Image and Vision Computing 28 (2010) 317–328

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Automatic cleaning and segmentation of web images based on colors to build learning databases

Christophe Millet [a,b,*], Isabelle Bloch [b], Patrick Hède [a], Pierre-Alain Moëllic [a]

[a] CEA, LIST, Laboratoire d'ingénierie de la connaissance multimédia multilingue, 18 Route du Panorama, BP6, F-92265 Fontenay-aux-Roses, France
[b] ENST (GET Télécom Paris), CNRS UMR 5141 LTCI, Paris, France

ARTICLE INFO

Article history:
Received 7 August 2007
Accepted 1 June 2009

Keywords:
Semantics
Web images
Automatic segmentation
Sorting images

ABSTRACT

This article proposes a method to segment Internet images, that is, a group of images corresponding to a specific object (the query) containing a significant amount of irrelevant images. The segmentation algorithm we propose is a combination of two distinct methods based on color. The first one considers all images to classify pixels into two sets: object pixels and background pixels. The second method segments images individually by trying to find a central object. The final segmentation is obtained by intersecting the results from both. The segmentation results are then used to re-rank images and display a clean set of images illustrating the query. The algorithm is tested on various queries for animals, natural and man-made objects, and results are discussed, showing that the obtained segmentation results are suitable for object learning.

1. Introduction

Whereas many recent publications reported good improvements in object learning [1–3], there is still the bottleneck that their learning databases have been manually and carefully constructed, cleaned and annotated, and are therefore limited in size because of the human time and cost required to do so. For example, the Corel database [2] contains about 60,000 images, the Caltech database [4] contains about 30,000 images for 256 objects and the IAPR TC-12 benchmark for the ImageCLEFphoto evaluation campaign [5] uses 20,000 photographs.

On the other hand, Web image search engines on the Internet (e.g. Google, Yahoo!, Ask) now give access to more than two billions of images. However, these images have been indexed only with the text surrounding the images in the web pages, which often leads to inaccurate annotation and subsequently brings a lot of noise when retrieving images. A quick study on the 50 first images of 50 queries using a simple word to retrieve animals or man-made objects showed that for Google and Yahoo!, the two most used image search engines, the average noise is about 50%: half of the images returned are not related to the query.

As an attempt to make Internet users label images from the Web, Carnegie Mellon University created ESP game [6] to make a game out of labeling. In order to increase the relevancy and objectivity of their annotations, two randomly selected players have to agree on the same keyword describing an image for it to be accepted. The ESP game has labeled about 30,000 images so far, which is still far away from the 2 billions of images over the Internet. Furthermore, the data are not made available to the public, so there is still a need for automatic grabbing of images from the Internet.

The same researchers later developed another game, Peekaboom [7], oriented toward manual segmentation of web images. One player is given an image and a label (that was manually attributed during the ESP game), and has to select the region of the image corresponding to that label while the second player tries to guess the label. The segmentation processed is stopped when the label is guessed. The same data as in the ESP game are used, so again, for now, there is no more than 30,000 images available.

Another issue in using directly images from the Internet is that even for those images which are relevant, further filtering and processing is needed if we want to use them for object learning:

- the position of the object in the image is unknown, and manual segmentation would be too long to do, so we need automatic segmentation,
- some images are relevant, but the object of interest can be too small to allow proper feature extraction, which makes them useless for learning,
- on the contrary, the photography can have been taken too close to the object, so that only a part of it is visible (Fig. 1).

In this article, we propose several automatic segmentation algorithms that are designed to segment web images, that is a group of

* Corresponding author. Address: CEA, LIST, Laboratoire d'ingénierie de la connaissance multimédia multilingue, 18 Route du Panorama, BP6, F-92265 Fontenay-aux-Roses, France. Tel.: +33 1 46 54 81 37; fax: +33 1 46 54 75 80.
E-mail addresses: chr.millet@gmail.com (C. Millet), isabelle.bloch@enst.fr (I. Bloch), patrick.hede@cea.fr (P. Hède), pierre-alain.moellic@cea.fr (P.-A. Moëllic).

0262-8856/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2009.06.005

images resulting from one given text query, based on the following hypotheses:

- The object is centered in the image. By center, we mean that most pixels of the object are contained in a window, centered in the image, and whose width and height are about one half or three quarters of the image width and height. This seems like a strong hypothesis, but it is actually verified by most relevant images.
- The object has well determined colors. Instead of segmenting a set of images corresponding to *dog*, we will consider images from the queries *German shepherd*, *golden retriever*, etc. that are subspecies of *dog*. This makes sense in the framework developed by Popescu [8], where he demonstrates that reformulating a query into multiple queries using its hyponyms in the WordNet hierarchy [9] gives better performances in terms of precision than the initial query.

We do not make the assumption that all images are relevant when segmenting, and try to deal with irrelevant images after the segmentation process. The capacity to reject irrelevant images using the segmentation results will be used to evaluate our segmentation algorithms. In order to do so, we rank the images according to some properties of their segmentation results: objects that are the largest, the closest to the center, and entirely contained in the image are favored. We then compute the precision on the first 20 images.

We first describe what queries we use to grab images from the Internet in Section 2 as well as some prefiltering we apply on images to remove cliparts. We then explain our segmentation algorithm (Section 5) as a combination of two other segmentation strategies: a segmentation that considers all the images to guess the colors of the object and those of the background (Section 3), and a method that segments each image individually by trying to extract a central object (Section 4). An algorithm to re-rank images given the segmentation results is developed in Section 6, and results are evaluated in Section 7, discussing on the precision of the 20 "best" images and the quality of the segmentation.

2. Grabbing images from the Internet

2.1. Which keywords to use?

There are basically three scenarios for grabbing images, related to the existence of a specific color for the query:

- The query is a natural object that is a leaf in the WordNet hierarchy (e.g. Granny Smith, toy poodle). This query is accurate enough to have one unique color, and is used as is for querying images.
- The query is a natural object that is not a leaf on the WordNet hierarchy (e.g. apple, dog). This object has hyponyms and we use all the object's hyponyms that are leaf nodes as queries, which corresponds to the previous scenario. The set of images for each hyponym is processed independently from the others, and all sets are eventually merged.
- The query is a man-made object (e.g. car, mug, house). Most man-made objects exist in many colors, and therefore have no specific color. In this case, we specify the color in the query, run queries for each color as in the first scenario, process each set of images independently, and eventually gather all results.

In this article, we will only consider queries that are leaves in the WordNet ontology, since any other query is, with our methodology, equivalent to querying for several leaves.

In order to improve precision, as detailed in [8], the category of the concept is added to the query. For example, the query *"golden retriever" dog* will be used to search for golden retrievers. This also helps disambiguating queries: *jaguar cat* and *jaguar car* are different concepts, and querying only for *jaguar* returns images of both concepts mixed together. The word we use for the category is an hypernym of the concept in the WordNet hierarchy, but is not always the direct hypernym, because this is not always a good choice to refine the query. With the above example of *golden retriever*, adding the direct hypernym would form the query *"golden retriever" retriever*, which in most (if not all) web image search engines returns the same images as the query *"golden retriever"*. Therefore, we choose generic names for category. These names are very limited and are chosen manually: e.g. *dog, cat, fish, horse, zebra*.

For each concept, we run one query for each synonym corresponding to that concept in the WordNet ontology and group all the resulting images in the same set. For example, *horned viper*, *cerastes*, *sand viper*, *horned asp* and *Cerastes cornutus* are all the same kind of snake according to WordNet. In practice, the number of images returned by a web search engine is limited to 1000, and we have decided to keep the first 300 images: we need enough images so that our re-ranking and selection of the first 20 images makes sense, and on the contrary, taking too many images increases the computational time and decreases the raw precision.

2.2. Removing cliparts

We decide to concentrate on photographs, and we therefore process the images to remove cliparts. What we mean by cliparts are computer drawn images that can easily be identified as such (excluding realistic rendering) and screen shots (see Fig. 2).

