Object Categorization using Kernels combining Graphs and Histograms of Gradients

F. Suard, A. Rakotomamonjy, A. Bensrhair
Email : frederic.suard@insa-rouen.fr

LITIS, PSI, INSA de Rouen
Avenue de l'université
76801 Saint Etienne du Rouvray, FRANCE

Abstract. This paper presents a method for object categorization. This problem is difficult and can be solved by combining different information sources such as shape or appearance. In this paper, we aim at performing object recognition by mixing kernels obtained from different cues. Our method is based on two complementary descriptions of an object. First, we describe its shape thanks to labeled graphs. This graph is obtained from morphological skeleton, extracted from the binary mask of the object image. The second description uses histograms of oriented gradients which aim at capturing objects appearance. The histogram descriptor is obtained by computing local histograms over the complete image of the object. These two descriptions are combined using a kernel product. Our approach has been validated on the ETH80 database which is composed of 3280 images gathered in 8 classes. The results we achieved show that this method can be very efficient.

1 Introduction

Object categorization problem is difficult and still presents open issues. Many researches have focused on this topic and yet it has been solved only for particular situations. An object categorization system contains two main parts. First, feature extraction has to be the most exhaustive object representation in order to keep maximum information concerning the object. The second part consists of a classifier which should be able to learn the category of an object from this representation and then to predict the most correctly as possible the category of a new unseen object.

Representing a static object can be done along two ways [12], globally and locally. The first way consists in representing the general shape of an object. The other way can bring information about the object appearance for a stronger discrimination. Then for a categorization purpose these two global representations have to be appropriately combined with an objective of perfomance enhancement. One way for mixing these representations has been presented in Leibe et al. [12] and is based on a decision tree. Another way of combination resides in the properties of the SVM classifier, which can deal with mixture of kernels [5, 1].

Recently, there has been a growing interest around object representation based on graphs. One major interest of graph resides in its property which keeps the object shape topology. This property has been used in different methods for object categorization using graphs : [14, 15, 18, 11], where graph classification is achieved by measuring their similarity [14, 15].

The result of similarity measure obtained from graph matching algorithms [3, 14, 8] is a metric. This metric is then used to find the most similar object, or to cluster objects, according to the distance between them. So that efficiency of these methods depends on the quality of the similarity measure. A weakness of graph matching algorithms is that they essentially deal with labelled graphs for which labels are a single numerical value. On top of that, information is mainly brought on edges and rarely on vertices [14, 18].

Another way to compare or to classify graphs is to use kernel methods and the so-called kernel trick.

Hence, measuring graph similarity can be addressed by considering kernels function on graphs. In [10], Kashima et al. defined a kernel function for labeled graphs. This function can be interpreted as an inner product on two graphs, obtained by comparing edges and vertices that have been crossed during random walks on the graphs. Then a major particularity of this kernel is the use of kernels between vertices and edges. It means that labels can be complex structures, like vectors, histograms or set of histograms, instead of a single real value, which is the case for most of graph matching algorithms.

Another representation of an object consists in describing the object appearance [9, 12, 7]. Recents works have shown that efficient and robust appearance-based cues can be obtained from histogram of oriented gradient (HOG) in images [13]. One advantage of this method is to bring information on both object appearance and object contours. This method is also an answer to the variability problem, since representing an image thanks to histogram of gradient is very robust for scale invariance, or different lighting conditions.

Recently, Dalal and Triggs have further developed this idea of histogram of gradient and have achieved excellent recognition rate of human detection in images [6]. This work pointed out the problem of variability, and proposed an efficient way to solve it.

In this paper we present a method for object categorization using a combination of representations based on kernels and a SVM classifier [17]. We decided to take advantage of graph properties for a global shape representation of an object. However, instead of using graph matching algorithms, we introduce the use of graph kernels for object recognition problem. The object shape representation is combined with an appearance representation based on local histogram of gradients. We build an appropriate kernel that mixes kernels from these two representations by a product. The resulting kernel is then fed into a SVM classifier for categorization. Our paper aims at analysing the categorization performance of the overall approach.

