Kernel on Graphs based on Dictionary of Paths for Image Retrieval

Jean-Emmanuel Haugeard, Sylvie Philipp-Foliguet and Philippe-Henri Gosselin *
ETIS, CNRS, ENSEA, Univ Cergy-Pontoise
6 avenue du Ponceau, BP44, F95014 Cergy-Pontoise, France
{jean-emmanuel.haugeard,sylvie.philipp,gosselin}@ensea.fr

Abstract

Recent approaches of graph comparison consider graphs as sets of paths [6, 5]. Kernels on graphs are then computed from kernels on paths. A common strategy for graph retrieval is to perform pairwise comparisons. In this paper, we propose to follow a different strategy, where we collect a set of paths into a dictionary, and then project each graph to this dictionary. Then, graphs can be classified using powerful classification methods, such as SVM. Furthermore, we collect the paths through interaction with a user. This strategy is ten times faster than a straight comparisons of paths. Experiments have been carried out on a database of city windows.

1. Introduction

The problem of graph comparison is a topic which has been widely studied in the literature for several decades [2]. Most of the applications concern chemistry where vertices are labeled with an atom name, but it is especially hard to compare graphs whose both vertices and edges are attributed with numerical values. In image indexing, an image can be represented by an adjacency graph (of regions, of points, of contours), and the problem is to compute a similarity between graphs, taking into account both the structure of the graph and the similarity between vertices and between edges. In the context of interactive image retrieval, retrieving a particular object is performed with an initial example of the object and the problem is solved as a semi-supervised classification problem, using a SVM classifier with kernel functions for instance. As the classification is iterated with examples annotated at each feedback loop, it is important to have a fast computation of the similarity between graphs. Recent approaches of graph comparison propose to consider graphs as sets of paths [6, 5]. Kernels on graphs are then computed from kernels on paths. The number of paths that can be drawn on a graph is infinite and thus the computational complexity is very high, which is penalizing for on-line database browsing. To solve this problem, we will use a dictionary of paths as others use a dictionary of visual words. Building a dictionary of paths is not an easy task, especially if graphs are attributed with numerical values (and not only with labels). We propose to build this dictionary on-line, from paths belonging to images annotated as relevant by the user. The idea is to build an initial dictionary with all paths of the query image and then to add the new paths from relevant images (different to those already in the dictionary). We thus build a dynamic dictionary, whose all words are different but characteristic of the relevant images.

As the dictionary is dynamically increased, the kernel on graphs must be updated as soon as a new word is added to the dictionary. The computation time is thus reduced.

Our contributions are first a way to compute dynamic dictionary of paths (section 3) and then two kernels on graphs (section 4). Applications for retrieving objects represented by sets of contours are presented in section 5.

2 Kernel on graphs, kernel on paths

A path h in a graph $G = (V, E)$ is a sequence of vertices of V linked by edges of $E : h = (v_0, v_1, ..., v_n)$, $v_i \in V$. The problem of inexact graph matching is twofold : first find a similarity measure between graphs of different sizes, then find the best match between graphs in an "acceptable" time. Sorlin [7] proposes a similarity measure which is the average value of the best similarities between vertices and edges. In [4], we have designed a new kernel on graphs:

*This work is sponsored by the project ANR 07-MDCO-007-03 iTowns [1].

$$K_{struct}(G,G') = \frac{1}{|V|}\sum_{i=1}^{|V|} \max_{\substack{h'\in G'\\ h_{v_i}\in G}} K_C(h_{v_i},h') + \frac{1}{|V'|}\sum_{i=1}^{|V'|} \max_{\substack{h\in G\\ h'_{v'_i}\in G'}} K_C(h,h'_{v'_i}) \quad (1)$$

with h_{v_i} a path starting from v_i and $|V|$ cardinal of V

K_C represents a kernel on paths. To address the computational complexity problem, the best matching path is efficiently found by the "branch and bound" algorithm. However the problem of comparing two graphs G and G' by comparing paths of same length of both graphs is still a high computational complexity problem. In this paper, we propose to compare a dictionary of paths with graphs in order to reduce computing time.

