Tree Covering within a Graph Kernel Framework for Shape Classification

François-Xavier Dupé* and Luc Brun

GREYC UMR CNRS 6072,
ENSICAEN-Université de Caen Basse-Normandie,
14050 Caen France
{francois-xavier.dupe,luc.brun}@greyc.ensicaen.fr

Abstract. Shape classification using graphs and skeletons usually involves edition processes in order to reduce the influence of structural noise. On the other hand, graph kernels provide a rich framework in which many classification algorithm may be applied on graphs. However, edit distances cannot be readily used within the kernel machine framework as they generally lead to indefinite kernels. In this paper, we propose a graph kernel based on bags of paths and edit operations which remains positive definite according to the bags. The robustness of this kernel is based on a selection of the paths according to their relevance in the graph. Several experiments prove the efficiency of this approach compared to alternative kernel.

Keywords: Shape, Skeleton, Graph Kernel, Kernel Machines.

1 Introduction

Shape matching is a challenging problem in computer vision with many applications: indexing, classification, clustering... Many frameworks have been developed based on different point of views. Two types of methods can be distinguished: *model-based* methods [1] where each shape is compared to models or prototypes and *shape-based* methods where shapes are compared one to one. These last methods, are often based on the skeletons [2,3].

Usually, skeletons are transformed into graphs, translating the shape matching problem into the more general graph matching problem. Different works have been developed in order to tackle this last problem. For example Siddiqi [4] extracts shock graphs from shapes and uses a greedy algorithm for the comparison task. Pelillo [5] transforms graphs into trees and models the problem as a maximal clique problem. Goh [3] splits graphs into linear parts using heuristics and then directly compares their features. These methods operate directly inside the graph space which lacks many common mathematical tools.

One way to avoid this issue is to map graphs into a richer space. This is the purpose of graph kernels whose development is growing with the great interest

* This work is performed in close collaboration with the laboratory Cycéron and is supported by the CNRS and the région Basse-Normandie.

P. Foggia, C. Sansone, and M. Vento (Eds.): ICIAP 2009, LNCS 5716, pp. 278–287, 2009.

in kernel machines for the last ten years. The most famous graph kernels are the random walk kernel [6], the marginalized graph kernel [6] and the geometric kernel [6]. As the skeletonization process is not continuous, two graphs representing similar shapes can show severe structural differences. Since the above graph kernels implicitly reduce graphs comparison into paths comparison, graph perturbations modify the paths and thus lead to an inaccurate comparison.

Neuhaus and Bunke have proposed several kernels [7,8] based on the graph edit distance in order to reduce the influence of graph perturbations. However the graph edit distance does not usually fulfills all the properties of a metric and the design of a positive definite kernel from such a distance requires some precaution. Our approach is slightly different. Indeed, instead of considering a direct edit distance between graphs, our kernel is based on a rewriting process applied independently on some paths of the two graphs. Such a rewriting scheme is introduced within a more general graph kernel framework based on bags of paths.

Suard [9] introduced a new graph kernel framework based on an explicit encoding of bags of paths. This framework offers 3 degrees of liberty for the design of a graph kernel: 1) construction scheme of the bag of paths of a graph, 2) definition of a kernel between bags of paths and 3) definition of a kernel between paths. The contribution of this paper lies in two of these three points: we propose 1) a new algorithm for the construction of the bag of path based on an edge-covering algorithm and 2) a new bag of paths kernel based on the mean kernel. This bag of path kernel is combined with the edition path kernel first introduced in [10] and latter extended in [11].

This paper is structured as follows: first we present our graph construction scheme from a skeleton (Section 2). Then, a construction algorithm for bag of paths based on a tree-covering algorithm is proposed and analysed from a shape point of view (Section 3). Following that, we propose a new bag of paths kernel based on the mean kernel (Section 4). Then, we rapidly present the edition path kernel [10,11] which is based on a hierarchical comparison of paths. Finally, the performances of the resulting graph kernel are measured through experiments in Section 6.