Images that contain both photography and clipart elements are considered as photographs by convention. These images are for example the photography of an object that has been displayed on a white background, or a photography to which a frame has been

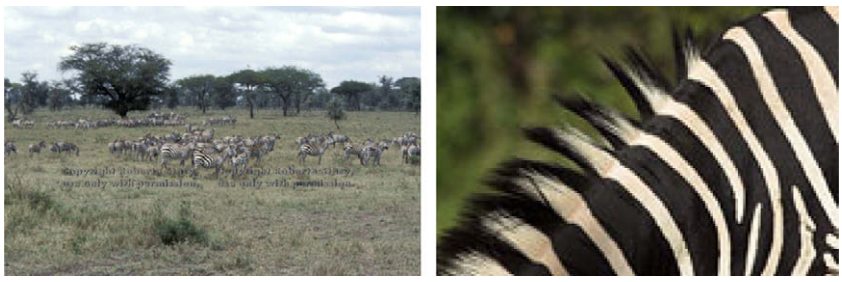

Fig. 1. Illustration of images that should be rejected if we consider using them for concept learning. In the picture on the left, the zebras in the pictures are too small for feature extraction, and in the picture on the right, we only have a partial view of the object.

added. Cliparts actually represent a significant part of the images resulting from our queries. A quick evaluation on some queries we considered in this article (Tables 1 and 2) shows that up to one quarter of the first 100 resulting images can be cliparts.

In order to detect cliparts, we convert the color image into greyscale and build its histogram by sampling the values between 0 and 255. We noticed that greyscale histograms of cliparts consist of several peaks that correspond to the colors used in the image, whereas the greyscale histograms of photographs tend to be more continuous, even when they have about the same number of colors. This is illustrated in Fig. 3.

The task of recognizing a clipart is therefore done by finding and analyzing the peaks of an image greyscale histogram H. We first look for the position p of the maximum of this histogram:

$$p = argmax_{x\in[0,255]}(H(x))$$

and then compute a standard deviation σ^2 around this point. Because of the extremities (0 and 255), we computed two standard deviations: one on the left side of the peak and one on the right side. If both sides can be computed, we consider the average.

$$\sigma^2 = \begin{cases} \sum_{x=p-5}^{p-1} r(x) & \text{if } p > 250 \\ \sum_{x=p+1}^{p+5} r(x) & \text{if } p < 5 \\ \frac{1}{2} * \left(\sum_{x=p-5}^{p-1} r(x), \sum_{x=p+1}^{p+5} r(x) \right) & \text{otherwise} \end{cases}$$

where

$$r(x) = \left(\frac{H(x)}{H(p)} * (x-p) \right)^2$$

If the standard deviation is small, that means we have a peak, and the image should be classified as a clipart. If the standard deviation is high, we classify the image as a photography. In order to deal with the images that have both photography and clipart elements, we propose to divide the image into 16 parts: four horizontally and four vertically. For example, let us imagine an image with a black frame all around the image. If we consider the image as a whole, the greyscale histogram will have a peak corresponding to that black color, and the image will be classified as a clipart. If we first divide the image into 4×4 parts, the 12 parts that are on the border of the image will have that peak, but the four parts in the center will have a histogram corresponding to that of a photography, and will be classified as such. The whole image is eventually classified as a clipart if all 16 parts have been classified as cliparts, and as a photography otherwise.

Table 1
Number of cliparts among the first 100 images for several queries on Google Image Search.

Bald eagle	11%
Bengal tiger	24%
Castor canadensis	19%
Cerastes	4%
Common dolphin	20%
Common zebra	7%
Dromedary	26%
Mean	15.9%

Table 2
Comparison of the precision (in %) computed on the first 20 images returned by our algorithm and by Yahoo! image search, on which our algorithm is based. There are two columns for Yahoo!, *all* takes into account all relevant images while *good* only considers the images where the object occupies at least 10% of the image, that is the kind of images that are of a quality comparable to those returned by our re-ranking algorithm. Twenty queries are considered: 10 animals and 10 man-made objects. We observe an average increase of about 20% for animals and for man-made objects when comparing images of similar qualities.

Animals	Yahoo!		Re-ranking	Man-made	Yahoo!		Re-ranking
	All	Good			All	Good	
Bald eagle	90	60	100	Black shirt	55	55	90
Bengal tiger	100	100	100	Blue mug	65	65	80
Bull	75	60	75	Boeing 777	60	60	90
Cerastes	75	70	80	Eiffel tower	100	95	80
Common dolphin	90	65	90	Fire engine	50	50	85
Common zebra	100	80	100	Red bottle	40	35	65
Dromedary	70	60	80	Sun glasses	75	40	85
Ewe	90	65	100	White Porsche	55	55	95
German Shepherd	95	90	100	Wood table	75	75	80
Monarch butterfly	90	70	100	Yellow Ferrari	70	70	95
Average animals	87.5	72	92.5	Average man-made	64.5	60	84.5

In practice, we found that most cliparts have a standard deviation σ^2 of 5 or less, whereas it is 40 or more for most photographs. We evaluated the proposed algorithm on a database of 11,252 photos and 5402 cliparts from the Internet and obtained the best results using 15 as a threshold: an image is a clipart if all its 16 sub-images verify $\sigma^2 < 15$ and a photography otherwise. Using this threshold, 99.78% of the photographs and 93.02% of the cliparts were correctly classified.

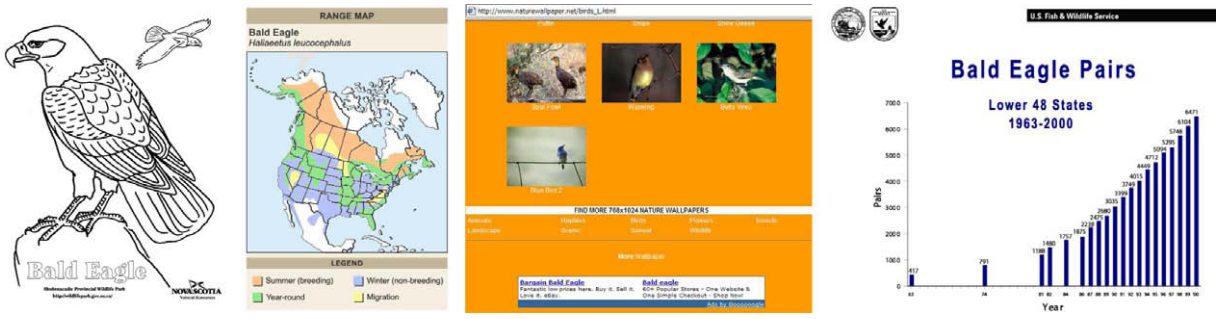

Fig. 2. Examples of cliparts found for the query "bald eagle" on Google Images. Four common kinds of cliparts are shown here: (from left to right) a hand drawn representation of the animal, a map of where the animal can be found, a screenshot of a website proposing information related to that animal, and a representation of statistics from scientific studies on the animal.

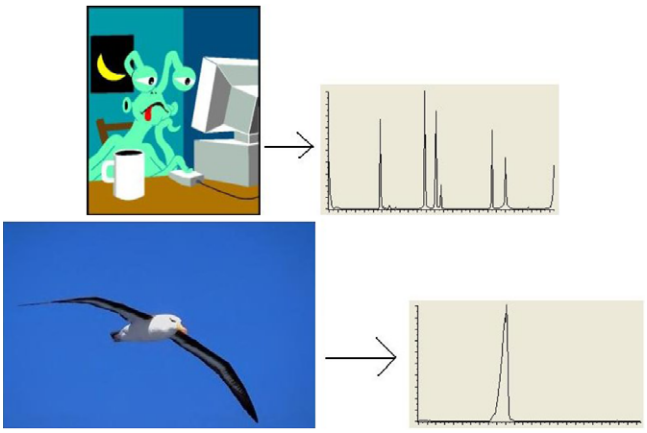

Fig. 3. Comparison of the greyscale histograms of a clipart and a photograph that have about the same number of colors.

