Color and Texture Indexing using an Object Segmentation Approach

A. Nodari, I. Gallo, M. Vanetti, and S. Albertini

University of Insubria, Dipartimento di Scienze Teoriche e Applicate
via Mazzini 5, 21100 Varese, Italy
{angelo.nodari,ignazio.gallo,marco.vanetti}@uninsubria.it

Abstract. In this study we propose a content based image indexing and retrieval system based on color and texture, which takes advantage of an object segmentation approach. Our case study is a web application for e-commerce whose dataset consists in a set of commercial offers related with the fashion domain. The strength of this application is twofold and consists in the extraction of the visual information from objects of interest and the textual representation of their features, in order to allow an easy integration with a text indexing search engines. Our object segmentation approach is based on a multi neural network strategy. The test results highlight it is reliable for all the object classes. Moreover, the experimental results show that the system can be efficiently used to perform queries by facets and query by example based on the products visual features.

Keywords: color and texture extraction, indexing, object segmentation, multiple neural networks

1 Introduction

In e-commerce applications the information about a product lies in the textual description, but in many cases it is often difficult to express in words what you want to buy, then it is much easier to choose an object using the information contained in the images associated to each commercial offer. Recent work on region based and semantic Content Based Image Retrieval (CBIR) systems established a common way to search using sample images as the search queries [1, 2]. However, a typical search on an e-commerce site should not start with a query based on a sample image, so it is necessary to provide the user with new navigation tools. Our method adopts the common CBIR techniques using an object segmentation strategy in order to combine navigation by facets and by examples. Object segmentation aims to assign predefined class labels to every pixel in an image and therefore is a crucial step in order to find the Object of Interest (OI) in the image and then its basic features required for multidimensional visual navigation, like color and texture. Recently, some online applications specialized in visual search technologies are emerging and let people to search for products using pictures of clothing, handbags, shoes or other items. Some image search engines like Google

Images, Bing Images and Ebay Image Search can be considered as the base line for the commercial CBIR applications. The most influent CBIR applications in the fashion domain were Like.com and Boutiques.com which has been recently integrated into Google Product Search, allowing users to make similarity queries on various features, like silhouette, color and texture. These search engines are still suffering of a lack of quality results mostly because of their wide application domain. A similar work is [3] where are used color, texture and shape to make a query by example. They firstly select those images that have similar color and texture to those of the query and then apply a deformable template matching using the query shape.

In our context, the use of shape as a feature was discarded due to the complexity on the different poses of the same object, the difficulty of the user in the composition of the shape query and in the impossibility in the extraction of this type of information for the facet navigation.

Moreover, unlike [3], we index all the images to facilitate the query process.

Our approach being limited to a specific class of objects, allows us to exploit segmentation algorithms that can identify the object of interest focusing on its visual features with a consequent improvement of the retrieved results. Anyway, there exist several websites which integrate CBIR algorithms in order to allow visual search of products. These applications are built on machine learning and visual recognition technologies, but at the best of our knowledge, the implementation details are not completely published, so it is very difficult to make a quantitative comparison running for example these algorithms on the same dataset. This lack of information is also highlight by Skopal which makes a survey [4] of the state of the art in commercial CBIR engines underlining that in many cases information about the used technology is not available.

Our visual search web application is deployed in a commercial version named Drezzy [5]. Figure 1 reports a screenshot of the web interface.

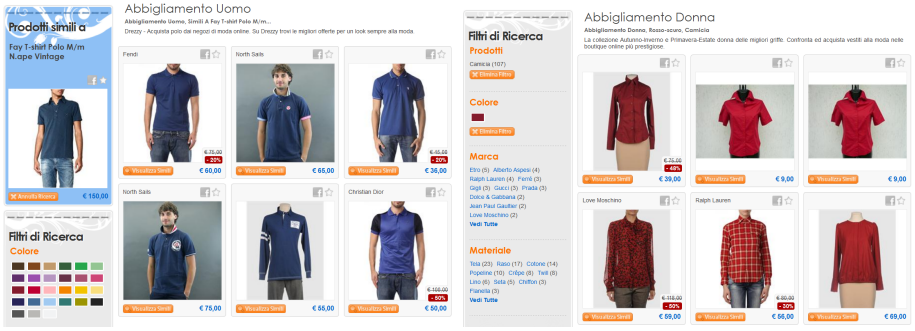

Fig. 1: Two screenshots of the application made using the method proposed in this article. On the left there is an example of color entities identified by the use of the proposed approach. The images shown on the right shows the search result obtained by selecting “red” as color and “sweater” as product.

2 The Proposed Method

The basic idea of our method consists in a first segmentation step of the OI before the visual feature extraction, using our recent technique based on a set of neural networks described in section 2.1. From each segmented object is possible to extract a precise information and in particular in sections 2.2 and 2.3 we analyze color and texture extraction. This visual information is then transformed into a textual description which takes advantages from the common indexing and retrieval tools for text mining and can be easily integrated in a text indexing system. Moreover, by the same process, it is possible to extend the proposed method extracting other visual information, such as the shape of the object or its subparts.

For the color and texture extraction phase we used a Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel, as it provides the state of the art for many classification problems [6].

2.1 Object Segmentation

In this section we present the object segmentation model used in this application. Our model, called Multi-Networks for Object Detection (MNOD) [7], has already been presented as a tool for objects detection and is here adapted to the more complex objects segmentation problem. We chose to use a model based on supervised neural networks because neural models have a high generalization ability and good robustness, in this way the same type of performances are always guaranteed varying the classes of objects to be segmented.

