Face Spotting in Color Images using Recursive Neural Networks

M. Bianchini, P. Mazzoni, L. Sarti and F. Scarselli
Dipartimento di Ingegneria dell’Informazione
Università degli Studi di Siena
Via Roma, 56 — 53100 Siena (ITALY)
Email:{monica,sarti,franco}@ing.unisi.it
mazzopaol@lucy.dii.unisi.it

Abstract

Images containing faces are fundamental for vision–based human computer interaction. Several problems concerning the presence of faces in images, including face recognition, pose estimation, and expression recognition have focused the research efforts in the last few years. Moreover, those problems are usually solved having assumed that the face was previously localized, often via heuristics based on prototypes of the whole face or significant details. Therefore, each fully automated system, which is able to analyze the information contained in images of faces, requires an efficient face localization method. In this paper, we propose a novel approach to the solution of the face localization problem using recursive neural networks. In particular, the proposed approach assumes a graph–based representation of images that combines structural and sub–symbolic visual features. Each image is represented by a forest of trees. Such trees are then processed by recursive neural networks, in order to establish the eventual presence and the position of faces inside the image. Some preliminary experiments on snapshots from video sequences are reported, showing very promising results.

1 Introduction

In several applications, ranging from static matching of controlled photographs to video surveillance, the problem of face recognition plays a crucial role. Face recognition compares a given image, containing one or more faces, against a database, and reports eventual matches. The localization of the face is a preliminary step, which is required in order to be able to recognize the face. Given an arbitrary image, the goal of face localization consists of spotting whether or not a face is present, also returning its eventual location.

Frequently, in order to solve the face recognition problem, the positions of faces are supposed to be known, since even the face localization problem is a very challenging task. In fact, the appearance of a face in an image is unsettled with respect to location, orientation and pose [1]. Face localization methods can be classified into four main categories: knowledge–based, feature invariant, template matching, and appearance–based methods. In knowledge–based methods, human knowledge about faces is encoded by some rules. The localized faces must respect the predefined rules [2]. The aim of feature invariant methods is to discover a set of features that are invariant with respect to pose, viewpoint, and lighting conditions. Those features are then used to localize the faces [3]. Template matching methods store several patterns of a face and describe each pattern by facial features. The correlation among an input image and the stored patterns is computed for detecting new faces [4]. Finally, in appearance–based methods, the templates describing the faces are learned by examples. Several machine learning techniques are used, from static neural networks [5] and Hidden Markov Models [6], to Support Vector Machines [7].

In this paper, we present a new appearance–based method that exploits recursive neural networks to learn templates. The novelty of the approach consists of using forests of trees to represent images. The trees are processed by a recursive network, a particular model that extends neural network processing to structured data [8]. Since, the graphical representation is invariant under image translations, rotations and scaling, the method does not suffer from those image transformations. Moreover, due to the intrinsic parallelism of recursive networks, the technique can be implemented very efficiently.

The paper is organized as follows. In Section 2 and 3 the tree–based representation of images and the recursive neural network model are described, respectively. Section 4 collects some preliminary experimental results and, finally, in Section 5 some conclusions are drawn.
Representing Images with Forests of Trees

In order to describe how a face can be localized in a complex image, we have to detail the two fundamental preprocessing phases which allow us, first, to extract an informative graph-based representation of the image, and then to transform the Region Adjacency Graph (RAG) representing the image into a forest of recursive-equivalent trees, with which the recursive neural network can be fed.

2.1 From images to graphs

The first step needed to perform the transformation from images to graphs is the image segmentation. Our segmentation method is based on color information. The effectiveness of using a color-based segmentation approach was already shown in [9] and [10]. In our approach, images are encoded using the HSV (Hue, Saturation, Value) color space. The HSV color space can be described by a hexcone shape (see Fig. 1(a)). The vertices of the hexagon represent Red, Yellow, Green, Cyan, Blue and Magenta. Adjacent vertices are separated by an hue angle \( H \) of 60 degrees. The purity of colors is represented by the saturation \( S \) that can vary from 0 to 1.

Moreover, the value \( V \) describes the darkness of a color and varies from 0 to 1. The bottom vertex of the hexcone is defined by \( V = 0 \) and the top level by \( V = 1 \). Each color is described by a triple \( (h, s, v) \) representing a point that belongs to the hexcone.

