of strokes S into a grouping of the form $G = \{g_1, \ldots, g_k, s_j, \ldots, s_n\}$ where each g is a group of strokes that is replaced with a stroke that represents the bounding box of the group and s_i (groups of cardinality 1) are strokes that are not grouped.
3. Classify the grouping G with SVM and obtain a set of labels $L = \{l_1, \ldots, l_n\}$. If L contains any duplicate, then the grouping G does not represent a clean sketch, so we reject this grouping. Otherwise, if L does not contain duplicates we accept the grouping G.
4. Rank all accepted groupings based on the heuristic described in section 6.
5. Repeat the steps for each subset generated in step 1.
6. Choose the grouping with the best rank.

Although the number of OGCs is generally fairly small, the number of subsets generated in step 1 may still be prohibitively large. However, performance is still acceptable since most subsets are either rejected or they get low rankings. In our experiments we only considered subsets of cardinality 2 or 3 without compromising the capability of our algorithm.

Semantic Grouping. In some cases, there are no strokes that overlap or there is not any subset of OGCs that results in a clean sketch. Semantic groupings consider groups that cannot be captured by either structural groupings or overlap groupings. Therefore, semantic groupings deal with groups of strokes that are generally not correlated by geometrical features, but are correlated by the semantics of the drawing that can be inferred from the training set.

We proceed similarly to overlap groups, but we replace step 1 with the power set over the strokes that are left ungrouped after performing structural grouping and overlap grouping. We refer to these as candidate strokes. Unfortunately there is no obvious way to prune the power set of candidate strokes, so we adopt a heuristic that was fairly effective in our experiments. We first run the classifier on all candidate strokes; the classification results are expected to contain duplicates at this stage. We then consider each group of duplicate as a candidate and run the same algorithm over the power set of these groups. The rationale is that duplicate strokes are likely to be related semantically.

Our approach for grouping gives us robustness against outliers. Outliers would always result in incorrect classifications if they are individually fed to the classifier. More precisely, outliers are very likely to generate classifications that contain duplicates by the pigeon hole principle. As a result, our approach for grouping forces outliers to be grouped with other strokes so that they will not reach the classifier in isolation. Moreover, our approach can cope with outliers gracefully, as we can improve the robustness by expanding the training set.

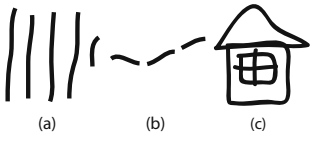

Fig. 2. Examples of different kinds of groups. (a) Proximity group. (b) Continuity Group. (c) Overlap group.

6 Ranking

Our approach for overlap and semantic groupings is effectively an optimization that searches for the grouping with a best rank. In turn, the ranking scheme is based on the results of the classifier. In general, we only want to rank classifications that do not contain duplicates - that is classifications that are known to correspond to clean sketches. We do this, because, we can only expect the classifier to perform robustly for clean sketches. We want to rank higher groupings that result in the largest number of distinct classification labels. This is because, we would like to exploit the variety of the training set as much as possible and we also want to avoid to group strokes that belong to distinct classes. Hence, we compute the rank by simply adding the number of distinct labels produced by the classification. In some cases, for semantic groupings we cannot simply reject groupings that contain duplicates; therefore, we actively penalize duplicates

by subtracting the number of groups containing duplicates from the number of distinct labels.

7 Shape Attributes

In this section, we describe shape attributes in detail. Shape attributes provide an effective representation of strokes that acts as a useful abstraction for analyzing hand-drawn sketches. We originally developed 12 shape attributes for the application described in section 9. Here we only describe the subset of shape attributes that we use in the classification. For a complete list of shape attributes refer to [11].

Bounding Box. The bounding box captures the proportions of the drawing and is used to inform the classifier about the relative size of objects. We always normalize the actual size of the bounding box for each stroke with respect to the bounding box enclosing the whole sketch. This way proportions remain consistent even if sketches are drawn at different scales.