The MNOD model consists of a tree of neural networks, where each network uses a sliding window of size W_S to read contextual information extracted from all the input images. Each MNOD node $C^n_{I_S,W_S}$ extracts an input pattern for the neural network by concatenating the contents of the sliding window, placed over each resize input image, according to a second parameter I_S. A real example of configuration created for our application is shown in Figure 6, where leaf nodes are the features extracted from the input image. In [7] you can find many details on the segmentation process that we have here omitted for lack of space. A node produces in output a soft classification map (segmentation map) where the pixels containing higher values identify the objects of interest. Only the root node of a configuration is interpreted as the segmentation map for a given object class, using a threshold in order to obtain a binary map. In the present work, the structure of a single node consists in a feed-forward Multi-Layer Perceptron (MLP), trained using the Rprop (Resilient Backpropagation) learning algorithm proposed by Riedmiller and Braun [8].

The particular aspect of this model lies in the connection between nodes, which in practice means that the output of a node becomes the input of a parent node. This means that some input images of a node can be the output of its child nodes. The links between nodes at different levels indicate which nodes make use of the output of a particular node in the tree structure. In [7] it is possible to find more details about the configuration of the segmentation algorithm.

The features set, such as information on edges, color, etc., can be directly extracted from the input images. In this work we used the normalized red ($RN_s = R/(R+G+B)$), green ($GN_s = G/(R+G+B)$) and blue ($BN_s = B/(R+G+B)$), and the Brightness (Br_s) as color features. Information about edges were extracted using a 1-D vertical (EV_s) and horizontal (EH_s) derivative filter (using the mask $[-1, 0, 1]$) and a simplified version of the *Histogram of Oriented Gradient* (HOG) [9] feature descriptors, here Oriented Gradient ($OG_{b,s}$). We first compute the gradient values and then, instead of creating the cell histograms as in [9], we create b input maps, one for each gradient orientation interval configured. The s parameter, used in all the features, represents the scale at which the input image is resampled, before the feature computation. Figure 2 shows an example of all the features used in this work. Figure 6 shows a real configuration with 4 levels, the figure also shows the segmentation map of the OI identified.

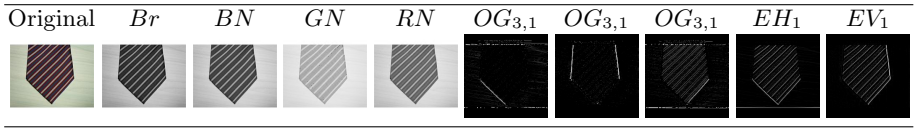

Fig. 2: An example of features extracted for MNOD. The input image is showed on the left. In our experiments, each node can be configured with a subset of this features.

The search space for an optimal configuration of the MNOD is very large because we have to choose the depth of the tree structure, the number of nodes for each level, the best resampling parameters and the most suitable set of features for each node. To restrict the search space of parameters, and make the configuration process much faster, we set 4 layers as a maximum depth for the tree structure, because we have noticed experimentally that adding more than 4 layers, the accuracy improves very slightly, as analyzed in our previous work [7]. In addition, we fix the number of nodes to 4 in the first level and 3 in the two intermediate levels. We have also observed experimentally that using small values of W_S and large I_S values for all the nodes in a layer, alternated by large W_S and small I_S in the subsequent layer, results in a better segmentation accuracy. For this reason, we configure each segmentation model by choosing the parameters for each level between the following subsets $I_S = \{50, 90\}, \{10, 30\}, \{40, 70\}, \{30, 70\}$ and $W_S = \{1, 3\}, \{7, 9\}, \{1, 3\}, \{3, 5\}$, where the first subsets of parameters on the left are for the first level and the last on the right for the output level. To speed up the configuration process we used only nodes pre-configured with the following subsets of features: $C^n_{I_S,W_S}(OG_{b,s})$ choosing $b \in \{3, 5\}$ and $s \in \{1, 0.5, 0.25\}$, or $C^n_{I_S,W_S}(EV_s, EH_s)$ choosing $s \in \{1, 0.5, 0.25\}$, or $C^n_{I_S,W_S}(OG_{b,s}, RN_s, GN_s, BN_s)$ choosing $b \in \{3, 5\}$ and $s \in \{1, 0.25\}$, or $C^n_{I_S,W_S}(OG_{b,s}, Br_s)$ choosing $b \in \{3, 5\}$ and $s \in \{1, 0.5, 0.25\}$.

2.2 Color Extraction

Our method, to extract the representative colors from the images, is based on a previous work [10] in which we presented an efficient and innovative approach to represent the color information. This method uses a new color similarity metric based on the human visual system and a new color indexing based on a textual approach. We used a text indexing engine to facilitate the integration of visual features in a database of text documents. The textual signature is build by weighting the image's colors in according to their occurrence in the image. A related method of color extraction based on textual features has been implemented in the Lucene[1]-based open source CBIR project Lire [11]. Its limitation consists of the image data extraction that Lucene is unable to index, exploiting only the level of access to the file system. Our approach differs in that because we extract a textual color description that can be directly handled by Apache Solr[2].

In order to manage the color feature, we have selected two sets of representative colors based on the natural language color descriptors derived from the ISCC/NBS system as proposed by the Inter-Society Council[3]. The first set is called *Main Colors* which is composed by 27 colors and it is used to perform a faceted query, whereas the second set is called *Descriptive Colors* and is composed by 267 colors and because is more descriptive it is used to perform queries by example. All these two sets of colors have been created in order to obtain a significant quantization of the entire RGB color space. These sets of colors are used to extract the principal colors and their quantity in the image considering all the pixels that belong to the object of interest identified by the segmentation mask, avoiding the extraction of the colors from the background.

