Mining Shock Graphs with Kernels

Fréderic Suard and Alain Rakotomamonjy and Abdelaziz Bensrhair

Manuscript submitted December, 2006;

F. Suard and A. Bensrhair are with LITIS INSA de Rouen, Avenue de l'Universite 76801 Saint Etienne du Rouvray (email frederic.suard@insa-rouen.fr)

A. Rakotomamonjy is with LITIS Univ. de Rouen, Avenue de l'Universite 76801 Saint Etienne du Rouvray. email: alain.rakoto@insa-rouen.fr

Abstract

A common approach for classifying shock graphs is to use a dissimilarity measure on graphs and a distance based classifier. In this paper, we propose the use of kernel functions for data mining problems on shock graphs. The first contribution of the paper is to extend the class of graph kernel by proposing kernels based on bag of paths. Then, we propose a methodology for using these kernels for shock graphs retrieval and other shape mining problems. Our experimental results show that our approach is very competitive compared to graph matching approaches and is rather robust. Others experiments illustrate that with such kernel functions it becomes possible to apply statistical pattern recognition algorithms to shock graphs.

Index Terms

Kernel methods, object recognition, shock graphs, Support Vector Machines.

I. INTRODUCTION

Object recognition is still a difficult and challenging problem. For solving such problem, it is expected that several cues, which give statistical and structural information about objects, are needed to be integrated. One of this important cue is the object shape. In this paper, we will focus on this cue. Like many real-world data, as texts or molecular structures, shape can be represented as graphs. These graphs are obtained after an appropriate skeletonization of the shape [8], [**?**]. Once these graph representations of shapes are obtained, the objective is to apply high-level algorithms for classifying or clustering objects. In that case, these algorithms suppose the definition of a similarity measure on graphs.

When addressing the problem of shape matching, several approaches have been used for defining graph similarity. For instance, approaches based on edit-distance [28] or based on *maximum common subgraph* have been considered [3]. In this paper, we propose to address this problem of measuring shape similarity through the theory of definite positive kernels. Using such theory, it becomes possible to define a kernel function that acts as an inner product on the graph space.

Our contribution in this paper is two-fold. First, we emphasize that several graph-based kernels [15] are built upon two ingredients : path generation and set of paths similarity measures. After, having highlighted this point, we propose other graph-based kernels that differ in how similarities between set of paths are computed.

Then, the second and principal contribution of this paper is the analysis of graph kernel usefulness for structural pattern recognition of shapes. We propose a methodology for applying such kernels to skeleton recognition, shape retrieval and shape clustering. We show that our approach compares favorably to current approach while opening interesting perspectives for statistical and structural shape classification and paving the way for applying powerful statistical pattern recognition algorithms to structural pattern recognition problem.

A. Related works

The idea of representing shapes from shape skeleton shock graphs [8] has become popular owing to the several properties that shock graphs bring. For instance, shock graphs are typically translation and rotation invariant representations of a silhouette. However, when dealing with such shock graphs, the problem of shape recognition or shape retrieval becomes difficult due to the need of graphs comparison.

In order to compute a similarity measure between two shock graphs, some works have considered a graph matching approach. Sharvit et al. [29] use graduated assignment graph matching algorithm [11] for evaluating matching structures in two shock graphs. This same graph matching algorithm has been used by Di Ruberto [24] for shape recognition. However, the proposed approach is different since Di Ruberto assigns numerical attributes on nodes and edges of the graph, these attributes being computed from the shape skeletons. Still belonging to the graph matching approach, Demirci et al. [6] have proposed a many-to-many matching approach. Their idea is based on the evidence that in shape matching the one-to-one node or edge matching can be restrictive. Hence, they reduce the many-to-many weighted graphs matching to the matching of weighted sets in a normed space by embedding the graph into that normed space. Interestingly, Demirci et al. could have proposed a graph kernel for their graph matching algorithm if they have embedded graphs into an inner product space instead of a normed vector space.

Other approaches consist in converting shock graphs into trees and then looking for a subgraph isomorphism [31] or by finding maximal cliques [31]. Sebastian et al. [28] have also considered an approach based on graph edit-distance.

Kernel methods such as Support Vector Machines or Kernel Principal Component Analysis are becoming popular for their high performance [26]. These kernel method algorithms rely

on the fact that computations involving the data are performed through a kernel function. This latter acts as an inner product between the data in some embedding space. Graph kernels has been essentially developed for classifying chemical compounds [14], [20], [18]. Some kernels for labeled graphs have already been presented in the literature [15]. For most of these kernels, the graph is embedded in a feature space composed of label sequences obtained by a graph transversal. Then the kernel value is computed by measuring similarity between label sequences. As stated by Kashima et al. [15] the difference between graph kernels lies in how the graph transversal is performed and how the label sequences are compared. For instance, Kashima et al. [14] proposed a random walk on the graph for obtaining labeled sequences (that can be of infinite length). Then they use a weighted average for computing the kernel value from the similarity of the sequences obtained on the two graphs. The main strength of Kashima's work is that they provided an efficient algorithm for computing the kernel even when sequences can be of infinite length. Mahé et al. [19] extended this approach by proposing a different model of random walk on graphs. It can also be shown [15] that the framework proposed by Kashima et al. also consider the so-called geometric and exponential graph kernels of Grtner [9].

B. Contributions

One of the purpose of this paper is to demonstrate the usefulness of the kernel approach for shock graph data mining problem. The expected results are that by using kernels one can get similarity measure at least as good as those proposed in the literature. The add-on values brought by kernels would be all the theoretical framework and the kernel method algorithms that have been recently developed. As far as our knowledge goes, very few works have been done on graph kernels for image recognition [32].

In this work, all the steps concerning the transformation of a shape into a graph have been borrowed from the literature. For instance, shape skeletonization has been performed by means of Dimitrov et al. algorithm [8] while the skeleton to graph transformation we applied is rather similar to the one proposed by Di Ruberto [24].

Mainly, we have focused on proposing some new graph kernels which still belong to the label sequences framework. In our case the labeled sequences have be obtained by traversing the graph using shortest path algorithms. Thus, we embedded a graph into a finite dimensional space of sequences and then compute inner product between two sets of sequences. Several approaches,

which are inspired from works about kernels on sets [16], [36], [7] are then proposed for evaluating the inner product between sets of sequences. Our main motivation for proposing these variants of label sequences approach is the need of considering only limited length sequences. Advantages of limited length sequences are three-fold. Firstly, computational complexity of the algorithm can be reduced by considering only paths of a given length. Secondly, considering infinite length sequences may produce paths that tend to "blur" the graph similarity. This rationale is the argument used by Mahé et al. for proposing a new probabilistic model for the random walk [19]. The third advantage, although it may have less practical consequences, is that condition for convergence of kernel value is no more needed.

C. Organisation

Section 2 gives a short introduction on kernel function and the theory of positive definite kernels. Section 3 presents our bag of path framework for graph kernels. We discuss in this section several approaches for computing similarities between bag of sequences. Section 4 describes the approach we used for producing graphs from shapes. We also address in this section the different attributes that can be used as node or edge labels of the graph. Section 5 shows several numerical experiments we carried out while section 6 concludes with some discussions and directions for future works.

II. KERNELS

Kernel methods and Support Vector Machines have been increasingly used during the last ten years for solving various data mining related problems. Strengths of such methods essentially lie in the flexibility they offer for analyzing and comparing many types of data (even non-vectorial data such as graphs). This section aims at briefly introducing the theory of positive definite kernels.

A. Definition

Let us define as $\mathcal{S} = \{x_1, \cdots, x_n\}$ be a set of n objects to be analyzed and each object $x_i \in \mathcal{X}$. A positive definite kernel on the set $\mathcal{X}$ is defined as :

Definition 1: A function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a positive definite kernel iff for any $n > 0$, any $\{x_i\}_{i=1}^n \in \mathcal{X}^n$ and any sequence $\{a_i\}_{i=1}^n \in \mathbb{R}^n$, it satisfies

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j K(x_i, x_j) \geq 0$$

and it is symmetric, that is, for any x and x' belonging to $\mathcal{X}$, $K(x, x') = K(x', x)$.

The main advantage on focusing on such positive definite kernels is that they provide an implicit mapping of the space $\mathcal{X}$ into a reproducing kernel Hilbert space.

Theorem 1: For any positive definite kernel K, there exists an unique reproducing kernel Hilbert space $\mathcal{H}_K$ and a mapping $\Phi : \mathcal{X} \rightarrow \mathcal{H}_K$ so that :

$$\forall x, x' \in \mathcal{X},\ K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}_K}$$

where $\langle \cdot, \cdot \rangle$ represents the dot product in $\mathcal{H}_K$.
Then owing to this implicit mapping, it becomes possible to use the space $\mathcal{H}_K$ for analyzing or applying inner product algorithms to elements of $\mathcal{X}$. This mapping, also known as the *kernel trick* has opened the way to applications of a large class of linear algorithms like Support Vector Machines [1], [4] or principal component analysis [27] to non-vectorial data [10], [30].

One contribution of our work is to highlight particularities of some graph kernels proposed in the literature [15] and to extend the family of positive definite kernels for graphs.

It is worth to note that new kernel functions can be obtained by using some closure properties of kernels that are summarized in the following [30]:

Proposition 1.1: Let K_1 and K_2 be two positive definite kernel functions over $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, $a \in \mathbb{R}^+$ then the following functions are positive definite kernels :

1) $K(x, x') = K_1(x, x') + K_2(x, x')$
2) $K(x, x') = aK_1(x, x')$
3) $K(x, x') = K_1(x, x')K_2(x, x')$

These properties will be used in the following for showing that the kernels we proposed are indeed positive definite.

