α-Shape Based Classification with Applications to Optical Character Recognition

Eli Packer, Asaf Tzadok, Vladimir Kluzner
IBM Research - Haifa, Israel
{elip,asaf,kvladi}@il.ibm.com

Abstract—We present a new classification engine based on the concept of α-shapes. Our technique is easy to implement and use, time-effective and generates good recognition results. We show how to efficiently use the concept of α-shapes of low dimension to support data in arbitrary dimension, thus overcoming the lack of α-shape algorithms in high dimensions. We further show how to inelegantly choose suitable α's to capture desirable shapes that tightly bound the data. We present experiments showing that our technique generates good results with Optical Character Recognition (OCR) tasks. Based also on strong theoretic properties, we believe that our technique can serve as a desirable classification engine for various domains in addition to OCR.

Keywords-Classification; Alpha Shapes; Optical Character Recognition

I. Introduction

In the machine learning domain, a common scenario is to train an engine with a set of known objects (both their descriptions and class identifications) and then classify unknown objects. The classification problem is motivated in many areas and plays particularly a central role in Optical Character Recognition (OCR for short), the application on which we focus in this work. Using feature vectors extracted from each sample is a popular classification scenario. In this process, the classification engine is trained with multiple feature vectors and then classify unknown objects by extracting their feature vectors and comparing them against the trained data. Intuitively, the closer the classified feature vectors to the trained ones, the higher the likelihood of the object to belong to the corresponding class. In this work we develop a classification engine based on the concept of α-shapes. We next present it. Let S be an input point set in $\mathbb{R}^m$ (m is an arbitrary integer). Given a parameter α, let $B(p,\alpha)$ be the n-dimensional ball of radius α centered at point p. Then the α-hull of S with radius α is defined as $H_\alpha(S) = (\bigcup_{interior(B(p,\alpha))\cap S=\emptyset, p\in\mathbb{R}^m} B(p,\alpha))^c$. In words, the α-hull is the set of points that do not lie in any empty open disk of radius α. If we substitute the arcs of the α-hull with straight edges, we get the α-shape. We denote the α-shape by $A_\alpha(S)$ (see Figure 1 for illustration). It is well known that the set of edges of the α-shape is a subset of the Delaunay triangulation. Note that the α-shape is a generalization of the convex hull: When α approaches infinity, the α-shape converges to the convex hull. As α shrinks, the shape shrinks too and may generate tunnels and voids and may also be split to several connected shapes. When $\alpha = 0$, the shape contains nothing but the input points. A central objective of the α-shape is to capture the shape of a set of points in $\mathbb{R}^m$. In this work, we construct α-shapes to form shapes to the feature vectors; each feature vector can be viewed as a point in $\mathbb{R}^n$. We can then determine the likelihood of any feature vector to belong to the class the α-shapes represent, based on its location with respect to a specific α-shape. The closer the feature vector to the α-shape, the higher the likelihood of it to belong to the corresponding class. Using α-shapes for that purpose has two limitations: 1) the unavailability of efficient α-shape algorithms for any dimension beyond three and 2) the necessity to determine a desirable α with which a suitable shape is generated. In this work, we explain how to efficiently overcome these limitations to construct a reliable classification engine.

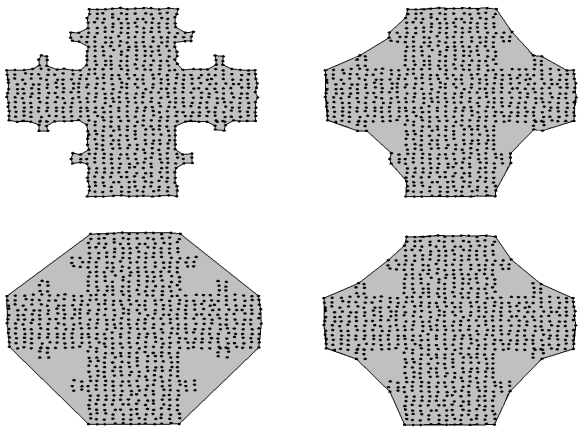


Figure 1. Clockwise from top-left: α-shapes (shaded) of increasing α values

In this work we are interested in applying our technique for OCR tasks. The common recognition process in this field usually consists of binarization, segmentation, classification and post processing improvement methods such as spelling checker. Our focus is in the classification phase in which segmented components have