One of the major problems encountered in color extraction from images arises from the metric adopted to evaluate the distances between colors. The Euclidean distance on the RGB space, or one of its linear transformations, is inadequate to represent the human visual similarity [10], so we have adopted the approach used in our previous work [10] which is based on a similarity metric estimated on human visual perception. This metric uses a machine learning approach which is trained on a set of user evaluations about the similarity of different colors. The consequent result is a system which is able to map few representative colors in the whole RGB space preserving the similarity perception between colors better than other quantization systems.

The color extraction phase is performed through an image quantization, in which we associate every RGB pixel of the object of interest to a representative color of both color sets. The results are two histograms, one composed by the main colors and the other by the descriptive colors, as shown in Figure 3.

The extracted color feature composed by main colors and descriptive colors is arranged in a textual document composed by a main colors field and a descriptive colors field respectively, suitable for an efficient indexing. In order

[1] `http://lucene.apache.org`
[2] `http://lucene.apache.org/solr`
[3] `http://www.iscc.org/`

to keep the information on the amount of color extracted, each color name is repeated in according to its occurrence to build the field of the document to index, in such way it is possible to use, in the retrieval phase, the term frequency as a measurement of color quantity as explained in details in [10]. For example if 20% of $Color_1$ and 80% of $Color_2$ are extracted from an image to build a document with a maximum number of words $t = 100$, we will repeat the textual labels identifying the $Color_1$ and $Color_2$, 20 times and 80 times respectively. The repetition of a color within the image according to its presence is needed in the retrieval phase in which Apache Solr makes a ranking depending on the term frequency of each words in the document. In this way is possible to sort the images according to the amount of the selected color which they contain.

Our indexing system can be queried in two different ways: using a query by facets or performing a query by example. In the first case is possible to search through all the images in the indexing engine which have among their main colors field the main color selected by the user. Furthermore, the result is ranked according to the term frequency within the documents that describe the image. In the second case a query by example is performed using the information relating to the descriptive colors field as contain a greater information. A typical feature, provided by the textual indexing engines, is the boosting factor used to weight the relevance of the terms which compose the query. Given a query image we use the descriptive colors boosted in according to their presence within the image. As an example, given the following descriptive colors: 60% of $c1$, 30% of $c2$ and 10% of $c3$ they can be used to build the following query: "$c1^{0.6}$ OR $c2^{0.3}$ OR $c3^{0.1}$", performed on the descriptive colors field of each documents in the dataset.

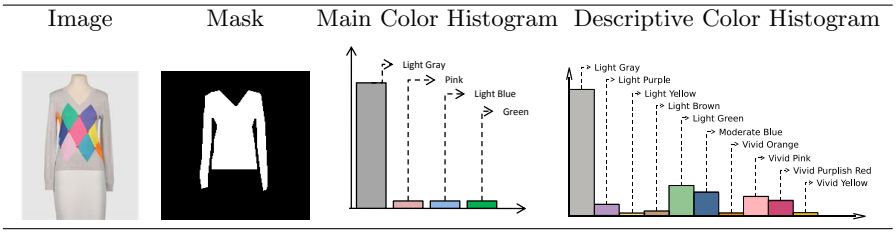

Fig. 3: An example of color extracted from an offer's image in according to the set of pixel belonging to the object of interest showed in the mask image. The histogram represents the amount of colors contained in the object of interest. For each histogram's bin the textual descriptor of the representative colors is displayed.

2.3 Texture Extraction

The texture extraction task is conceived as a classification problem. Dealing with a massive amounts of offers, a key requirement of the system is the extrac-

tion speed for visual features and the ability to rank images according to the amount of texture contained. The texture classes with which we deal presents strongly spatial oriented patterns, this is why a simple texture descriptor, based on convolution filters, is powerful enough to separate the problem classes.

Processing the original image with a convolution filters bank (see Figure 4), we obtain the maps $M_1, M_2, M_3, M_4, M_5, M_6$ from which the final features are extracted as follows. Given $k \in \{1, 2, 3, 4, 5\}$

$f_k^{max} = \sum_{p \in I} \max(0, M_k(p))/|I|,$
$f_k^{min} = \sum_{p \in I} \min(0, M_k(p))/|I|,$
$f_6 = \sum_{p \in I} \max(0, M_1(p), M_2(p), M_3(p), M_4(p))/|I|,$

where I represents the set of points contained in the OI segmentation mask.

The first four maps are obtained using standard edge detection filters (convolution kernels are shown in Figure 4), M_5 is obtained using a custom convolution kernel useful to enhance small closed shapes such as circles and squares typical of some textures, while M_6 is computed by taking the maximum punctual value from the first 4 maps.

The feature set is small (a total of 11 features), but in our domain contains important visual information able to discriminate the chosen texture classes.

The class predicted by the trained SVM is translated into a linguistic label to easily index the texture information extracted, using the same engine used for text and color.

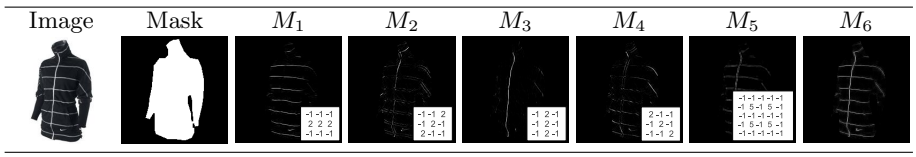

Fig. 4: An example of the input image, the segmentation mask containing the OI and the results of the convolution filters used to extract the texture features, together with corresponding convolution kernels.

