Object Recognition Using Multiresolution Trees

Monica Bianchini, Marco Maggini, and Lorenzo Sarti

DII - Università degli Studi di Siena
Via Roma, 56 - 53100 Siena - Italy
{monica, maggini, sarti}@dii.unisi.it

Abstract. This paper presents an object recognition method based on recursive neural networks (RNNs) and multiresolution trees (MRTs). MRTs are a novel hierarchical structure proposed to represent both the set of homogeneous regions in which images can be divided and the evolution of the segmentation process performed to determine such regions. Moreover, knowing the optimal number of regions that should be extracted from the images is not critical for the construction of MRTs, that are also invariant w.r.t. rotations and translations. A set of experiments was performed on a subset of the Caltech benchmark database, comparing the performances of the MRT and directed acyclic graph (DAG) representations. The results obtained by the proposed object detection technique are also very promising in comparison with other state-of-the-art approaches available in the literature.

1 Introduction

In graphical pattern recognition, data is represented as an arrangement of elements, that encodes both the properties of each element and the relations between them. Hence, patterns are modeled as labeled graphs where, in general, labels can be attached to nodes and edges.

In the last few years, a new connectionist model, that exploits the above definition of pattern, has been developed [1]. In fact, recursive neural networks (RNNs) have been devised to face one of the most challenging task in pattern recognition: realizing functions from graphs to vectors in an automatic and adaptive way. The original RNN model and its evolutions were recently applied to image processing tasks [2,3], obtaining interesting results. However, in order to exploit RNNs, a crucial role is played by the graphical representation of patterns, i.e. the way in which each image is represented by a graph. This choice affects the performances of the whole process.

In this paper we propose a new graphical representation of images based on multiresolution trees (MRTs), that are hierarchical data structures, somehow related to other representations used in the past to describe images. MRTs are generated during the segmentation of the images, like, for instance, quad–trees [4]. Other hierarchical structures, so as monotonic trees [5] or contour trees [6], can be exploited to describe the set of regions obtained at the end of the segmentation process, representing the inclusion relationship established among the

D.-Y. Yeung et al. (Eds.): SSPR&SPR 2006, LNCS 4109, pp. 331–339, 2006.

region boundaries. However, MRTs represent both the result of the segmentation, and the sequence of steps that produces the final set of regions. Moreover, the construction of MRTs does not depend on a priori knowledge on the number of regions needed to represent the images. Finally, MRTs combined with RNNs allow us to develop efficient object detection and object recognition systems.

This paper proposes an object recognition method and evaluates its performances on the Caltech benchmark dataset [7]. Two comparisons are presented: first, the images are represented by MRTs and directed acyclic graphs (DAGs) to assess which kind of structure allows to achieve better results; then the object recognition technique is compared against the state of the art methods based on vectorial representation and classical pattern recognition approaches [7,8,9,10].

The paper is organized as follows. In the next section, the RNN model is described, whereas in Section 3 the algorithm to extract MRTs is defined. Section 4 collects the experimental results and, finally, Section 5 draws some conclusions.

2 Recursive Neural Networks

Recursive neural networks were originally proposed to process directed positional acyclic graphs (DPAGs) [1,11]. More recently, an extended model, able to map rooted nonpositional graphs with labeled edges (DAGs-LE) into real vectors, was described [12]. This last RNN model is implemented based on a state transition function which has not a predefined number of arguments and which does not depend on the argument position. The different contribution of each child to the state of its parents depends on the label attached to the corresponding edges. At each node v, the total contribution $\overline{X}_v \in I\!R^n$ of the states of its children is computed as

$$\overline{X}_v = \sum_{i=1}^{od[v]} \Phi(X_{ch_i[v]}, L_{(v,ch_i[v])}, \theta_\Phi),$$

where $od[v]$ is the outdegree of the node v, i.e. the number of its children, $\Phi : I\!R^{(n+k)} \rightarrow I\!R^n$ is a function depending on a set of parameters θ_Φ, $X_{ch_i[v]} \in I\!R^n$ is the state of i-the child of node v, and $L_{(v,ch_i[v])} \in I\!R^k$ is the label attached to the edge $(v, ch_i[v])$. The state at the node v is then computed by another function $f : I\!R^{(n+m)} \rightarrow I\!R^n$ that combines the contribution of $\overline{X}_v$ with the node label $U_v \in I\!R^m$:

$$X_v = f(\overline{X}_v, U_v, \theta_f),$$

being f a parametric function depending on the parameters θ_f. Moreover, at the root node s, also an output function g is computed by another function g as

$$Y_s = g(X_s, \theta_g).$$

The functions Φ, f and g can be implemented by feedforward neural networks, in which the parameters θ_Φ, θ_f and θ_g are connection weights (see Figure 1(b)). As shown in Figure 1(c), the processing of an input graph is obtained by applying the recursive neural network (Figure 1(b)) recursively on the graph nodes,

starting from the leaves. This processing scheme yields an *unfolding network* whose structure depends on the topology of the input graph. The state X_v at each node encodes a representation of the subgraph rooted at v.

