Learning Taxonomies in Large Image Databases

Lokesh Setia and Hans Burkhardt
Chair for Pattern Recognition and Image Processing
Albert-Ludwigs-University Freiburg
79110 Freiburg im Breisgau, Germany
{setia, burkhardt} @informatik.uni-freiburg.de

ABSTRACT

Growing image collections have created a need for effective retrieval mechanisms. Although content-based image retrieval systems have made huge strides in the last decade, they often are not sufficient by themselves. Many databases, such as those at *Flickr* are augmented by keywords supplied by its users. A big stumbling block however lies in the fact that many keywords are actually similar or occur in common combinations which is not captured by the linear metadata system employed in the databases. This paper proposes a novel algorithm to learn a visual taxonomy for an image database, given only a set of labels and a set of extracted feature vectors for each image. The taxonomy tree could be used to enhance the user search experience in several ways. Encouraging results are reported with experiments performed on a subset of the well known Corel Database.

Categories and Subject Descriptors: H.3 [Information Storage and Retrieval]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; I.5 [Pattern Recognition]: I.5.3 Clustering; I.5.4 Applications;

General Terms: Algorithms, Design

Keywords: Taxonomy, Image Retrieval, Clustering

1. INTRODUCTION

Image retrieval has received its fair share of attention since the past decade. But it would not be an exaggeration to say that the demand outsizes what the research can currently supply. The core of the problem lies in the so-called *semantic gap* [5], or the inability of low-level features to capture effectively the high-level semantic meaning present in the image. To a limited extent, this problem can be alleviated using *relevance feedback* [10]. However, there are other practical issues to be resolved as well. The most common paradigm used in the image retrieval research community is the *query-by-example* paradigm, where the aim is to find best matches to a query image supplied by the user. In practice, however, a good query image might not be easy to find. Another possibility is to perform automatic annotation of the image collection [2][4], so as to allow a keyword search like in text search engines. This automatic annotation can be performed in addition to any manual labelling that might already exist for the database. An important factor to keep in mind is that manual labelling is often subjective, and in the absence of a small and fixed vocabulary often different users would arrive at different labellings.

In this work we propose building a visual taxonomy for a labelled database. The taxonomy is built offline using a fully automated method which needs just a set of labelled images and a set of features extracted from each image. Obviously, the features play an important part in the method, and better results can be expected by incorporating more discriminative features for the categories in hand. Still, in order to demonstrate the effectiveness of our visual taxonomy generation algorithm, we use a very simple set of visual features as we describe in Section 4. The constructed taxonomy tree can have a variety of uses. It can be used to automatically weed out redundant labels in a large database. It can guide the user by offering images not just from the query keyword, but also from the keywords close to it. Since the taxonomy is built offline, this step does not introduce any extra delay. Furthermore, by offering the user direct access to the taxonomy, it can allow the user to choose the right keywords quickly and effectively, thus shortening the average search time. Finally, the taxonomy can be used as a data mining tool, providing insights to the patterns and correlations existing naturally in the database.

2. ALGORITHM

The main stages of the proposed algorithm for taxonomy generation are listed in Algorithm 1, and would be elaborated in the following section.

2.1 Estimating Closeness of Labels

We determine the closeness of labels in a novel way. Rather than having an absolute measure of closeness of two labels, we estimate it by taking their relationships to other labels in the database into account. To give an example by analogy on a semantic level: to ask if a *spoon* is similar to a *fork* is a rather abstract question, but when given the fact that the other objects are *industrial tools*, one can say that the two are similar because they distinguish themselves using the same property (*usage*). We wish to model this mathematically in the following. We propose two methods, the first being more intuitive, but with the disadvantage that it is not kernelizable, i.e. works only in the original input

SIGIR Workshop on Multimedia Information Retrieval, 2007

Algorithm 1 Taxonomy Generation Algorithm

1: Given an image database, labeled with L labels (keywords) $l_i, i = 1, \ldots, L$ along with a set of images for each label.
2: Extract suitable features for each image. We assume that the set of features for an image is represented by a single feature vector $\mathbf{x}$ (though we would later propose a modification such that this is not strictly required).
3: For all pairs of labels, estimate the closeness of the label pair (see Section 2.1).
4: Perform agglomerative clustering to generate a hierarchy tree for the labels.
5: Postprocessing: Pruning/joining of the nodes of the hierarchical tree to relax the binary tree constraint.