III. Bag-of-paths based graph kernel

Let us introduce the notation that will be used throughout the paper. Define V as a finite set of vertices and $E \subset V \times V$ a set of edges. A graph G is defined as $G = (V, E)$. The order of G is given by the cardinality $|V|$ of V.

For a labeled graph, a labeling function is also defined. This function $l : V \cup E \rightarrow \mathcal{X}$ assigns a label $l(x)$ to any vertices or edges x. A path h on a graph G can be defined by a finite-length sequence of vertices. A path h of length n is then defined as $h = (v_1, \cdots, v_n)$ with $\forall i \in [1, n-1]$, $(v_i, v_{i+1}) \in E$. Associated to such path, a labeled sequence can be defined as $h_S = (l(v_1), l(v_1, v_2), \cdots l(v_{n-1}, v_n), l(v_n))$, where h_S describes a sequence of nodes and edges label. In this paper, we will only consider connected graphs which means that there exists a path from any vertex to any other vertex.

When designing a kernel function on graphs, the problem is to define an embedding of the graph into a inner product space. For instance, we can note that a graph can be represented by a set of paths resulting from the transversal graph by a random walk or by any other method such as collecting all the paths between any two pairs of vertices. Of course, this new representation of a graph by a set of paths is less informative since some structural information are lost through the embedding. However, the advantage of such representation is that similarity between graphs can be based on the similarity between these bags of paths.

The graph kernel of Kashima et al. [14] fits into this bag-of paths kernel framework. In fact, for this kernel each graph is represented by a set of paths (of possibly infinite dimension) obtained through a random walk on the graph, similarity between paths is defined with the kernel function $K_L(h, h')$. Usually, $K_L(h, h') = 0$ if their lengths are different and otherwise we have :

$$K_L(h, h') = K_v(l(v_1), l(v'_1)) \prod_{i=2}^{n} K_e(l(v_{i-1}, v_i), l(v'_{i-1}, v'_i)) K_v(l(v_i), l(v'_i)) \tag{1}$$

This latter equation suggests that the kernel on path K_L needs the definition of kernels on vertex label K_v and edge label K_e. These two kernels can be any positive definite kernels defined on the sets of labels. Hence, graph labels can be any structured data that admits a kernel function. For this work, when considering kernel on paths, we will use equation 1.

The last ingredient for the Kashima's bag of path kernel is merging all pair of paths similarity measures into a single score. They used a weighted averaged approach :

$$K(G_1, G_2) = \sum_{h_1, h_2 \in V_1^\star, V_2^\star} p_1(h_1) p_2(h_2) K_L(h_1, h_2) \tag{2}$$

where p_1 and p_2 are some probability distributions on the set of finite-length sequences of vertices $V_1^\star$ and $V_2^\star$. Actually, only paths have positive probabilities under p_1 and p_2 and these probabilities

distributions are defined according to the path generation on the graph. We can also note that the kernel proposed by Mahé et al. [19] also fits in this framework. They have proposed an extension of the Kashima's graph kernel by modifying the probability model of the random walk. They introduced a second order Markov random walk model that eliminates paths containing back and forth walks. Avoiding such tottering phenomenon allows them to improve the graph similarity measure and to improve the performance result of their discrimination problem.

Extensions of Kashima's idea can be brought further for instance by proposing different approaches for generating paths or for merging several path similarity measures. As an example, paths can be generated by means of a deterministic approach rather than a random walk graph transversal.

Before dealing with the way we generate paths on the graph, in the next paragraph, we consider that graphs have already been embedded into bag-of-paths and we look for different methods for merging path similarity measures. This part of the work exploits results from kernel on sets which is a very similar problem [16], [35]. In fact, kernels on sets allow one to define an inner-product on unordered set of objects (in our case, the objects are paths). One possible approach [35] for building a kernel on sets is first to compute a similarity score between elements of each set according to a so-called minor kernel and then is to merge the resulting scores into a higher level similarity score that defines the inner product between the two sets.

A. Merging path similarity measures

Supposing that each graph G_1, G_2 has been transformed into a set of paths respectively P_1 and P_2. A first easy way for obtaining a graph kernel would be the mean average kernel :

$$K(G_1, G_2) = K(P_1, P_2) = \frac{1}{N_1}\frac{1}{N_2}\sum_{i:h_i \in P_1}\sum_{j:h_j \in P_2} K_L(h_i, h_j) \tag{3}$$

where N_1 and N_2 are respectively the cardinality of the sets P_1 and P_2. This kernel is very simple but has the disadvantage of using all pairs of similarity between paths. Hence, a large number of pairs of paths with low similarity measures can "hide" a large similarity between two paths. For addressing such problem, it is possible to consider a matching kernel [35]. For such kernel, the similarity score of the two graphs is :

$$K(G_1, G_2) = K(P_1, P_2) = \frac{1}{2}[\hat{K}(P_1, P_2) + \hat{K}(P_2, P_1)] \tag{4}$$

with

$$\hat{K}(P_1, P_2) = \frac{1}{|P_1|} \sum_{i:h_i \in P_1} \max_{j:h_j \in P_2} K_L(h_i, h_j)$$

This kernel aims at matching each path of P_1 with a path of P_2, which is an interesting approach but leads to a non positive-definite kernel. Although such non-positive kernel can be used for learning [12], we propose here a positive-definite approximation of this matching kernel. If we consider the distance d_L induced by the kernel K_L as :

$$d_L(h_1, h_2)^2 = K_L(h_1, h_1) + K_L(h_2, h_2) - 2K_L(h_1, h_2)$$

then, it has been shown by Haasdonk et al. [13] that the kernel $K_{d_L}(h_1, h_2) = \exp\left(-\frac{d_L(h_1,h_2)^2}{2\sigma^2}\right)$ is positive definite for all $\sigma > 0$. Our positive-definite matching kernel is then :

$$\hat{K}(P_1, P_2) = \frac{1}{|P_1|}\frac{1}{|P_2|} \sum_{i:h_i \in P_1} \sum_{j:h_j \in P_2} K_{d_L}(h_i, h_j) \tag{5}$$

which is based on the approximation of

$$\max_{j:h_j \in P_2} K_L(h_i, h_j) \quad \text{with} \quad \sum_{j:h_j \in P_2} K_{d_L}(h_i, h_j)$$

With this approximation, we include in the kernel values all the similarity measures between pairs of paths, but with an exponentially decreasing influence as the distance d_L between the two paths increases. Hence, the contribution of the most matching pair of paths is still large compared to other pair of paths. In fact, if σ is small (and appropriately chosen), the kernel value $K_{d_L}(h_i, h_j)$ is equal to 1 when $h_i = h_j$, while otherwise it will take small values for other paths. The width σ of the gaussian kernel can be considered as a parameter which defines the zone of influence of a path h_i while looking for similar paths. In this sense, the gaussian kernel acts like the Geometric Compactly Supported Kernel of Boughorbel et al. [2].

Another approach we have considered for building kernel on bag of paths is based on the kernel on sets described by Desobry et al [7]. The underlying idea of this kernel is the following. Each graph is represented according to a set of paths, and the two graphs can be considered as similar if their respective sets of paths lie in the same part of the space of finite-length sequences. According to Desobry and Davy, this boils down to measure the similarity of the two sets of path P_1 and P_2 by comparing the support of the probability density of each set of paths. Since, we have to deal with estimating such support, Desobry et al. propose to use a one-class SVM

[7]. The estimation f of the distribution support is obtained as a solution of the One-Class SVM optimization problem :

$$\min_{f\in\mathcal{H},b} \frac{1}{2}\|f\|^2_{\mathcal{H}} + \frac{1}{\nu n}\sum_i \max(0, b - f(x_i)) - b$$

where $\mathcal{H}$ is the reproducing kernel Hilbert space induced by the kernel on path K_L, which we suppose is so that $K_L(h,h) = 1$ and ρ is a bias. $\nu \in [0,1]$ is a regularization parameter that is directly related a given level-set of the distribution support [26]. The distribution support of each set of paths S_1 and S_2 of are obtained by applying the One-Class SVM to each set and the contour of S_1 and S_2 are respectively :

$$f_{P_1}(h) = \sum_i \alpha_i^{P_1} K_L(h_i, h) - b_{P_1} \qquad f_{P_2}(h) = \sum_j \alpha_j^{P_2} K_L(h_j, h) - b_{P_2}$$

Then, we have defined the inner product between regions and thus between the graphs generating the paths as :

$$K(G_1, G_2) = K(P_1, P_2) = \langle b_{P_1}, b_{P_2}\rangle \cdot \sum_{i:h_i\in P_1} \sum_{j:h_j\in P_2} \alpha_i^{P_1}\alpha_j^{P_2} K_L(h_i, h_j) \tag{6}$$

which is obtained through the product of two inner products respectively between the bias of the two functions and the two RKHS functions defining S_1 and S_2. According to properties of positive definite kernels, the resulting kernel is positive definite.

B. Set of paths

The different graph kernels we proposed above suppose that each graph has been represented by a set of paths. Generating this set of paths can be done in several ways. For instance, Kashima et al. have proposed to use a random walk. This approach poses the problem of the convergence of K_L (equation 1) when the number of random paths becomes infinite. Besides, depending on the expression of K_L, long paths generate small kernel values that can be neglected compared to short paths. This can be easily understood when considering equation 1 with kernels on edges and nodes that provide values lower than 1.

In this work, we propose to represent a graph according to the set of shortest paths between each pair of vertices instead of using all the paths obtained by random walks. This approach has several advantages. At first, the problem of finding the shortest path between two vertices is

already a classical result in the domain of graph theory. Dijsktra's algorithm is the most popular and a very simple algorithm for solving such problem. We will use this algorithm throughout the paper.