The face segmentation in the HSV space can be performed using appropriately defined ranges of hue and saturation, which describe the human skin color [11]. This is equivalent to cut off a sector of the hexagon obtaining fixing a value of \( V \) (see Fig. 1(b)). Using the defined ranges, each image is binarized and it is split into homogeneous regions (see Fig. 2). Very small regions are removed and their pixels are inserted into the adjacent regions.

The structural information related to the spatial relationships between pairs of regions can be coded by a Region Adjacency Graph (RAG), using the method proposed in [12]. Two connected regions \( R_1, R_2 \) are adjacent if, for each pixel \( a \in R_1 \) and \( b \in R_2 \), there exists a path connecting \( a \) and \( b \), entirely lying into \( R_1 \cup R_2 \). The RAG is extracted from the segmented image by:

- Associating a node to each region. Each node is labeled with a real vector of features extracted from the original image (area, barycenter coordinates, minimum bounding box, etc.);
- Linking the nodes associated to adjacent regions. The edges are not oriented.

Therefore, a RAG takes into account both the topological arrangement of the regions and the sub-symbolic visual information. Notice also that the RAG connectivity is invariant under translations and rotations, which is a useful property for the goal of locating faces. Moreover, in order to set up a learning environment, a target equal to 1 is attached to each node of RAGs that corresponds to faces, whereas a 0 is attached otherwise. For example, in Fig. 2, nodes with a cross inside denote parts of the face and have target 1, whereas the other nodes have target 0.

2.2 From RAGs to trees

Since recursive neural networks can process only Directed Acyclic Graphs (DAGs), each RAG must be transformed into some directed structure. In the following, we describe a method to carry out this task. The procedure takes a RAG \( R \), along with a selected node \( n \), as input and
produces a tree $T$ having $n$ as its root. The method must be repeated for each node $n$ of the RAG or, more practically, for a random set of the nodes. It can be proved that the forest of trees built from $R$ contains the same information as $R$ [13].

The first step of the procedure is a preprocessing phase that transforms $R$ into a directed graph $G$, by assuming that a couple of edges is attached to each undirected one, thus preserving the duplex information exchange (see Fig. 3(a)). Then, $G$ is unfolded by the following algorithm:

1. Insert a copy of $n$ into $T$;
2. Visit $G$, starting from $n$, using a breadth–first strategy: for each visited node $v$, insert a copy of $v$ into $T$ and link $v$ to its parents;
3. Repeat step 2 until a predefined stop criterion is satisfied and, however, until all arcs have been visited at least one time.

According to [13], the above unfolding strategy produces a recursive–equivalent tree that holds the same information contained in $R$. With respect to the stop criterion chosen, if the breadth–first visit is stopped when all the arcs have been visited one time, the minimal recursive–equivalent tree is obtained (Minimal Unfolding, Fig. 3(b)). However, other strategies are possible. For example, each edge can be visited one time, then step 2 is repeated until a stochastic variable $x$ becomes true (Mixed Unfolding, Fig. 3(c)). Otherwise, we can replace the breadth–first visit strategy of step 2 with a random visit of the graph (Random Unfolding, Fig. 3(d)).

3 The Recursive Neural Network Model

Recursive neural networks are a generalization of recurrent networks conceived to deal with DAGs and have been already used in some applications [8]. In order to process a graph $G$, the recursive network is unfolded through the graph structure, yielding an encoding network (see Fig. 4). Each graph $G$ must have a supersource, i.e. a node $s$ having no incoming edges, such that any other node in $G$ can be reached by a directed path starting from $s$. If $G$ does not possess a supersource, an extra node $s$ with a minimal number of outgoing edges can be attached to it, such that $s$ is a supersource [14]. The presence of $s$ is related to the processing schema which will be subsequently described.

At each node $v$ of the graph, the state $X_v \in \mathbb{R}^n$ is computed by a feedforward network, which implements the state transition function $f$, using the node label $U_v \in \mathbb{R}^n$ and the states of its children,

$$X_v = f(X_{ch[v]}, U_v, \theta_f),$$

being

$$X_{ch[v]} = [X'_{ch_1[v]}, \ldots, X'_{ch_n[v]}], \quad o = \max_{v \in V} \{\text{od}[v]\}$$

with $\text{od}[v]$ number of the children of $v$. Eq. (1) is applied recursively by setting $X_{ch[v]}$ equal to the frontier state $X_0$, if node $v$ lacks of its $i$–th child.