The results are similar to what is reported in the literature [10] on a different test database of about 5000 images, but these methods use machine learning and various features whereas our method is designed to be fast, using a very simple feature and a single threshold.

We will now describe and discuss the three fully automatic segmentation algorithms that we propose in this article to deal with the collected images in the following Sections 3–5. A schema of the three proposed segmentation algorithms is shown in Fig. 4.

3. Global segmentation: segmentation considering all images

3.1. Previous experiments

In [11], we considered using object color names to automatically locate the object in the image. Given the name of an object, the names of the colors in which the object can appear were auto-

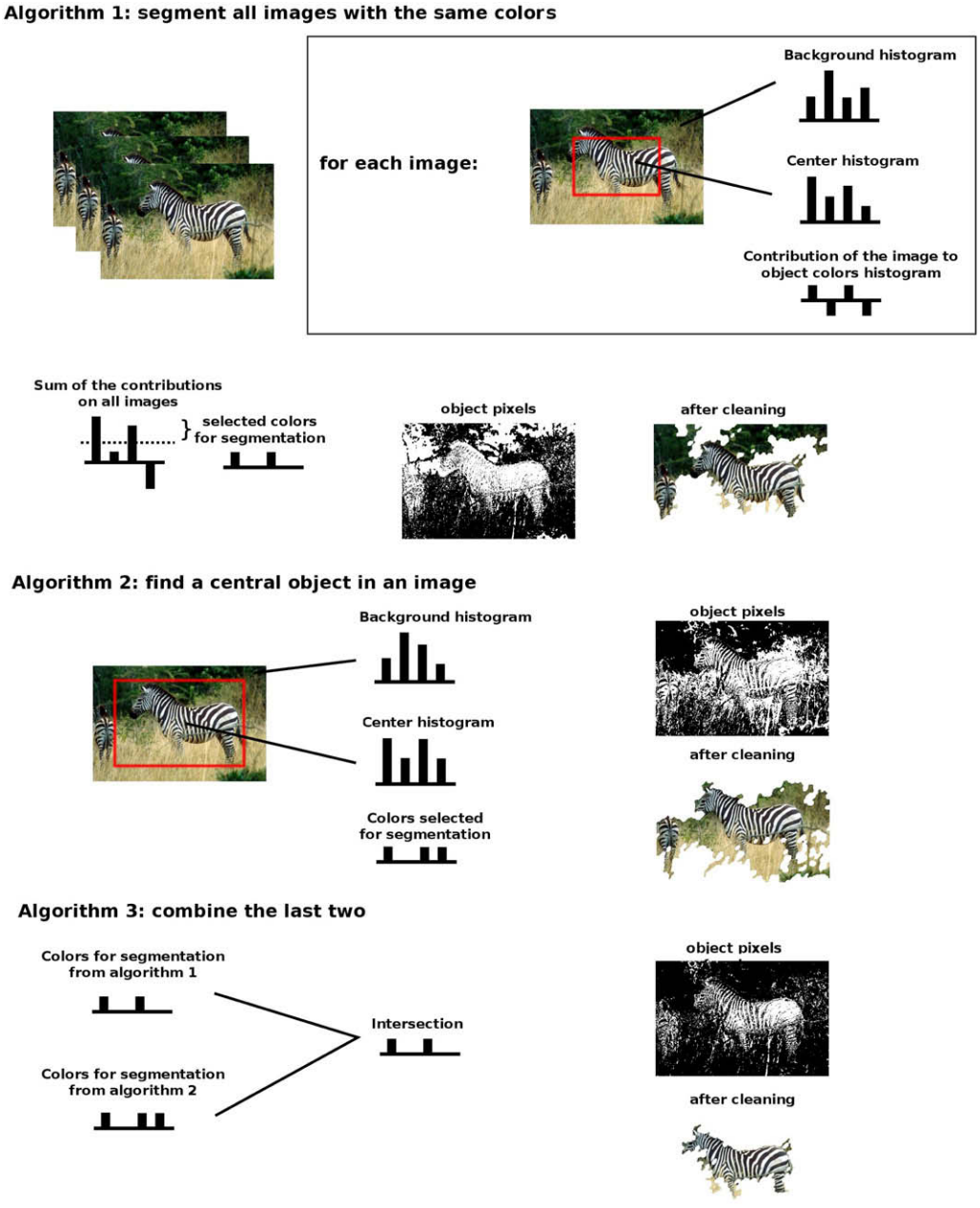

Fig. 4. Schema of the method proposed in this article: algorithm 1 is described in Section 3, algorithm 2 in Section 4 and the final segmentation, algorithm 3, is explained in Section 5.

matically determined using either a text based method, finding what are the most co-occurrent colors in a text corpus – we used the web for our experiments – or with an image based method, downloading images of that object, and finding the most common colors that appear in the center of the image.

We then defined a function that matches color names and pixel values in the HSV color space. Then, an image can be segmented using one or more color names. For example, it is possible to segment the image of a zebra using both colors *white* and *black*.

The main issue of the algorithm based on the names of colors is that the use of the names limits the possibilities of the segmentation. For example, this algorithm cannot segment any brown animal that appears on a brown background, as shown in Fig. 5. They are different shades of brown, but for this algorithm, they all belong to the single color name *brown*.

It would be possible to define more color names, such as *dark brown*, *light brown*, *reddish brown* and associate them with pixel values in the HSV space, but it seems difficult to automatically obtain with such accuracy the colors of an object. Moreover, with only 11 colors it is not always easy to determine the colors of an object. For example, the color of a dolphin seems to be some color between blue and grey, but blue is also the color of the water they are in.

We worked on extending that algorithm so that it does not use color names anymore, but directly pixel values. Doing so, the text based method to know the color of an object cannot be used, but the image based method still can be, which is the one of the two methods that performed best according to our experiments in [11].

The following algorithm solves the second type of problem (stop using color names) while the first one (Fig. 5) is addressed in Section 4.

3.2. Algorithm

The first algorithm consists in taking into account all the images grabbed from the Internet corresponding to the same query in order to identify which pixels are object pixels and which ones are background pixels, in order to be able to locate the object of interest in any image supposed to contain that object. The segmentation algorithm described here is similar to the one based on color names [11], but instead of being limited to 11 colors, we have 125 colors and it can easily be extended to more colors. Another difference is that background pixels are also taken into account as negative contributions.

It consists of the following steps:

(1) Each plane of the RGB color space is quantized into five values. Therefore, we work with 125 colors instead of the 11 colors previously mentioned.

Fig. 5. Example of a bad segmentation when using color names: the whole image is considered as the result of the segmentation using the colors *brown*. The Bengal tiger and the background in the image are seen by the algorithm as the same color name: *brown*.

(2) A central window is defined as a window whose width and height are half of the image width and height.
(3) For each image, we build two 125-bins RGB histograms: *histocenter* for the pixels contained in the central window and *histoborder* for the pixels contained outside of this window.
(4) Both histograms are normalized by the number of pixels considered (i.e. the surface) so that they can be compared.
(5) For each possible (r,g,b) value, we compute a score $S(r,g,b)$ over all images that is increased by one when, for an image, $histocenter(r,g,b) > histoborder(r,g,b)$, and decreased by one otherwise.
(6) Eventually, a (r,g,b) value is considered as an object color if $S(r,g,b) > \frac{\max(S)}{5}$, and as a background color otherwise.
(7) The set of object colors pixels is then cleaned as described thereafter to keep only one region in the image corresponding to the object.

Cleaning an image is a process that allows us, from a set of points of the image that have a given colors to obtain a connected region with a smooth shape and no hole. This is done with several steps, using mathematical morphology:

(1) Remove noise or small thin objects, using an opening by a structuring element of size 1.
(2) Apply a closing by a structuring element of size 5 to merge close object regions together.
(3) Select the largest region.
(4) Remove holes, defined by background pixels entirely surrounded by object pixels, based upon the assumption that the objects have no hole, which is the case for most objects.