In order to obtain a ranking procedure, we exploited the texture classification model described above. Given a sample image and its segmentation mask, like the one in Figure 4, we would like to estimate how much the input image fits each descriptive texture class. The target is to use such information in order to arrange big image datasets by a text indexing engine, so setting up the possibility to query them. That procedure aims to select two descriptors. The first is based on the whole input image which falls under the segmentation mask. Using this descriptor, we obtain a single texture class classifying the object in its integrity. The second descriptor consists of a histogram of texture names occurrences, extracted from a fixed set of fixed size squared patches. Each patch falls completely inside the segmented image region. To compose the histogram,

patches are then classified by the SVM and the relative frequency of each texture name is normalized so that it sums to 1.

3 Experiments

The application we propose was evaluated on different datasets in order to isolate and measure the performance of the three main phases. In a first experiment (see section 3.2) we evaluated the neural segmentation algorithm that underlies the other two following steps. In particular in the section 3.3 we evaluate the automatic extraction of color and in section 3.4 we evaluate the automatic extraction of the main textures. The two phases that follow the objects segmentation have been evaluated independently, extracting the information of color and texture only from the pixels belonging to a mask that identifies the object, created by hand for each image in the dataset. Finally, the last experiment consists in evaluating the color and the texture extraction after the segmentation step, thus not using the ground truth mask but the segmentation map generated by the trained MNOD.

All the tests reported in the following sections were performed using a single thread C# code, on an Intel®Core™2 Duo processor T8100 @2.10Ghz.

3.1 Datasets and Evaluation Metrics

Analyzing the literature we realized that there are no image datasets available to evaluate an application that deals with objects segmentation in the fashion field. For this reason we created seven datasets of images, each of which contains images of one of the following categories: *Underwear*, *Lingerie*, *Menswear*, *Womenswear*, *Tie*, *Shoe* and *Hat*. Each dataset contains about 200 images, divided into train and test subsets. Figure 7 contains some representative examples of each dataset.

To evaluate the performances in color and texture retrieval we have built the *color-texture* dataset containing nine classes of textures with a wide range of colors, typical in the domain of women's clothing. The texture classes represent the typical decorations found on tissues, in particular we chose horizontal lines (*horiz*), *leopard*, polka dots (*polka*), *rhombus*, *squares*, vertical lines (*vertic*), *zebra* as texture classes. The solid color (*color*) and "*other*" classes were also been included to represent, respectively, tissues with homogeneous color and with decorative patterns that do not fall into the 7 textures listed above. The images in these datasets were also manually labeled associating at each image the most representative colors. In Figure 5 we show some sample images highlighting the manually segmented OI.

All the images in the datasets used in this work, were taken from the Drezzy application and are available for research purposes [12].

For evaluating segmentation results, we used the segmentation accuracy showed in (1) proposed by the contest "The PASCAL Visual Object Classes Chal-

lenge" [13] and usually called *Jaccard index* [14].

$$\text{Acc} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{1}$$

The values under consideration are calculated pixel-by-pixel: TP are the True Positives, FP the False Positives and FN the False Negatives. We consider the problem of segmentation as a classification problem formalized as a function that takes a pixel and returns 1 if the pixel belongs to the foreground or 0 if it belongs to the background.

To assess the texture classification we computed the *Confusion Matrix* [15], extracting the *overall accuracy* and *kappa* coefficient [16].

To evaluate a ranked retrieval results from the texture classification, *precision* and *recall* values can be plotted to give a *precision-recall curve* [13]. Examining the entire precision-recall curve is very informative, but there is often a desire to boil this information down to a few numbers, or perhaps even a single number. The traditional way of doing this is the area under the 11-point interpolated precision vs recall graph, obtaining the *Average Precision* (AP) measure.

We used two different metrics for representing the color extraction accuracy. The first metric, called "coarse evaluation", takes into account what are the colors that have been extracted and those not, while the second metric called "fine evaluation" also considers the percentage of extracted color and compares it with the percentage of color in the dataset, as explained in [17]. In order to evaluate the ranking performance in the retrieval phase using the color descriptor we have performed two different experiments. In the first experiment we used the Normalized Discounted Cumulative Gain [18] as explained in details in our previous work [10].

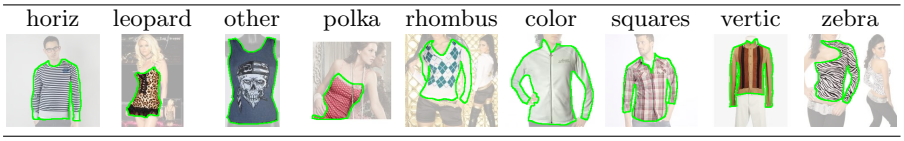

Fig. 5: Image examples taken from the *color-texture* dataset. The OI class was segmented by hand.

3.2 Object Segmentation

In this experiment we evaluated the performance of our segmentation algorithm on the datasets presented in Section 3.1. In order to evaluate the segmentation result we used the per class measure adopted in VOC2010 [19].

For each of the seven classes we trained an MNOD model using the configuration process described in Section 2.1. An example of MNOD configured for the dataset "Menswear" is shown in Figure 6, which also shows the maps produced