When using a set of shortest paths, all the paths are preprocessed and then compared. Hence it becomes easy to discard some paths from the set. Within this context, the maximal length of paths in the set can be easily handled. It is then possible to reduce the computational time of the overall kernel computation by limiting path length. Some flexibility in the representation is introduced since one can, for instance, limit the paths to length 0 which results in a graph kernel based on a set of vertices.

A resulting side effect of using shortest paths is that the set of paths is of finite cardinality and thus, no convergence problem appears. Furthermore, considering shortest paths between vertices naturally prevents from tottering phenomenon.

C. Comments on computational complexity

In this paragraph, we provide a short analysis of the computational complexity of our bag-of-paths based graph kernel.

Supposing that the graphs we considered are of order N, the complexity of our approach depends on several points. The first point is related to the way how paths on the graph are computed. Another point deals with the way how kernels on paths are computed and the last point is related to the way how kernels on sets of paths are evaluated.

For finding the shortest paths between all pairs of nodes, in all our experiments, we have used the Dijkstra's algorithm which scales of the order of $\mathcal{O}(N^3)$. Of course, depending on the problem at hand and on the graph properties, one can use any other algorithm which may be faster.

Computation of the kernel on paths, in our case, is simply based on the processing of the minor kernels for all nodes and edges which compose paths (see equation 1). Figure 1 shows a plot of the computational time needed on a PC of a Pentium 4D with 1GB of RAM for computing Gram matrix of paths. The code for producing this result has been written in Matlab. We can see for instance that it took about 220 seconds for computing 4 millions path inner products.

If we consider that the number of paths in each bag of paths is M then the computational cost of the mean kernel (equation 3) or the matching kernels (equations 4 and 5) is of the order

of $\mathcal{O}(M^2)$, which gives a worst case complexity of $\mathcal{O}(N^4)$. For the path level-set kernel, once the Gram matrix of each bag of paths is computed, a one-class SVM has to be run. It has been shown [34] that complexity of SVM algorithms empirically scales in $\mathcal{O}(M^2)$. In the right part of Figure 1, an example of averaged computational cost of one-class SVM using set of paths Gram matrices is depicted. Remark that this example cannot be generalized to other problems since the one-class SVM complexity depends on the problem at hand, however it gives some flavor on the computational burden. We can note that, considering a given number of paths in a bag, typically, the one-class SVM contributes far less than the Gram matrix on the overall computational burden. However, better designed SVM algorithm such as SimpleSVM [34] can be used to avoid the computation of the full Gram matrix.

As we have noted, the computational complexity of the kernel depends on the number of paths considered. Typically, if all paths in a graph are considered, the bag is composed of N^2 paths (remember that we have supposed that all the graphs are connected). However by limiting the path lengths, one can reduce considerably the size of the bag and thus can also reduce the computational complexity of the kernel evaluation. Figure 2 shows the influence of the path length on the number of paths to be included in the bag. These plots have been obtained on simulated graphs with 30 and 100 nodes and with randomly attributed edges. The parameter that has been fixed is the percentage of connectivity of a given random graph. We can see on this figure that, for both number of nodes and for a connectivity rate of 30%, all paths have a length lower than 3. This means that when using random walk path of possibly infinite length, the graph representation can rapidly be redundant and may be harmful for classification performance as noted by Mah et al. [20]. This is then a rationale for using bag of paths with controlled path length.

We have shown in this section that several graph kernels can be considered as kernel on sets of paths and thus we have proposed to extend this family of graph kernels by proposing other ways of building the set of paths and other ways for computing the similarity measure between sets of path.

IV. Application to Shock Graphs Mining

In this section, we expose the methodology we used for representing shapes into graphs. Then these graphs can be used for solving a pattern recognition problem on shape as we show in the

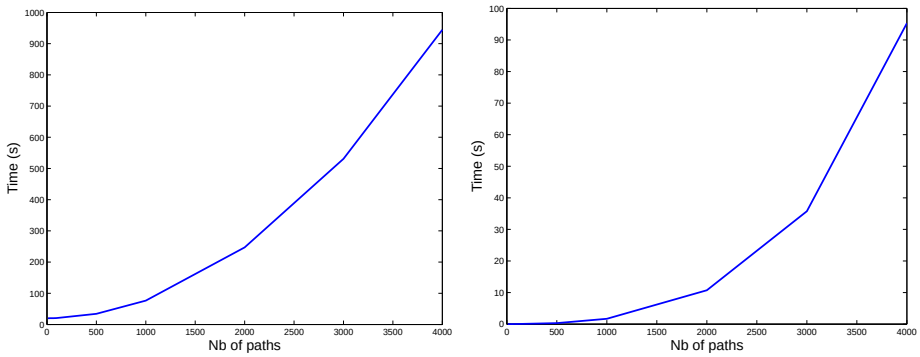


Fig. 1. Examples of computational complexity. (left) Time needed for building the Gram matrix of a set of paths with respects to the number of paths. (right) Time needed for computing the one-class SVM of a set of paths with respects to the number of paths.

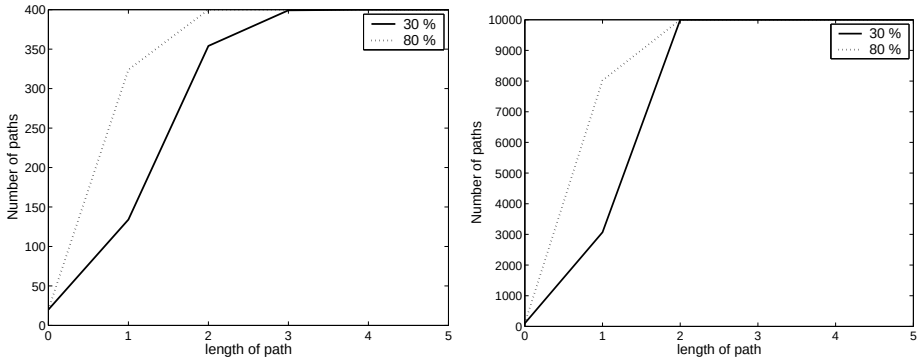


Fig. 2. Number of paths of a given length simulation in a random graph. (left) Graph order is 30. (right) Graph order is 100. The two curves in each plot depends on the percentage of graph connectivity.

numerical experiment section.

The dataset we used for illustrating our methodology are shapes extracted from the Rudgers tools datasets and some biological shapes used by Siddiqi et al [31]. These shapes have been extensively used in Demirci et al. and Sebastian et al. [6], [28]. Figure 3 presents the shapes than have been used.

In this paper, we focus essentially on similarity measures of shock graphs by using kernels. Hence, the problem of representing shapes using medial axis has been addressed using a classical approach [8]. Figure 4 presents some examples of shapes and their associated skeleton graphs we obtained with that algorithm.

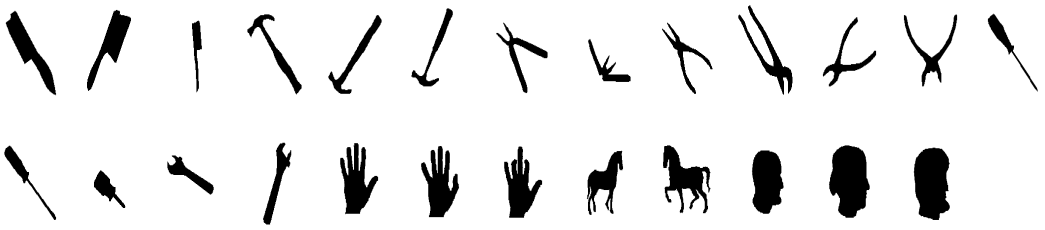


Fig. 3. Examples of shapes that have been used throughout the paper : 17 tools and 8 biological shapes. The top row shapes are numbered from left to rigth, 1 to 13 whereas the bottom row from 14 to 25.

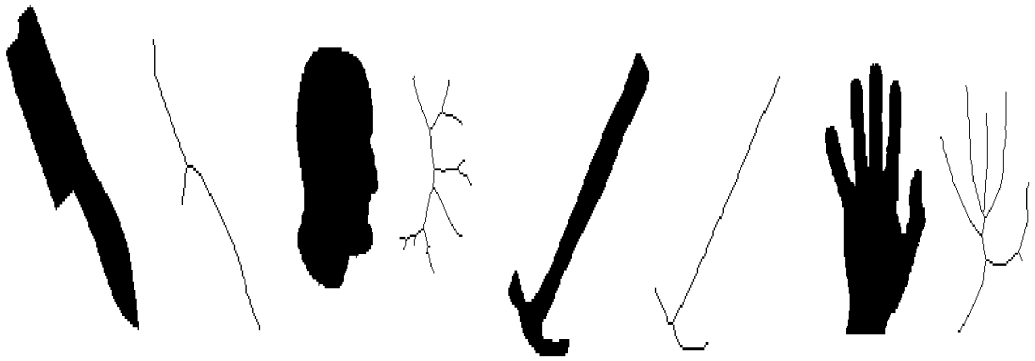


Fig. 4. Examples of shapes and associated skeleton graphs.

A. *From skeleton to graph*

Once the shape skeleton is obtained, we can now build the associated graph. The method we are using here is similar to the one proposed by Di Ruberto [24]. The principle is to classify each pixel of the skeleton as an edge point, end point or a junction point (this family of points will be defined in the following). Once this part is done, the next step is to figure out which sets of edge points correspond to a path between two end or junction points.