This process is illustrated in Fig. 6.

3.3. Results and discussion

In comparison with our previous work on segmenting an object in images using color names [11], the resulting segmentation is more accurate. Using color names is meaningful for us, but for the computer it limited the number of colors to consider to 11. Among those 11 colors, typically 1 or 2 where selected as "object colors" used to segment an object, but it was not clearly determined how to know if 1 or 2 colors were to be considered. The new algorithm we presented here considers 125 colors, and could easily be extended to more. The number of colors to consider as "object colors" is automatically determined (step *vi*), usually above 10 but that number depends greatly on the object being studied.

Fig. 7 shows an example of the limitations caused by the use of color names. The small zebra is considered as being composed of mainly light brown and dark brown pixels.

However, as expected, this algorithm still has an issue that appeared also in our previous algorithm using color names [11]: it cannot distinguish easily an object in an image where the pixels of its background are defined as object pixels because the same color was often found in objects of other images. This is illustrated in Fig. 8: the color of the brown background is also found as being a possible color for Bengal tigers in many other images and therefore considered as part of the object.

The algorithm that we will introduce in the next section concentrates on solving that issue.

About the parameters, in the fifth step of the segmentation, we tried introducing a factor $k > 1$, to increase S only if $histocenter(r,g,b) > k * histoborder(r,g,b)$ and decrease it if $histocenter(r,g,b) < (1/k) * histoborder(r,g,b)$ in order to ignore the colors for which a pixel is not clearly classified as object or background, but this have in fact very little consequences on the results.

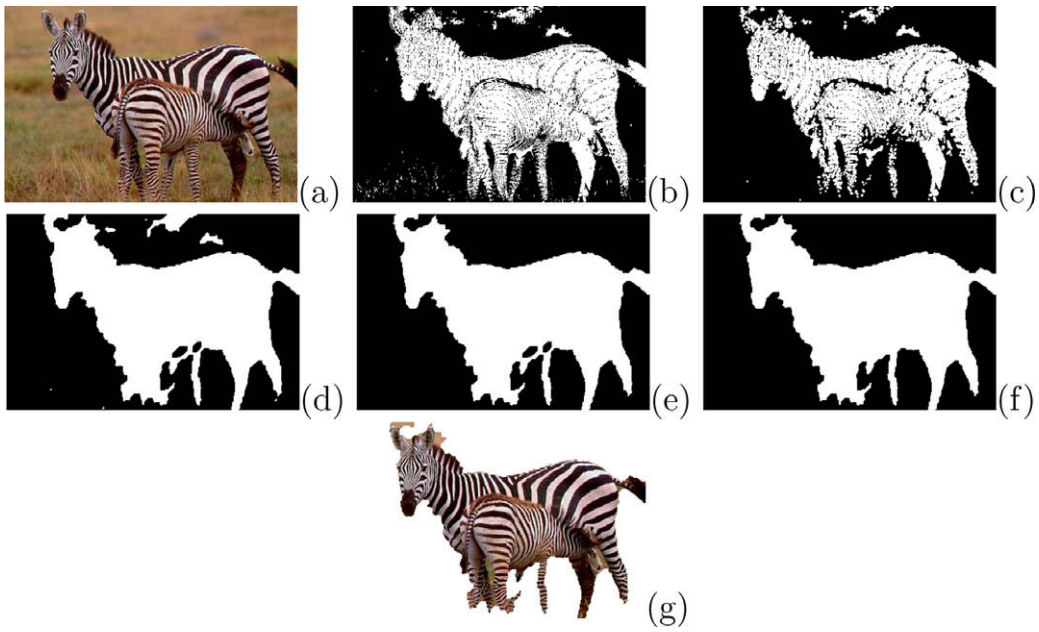

Fig. 6. Cleaning a set of points matching the colors used for the segmentation to obtain a connected region. (a) Original image. (b) Pixels identified as "object pixels". (c) Opening on (b). (d) Closing on (c). (e) Keeping only the largest region from (d). (f) Filling holes in the object. (g) Corresponding final cleaned segmentation.

In the sixth step, using $S(r,g,b) > \frac{\max(S)}{5}$ instead of $S(r,g,b) > 0$ to decide which colors are considered as object colors is another way to ignore colors that are weakly classified as object colors, and have more visible effects. Taking $S(r,g,b) > 0$ as a threshold would mean we consider that the number of object colors is potentially equal or superior to the number of background colors. It is not restrictive enough and led to parts of the background being considered as objects. Using a positive threshold is more restrictive on the number of object colors. Making it depend on $\max(S)$ instead of the number of images (the biggest value that S can reach is the number of images) ensures us that we keep at least one color. We tried several threshold valued, and found out that $S(r,g,b) > \frac{\max(S)}{10}$ or $S(r,g,b) > \frac{\max(S)}{5}$ (depending on the objects) offered a good compromise whereas $S(r,g,b) > \frac{\max(S)}{2}$ lacked some tolerance. A comparison of the effects of these different thresholds is shown in Fig. 9.

We notice that for $S(r,g,b) > 0$ and $S(r,g,b) > \frac{\max(S)}{10}$, most of the pixels on the zebras have been correctly recognized as object pixels, but there is also some noise from the background, for example between the legs. For $S(r,g,b) > \frac{\max(S)}{5}$, there is less noise, and what is remaining will be removed by the cleaning algorithm, but some of the white stripes are missing (they will be recovered with the opening of the cleaning algorithm). With the threshold $S(r,g,b) > \frac{\max(S)}{2}$, there is almost no noise left, but the missing stripes are more visible, and will not be recovered with the cleaning phase. Eventually, we decided empirically to use $S(r,g,b) > \frac{\max(S)}{5}$ as a threshold for all objects.

Fig. 7. Example of the segmentation of the image in Fig. 6 with the algorithm from [11] using colors black and white. The pixels of the small zebra appear brown (light brown for what we see as white and dark brown for what we see as black) and are not considered as object pixels. Extending the definitions of the colors black and white to include such pixels causes too broad segmentation results on other images for other objects.

Fig. 8. Result of the global segmentation algorithm for an image of *Bengal tiger*. This algorithm has the issue that it cannot segment objects when the color of the background is also a common color for the object, in other images.

4. Individual segmentation

In this section, we propose to deal with the main issue from the previous algorithm: how to segment an object in an image where the object colors and the background colors are close to each other? With the previous algorithm, it is difficult to segment a dark brown object on a light brown background, if the light brown color is the color of the object in many other images and has thus been identified as object color.

4.1. Algorithm

The algorithm we propose here tries to segment a central object in the image, regardless of what should be the colors of the object. It aims particularly at being able to segment correctly images where the colors of the object and of the background are close to each others, such as the image in Fig. 8, for which the segmentation algorithm developed in the previous section fails to separate the object from its background.