Centroid. The centroid is computed in the usual manner by averaging the x and y components of the points comprising the stroke. The centroid effectively captures the region of a stroke that carries the largest perceptual weight [9] and is used to inform the classifier about the relative position of strokes in the sketch. The centroid is also used to compute several other shape attributes described in [11].

Horizontal and Vertical Ordering. Horizontal and vertical ordering are very effective attributes that allow us to quantify the relation in the position of strokes without having to know the class of neighboring attributes in advance. Figure 3 shows a sample face with values of horizontal ordering for every stroke.

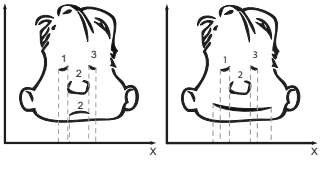

Fig. 3. Horizontal ordering. The image on the left shows a typical example in which the value of horizontal ordering is well defined. The image on the right shows an ambigous case in which the horizontal ordering for the mouth is not well defined.

Given an input sketch $S = \{s_1, s_2, \ldots, s_n\}$, let $ho : S \mapsto N$ be a function that maps a stroke to its value of horizontal ordering. The horizontal ordering is determined by applying the following rules in succession.

- Given strokes X and Y, $ho(X) = ho(Y)$ if the projection of their bounding boxes on the x axis overlaps

– If X and Y are not overlapping and the centroid of $X > Y$, then $ho(X) > ho(Y)$

It is not always possible to assign values of horizontal ordering and simultaneously satisfy both rules. Figure 3 shows one example of an ambiguous situation. If a stroke generates an ambiguity it means that the value of horizontal ordering is not relevant for that particular stroke and we can assign to it some arbitrary value. Referring again to Figure 3 we observe in fact that the value of horizontal ordering is essential to distinguish the left eye from the right eye, but it doesn't have any meaning for the mouth. On the other hand, the vertical ordering is very useful to distinguish the eyes from the mouth. We say that strokes s_i and s_j x-overlap iff the projection on the x axis of their bounding box overlaps and $width(s_1) < width(s_2)$. The algorithm for computing the horizontal ordering follows:

1. Scan strokes from the ones with smaller width to the ones with larger width and group them based on their x-overlap. A group is a vector of the form $g = [s_1, s_2, \ldots, s_n]$ where s_i is a stroke in the sketch
2. Compute the average x position for each group and arrange the values in a vector.
$$a = [avg_x(g_1), avg_x(g_2), \ldots, avg_x(g_n)]$$
3. Sort the vector a.
4. Assign a value of horizontal ordering to each g_i based on its sorted order in the vector.
5. Assign the value of horizontal ordering for group g_i to every stroke $s_i \in g_i$.

Depth. The depth attribute informs the classifier about the overlap of strokes. We say that stroke s_i overlaps s_j if the bounding box of s_i is fully contained in the bounding box of s_j. Note that this definition of overlap is different from the one given in section 5. Given a set of strokes $S = \{s_1, s_2, \ldots, s_n\}$, the depth satisfies the following rules:

– If s_i does not overlap with any stroke in the set $S \setminus \{s_i\}$ then its depth is zero
– If s_i overlaps with $s_j \in S \setminus \{s_i\}$ then $depth(s_i) < depth(s_j)$

The depth is computed as follows:

1. group all strokes that overlap in the same set.
2. Sort the items in the set based on the area of their bounding boxes.
3. Assign values of depth to each stroke in sorted order. From the way we defined overlap we are guaranteed that small strokes have a value of depth that is higher than the strokes that enclose them.

In practice we relax in our implementation the definition of overlap to detect overlap even if two strokes are partially overlapping.

8 Discussion and Results

Our classifier is based on a support vector machine and it is trained by the user. We are not making strong assumptions on the content of the sketches; therefore our classifier is expected to be flexible and work effectively for a variety of sketch-based applications. The main restriction on the capabilities of the classifier is determined by representation of sketches we construct by means of shape attributes. Specifically our method is designed to work best with diagrams or graphic drawings, such as the ones discussed in section 9. We assess the quality of the classifier on a set of controlled sketches we call clean sketches. We then extend the classifier to accomodate more complex sketches by using the concept of groupings. Our approach for grouping relies for the most part on the training set and it is therefore very adaptable. One of the most attractive aspects of grouping is that it gives us robustness against outliers.