This paper is organized as follows. The first section 2.1 presents our method to design a graph. Starting from an image, the morphological skeleton is obtained thanks to the image binary mask. When the graph structure is complete, we add some labels on both vertices and edges, with regards of the original image. The second part 2.2 presents the HOG descriptor. The third part 2.3 presents briefly the SVM classifier used for multiclass. Next, we describe the graph kernel of Kashima 2.4. Finally, we depict some results in section 3. The test has been accomplished on the ETH-80 database, which has already been used in different approaches to test their efficiency [9, 18, 12].

2 Method description

2.1 Graph

In this part, we will describe briefly our method used to transform an image into a graph. The aim is to keep the main information contained in a shape, that is to say geometric properties or topologic properties. This last property is particularly interesting in our case. If a shape is made of a set of sub-parts, the skeleton will preserve the connectivity and the shape arrangement.

Graph designing As we mentioned before, we tackle the problem of object recognition represented with an image. In our case, one image corresponds to a single object. The first step consists in extracting the morphological skeleton from the binary mask. We used the same method which is described in [14].

A skeleton can be defined as a line representation of an object, that is to say it is one-pixel large and placed around the middle of the object and preserves the geometry and topology object

Given the definition of Lantuejoul [4], a skeleton subset of a black and white image $S_k(A)$ is defined as : $S_k(A) = E(A, kB) - [E(A, kB)\, o\, B]\, k = 0, 1, ...K$ where B is a structuring element, and K is the largest value of k before the set $S_k(A)$ becomes empty. The skeleton is then the union of the skeleton subsets : $S(A) = \cup_{k=0}^{K} S_k(A)$

Once the skeleton is obtained, we can build the graph. A graph G is made up of vertices and edges $G = (V, E)$. A vertex is a junction between different edges. We can differentiate two types of vertices : nodes (V_n), which are a junction of many edges and vertices which are edge ending (V_s).
To build the graph we look at the type of each skeleton pixel. If the pixel has only one neighbor, this pixel is an edge ending. If the pixel neighboring corresponds to a defined mask, for example $\begin{pmatrix} 0\,1\,0 \\ 1\,1\,1 \\ 0\,0\,0 \end{pmatrix}$, then this pixel is defined as a vertex.

The next step consists in searching paths between vertices. The aim is to search, for each vertex, all vertices directly linked by a branch of the skeleton. These vertices are found by walking along the skeleton pixels. The path between two vertices should not contain another vertex.

Graph labeling One important aspect in our work, is that we deal with attributed graphs. It means that graph components are labeled. A particularity is that we can have a vector of structured data labels for each component, since owing to kernel methods we have no limitation when we compare two graphs (see 2.4).

– For a vertex, for instance, we can compute the following labels :
  - node coordinates,
  - size of the structured element at node,
  - color mean and variance of the region described by the structured element.
– For edges :
  - length,

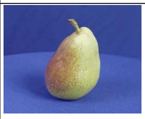
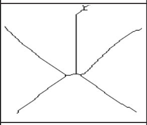
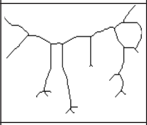
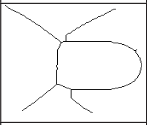
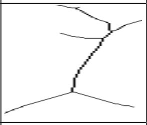
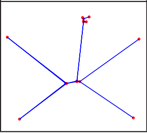
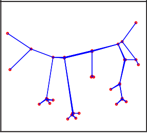
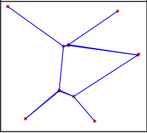
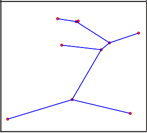
	Apple	Cow	Cup	Pear
Image				
Skeleton				
Graph				

Fig. 1. Example of objects coming from the database ETH-80, their skeleton, and their graph.

- orientation,
- area defined by the intersection of all structured elements placed on the edge,
- luminosity, colour mean, variance and texture characteristics (homogeneity, dissimilarity, contrast, entrophy, energy) of the region defining above.

As we can see, we could obtain various information concerning shape. Some features are able to describe the shape topology, like edge's length, orientation and area. We could complete them with information about the shape texture.