feature space or with the computation cost of an explicit feature mapping. In the two sections below, we are estimating the distance between two labels l_p and l_q, using other labels for intermediate calculations. We define D^k_{pq} as the distance or dissimilarity between l_p and l_q using the intermediate label class l_k. The overall distance D_{pq} can thus be defined over all intermediate labels excluding the labels in question:

$$D_{pq} = \sum_{\substack{k=1,\\ k\neq p,q}}^{L} D^k_{pq}$$

2.1.1 Using Feature Similarity

This method works by recognizing features essential for classifying l_p against l_k, and the ones for classifying l_q against l_k. The commonness of these sets of features is an indication of the similarity of the classes l_p and l_q. One simple way to get an estimate of the feature importance is to train a linear classifier (e.g. a linear SVM), and use the absolute value of the normal vector coefficients as the measure of the relative feature importance. The principle is illustrated in Figure 1, where the classes l_p and l_q are deemed similar on virtue of using the same feature x_2 for classification against the current intermediate class l_k.

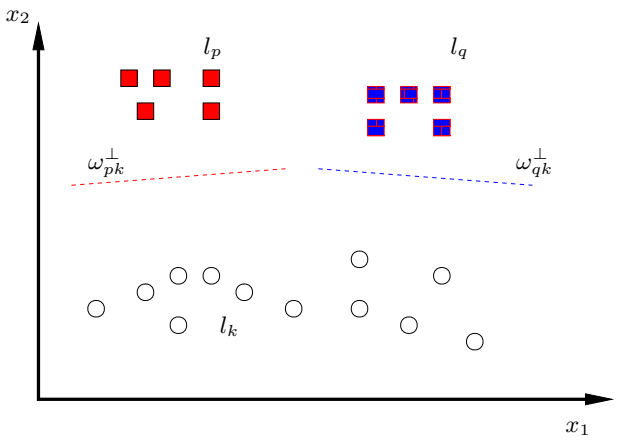

Figure 1: Using feature similarity to compute dissimilarity between classes

Let the linear classification boundary between classes l_p and l_k be given through

$$\langle \omega_{pk}, \mathbf{x} \rangle + b_{pk} = 0$$

and the boundary between l_q and l_k through

$$\langle \omega_{qk}, \mathbf{x} \rangle + b_{qk} = 0$$

We assume, without any loss of generality that the hyperplane normal vectors ω_{pk} and ω_{qk} are already normalised with respect to their L_2 norm. The distance D^k_{pq} can thus be defined as

$$D^k_{pq} = \sum_{f=1}^{F} \big|\, |\omega^f_{pk}| - |\omega^f_{qk}| \,\big|$$

where F is the number of features, or the dimensionality of the norm vectors. In words, this is the L_1 distance between the *magnitudes* of the normalised hyperplane normal vectors. Although this measure gives good results, it is not kernelizable as it uses properties other than the dot product. Thus, in cases where the hyperplane yielded by the linear classifer was very crude, this may give poor results. The alternative method proposed in the next section is easily kernalizable, and thus may be preferable in such cases.

2.1.2 Using Hyperplane Similarity

Here we use the dot products between the SVM hyperplanes as the similarity measure, i.e.

$$D^k_{pq} = 1 - \frac{\langle \omega_{pk}, \omega_{qk} \rangle}{\|\omega_{pk}\| \|\omega_{pk}\|}$$

The hyperplane is expressed as a sum of transformed training vectors, i.e.: $\omega_{pk} = \sum_i \alpha_i \cdot \mathbf{\Phi}(\mathbf{x}_i)$ and $\omega_{qk} = \sum_j \beta_j \cdot \mathbf{\Phi}(\mathbf{y}_j)$. $\mathbf{\Phi}$ is a (typically non-linear) transformation to a new feature space, where the data might be better seperable using a linear classifier. The SVM algorithm needs access only to dot products between feature vectors $\langle \mathbf{\Phi}(\mathbf{x}_1), \mathbf{\Phi}(\mathbf{x}_2) \rangle$ which might be directly computable in the original feature space [9], thus sparing the computation cost of the transformation $\mathbf{\Phi}$. In some case, it might not even be possible to compute $\mathbf{\Phi}(\mathbf{x})$. The dot product in the transformed space is known as kernel evaluation and would be denoted by:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \langle \mathbf{\Phi}(\mathbf{x}_1), \mathbf{\Phi}(\mathbf{x}_2) \rangle$$

