Exploiting User Interaction and Object Candidates for Instance Retrieval and Object Segmentation

Amaia Salvador Aguilera, Universitat Politècnica de Catalunya

Supervised by: Xavier Giró i Nieto, Universitat Politècnica de Catalunya
Kevin McGuinness, Dublin City University

Abstract

This thesis addresses two of the main challenges nowadays for computer vision: object segmentation and visual instance retrieval. The methodologies proposed to solve both problems are based on the use of object candidates and human computation in the computer vision loop. In the object segmentation side, this work explores how human computation can be useful to achieve better segmentation results, by combining users' traces with a segmentation algorithm based on object candidates. On the other hand, the instance retrieval problem is also addressed using object candidates to compute local features, and involving the user in the retrieval loop by applying relevance feedback strategies.

Index Terms

Computer Vision, segmentation, instance search, object candidates, human computing

I. Motivation

Computer vision is nowadays an active research area, which has became increasingly relevant due to the growing amount of digital cameras that generate large amounts of visual data. One of the classic computer vision problems refers to object detection, a task in which machines are expected to locate and recognize the objects present in a scene with an accuracy similar or better than a human would achieve. This task is still a challenge for computer vision systems, and many approaches have been implemented over multiple decades.

The most common approach for object localization is to evaluate a large number of windows at different scales at a pixel level, and determine whether they contain an object or not. One classic example of this approach is the Viola Jones face detector [1], which is still a state-of-the art technique for object detection but presents several major drawbacks:

1) Its output consists of bounding boxes centered in the location of the detected object, not their pixel-wise segmentation.
2) It relies on the pixel-wise analysis of the image. Even though the Viola-Jones detector is fast and accurate, it is somewhat unnatural to analyze the image pixel by pixel to identify objects.
3) It is only applicable to detect trained categories. That is, these approaches cannot provide information about those object classes which were not specifically considered during the training stage.

In the past decade, other image representations have been proposed to replace the structure of the pixel grid, such as region-based representations.

In a region-based image representation, objects in the scene are described by one or multiple regions in an initial partition. These regions are commonly referred as *superpixels*. Superpixel algorithms [2] [3] [4] [5] [6] group pixels into perceptually meaningful atomic regions, which capture image redundancy and greatly reduce the complexity of subsequent image processing tasks. A step forward into more efficient image representations are the *hierarchical region-based representations* [7] [8]. Hierarchies of regions provide partitions of the image at different resolution levels. This way, it is expected that the objects in the image are described by a single region in at least one of the levels of the hierarchy.

Recently, different works have presented approaches that are able to provide a limited number of high-quality and category-independent object candidates using segmentation. These techniques [9] [10] [11] generate a ranked list of object candidates for the image, based on its visual features together with additional parameters learned from a collection of semantically meaningful regions. These object candidates have been used in recent works [12] [13] [14] for object detection and segmentation tasks, achieving state-of-the art performances.

Despite of all the achievements in the research field during the past decades, and due to the fact that the way the human brain operates in order to analyze the content of a scene is still not quite understood, machines still have not matched human performance when it comes to solving these type of problems in the computer vision field. This has been commonly characterized as the *semantic gap*. This gap has not been completely bridged yet, but it is expected that, by bringing human computation into the computer vision loop, the semantic gap will be significantly narrowed.

This thesis focuses on two of the most important problems in computer vision: object segmentation and instance search. **Object segmentation** is the task of locating the objects in an image and giving an accurate segment by labelling each pixel of the image as either foreground or background. The result of this process is a binary mask in which white pixels indicate the foreground and black ones indicate the background. **Instance Retrieval** in computer vision is the problem of searching and retrieving images from large databases. This thesis aims at studying the potential of both object candidates and user interaction to address these two computer vision problems.

This document is organized as follows: First, Section II reviews several techniques for object candidate generation (section II-A) and computer vision works involving the user in the loop (section II-B). Section III is dedicated to the problem of Interactive Object Segmentation. This section introduces two different user interfaces to collect traces from users: *Ask'nSeek* and *Click'n'Cut* (section III-A) and three different interactive segmentation algorithms (section III-B) that can be combined to achieve a good segmentation quality (shown in section III-C). Section IV is dedicated to the Instance Retrieval problem in the context of TRECVID Instance Search. This section presents the adopted pipeline for our submission to TRECVID, giving insights into the potential of using object candidates to compute local features and studying the impact of the users' feedback in the results of a retrieval system.

II. State of the Art

For the purposes of this thesis, the state of the art is dedicated to review several object candidate algorithms and to describe recent works involving humans in the loop to solve computer vision problems.

A. *Object Candidates*

Object proposal methods can be divided in two gropus: those whose output is a set of image windows around the objects in the image, and those that provide accurate segmented object candidates.

In the first category, Cheng *et al.* introduced in [15] a method to quickly generate a small set of object location proposals by using edges. They claim that the number of contours that are wholly contained in a bounding box is representative of its likelihood to contain an object, and use this metric as a score to rank their candidate boxes. Lawrence and Dollár presented in [16] a method that is able to give a high object detection rate within a small number of proposals at a speed of 300fps by using binarized normed gradients to determine the objectness of an image window. While these methods only provide the location of the objects and not their pixel segmentation, they can be extremely useful for object recognition tasks in which speed is more critical than accuracy.

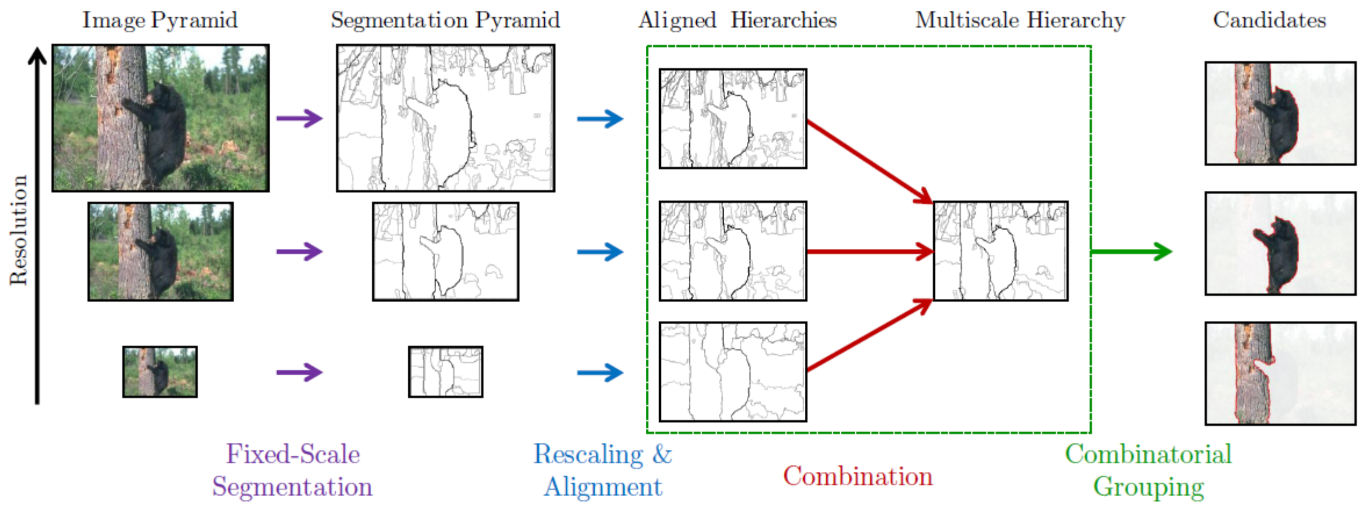


Fig. 1: Multiscale Combinatorial Grouping (MCG) [11].

In the second category, Van de Sade *et al.* [17] proposed a selective window search method based on computing multiple hierarchical segmentations based on superpixels from [3] and placing bounding boxes around the groups of regions at different levels of the hierarchies. In order to diversify the set of object location proposals, they use several grouping strategies to identify clusters of similar object candidates. After the grouping, the object hypotheses are sorted according to their likelihood of containing an object, which is estimated based on the order in which the hypotheses were generated in each individual grouping strategy. Selective Search has been widely used for object detection methods [18] [19] [20] due to its relatively fast speed and high recall. Carreira and Sminchiescu presented in [9] a framework to generate object proposals by seeding a segmentation algorithm. They address this problem by solving a sequence of Constrained Parametric Min-Cuts (CPMC). To do so, regions are grown around regularly placed foreground seeds which are combined with various background seeds for all levels of foreground bias. The resulting set of segments is filtered and ranked according to their plausibility of being good object hypotheses, based on mid-level properties. Arbeláez *et al.* introduced Multiscale Combinatorial Grouping (MCG) in [11], yet another approach for object candidate generation by using a fast normalized cuts algorithm combined with a hierarchical segmenter used at different scales and a grouping strategy that combines multiscale regions to produce highly accurate object candidates. The biggest advantage of these approaches is that they are able to generate high quality segmentation masks, but they generally have a high computational cost (up to several minutes per image). To overcome this limitation, the authors of [11] presented Singlescale Combinatorial Grouping (SCG): a faster method of their algorithm, which only uses a single scale hierarchy. Using this technique, they still show competitive results while reducing the computational time one order of magnitude.

B. Human Computation

The combination of image processing and computer vision algorithms with human interaction has been extensively explored in the literature. Many works related to object segmentation have shown that user inputs throughout a series of weak annotations (scribbles, bounding boxes, clicks, ...) can be used to seed segmentation algorithms, resulting in a great increase of accuracy. Rother *et al.* introduced Grabcut [21], a segmentation algorithm based on iterative graph-cuts. In [21], GrabCut is initialized with a bounding box to produce a rough segmentation, which can later be refined by the user with scribbles. Arbelaez *et al.* [22] showed a segmentation method based on hierarchical segmentation using Ultrametric Contour Maps [8], in which the user is only asked to make two clicks: one in the foreground and one in the background. McGuinness *et al.* [23] presented a scribble driven segmentation tool for users to be able to mark foreground and background areas of the image by simply clicking and dragging using the mouse. The tool provides the user with feedback of the segment generated with his/her traces, allowing him/her to refine it by generating more scribbles where they are needed. Jain and Grauman [24] introduced an interactive segmentation algorithm that is able to predict which type of user input is better to initialize a graph-cuts segmentation for each image, and directly ask for that type of annotation to the user. Giró *et al.* introduced in [25] a graphical environment that uses scribbles and bounding boxes as user inputs to seed a hierarchical segmentation algorithm [26]. Other works [27] [28] have also shown the potential of directly introducing the human in the loop by measuring brain responses to visual stimulus for the tasks of object segmentation and detection.

Another common approach for human computation is *crowdsourcing*, in which the task that a single user would have to do is splitted into several incremental steps that many users can do separately. A popular framework that uses crowdsourcing is LabelMe, introduced by Russell in [29]. LabelMe is a collaborative web-based annotation tool that allows users to label object categories within images, and to perform a rough segmentation of them. LabelMe has been used to annotate datasets for object detection and recognition tasks. Another example is the project involving the Microsoft COCO dataset [30], in which users from Amazon's Mechanical Turk are used to segment the objects in their images.