2 Skeleton-Based Graph

The skeleton of a shape is usually constructed from the medial axis which is defined as the centers of the circles of maximal radius [2]. Many graphs may be associated to the skeleton of a shape. Our graph construction scheme follows the approach proposed by Siddiqi [4]. It considers the enriched translation of the skeleton structure to a graph structure: the terminal points, the junction points and any point encoding an important change of slope of the radius function along the medial axis define the nodes of the graph. The edges of the graph correspond to skeleton's branches between two nodes. The slopes can be obtained using regression methods based on first order splines [11,12]. Finally, for the sake of simplicity, we consider the maximal spanning tree of such graphs. Note that, as skeletonization is an homotopic transform, a shape with no hole leads directly to a tree.

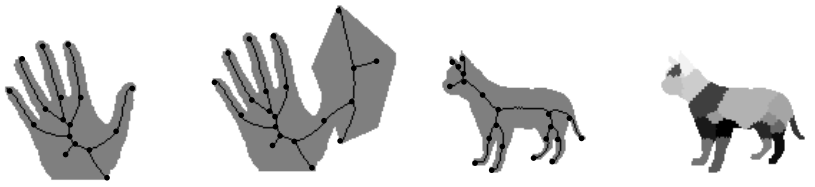


Fig. 1. Three examples of skeletons with their inflexion points

Fig. 1 shows three shapes with their inflexion points. The detection of these inflexion points may slightly vary according to the parameter of the regression method used to detect them. Such a variability of the graph should be compensated by the flexibility of our graph kernel.

The graph associated to a shape only provides information about its structural properties. Additional geometrical properties of the shape must be encoded using node and edge attributes in order to keep most of the shape properties. From a structural point of view, a node represents a particular point inside the shape skeleton and an edge a skeleton's branch. However, a branch also represents the set of points which are closer to the branch than any other branch. This set of points is defined as the *influence zone* of the branch and can be computed using a SKIZ transform [13] (Fig. 1d).

We have selected a set of attributes which provides a good approximation of the description of the shape. Torsello [14] proposes as edge attribute an approximation of the perimeter of the boundary which contributes to the formation of the edge, normalized by the approximated perimeter of the whole shape. This feature presents the double advantage of being additive and of encoding the relevance of edges inside graphs. Suard [15] proposes as node attribute the distance between the node position and the gravity center of the shape divided by the square of the shape area.

Two descriptors describing a branch of the skeleton and the evolution of the radius of the maximal circle along it are also considered. For each point $(x(t), y(t))$ of a branch, $t \in [0, 1]$, we consider the radius $R(t)$ of its maximal circle. In order to normalize the data, the radius is divided by the square root of the area of the influence zone of the branch. We also introduce $\alpha(t)$, the angle formed by the tangent vector at $(x(t), y(t))$ and the x-axis. Then the two considered descriptors are $(a_k)_{k\in\mathbb{N}}$ and $(b_k)_{k\in\mathbb{N}}$ the coefficients of two regression polynomials that fit respectively $R(t)$ and $\alpha(t)$ in the least square sense. If both polynomials are of sufficient orders, the skeleton can be reconstructed from the graph and so the shape. A more in depth analysis of this construction scheme of the graph may be found in [11].

Within Suard [9] framework, the comparison of 2 graphs requires to build their associated bags of paths. The construction of such bags remains problematic, if we want to keep the link between the graph and the represented shape. In the next section, we propose a method based on this link between the structure of the graph and its semantics.

3 Covering the Graph

The heuristic used to build a bag of paths from a graph constitutes a major step in the design of a graph kernel within the explicit bag of path framework. Paths extracted from a graph describe skeleton parts of the shape. So if a bag of paths contains all the paths of a graph (or all the paths up to a fixed length), it will describe several times the same part of the shape. This redundancy may be inaccurate since it artificially enforces the importance of some parts of the shape.

This last remark tends to prove that selecting paths for the bag opens a door to a more efficient comparison. One example of selection process is to keep only a fixed percentage of the paths amongst the weightier, however such a scheme induces a loss of some parts of the shape, since many edges may not be present inside such a bag of paths.

A proper solution would be to cut the shape into several parts and to consider only one path for each part. From a graph point of view, this algorithm is related to the tree covering [16] problem. More precisely, we would like to have the minimal sub-set of heaviest paths from a given set of paths which covers the set of edges of the graph. It is equivalent to the edge covering problem using a given set of paths with the constraints of minimal cardinality and maximum weight. Such a problem is NP-hard even for trees [16].