2.2 Histograms of Oriented Gradients

In the context of object recognition, the use of edge orientation histogram has gain popularity [16, 6]. However, the concept of dense and local histogram of oriented gradients (HOG) is a method introduced by Dalal et al.[6]. The aim of such method is to describe an image by a set of local histograms. These histograms count occurences of gradient orientation in a local part of the image. In this work, in order to obtain a complete descriptor of an image, we have computed such local histograms of gradient according to the following steps :

1. compute gradients of the image,
2. build histogram of orientation for each cell,
3. normalize histograms within each block of cells.

Gradient computation The gradient of an image has been simply obtained by filtering it with two one-dimensional filters :

- horizontal : $\begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$
- vertical : $\begin{pmatrix} -1 & 0 & 1 \end{pmatrix}^T$

Gradient could be signed or unsigned. This last case is justified by the fact that the direction of the contrast has no importance. In other words, we would have the same results with a white object placed on a black background, compared with a black object placed on a white background. In our case, we have considered an unsigned gradient which value goes from 0 to π.

The next step is orientation binning, that is to say to compute the histogram of orientation. One histogram is computed for each cell according to the number of bins.

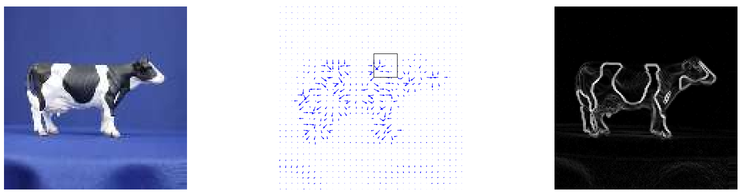


Fig. 2. This figure shows the gradient computation of an image. (left) is the original image, (middle) shows the direction of the gradient, (right) depicts the original image according to the gradient norm.

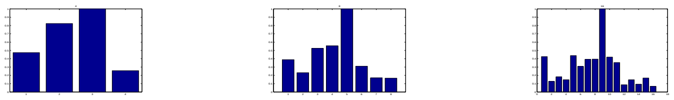


Fig. 3. This figure shows the histograms of gradient orientation for (left) 4 bins, (middle) 8 bins (right) 16 bins.

Cell and block descriptors The particularity of this method is to split the image into different cells. A cell can be defined as a spatial region like a square with a predefined size in pixels. For each cell, we then compute the histogram of gradient by accumulating votes into bins for each orientation. Votes can be weighted by the magnitude of a gradient, so that histogram takes into account the importance of gradient at a given point. This can be justified by the fact that a gradient orientation around an edge should be more significant than the one of a point in a nearly uniform region. Some examples of histogram obtained for the square region given in the middle image of figure 2 are shown in figure 3. As expected, the larger the number of bins is, the more detailed the histogram is.

Block Normalization When all histograms have been computed for each cell, we can build the descriptor vector of an image concatenating all histograms in a single vector. However, due to the illumination variations and other variability in the images, it is necessary to normalize cells histograms. Cells histograms are locally normalized, according to the values of the neighboured cells histograms. The normalization is done among a group of cells, which is called a block.

A normalization factor is then computed over the block and all histograms within this block are normalized according to this normalization factor. Once this normalization step has been performed, all the histograms can be concatenated in a single feature vector.

Different normalization schemes are possible for a vector V containing all histograms of a given block. The normalization factor nf could be obtained along these schemes :

– none : no normalization is applied on the cells, $nf = 1$.
– L1-norm : $nf = \frac{V}{\|V\|_1 + \varepsilon}$
– L2-norm : $nf = \frac{V}{\sqrt{\|V\|_2^2 + \varepsilon^2}}$

ε is a small regularization constant. It is needed as we sometime evaluate empty gradients. The value of ε has no influence on the results.

Note that according to how each block has been built, a histogram from a given cell can be involved in several block normalization. In this case, the final feature vector contains redundant information which has been normalized in a different way. This is especially the case if blocks of cells have overlapping.