Hence, the first step we perform for transforming the skeleton into a graph is to compare the 3×3 neighborhood of each skeleton pixel to a set of masks, in order to determine whether this pixel is an end point, an edge point or a junction point. The difference between these three families of points is that a junction point has at least three pixels in its neighborhood corresponding to a finite set of configurations, an end point has a single neighbor or two adjacent neighbors while an edge point has two-non adjacent neighbors. Figure 5 gives examples of

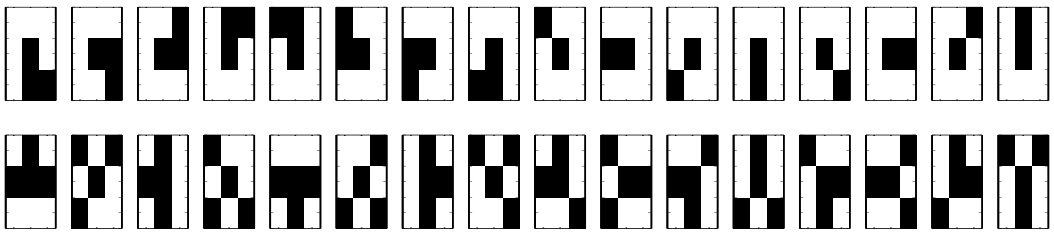


Fig. 5. Examples of pixels that are end points (top row) or junction points (bottow row).

configurations of end and junction points. This classification task can be efficiently performed through a single sweep of all skeleton pixels and a neighborhood comparisons.

Once this is done, the second step consists in determining which set of edge points corresponds to an edge between two end or junction points. This part of the task is simply performed in an iterative way by starting from an end point, following the skeleton from that point until reaching a junction point or an end point. Then all the edge points that have been crossed during the path following form an edge between the two end/junction points. All the points involved in the path are then removed from the skeleton, and we continue this procedure until no end point is left.

After this step, the graph can be built. Let G be the graph composed of vertices and edges $G = (V, E)$. The set of vertices is built from the set of end points and junction points. An edge exists between two vertices if there exists a set of edge points that connects the two associated end/junction points. Two examples of shapes and their corresponding skeletons and graphs are depicted in Figure 6.

B. Labelling the graph

A skeleton provides some structural insights on a shape. Hence, the graph obtained from this skeleton can also be informative of the shape structure if it is provided with some shape characteristics.

This task of labeling the graph is probably the most important one for our problem of classifying shapes by means of graphs associated to shape skeletons. In fact, according to the expression of the graph kernels we proposed (see equations 1, 3, 4 and 5), the similarity measure depends heavily on the labels associated to vertices and edges of the graph. Hence, while we provide here some attributes that can be discriminative, these attributes are highly problem-

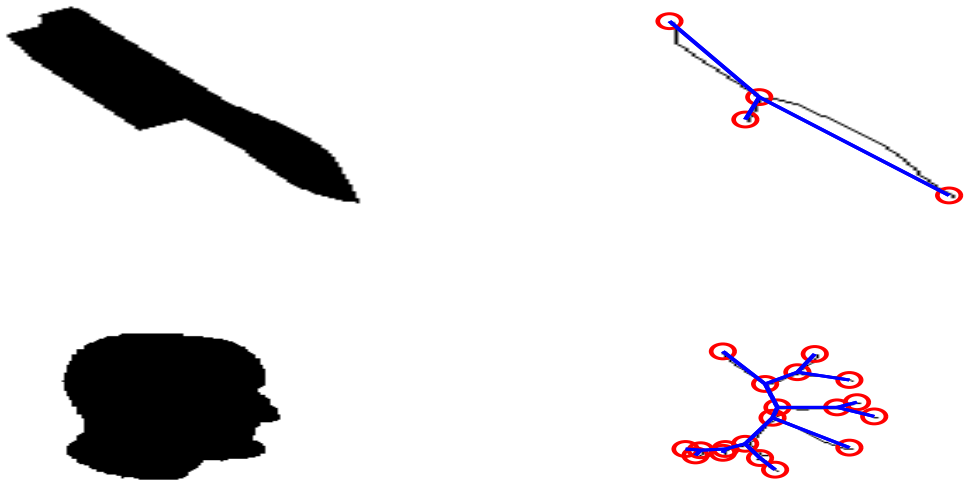


Fig. 6. Examples of shapes (left) and their corresponding skeleton and graphs (right). The circled points denotes end and junctions points whereas the straigh line connects two adjacent vertices.

dependent and should be chosen carefully or should be selected automatically according to the performance objective at hand *e.g* classification error. However, this poses the problem of attributes/features selection in vertices and edges attributes of graphs. This is a difficult problem that we have postponed for future researches.

In an aim of invariance properties, all the features that we have extracted from the shape and that we have associated to a vertex or an edge have been normalized according to some general characteristics of the shape. These general characteristics, which are illustrated in Figure 7 are :

- the coordinates $\{C_X, C_Y\}$ of the center of mass,
- the length (in pixels) L_M of the major axis of the ellipse EL that has the same second-moments as the shape,
- the length (in pixels) L_m of the minor axis of ellipse EL,
- the angle θ_0 between the x-axis and the major axis of ellipse EL.

Then for each vertex of the graph, some possible features are :

- the normalized distance between the vertex and the centroid :

$$\rho = \frac{\sqrt{(V_X - C_X)^2 - (V_Y - C_Y)^2}}{L_M}$$

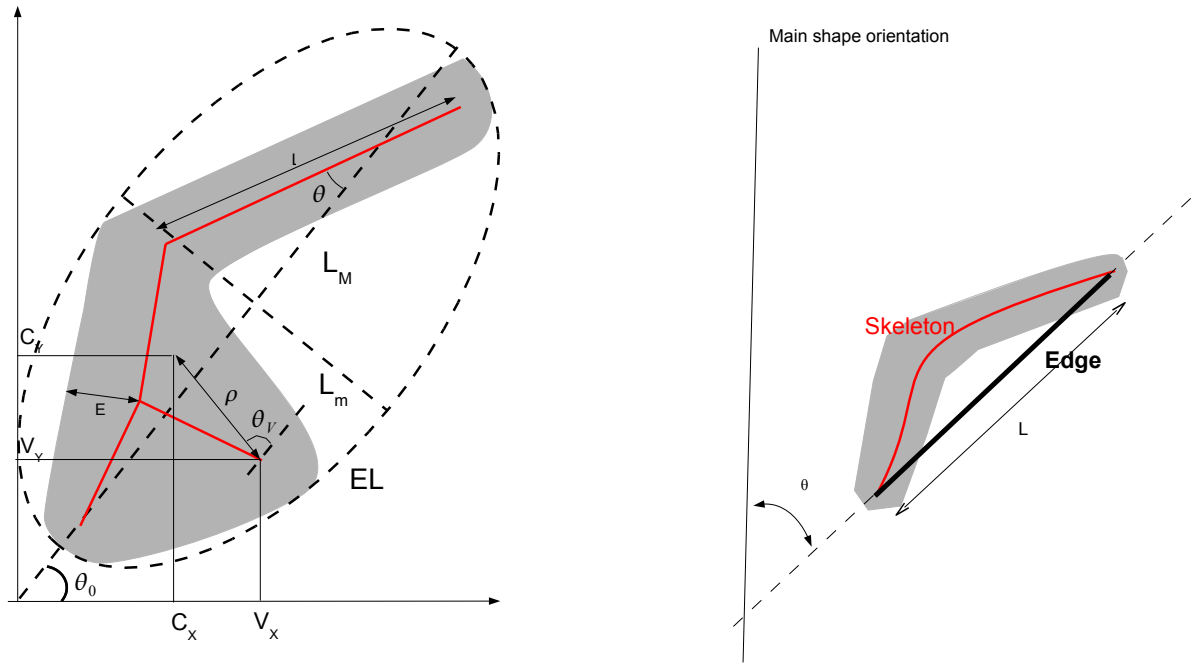


Fig. 7. Illustration of different general characteristics of a shape and some features associated to nodes and edges. The shape is depicted in gray. EL denotes the ellipse that has the same second-moments as the shape. The left part shows some general shape characteristics and the full skeleton associated to a shape. We have also plotted the center of mass and some features associated to nodes. The right part of the figure illustrate the computation of some edge features which are based on the edge skeleton curve and the edge associated to the graph (the straight line). The edge skeleton curve is the branch that is used for computing curvature and strength features.

- the angle θ_V between the centroid-vertex axis and the major axis of the ellipse
- The normalized width of the shape at the vertex position :

$$E = \frac{S(V_X, V_Y)}{L_m}$$

where $S(x, y)$ is the size of the largest structural element that can be applied to the shape at location (x, y).

- the distance function D:

$$D = \frac{Dist(V_X, V_Y)}{D_{max}}$$

where $Dist(x, y)$ gives the distance between the pixel (x, y) and the nearest pixel which does not belong to the shape. D_{max} is the maximal value of $Dist(x, y)$ for all possible x

and y.

For the edge, we have processed the following features :

- the normalized length L of the edge
- the relative orientation θ of the edge compared to the global orientation of the shape.

Similarly to Di Ruberto, some more sophisticated features can be used for characterizing a skeleton edge. For instance, we can use the following features illustrated in Figure 7 :

- strength : this feature is defined as the ratio between the number of pixels defining the edge skeleton and the edge length L.
- variance of edge skeleton curvature : if an edge is parametrized according to the arc-length t, we can expressed this edge as :

  $$c(t) = \{x(t), y(t)\} \qquad t \in [0, 1]$$

  and the instantaneous radius curvature of this curve is :

  $$r(t) = \frac{(x'^2 + y'^2)^{3/2}}{x'y'' - x''y'}$$

  with $x' = dx/dt$ and $y' = dy/dt$. Finally, we define the variance of edge's curvature as the variance of the curvature function $\frac{1}{r(t)}$.
- variance of distance function : this feature is defined as the variance of the distances of each pixel defining the edge skeleton to the border of the mask.