Thus, we can simplify:

$$\begin{aligned}
D^k_{pq} &= 1 - \frac{\langle \omega_{pk}, \omega_{qk} \rangle}{\|\omega_{pk}\| \|\omega_{pk}\|} \\
&= 1 - \frac{\langle \sum_i \alpha_i \mathbf{\Phi}(\mathbf{x}_i), \sum_j \beta_j \mathbf{\Phi}(\mathbf{y}_j) \rangle}{\langle \sum_i \alpha_i \mathbf{\Phi}(\mathbf{x}_i), \sum_j \alpha_j \mathbf{\Phi}(\mathbf{x}_j) \rangle^{\frac{1}{2}} \langle \sum_i \beta_i \mathbf{\Phi}(\mathbf{y}_i), \sum_j \beta_j \mathbf{\Phi}(\mathbf{y}_j) \rangle^{\frac{1}{2}}} \\
&= 1 - \frac{\sum_i \sum_j \alpha_i \beta_j \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{y}_j) \rangle}{(\sum_i \sum_j \alpha_i \alpha_j \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle)^{\frac{1}{2}} (\sum_i \sum_j \beta_i \beta_j \langle \mathbf{\Phi}(\mathbf{y}_i), \mathbf{\Phi}(\mathbf{y}_j) \rangle)^{\frac{1}{2}}} \\
&= 1 - \frac{\sum_i \sum_j \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{y}_j)}{(\sum_i \sum_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j))^{\frac{1}{2}} (\sum_i \sum_j \beta_i \beta_j K(\mathbf{y}_i, \mathbf{y}_j))^{\frac{1}{2}}}
\end{aligned}$$

which can be computed using kernel evaluations between the support vectors of the two hyperplanes. It should be noted that the bias terms b_{pk} and b_{qk} do not play a role during the calculation of D^k_{pq}, and it is not clear if it should.

After all, the role of the bias term is only to shift the hyperplane in the direction of the normal vector, thus making the first class more preferable (for positive bias), or the second class (for negative bias). At this stage, it is interesting to note the fact that one would expect a bias value of 0 for a *neutral classifier*. This statement is in sense of the test points $\mathbf{x}$ which are very dissimilar to all support vectors, i.e. $K(\mathbf{x}, \mathbf{x}_i) \approx 0, \forall i = 1 \ldots N_{SV}$. A classifier with a non-zero bias would classify all such points to exactly one of the classes which may not be the desired behaviour, especially for distance based kernels.

3. CLUSTERING OF LABELS

We use *agglomerative clustering* [1] to construct a hierarchical tree of labels, once the pair-wise dissimilarities between labels have been computed. This is a *bottom-up* algorithm, where initially all labels are disjunct clusters. The algorithm proceeds iteratively. At each step, the pair of clusters having the smallest distance is combined to form a single cluster entity. The process continues until just a single cluster remains which is the ancestor cluster for all previous clusters. An important design decision is the choice of the distance function between two clusters. Since each cluster is a collection of one or more labels, for which we already have computed pairwise distances, we can define the distance between two clusters as the mean distance between their constituent labels. This is the version used in our experiments. Another option, though more time-consuming is to view a newly-formed cluster as a new label (a *superlabel*), and re-compute distances as in the previous section. A hierarchical binary tree obtained by this method can be viewed as a dendrogram [1] which we would also use to display our results. The y-axis shows the distance between two cluster nodes when they are combined. The smaller the distance (relatively), the more confident the system is about the pairing.

4. EXPERIMENTS AND RESULTS

We perform a controlled experiment to evaluate our visual taxonomy generation scheme. Below we state our choice of databases, features used, experiment setting, and clustering results.

4.1 Database

For experiments, we use selected categories from the Corel collection which was kindly made available by James Wang[1]. Out of the 600 categories available, we use 27 categories, examples from all of these are shown in Figure 2. The reason for not taking the complete corel database is that we wish to keep manual validation of the taxonomy tree convenient, especially on printed paper. In a software environment, effective navigation mechanisms can be easily built in order to traverse the taxonomy tree. The categories were selected with the simple rule of having a few categories containing human beings, few nature categories, and a few categories containing man-made objects. Apart from this, the selection process was arbitrary. Each category consists of 100 images of size 384×256 or 256×384. The following features were extracted from each image:

[1]http://wang.ist.psu.edu/IMAGE/

4.2 Features

To demonstrate the effectiveness of the taxonomy generation process, we use a small set of simple visual features comprising of the following:

Colour Features: Colour features have been widely used for image representation because of their simplicity and effectiveness [6]. We use color moments calculated on HSV images. For each of the three layers we compute the layer mean, layer variance and layer skewness respectively. This yields a 9-dimensional vector. Since this does not incorporate any interlayer information, we calculate three new layers SV, VH and HS non-linearly by point-wise multiplication of pairs from original layers and calculate the same 3 moments also for the new layers. The final 18-dimensional vector outperformed a 512-bin 3D Joint Colour Histogram in CBIR tests that we performed.