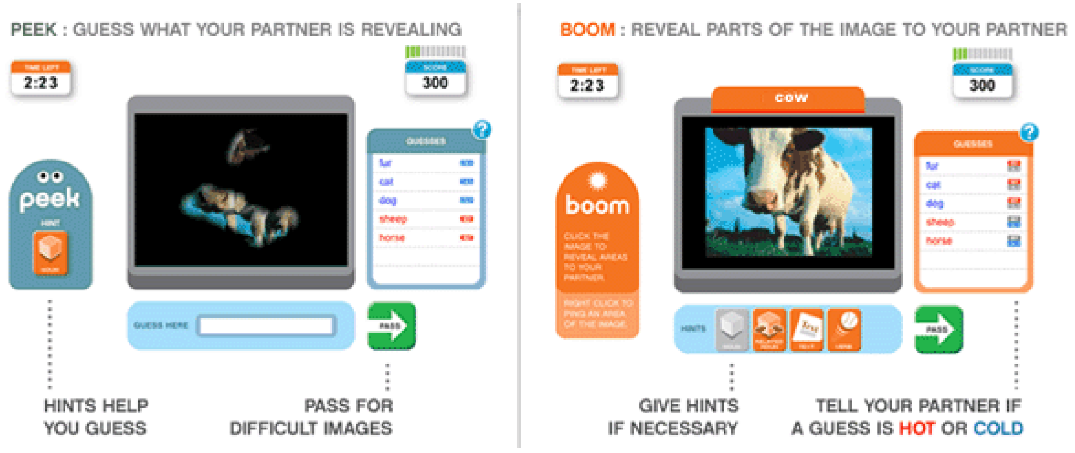

PEEK : GUESS WHAT YOUR PARTNER IS REVEALING
2:23
300
peek
HINTS HELP YOU GUESS
PASS FOR DIFFICULT IMAGES
BOOM : REVEAL PARTS OF THE IMAGE TO YOUR PARTNER
2:23
300
cow
boom
GIVE HINTS IF NECESSARY
TELL YOUR PARTNER IF A GUESS IS HOT OR COLD

Fig. 2: Peekaboom, a GWAP for object recognition [31].

Following the trend of *crowdsourcing*, recent works have also shown the potential of building user interfaces in the form of games. These are called *games with a purpose* (GWAPs), and their goal is to make the annotation tasks less tedious for the users. Luis von Ahn was the first to introduce a GWAP in [32], called the ESP game. This game pairs two random players and shows the same picture to each one of them. The users type keyword descriptions to the image and, when both players agree with a keyword, it becomes part of the tags describing the image. In order to make the game more challenging, keywords that are added as tags appear as taboo words the next time the image is shown in a game play. Using the ESP game, the resulting annotation refers to the whole image, without spatial precision regarding to location of the represented objects. Luis von Ahn went a step further with Peekaboom [31], another GWAP in which users are paired randomly and, while one player progressively reveals parts of the image, the other has to guess the correct keyword for it. Finally, Snoek *et al.* introduced the Name-it-Game [33], an interactive multimedia game in which the two players switch roles after each turn. In every game play, one player is the revealer and the other one is the guesser. The revealer outlines an object within the shown image and picks the object name and definition from a given list of words. Then the object is revealed progressively to the guesser, who may ask for hints during the guessing process. Examples of hints provided to the guesser are the number of letters of the word, antonyms, hypernyms or the definition of the word.

Human computation has also played an important role in the task of object retrieval. Many research groups have presented their user interfaces in that context in workshops such as VideOlympics [34], Video Browser Showdown [35] and TRECVID [36]. Typically, instance retrieval systems provide an initial ranking list of results to be displayed to the user, who is asked to annotate each one of them as relevant or not relevant to the queried topic. These annotations can then be taken into account to generate a better ranking using relevance feedback approaches for query reformulation [37] [38] or re-ranking [39].

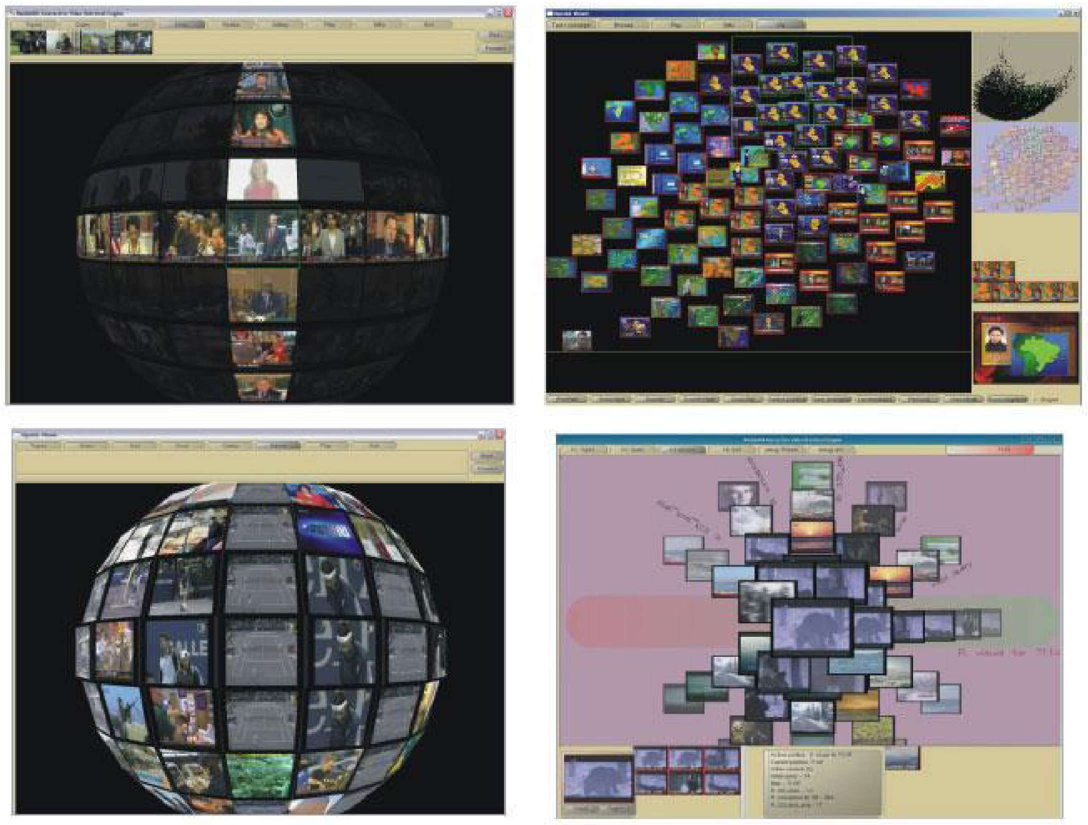


Fig. 3: Mediamill, a user interface for instance search presented in TRECVID 2009 [40].

An example of one of such systems is VisionGo [41], a video retrieval system that allows users to annotate the displayed keyframes as ”positive”, when they are relevant to the query topic, ”negative”, when they are not, and also as ”near positive”, which refers to those keyframes that are not entirely positive, but could provide valuable feedback. These three types of annotations are used to assist a query expansion and relevance feedback process that generates a new ranking list of results to the user. Snoek *et al.* presented Mediamill [40] in TRECVID 2009. This interface provides an ordered list of keyframes in four different dimensions that were obtained based on time, position, concept classifier and perceptual similarity. Users can navigate in any of the four dimensions to find those keyframes that best answer to the queries. Their system uses a relevance feedback strategy which consists on training a support vector machine model based on explicit positive annotations from the user and negative examples obtained by passive monitoring. Another case is the AXES project [42] [43] for multimedia retrieval, which, in addition to the usual annotation of positive and negative keyframes, it allows the user to perform text-based, concept-based or image-based searches to complement the query topics and to refine the results using data from Google Images. The system uses all this data to train a linear SVM using VLAD features [44]. Finally, Ventura *et al.* presented in Video Browser Showdown 2012, a video browser [45] that allows users to navigate through a hierarchical index of video frames organized according to their visual similarities.

As mentioned earlier in this section, another way of introducing human computation are brain-computer interfaces, which can capture users' brain responses to visual stimulus. Recent papers [46] [47] have shown the potential of this approach also for the image retrieval case.

III. Interactive Segmentation

This section addresses the problem of how to enroll the human power of the crowd into solving the task of pixel-wise object segmentation. The goal of an object segmentation process is to label all pixels of an image depending on whether they represent a certain object or not. Typically, the output of such process is a binary mask in which the white pixels represent the object and the black pixels represent the background. This work explores how crowdsourcing can also be a valid strategy to obtain a large amount of high quality object segmentation results (masks), as long as the appropriate tools and data collection strategies are selected. Particularly, this work explores different interactive segmentation algorithms that use foreground and background clicks as seeds.

A. Data Collection

This section introduces the different tools that were used to collect data from users: a collaborative game and an interactive segmentation tool.

1) Ask'nSeek: The first one is the Ask'nSeek game, which was introduced by Carlier *et al.* [48] in 2012. Ask'nSeek is a two-player web-based game that asks users to find the location of a hidden region in an image. In every game play, two users are paired randomly and one is given the master's role and the other one the seeker's. The *master* hides a rectangular region in the given image and the *seeker* tries to find this region by clicking on the image and asking for clues to the master by typing the name of the different objects in the image. The clues that the master can give back to the seeker provide spatial relations selected from the list: *above, below, to the right of, to the left of, on, partially on, none of the above*, and the *seeker* takes them into account to locate the region as soon as possible. In the end, game logs contain clicks that refer to the different objects in the image. Figure 4 is an example of the end of an Ask'nSeek game from the *seeker's* point of view.

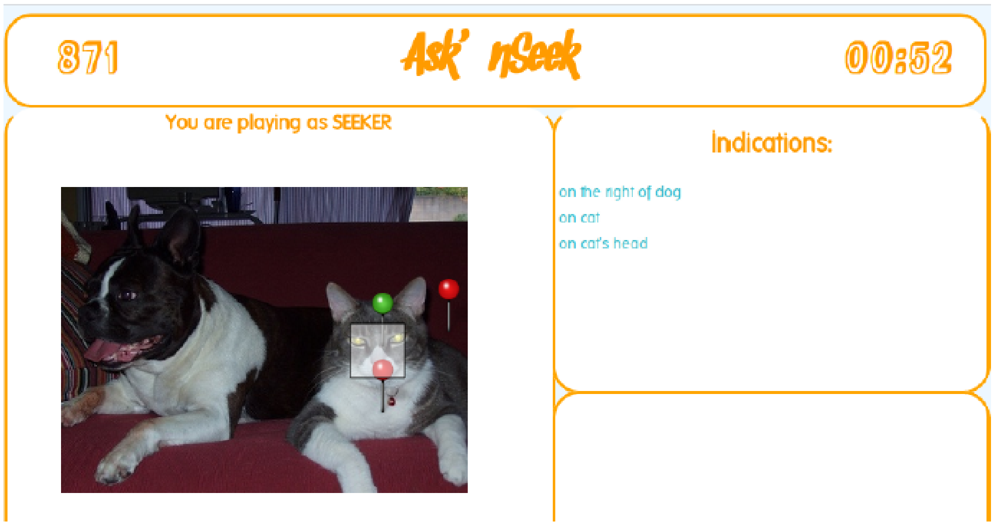


Fig. 4: Screenshot of the Ask'nSeek game, from the *seeker's* point of view.

After several games of Ask'nSeek, it is assumed that the most prominent objects will have a sufficient amount of user interactions in order to obtain foreground and background seeds. Foreground seeds would correspond to all those game traces that refer to the object with the tag *on*, whereas background seeds would be composed by those that refer to it with the tags *above, below, to the right of, to the left of*. The potential of this game's logs for object segmentation has been shown in [49], in which game traces are combined with the object proposals from [9] to produce accurate segments.