The main difference with the previous algorithm is that this one considers images individually and uses only one image to learn the difference between object colors and background colors, whereas the previous algorithm determined object colors

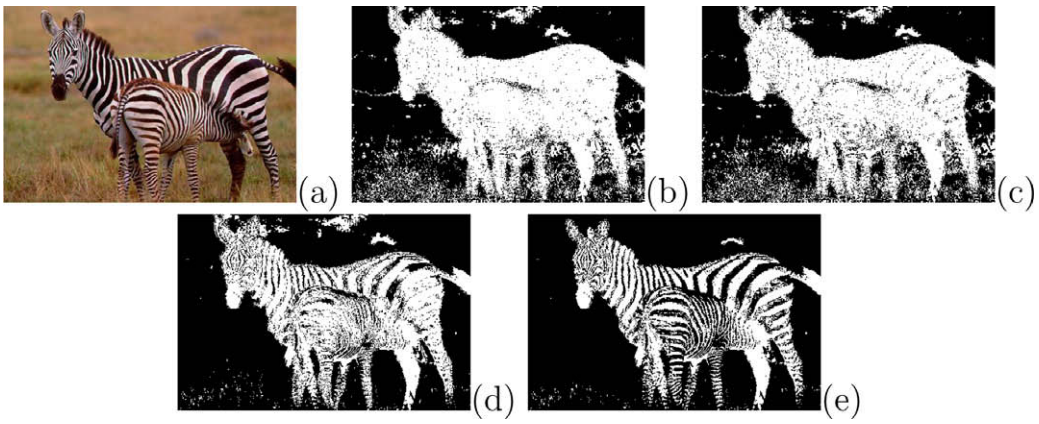

Fig. 9. Variation of the segmentation for different values of the threshold, and the number of colors considered as objects, out of the 125 colors from the RGB quantization. (a) Original image. (b) $S(r,g,b) > 0$: 37 object colors. (c) $S(r,g,b) > \frac{\max(S)}{10}$: 12 object colors. (d) $S(r,g,b) > \frac{\max(S)}{5}$: 8 object colors. (e) $S(r,g,b) > \frac{\max(S)}{2}$: 5 object colors. On this particular example, the segmented region in (b) and (c) are very close because the 25 additional colors in (b) are in very small quantity in this image. The difference is more obvious on other images, where it is often a part of the background that is considered as object with $S(r,g,b) > 0$ but background with $S(r,g,b) > \frac{\max(S)}{10}$.

and background colors on the basis of all the images corresponding to the same concept. Since the object colors were determined on several images, it was not problematic if for some images the object was not very well centered, as long as in average, most objects are centered. Therefore, what we called the central window was relatively small: only half of the image width and height. On the contrary, in the algorithm we propose in this section, there is no average over several images, and it is therefore better to consider a bigger central window to find the objects. Once these object and background colors are determined, the segmentation process itself is the same.

The complete algorithm consists of the following steps:

(1) Quantify each plane of the RGB color space into five values.
(2) Define a central window, as a window whose width and height are three quarters of the image width and height.
(3) Build two 125-bins RGB histograms: *histocenter* for the pixels contained in the central window and *histoborder* for the pixels contained outside of this window.
(4) Normalize both histograms by the number of pixels considered (i.e. the surface) so that they can be compared.
(5) Classify each pixel according to its quantized (r,g,b) value: a pixel is considered as an object pixel if $histocenter(r,g,b) > histoborder(r,g,b)$, and as a background pixel otherwise.
(6) Eventually, the resulting binary image is cleaned as described in Section 3.2.

4.2. Results and discussion

First of all, our main goal is reached, since this algorithm can segment an object when the background has a close color. For example, a brown object on a (different) brown background can be correctly located (Fig. 10), whereas this was not always possible with the segmentation algorithm from the previous section (Fig. 8).

In our first experiments, we used a window whose width and height were only half of the image width and height, as used for the previous algorithm (Section 3). However, for many images, there are significant parts of the object that are outside of this window, so that colors of the object appeared both inside and outside of the window, and were not clearly identified as belonging to the object. Increasing the window size gave better results for most images. A comparison of the segmentation results obtained with two different sizes of window is shown in Fig. 11.

Fig. 10. Example of an improvement over the previous algorithm (see Fig. 8). The fact that the brown background can also be the color of a Bengal tiger does not matter for that algorithm.

It is questionable whether the best is to miss some part of the object but include less background in the object, which is usually what happens with a small central window, or to have a better chance not to miss any part of the object, with the risk of obtaining more background identified as object, with a larger central window. Though, in the following, we will be intersecting the result of this segmentation with the result from the previous segmentation algorithm (both before the cleaning post-processing), and in this case it is better to concentrate on having as many object parts as possible, which makes us prefer the second option (a larger central window).

The main issue of this algorithm for our application, as we stated before, is that it will segment any central object regardless of the fact that the object is supposed to correspond to a given query and be similar to other images from the same query. The consistency between the various images grabbed for that query should be taken into account in order to identify irrelevant images, as we did in the first segmentation algorithm we presented, taking all images into account to define object colors and background colors. Combining the two algorithms would allow to take advantage of both.

5. Combining global and individual segmentations

The idea now is to combine the algorithm from Section 3 that uses all the images to determine the object colors and find objects in images that have this color with the algorithm developed in Section 4 that tries to find a central object in any image, able to deal with the case where the background has a color close to that of the object.

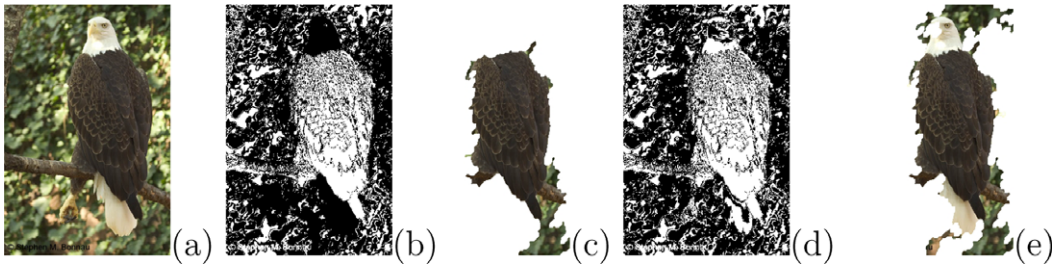

Fig. 11. Effect of the size of the central window on the segmentation. (a) Original image. (b, c) Pixels identified as objects with a window whose width and height are half of the image width and height, and the resulting segmentation after cleaning. (d, e) Pixels identified as objects with a window whose width and height are three quarters of the image width and height, and the resulting segmentation after cleaning. In (b, c) only the brown part of the eagle is identified as the object. In (d, e) some white parts of the eagle (the tail and part of the head) are also included in the object, but with the drawback of obtaining more background, on top of the image.

5.1. Algorithm

It is done by intersecting the previous two algorithms, just before the cleaning step. It can be written as a single algorithm in this form:

(1) Each plane of the RGB color space is quantized into five values.
(2) A small central window W_S is defined as a window whose width and height are half of the image width and height, and a large central window W_L whose width and height are three quarters of the image width and height.
(3) For each image, we build two 125-bins RGB histograms: $histocenter_S$ for the pixels contained in the central window and $histoborder_S$ for the pixels contained outside of the window W_S and also compute $histocenter_L$ and $histoborder_L$ with the window W_L.
(4) Each histogram is normalized by the number of pixels considered (i.e. the surface) so that they can be compared.
(5) For each (r,g,b) value, we compute a score S that is increased by one when, for an image, $histocenter_S(r,g,b) > histoborder_S(r,g,b)$, and decrease by one otherwise.
(6) Eventually, a (r,g,b) value is considered as an object color if $S(r,g,b) > \frac{\max(S)}{5}$ and $histocenter_L(r,g,b) > histoborder_L(r,g,b)$. It is considered as a background color otherwise.
(7) Each pixel is classified as object or background.
(8) Eventually, the resulting binary image is cleaned as described in Section 3.2.

This is equivalent to intersecting the object pixels obtained from the two algorithms after the pixel classification step, and before the cleaning post processing.

5.2. Results and discussion

We tried intersecting the two segmentation algorithms before or after the cleaning that uses mathematical morphology, and it appeared that it is better to merge them before. For example, let us consider the irrelevant image in Fig. 12, an image returned for the query *castor canadensis* (beaver).