2.3 SVM Classifier

Support Vector Machine The Support Vector Machines classifier is a binary classifier algorithm that looks for an optimal hyperplane as a decision function in a high-dimensional space [2, 17, 5]. Thus, consider one has a training data set $\{\mathbf{x}_k, y_k\} \in \mathcal{X} \times \{-1, 1\}$ where $\mathbf{x}_k$ are the training examples and y_k the class label. At first, the method consists in mapping $\mathbf{x}_k$ in a high dimensional space owing to a function Φ. Then, it looks for a decision function of the form : $f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b$ and $f(\mathbf{x})$ is optimal in the sense that it maximizes the distance between the nearest point $\Phi(\mathbf{x}_i)$ and the hyperplane. The class label of $\mathbf{x}$ is then obtained by considering the sign of $f(\mathbf{x})$. This optimization problem can be turned in this following one :

$$\min_{\mathbf{w},\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{k=1}^{m} \xi_k \tag{1}$$

under the constraint $\forall k, \quad y_k f(\mathbf{x}_k) \geq 1 - \xi_k$, where the ξ_k are the slack variables. The solution of this problem is obtained using the Lagrangian theory and it is possible to show that the vector $\mathbf{w}$ is of the form :

$$\mathbf{w} = \sum_{k=1}^{m} \alpha_k^* y_k \Phi(\mathbf{x}_k) \tag{2}$$

where α_i^* is the solution of the following quadratic optimization problem :

$$\max_{\alpha} W(\alpha) = \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k,\ell}^{m} \alpha_k \alpha_\ell y_k y_\ell K(\mathbf{x}_k, \mathbf{x}_\ell) \tag{3}$$

subject to $\sum_{k=1}^{m} y_k \alpha_k = 0$ and $\forall k, 0 \leq \alpha_k \leq C$, where $K(\mathbf{x}_k, \mathbf{x}_\ell) = \langle \Phi(\mathbf{x}_k), \Phi(\mathbf{x}_\ell) \rangle$. According to equation (2) and (3), the solution of the SVM problem depends only on the Gram matrix K. Hence, in our case, the classification with SVMs only needs a kernel, which is, in our case, a combined kernel.

Kernels combination In our method, we use a combination of kernels [5, 1]. A kernel can be defined as a combination of positive-definite kernels.

Let K_1 and K_2 be kernels over $\mathcal{X} \times \mathcal{X}$, $0 \leq \lambda \leq 1$, $a \geq 0$, the following functions are kernels :

– $K(x, y) = \lambda K_1(x, y) + (1 - \lambda) K_2(x, y)$
– $K(x, y) = a K_1(x, y)$
– $K(x, y) = K_1(x, y) \times K_2(x, y)$

In our case, we combine a graph kernel with a linear kernel computed from the HOG datas thanks to a product.

2.4 Graph Kernel

We use the inner product between graphical representations based on Kashima et al. paper's [10]. The idea is to compare two label sequences generated by two synchronized random walks on the two graphs. This formula gives the comparison of each vertices values and edges values, which are encountered for each paths starting from each vertex of graphs, weighted by the probability to cross the vertices and the edges. This formula shows that computation could be exhaustive and test every possible path combination.

$$K(G, G') = \sum_{h} \sum_{h'} K_z(h, h') \times p(h, G) \times p(h', G')$$
$$K_z(h, h') = K_v(h_1, h'_1) \prod_{i=2}^{l} K_e(h_{2i-2}, h'_{2i-2}) \times K_v(h_{2i-1}, h'_{2i-1}) \tag{4}$$

with $K_v(h, h')$, the kernel function defined for vertices, $K_e(h, h')$, the kernel function defined for edges. $p(h, G)$ is the path probability for a path h in graph G. This probability is function of the probability to start at a given vertex and the transition probability between two vertices.

The detailed computation of $K(G, G')$ is given in the paper of Kashima and al. [10].

The complexity of this computation is $\mathcal{O}\left((|G||G'|)^2\right)$, with $|G|$ the number of nodes in graph G. For this reason the number of vertices in each graph have to be as small as possible.