All the concepts described in this paper are implemented in a framework called Sketch Analyzer. Several examples of our work and of Sketch Analyzer are shown in the video accompanying this paper. We tested our approach with sketches representing a house and a human face. We ran our experiments on a 2.13 Ghz Pentium 4 machine and the classifications took no more than 3 seconds to complete. Figure 4 show several sketches that were classified successfully. All the examples contain many outlier strokes that are not part of the training set, such as the window grille for the house. The example on the right contains two semantic groups that belong to the same class – the windows.

Fig. 4. Several sketches that are successfully handled by our technique using respectively a training set for the components of a face and a house. All examples contain several outlier strokes and are designed to put groupings to the test. The windows of the house on the right are an example of semantic grouping.

9 Application: Driving Facial Expressions

We used the concepts described in this paper to implement a sketch-based interface for driving facial expressions. The interface is described in detail in [11] and several examples are shown in the accompanying video. The classifier described in this paper along with grouping is used to recognize the components of the sketch and establish a correspondence with the face model. Each component of the face is then matched with a library of templates to establish the primary features of the facial expression. Lastly, the result is refined by varying the intensity of the various aspects of the facial expression through a parameterization of strokes.

10 Conclusion

We presented an approach for the classification of strokes that is based on SVM and grouping. Our work can handle fairly complex sketches and strives for robustness in the presence of outliers. Shape attributes are one of the key components of our approach and are designed to produce a well-behaved statistical characterization of strokes for generic drawings. Our results show that the classifier can generate robust classifications even with relatively small training sets. Two areas that need improvement and are interesting avenues for future research are the heuristics we use for ranking classification results and pruning the space of groupings, especially for semantic groupings. This work was partially supported by NSF grant No. CCF-0429983. We would also like to thank eFrontier for their generous donations of software licences.

References

1. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: SIGGRAPH 1999. Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 409–416. ACM Press/Addison-Wesley Publishing Co., New York (1999)
2. Chang, E., Jenkins, O.C.: Sketching articulation and pose for facial animation, 19–26 (2006)
3. Thorne, M., Burke, D., van de Panne, M.: Motion doodles: an interface for sketching character motion. ACM Trans. Graph. 23, 424–431 (2004)
4. Sharon, D., van de Panne, M.: Constellation models for sketch recognition. SBIM 2006, 19–26 (2006)
5. Shilman, M., Viola, P., Chellapilla, K.: Recognition and grouping of handwritten text in diagrams and equations. In: IWFHR 2004, pp. 569–574. IEEE Computer Society Press, Washington, DC, USA (2004)
6. Cheong, C.E., Kim, H.Y., Suh, J.W., Kim, H.: Handwritten numeral string recognition with stroke grouping. In: ICDAR 1999, p. 745. IEEE Computer Society, Washington, DC, USA (1999)
7. Saund, E., Mahoney, J., Fleet, D., Larner, D., Lank, E.: Perceptual organization as a foundation for intelligent sketch editing (2002)
8. Saund, E., Moran, T.P.: A perceptually-supported sketch editor. In: ACM Symposium on User Interface Software and Technology, pp. 175–184. ACM, New York (1994)
9. Arnheim, R.: Art and Visual Perception: A Psychology of the Creative Eye (1974)
10. Yang, C., Sharon, D., van de Panne, M.: Sketch-based modeling of parameterized objects. In: Eurographics Workshop on Sketch-Based Interfaces and Modeling, pp. 63–72 (2005)
11. Nataneli, G., Faloutsos, P.: Technical report: Sketching facial expressions. UCLA (2007), URL: `http://www.cs.ucla.edu/~nataneli/pages/publications.html`