The two segmentation algorithms play their roles: the central object segmentation finds that blue is a central color, and brown is a background color, and therefore identifies the blue ball as the central object. On the contrary, the algorithm that uses all the images to determine the object colors finds that brown is an object color whereas blue is not. Cleaning the image includes a step removing holes, and for this segmentation, this leads to considering the whole image as the object, since the whole image is surrounded by a border that has object colors. The intersection of both segmentations is then equal to the central object segmentation, giving an object which is not of the right color. Intersecting the two segmentations before the cleaning step gives a brown ob-

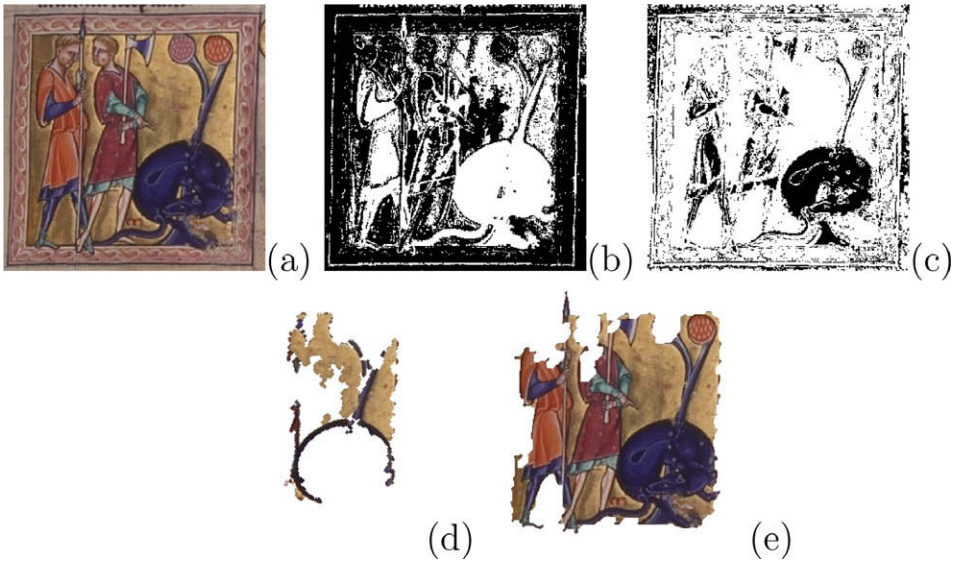

Fig. 12. Comparison of merging before and after cleaning, for an irrelevant image of *beaver*. (a) Original Image. (b) Object pixels found by the central object segmentation algorithm. (c) Object pixels found by the algorithm that considers all images. (d) Resulting segmentation if the merging is done before the cleaning step or (e) after. It is better to have a smaller image (d), since we will then favor objects with large surfaces during the re-ranking and filtering step.

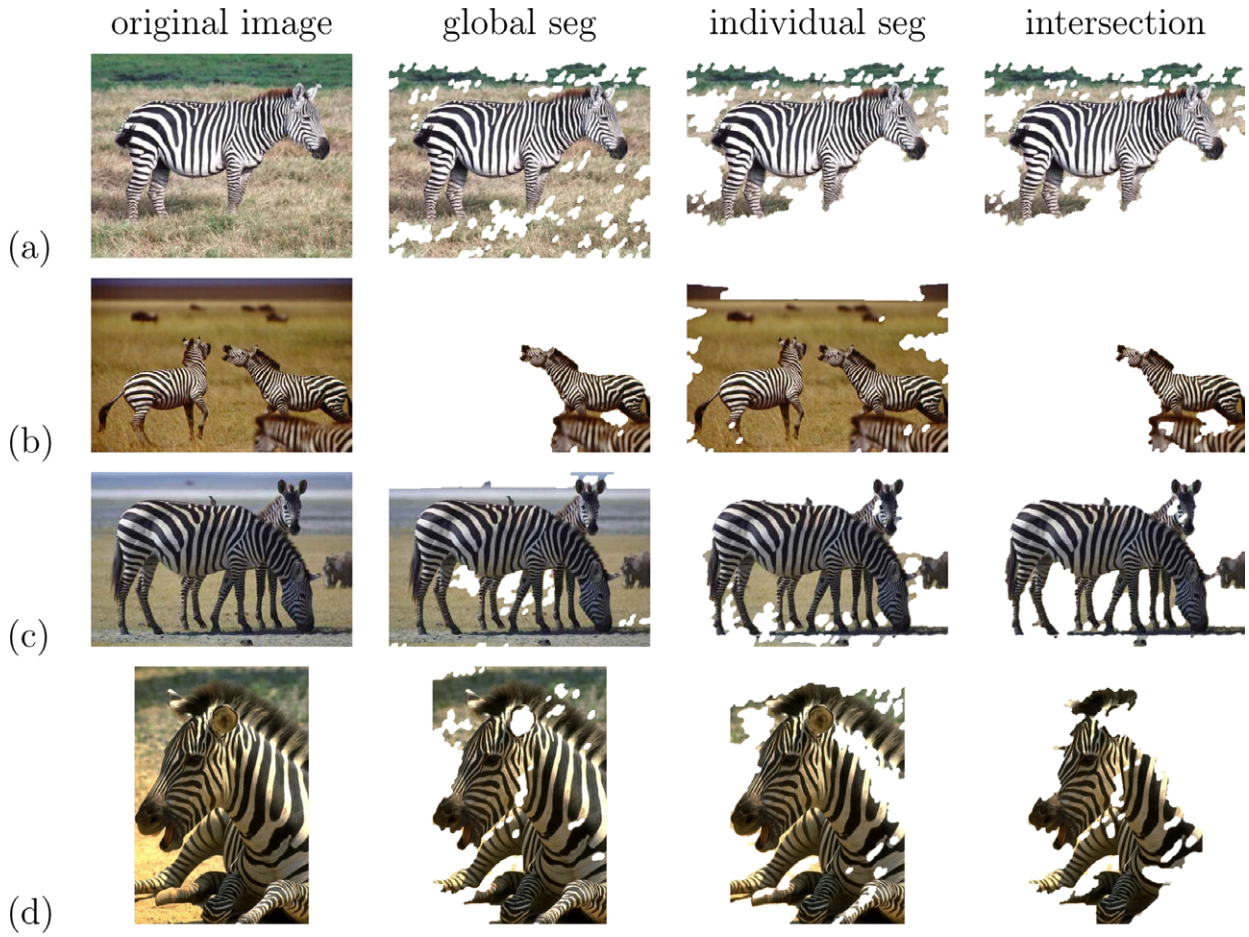

Fig. 13. Comparison of the merged segmentation results with the global and the individual segmentations. (a) The global segmentation has kept the grey grass in the segmentation and the intersection is equal to the individual segmentation which is included in the global segmentation. (b) The individual segmentation found that the grass is centered in the image, and the global segmentation is better since it has the knowledge that the grass color is not a color of the zebra. In this case, the intersection is close to the global segmentation. (c) Both individual and global segmentations include a part of the background, but not the same part. Therefore, the intersection gives a segmentation that is better than both. (d) In this example, the individual segmentation is the best. The intersection does not contain any background part, but a part of the zebra is missing.

ject, which is more consistent since we are looking for a beaver. This example has been chosen to show well the effect of changing the moment when we do the intersection. In most cases, the effect is not that visible. We noticed, though, that merging the segmentations after the post-processing tended to find an object with parts of the wrong colors that did not appear if we merged them before.

In most cases, one of the two segmentations that are merged with this algorithm is included in the other, but the smaller segmentation may be the individual or the global one, depending on the image. Therefore, for these cases, the consequence of intersecting the two segmentations is in fact to select the one of these two segmentations that is the smaller, and often the better. In other cases, both segmentations contain a part of the background, but two different parts, and intersecting them allow to obtain a segmentation that is better than the two others. This is illustrated in Fig. 13.

We will try in the next section to define criteria to re-rank images, allowing us to tag such image as irrelevant.

6. Re-ranking images

We extend the criteria that we developed and analyzed in [11] to re-rank images regarding the shape and size of the segmentation obtained. For a given segmented object in an image, we define several variables:

- v is the region surface divided by the image surface. It is proportional to the region size,
- B is the number of image border pixels included in the object region divided by the total number of image border pixels. It equals 0 if the object is totally included in the image, and increases if the object touches the border of the image, meaning that maybe there is only a part of the object in the image, as for example in the right image in Fig. 1.