Despite of all the opportunities offered by the Ask'nSeek game, this gamification approach poses some problems associated to the collection of textual labels which require a post-processing of the user traces. These limitations are the following:

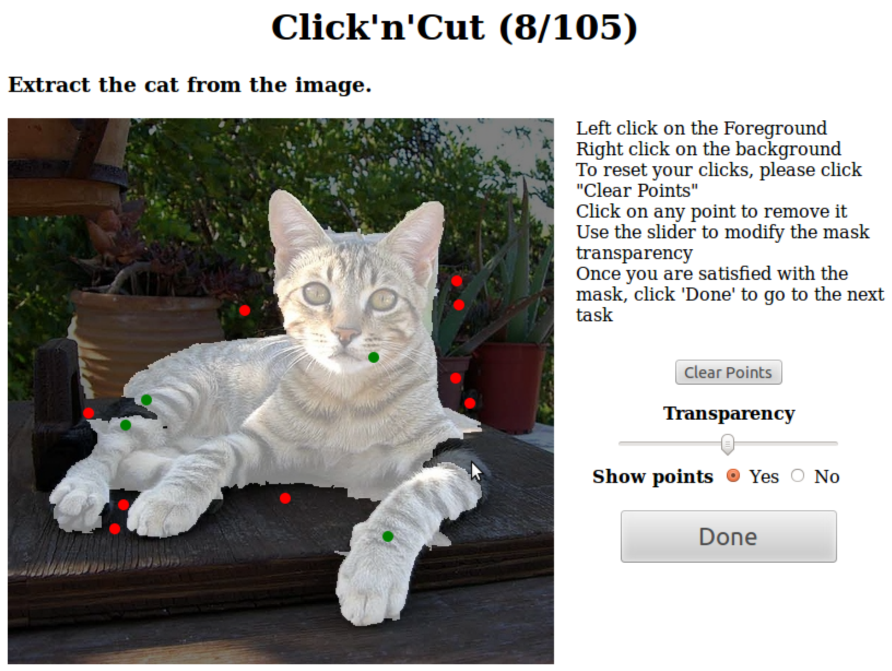
- **Grouping and filtering of object labels.** The labels generated by the users who played the *seeker's* role in Ask'nSeek can have spelling mistakes, can be in many different languages and can contain synonyms or equivalent terms that need to be identified and either be removed or grouped together when necessary.
- **Mapping labels with ground truth descriptions.** Similarly to the previous point, to be able to evaluate the segmentation results obtained with *Ask'nSeek*, it is necessary to map the object labels of Ask'nSeek with the ground truth descriptions of the objects in the images. At this stage, and because we are dealing with a small amount of images, this mapping is done manually.

- **Multiple instances.** Most errors in the Ask'nSeek traces are due to misunderstandings between the *master* and the *seeker*. The most common scenario for these type of mistakes are the images in which there appear multiple instances of the same object. In these cases, what happens usually is that the *seeker* asks for a clue for one of the instances of this object, but does not specify which one. Then, it is possible that the *master* gives an indication, but referring to a different instance of the object specified by the *seeker*. Finally, the *seeker* would follow the *master's* indications according to the object instance that he/she asked for in the first place. If this situation is repeated in different gameplays for the same object, the collected traces for a single label would be ambiguous.

For this project, these limitations were solved by a manual processing of the collected labels. This way, we could focus on the challenges associated to the compuer vision part of the system.

2) Click'n'Cut: To be able to reduce the complexity at the data collection stage and focus only on the object segmentation, we came up with *Click'n'Cut*, a user interface designed for interactive object segmentation. Figure 5 shows a screenshot of the *Click'n'Cut* interface. The different parts of the system are the following:

- The name of the tool and the number of completed images out of the total (top-center part of the screen).
- A description of the object that needs to be segmented (top-left part of the screen).
- A panel in which the current image is displayed. This panel also shows the segmentation result overlayed with the image.
- A reminder of how the interface works (on the top-right part of the screen).
- The basic interactions available for the user (bottom-right of the screen).



Click'n'Cut (8/105)

Extract the cat from the image.

Left click on the Foreground
Right click on the background
To reset your clicks, please click "Clear Points"
Click on any point to remove it
Use the slider to modify the mask transparency
Once you are satisfied with the mask, click 'Done' to go to the next task

Show points [✓ Yes] No

Fig. 5: Screenshot of the Click'n'Cut interface.

In *Click'n'Cut*, users are asked to produce foreground and background clicks to perform a segmentation of the object that is indicated in the provided description. The fundamental interactions available to the user are the left and right clicks, which generate foreground and background points, respectively. Every time that a user generates a click, the segmentation result is updated and displayed over the image with an alpha value of 0.5 (which can be changed by the user using the *Transparency* toggle). Users can also correct a wrong click by just clicking on it again to make it disappear. The *Clear points* button removes the entire set of clicks that have been made by the user. Finally, once satisfied with the result, the user can go on to the next task by clicking the *Done* button.

B. Segmentation algorithms

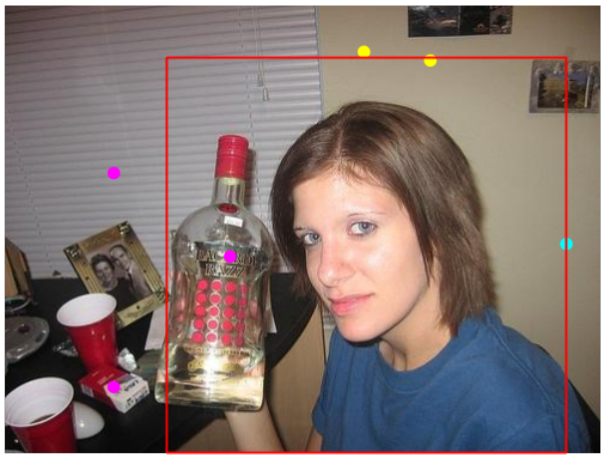
This section explains the different segmentation algorithms that have been tested. Different techniques have been explored in order to use these user traces and obtain the best segmentation possible. Two main directions have been addressed: a first one based on graph cuts and a second one based on object candidates.

1) GrabCut: The GrabCut segmentation algorithm [50] aims at achieving a high performance in the segmentation task at the cost of little interaction on part of the user. This segmentation is achieved by solving an iterative graph cuts [51] using foreground and background seeds as initialization. For these experiments, the OpenCV's [52] GrabCut implementation was used. This implementation allows for a level of uncertainty in the labels of the seeds. At the initialization stage, each pixel of the image needs to be assigned a label within the list of: *fixed foreground (F), fixed background (B), probable foreground (PF), probable background (PB)*. GrabCut uses all the pixels in both *probable* and *fixed* seeds to initialize Gaussian Mixture Models for the foreground and the background. The only difference between the *probable* and *fixed* labels is that pixels initialized as *probable* can change labels during the segmentation process, whereas pixels with *fixed* labels will certainly have the same one in the segmentation result.

The first intuitive solution to combine user traces with this algorithm would be to simply initialize GrabCut using only foreground and background clicks obtained from users as seeds. However, as mentioned earlier in this section, GrabCut uses the color information of the pixels in the foreground and background seeds to initialize a Gaussian Mixture Model for each of the two labels, which require much more samples than just a few pixels. To overcome this issue, the superpixels in [2] are computed for the image, so that the ones that contain the foreground and background clicks are also used to initialize their corresponding Gaussian Mixture Model. Figure 7 gives an example of the described process of using GrabCut for interactive object segmentation.

Bounding box generation from traces. Aside from using foreground and background clicks to initialize GrabCut, it is also possible to do so with a bounding box. This is achieved by assigning all the pixels outside the box as *fixed background* and the ones inside of it as *probable foreground*. User traces collected with the Ask'nSeek game (see section III-A1 for a description of the game) can be used to produce a bounding box. This can easily be done by taking the mean positions of each set of clicks with associated indications in: *to the left of, to the right of, above, below* as the extreme values of the bounding box. Figure 6a illustrates a practical example of this process and its outcome.

Cleanup of traces. As introduced in section III-A1, the Ask'nSeek traces tend to be noisy due to user errors. In particular, users tend to make more mistakes when it comes to follow spatial relations related to the background: *to the left of, to the right of, above and below*. Having that in mind, it is easy to come up with a simple algorithm that relies on the foreground clicks to filter out bad *left, right, above and below* clicks. Namely, a click is discarded if its label is not consistent with all the foreground clicks (see figure 6b for an example).


(a) Bounding box from traces

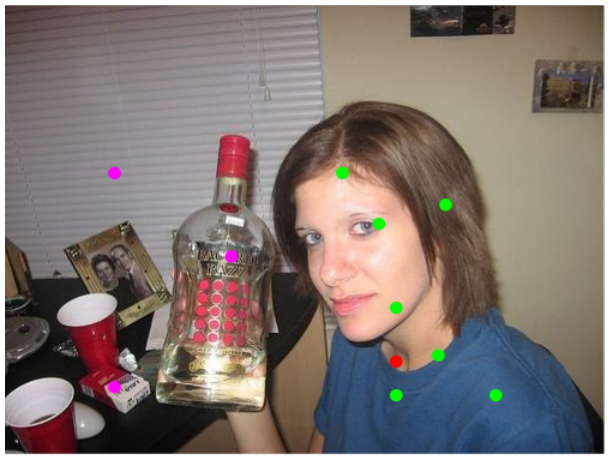

(b) Cleanup of traces

Fig. 6: Examples of post-processing algorithms. **(a) Bounding box from traces** gives an example of the bounding box algorithm. In this figure, all the clicks refer to the girl in the image. Pink, yellow and blue clicks refer to the spatial indications *to the left of, above and to the right of*, respectively. For each spatial indication, the mean of the positions of all the clicks is taken as the extreme value of the bounding box. If there are no clicks for one of the indications, the bounding box is limited by the size of the image in that direction. **(b) Cleanup of traces** is an example of the algorithm to clean up the traces. The green clicks indicate foreground (in this case, the girl on the image) and the pink and red clicks represent clicks *at the left of* the girl. The pink clicks are those that are consistent with all the green clicks, which are considered correct by the algorithm. However, the red click is not consistent with all the foreground clicks (it is not *at the left of* all of them, and thus it is discarded by the algorithm.).

2) Object Candidates: The segmentation algorithms explained in this section are based on the mapping of the collected traces to a set of precomputed object proposals, introduced in Section II-A.