The graph kernel suggests that a kernel between vertices and a kernel between edges have to be defined. In our case, we have chosen to use a classical gaussian kernel since nodes and edges are labeled with vectorial values :

$$K_n(h, h') = K_a(h, h') = \exp\left(-\frac{||h - h'||^2}{2\sigma^2}\right) \tag{5}$$

where σ is the bandwidth of the gaussian kernel.

Note that the graph kernel depends only on the probability transition between vertices and kernels between vertices and kernel between edges. This means that the label information of edges and vertices can be richer than it is at the present time. In fact, since we only need an inner product values, labels can be a non-vectorial data which admit a kernel.

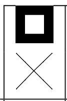
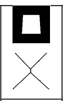
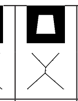
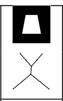
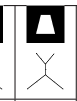
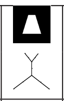
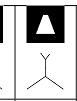
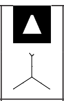
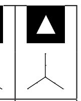
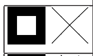
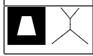
									
	1.000	0.954	0.946	0.932	0.911	0.892	0.856	0.866	0.943
	0.911	0.975	0.989	0.996	1.000	0.997	0.983	0.979	0.925

Fig. 4. Value of the inner product between a square graph and a trapezoid graph with the graphs coming from the transform of a square to a triangle.

In order to evaluate the pertinence of this method, we have computed values of the inner product between the graph of a square with graphs of objects coming from the progressive transformation of a square (■) to a triangle (▲). Results are shown on figure 4. Since the inner product is normalized, it can be considered as a distance :

$$d(x,y)^2 = \langle x,x\rangle + \langle y,y\rangle - 2\langle x,y\rangle = \langle x,x\rangle + \langle y,y\rangle - 2\langle x,y\rangle = 2(1-\langle x,y\rangle) \quad (6)$$

We conclude from that :

– when we compare the inner product $K(G(■), G(■))$ with $K(G(■), G(▲))$ and $K(G(■), G(⏢))$, we can see that the triangle graph seems to be closer of the square graph than the trapezoid graph, but the trapezoid graph seems to be closer of the square graph. This can be explained by the fact that the graphs of square and triangle are adjacent, compared with the graph of trapezoid. In fact, trapezoid graph has a central branch which is not present in the other graphs. So the triangle graph differs from square graph only by one missing branch.
– We also can notice a symmetry concerning the trapezoid result. In our example, the triangle and the square are equally placed from the central trapezoid.
– the result is symmetric : $K(G_1, G_2) = K(G_2, G_1)$.

Compared with graph matching algorithms, [3, 14, 8], the method proposed by Kashima has a great interest. It can be used by a SVM classifier, since some similarity measures have not the required properties, in particular the symmetry. This method can also deal with complex structures as labels, when other methods only treat single numerical value.

3 Results

We now have to evaluate the efficiency of this method. The test is accomplished on a complete database : ETH-80 (`http://www.mis.informatik.tu-darmstadt.`

`de/Research/Projects/categorization/`). This base contains 80 objects, dispatched over 8 classes : apple, pear, cow, dog, horse, cup, car, tomatoe. Each object has been captured from different points of view to produce 41 views for each object.

Our SVM is used for multi-class, with one-against-one method: we trained $(\frac{n(n-1)}{2})$ binary classifiers for n classes. For classification, a vote is established for all classifiers. The tested data will be labeled as the class that as more votes. The weight for miss-classified points C (1) was established at 1000. The following results are given for a leave-one-object-out crossvalidation method. We remove 41 images corresponding to the same object at a time. The learning set is composed of all remaining objects, and we classify each view of the tested object. To evaluate the efficiency of the different methods, we first studied independently the graph kernel and the HOG method.

3.1 Graph kernel

The aim of graph is mainly to discriminate object shape. To fill this condition, we choose to use the labels which are pertinent for a shape description. In other words, we conserve labels which give information about the object shape topology. We retain the following labels :

- for vertices : size of structured element, coordinates,
- for edges : orientation, length, strength, area,

Each characteric was normalized to have a mean equal to 0, and a standard deviation equal to 1.