With v and B, we compute the following Σ score:

$$\Sigma = (1 - B) \times f(v) \quad \text{with } f(v) = \begin{cases} 1 & \text{if } 0.2 \leqslant v \leqslant 0.6 \\ \frac{v}{0.2} & \text{if } v < 0.2 \\ \frac{1-v}{0.4} & \text{if } v > 0.6 \end{cases}$$

Images are then sorted by decreasing order of Σ. An object with the highest value of Σ is thus an object whose size is between 20% and 60% of the image size, and which is totally surrounded by a background, that is does not touches the borders of the image. In Millet et al. [11] we also included a criterion that gave a better score to objects close to the center, but finding a centered object is now part of the segmentation algorithm itself, and therefore is less relevant for sorting images.

For the shape criterion B, the bounding box of the region can be used rather than the border of the image, leading to better results when there are many images to which frames were added. It only

causes worse results if the object has straight horizontal or vertical lines in its shape, which usually does not occur in natural object that we studied. If we want to extend our algorithm to man-made objects with straight horizontal and vertical lines such as building, monitors or shelves, implementing an algorithm to identify and remove frames would be the best solution.

There is still another issue in the above re-ranking strategy. Let us suppose that we are re-ranking for example images of zebras. The colors used for the segmentation are basically black and white colors. With the above re-ranking, a black region, a white region and a zebra of the same shape will have the same rank. In order to give a better rank to regions that have both white and black in a given proportion, we propose the following:

(1) Compute the median 125-bins RGB histogram H_M of all segmentation results. It is obtained by computing the median value for each bin of the histogram among all the images.
(2) For each segmentation, compute the color similarity C_s, as the histogram intersection between the histogram H_I of the image and the median histogram H_M:

$$C_s = \sum_{k=1}^{125} \min(H_I(k), H_M(k))$$

We then define the ranking score $R_s = C_s \times \Sigma$ and consider that the most relevant images are those with the highest ranking score.

7. Results

In this section we evaluate only the final segmentation algorithm that is a combination of the two others and its associated re-ranking. It is difficult to provide a subjective evaluation of this algorithm, since we have to evaluate both the quality of the segmentation, and the precision of the "best" images after re-ranking. Therefore, we first discuss the first twenty images for three queries shown in Figs. 14 to 16. We then quantify separately the performances of the re-ranking and the segmentation on twenty queries: 10 animals and 10 man-made objects.

Two natural objects that are not uniform in colors are shown here: *zebra* (Fig. 14) and *Bengal tiger* (Fig. 15). Traditional segmentation algorithms usually fail on the segmentation of such objects. Our segmentation has also been tested on a man-made object: *yellow Ferrari* (Fig. 16). The results, both in precision and segmentation, are very good for the animals. For the Ferraris, the precision of the selected images is good, only the body of the car has been kept in the segmentation. Wheels and windshields are missing because their black color has been mostly observed in the background. On this particular example, results could be improved by taking the convex hull of these objects.

This algorithm works better in average than the one proposed earlier in [11]: the quality of the segmentation is improved since this algorithm can isolate better the object from its background, even if their colors are close. The precision obtain after re-ranking is also better.

However, for queries that come with too many noise when grabbing the images on the Internet, it shows poor performances. For example, we tried the query *banana fruit* expecting to obtain a group of images corresponding to yellow bananas. The 20 "best" images we obtain are shown in Fig. 17.

In fact, in the 100 first images from Yahoo! Image Search, after removing cliparts, there are only 20 images that contain a yellow central banana, which means that the noise is about 80%. Other images are mainly about banana trees (or other trees), pictures of several fruits together, or products derived from banana. For the segmentation, yellow has been identified correctly as an object color, but green, red and orange as well.

Our algorithm was based on the assumption that the noise is 50% or less. For *yellow Ferrari*, on the first 100 images, about 43% does not show an image where one can recognize a yellow Ferrari. The noise for the queries *common zebra* and *Bengal tiger* is much less: around 10%. Therefore, choosing the right keywords to have a good set of images to start with is a fundamental step that should not be underestimated. In that sense, approaches like the ESP game that we described in the introduction might prove useful in the future.

7.1. Evaluation of the re-ranking

A picture is considered as relevant if the queried object can be identified in the picture. Objects such as toy, sculptures or paintings where the represented object is easily identified are also judged relevant. Images were the object cannot be identified, whether it is too far, too blur or the part shown is not characteristic of the object, go in the irrelevant category. For example, pictures of insides of the aircraft are not considered as relevant for the query *Boeing 777*.

Fig. 14. The 20 first segmentation results for the query *common zebra*. The precision is 100%, the zebra shape has been found accurately in most images.

Fig. 15. The 20 first segmentation results for the query *Bengal tiger*. The precision is 100%: all images are related to Bengal tigers, and we have both head images and body images.

Fig. 16. The 20 first segmentation results for the query *yellow Ferrari*. The color has been added to the query, as explained in Section 2. The precision is 95%, the irrelevant image is identified by a red frame.

First, let us remark that the precision of the images returned by Yahoo! is already higher than the 50% announced in the introduction. This is a consequence of the way we ask queries, as explained in Section 2. If we consider all images from Yahoo! as relevant, without taking into account their qualities as possible images used for learning, we measure an average increase in precision of 5% for animals and 20% for man-made images. However, the aim of our re-ranking algorithm is to select the best images for a learning database, that is images where the object's size is sufficient to allow feature extraction. It is therefore fairer to compare the precision of our algorithm with the images from Yahoo! where the object occupies at least 10% of the image (our re-ranking algorithm is set to favor objects whose size is between 20% and 60%). Considering only such images in Yahoo! as relevant, the increase in precision becomes 20.5% for animals and 24.5% for man-made objects.

The next part, on the evaluation of the segmentation, gives some statistics on the usual size of the objects in the first 20 selected images and what part of it are correctly segmented.

7.2. Evaluation of the segmentation

In this work, we are in the objective of using the segmentation results to build a good database for object classification. Therefore, we are not aiming at obtaining a perfect segmentation, but rather a segmentation that does not contain too much background in it, and enough parts of the object to allow proper feature extraction, but not necessarily the whole object. We can however evaluate our segmentation algorithms as if the objective was to segment perfectly the objects, in order to see how it would perform in such task.

In order to evaluate the quality of the segmentation, we manually segmented some images and compared them with the automatic segmentation. The manual segmentation has been done with SAIST (Semi-Automatic Image Segmentation Tool), a software developed by the PRIP laboratory in Vienna [12]. It computes a user-guided marker-based watershed segmentation.

For the ground truth, we selected any pixel that belongs to the queried object as object, and the others as background. That is, if two objects are present in the image, the two will be segmented, even though our segmentation algorithm is designed to ideally find only the larger of the two. If an object is occluded, for example a horse occluded by a saddle, the occluding object is considered as background, altering the shape of the occluded object.

Since we cannot possibly evaluate the segmentation on all the 300 images, because of the time it takes to manually segment the images, we have chosen to evaluate the segmentation on the relevant images among the first 20 images selected by our re-ranking algorithm.

There are two measures to consider when evaluating the segmentation of an object: the proportion of the object pixels that are correctly retrieved M_1, and the proportion of pixels that are correct in the automatic segmentation M_2. If we call S_A the region identified as the object by our algorithm and S_T the ground truth, we have:

$$M_1 = \frac{surface(S_A \cap S_T)}{surface(S_T)}$$
$$M_2 = \frac{surface(S_A \cap S_T)}{surface(S_A)}$$

The two measures are strongly linked. We can easily have $M_1 = 100\%$ by considering the whole image as the automatic segmentation, but then M_2 equals (only) to the surface of the object divided by the surface of the image. Therefore, the two measures should be represented together. Their values for our 20 test queries are shown in Fig. 18.

We see that except for some queries, we can expect in average between 70% and 90% of correct pixels (M_2) in the automatic segmentation, while having a retrieving accuracy (M_1) also between 70% and 90%.