Texture Features: Texture features can describe many visual properties that are perceived by human beings, such as coarseness, contrast etc. [7]. We use the Discrete Wavelet Transformation (DWT) for calculating texture features. The original image is recursively subjected to the DWT using the Haar (db1) wavelet. Each decomposition yields 4 subimages which are the low-pass filtered image and the wavelets in three orientations: horizontal, vertical and diagonal. We perform 4 level of decompositions and for the orientation subimages we use the entropy ($-\sum_{i=1}^{L} H(i) \cdot \log(H(i))$, with $\mathbf{H} \in \mathbb{R}^L$ being the normalized intensity histogram of the subimage) as the feature, thus resulting in a 12-dimensional vector.

Edge Features: Shape features are particularly effective when image background is uncluttered and the object contour dominates. We use the edge-orientation histogram [8] which we compute directly on gray-scale images by first calculating the gradient at each point. For all points where the gradient magnitude exceeds a certain threshold, the gradient direction is correspondingly binned in the histogram. We use an 18-bin histogram which yields bins of size 20 degrees each.

The final feature vector is a concatenation of the above three vectors and has a dimensionality of 48.

4.3 Results

The hierarchy tree obtained by applying the *hyperplane similarity* method with the gaussian kernel is shown in Figure 3. The tree obtained by the feature similarity method is omitted as in this case it is quite similar. As can be seen, the tree is meaningful even at the upper nodes of the tree. One possible explanation of this can be that the images within a Corel category are relatively uniform, which aids the algorithm. Some peculiarities, such as the placement of the category `dogs` can be understood by viewing the example images for that category, and taking into account that only simple low-level visual features were used.

4.4 Computation Time

For this experiment, i.e. with 27 labels each with 100 training vectors of dimensionality 48, the construction of the taxonomy tree takes about a minute on a standard modern PC (P IV 2.8 GHz running Ubuntu Linux). However, since in reality the databases are expected to be much larger, it is interesting to note how the algorithm would scale. The running time t is proportional to the dimensionality F of the features as the L_1 and L_2 similarity measures are both lin-

ear with respect to F. The effect of the number of training vectors (per label) N is not exactly definable, but empirical studies [3] suggest that SVM training is superlinear with respect to number of training vectors, perhaps even quadratic. Finally, the algorithm scales cubically with respect to the number of labels L. This can be easily seen from the fact that the number of label pairs increase quadratically, and the calculation of distance between a label pair increases linearly with L. Thus, the running time can be expressed in the **O** notation as $\mathbf{O}(L^3 \cdot N^2 \cdot F)$.

5. DISCUSSION

We presented a method for visual taxonomy generation and results for selected keywords of the Corel collection. The natural question is about how the method would scale to larger number of keywords and how would it react to wrongly or inconsistently labelled images. In our opinion, the feature extraction step is the key here. If the features are not representative enough for the keyword in hand, then no subsequent machine learning algorithm can account for this deficiency. It is for this reason that 'loaded' keywords like `Amsterdam` were not used in our experiments. A user at *Flickr* might label an image with the keyword `Amsterdam` because the image was taken in a friend's apartment in Amsterdam, a fact which is impossible to learn from pixel information alone.

However, many other keywords would have more visually consistent content, and the user would likely benefit from a hierarchy tree of the used keywords. With a large number of keywords, it is possible that some parts of the hierarchy tree, especially the upper ones, do not convey much information. However, if the subtrees containing the user-supplied keyword are meaningful, the system could still prove to be useful.

In conclusion, we would like to mention the fact that the method can be used to group any kind of entities from which meaningful features can be extracted. This includes other multimedia types and multi-class problems in general.

6. REFERENCES

[1] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[2] J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1075–1088, September 2003.

[3] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In S. Solla, T. Leen, and K.-R. Mueller, editors, *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

[4] L. Setia and H. Burkhardt. Feature selection for automatic image annotation. In *Proceedings of the 28th Pattern Recognition Symposium of the German Association for Pattern Recognition (DAGM 2006), Berlin, Germany*. LNCS, Springer, September 2006.

[5] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, December 2000.