The exact algorithm proposed by Guo [16] is based on dynamic programming and computes the whole set of covering solutions from a given set of paths. This algorithm computes a vertex covering, but can be easily converted into an edge covering algorithm. We rapidly present the principle of this method: first, the authors root the tree using an arbitrary node and then apply an ascendant algorithm. At each step, a node is considered with all the paths which cover it, then for each combination of paths, the weight of the covering of the sub-tree rooted on the node is computed using information from children. The algorithm finishes at the root node and the result is given by taking only the solution of maximal weight (using a top-down approach). The final complexity of the algorithm is $\mathcal{O}(2^C C\,|V|)$ where $|V|$ is the number of vertices and C the maximal number of paths covering one node. The edge-covering version of this algorithm is built by 1) considering at each node the paths which cover the node and its parent and 2) adding a fictive path which links the root node to a fictive node, this fictive node does not appear in the covering but its presence is needed for the computation of the final covering.

Though the complexity of the algorithm is exponential according to the measure of redundancy C, the optimal solution is computable in reasonable time using a proper choice of the initial set of paths. In order to avoid the complexity issue, we use a simple heuristic in order to reduce the redundancy while constructing the input set of paths: first, we compute all the paths up to a fixed length (Fig. 2a), then the redundant paths with the smallest weight are removed until C is lower than a given threshold (Fig. 2b). By redundant paths, we mean paths which can be removed without breaking the covering clause i.e. all edges

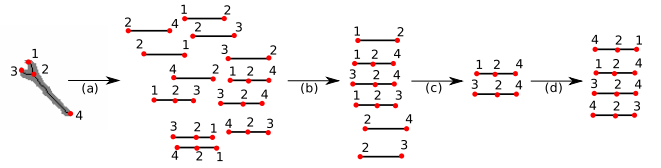


Fig. 2. Computation of the tree covering by paths: (a) Extraction of paths, (b) Reduction of redundancy, (c) Edge-covering, (d) Addition of the symmetric of the paths

must remain covered by at least one path. Finally, we run the covering algorithm on this last set of paths which produces the final bag of paths covering the whole graph (Fig. 2c).

For the sake of completeness, the symmetric of each path belonging to the bag is added to it (Fig. 2d). The symmetric of a path is the path which has the inverse route, i.e. the inverse of the path 124 is 421. Note, that the resulting bag of paths offers a complete representation of the shape as no edge is forgotten. The next section considers the construction of a bag of paths kernels.

4 Bag of Paths Kernel

Let us consider a graph $G = (V, E)$ where V denotes the set of vertices and $E \subset V \times V$ the set of edges. Let us additionally consider a bag of paths P associated to G, we denote by $|P|$ the number of paths inside P. We suppose that a positive definite path kernel, denoted K_{path}, is available.

By considering two bags of paths P_1, associated to a graph G_1 and P_2, associated to a graph G_2, as sets, we construct a bag of paths kernel by averaging the path kernel [15,9,10] results between all couples of paths from each bag:

$$K_{mean}(P_1, P_2) = \frac{1}{|P_1|}\frac{1}{|P_2|}\sum_{h \in P_1}\sum_{h' \in P_2} K_{path}(h, h'). \tag{1}$$

This kernel defines a positive definite kernel in the bag of paths domain [17]. However if the bags contain lots of paths, the kernel tends to average the information and so looses its efficiency. If the data follow a Gaussian law, a one-class SVM can be used to estimate the characteristics of the law. This leads to a kernel [10] based on the angle between the mean vectors which reduces the two bags of paths to their main characteristic, but is hence only positive semi-definite on the bags of paths.