Algorithm 1: Finding the best mask. The basic idea of the first algorithm is to find the mask among all the object candidates that best fits the set of *foreground* points and *background* points, and give this mask as the segmentation result. To

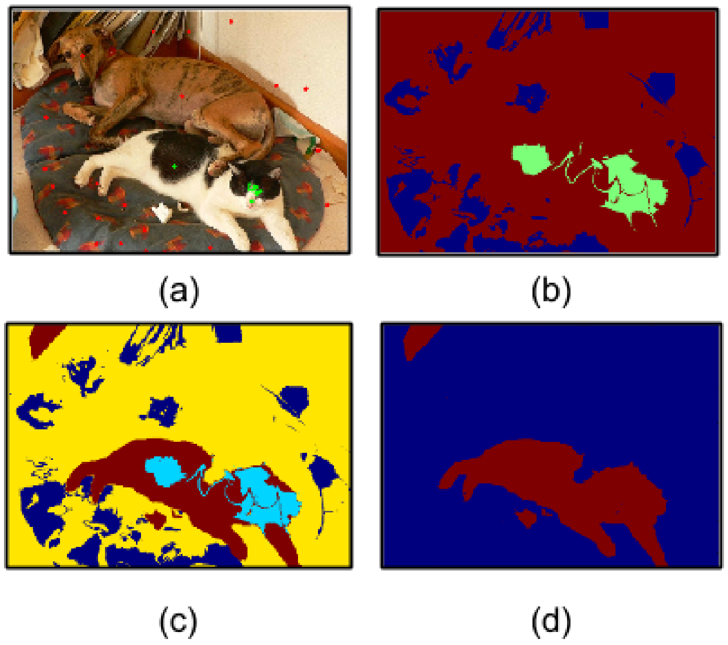


Fig. 7: Steps of the GrabCut segmentation algorithm. In (a), the image with the foreground and background clicks collected with the Ask'nSeek game. (b) is the mask used to initialize GrabCut with superpixels [2]. The green superpixels are *fixed foreground*, the blue ones are *fixed background* and the rest of the image is set as *probable background*. (c) is the result of GrabCut after 5 iterations. Light blue pixels indicate the *fixed foreground* seeds and red pixels indicate the pixels that were predicted as *probable foreground* by the algorithm (and the same for dark blue and yellow pixels). Finally, (d) is the segmentation result obtained after grouping *probable* and *fixed* pixels for both *foreground* and *background*, which are represented in (d) with red and blue, respectively.

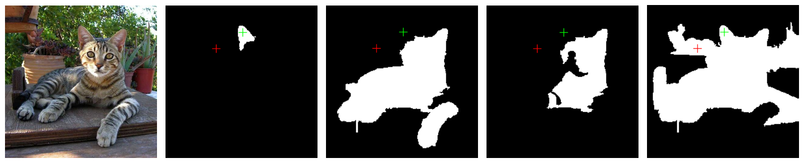


Fig. 8: Examples of MCG candidates.

do so, the following algorithm is adopted: For each mask $m \in O$, where O is the set of object candidates, penalization scores fg_p and bg_p are computed for *foreground* and *background* points, respectively. fg_p (resp. bg_p) corresponds to the number of foreground (resp. background) clicks that are incorrect with respect to m. For example on figure 8 the foreground point (in green) is incorrect with respect to the masks 2 and 3. The sum of fg_p and bg_p gives the overall penalization score p_m for mask m. Then, the mask m with the minimum p_m associated is selected. If several masks have the minimum penalization score, the one that appears first in the precomputed ranked list of object proposals is selected.

Algorithm 2: Union of masks. This algorithm is an extension of the previous one, and aims at combining different object candidates when none of them fits all the *foreground* and *background* points. To compute the best mask with respect to a set of f *foreground* points and b *background* points, the following algorithm is adopted: For each mask $m \in O$, where O is the set of precomputed object candidates, we start by computing two scores fg_m and bg_m. fg_m (resp. bg_m) is the number of foreground (resp. background) points that are correct with respect to m. Then, if there exist a mask m^* for which $fg_{m^*} = f$ and $bg_{m^*} = b$ then m^* is the best possible mask and this is the selected mask. Else, it means that no mask is correct with respect to all the clicks. In that case, we build a set of masks $M^* = \{m \in O, bg_m = b \text{ and } fg_m > 0\}$. This means that M^* contains the masks that have not been defined as background and for which there is at least one foreground point. The union of all masks that belong to M^* form the best mask that is returned as the segmentation result.

C. Experiments and Results

This section explains the different experiments that have been carried out for the object segmentation task. The first experiment compares the segmentation results using different initialization criterias of GrabCut and the segmentation algorithm for object candidates. The second experiment uses the enhanced segmentation method for object candidates using Click'n'Cut to perform interactive segmentation using different types of users. See section III-B for a detailed explanation of the segmentation algorithms.

Comparison of GrabCut with Object Candidates. The dataset used for evaluation of the first pool of experiments consists of a subset of images from the PASCAL VOC 2012 dataset for which a sufficient amount of traces (more than 15 clicks between foreground and background) were collected using Ask'nSeek game (See Section III-A). With this criteria, 10 objects from 8 different images were selected to evaluate the results. This data was used along with the different segmentation algorithms, in order to obtain a fair comparison between them. The metric used to assess the segmentation results is the Jaccard Index $J = \frac{P \cap GT}{P \cup GT}$ between the Predicted (P) and Ground Truth (GT) masks.

Method	Algorithm description	Jaccard Index using raw clicks	Jaccard index using cleaned clicks
Object Candidates (CPMC [9])	Algorithm 1 in III-B2	0.5430	0.5430
Object Candidates (MCG [11])	Algorithm 1 in III-B2	0.6290	0.6290
GrabCut	Initialized with foreground/background superpixels with *fixed* labels.	0.4969	0.5152
GrabCut	Initialized with foreground/background superpixels with *fixed* labels including the bounding box information: Pixels outside the bounding box are *fixed background* and pixels inside the bounding box are assigned *probable foreground*.	0.4001	0.5777

TABLE I: Jaccard index for different segmentation algorithms involving GrabCut and Object Candidates

Results in Table I show that it is useful to initialize GrabCut with a large amount of pixels marked as *fixed background*, and using the bounding box along with the algorithm to clean the traces helps on that. This approach is specially helpful for small (or less salient) objects, because GrabCut sometimes "grows" parts of the image (e.g. other objects) that are not marked as foreground, just because they are somewhat similar to the foreground data (even though they are not connected to the foreground superpixels). Marking these pixels as *fixed background* from the beginning prevents this from happening. This approach does not work well when using raw traces, because the bounding box obtained with the previously introduced algorithm is not correct due to the errors in the clicks.

It is also interesting to see how the cleanup of the traces does not affect the segmentation results of the approaches using Object Candidates. This is due to the design of the segmentation algorithm, which does not aim at finding the segmentation mask that perfectly fits all the points, but the one with fewer errors. This means that minor mistakes in the clicks do not necessarily change the mask that is selected by the algorithm.

Click'n'Cut and Object Candidates. For this experiment, Algorithm 2 from section III-B is used along with the clicks obtained with the Click'n'Cut tool. The dataset used for evaluation has been proposed by [53] and it consists in 100 objects to segment from 96 different images from the Berkeley Segmentation dataset (BSDS) [54]. In this dataset, each object is described by a sentence and a binary mask is provided as ground truth. Additionally to these images, 5 more images from the PASCAL VOC dataset were added in order to be able to discard users with poor performances. This technique is commonly referred to as *gold standard*.

1) Protocol: The experiments in this section are split in three campaigns involving different tasks and different types of users. The first two campaigns were submitted to www.microworkers.com in order to collect data from the crowd, whereas the third campaign was organized in a smaller setup in which only researchers and colleagues from different institutions participated.

- **Campaign #1: drawing a bounding box**
  The algorithm for object proposal generation that was used for these experiments [11] gives around 5000 object candidates for every image, which were difficult to handle in the server side of the Click'n'Cut interface. To overcome this issue, the purpose of this campaign was to reduce the set of object candidates by asking users to draw a bounding box around an object. Then, these bounding boxes are used to filter the object candidates by only keeping the ones that are surrounded by them. The interface for this experiment was similar to *Click'n'Cut*, except that workers did not click on the image but just draw a rectangle.
  25 jobs were created in Microworkers that consisted in completing the described task for 105 images. The offered salary for successfully completing this task was of 2.5 USD (2.38 cents for each bounding box). A total of 52 users participated to this campaign but only 25 of them completed it until the end.
- **Campaign #2: segmenting objects with a crowd**

Using the bounding boxes from the 25 users who completed the first campaign, a bounding box for each object was generated by computing the median of all bounding boxes from all the users. These bounding boxes were used to reduce the number of masks (i.e. objects candidates from the MCG algorithm). This procedure significantly reduced the amount of data to be uploaded to the server, and it also caused a quicker convergence towards a good segmentation.
A second campaign was started on Microworkers, for which 20 jobs were created that consisted in segmenting 105 objects. The offered salary for completing this task was of 4 USD for the job (every segmented object was worth 3.8 cents.).
A total of 99 users participated to the campaign but only 20 of them completed the 105 tasks.

- **Campaign #3: segmenting objects with experts** The third experiment consisted in conducting the exact same study than on the second campaign except that this time the users who were asked to participate were experts from different research labs. A total of 15 experts (11 Males, 4 Females) participated to this study, with ages ranging from 19 to 55. These users did not get paid, they volunteered to participate in the experiment.

2) Quality of the traces: The first study focused on the quality of the collected traces. The first observation to make is that crowdsourced traces are noisy due to several reasons, being the main one the misunderstanding of the provided instructions. Figure 9 shows three examples in which workers did not follow the provided instructions and selected a larger region than the requested one, most likely because they did not carefully read the description.

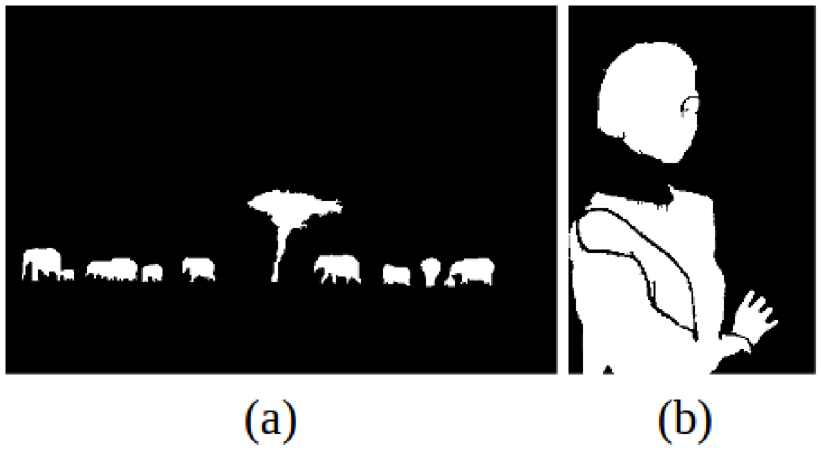


Fig. 9: Errors due to misunderstanding: (a) only the central object (a tree) should be segmented, (b) only the head of the person should be segmented.

The quality of the users was estimated through 5 gold images from the Pascal VOC dataset [55]. The comparison between the error percentage for the gold or test datasets indicates that quality estimation is actually a challenging task. Figure 10 shows the comparison of error percentage for the 20 users in both the 5 gold images and the 100 test images. Users with a error percentage higher than 20% in the gold images were discarded. It can be observed in the figure that while users 3, 5, 8, 10, 11, 18 and 19 were correctly discarded, users 12 and 15 were over-penalized by their performance on the gold images (their error rates for the test images were much lower).

Another observation from Figure 10 is that users 3 and 18 most likely confused the foreground and background alpha masks and basically segmented the complementary shape. Their error percentages are higher than 90%, which would turn into a 10% by switching the labels of their foreground and background clicks. Despite these users were just discarded for our further experiments, their behavior could have easily been detected and corrected automatically.