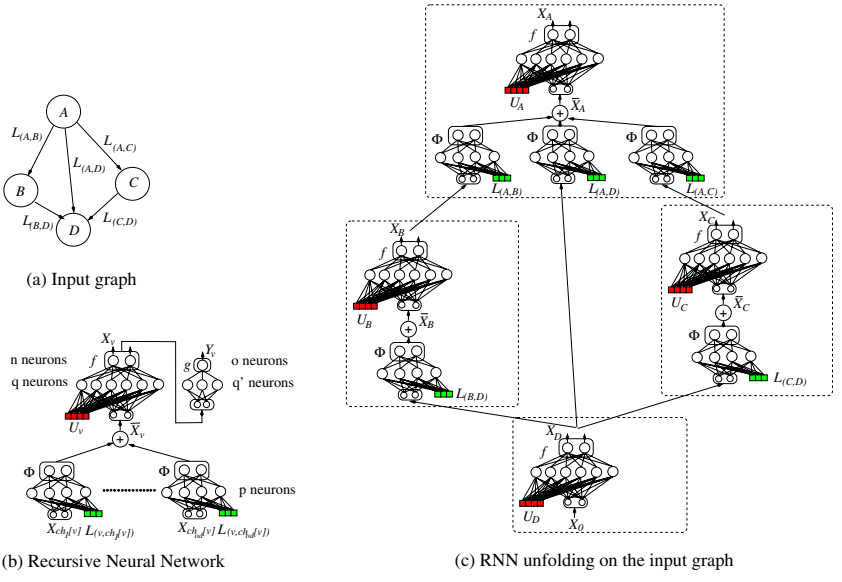

Fig. 1. The RNN processing scheme

RNNs can be trained to categorize images represented as graphs. Therefore an object recognition system can be developed based on a pool of RNNs where each network is specialized in recognizing a particular object class.

3 Multiresolution Trees

The neural network model proposed in Section 2 assumes to process structured data. Therefore a preprocessing phase that allows to represent images by graphs is needed, in order to exploit such model to perform any task on images (classification, localization or detection of objects, etc.). In the last few years, some image analysis systems based on RNNs and graph–based representation of images were proposed [12,13,14]. In these approaches, images are segmented to obtain a set of homogeneous regions that are subsequently represented by region adjacency graphs (RAGs). Then, since RNNs can process only directed structures, while RAGs are not, a further step is needed, to transform each RAG into a directed acyclic graph (DAG) [14] or into a set of trees [2]. The transformation from a RAG to a DAG can be obtained very efficiently, performing a breadth–first visit of the RAG, until each node is visited once. However, during this process each edge is transformed into a directed arc depending on arbitrary choices, in

particular the starting node for the breadth–first visit. In fact, the assignment of a direction to the adjacency relation changes the semantics of this relation, that is naturally undirected. On the contrary, the mapping from a DAG to a set of trees allows us to preserve the structural information, but it is particularly time consuming. As a matter of fact, for each node belonging to the RAG, a breadth–first visit must be performed to obtain a tree.

In this section, the representation of images based on MRTs is proposed. This kind of structure presents two advantages: first MRTs can be processed directly by RNNs without the need of any transformation; second they somehow reduce the dependence of the representation from the choice of the number of regions to be extracted in the segmentation process.

Nowadays, there is no a universal theory on image segmentation, and all the existing methods are, by nature, ad hoc. Moreover, the determination of the exact number of regions that should be extracted from a given image is a very challenging task, and can affect the performances of the system that takes the computed representation as input. MRTs allow us to ignore this information, since they collect, at each level, a distinct segmentation of the input image, obtained during a region growing procedure.