3 Dynamic dictionary and incremental kernel

Each image i is represented by an attributed relational graph $G_i = (V_i, E_i)$ where V_i can represent regions, points or contours and E_i their proximity. We want to compare these graphs according to a dictionary $D_L = \{\hat{\mathbf{h}}_l\}_{l\in[1,L]}$ of paths $\hat{\mathbf{h}}_l$. We denote by words the paths selected to enter the dictionary. The first aim of the method is to build dictionary $D_L = \{\hat{\mathbf{h}}_l\}_{l\in[1,L]}$ of paths $\hat{\mathbf{h}}_l$, but with the constraint that they are paths from graphs of images of the database.

At the beginning of a retrieval session, the dictionary D_0 is empty. Then, at each feedback step, we add paths from labeled images to the dictionary.

For each new word $\hat{\mathbf{h}}_l$, we build a minor kernel function $k_{\hat{\mathbf{h}}_l}(G_i, G_j)$, which is the similarity between image i and image j relatively to word $\hat{\mathbf{h}}_l$. We sum all these minor kernels to get the kernel according to current dictionary D_L:

$$K_L(G_i,G_j) = S\left(\sum_{l=1}^{L} k_{\hat{\mathbf{h}}_l}(G_i,G_j)\right) \quad (2)$$

with L the number of words in dictionary D_L and S a function depending on sum of minor kernels.

The incremental computation of this kernel is straightforward:

$$K_{L+1}(G_i,G_j) = S\left(\sum_{l=1}^{L} k_{\hat{\mathbf{h}}_l}(G_i,G_j) + k_{\hat{\mathbf{h}}_{L+1}}(G_i,G_j)\right)$$

If $\mathcal{H}$ is the space of paths, minor kernels $k_{\hat{\mathbf{h}}_l}(G_i, G_j)$ are computed using evaluation function $e_{\hat{\mathbf{h}}_l} : \mathcal{H} \rightarrow \mathbb{R}$ defined for path $\hat{\mathbf{h}}_l$ and distance function δ by :

$$k_{\hat{\mathbf{h}}_l}(G_i,G_j) = \delta(e_{\hat{\mathbf{h}}_l}(G_i), e_{\hat{\mathbf{h}}_l}(G_j)) \quad (3)$$

The simplest evaluation function $e_{\hat{\mathbf{h}}_l}$ equals 1 if $\hat{\mathbf{h}}_l$ is in G_i and 0 otherwise. This formula can be used for dictionary of visual words, where one only checks whether keyword $\hat{\mathbf{h}}_l$ belongs to image G_i (for example for labeled graphs).

The distance function δ compares the evaluation of graphs of images i and j regarding feature $\hat{\mathbf{h}}_l$. This function must be chosen such that the expansion of function K_L is a kernel function. Thus function $e_{\hat{\mathbf{h}}_l}$ does not need to satisfy any mathematical property to lead to a kernel function.

4 Two new kernels on graphs

In equation 3, we need a function $e_{\hat{\mathbf{h}}_l}(G_i)$ which evaluate at which point path of dictionary $\hat{\mathbf{h}}_l$ is similar to one of G_i. This function can simply be the maximal similarity between $\hat{\mathbf{h}}_l$ and a path of G_i:

$$e_{\hat{\mathbf{h}}_l}(G_i) = \max_{\mathbf{h}\in G_i} K_C(\hat{\mathbf{h}}_l, \mathbf{h}) \quad (4)$$

K_C is the similarity between two paths.

This function leads to two new kernels. In contrast to equation 1, which is not a Mercer kernel because of the "max", the kernels defined in this paper are Mercer kernels:

- Triangular kernel

$$\begin{aligned} K_L^1(G_i,G_j) &= -\sum_{l=1}^{L} k_{\hat{\mathbf{h}}_l}(G_i,G_j) \\ &= -\sum_{l} \delta(e_{\hat{\mathbf{h}}_l}(G_i), e_{\hat{\mathbf{h}}_l}(G_j)) \\ &= -d(\mathbf{x}_i,\mathbf{x}_j) \quad (5) \end{aligned}$$

  with d a distance between two vectors.