At the supersource, an output function is evaluated by another feedforward network (output network) using the corresponding state vector $X_s$,

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

The parameters $\theta_f$ and $\theta_g$ are the connection weights of the encoding and the output network, respectively. The
parametric representations of \( f \) and \( g \) can be implemented by a variety of neural network models. In the case of a two–layer perceptron, with sigmoidal activation functions in the hidden units and linear activation functions in the output units, the state is calculated according to

\[
X_v = V \cdot \sigma \left( \sum_{k=1}^{n} A_k \cdot X_{ch} + B \cdot U_v + C \right) + D,
\]

where \( \sigma \) is a vectorial sigmoidal function and \( \theta_f \) collects \( A_k \in \mathbb{R}^{m \times n}, k = 1, \ldots, n, B \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^n, D \in \mathbb{R}^m, \) and \( V \in \mathbb{R}^{m \times q} \). Here, \( m \) is the dimension of the label space, \( n \) the dimension of the state space, and \( q \) represents the number of hidden neurons. A similar equation holds for the output network at the supersource:

\[
Y_s = W \cdot \sigma \left( E \cdot X_s + F \right) + G,
\]

where \( \theta_g \) collects \( E \in \mathbb{R}^{q \times m}, F \in \mathbb{R}^q, G \in \mathbb{R}^r, W \in \mathbb{R}^{r \times q}, \) being \( q' \) the number of hidden units in the output network and \( r \) the number of output units. Therefore, a recursive neural network implements a function \( h : \mathbb{R}^{q'} \to \mathbb{R}^r \), where \( h(G) = Y_s \). Formally, \( h = g \circ f \), where \( f(G) = X_s \) denotes the process that takes a graph and returns the state at the supersource.

Recursive neural networks can be trained using BackPropagation Through Structure (BPTS) [14]. BPTS is essentially an extension of BackPropagation and BackPropagagtion Through Time. BPTS allows us to compute efficiently the gradient of a cost function with respect to the parameters \( \theta_f \) and \( \theta_g \). Basically, the computation consists of an error backpropagation through the output and the encoding network, obtained unfolding the recursive neural network on the graph structure.

4 Experimental Results

In order to evaluate the effectiveness of our approach, a preliminary experimentation was performed. Faces were searched in a dataset containing 201 images and 238 faces (each image contains at least one face), acquired by TV video sequences. The appearance of faces in the images is unsettled with respect to orientation, light conditions, dimensions, etc. (see Fig. 5).

The images were divided in three sets: training set, cross–validation set, and test set. Both the training and the cross–validation set contain 50 images, whereas 101 images constitute the test set.

Each image belonging to the three sets was segmented obtaining a RAG. Subsequently each RAG was unfolded using the three methods described in Section 2.2. Finally, the recursive neural network was trained to predict whether the nodes in the trees belong to faces or not. The obtained results are evaluated and compared using accuracy and detection rates. The accuracy rate is obtained dividing the number of nodes correctly classified by the number of nodes in the trees. Instead, the detection rate is obtained dividing the number of faces correctly localized by the number of faces in the images. Several recursive neural networks were trained with the aim of determining the best network architecture. The network architectures were varied both with respect to the number of layers and the number of state and hidden neurons. With regard to the network architecture, one–layer and two-layer networks were trained (see Fig. 6). The results obtained are reported in Table 1 and show how a very simple network architecture is the most promising one.

5 Conclusions

In this paper we have proposed a new method to localize faces in images. The images are segmented using the peculiarity of HSV color space and then are represented by forests of trees. Exploiting this kind of image representation, the approach is invariant under image translations and rotations. The graphical representation of images are then processed by a recursive neural network to determine if the nodes of the trees belong to a part of a face or not. The preliminary results have shown the method to be promising. Moreover, the technique can be implemented very efficiently exploiting the intrinsic parallelism of neural networks.
Figure 6. General (a) one–layer, and (b) two–layer recursive neural network architecture.

References