We propose to control the average effect of the mean kernel by enforcing the weight of the paths near the mean path. This trick assumes that the distribution of the paths inside the bag almost follows a Gaussian law. Our kernel is thus constructed from the mean kernel with two additional control terms: first, we weight the paths by their relevance inside the graphs and second, we weight

the paths by their distance to the center of their sets. These two weights lead to the following kernel:

$$K_{weighted}(P_1, P_2) = \frac{1}{|P_1|}\frac{1}{|P_2|}\sum_{h\in P_1}\sum_{h'\in P_2} < K_{path}(h,m), K_{path}(h',m') >^d \frac{\omega(h)}{W}\frac{\omega(h')}{W'}K_{path}(h,h'). \quad (2)$$

where $d \in \mathbb{R}^+$, m and m' denote the mean paths of P_1 and P_2, $\omega(h)$ (resp. $\omega(h')$) denotes the sum of the edge's weights of h (resp. h') and W (resp. W') the whole weight of the graph containing h (resp. h'). The value of the path kernel between a path and the mean path of its bag P is defined as: $K_{path}(h,w) = \frac{1}{|P|}\sum_{h_i\in P} K_{path}(h,h_i)$. The kernel $K_{weighted}$ is positive definite [17] and so defines a metric between bags of paths.

We directly use this last kernel, as graph kernel i.e. $K(G_1, G_2) = K_{weighted}(P_1, P_2)$ where P_1 (resp. P_2) is the bag of paths associated to G_1 (resp. G_2). Obviously, this graph kernel is only positive semi-definite as two similar bags of paths can be defined from two different graphs. However, the positive semi-definitiveness is not a real issue since shape graphs are closely characterized by their features attributed to edges and nodes. So equality between two bags of paths would mean that the two shapes share many common sub-parts and are therefore almost similar.

5 Hierarchical Kernel

Bags of paths kernel are built upon a path kernel. This kernel can be viewed as a similarity measure for paths. Kashima [18] proposes the following path kernel:

$$K_{path}(h,h') = K_v(\varphi(v_1),\varphi(v'_1))\prod_{i=2}^{|h|} K_e(\psi(e_{v_{i-1}v_i}),\psi(e_{v'_{i-1}v'_i}))K_v(\varphi(v_i),\varphi(v'_i)), \quad (3)$$

where $\varphi(v)$ and $\psi(e)$ denote respectively the vectors of features associated to the node v and the edge e and K_v and K_e denote respectively vertices and edges kernels. This kernel is positive definite, if K_v and K_e are positive definite. Edges and vertices kernels are usually built using Gaussian radial basis kernel on the difference of feature vectors.

However, since graphs are constructed from shape skeletons, they are sensitive to shape perturbations. On complex shapes, severe shape modifications may lead to inaccurate comparison while working on paths. In order, to deal with this problem, we introduce an edit process inside the path kernel.

From a path point of view, the perturbation result in additional nodes and edges. Using this observation, we construct two elementary operations on paths: *node suppression* and *edge contraction*. The edit process is described in [11] and briefly recalled bellow for completeness.

Each edition corresponds to a deformation of the shape. Fig. 3 shows the effect of each edition operation on a simple shape (Fig. 3a): the suppression of a node

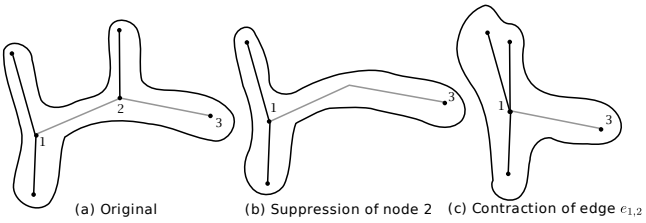


Fig. 3. Effect of the edit process

results in the partial loss of the shape (Fig. 3b) and the contraction of an edge results in the contraction of the shape (Fig. 3c). As these operations change the shape, all the attributes are updated using the new shape information. Finally, we construct a path kernel by comparing paths and their rewritings.

5.1 Edition Path Kernel

Let us denote by κ the function which applies the cheapest operation on a path [10,11] and D the maximal number of reductions. The successive applications of the function κ associates to each path h a sequence of reduced paths $(h, \kappa(h), \ldots, \kappa^D(h))$. A cost $cost_k(h)$ is associated to each reduced path $\kappa^k(h)$. This cost is defined as the sum of the costs of the k operations needed to obtain $\kappa^k(h)$ [11].