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.77

to be classified as a certain character. Related Work. Over the years many classification techniques have been developed. Among the most famous ones are: support vector machines, neural nets, Ada-boost, decision trees and K-nearest neighbors. The α-shape/hull concept was introduced in [1]. A more recent work [2] extends the α-hape concept to $\mathbb{R}^3$. Lucieer and Kraak [3] constructed α-shapes mainly for visualization purposes, but they also used them for classification. However, their technique is limited to $\mathbb{R}^3$ and to a small number of features. Their work is also limited by the fact that they appear to have manually determined the value of α to fit their model. Edelsbruner [4] presented an excellent survey of the work devoted on α-shapes over the years. The rest of the paper is constructed as follows. In the next section we describe our classification engine. In Section III we describe the advantages and the usefulness of our technique. In Section IV we present experiments we performed with our implementation. We conclude in Section V.

II. α-Shape Classification Engine

Given a set of classes C, each of which is represented by multiple feature vectors obtained from multiple samples, we build a classification engine as follows. Let $c \in C$ be any class with feature vectors V_c. We partition V_c to pairs and for each construct a two-dimensional α-shape. When classifying an unknown sample u, we test it against any $c \in C$. For each $c \in C$, we partition u to pairs in the same way we partitioned V_c and compute the distance of each pair of u from the corresponding α-shape[1]. Summing up the results of all pairs, we can determine the likelihood of the sample to be c. We explain our technique by showing how to build one α-shape (Subsection II-A). Then we describe how we collect the α-shapes into a set that represents a single class. We then explain how we score a sample against a class (Subsection II-B).

A. Determining α

Determining α is a crucial step in the construction of the α-shape. Only certain values of α will incur desirable shapes. The right shape is notably subjective and may be ambiguous in different scenarios. In Figure 1, for example, the top-left figure might appear to be the shape of the points, since the other figures seem to capture redundant space. However, only certain values of α will yield the shape on the top-left figure. We next formalize the way to choose α so that the corresponding α-shape will reliably capture the areas that include points. We choose an α that is the minimum with which the following condition holds true: Condition 1. $A_\alpha(S)$ contains disjointed polygons that collectively contain all input points. The polygons may have holes, each of which may have islands and so on. We continue by making a few observations about the properties of the α-shapes.

[1]If u is contained in the α-shape, the distance is defined to be 0. Alternatively, we can compute the distance from the skeleton of the α-shape. We did not implement this alternative for this work.

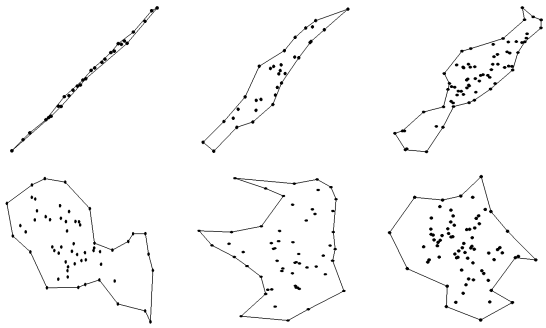


Figure 2. Different α-shapes obtained with our software; The ratio $|\mathcal{A}_\alpha(S)|/|B(S)|$ increases from top to bottom and from left to right

Lemma 2.1: Let α_1 and α_2 be two positive values such that $\alpha_2 < \alpha_1$. Then $A_{\alpha_2}(S) \subseteq A_{\alpha_1}(S)$.

Proof: Let $C_\alpha(S)$ be the union of all empty balls of radius α with respect to S. Note that every empty ball of radius α_1 can be formed by an infinite collection of empty balls of radius $\alpha_2 < \alpha_1$. Then $C_{\alpha_1}(S) \subseteq C_{\alpha_2}(S)$. Based on the definition of α-shape, it follows that $A_{\alpha_2}(S) \subseteq A_{\alpha_1}(S)$. ■

Lemma 2.2: There exists a minimum $\alpha > 0$ for which condition 1 holds and any $\alpha' > \alpha$ satisfies condition 1 as well.