Since an MRT is generated during the segmentation, we need to describe how a set of homogeneous regions is extracted from an input image. First, a K–means clustering of the pixels belonging to the input image is performed. The clustering algorithm minimizes the Euclidean distance (defined in the chosen color space) of each pixel from its centroid. The number of clusters computed during this step is determined considering the average texture of the input image, since such a parameter provides useful information about the complexity of the depicted scene. It is worth noting that the number of extracted regions is greater than the number of clusters, since each cluster typically corresponds to several connected components. At the end of the K–means, a region growing step is carried out to reduce the number of regions, and, at the same time, to generate the MRT. The region growing procedure is sketched in Algorithm 1.1. Actually, the proposed method assumes to merge together groups of homogeneous regions, with the aim of bounding the maximum outdegree of the MRT, and, consequently, its depth. The algorithm takes as parameters the set of regions ($Rset$) obtained at the end of the K–means, and $maxGroupSize$, the maximum number of regions that can belong to a group. The goal of the algorithm is to reduce the number of regions, and to compute the set of nodes V, and the set of edges E, that define the MRT.

Initially, the set V is updated exploiting the function *createNode* that creates a new node and computes the node label (a set of visual and geometric features). These nodes represent the leaves of the MRT. Then, for each region r belonging to $Rset$, a region group g is created and stored in $Gset$, using the function *createGroup*. The region r represents the seed of g. Moreover, each region adjacent to r is added to the group that has r as its seed. When the number of adjacent regions is greater than $maxGroupSize$, the color distance between r and its adjacent regions is computed, and only the $maxGroupSize$ nearest adjacent regions are added to g. Note that, after this step, $\bigcup_{i=1}^{|Gset|} g_i = I$, being I the

whole image, but $\bigcap_{i=1}^{|Gset|} g_i \neq \emptyset$, and then $Gset$ must be rearranged with the aim of obtaining a partitioning of the image, such that $\bigcap_{i=1}^{|Gset|} g_i = \emptyset$.

Algorithm 1.1. `CreateMRT(`$Rset$`,`$maxGroupSize$`)`

```
{
   V←E←Gset←∅;
   for each r ∈ Rset
      V←V ⋃ {createNode(r)};
   while(|Rset|≥ maxGroupSize) {
      for each r ∈ Rset
         Gset←Gset ⋃ {createGroup(r)};
      Gset←cleanGroups(Gset,H());
      for each g ∈ Gset{
         newr←mergeGroup(g);
         Rset←Rset - members(g) ⋃ {newr};
         newn←createNode(newr);
         V←V ⋃ {newn};
         for each r ∈ members(g)
            E←E ⋃ {(newn,getNodeAssociatedWith(r))};
      }
      Gset←∅;
   }
   root←createNode(⋃_{i=1}^{|Rset|} r_i);
   V←V ⋃ {root};
   for each r ∈ Rset
      E←E ⋃ {(root,getNodeAssociatedWith(r))};
}
```

This phase is performed by the function *cleangroups*, that is described by Algorithm 1.2. This function takes as input $Gset$ and a homogeneity function H(), that is used to compute the degree of similarity of the regions that belong to a given group. The function H() is a parameter of the segmentation algorithm and must be chosen such that a high value of H(g) corresponds to a high probability of merging g. First, the groups are sorted in descending order w.r.t. their homogeneity. As a matter of fact, if a region r belongs to a group having high homogeneity, the group that has r as its seed must be removed from $Gset$. Considering that *adjacent*(g) collects the set of regions that belong to g, except for its seed, and the function *getGroupBySeed*(r) returns the group that has r as its seed, the first iterative block of the algorithm performs the following steps. For each group g, the set *adjacent*(g) is analyzed. For each region r in *adjacent*(g), if r is the seed of a group that has lower homogeneity than g, then the group is removed from $Gset$, otherwise r is removed from *adjacent*(g). However, the goal of obtaining a partitioning of the original image is still not reached, because the algorithm removed only whole groups, but some regions can belong to more than one group. Then, the function *cleanGroups* scans again all the members of the groups looking for regions that belong to two or more groups, and, if they exist, removes them from the groups that have lower homogeneity.

Algorithm 1.2. `cleanGroups(`$Gset$`,H())`

```
{
   Gset←sort(Gset,H());
   for each g ∈ Gset
      for each r ∈ adjacent(g)
         if (H(g) ≥ H(getGroupBySeed(r)))
            Gset←Gset - getGroupBySeed(r);
         else
            adjacent(g)←adjacent(g)-r;
   for each g^1 ∈ Gset
      for each g^2 ∈ Gset and g^1 ≠ g^2
         for each r^1 ∈ adjacent(g^1)
            for each r^2 ∈ adjacent(g^2)
               if (r^1 = r^2)
                  if (H(g^1) ≥ H(g^2))
                        adjacent(g^2)←adjacent(g^2)-r^2;
                     else
                        adjacent(g^1)←adjacent(g^1)-r^1;
}
```

At the end of Algorithm 1.2, $Gset$ collects a partitioning of the whole image and Algorithm 1.1 merges together the regions that belong to the same groups. The set of regions $Rset$ is updated considering the new regions, and a new level of the MRT is created. For each new region $newr$, a new node $newn$ is created. Finally, $newn$ is linked to the nodes that are associated with the regions merged to obtain $newr$.