These last features enhance the information brought by edges by incorporating higher level informations. The *strengh* and *variance of curvature* features tend in some sense to encode the complexity of the edge. In fact, these features are larger for a complicated edge with strong curvature than for a straight line edge. Furthermore, the *variance of distance function* feature aims at encoding some informations on the object structure along a given edge skeleton. Indeed, this feature measures the variation of the object width.

At this point, we are able to transform a shape to a graph. The next step is then to investigate the usefulness of graph kernels for mining such shape representations.

V. NUMERICAL EXPERIMENTS

In this section, our objective is to provide a proof of concept of the usefulness of our graph kernel and the methodology we propose for shapes mining. Thus, we present here the different

numerical experiments and the analyzes we carried out when addressing a classical problem of shape retrieval and a problem of shape clustering.

A. Shape retrieval : comparison with others approaches

As a first experiment, we have tested our graph kernel approach on a problem of shape retrieval in a database and compared this approach to state-of-the art algorithms for graph matching. This comparison has been performed on the Rudgers tool database. This shape database contains 25 objects separated in 8 different classes five of which can be categorized as "tool" whereas the three other are biological shapes. These shapes are depicted on figure 3.

The shape retrieval problem is the following : each of the 25 objects is used as a shape query and for each query, we rank the 24 other shapes by decreasing similarity to the query shape (the similarity being defined according to the distance induced by the graph kernel). For an ideal similarity measure, supposing that the query shape belongs to a class with n elements, the $n-1$ first ranked shapes should belong to the same class of the query.

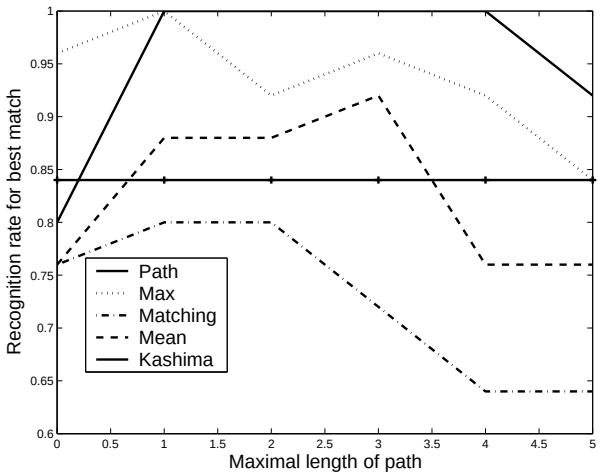


Fig. 8. Query result with the Rudger tools dataset. Performance on the best match retrieved shape for different graph kernels according to the length of path considered for the kernel computation. (solid) Path level set kernel. (dotted) max matching kernel (dash-dotted) matching kernel. (dashed) mean kernel. (constant solid) Kashima's kernel which considers all path length at once.

For our graph kernel, we have used ρ and E as node labels, and only the normalized length as edge label. Several parameters have to be fixed for our similarity measure which are the

minor kernel for edges and vertices. In all our experiments, we have used a gaussian kernel $k(x,y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2})$ for all these kernels. Hence we have to fixed the width σ of each kernel. In this paper, we have not addressed the problem of model selection although we have analyzed the influence of the gaussian kernel widths on the shape retrieval performance. Thus, for this comparison, we have arbitrarily fixed the gaussian kernel widths σ to 0.1. Since all the features have been normalized and belongs to the interval $[0,1]$, we believe that this *ad-hoc* parametrization is a good one. The gaussian hyperparameter in the matching kernel has also been set to 0.1.

When considering the best ranked shapes, Pelillo et al. [23], Sebastian et al. [28] and Demirci et al. [6] respectively reported 3, 1 and 1 mismatching retrieved shapes, which corresponds to 88%, 96% and 96% recognition rate. Figure 8 reports our results for different graph kernels and for different lengths of paths used for processing each kernel. When path length is equal 0, this means that kernel values have been computed considering only node similarities. In that case, we can note the max matching kernel performs very well since it reaches a recognition performance of 96% whereas other kernels give performances lower than 90%. Note that the Kashima's kernel gives a constant performance since its computation does not depend on a fixed path length. As the considered path length increases, we can see that all the kernels have a similar behavior, that is, the recognition rate increases then decreases when a given path length is reached. Hence, we can conclude that the path length plays an important role since it can bring discriminant information on the shape. This can be easily understood since increasing path length increases the structural information brought by the bag of paths through the major role played by edge labels. However, when path lengths become large performance tends to decrease, we believe that this behavior is due to redundancy added in the similarity measure by longer paths. For this problem, it seems that a path length of 2 or 3 seems to be a good compromise.

We can also note that, although the labels used for nodes and edges are rather simple, the path level-set based kernel (equation 6) and the max matching kernel (equation 4) are able to retrieve a correct shape on all the queries. For this experiment, the path level-set kernel seems to be robust to the path length since it provides perfect recognition rate for a path length from 1 to 4. Figure 8 clearly shows that the matching kernel (equation 5) and the mean kernel (equation 3) performs poorly compared to the max matching kernel that they approximated. Although, this lack of performance may be due to bad kernel parameters, this performance difference illustrates

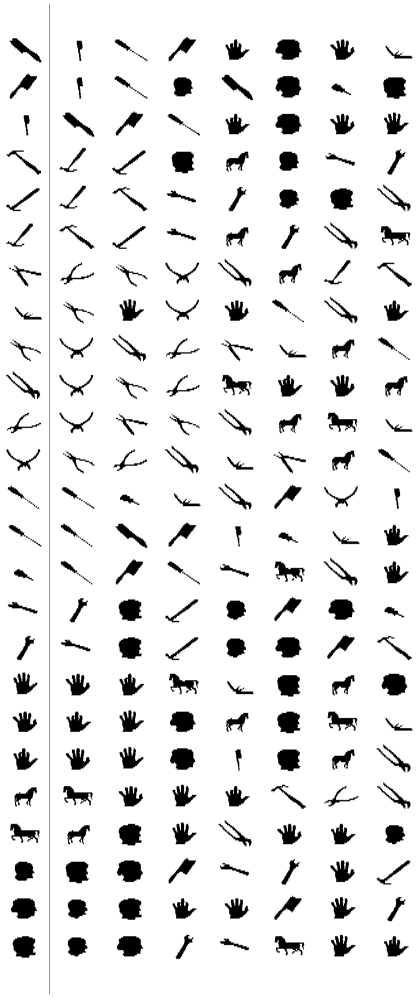


Fig. 9. Query result with the Rudger tools dataset. Left column images represent the query and then we have from left to right the most similar objects according to a path level-set kernel. The maximal length of paths used for computing the kernel is equal to 2.

the price that has to be paid for having a positive definite approximation of a kernel.

Figure 9 gives an example of shape retrieval for the path level-set kernel for a maximal path length of 2. We can see in this figure that the best matches are all correct but several incorrect matches have been obtained for the second best matches. These errors are essentially due to the incapacity of the similarity measure of making the difference between the class “brush”

and "screwdriver". A rationale for this may be that labels used for nodes and edges are not discriminating enough for these two classes.

The first conclusion that can be drawn from this experiment is that a graph kernel approach is very competitive compared to state of the art approaches for graph matching problems.

B. Shape retrieval : analyzing different parameters

Next reported experiments aim at empirically analyzing the ability of our approach to address the shape retrieval problem, in different contexts.

1) Impact of kernel parameters: For the previous experiment, we have kept kernel parameters fixed while the length of paths was varying. In this next experiment, we have investigated the influence of these kernel parameters on the retrieval performance. The maximal path length considered has been fixed to 2. Note that for the matching kernel, the gaussian width has been kept at 0.1.

Figure 10 presents the variation of the best match performance for the different kernels with respect to the widths of the node and edge gaussian kernels. As expected, for all kernels, these parameters play a crucial role on the similarity measure of the graph and hence on the matching performance.

We can also note that the different kernels have different behavior with respects to these parameters. While the path level-set, mean and Kashima's kernel performs best for small values of σ, the matching kernel seems to work better for large values of σ. The max matching kernel is the kernel that is less sensitive to the parameters. In fact, for a large range of parameters, it is able to provide a very good similarity measure. This is an advantage of directly matching paths at the expense of the kernel being not positive definite. The approximate matching kernel we propose behaves in a similar way with respect to parameters although the performance is lower.

This experience clearly shows that some efforts have to be spent for selecting the best parameters. This model selection problem can be addressed by using classical techniques like cross-validation. However, efficient algorithms that address the problem of parameters selection in graph kernels are our future objectives.

2) Impact of nodes and edge labels: This next experiment aims at illustrating that the edges and nodes labels have to be chosen carefully and have to be adapted to the problem at hand.