- Gaussian kernel

  The method can also be used to incrementally build a Gaussian kernel.If we compute the exponential value of kernel $K_L(G_i, G_j)$, we get a Gaussian kernel:

$$\begin{aligned} K_L^2(G_i,G_j) &= \exp\left(-\frac{1}{2\sigma^2}\sum_{l=1}^{L} k_{\hat{\mathbf{h}}_l}(G_i,G_j)\right) \\ &= \exp\left(-\frac{1}{2\sigma^2} d(\mathbf{x}_i,\mathbf{x}_j)\right) \quad (6) \end{aligned}$$

We propose to use equations 5 and 6 with other distances and compare these different kernels. We use these distances:

$$d_{L^1}(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_1 \; ; d_{\chi^1}(\mathbf{x},\mathbf{y}) = \sum_{i=0}^{n} |\frac{x_i - y_i}{x_i + y_i}|$$

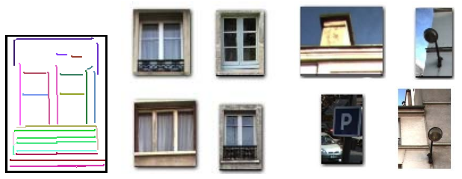
$$d_{L^2}(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_2 \; ; d_{\chi^2}(\mathbf{x},\mathbf{y}) = \sum_{i=0}^{n} \frac{(x_i - y_i)^2}{x_i + y_i}$$

5 Application

In this section,we present an application for dynamic dictionary of paths for window classification. In the previous sections, we have presented the general case of kernel on paths. Thereafter, we are interested in the special case of paths of segments of contours.

5.1 Representation of architectural entities by Attributed Relational Graphs of contours

Contours seem intuitively relevant to hold architecture information from building facades. Following the idea of perceptual grouping of contours used by Ferrari et al. [3], we propose to use our kernels for structured sets of contours. Objects are then represented by fragments of contours (figure 1 (a)) with their own characteristics and by an Attributed Relational Graph (ARG) on these fragments, where the vertices of the graph are contour segments extracted from the image and the edges of the graph represent their spatial relationships.



(a) Window graph (b) Windows (c) Negatives

Figure 1. Windows database: 70 windows and 150 negatives

In each image, edges are extracted, extended and polygonalized. Each line segment C_i is represented by a vertex v_i of this graph and the relative position of line segments C_i and C_j are represented by edge e_{ij} of the graph. The topological information (such as parallelism, proximity) can be considered only for the nearest neighbors of each line segment. We use the Voronoi diagram to find the segments that are the closest to a given segment. An edge in the ARG represents the adjacency of two Voronoi cells that is to say the proximity of two line segments.

In order to be robust to scale changes, a segment is only characterized by its direction (horizontal or vertical). If Θ is the angle between line segment C_i and the horizontal axis ($\Theta \in [0, 180[$), C_i is represented by the vertex $v_i = (cos(2\Theta), sin(2\Theta))^T$.

The edge (of the graph) $e_{ij} = (v_i, v_j)$ represents the adjacency between line segments C_i and C_j. It is characterized by the relative positions of the centers of gravity of segments C_i and C_j, denoted $g_{C_i}(Xg_{C_i}, Yg_{C_i})$ and $g_{C_j}(Xg_{C_j}, Yg_{C_j})$. The edge e_{ij} is then characterized by $e_{ij} = (Xg_{C_j} - Xg_{C_i}, Yg_{C_j} - Yg_{C_i})^T$.

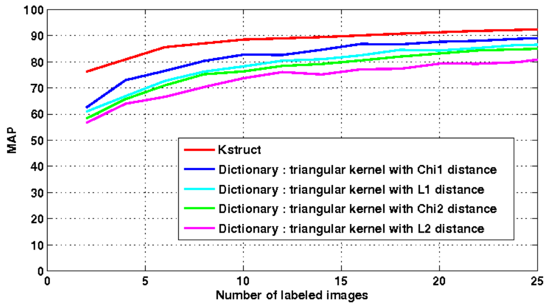
And we propose to use in 4 :

$$\begin{aligned} K_C(h_{v_i}, h') &= K_v(v_i, v'_0) \qquad (7) \\ &+ \sum_{j=1}^{|h|} Se_j O_{j,j-1} K_e(e_j, e'_j)\, K_v(v_j, v'_j) \end{aligned}$$

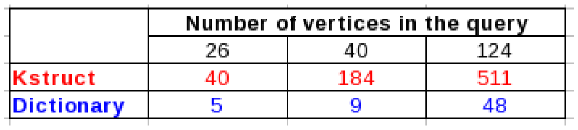
K_v and K_e are the minor kernels which define the vertex similarity and the edge similarity. Se_i is a scale penalty and $O_{i,j}$ is a weight which penalizes the paths whose segment orientations do not vary ([4]).