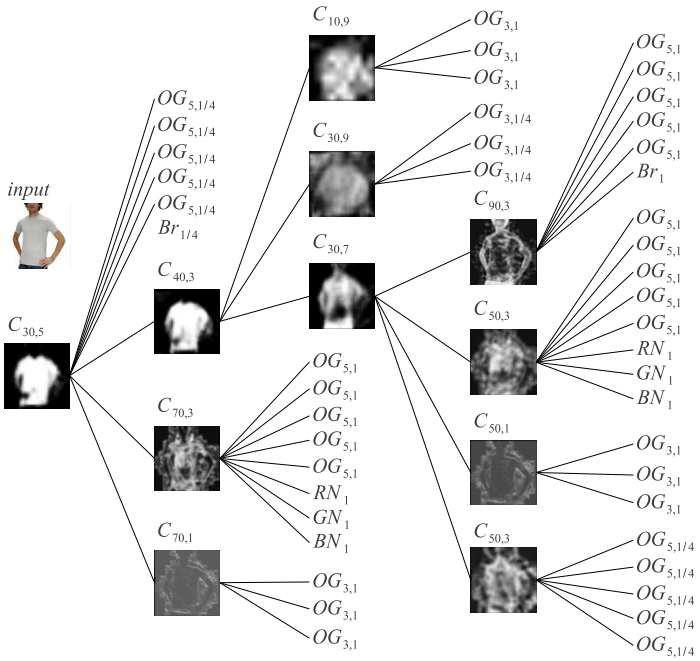

Fig. 6: MNOD configuration used for the dataset "Menswear". The maps produced by each node $C^n_{I_S,W_S}(F_1,\ldots,F_m)$ and the features used in input, all related to the same test image, are showed.

by each node for a given input image, and parameters and leaf nodes with which each node has been configured. Finally, we applied the trained models to the images of each test set and the results are shown in Table 1.

As showed in Table 1, we have obtained a high quality segmentation result and in Figure 7 we have included an example for each dataset. These results lead to significant advantages in terms of quality of the extracted features, which are related only to the OI and are not influenced by the background. The configuration with the set of constraints established is nearly automatic, and then requires low computational time. Moreover, the results in Table 1 highlight that our algorithm can always be reliable regardless the class of objects on which you choose to work.

The computational complexity of the segmentation algorithm was computed on the seven data sets we used in our experiments. The average time to process a test image is $1196ms$. If an image has already been processed it is possible to cache parts of the processed information and give an answer in an average time of $1ms$.

Table 1: Training and test accuracies obtained with the MNOD object segmentation algorithm when applied to the seven object segmentation datasets. Each class of objects was assessed individually by measuring the object (Obj) and background (Bg) accuracy, using the VOC2010 [19] evaluation metric.

Dataset	**Obj Train**	**- Bg Train**	**- Obj Test**	**- Bg Test**
Underwear	73%	91%	68%	88%
Lingerie	67%	91%	64%	91%
Menswear	79%	83%	73%	79%
Womenswear	64%	85%	57%	83%
Tie	84%	96%	75%	93%
Shoe	81%	94%	69%	87%
Hat	78%	91%	76%	88%

3.3 Color Extraction

In order to evaluate the quality of the colors extracted by the proposed system, we have extended the *color-texture* dataset and for each image of this dataset, we have manually associated a set of color labels taken from the *Main Colors* set. In addition, for each added color we have automatically estimated the percentage of the selected color within the image in order to obtain a measure of the distribution of colors in the image. In this way we have collected a dataset in which every image is associated with the colors of the object of interest and the value of occurrence of each color.

Since the computational time of the algorithm depends on the size of the image, we have investigated the way to reduce it, resizing the images. We have thus reached a trade off between speed and quality of results by resizing the images to 50×50 pixels preserving the color information. The average time required to extract the principal colors for each image is $37ms$ and the result of this experiment is showed in Table 2.

Table 2: Color extraction Coarse and Fine Evaluation using the Metric explained in Section 3.1 on the *color-texture* dataset. The Coarse experiments take into account only the correctness of the extracted colors, instead the Fine Evaluation also consider the amount of color extracted.

	Coarse Evaluation	Fine Evaluation
Precision	0,41	0,49
Recall	0,78	0,53
Overall Accuracy	0,85	0,64
F-measure	0,51	0,50

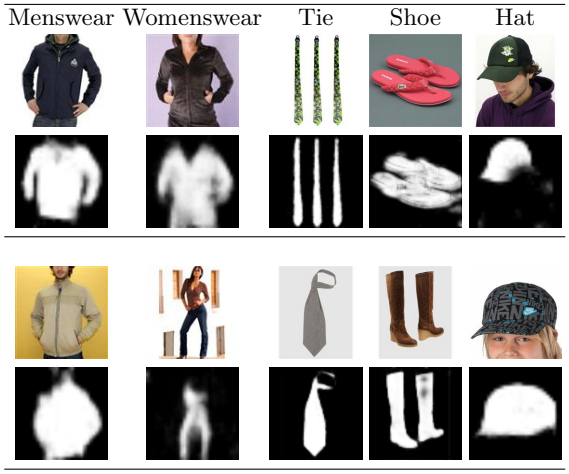

Fig. 7: In each column two examples of input images and segmented objects, belonging to the datasets used in this work. All the images belong to the validation set.

The color feature can be represented as a histogram where each bar corresponds to a color whose height corresponds to the amount of extracted color within the image. In Figure 8 we report the results of the proposed algorithm compared with the values of truth of some patterns in the dataset.

To perform this experiment we built a set of ranked documents using the *color-texture* dataset weighting each document by their amount of colors. In this way is it possible to order the dataset starting from a single color (for the *Query By Facet* tests) or starting from a query image (for the *Query by Similarity* tests) ordering the most similar images in according to the colors and their quantities that matches with the query image. As a consequence it is possible to compare the retrieved ranked results with the ideal ranked documents. We performed a query for each *Main Colors* as a *Query By Facet* test to evaluate how many correctly documents are retrieved and how their ranked order is correct. We performed also an evaluation of the *Query by Similarity* comparing, given a query image, the retrieved documents to the truth ranking generated.

Once the documents have been indexed, we used the Normalized Discounted Cumulative Gain to evaluate the ranked results of our indexing system as summarized in Equation 3.