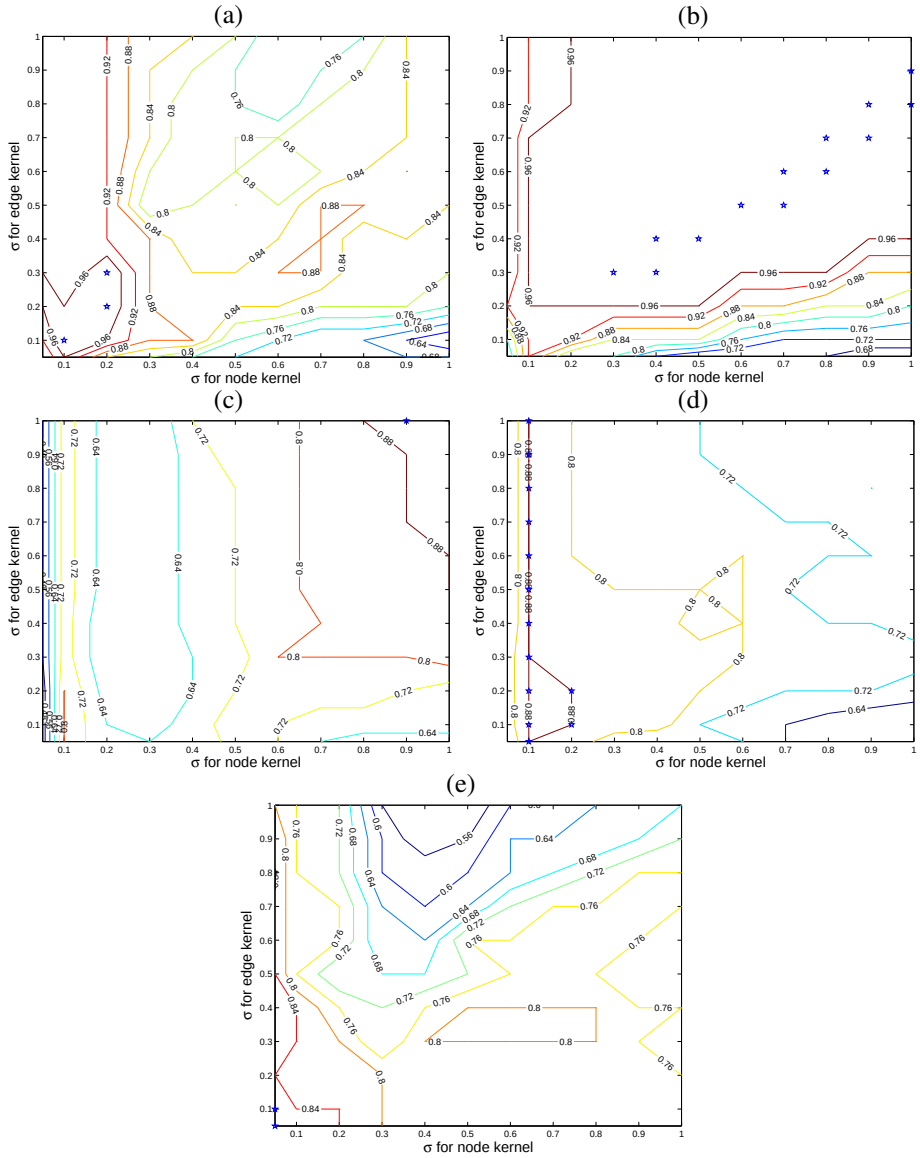


Fig. 10. Illustration of the impact of kernel parameters on the shape retrieval problem for the different kernels. a) Path level-set kernel. b) max-matching kernel. c) matching kernel. d) mean kernel. e) Kashima's kernel.

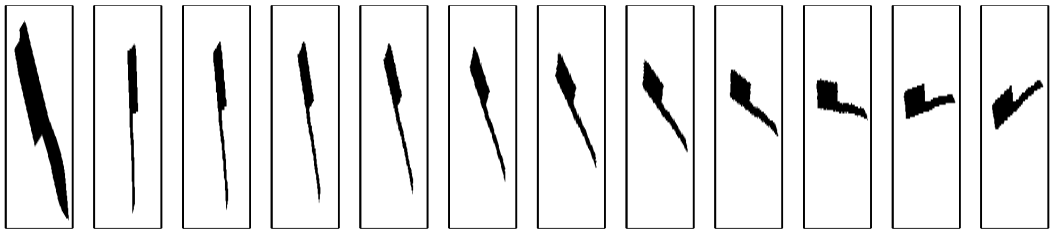
For instance, we have suggested that orientations of edges or angular positions of nodes relative to the centroid and the major axis can be used as possible labels. These two labels would be able to provide some rotation invariant characteristics of the shape. However, some invariance properties may not be desired in some problems.

For illustration, we have measured the similarity between two brush shapes while one of the shape has been kept fixed and the other is rotating. The similarity measure is defined as

the squared distance induced by the path level-set kernel with the same parameters as in the comparison experiment (section V-A) and it has been computed using different edge labels.

Figure 11 shows the shapes that has been used. Two different edge labels have been used for computing the graph kernel. For the set 1, the edge label is "length" whereas for the set 2, we have used "length" and "orientation", which corresponds to the angle between the edge and the general orientation of the shape. The results given in the table illustrate two points. The first interesting point is that, in this example, adding the edge orientation as a label reduces the similarity measure between the two brush shapes. Again, this reinforces the point that labels have to be selected carefully. In this case, the decrease in similarity can be explained by the vertical symmetry of the two brushes. The second interesting point is that the two label sets provided to the graph are somewhat rotation invariant. In fact, the similarity measure between the two shapes is stable with respects to the angle of rotations.

On the overall, this brings evidence that some prior knowledge on the problem at hand can be taken into account through the kernels (in our case by providing some invariant information on the labels or by eliminating any variant information).



	Angle of rotations in degrees										
Label Sets	0	10	20	30	40	50	60	70	80	90	100
1	1.00	0.93	1.40	0.77	1.24	0.93	0.77	1.40	1.47	1.70	1.31
2	1.61	1.63	1.67	1.29	1.37	1.57	1.57	1.66	1.67	1.73	1.67

Fig. 11. Examples of the impact of node or edge labels. The table gives the squared distance between the brush shape at the most left of the figure and the other shapes which are rotations of the second left shape. Captions of the figure denotes the angle of rotations. The edge label used for the set 1 is "length" whereas for the set 2 "length" and "orientation" has been used.

3) Impact of occlusions on the kernel: This other experiment aims at analyzing how does the overall approach for mining shapes behaves in presence of occlusions.

Occluded shapes have been generated as following. Given a shape and its height, a $x\%$ of occlusion in the shape has been obtained by canceling the $x\%$ part of this shape starting from the top of the shape. This results in an occluded shape which height is $x\%$ smaller than the original. Figure 12 gives examples of occluded shapes. The occlusion of the "wrench" shape has been stopped at 35% since larger occlusions generates disconnected shapes.

In order to evaluate how our approach is sensitive to occlusions, we have measured the similarity between a shape and its occlusions according to equation 6 and using the same parameters as in the first experiment (section V-A).

Figure 13 depicts the behavior of this measure according to the percent of occlusion for the two shapes in Figure 12. As we can see, these two shapes illustrates different situations. The "hand" shape is more robust to occlusions than the "wrench". Indeed, the similarity measure of "hand" is still at about 0.7 for 25% of occlusion and is about 0.4 at 50%. On the contrary, similarity measure of wrench decreases rapidly and it is already around 0.2 with 20% of occlusion.

This difference of robustness to occlusion is simply due to the impact of occlusions on the shape structure. On the first hand, for the "hand" shape, the occlusion, up to a certain point, only reduces the length of some edges and introduces extra-nodes at the end of fingers without modifying the overall graph structure. On the other hand, for the "wrench" structure, as soon as the top part of the wrench is occluded, some nodes and edges disappear, the structure of the graph is considerably modified, thus the decrease in similarity. What we understand from this experiment is that occlusions have two effects on the similarity measure : one due to modifications of the shape structure and the other due to the modifications of the structure labels. Experiments also show that modifications of shape structure have bigger influence on the similarity measure when the number of nodes and edges is smaller (*e.g* for the wrench).

We have carried out another experiment in order to investigate the impact of occlusions. Starting from the best shape retrieval of the first experiment (section V-A), we have done the following. For a given query shape, we take the best matching shape and occlude this shape following the above described procedure. Then we have measured the similarity between the query shape and its occluded best match shape. Next, we have evaluated which percentage of occlusion is necessary for that occluded shape, to have a similarity score lower than the score of the most similar shape not belonging to the same class as the query. The table of percentage of occlusions for each shape is given in Table I. On average over all query shapes, the mean

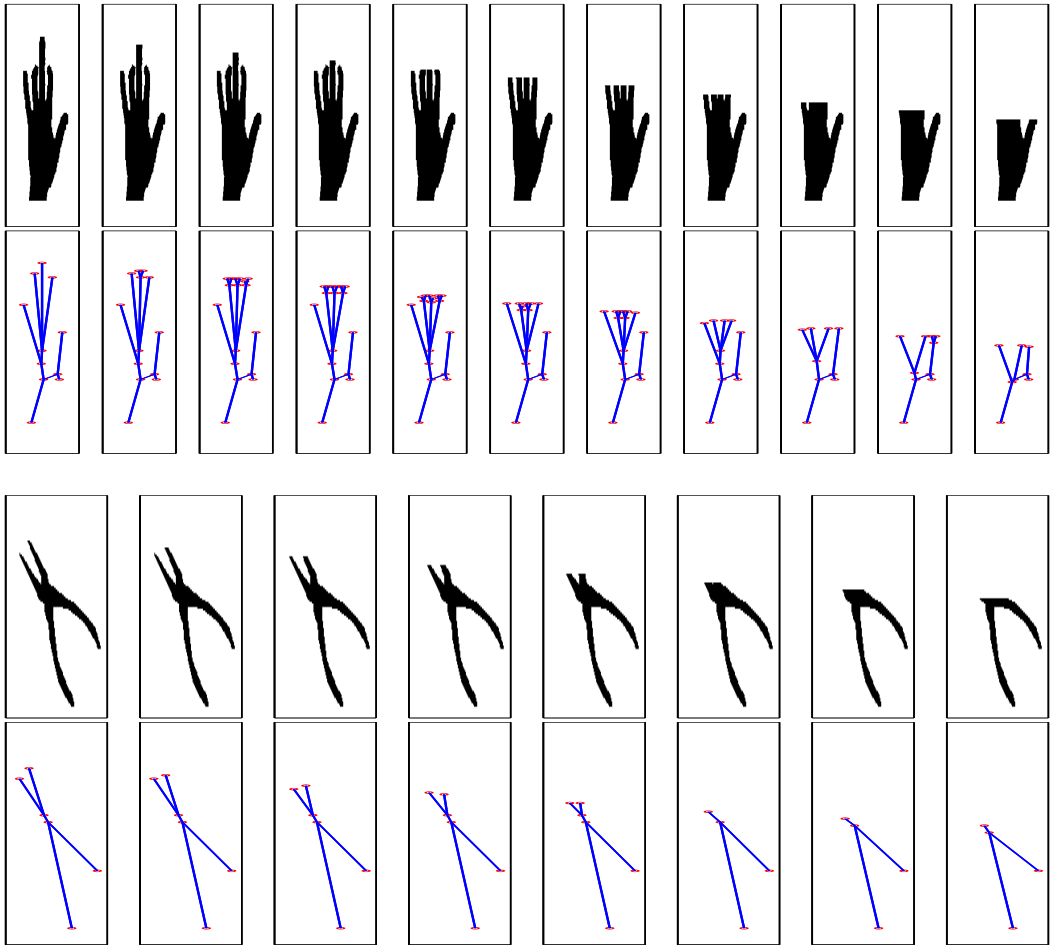


Fig. 12. Examples of 2 shapes that with occlusions of 0% to 50% for the "hand" shape and 0% to 35% for the "wrench" shape. The step size of occlusion if 5%. Utmost left shape is the original.