Proof: From Lemma 2.1, if condition 1 holds for a certain α, it follows that it also holds for any $\alpha' > \alpha$. Since when $\alpha = 0$, $A_\alpha(S)$ contains just the input points, and condition 1 is violated. For sufficiently large α, the α-shape notably coincides with the convex hull, and condition 1 clearly holds. Then there must be a minimum $\alpha > 0$ as the lemma claims. ■

Finding the minimum α for condition 1 and building the corresponding α-shape, we get a tight shape that includes all points. Let R be the diameter of S. Based on Lemma 2.2 we find the minimum desired α with a binary search on the values $(0 \dots R)$ with a sufficient small ε-tolerance (Note that for any $\alpha \geq R$, $A_\alpha(S)$ coincides with the convex hull of S.) For each α of the binary search, we build the α-shape [1] in $O(n \log n)$ time and check whether it consists of disjoint polygons that collectively include all points of S. We do this by constructing a trapezoidal decomposition of the shape and marking each trapezoid as belonging or not belonging to the shape (time $O(n \log n)$). Then we perform point location queries for each point of the input to test

whether it is located inside a cell that belongs to the shape (together time $O(n \log n)$). Based on the above description, the number of iterations of the binary search is $O(\log \frac{R}{\varepsilon})$ and each iteration consumes time $O(n \log n)$. We conclude with the following theorem:

Theorem 2.3: We find the minimum α and construct its α-shape in time $O(n \log(\frac{R}{\varepsilon}) \log n)$.

B. Using α-shapes for Classification

Since no available efficient constructions exist for α-shapes in high dimensions (today the known efficient algorithms are in $\mathbb{R}^2$ and $\mathbb{R}^3$), we partition the feature vectors to pairs and construct an α-shape for each[2]. Thus, we must come up with a criteria for pairing up the features. We use the following rule. Consider all possible pairs and let Q be the set of points in any. Sort the pairs by increasing ratio of $|A_\alpha(Q)|/|B(Q)|$ where B is the axis aligned bounding box of Q (see Figure 2). Then we greedily collect α-shapes such that each feature vector appears in at most one α-shape. We further hold a threshold T to avoid using pairs with ratios larger than T. This means that potentially some of the feature vectors will have no partners. We will use them instead in an $\mathbb{R}^1$ structure that calculates the normalized distance from the center of mass. These structures will appear after the α-shapes in the sort. In the classification phase, we then partition the features of the tested object for each class accordingly and sum up the distances for each class to get a numeric result for each. The smaller this sum is, the higher the likelihood that the tested instance belongs to the class[3]. We use the following optimizations for speeding up the process. 1) While constructing the α-shapes, we use some of the input samples for testing. We continue adding pairs in their order (see above) and after each addition, we test the rate of correctly classifying the tested characters. We stop when we see a high correct classification rate. 2) After the construction of the α-shapes, we should have a solid numeric criteria on a threshold for accepting or rejecting a character. Then, during the classification phase, we reject when we reach this threshold, and we do not continue processing the input with all α-shapes. Since the tested sample should be rejected by most classes, we can likely stop processing and reject it early. This was the case in our experiments. To determine whether a point is inside the α-shape and calculate the distance from its boundary, we use a line-segment Voronoi diagram [5]. Constructing the Voronoi diagram takes $O(n \log n)$ time and each query takes $O(\log n)$ time. To save multiple location queries, we overlay α-shapes of different classes that share the pair of features. We mark each induced cell with the nearest edges of each class. Suppose there are k classes, each of which has a certain α-shape of the same pair of features. then instead of performing k queries we can then query once. The following theorem summarizes our technique.

Theorem 2.4: Constructing the α-shape classification engine takes $O(km^2 n \log(\frac{R}{\varepsilon}) \log n)$ time and $O(kmn)$ space where k is the number of classes, m is the size of the feature vector space, and n is the maximum number of samples taken for a class, R is the largest diameter of any points obtained with pairing up two features in any α-shape, and ε is the tolerance parameter for the binary search we defined above. Each query takes $O(km \log n)$ time.

Proof: The complexities of the classification engine derive from multiplying the construction complexity of each α-shape by the number of times we construct (for time) and keep (for space). Each classification query involves querying each class; note that each has $O(m)$ α-shapes. Since each α-shape query takes $O(\log n)$ time, the query time follows. ■