Using K_{path} as path kernel, the idea is to construct another path kernel with a control on the edition process. The kernel K_{edit} is defined as a sum of kernels between reduced paths thus leading to a hierarchical comparison of paths. Given two paths h and h', the kernel $K_{edit}(h, h')$ is thus defined as:

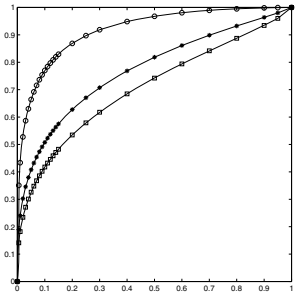
$$K_{edit}(h, h') = \frac{1}{D+1} \sum_{k=0}^{D} \sum_{l=0}^{D} \exp\left(-\frac{\text{cost}_k(h)+\text{cost}_l(h')}{2\sigma_{cost}^2}\right) K_{path}(\kappa^k(h), \kappa^l(h')), \quad (4)$$

where σ_{cost} is a tuning variable.

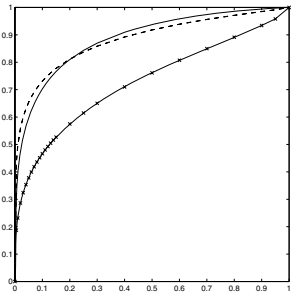
This kernel is composed of two parts: a scalar product of the edition costs in a particular space and a path kernel. For a small value of σ_{cost} the behavior of the kernel will be close to K_{path} as only low edition costs will contribute to the sum. For a high value every editions will contribute with an approximately equal importance. This kernel is positive definite on the domain of paths as it is a kernel between the hierarchies of paths (see [17,11] for more details).

6 Results

We perform in the following two experiments which compare the performances of 1) the mean kernels with the random walk kernel and 2) kernels based on different kind of bags of paths. For all these experiments, we used the following RBF coefficients (Section 2): for the perimeter: $\sigma_{perimeter} = 0.1$, for the distance of the coefficients of the order 2 polynomials describing the radius evolution: $\sigma_{radius} = 5.0$,



(a) Difference between mean kernels and the random walk kernel



(b) Difference between other kind of bags

(c) Three birds

Fig. 4. ROC curves: (a) using K_w (the curve with circles) and K_{unw} (the curve with squares) compared to the random walk kernel (the curves with the stars), (b) using K_w with all the paths (the curve with crosses), with only one percent of the heaviest paths (the solid curve) and only ten percent of the heaviest path (the dashed curve). (c) The three birds of the training set.

for the distance between the coefficients of the order 5 polynomials describing the orientation evolution: $\sigma_{orientation} = 2000$ and for the distance to the gravity center: $\sigma_{gravity\ center} = 0.5$. Each of these values have been set in order to provide the best results on average through several experiments. Concerning parameter used by several kernels, the best value of a coefficient for a particular kernel was also the best value for the remaining kernels. This last result may be explained by the fact that all the kernels share the same path kernel.

When building the different bags of paths, we only consider paths up to a length of three edges. For the edition process, we perform up to 3 editions on each paths [11] and fix the coefficient of the kernel over the edition cost as $\sigma_{cost} = 0.5$ (Eq. (4)). The weighted mean kernel is built using $K_{weighted}$ with $d = 5$ and is denoted by K_w and the unweighted mean kernel is built using $K_{weighted}$ with $d = 0$ and is denoted by K_{unw}.

For the following two experiments, we use the 216 shapes of the LEMS database [19] and consider the one-class classification of birds against all the shapes of this database. This classification is performed using the kernel PCA approach proposed by Hoffmann [20] and the results are used to built the ROC curves. The training set contains only three birds (Fig. 4c). The classification times for the different kernels are given in Tab 1.

6.1 Experiment I: Difference between the Mean Kernels and Random Walk Kernel

Fig. 4a shows the ROC curves obtained using the random walk kernel and the K_w and K_{unw} kernels combined with the covering algorithm for the construction of the bag of paths. K_{unw} shows the worst performance as the information is lost by the average. The random walk kernel shows an improvement compared

Table 1. Classification times: training and prediction

Kernel	Times
Random walk	12min40
K_{unw} combined with covering algorithm	3min50
K_w combined with covering algorithm	6min40
K_w using all the paths	153min50
K_w using 1 percent of the heaviest paths	10s
K_w using 10 percent of the heaviest paths	3min40

to the latter kernel, however since it considers all the walks of the graphs, it is sensitive to the structural noise. The best result is provided by K_w, as expected the average drawbacks are attenuated by the weighting.