occlusions needed in about 20%, which means that our approach for measuring graph similarity is rather robust to occlusions, at least for this dataset. This robustness can be explained by the fact that several best matching shapes have a shape structure that does not vary a lot (*e.g* hammer, some wrenchs, ...) during the occlusions.

Hence, we can conclude that under some occlusions that does not modify considerably the structure of the shape graph, our similarity measure based on kernels is rather robust.

C. Application to clustering and low dimensional embedding

The following experiments show some other possible applications for the proposed graph kernels for object and shape mining. In this part, we have used as a database the ETH Zurich

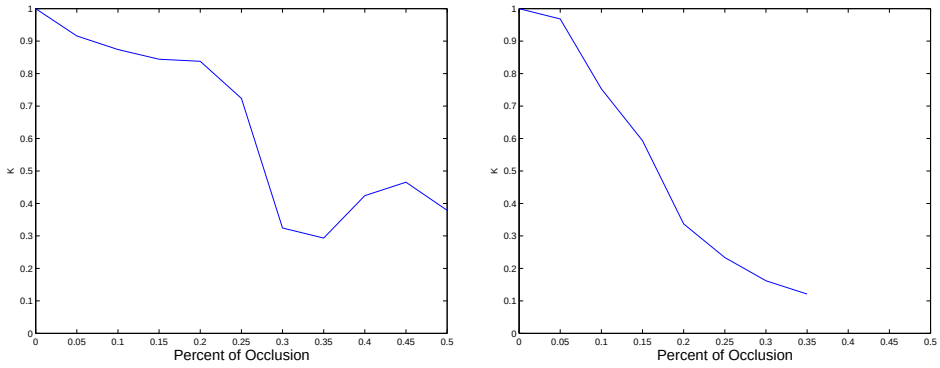


Fig. 13. Variation of similarity between a shape and its occlusion with respect to the percentage of occlusion. (left) "hand" shape. (right) "wrench" shape.

	Shape Query											
Shape number	1	2	3	4	5	6	7	8	9	10	11	12
% Occlusions	5	15	30	35	35	25	50	0	50	40	30	35

	Shape Query												
Shape number	13	14	15	16	17	18	19	20	21	22	23	24	25
% Occlusions	10	5	5	15	10	45	30	10	35	30	10	20	20

TABLE I

SHAPE RETRIEVAL AND OCCLUSIONS. THIS TABLE GIVES THE PERCENTAGE OF OCCLUSION NEEDED FOR THE BEST MATCHING SHAPE OF THE QUERY TO HAVE SIMILARITY SCORE LOWER THAN THE ONE OF THE BEST SHAPE NOT BELONGING TO THE CLASS OF QUERY SHAPE.

ETH-80 image library [17]. This database is composed of 8 classes with 10 objects per class and 41 views for each object. In our illustration, we have used only a subset of all examples (the first two objects for each class). Examples of objects are given in Figure 14. The size of each image is 256×256 pixels.

The first experiment we have carried out is a clustering experiment which objectives are three-fold. First, we want to verify that our graph kernel and the similarity measure we derive is able to cluster appropriately this difficult dataset. Then, we want to analyze the contribution of a graph structure compared to a bag of nodes approach and finally, we want to prove that high-level

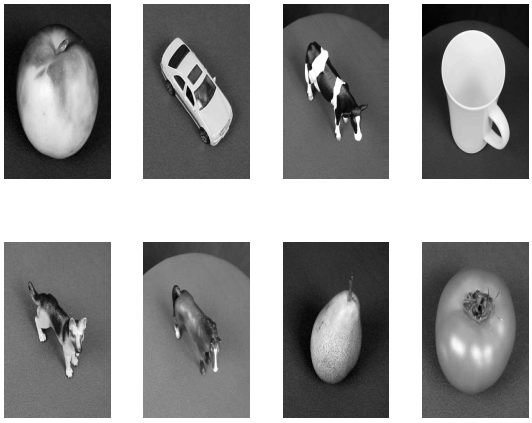


Fig. 14. Views of objects from the ETH-80 image dataset.

information, such as histogram, can be useful as nodes or edges labels.

For each example, the graph representation has been obtained as described in Section IV by means of the binary mask that is provided for each image example. For this experiment, the node and edge labels are different from those used in the shape retrieval problem.

Edge labels are composed of the following features : *length*, *strength*, *variance of curvature* and *variance of distance function*. The characteristics we used are more elaborated than those used for the Rudger tools datasets because for this dataset, shapes are more difficult to classify.

Nodes labels are composed of the normalized distance ρ of each node to the centroid and the local histogram of oriented gradients. This histogram is processed as described by Dalal et al. [5] in a neighborhood of 32×32 around each node. For this window, several 8-bin histograms computed from 4×4 pixels denoted as cell have been obtained by weighting orientation with the norm of the gradient. As stated by Dalal et al., a normalization step is necessary in order to reduce variability effects. For normalization, we have used block size of 2×2 cells with an overlap of 1 cell and a L_2 normalization factor. More details on this histogram of gradient procedure are available in the original paper [5]. According to this procedure, we obtained a node label of dimension 2049.

A single graph kernel has been tested for this experiment which is the path level-set kernel. The minor kernel for nodes and edge is still the gaussian kernel with respectively $\sigma = 2$ and $\sigma = 1$. For the graph kernel, paths of length 0 and 1 have been used. This means that for the first case, only the node labels are taken into account for measuring similarity between objects.

Whereas for path length of size 1, we consider nodes, neighboring nodes and edges between these nodes for defining similarities.

The clustering algorithm is the spectral clustering approach proposed by Ng et al [22]. The k-means algorithm is parametrized so that 8 classes are defined.

Examples of clustering results are given in Figure 15 and Table II. The figure shows how the different classes of objects have been clustered when using only nodes information and nodes and edges information. The classes have been ordered and the most frequent labels assigned to each class is highlighted. On the left part, when using only nodes informations, we can see that the majority labels assigned to classes "car", "cup", and "tomato" are respectively "8","3" and "7". Classes "apple" and "pear" have been essentially assigned to class "6" and the animals have been clustered together and majoritarily assigned to label "5". Note that the label "1" has been associated to particular views of the animals, label "4" associated to views of "cup" while the cluster label "2" seems to be a cluster with views of different objects. On the overall, we can see that the clusters provided by our graph kernel with only nodes information and this spectral clustering algorithm are highly mixed. It is only able to cluster high-level categories that can be considered as "fruits" or "animals". On the right part on the figure, we see that as soon as edge informations are used in the graph kernel then the clusters provided by the algorithm are better. From the 8 clusters, only a single one is a spurious cluster (label "4") composed of particular views of "cup". "cow" and "horse" are still mixed in a single cluster but separating these classes is difficult even in a supervised setting [17]. We can also remark that several classes, such as "pear", "tomato" or "cup" are nicely clustered into separate classes. The confusion matrix given in Table II confirms these findings.

In addition to this task of clustering, we have also applied our graph kernel for embedding these ETH images into a two-dimensional space. Typically, this approach is useful for a purpose of visualization or low-dimensional clustering. As a projection algorithm, we have used the Isomap algorithm [33] based on the metric induced by the kernel which has produced the above clustering results. The two-dimensional linear embedding obtained is depicted in Figure 16. Observe that the embedding is able to capture some information about the object shapes and views. For instance, top-right corner images are essentially composed of "cup" which views are mostly rectangular. While, still considering the right side of the embedding, when moving towards the bottom-right, objects become more and more round : we have some apples and