[6] M. J. Swain. *Color indexing*. PhD thesis, 1990. Supervisor-Dana H. Ballard.

[7] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. *IEEE Trans. Systems Man Cybernet*, 1978.

[8] A. Vailaya, A. Jain, and H. Zhang. On image classification: city images vs. landscapes, 1998.

[9] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[10] X. S. Zhou and T. S. Huang. Relevance feedback in content-based image retrieval: some recent advances. *Inf. Sci. Appl.*, 148(1-4):129–137, 2002.

Figure 2: Sample images from the Corel categories used for the experiments

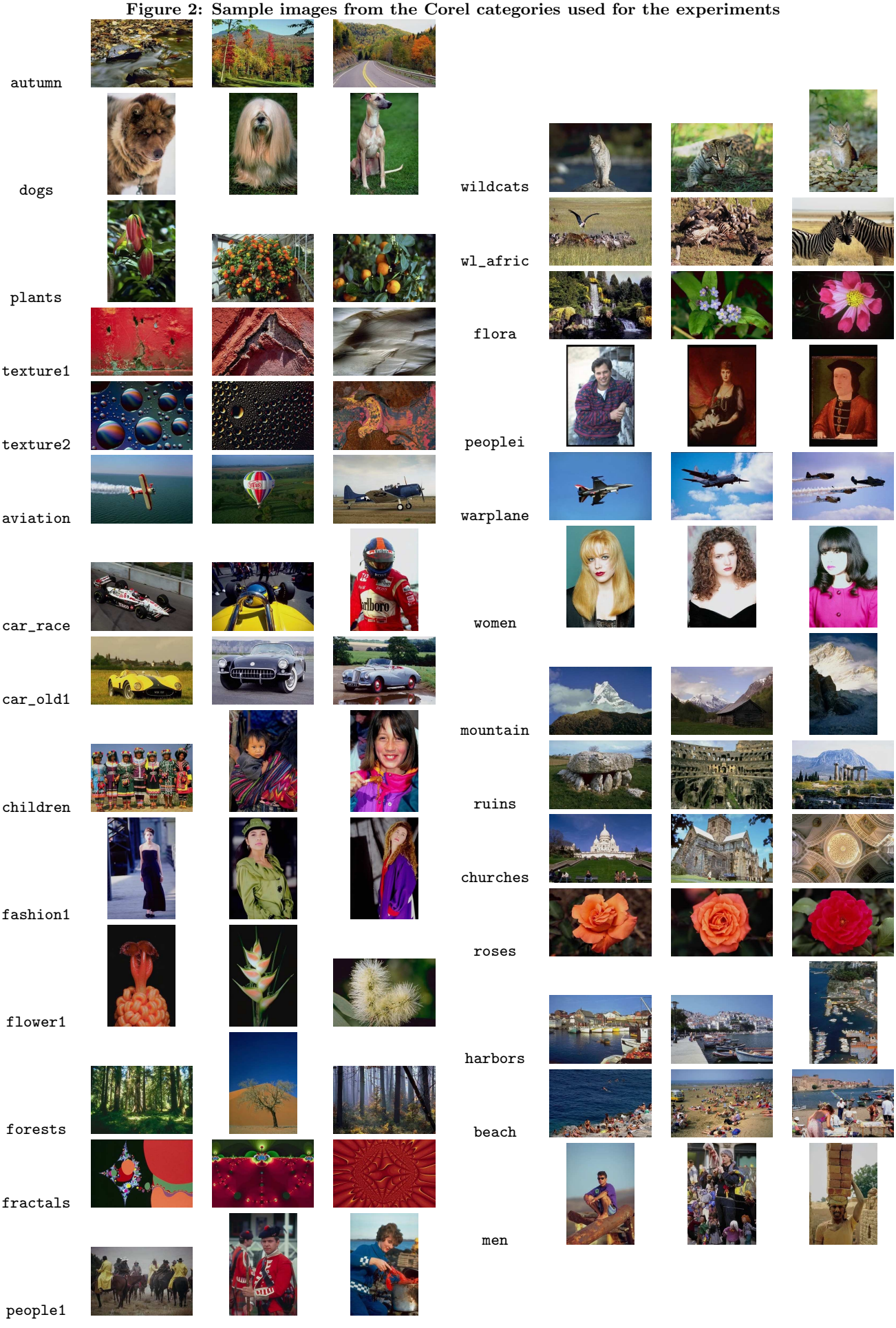

Figure 3: Taxonomy obtained for the 27 selected classes from the corel collection