III. Discussion: Advantages of our Technique

We found our technique very useful in several aspects. We discuss them and the motivation to use our technique for classification applications. Using shapes that geometrically capture the feature vector data is the basis of our technique. By computing suitable α values, we obtain a data structure that tightly bounds the regions belonging to a specific class. As opposed to statistical techniques like Mahanalobis distance, we do not make any assumptions regarding the distribution, and thus we are more robust to any distribution. Having these properties, our technique is suitable for determining whether a specific element belongs to a class (this process is called one-class classification). Performing one-class classification is important in several applications, such as in marking noise for exclusion from a result. Bounding the region tightly, we increase the chances that noise will be excluded from any class and thus not included in the result. In Section IV-A, we demonstrate the importance of removing noise in OCR tasks. Our technique takes advantage of the correlation of two features. Consider Figure 2: suppose we project the features onto one of the axes. This results in the samples covering the entire range. However, it is the shape in $\mathbb{R}^2$ that adds important classification information that may be crucial for obtaining reliable results. Our technique is simple to understand, implement, and use. Its direct geometric definition makes it easy to understand and should provide improvement ideas in the future. Our technique uses well-known geometric data structures that are today available in geometric libraries. It is easy to use because it does not require parameter settings

[2]We are planning to extend the engine to $\mathbb{R}^3$ in the future—the same ideas we present in this report hold in $\mathbb{R}^3$ too

[3]Note that we normalize the feature vector values.

as other techniques do (e.g. Neural Nets and Support Vector Machines). Automatically computing the data structures, we release the user from making tedious parameter settings. Our technique is flexible for different kinds of data and tasks. We can use it to seamlessly work on one, two or multiple classes. It can also work on any dimension with arbitrary large feature vectors.

IV. Experiments

We experimented with two kinds of OCR tasks. The first is related to Gothic characters obtained by scanning old library books. The second is the segmentation of screen antialiased text.

A. Old Gothic Characters

Printed heritage digitization is a central challenge in the IMPACT project [6] that deals with text recognition of old library books for indexing and searching purposes, using OCR techniques. A major obstacle in recognizing the text is that many defects appear over time, as illustrated in Figure 3. These defects result in meaningless segmented components that confuse the OCR engines which classify them as characters. To solve this problem, we must classify these cases as junk instead. Our data consist of Gothic characters retrieved from [7]. We segmented the characters using the OCR software of ABBYY [8]. We used our engine to classify junks in the following way. After training the engine, we used other samples to learn two types of scores—one is when a sample is computed against its class and the second is when a sample is computed against different classes. Using k-means (where $k = 2$), we find a threshold T, such that if the score we get is larger than T, we conclude that it does not belong to the class. Otherwise, it does belongs.

Our results showed a clear gap between these two types of scores. We omitted a very few outliers that gave large scores to images of the same character. To choose whether a segmented character is junk, we then calculated the scores and compared the lowest one (L) against T. We classified this as junk if $L > T$. To evaluate our technique, we compared it with the K-nearest neighbor (where we set $K = 5$; this value gave us the best results), Mahanalobis distance and convex hulls techniques. The later comparison is done by simply replacing the α-shape of our engine with the convex hull. The idea is to see by how much, if at all, shrinking the convex hull helps. For each competing technique, we also found a corresponding threshold for rejecting characters. We tested 70 classes and incidents of junk (spot, smearing elements, etc.). For each character, we had at least 30 input samples. Together the database includes 3200 samples. Out of

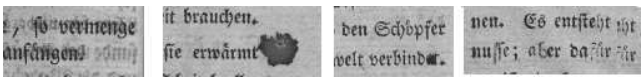


Figure 3. Examples of common defects that appear in old books; left to right: reflection of the opposite page, spots, erased characters and smearing

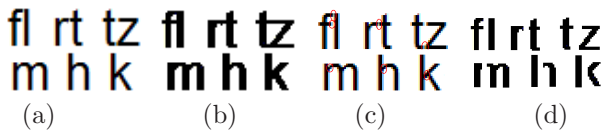
these, we used all but five samples to construct the α-shapes and used the remaining five for testing. In our experiment, we calculated how many samples had false negative classification (rejecting them from their correct class) and how many had false positive classification (accepting the character as belonging to a wrong class). Our α-shape engine gave the best results, as summarized in Table I. The advantage over using the Mahanalobis distance can be explained by the fact that the feature vector distribution is not Gaussian. We also calculated the training and querying time. The average time it took to build a set of α-shapes for a single class was 5.25 seconds. This time included the computation of 2500 α-shapes, each with a time of 0.0021. This process should occur once during the training of the engine, and then the data can be saved for further processing in the future. The average time of each test query was $3.68e^{-4}$. We finally note that the features we used to construct the feature vectors are Harr, Schlick, Affine and Projection [9]. Since we used different variations for each kind of feature, we generated a vector 58 atomic features for each sample.