3) Accuracy vs Time trade-off: In this section, Click'n'Cut is compared with the top two best configurations proposed in [53]: GrabCut [21] and hierarchical partition with BPTs [26] [56]. The different solutions are assessed in terms of the accuracy vs user time trade-off, where segmentation accuracy is measured with the Jaccard Index. The graph in Figure 11 plots the average Jaccard obtained with the amount of time users spent creating their annotations. These experiments indicate that *Click'n'Cut* with expert users converges more rapidly than the two graph-based approaches, but also that accuracy saturates sooner in our proposal. However, a crowd of non-expert users perform poorly when using *Click'n'Cut*, because of the high number of errors they make. Filtering out the worst out of the crowd according to their error rate in the gold images, the resulting curve improves significantly. One of the reasons why the approaches proposed in [53] are able to reach a higher Jaccard index with their experiments is because the spatial resolution of the regions or pixels used in GrabCut and BPTs is higher than the one used when combining object candidates in Click'n'Cut. On the other hand, the Click'n'Cut approach with object candidates converges much faster that the ones in [53], being able to produce a first accurate segmentation results within the first 10 seconds.

4) Budget: Budget in this crowdsourced task can be approximated from two perspectives: based on *user time* or on *money*. Table II compares the necessary *user time* on *Click'n'Cut* with the data collected from other related publications, previously

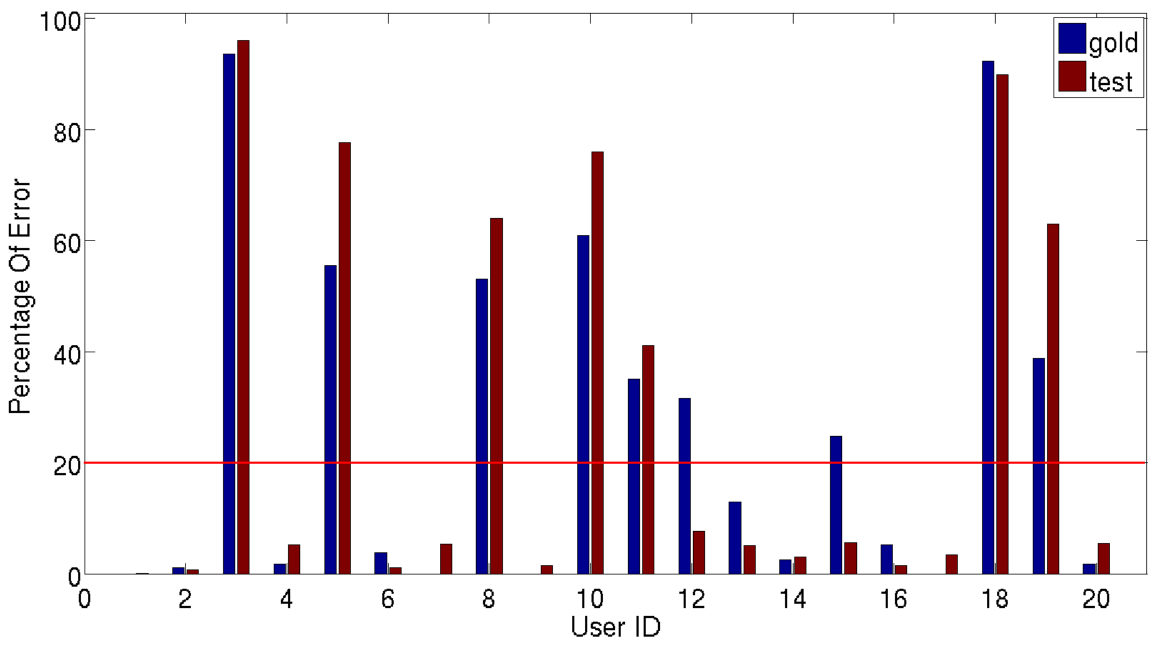


Fig. 10: Error rate for each worker for the test dataset (BSDS) and the gold dataset (5 Pascal VOC images).

	Dataset	User / image	Users type	Input	Avg. Time (sec.)	Best Jaccard
Lin [30]	Microsoft COCO	1	Crowd	Tight Polygon	79	(Used as GT)
Jain [24]	IIS+MSRC+CoSeg	5	Crowd	Box	7	-
		5	Crowd	Sloppy contour	20	-
		5	Crowd	Polygon	54	0.51-0.76
McGuinness [53]	BSDS (DCU subset)	20	Experts	Scribble	60-85	0.93
Click'n'Cut	BSDS (DCU subset)	15	Experts	Click	32	0.87
		20	Crowd	Click	23	0.75 $\rightarrow$ 0.83

TABLE II: Comparison of *Click'n'Cut* with similar approaches, including average user time and best Jaccard.

introduced in Section II-B. The first observation is the diversity of datasets used to solve interactive segmentation tasks prevents a direct comparison of the resulting values, with the exception of *Click'n'Cut* and [53], as detailed in Section III-C3.

The experiments also show that, given the same interface, experts tend to spend almost 50% more time than the crowd in generating the annotation, but that this higher dedication only produces small increase in the quality of the segmentation. According to these data, the faster responsiveness of Click'n'Cut (already pointed in Section III-C3) seems to be confirmed with the comparison with other solutions. This may be explained because the rest of crowdsourced systems are not exploiting any image processing algorithm to assist the user in his task, which forces the user to manually draw the whole contour around the objects.

The comparison of cost in terms of *money* is more complicated, especially because the cost of the experts is assumed as zero in most works, since expert users tend to be volunteers. However, this cost should not be neglected, as experts normally participate in this evaluation campaigns as part of their professional activity. For this reason, it is possible to estimate the cost of the expert time to be able to assign a cost to this recruiting strategy. To do so, for this experiment experts are assumed to be either undergraduate students, graduate students, research assistants or professors. The standard salary for a Phd grant in Ireland has been adopted as a reference, for a fairer comparison with the reference work [53], whose experts where recruited in a research lab at Dublin City University. Taken as a reference a wage of 1,808 USD a month, the annotation experiment involving 100 images and 20 experts, would require a budget estimated in 377 USD. Nevertheless, the crowdsourcing Click'n'Cut campaign had a total cost of 130 USD, nearly three times cheaper than an experiment of the same nature using expert users.

5) Improving the results: The curve shown on figure 11 establishes that the method described in section III-C3 for computing the best mask is not robust enough to noisy clicks to produce optimal quality results. In addition, the results on figure 11 are averaged on all users, to be comparable with the experts interactive segmentation. However, the real power of the crowd resides in using the contributions of the workers altogether.

The adopted algorithm to test this idea was the following: Since a lot of clicks were collected from the crowd users, it is possible to generate a probability mask by painting a superpixel segmentation with these clicks. To do so, each superpixel is labeled with a number between 0 (*background*) and 1 (*foreground*). These labels are given according to the confidence score

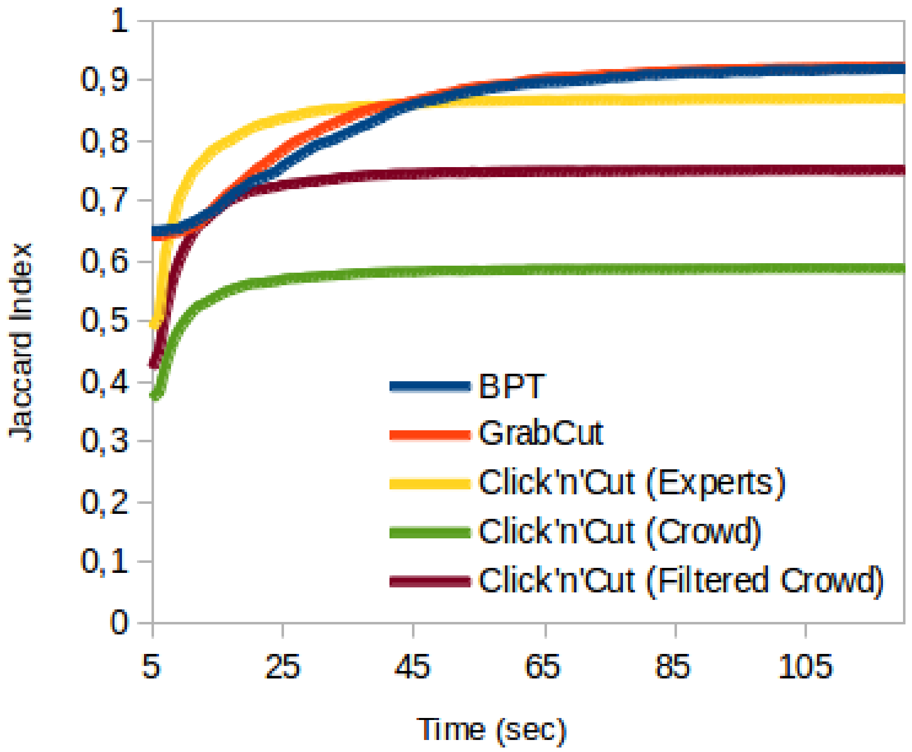


Fig. 11: Average Jaccard vs User time.

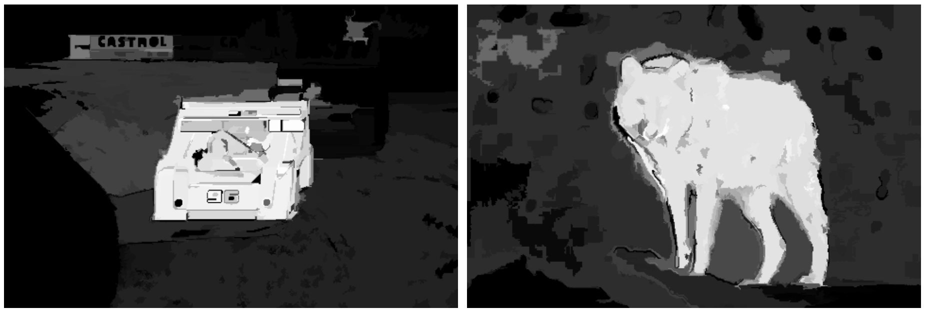


Fig. 12: Probability map of object's presence, based on user's clicks.

that is given to each user, which is stablished based on the user's performance on the gold standard images. For instance, if a user u has a 5% error rate on the gold standard images (see figure 10), the measure of confidence c_u for this user will be 0.95. A *foreground* (resp. *background*) click brings a contribution to the superpixel of c_u (resp. $1 - c_u$). To limit the influence of the superpixels segmentation, we perform the computation on several different superpixel segmentations and average the respective results. Namely, Felzenszwalb's algorithm [3] was used with different parameters. Figure 12 illustrates the performance of this simple algorithm. The object to be segmented is the brightest region, and the bright areas that can be observed in the background regions indicate noisy clicks produced by user's mistakes. Thresholding these maps with a value of 0.7 and performing a simple hole filling algorithm allows to improve the results for the crowd users, increasing the Jaccard Index from 0.75 to 0.83 (see Table II).

D. *Conclusions and Future Work*

The presented work has explored different interactive segmentation algorithms using GrabCut and object candidates using traces collected with a game. Additionally, the opportunities and risks of launching a crowdsourced campaign to collect object segmentations have been assessed.

The first lesson to be learned is that the quality of the users traces is far worse than the common quality obtained from in-lab experiments. Any campaign must include gold standard images to assess the quality of each user. Depending on the severity of the errors and redundancy of collected traces on the same image, different image processing techniques may help into cleaning the segmentations or just discarding those traces that will not meet a minimum quality.

Despite the noisy traces, the presented online interface (Click'n'Cut) has been proved effective in providing users a quick and informative feedback. Showing the current segmentation result overlaid on the image helped guiding users into generating clicks in the most informative locations, a scheme that can be understood as an active learning. *Click'n'Cut* has been proven an effective tool to collect object segmentations for crowdsourcing tasks, especially in terms of fast convergence to high quality results. However, the tool can still be improved in terms of the maximum achievable accuracy, maybe relaxing the strict dependency from the object candidates.