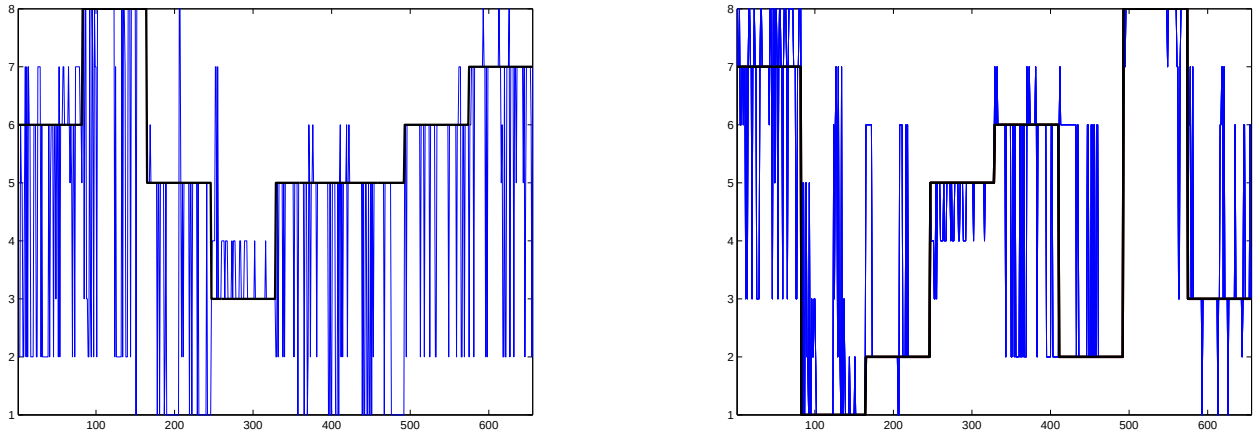


Fig. 15. Example of clustering results when using graph kernel with only node information (left) and path of length 1 (right). Each class contains 2 objects of 41 views each. The classes are ordered in the following way : apple, car, cow, cup, dog, horse, pear, tomato. In each figure, the thinner plot gives the class assigned by the clustering algorithm to each example whereas the wider plot gives the majority class assignment for each class. For instance, in the left figure, for the class "apple" the most frequent class label is 6 while some apples have been assigned label 7 which is the most frequent in the class "tomato".

tomato and other views (which are rounder) of cups. Bottom-left images are essentially made complex animal shapes.

These first results on clustering and low-dimensional embedding are encouraging and show the potential of using graph kernels for mining object shapes. We will investigate the full potential of the proposed approach for specific images or shapes retrieval problems.

VI. Discussions and Conclusions

In this paper, we have presented a novel approach for mining shape shock graphs by using kernels. At first, we have analyzed a typical way for building kernels for graphs and then proposed different ways for extending such class of kernels. These kernels are essentially based on the comparison of bag-of-paths obtained by traversing graphs.

When applying these kernels to shock graphs retrieval or recognition problem, the essential work is to provide some meaningful and discriminative information and labels to graph nodes and edges. The second part of the work described in this paper is then devoted to the efficiency of such kernels for shock graphs mining tasks. We have shown that using simple labels based on the shape geometry, we can achieve shape retrieval results (on the Rudger tools datasets) that are comparable to the current state-of-the art results. Other experiments have also shown

	predicted cluster							
	apple	car	cow/horse	cup	dog	pear	tomato	unknown
apple	37	0	0	1	4	27	13	0
car	3	55	11	4	3	0	6	0
cow	0	2	66	0	14	0	0	0
cup	0	0	0	56	0	0	3	23
dog	5	0	30	0	47	0	0	0
horse	1	0	51	0	30	0	0	0
pear	5	0	0	0	0	75	2	0
tomato	5	3	1	0	8	0	65	0

TABLE II

CONFUSION MATRIX AFTER CLUSTERING USING PATH LEVEL-SET KERNEL.

that using these graphs kernels can be useful for solving problems such as clustering and low-dimensional embedding. For these examples, we have used more complex labels (histogram) for nodes. Hence, using kernels also open the way to similarity measure for graph with high-level and structured labels on nodes or edges. Another interesting point highlighted by our numerical results is that, bringing structural information can greatly enhance similarity measures. Indeed, when considering only nodes in the computation of the graph kernel, the kernel is built by means of a bag of nodes with only local information. At the present time, this is a typical approach in the object recognition community when using bag of points of interest [21], [25]. However, our results clearly show that adding information regarding nodes topology produces better similarity measure.

Future works will focus on developing applications of this approach on specific problems such as shape retrieval, image categorization and retrieval for which nodes will consist of image regions. Another approach we plan to consider is to investigate the use of graph kernels for classifying object images based on sets of local descriptors and some structural information.

One important point that has been evoked throughout the paper is the problem of feature selection in nodes and edges. Indeed, the success of the approach is dependent of the node and edge labels and ideally, the classification or clustering algorithm should be able to select these features automatically posing then the problem of feature selection in graphs.

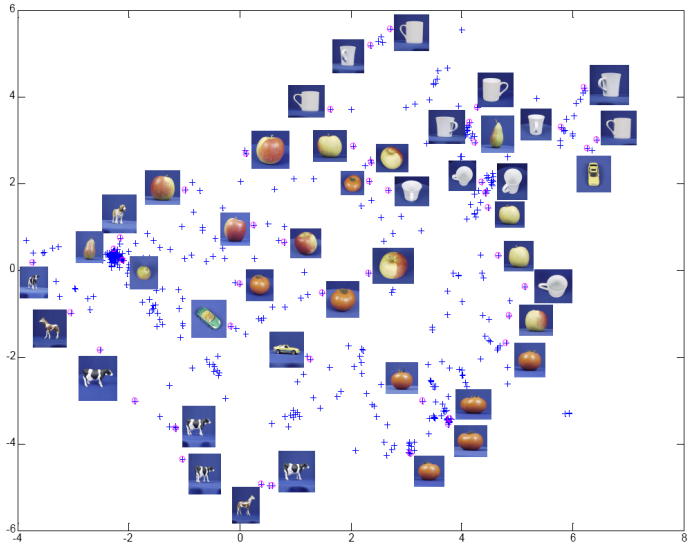


Fig. 16. Example of low-dimensional embedding of subset images of the ETH80 Dataset.

REFERENCES

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[2] S. Boughorbel, J.P Tarel, and N. Boujemaa. The gcs kernel for svm based image recognition. In *The International Conference on Artificial Neural Networks (ICANN'05)*, Poland, 2005.

[3] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19:255–259, 1998.

[4] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:pp 1–25, 1995.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[6] F. Demirci, A. Shokoufandeh, L. Bretzner, and S. Dickinson. Object recognition as many-to-many feature matching. *Internation Journal of Computer Vision*, 69(2):203–222, 2006.

[7] F. Desobry, M. Davy, and W.J. Fitzgerald. A class of kernels for sets of vectors. In *Proceedings of the 13th European Symposium on Artificial Neural Networks*, 2005.

[8] P. Dimitrov, C. Phillips, and K. Siddiqi. Robust and efficient skeletal graphs. In *Conference on Computer Vision and Pattern Recognition*, 2000.

[9] T. Gartner. Exponential and geometric kernels for graphs. In *NIPS 02 Workshop on Unreal Data : Principle of Modelling Nonvectorial data*, 2002.

[10] T. Gartner, J. Lloyd, and P. Flach. Kernels for structured data. In *Proceedings of Twelfth International Conference on Inductive Logic Programming*, 2002.

[11] S. Gold and A. Rangarajan. A graduated assignment algorithm for shape matching. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996.

[12] B. Haasdonk. Feature space interpretation of svms with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492, 2005.

[13] B. Haasdonk and C. Bahlmann. Learning with distance substitution kernels. In Springer, editor, *Pattern Recognition - Proc. of the 26th DAGM Symposium*, 2004.

[14] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieh International Conference on Machine Learning*, 2003.

[15] H. Kashima, K. Tsuda, and A. Inokuchi. Kernel for graphs. In B. Schlkopf, K. Tsuda, and J-P. Vert, editors, *Kernel Methods in Computational Biology*, pages 155–170. MIT Press, 2004.

[16] R. Kondor and T. Jebara. A kernel between sets of vectors. In *Proceedings of the 19th International Conference on Machine Learning*, 2003.

[17] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *nternational Conference on Computer Vision and Pattern Recognition*, 2003.

[18] P. Mahé, L. Ralaivola, V. Stoven, and J.-P. Vert. The pharmacophore kernel for virtual screening with support vector machines. *J. Chem. Inf. Model.*, to appear, 2006.

[19] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*, pages 552–559. ACM Press, 2004.

[20] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *J. Chem. Inf. Model.*, 45(4):939–51, 2005.

[21] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[22] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, 2002.

[23] M. Pelillo, K. Siddiqi, and S. W. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120, 1999.

[24] C. Di Ruberto. Recognition of shapes by attributed skeletal graphs. *Pattern Recognition*, 37(1):21–31, 2004.

[25] Cordelia Schmid, Gyuri Dorkó, Svetlana Lazebnik, Krystian Mikolajczyk, and Jean Ponce. Pattern recognition with local invariant features. In C.H. Chen and P.S.P Wang, editors, *Handbook of Pattern Recognition and Computer Vision*. World Scientific, third edition, 2005. to appear.

[26] B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.

[27] B. Scholkopf, A. Smola, and K-R Muller. Non linear component analysis as a kernel eigenvalue problem. *Neural Computation*, (10):1299–1319, 1998.

[28] T. Sebastian, P. Klein, and B. Kimia. Recognition of shapes by editing shock graphs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(5):550–571, 2001.

[29] D. Sharvit, J.Chan, H.Tek, and B.B.Kimia. Symmetry-based indexing of image databases. *Journal of Visual Communication and Image Representation*, 9(4):366–380, 1998.

[30] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambrigde University Press, 2004.

[31] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35:13–32, 1999.

[32] F. Suard, V. Guigue, A. Rakotomamonjy, and A. Bensrhair. Pedestrian detection using stereovision and graph kernels. In *IEEE Intelligent Vehicles Symposium*, 2005.

[33] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[34] S. V. N. Vishwanathan, A. J. Smola, and M. Murty. Simplesvm. In *International Conference on Machine Learning*, 2003.

[35] C. Wallraven, B.Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *Proceedings of International Conference on Computer Vision*, pages 257–264, 2003.

[36] L. Wolf and A. Shashua. Learning over sets using kernel principal angles. *Journal of Machine Learning Research*, 4:913–931, 2003.