6.2 Experiment II: Bag of Paths Kernels

Fig. 4b shows the ROC curves obtained using different bags of paths: one with all the paths, one with only 1 percent of the heaviest paths and the last one with 10 percent of the paths. All these bags of paths have been used with the weighted mean kernel K_w. The results prove that considering all the paths is not the best idea and the resulting kernel performs like K_{unw} combined with the covering algorithm, but requires more times to classify (Tab. 1). Using one percent of the heaviest paths or ten percent lead to similar results. This last result proves that most of the information is hold by the heaviest paths, however using the covering algorithm leads to better results as the description of shapes is more complete.

7 Conclusion

In this paper, we have defined a graph kernel based on an edition process and a selection of paths. Since the whole shape is covered by a set of paths with a minimal redundancy, the covering algorithm leads to accurate comparisons. Our results prove that removing redundancy inside the bag of paths leads to more efficient and faster kernel. In the future, we would like to improve the bag of paths kernel using more specialized set kernels.

References

1. Pope, A.R.: Model-based object recognition: A survey of recent research. Technical Report TR-94-04, University of British Columbia (1994)
2. Siddiqi, K., Bouix, S., Tannenbaum, A., Zucker, S.W.: Hamilton-jacobi skeletons. International Journal of Computer Vision 48(3), 215–231 (2002)
3. Goh, W.B.: Strategies for shape matching using skeletons. Computer Vision and Image Understanding 110, 326–345 (2008)

4. Siddiqi, K., Shokoufandeh, A., Dickinson, S.J., Zucker, S.W.: Shock graphs and shape matching. Int. J. Comput. Vision 35(1), 13–32 (1999)
5. Pelillo, M., Siddiqi, K., Zucker, S.: Matching hierarchical structures using association graphs. IEEE Trans. on PAMI 21(11), 1105–1120 (1999)
6. Vishwanathan, S., Borgwardt, K.M., Kondor, I.R., Schraudolph, N.N.: Graph kernels. Journal of Machine Learning Research 9, 1–37 (2008)
7. Neuhaus, M., Bunke, H.: Edit distance based kernel functions for structural pattern classification. Pattern Recognition 39, 1852–1863 (2006)
8. Neuhaus, M., Bunke, H.: Bridging the Gap Between Graph Edit Distance and Kernel Machines. World Scientific, Singapore (2007)
9. Suard, F., Rakotomamonjy, A., Bensrhair, A.: Kernel on bag of paths for measuring similarity of shapes. In: Proceedings of ESANN 2007, 15th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 25-27, 2007, pp. 355–360 (2007)
10. Dupé, F.X., Brun, L.: Hierarchical bag of paths for kernel based shape classification. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 227–236. Springer, Heidelberg (2008)
11. Dupé, F.X., Brun, L.: Edition within a graph kernel framework for shape recognition. In: Torsello, A., Escolano, F., Brun, L. (eds.) GBR, pp. 11–20. Springer, Heidelberg (2009)
12. DiMatteo, I., Genovese, C., Kass, R.: Bayesian curve fitting with free-knot splines. Biometrika 88, 1055–1071 (2001)
13. Meyer, F.: Topographic distance and watershed lines. Signal Proc. 38(1) (1994)
14. Torsello, A., Hancock, E.R.: A skeletal measure of 2d shape similarity. CVIU 95, 1–29 (2004)
15. Suard, F., Rakotomamonjy, A., Bensrhair, A.: Mining shock graphs with kernels. Technical report, LITIS (2006), `http://hal.archives-ouvertes.fr/hal-00121988/en/`
16. Guo, J., Niedermeier, R., Uhlmann, J.: Two fixed-parameter algorithms for vertex covering by paths on trees. Information Processing Letters 106, 81–86 (2008)
17. Haussler, D.: Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz (1999)
18. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernel between labeled graphs. In: Proc. of the Twentieth International conference on machine Learning, pp. 321–328 (2003)
19. LEMS: shapes databases, `http://www.lems.brown.edu/vision/software/`
20. Hoffmann, H.: Kernel PCA for novelty detection. Pattern Recognition 40, 863–874 (2007)