Our campaign has taken special care into comparing the performance of experts and crowd users. The experiments indicate that the cost of experts can triplicate a crowdsourced solution and increase the average duration around 50%, which just provided little gain in accuracy.

IV. Instance Search

This section of the thesis addresses the problem of Instance Search (or instance retrieval), which consists of searching for occurrences of a certain visual instance on a large collection of visual content, and generate a ranked list of results sorted according to their relevance to a user query. More specifically, this thesis adresses the retrieval problem in the case where the queries correspond to visual objects, which are represented using low level visual descriptors extracted directly from the image pixels to be compared with the contents in a target database. This section addresses this problem using pre-trained convolutional neural networks as features, combined with the object candidates from [11] and involving the user in the image retrieval loop throughout a novel user interface.

The system was designed using the dataset, ground truth and evaluation software from the well-known TRECVID Instance Search 2013 [36], a scientific benchmark with dozens of participants from around the world. The goal of TRECVID Instance Search task is to find particular semantic instance (object, person or location) in a large video collection, and return a rank list of video shots in which the instance appears. In addition, the system was also used to participate in TRECVID Instance Search 2014, for which we submitted the results of the four configurations that we considered most scientifically relevant in terms of human computation. Figure 13 shows the typical framework for TRECVID Instance Search task, which will be described in detail in the following sections.

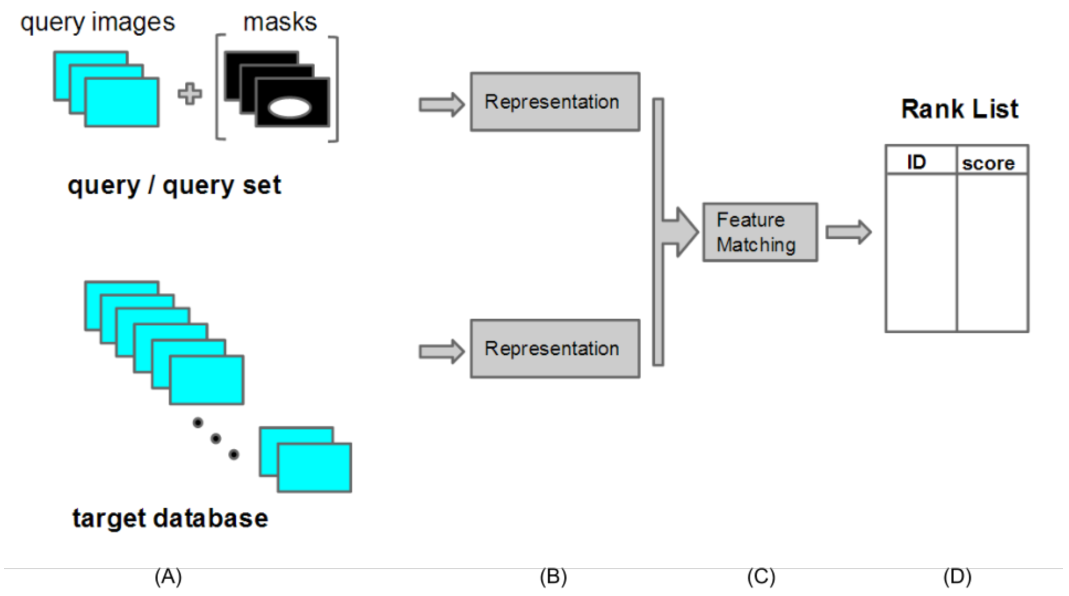


Fig. 13: Instance Search framework for TRECVID

A. Dataset

Participants to TRECVID Instance Search are provided with 244 video files (totally 300 GB, 464 h) with associated metadata, each containing a week's worth of BBC EastEnders programs in MPEG-4/H.264 format. Additionally, each video is divided in different shots of short duration (between 5 seconds and 2 minutes).

Query Set. The query set consists of 30 different topics, each one including: (1) a *category* among the list of: *object, person, location*. (2) A *description* of the topic. (3) 4 *image examples* of the topic. These images are taken out of the first video out of the 244 videos provided by TRECVID. (4) For each one of the 4 image examples, a *binary mask* of the object is also provided. Figure 14 is an example of a query from TRECVID Instance Search 2013.

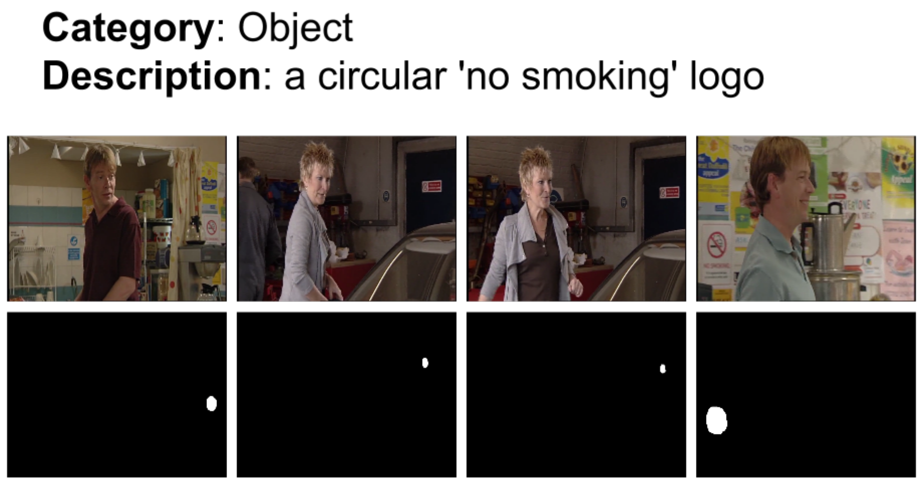


Fig. 14: Information provided for a query topic in TRECVID Instance Search.

Target Database. To handle this large amount of video information, a target image dataset was built by uniformly extracting keyframes for every shot with a sample rate of 1/4 fps. The resulting dataset contained 647,628 keyframes and had a size of 66GB. In further sections of this thesis, this dataset will be refered to as *Full Dataset*. Additionally to this one, a subset was also used during the development stage of the retrieval system. This subset consists of only the *relevant* shots for each query topic of TRECVID Instance Search 2013. This subset consisted of 23,614 keyframes, which was useful to quickly test the different approaches during the development of the pipeline. This dataset will be refered to as *Ground Truth Subset* in the following sections.

B. Feature Extraction

State-of-the art image retrieval pipelines usually use image representations such as VLAD [44], Bag of Words or Fisher Vectors [57]. More recently, many works [58] [59] have shown the potential of deep learning features, using convolutional neural networks (CNN) as image descriptors for image classification and retrieval tasks. The adopted system for this project uses *Caffe* [60], a publicly available code to extract the CNN features of the images. *Caffe* also provides pre-trained ImageNet [61] models that can be used off-the-shelf to directly obtain feature vectors. Convolutional neural networks are composed of different layers, each one encoding different parts and features of the image. While lower levels focus on details and local parts, upper levels contain a more global description of the image. For this reason, it has been suggested in several works [62] [63] that the features from the upper layers of the CNN are the best ones to be used as descriptors for image retrieval. Following those insights, our system uses Layer 7 of the network as **global feature vectors** (see Figure 15).

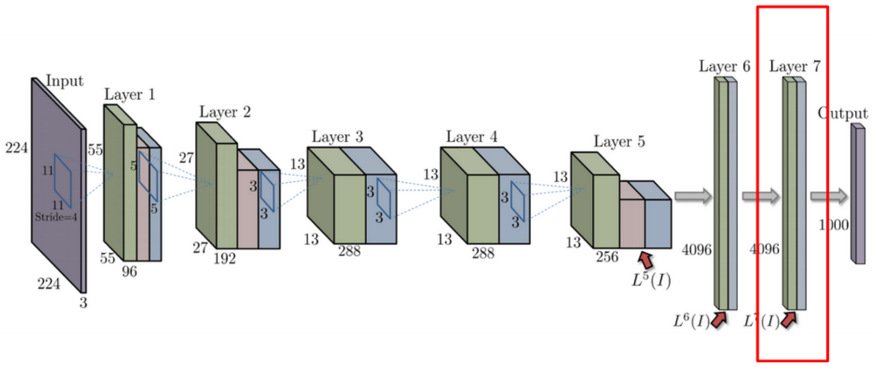


Fig. 15: Example of a convolutional neural network architecture (from [63]).

Global CNN features are useful to describe the general appearance of the image, but the goal of TRECVID Instance Search is not just to retrieve similar images, but images that contain a specific object. Because TRECVID provides the binary masks of the objects in the topic images, our first intuition was to compute the CNN features using as input a crop of the image, keeping only the pixel values inside of the binary mask. This way, we would obtain a representation of the object itself, not the whole image where it is included. However, these new local features on the query images should be matched with local features on the target database, for which no binary masks are provided. To make this possible, object candidates were computed using the SCG algorithm from [11] (see section II-A) for all the keyframes in the target database. This way, **local features** can be computed in both the query and target sets. Section IV-F1 explores the different configurations that were tested for this local approach.

C. *Feature matching*

The feature matching of the query and keyframe feature vectors for the experiments with the *Ground Truth Subset* was performed exhaustively, by computing their cosine similarities to the query vector and sorting them in descending order (more details on this in section IV-D).

However, this approach was not feasible to perform the experiments with the *Full Dataset*, for which the number of keyframes in the target database is much higher. To quickly retrieve the most similar keyframes for a query topic in that case, feature vectors are embedded in a Hamming space building up a forest of 50 binary trees by using random projections. Similar features are then retrieved using Approximate Nearest Neighbor indexes. To do this, we use *Annoy*: a Spotify's implementation for Approximate Nearest Neighbors.

D. *Ranking the results*

The keyframes (either all of them for the *Ground Truth Subset* or the ones retrieved using ANN for the *Full Dataset*) are scored according to the cosine similarity between their feature vectors and the query feature vector, and sorted in descending order to produce a ranked list. However, this procedure is not enough to generate the final ranking. As mentioned in section IV-A, TRECVID provides 4 images for each query topic. Using global CNN features, each query topic would consist of 4 feature vectors that are used separately to retrieve similar images out of the target database, consequently producing 4 independent rankings, which need to be merged in a single one. This is done by fusing them and sorting them again by their score. Then, if a keyframe appears several times, the score associated to it would be the maximum among all (max-pooling). The final step to produce the final ranking is to group the keyframes that belong to the same shot, in order for them to represent a single entry of the ranking. Again, the keyframe with maximum score is kept as the one representative of the shot (max-pooling).

E. *User Interface*

This section introduces the interface used to collect annotations for our submission to TRECVID Instance Search 2014. Figure 16 shows a screenshot of the interface. Its different parts and possible user interactions are:

- A dropdown list containing the IDs of the query topics (top-left). Users can select the one that they want to use to retrieve similar results.
- A textual description of the query topic along with its category (object, location or person).
- An expandable tag *Examples*, displaying 4 images containing the query topic (left-column).
- An expandable tag *Saved*, displaying the positive images annotated by the user (left-column).
- A textbox to introduce the user name (top-center bar).
- The results panel. It displays the retrieved keyframes by the system to the user. The users' task is to annotate them as either positive or negative.
- The *Search* button. Users can press it at the beginning and after they have annotated all the keyframes they see in the results panel to obtain different ones.
- A counter displaying the amount of time remaining for a query topic. It is possible to pause and unpause it by the user.

This interface was built using HTML5 and AngularJS on the client side and Python on the server side, using MongoDB as connected database.