The main loop of Algorithm 1.1 is repeated until the cardinality of $Rset$ is greater than $maxgroupSize$. At the end of the main loop, the root node is added to the MRT and linked to all the nodes that correspond to the regions collected in $Rset$. All the nodes in the MRT have a label that describes visual and geometric properties of the associated regions, and also each edge has a label that collects some features regarding the merging process.

The proposed technique presents some advantages. First, segmentation methods based on region growing generally produce results that depend on the order in which the regions are selected during the merging process, and, as a side effect, the final set of regions is not invariant w.r.t. rotations, translations, and other transformations of the input image. Instead, the region growing method proposed in Algorithm 1.1 is independent from rotations and translations, since the regions are selected considering the order defined by the homogeneity function.

The main advantage of MRTs consists in being independent of the number of regions needed to represent the image. Actually, a distinct segmentation, with a different number of regions, is stored in each level of the tree. The key idea consists in exploiting the capabilities of adaptive models, like RNNs, to discover at which level the best segmentation is stored. Moreover, MRTs are invariant w.r.t. rotations and translations of the images and do not describe directly the topological arrangement of the regions, that however can be inferred considering both the geometric features associated to each node (for instance, the

coordinates of the bounding box of each region can be stored in the node label) and the MRT structure. Finally, the region growing is performed merging together groups of regions instead of pairs, to avoid the generation of binary trees. As a matter of fact, the generation of binary trees implies the generation of deeper structures, and RNNs suffer in processing such structures, due to the "long-term dependency" phenomenon, that was originally investigated for recurrent neural networks [15].

4 Experimental Results

In order to evaluate the capability of MRTs to represent the contents of images, some experiments were carried out, addressing an object recognition problem. The experiments were performed using the Caltech Database[1], since it represents a popular benchmark for object class recognition. The Caltech database collects six classes of objects: motorbikes, airplanes, faces, cars (side view), cars (rear view), and spotted cats. Our experiments were focused on a subset of the dataset, that consists only of images from the motorbikes, airplanes, faces, and cars (rear view) classes.

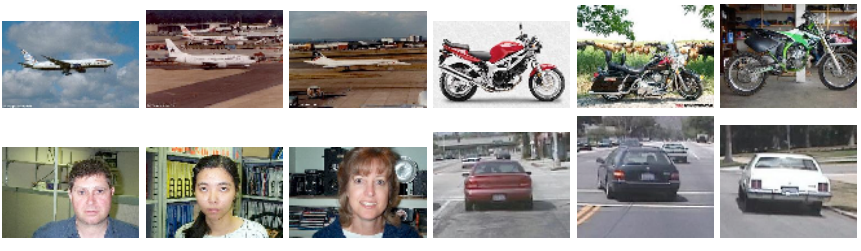

Fig. 2. Samples of images from the Caltech database

For each class, three datasets were created: training, test, and cross–validation sets. The training and test sets collect 96 images each, while 48 images belong to the cross–validation set. For each set, half images correspond to positive examples, while the other images are examples of the negative class (i.e. images from the other classes). All the images were selected randomly from the Caltech database and segmented in order to obtain both MRTs and DAGs.

MRTs were obtained using Algorithm 1.1 and a maximum group dimension equal to 7. The homogeneity function, that affects directly the segmentation and the MRT generation, was chosen to be $H(g) = \frac{1}{\sigma^2}$, being σ^2 the color variance of the group g in the image color space. The node labels in the MRTs collect geometric and visual information, like area, perimeter, barycenter coordinates, momentum, etc., while color distance, barycenter distance, and the ratio obtained dividing the area of the child region by the area of the parent region, are

[1] The Caltech database is available at http://www.robots.ox.ac.uk/∼vgg/data3.html

Table 1. Results obtained representing images by DAGs or MRTs. The second column shows the number of state neurons of the recursive network. The results are reported using the average ROC equal error rate, obtained performing ten learning runs.

	State neurons	Airplanes	Motorbikes	Faces	Cars(rear)
M	5	100	97.91	100	92.7
R	7	95.83	92.7	100	90.6
T	10	94.79	92.7	100	92.7
D	5	75	69.79	73.54	76.04
A	7	75	70.83	70.41	77
G	10	75	68.75	71.67	78.12

Table 2. Best results obtained using MRTs compared against results available in the literature. The results are reported using the average ROC equal error rate.