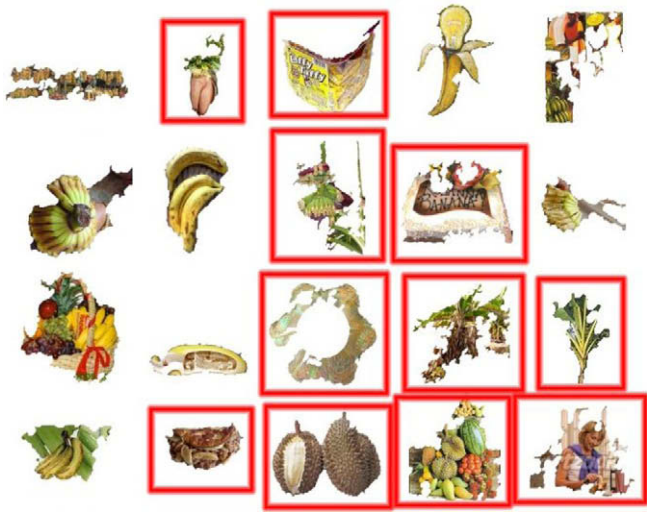

Fig. 17. The 20 first segmentation results for the query *banana fruit*. A red frame shows the irrelevant images. There were not enough yellow banana in the images to allow the algorithm to identify yellow as the main color of interest. That happens with queries where the proportion of irrelevant images from the Internet image search engine is too high.

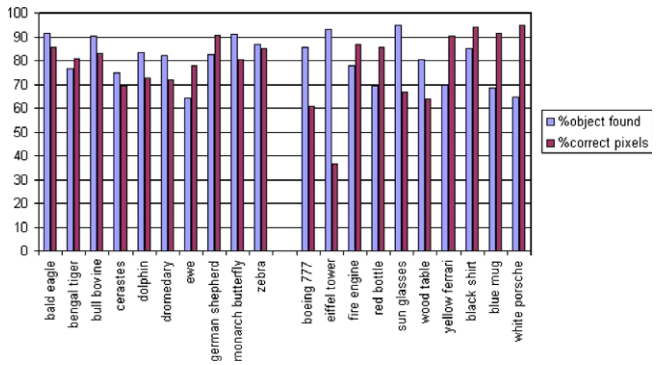

Fig. 18. Segmentation results showing the proportion of the object pixels that are correctly retrieved and the proportion of pixels that are correct in the automatic segmentation for 20 queries: 10 animals and 10 man-made objects.

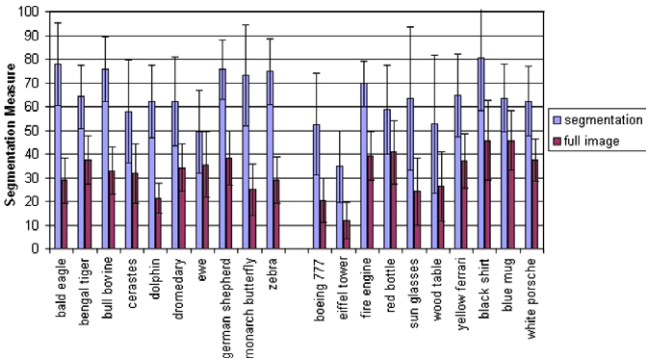

Fig. 19. Evaluation of the accuracy of the segmentation on 20 queries.

Deciding whether it is better to have a high M_1 or a high M_2 depends on the considered application. If one considers using the segmentation results for example for learning, then it is better to have as less noise as possible in it, that is to maximize M_2. In order to evaluate, we have decided to use a measures that takes both into account the rate of pixels correctly retrieved and the proportion of the retrieved pixels that are correct. This measure is also often used in segmentation evaluation:

$$M_S = \frac{surface(S_A \cap S_T)}{surface(S_A \cup S_T)}$$

We compare it in Fig. 19 with the score that we would obtain if we kept the whole image and considered it as the segmentation. This score also represents the size of the objects in the image.

The obtained results are good: the measure goes on average from about 30% for the full image to about 60% with our segmentation algorithm. Specifically, it works well with objects with two colors, be it stripes (*zebra*, *Bengal tiger*) patches (*monarch butterfly*) or two clearly separated colors (textitbald eagle, *German shepherd*). The task is more difficult for objects that have the same color than their environment (*cerastes*, *dolphin*, *dromedary*). We notice better results in general for animals than for man-made objects. This is mainly because animals tend to have less variations in terms of colors than man-made objects, making them easier to identify when comparing all the images.

Eiffel tower is the query with the worst score. It is also the query for which the objects have the smallest size. If we compare this result with Fig. 18, we understand that the segmentation results always contain most of the object, but the object represents only 40% of the segmented region, meaning that the background occupies the other 60%. The main reason why it did not perform well with *Eiffel tower* is because our re-ranking algorithm is designed to favor objects whose size is comprised between 20% and 60% of the image size, and the Eiffel tower is a thin object which occupies 10% according to Fig. 18.

8. Conclusion

We have developed an algorithm which deals with a set of images containing both relevant images corresponding to a single concept and irrelevant images. Such set is typically the result of a web image query. Our algorithm automatically segments the images, and decide which one are the most relevant. We are able to increase the relevancy of the first 20 images, while providing a segmentation. On average, 78% of the pixels in that segmentation belong to the object, while 81% of the pixels belonging to the object can be found in this segmentation.

The algorithm we proposed is based on color histograms, but could work with texture histograms or any other histogram. We are planning in further works to use the obtained filtered and segmented images to automatically build databases for learning concepts with images from the Internet.

References

[1] P. Duygulu, K. Barnard, J. de Freitas, D. Forsyth, Object recognition as machine translation: learning a lexicon for a fixed image vocabulary, in: Proceedings of the European Conference on Computer Vision, ECCV, 2002, pp. 97–112. Available from: <http://citeseer.ist.psu.edu/duygulu02object.html>.
[2] J. Li, J.Z. Wang, Automatic linguistic indexing of pictures by a statistical modeling approach, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (9) (2003) 1075–1088.
[3] G. Carneiro, A.B. Chan, P.J. Moreno, N. Vasconcelos, Supervised learning of semantic classes for image annotation and retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (3) (2007) 394–410.
[4] G. Griffin, A. Holub, P. Perona, Caltech-256 object category dataset, Tech. Rep. 7694, California Institute of Technology, 2007. Available from: <http://authors.library.caltech.edu/7694/>.
[5] M. Grubinger, P. Clough, H. Mller, T. Deselaers, The IAPR TC-12 benchmark: a new evaluation resource for visual information systems, in: International Workshop OntoImage 2006 Language Resources for Content-Based Image Retrieval, 2006.
[6] L. von Ahn, L. Dabbish, Labeling images with a computer game, in: CHI'04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM Press, New York, NY, USA, 2004, pp. 319–326.
[7] L. von Ahn, R. Liu, M. Blum, Peekaboom: a game for locating objects in images, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM Press, New York, NY, USA, Montréal, Que., Canada, 2006, pp. 55–64.
[8] A. Popescu, Image retrieval using a multilingual ontology, in: Proceedings of Recherche d'Information Assistée par Ordinateur, RIAO 2007, Eighth International Conference, May 30–June 1, Carnegie Mellon University, Pittsburgh PA, USA, 2007.
[9] C. Fellbaum, WordNet – An Electronic Lexical Database, Bradford Books, Cambridge, MA, USA, 1998.
[10] R. Lienhart, A. Hartmann, Classifying images on the web automatically, Journal of Electronic Imaging 11 (2002) 445–454.
[11] C. Millet, I. Bloch, A. Popescu, Using the knowledge of object colors to segment images and improve web image search, in: Proceedings of Recherche d'Information Assistée par Ordinateur, RIAO 2007, Eighth International Conference, May 30–June 1, Carnegie Mellon University, Pittsburgh PA, USA, 2007.
[12] A. Hanbury, Review of image annotation for the evaluation of computer vision algorithms, Tech. Rep. PRIP-TR-102, PRIP, TU Wien, 2006.