5.2 Words of the dictionary of paths

In [4], we addressed the problem of inexact graph matching and we were confronted with problems of computing time. With the method we propose in section 3, we get rid of global paths dictionaries and we reduce the computational time. Before any retrieval session, we extract the most representative paths from each graph G_i. Previous studies [4, 6] have shown that paths of fixed lenght l are as efficient for graph matching than paths of any lenght. To be sure to cover the whole graph, we put in H_{G_i} one path of lenght l beginning with each vertex of G_i. H_{G_i} thus contains at most $|V_i|$ paths (paths very similar can be represented by a single path). We have also shown in [4] that paths whose successive vertices (which represent contour segments) are not parallel (cf Eq 8) perform better retrieval. For this reason we choose for path $h = (v_j, ...)$ starting with vertex v_j, the one which maximizes $\max_{h_{v_i}} \sum_{k=j}^{j+|l|} \sqrt{\frac{1}{2}(1 - \langle v_k, v_{k+1} \rangle)}$. Paths with cycles, i.e. including several time the same vertex, are not accepted.



(a) Classification of the base with different distances



	Number of vertices in the query		
	26	40	124
Kstruct	40	184	511
Dictionary	5	9	48

(b) Average computation time (in milliseconds) for graph similarity computation between query and another graph

Figure 2. Comparison of various kernels on windows database.

5.3 Active classification of window candidates

We have carried out the classification with dictionary on database of 220 images : 70 windows and 150 negatives (some examples are shown in figures 1).

We simulated an active learning scheme, where the user annotates a few images at each iteration of relevance feedback, using the interactive retrieval system [6]. Each retrieval session is initialized with one image containing an example of window. The system labels at each iteration one image either as window or as false detection, and the system updates the ranking of the database according to these new labels.The whole process is iterated 100 times with different initial images and the Mean Average Precision (MAP) is computed from all these sessions (figure 2 (a)).

We tested our dictionary kernels on graphs with different sizes of paths and compared with kernel of equation 1 ([4]). With the dictionary, the best result is obtained with path of size 4 whereas with $Kstruct$ kernel (equation 1), the best result is obtained with path of size 8. In addition we tested the different kernels and distances (figure 2(a)) and compare with the $Kstruct$ kernel.

With only four examples (2 positives and 2 negatives) (figure 2 (a)), we obtain 80% of average precision with Kstruct, whereas with our dictionary we have 74%. However our method is about ten times faster. On average, an annotated graph with 26 vertices, the comparison with another graph is obtained in 40 milliseconds with $Kstruct$ (figure 2 (b)) and we obtain a comparison in 5 milliseconds for dictionary of paths. For a graph of 120 vertices, time decreases from 511 milliseconds to 48 milliseconds.

6 Conclusion

In this paper, we addressed the problem of inexact graph matching, by working out a new kernel on graphs based on a dynamic dictionary of paths. The results are quite encouraging. If results are slightly less accurate, we obtain a time saving of ten. These results will probably be improved by removing similar paths from the dictionary.

References

[1] Image-based Town On-line Web Navigation and Search engine: http://www.itowns.eu/.

[2] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[3] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *PAMI*, 30(1):36–51, 2008.

[4] J.-E. Haugeard, S. Philipp-Foliguet, and F. Precioso. Windows and facades retrieval using similarity on graph of contours. In *IEEE International Conference on Image Processing (ICIP 09)*, November 2009.

[5] H. Kashima and Y. Tsuboi. Kernel-based discriminative learning algorithms for labeling sequences, trees and graphs. In *ICML*, 2004.

[6] J. Lebrun, S. Philipp-Foliguet, and P.-H. Gosselin. Image retrieval with graph kernel on regions. In *ICPR*, dec 2008.

[7] S. Sorlin, O. Sammoud, C. Solnon, and J. Jolion. Mesurer la similarité de graphes. *Extraction de Connaissance à partir d'Images (ECOI06)*, pages 21–30.