	α-shape	Convex hull	K-nearest neighbor	Majanalobis distance
False negative rate	1.8%	4.5%	2.2%	4.7%
False positive rate	5.2%	13.5%	7.3%	9.6%

Table I
False positive and false negative rates for classifying old Gothic font

B. Segmenting Antialiased Text

the most popular way to make screen characters appear smoother is to convert them to antialiased characters. During the conversion, characters tend to smear and sometimes touch neighbor characters. This, in turn, makes the segmentation and recognition more difficult. Another issue is that the pixels of the characters depend on the neighborhood and thus cannot be precisely learnt in advance. When characters touch one another, we need to separate them for further recognition. However, determining which components belong to which character (or characters) is diffcult. In most cases, the erroneous connection of two characters is weak (see Figure 4) and can be detected to assist with the separation of the characters. Unfortunately, we cannot simply separate



(a) (b) (c) (d)

Figure 4. Antialiased text: the top row of each subfigure contains characters that became connected in the process of antialiasing; the bottom of each subfigure contains single characters that have weak connections between two parts. (a) The characters being antialiased (b) Binarization of the antialiased text (c) Marking the areas of weak connections (d) The two parts of each component after separation

components at weak connections because some single characters also have weak connections among their own parts (see Figure 4). Thus, we must have good criteria for determining when to break components. We used our technique for this task on image characters generated with a Windows character generator API. The idea is that components that correspond to single characters will get low score in comparison to components that belong to more than one character. For each component with a weak connection, we computed the score of both the entire component and the two parts. When separating the character into two parts, we computed the average score of each. We compared the scores to thresholds learned in advance to decide whether to separate the characters. We tested against the same classifiers we used in Section IV-A. For each case, we compared the score obtained for testing both the full component and the two induced parts. We measured the rate of errors (categorizing two merged characters as one and categorizing one character as two). The error rates are summarized in Table II. We again we achieved good results that defeated the other classification techniques we tested against.

	α-Shape	Convex Hull	K-Nearest Neighbor	Majanalobis Distance
Error rate	7.5%	16.7%	8.1%	17.3%

Table II
Error rates for segmenting antialiased text

V. Conclusions

We presented a novel technique for classification tasks in feature-based learning engines. We developed a new data structure based on α-shapes. Our technique takes advantage of the correlation of features and their ability to tightly define regions that represent classes. We implemented our technique and experimented with two kinds of OCR tasks, from which we obtained good results. Using our technique as a general classifier gave good results as well. We hope to also establish our technique for that purpose in the future. We envision continuing in several directions to improve the success rate of our technique. By efficiently expanding our method to higher dimensions, we may be able to capture correlations of several features that can assist in further improving our results. This is a challenging task, since the α-shape algorithm was so far devoted to only $\mathbb{R}^2$ and $\mathbb{R}^3$. Other possible extensions involve using weighted α-shapes, in which different points have different weight, and modifying the α-shape by dilating it based on local curvature. Both approaches should provide us more flexibility with the distance calculation.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-201 under grant agreement 215064. The authors thank Dan Chevion for helpful discussions.

References

[1] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," IEEE Trans. Inform. Theory IT-29, pp. 551–559, 1983.

[2] H. Edelsbrunner and E. P. Mucke, "Three-dimensional alpha shapes," ACM Trans. Graph., 1994.

[3] A. Lucieer and M.-J. Kraak, "Alpha-shapes for visualizing irregular-shaped class clusters in 3d feature space for classification of remotely sensed imagery," Visualization and Data Analysis, 2004.

[4] H. Edelsbrunner, "Alpha shapes - a survey," in Tessellations in the Sciences; Virtues, Techniques and Applications of Geometric TilingsVisualization and Data Analysis, 2010.

[5] C. Burnikel, K. Mehlhorn, and S. Schirra, "How to compute the voronoi diagram of line segments: Theoretical and experimental results," in Proceedings of the Second Annual European Symposium on Algorithms, ser. ESA '94, 1994, pp. 227–239.

[6] IMPACT Project. http://www.impact-project.eu.

[7] C. von Eckartshausen, Aufschlüsse zur Magie aus geprüften Erfahrungen über verborgene philosophische Wissenschaften und verdeckte Geheimnisse der Natur. Lentner, 1791.

[8] ABBYY. http://www.abbyy.com/.

[9] M.-K. Hu, "Visual pattern recognition by moment invariants," Information Theory, IRE Transactions on, vol. 8, pp. 179–187, 1962.