	RNNs and MRTs	Zhang [8]	Fergus [7]	Opelt [9]	Thureson [10]
Motorbikes	97.91	99	92.5	92.2	93.2
Airplanes	100	98.3	90.2	88.9	83.8
Faces	100	99.7	96.4	93.5	83.1

used as edge labels. With respect to the generation of DAGs, a modified version of Algorithm 1.1 was exploited. The instructions related to the MRT generation were removed, and the main loop was halted when the number of regions become smaller than the parameter that was used to determine the number of initial K–means clusters. Finally, at the end of the region growing phase, the DAG was generated following the steps described in [14]. The generated MRTs collect 400 nodes and are composed by 8 levels, on average, whereas DAGs contain about 70 nodes. For each class, several RNN classifiers were trained, using both MRTs and DAGs, in order to determine the best network architecture. The transition function f is realized by an MLP with $n+1$ hidden units (using the hyperbolic tangent as output function) and n linear outputs, being n the number of state neurons. The function Φ, that combines the state of each child with the corresponding edge label, is implemented by an MLP with a layer of $n+1$ sigmoid hidden units and n linear outputs. Finally, the output network g is an MLP with n inputs, $n-2$ sigmoid hidden units and one sigmoid output. The obtained results are reported in Table 1. Even if the main goal of the experiments is the comparison between MRTs and DAGs, Table 2 collects also a comparison between the presented object recognition system and other methods known in the literature, that were evaluated using the same benchmark database. The method based on MRTs definitely outperforms the DAG–based representation. Moreover, the comparison with the other methods reported in Table 2 shows very promising results, even if our experiments were performed considering only a subset of the Caltech database.

5 Conclusions

In this paper, we proposed a new hierarchical representation of images, based on multiresolution trees. An MRT represents, in a unique structure, the result

of image segmentation, and the sequence of steps that produces the final set of regions. The performances of the proposed representation technique were evaluated addressing an object recognition task. A method based on RNNs and MRTs was proposed and evaluated on the Caltech benchmark dataset, showing promising results.

References

1. Frasconi, P., Gori, M., Sperduti, A.: A general framework for adaptive processing of data structures. IEEE Transactions on Neural Networks **9** (1998) 768–786
2. Bianchini, M., Maggini, M., Sarti, L., Scarselli, F.: Recursive neural networks learn to localize faces. Patter Recognition Letters (2005) 1885–1895
3. Bianchini, M., Maggini, M., Sarti, L., Scarselli, F.: Recursive neural networks for object detection. In: Proceedings of IEEE IJCNN. (2004) 1911–1915
4. Hunter, G.M., Steiglitz, K.: Operations on images using quadtrees. IEEE Transactions PAMI **1** (1979) 145–153
5. Song, Y., Zhang, A.: Monotonic tree. In: Proceedings of the 10th Intl. Conf.on Discrete Geometry for Computer Imagery, Bordeaux – France (2002)
6. Roubal, J., Peucker, T.: Automated contour labeling and the contour tree. In: Proceedings of AUTO-CARTO 7. (1985) 472–481
7. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale–invariant learning. In: Proceedings of IEEE CVPR. (2003) 264–271
8. Zhang, W., Yu, B., Zelinsky, G., Samaras, D.: Object class recognition using multiple layer boosting with heterogeneous features. In: Proceedings of CVPR. Volume 2. (2005) 323–330
9. Opelt, A., Fussenegger, M., Pinz, A., Auer, A.: Weak hypoteses and boosting for object detection and recognition. In: Proceedings of ECCV. Volume 2. (2004) 71–84
10. Thureson, J., Carlsson, S.: Appearance based qualitative image description for object class recognition. In: Proceedings of ECCV. Volume 2. (2004) 518–529
11. Küchler, A., Goller, C.: Inductive learning in symbolic domains using structure–driven recurrent neural networks. In Görz, G., Hölldobler, S., eds.: Advances in Artificial Intelligence. Springer, Berlin (1996) 183–197
12. Bianchini, M., Maggini, M., Sarti, L., Scarselli, F.: Recursive neural networks for processing graphs with labelled edges: Theory and applications. Neural Networks (2005) 1040–1050
13. de Mauro, C., Diligenti, M., Gori, M., Maggini, M.: Similarity learning for graph based image representation. Pattern Recognition Letters **24** (2003) 1115–1122
14. Gori, M., Maggini, M., Sarti, L.: A recursive neural network model for processing directed acyclic graphs with labeled edges. In: Proceedings of IEEE IJCNN. (2003) 1351–1355
15. Bengio, Y., Frasconi, P., Simard, P.: Learning long–term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks **5** (1994) 157–166