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2 (1 + i)} \tag{2}$$

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{3}$$

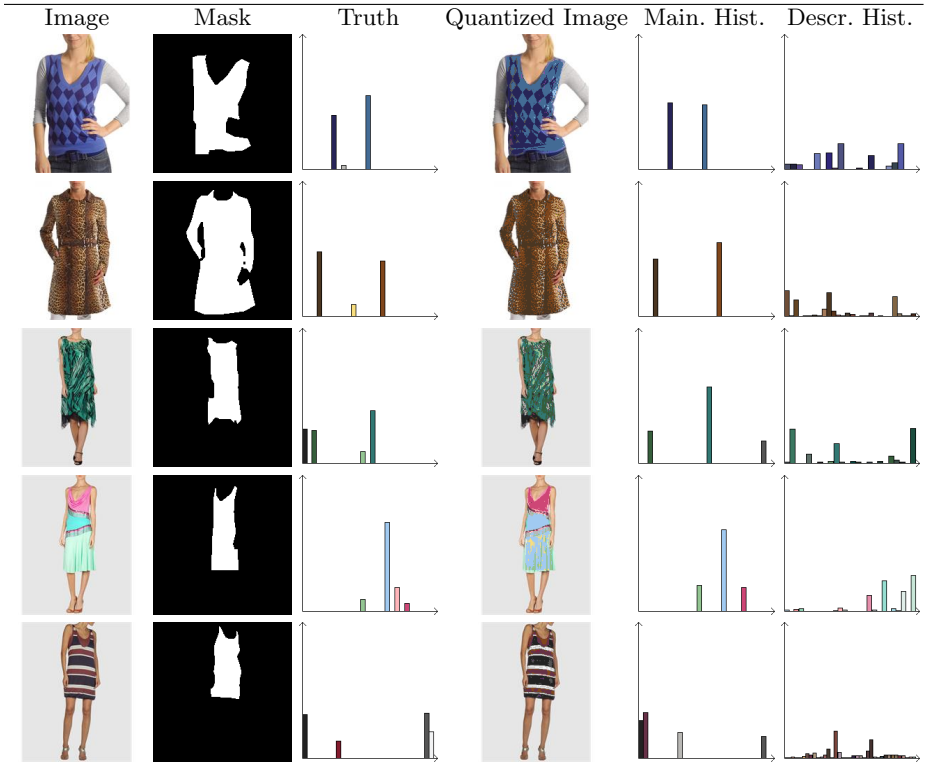

Fig. 8: Some examples of color extraction from the *color-texture* dataset. For each row, the original image (Image), the segmentation mask (Mask), the histogram of truth (Truth) and the color extracted histogram (Color Histogram) are summarized.

Where i is the rank position of a specific retrieved document, rel_i is the relevance value associated to a specific type of document (in our case $rel_i \in \{0, 1\}$ such that $rel_i = 0$ is a not relevant document and $rel_i = 1$ is a relevant document). The $nDDG$ take also in consideration the ($IDCG$) Ideal Discounted Cumulative Gain computed on the truth corpus of documents. The value of the $nDCG$ represents the goodness of the ranking and in particular $nDCG \in [0, 1]$ where a value of 1 represents the ideal ranking result. We have performed two type of experiments, set $p = 50$ the number of the first ranked documents to evaluate, which is a reasonable number that summarize the average number of elements usually returned by a search engine in the first page. In the first experiment (Query by Facet) we performed a query for each Main colors and the Ideal Ranking was built ordering the documents in according to the quantity of color information in the truth dataset. In the second experiment (Query by Example) for each document of the dataset we have performed a query as explained in Section 2.2. In this experiment the Ideal Ranking, for each example, was built

using the distance between the truth color histograms of the dataset. As a result of these experiments, we have obtained a $nDDG$ of $0,82$ for the *Query By Facet* and $0,93$ for the *Query by Similarity*.

In the second experiments, summarized in Figure 9, we uses Interpolated Precision vs. Recall Graph to test the behavior of the *Query By Facet*.

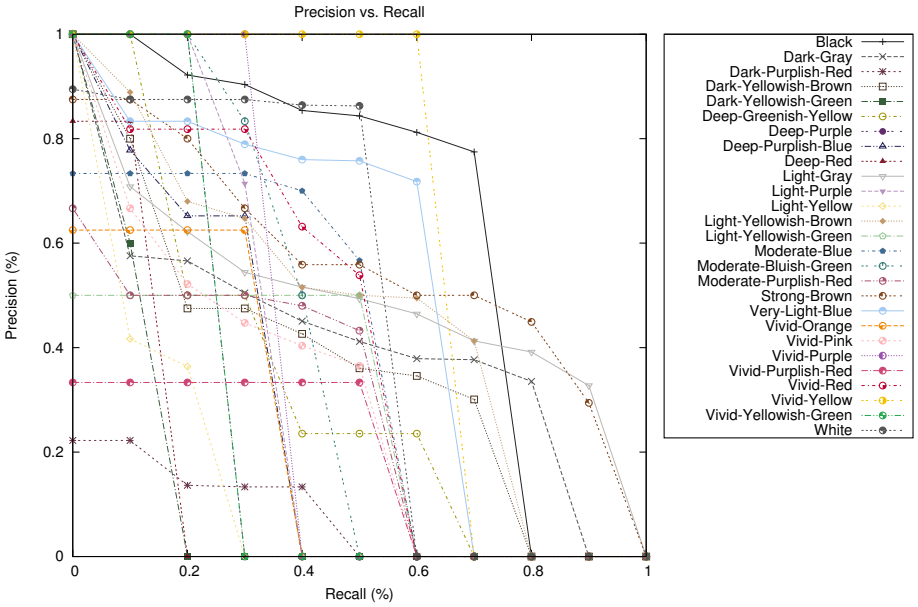

Fig. 9: Interpolated precision vs. recall graph of all the colors computed over the test data set using GT segmentation masks. It is possible to notice how there are more difficult classes (for example other texture, polka dots and leopard), whose texture variability causes a decreasing on the classification accuracy.