We also evaluate the influence of graph size by limiting the number of nodes for each graph. Results are shown on the table below :

number of nodes	5	10	15	20	25	30
good recognition rate	75,2%	79,4%	80,5%	82,1%	82,9%	83,8%

Due to the complexity of the computation (2.4), we limit the number of vertices to 30 per graph. We obtained a good recognition rate of 83,8%. This result can be explained by the fact that classes dog, horse and cow are strongly mixed as it is shown on the confusion matrix (figure 5). One reason resides in the skeleton designing, which is too similar for these objects.

3.2 HOG

As we saw in section 2.2, the HOG descriptor actually involves many parameters. To tune these parameters correctly, we completed a test to evaluate optimal set of parameters, retrieve this following set :

- image size : 96×96 pixels,
- size of cell : 4×4 pixels,
- size of block : 2×2 cells,
- overlap of blocks : 1 cell,
- normalization factor for block : L2,
- number of bins for histogram : 4.

It should be noticed that the majority of these parameters have a small influence, like block overlap, size of block, number of bins for histogram, and results are less than 5% better compared to non-optimal set of parameters. On the contrary, the normalization factor and size of cell have more influence on the result which are up to 10% better.

The rate obtained with the HOG descriptor is up to 90%. The errors (figure 6) still come from the mix between classes dog, horse and cow. Compared with the graph method, the HOG method uses texture information, which enables us to distinguish more easily some kinds of objects like apple and tomatœ.

HOG								
Prediction / True								
	402	0	0	0	0	0	0	8
	0	409	0	0	1	0	0	0
	0	5	323	0	38	43	1	0
	0	0	0	410	0	0	0	0
	0	2	30	0	354	24	0	0
	0	1	59	0	24	326	0	0
	0	0	0	0	0	0	410	0
	5	0	0	0	0	0	0	405

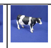
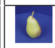
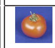
Graph								
P / T								
	317	0	0	6	0	0	0	87
	0	397	6	0	1	1	5	0
	0	21	310	0	36	43	0	0
	4	1	2	391	3	0	0	9
	0	6	36	0	313	55	0	0
	0	2	55	0	51	302	0	0
	0	3	2	0	0	0	405	0
	88	0	0	9	0	0	0	313

Fig. 5. Left :Results obtained for the HOG Kernel, with leave-one-object-out crossvalidation test. Good recognition rate : 90,1%. Right :Results obtained for the Graph Kernel, with leave-one-object-out crossvalidation test. Good recognition rate : 83,8%

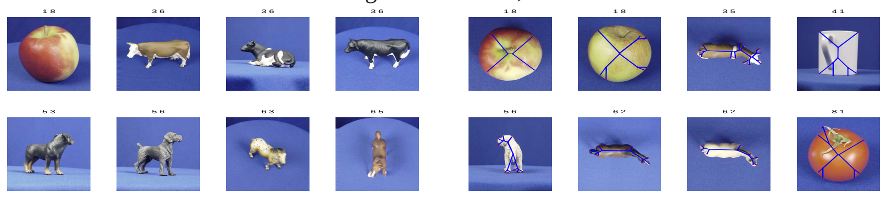


Fig. 6. Left : misclassified object with HOG method. Right : misclassified objects with graph method.

3.3 Kernel combination

We notice a certain complementarity between these methods, since some objects are badly discriminated by a method, but well recognized with the other.

We now describe the final test realized over the complete database. We combine kernels with a product : $K(x, y) = K_{HOG}(x, y) \times K_{graph}(x, y)$.

Each kernel K_{HOG} and K_{graph} was normalized previously : $k_n(x, y) = \frac{k(x,y)}{\sqrt{k(x,x) \times k(y,y)}}$

Prediction / True								
	403	0	0	0	0	0	0	7
	0	409	1	0	0	0	0	0
	0	1	345	0	37	27	0	0
	0	0	0	410	0	0	0	0
	0	1	31	0	353	25	0	0
	0	0	35	0	21	354	0	0
	0	0	0	0	0	0	410	0
	7	0	0	0	0	0	0	403

Fig. 7. Results obtained for the combined kernel, with leave-one-object-out crossvalidation test. Recognition rate : 94,1%.