F. *Experiments and Results*

This section describes the experiments that were carried out in order to choose the best configuration four our submission to TRECVID. First, a study on the CNN features is presented, exploring different approaches using both global and local features. Then, different relevance feedback techniques are evaluated. The metric used to assess the results is the Average Precision (equation 1), which is computed as the average precision at the positions of the relevant documents in a ranked list. To evaluate the performance of the system for the whole set of query topics provided by TRECVID, the Mean Average Precision (equation 2) will be the metric used.

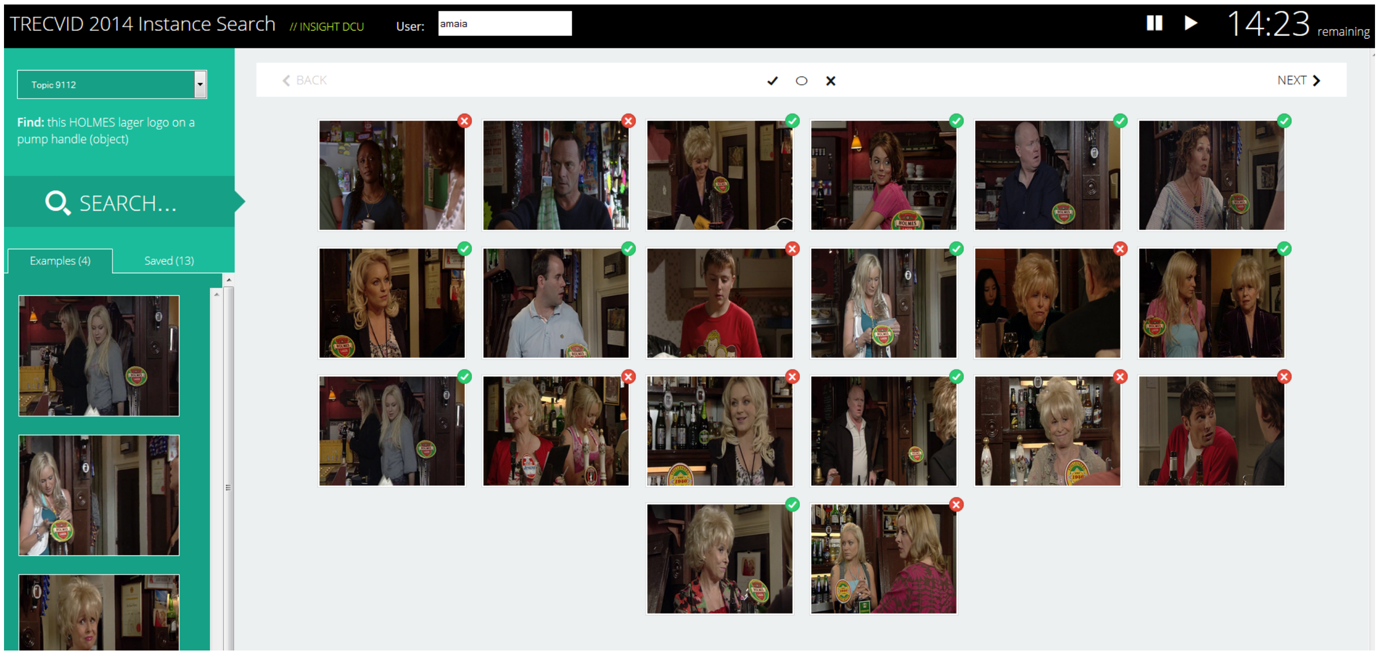


Fig. 16: Screenshot of the interface for TRECVID Instance Search 2014.

$$AP = \frac{1}{|R|} \sum_{k=1}^{N} P(k) \cdot rel(k), \; rel(k) = \begin{Bmatrix} rel(k) = 1 \Leftrightarrow k \in R \\ rel(k) = 0 \Leftrightarrow k \notin R \end{Bmatrix} \quad (1)$$

$$MAP = \frac{1}{|Q|} AP(q) \quad (2)$$

1) Global and Local features: This section describes the experiments performed with both global and local CNN features. First, a study of different local approaches involving object candidates is done using the a toy dataset. After that, the best local approach from the first experiment is tested with the *Ground Truth Subset* using several configurations combining local and global descriptors.

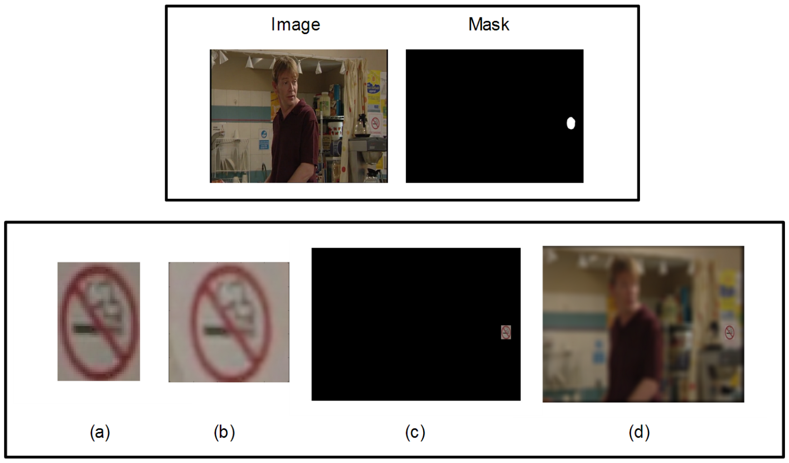


Fig. 17: Examples of binary mask usage. (a) is a cropped bounding box around the mask. (b) is a squared crop around the mask, (c) is a zeroing of the background and (d) is a blurring of the background using a Gaussian filter.

Local approaches with object candidates. As introduced in section IV-B, CNN features can be extracted from either full images or local patches defined by binary masks. The local information can be used in many different ways to discard the background or reduce its importance. Figure 17 provides examples of four different ways to use a binary mask. The

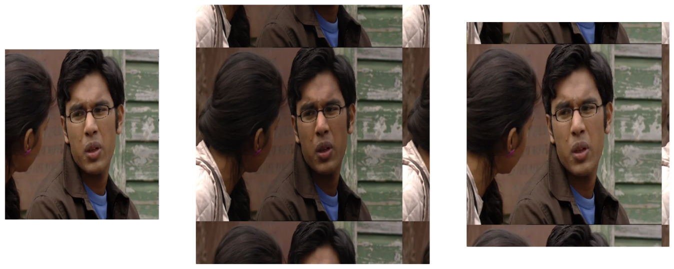


Fig. 18: Adding context with padding of different sizes with respect to the square crop size. 0 padding (left), 1/2 padding (middle), 1/4 padding (right).

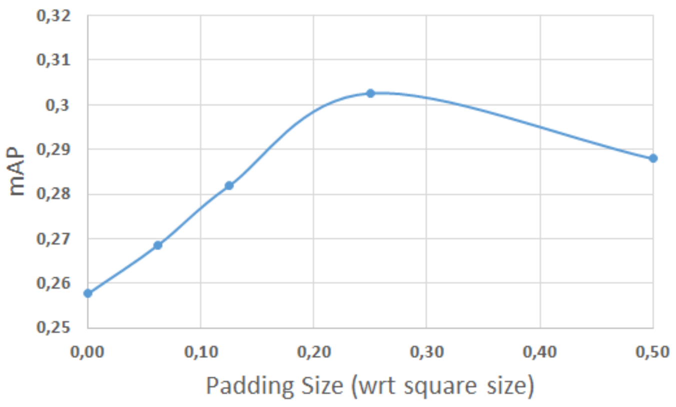


Fig. 19: mAP VS Padding size.

experiment to decide which one is the best configuration was carried out using a *toy subset*, composed by the 120 query images of TRECVID 2013. Every image was considered as a different topic, and the goal was to retrieve the 3 other images of the same topic at the beginning of the ranking list. This toy subset was helpful to quickly find which was the best local technique to use with the binary masks in an ideal scenario, for which we have binary masks for both the query and target images. Table 21 compares the performance of the different masking techniques shown in figure 21, from which we can conclude that the best local approach is to use a cropped square around the object. Then, using squared crops as masking method to compute local CNN features, a more realistic approach was tested. Instead of using the binary masks provided with the *Query subset*, these were replaced by the first N object candidates from SCG [11].

The last experiment that was carried out had the motivation of improving the local features by adding context to the masking method (see figure 18). Figure 19 shows the results that were obtained with different padding sizes, which indicate that adding context with a padding of 1/4 of the size of the square crop outperforms the results using global features. Figure 20 shows the results for several masking techniques using 20, 100 and 250 candidates. Despite these results show that the higher the number of candidates the better the system's performance is, the number of candidates had to be restricted to 20 due to time constraints and lack of computational power when dealing with the *Full dataset*. Namely, the chosen masking method for further experiments was the squared crop with 1/4 of padding.

Combining global and local descriptors. Once a decision was made to compute the local CNN features, the performance was evaluated using the *Ground Truth Subset*. The first local approach consisted in replacing the global features by the local ones using squared crops with 1/4 padding. Then, we tried two approaches combining global and local features:

- Concatenation: CNN features for both global and local approaches are concatenated in a single feature vector.
- Aggregation: CNN features for both global and local approaches are merged but treated independently (global and local feature vectors for the query images are compared to both global and local feature vectors of the target dataset.).

Table 22 compares the performance of the system using global CNN features with the different local approaches mentioned above. The results indicate an increase of performance only when using the concatenation of global and local CNN features.

2) Re-ranking strategies: This section explores different re-ranking techniques that were tested for the design of the pipeline for our submission to TRECVID Instance Search 2014. The goal of this section is to estimate the results that could be achieved with the help of humans using the interface introduced in IV-E. First, the impact of using local features to automatically re-rank the results obtained with global features is assessed. Then, a study on the impact of the user interaction is performed using two different relevance feedback approaches.

Local features. In section IV-F1, we saw how the concatenation of global and local features increased the performance of

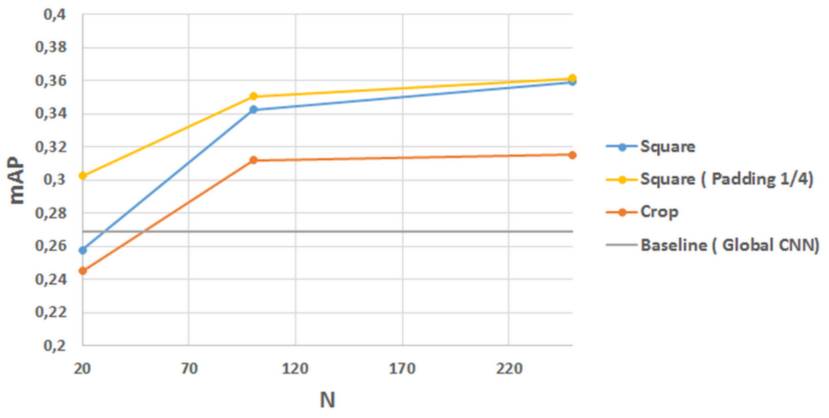


Fig. 20: mAP vs Number of object object candidates for different types of masking techniques.