.

3.4 Texture Extraction

For this experiment, an SVM classifier was trained using the *color-texture* dataset and using a 5-fold auto-configuration strategy to tune the SVM parameters. The patch size employed in our experiments is 20×20 pixels, a value chosen as the result of a simple experimental phase. We extract 31 patches, eventually overlapped, in random locations upon the object segmentation mask in order to perform the texture ranking. We choose a prime number so we can easily avoid majority voting conflicts. Table 3 shows the confusion matrix obtained evaluating the texture classification method, which can be summarized in an overall accuracy of 75.8% and a Kappa coefficient of 0.73, resulting in a substantial

agreement between the truth data and the classifier predictions [16]. Experiments show that the chosen features may have a good discrimination power on the used texture classes. Anyway, analyzing the precision vs recall graph, showed in Figure 10, we observe poor results on some texture class. Each class has a fairly good ranking behavior except for "polka" and "leopard" classes. The main reason why this happens is that while the typical images for the other classes, such as vertical and horizontal lines, rhombus, etc., are characterized by homogeneous patterns frequently repeated inside the same OI, often polka dots and leopard images are characterized by patterns with non-uniform shape, scale and color, resulting in an ambiguous feature values for each patch extracted inside the OI and a consequently probable misclassification. Regarding the time performance, using 200×200 pixels images, the average time required to extract texture features and predict texture class is $50ms$, where almost all of the time is spent in the feature extraction phase.

Table 3: Confusion matrix for the texture classification problem.

	horiz	**leopard**	**other**	**polka**	**rhombus**	**color**	**squares**	**vertic**	**zebra**	**PA**
horiz	19	0	0	0	0	0	0	0	2	91%
leopard	0	14	4	4	0	0	1	0	1	58%
other	0	2	13	7	2	0	4	0	2	43%
polka	0	0	0	10	0	0	0	0	0	100%
rhombus	0	0	1	1	13	0	0	0	2	77%
color	4	0	2	0	0	37	1	0	0	84%
squares	0	0	0	0	0	0	14	0	0	100%
vertic	0	0	2	0	0	3	0	22	0	82%
zebra	2	0	2	1	0	0	0	0	15	75%
UA	76%	88%	54%	44%	87%	93%	70%	100%	68%	

3.5 Integration Test

In order to evaluate the overall behavior of the proposed system, we conducted a full experiment integrating all the phases already described and tested separately. We want to investigate how the results obtained in Sections 3.3 and 3.4 vary taking into account the segmentation step performed by our segmentation algorithm.

In Tables 4 and 5 we summarize the results of some experiments using output segmentation maps taken from the MNOD algorithm instead of the ground truth masks. The first tests, showed in the first column of each table, could be considered as they were run using a 100% accuracy segmentation algorithm to identify the OI.

About the color extraction, when the AP is calculated using ground truth segmentation masks, the results are slightly better then the results obtained with real segmentation masks generated by our algorithm (MNOD). So we can assess

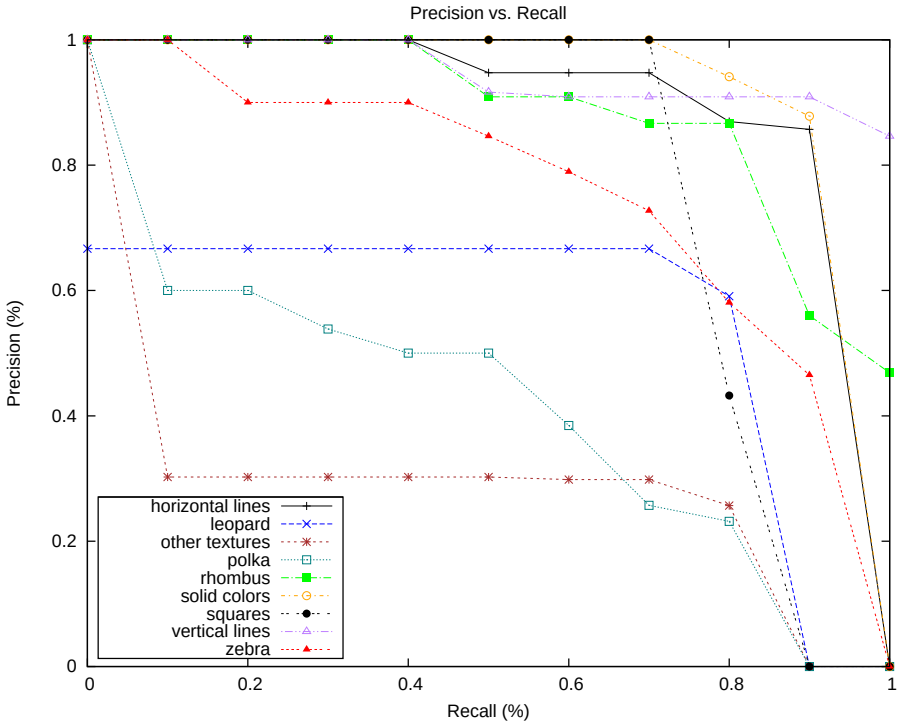

Fig. 10: Interpolated Precision vs. Recall graph of all the texture classes computed over the test data set using GT segmentation masks.

there are no drastic differences using the MNOD mask rather then the ground truth segmentation mask.

For the textures, we can notice the ideal situation is the executions performed with the ground truth segmentation masks, both in training and test. However, in a real situation, we can see that it is better to use the real segmentation masks generated by the MNOD rather then the ground truth masks, even if used only in the training phase. That approach gives a more robust result in terms of Kappa and AP mean values.