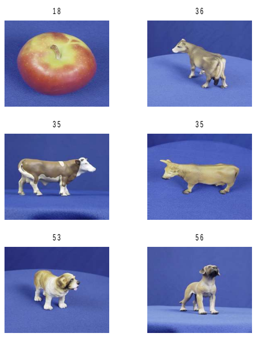


Fig. 8. Examples of miss-classified objects for complete test.

Figure 7 shows results for this test, which gives 94,1% of good recognition.

We can compare our results with other methods tested on the same database. In [12], Leibe and Schiele uses 7 classifiers in an optimal-decision tree. They obtained 93% of good recognition rate for a leave-one-object-out crossvalidation method. Results are comparable, but their method depends on the classes used for the classification, and may not be as efficient with additional classes. In our case, we can add other categories, without redefining completely our classification method.

In [18], results are given for a leave-one-image-out crossvalidation method for only 32 objects (4 from each class). This query method gives a recognition rate of 95%. In our case, we test our method over the complete database. Moreover, we remove completely an object, no image of the tested object was present in the learning set.

4 Conclusion

This paper presents a method for object categorization. The aim is to depict images thanks to labeled graphs and histograms of oriented gradients.

The first representation is a labeled graph, which enables us to describe the global shape of an object. The second representation is based on histograms of oriented gradients, which brings more information concerning the appearance of the object. A graph kernel is obtained by random walk on graphs, and we combine this kernel with a linear kernel obtained from HOG descriptors. We combined these kernels to use them with the SVM classifier.

The advantage of this method is to combine two kinds of representations to categorize an object. Using a classifier like SVM is well adapted to this combination which clearly improves recognition performance, compared when only one representation is used.

A complete test of this method on the ETH-80 database has proved that this approach is very promising, with 94% of good recognition rate for a leave-one-object-out crossvalidation test. This result proved that our method is efficient compared with existing methods [12, 18]. Our results could also be improved by combining additional object representations.

We have also some perspectives to improve this method. First, we would like to integrate histograms into the graph, and define a kernel for the vertices which deals with these histogramms. Another point resides in using multiple kernels. This point could help us to improve classification results.

References

1. Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 6, New York, NY, USA, 2004. ACM Press.
2. 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.
3. Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, 19(3-4):255–259, 1998.
4. S. Beucher C. Lantuejoul. On the use of the geodesic metric in image analysis. *Journal of miscrocopy*, 121(1):39–49, 1981.
5. N. Cristianini and J. Shawe-Taylor. *Introduction to Support Vector Machines*. Cambridge Univeristy Press, 2000.
6. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 886–893, INRIA Rhone-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.
7. J. Eichhorn and O. Chapelle. Object categorization with svm: kernels for local features. Technical report, MPIK, July 2004.
8. Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996.
9. T. Darrell K. Grauman. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the IEEE International Conference on Computer Vision, Beijing, China*, 2005.
10. H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieh International Conference on Machine Learning*, 2003.
11. Borgwardt K.M., Ong C.S., Schnauer S., Vishwanathan S.V.N., Smola A.J., and Kriegel H.-P. Protein function prediction via graph kernels. In *Intelligent Systems in Molecular Biology*, 2005.
12. B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Madison, WI, June 2003.
13. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
14. C. Di Ruberto. Recognition of shapes by attributed skeletal graphs. *Pattern Recognition*, 37(1):21–31, 2004.
15. D. Sharvit, J. Chan, H. Tek, and B. Kimia. Symmetry-based indexing of image databases, 1998.
16. A. Shashua, Y. Gdalyahu, and G. Hayon. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Proceedings of IEEE Intelligent Vehicles Symposium*, 2004.
17. V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
18. F. Demirci Y. Keselman, A. Shokoufandeh and S. Dickinson. Many-to-many feature matching using spherical coding of directed graphs. In *Proceedings, 8th European Conference on Computer Vision, Prague, Czech Republic*, pages 322–335, May 2004.