4 Conclusions

In this study, we presented an innovative CBIR system able to extract color and texture information from the object of interest and convert them into a textual representation. The method has been successfully integrated into the search engine of an e-commerce web site exploiting existing text-based search capabilities. The overall system is a production-ready CBIR application able to index thousand of documents, it supports real time query by similarity and provides multidimensional navigation by facets. It also successfully deals with low

Table 4: Average precisions obtained from the PR graphs in Figure 10 for color extraction. The first column collects the names of the main colors, the second column shows ranking results for the method tested with the ground truth (GT) segmentation masks while in the third column the method was tested with the masks obtained by the MNOD segmentation algorithm.

Test masks	GT	MNOD
Black	0.65	0.61
Dark Purplish Red	0.08	0.09
Dark Yellowish Green	0.15	0.15
Dark Yellowish Brown	0.38	0.34
Deep Greenish Yellow	0.34	0.26
Deep Purple	0.27	0.18
Deep Purplish Blue	0.28	0.22
Deep Red	0.15	0.11
Light Gray	0.50	0.48
Light Purple	0.34	0.34
Light Yellow	0.16	0.18
Light Yellowish Brown	0.47	0.36
Light Yellowish Green	0.27	0.27
Moderate Blue	0.38	0.42
Moderate Bluish Green	0.39	0.39
Moderate Purplish Red	0.28	0.34
Strong Brown	0.55	0.51
Very Light Blue	0.52	0.51
Vivid Orange	0.23	0.23
Vivid Pink	0.31	0.24
Vivid Purple	0.36	0.36
Vivid Purplish Red	0.18	0.20
Vivid Red	0.42	0.43
Vivid Yellow	0.64	0.55
Vivid Yellowish Green	0.27	0.27
Dark Gray	0.42	0.40
White	0.48	0.57
Mean values	0.35	0.33

Table 5: Average precisions obtained from the PR graphs in Figure 9 and Kappa values for textures classification. First column shows the names of texture classes, second column collects ranking results for the method trained and tested with ground truth (GT) segmentation masks. The results in the third column was obtained training the method with the ground truth masks and testing it with the masks obtained using the MNOD segmentation algorithm (MNOD). The last column collects the results obtained using the MNOD masks both for the training and for the testing.

Train masks **Test masks**	GT GT	GT MNOD	MNOD MNOD
Horizontal lines	0.87	0.57	0.68
Vertical lines	0.95	0.87	0.85
Leopard	0.54	0.55	0.70
Polka	0.42	0.46	0.61
Rhombus	0.87	0.48	0.57
Solid Color	0.89	0.73	0.69
Squares	0.77	0.78	0.78
Zebra	0.74	0.76	0.84
Other texture	0.30	0.21	0.29
Mean values	0.68	0.57	0.67
Kappa	0.73	0.58	0.64

resolution images and permits the information extraction even with commercial offers having ill formed or empty textual description.

The principal drawbacks are the time required for the feature extraction stage before the indexing and the dependency from the dataset chose for the training phase. In fact, the use of supervised algorithms requires a time consuming phase to build a training dataset for each product type, but on the other hand, we have the advantage of obtaining high accuracy results as showed by the experimental results.

The proposed method can be easily extended exploring new features which can be represented in a textual form. A good candidate might be the visual attributes or the shape of the object of interest.

References

1. Wang, J.Z., Li, J., Wiederhold, G.: Simplicity: Semantics-sensitive integrated matching for picture libraries. IEEE Trans. Pattern Anal. Mach. Intell. **23** (2001) 947–963
2. Liu, Y., Zhang, D., Lu, G., Ma, W.Y.: A survey of content-based image retrieval with high-level semantics. Pattern Recognition **40** (2007) 262–282
3. Zhong, Y., Jain, A.K.: Object localization using color, texture and shape. Pattern Recognition **33** (2000) 671 – 684
4. Skopal, T.: Where are you heading, metric access methods?: a provocative survey. In: Proc. of SISAP 2010. SISAP 2010, New York, NY, USA, ACM (2010) 13–21
5. 7pixel Srl: (Drezzy, a search comparison engine for visual online shopping - http://www.drezzy.com/)
6. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning **20** (1995) 273–297
7. Gallo, I., Nodari, A.: Learning object detection using multiple neural netwoks. In: VISAP 2011, INSTICC Press (2011)
8. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: IEEE conf. on Neural Networks. (1993) 586–591
9. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proc. CVPR. (2005) 886–893
10. Nodari, A., Gallo, I.: Image indexing using a color similarity metric based on the human visual system. In: International Conference on Machine Vision, Image Processing, and Pattern Analysis (ICMVIPPA 2011). (2011)
11. Lux, M., Chatzichristofis, S.A.: Lire: lucene image retrieval: an extensible java cbir library. In: MM '08, New York, NY, USA, ACM (2008) 1085–1088
12. ArteLab: (Applied recognition technology laboratory, dipartimento di scienze teoriche ed applicate - http://www.dicom.uninsubria.it/artelab/)
13. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/ (2011)
14. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison Wesley (2005)
15. Congalton, R., Green, K.: Assessing the accuracy of remotely sensed data: principles and practices. Lewis, Boca Raton, FL (1999)
16. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. Biometrics **33** (1977) 159–174

17. Pedoia, V., Colli, V., e.a.: fMRI analysis software tools: an evaluation framework. In: SPIE Medical Imaging 2011. (2011)
18. Jarvelin, K., Kekalainen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems **20** (2002) 422–446
19. Everingham, M., Van Gool, L., e.a.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision **88** (2010) 303–338