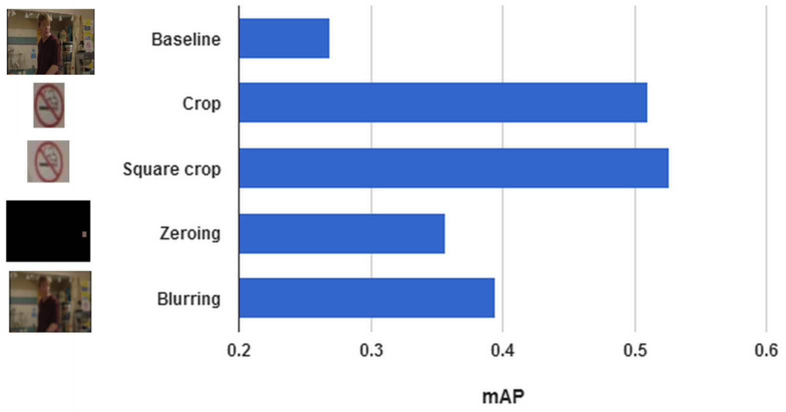


Fig. 21: Results of different local approaches on the toy subset.

the system. However, due to the computational challenges and memory issues that we were facing with the *Full Dataset*, we could only use the local approach as a re-ranking technique after obtaining a ranking using global features. By doing so, mAP increases from 0.1467 to 0.1663. This new ranked lists were the ones displayed in the user interface at the beginning (i.e. the first time that a user performs a search for a topic).

Relevance feedback. With the interface introduced in IV-E, users are asked to annotate the results of the ranked list as either relevant (positive) or non-relevant (negative). This information can be used in several ways in order to improve the ranking.

- **Query expansion.** One thing that can be done is to use the positively annotated keyframes as query images. This means that the ranking produced by each one of those keyframes will be merged with the ones produced by the topic images using the methodology explained in Section IV-D. Conversely, negative annotations are removed from the ranking.
- **SVM Scoring.** Another relevance feedback strategy is to train a classifier using positive and negative annotations collected with the user interface. Then, this classifier can return confidence scores for each one of the keyframes in the database. Then, the new ranking would contain the keyframes sorted by the score returned by the classifier. In this case, both a Linear SVM and a SVM with RBF were tested.

For both configurations, the new ranking would be built by (1) pushing the positive annotations to the top of the ranking, (2) adding the results of the relevance feedback strategy and (3) removing the negative annotations. To be able to estimate the impact of these relevance feedback techniques, the *Ground Truth Subset* was used. The adopted strategy consisted in simulating the users' annotations and use them to evaluate the different relevance feedback techniques. To do so, the results using global features with local features for re-ranking were taken as baseline ranking lists. The positive and negative annotations are simulated by taking all the relevant and non relevant keyframes of the ranking, respectively. This experiment can be repeated several times changing the percentage of the ranking that is being observed, which would simulate the relation between the amount of effort done by the user and the mAP.

Figure 24 shows the results of this experiment by simulating different levels of user effort by truncating the ranking at different positions. As expected, the higher the number of annotations, the higher the mAP. It can also be observed that the Linear SVM strategy greatly outperforms the one using query expansion. Finally, an experiment was carried out in order to

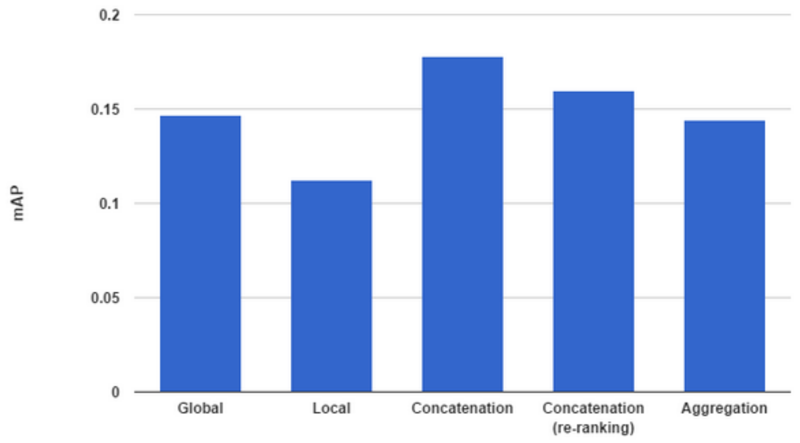


Fig. 22: Results using local features

evaluate the difference in performance for the different query topics. Figure 25 shows the results for each topic using different strategies. This figure clearly shows the difference in performance of our retrieval system for the different queries. Taking these results into account, figure 23 shows a few examples of query topics for which we achieve good and bad results. These examples indicate that our system performs well for those topics in which the context is more useful (outdoors objects), but does not work well for small textured objects (such as logos) or people.

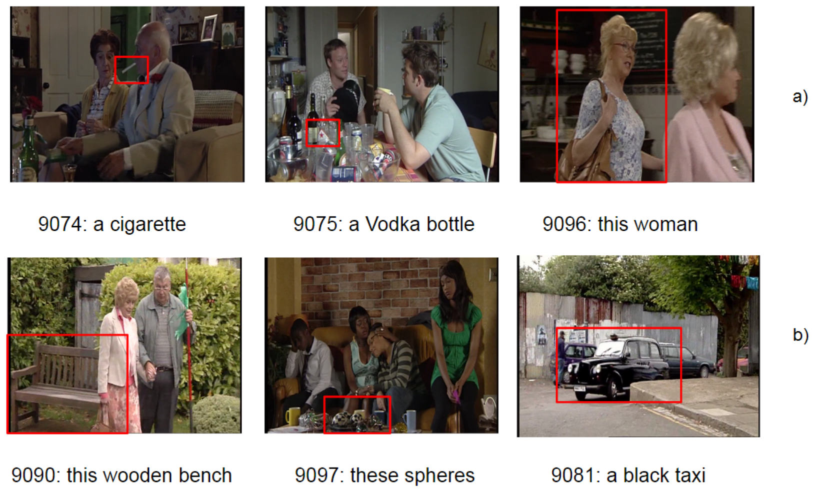


Fig. 23: Examples of query topics for which the retrieval system performs differently. In a), three examples for which the system does not work well. In b) examples of queries with good results (see figure 25 for correspondence).

G. Conclusions and Future Work

This section of the thesis has presented a pipeline for visual instance retrieval, which was used for a submission to TRECVID Instance Search 2014.

First, this work has shown how CNN features can be powerful for the instance retrieval task. The first lesson learned is that our system does not work well for all the query topics. Poor performance has been observed for small textured objects and people, while good results have been achieved for topics for which context is important. This could be solved by treating each query topic differently, according to its category (object, person, location) and its description. For instance, face detectors could be used for queries including people.

Additionally, a study has been performed exploring different possibilities for using local information to have better CNN feature vectors. It has been shown that such approaches can be performed with the help of object candidates to increase the quality of the ranking results.

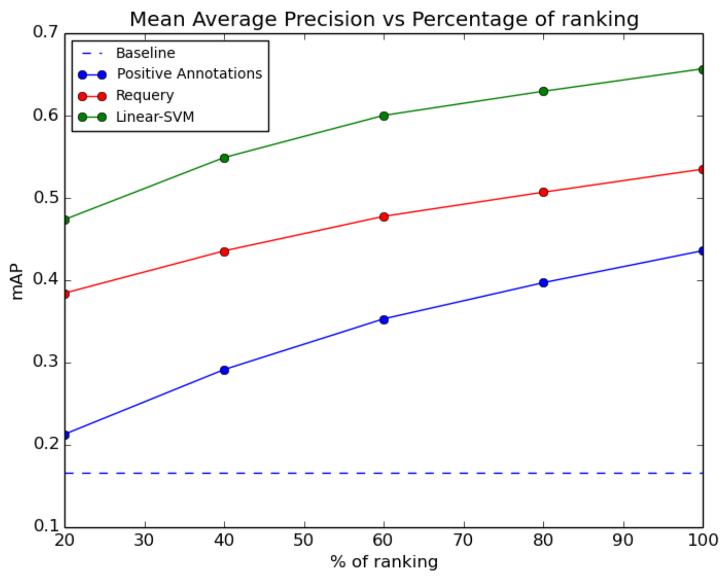


Fig. 24: Simulation of mAP VS Percentage of ranking observed by the user using different relevance feedback strategies. *Baseline* is the mAP of the original ranking. *Positive Annotations* displays the result of a new ranking built only with the positive annotations contained in the observed ranking. Finally, *Requery* shows the results for the query expansion technique and *Linear-SVM* gives the results for the SVM scoring technique (both explained earlier in this section.).

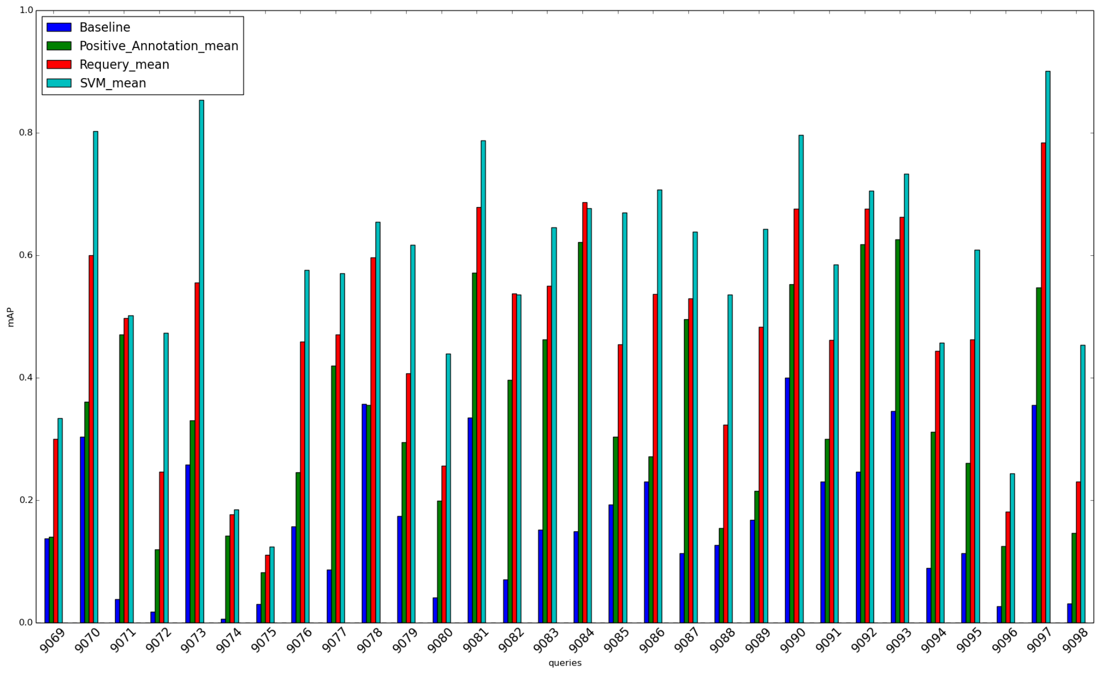


Fig. 25: Results of the simulation of different relevance feedback strategies (by query topic). The baseline mAP and the mean mAP for the different percentages of observed ranking is displayed for each query.

On the relevance feedback side, this section has shown two strategies to use users' annotations to improve the ranking results, and has simulated their impact to the overall performance of the system. Taking these results into account, the presented interface was used to collect annotations from users, which were used along with the introduced relevance feedback techniques for our submission to TRECVID.

All the decisions that were taken during the design of the retrieval system for TRECVID were significantly influenced by the amount of data to process. While some experiments that were performed clearly indicated huge improvements in performance (e.g. higher number of object candidates), some enhancements needed to be discarded to make the problem more treatable with the available resources.

V. Work plan

The purpose of this section is to give a detailed overview of the main tasks that were carried out for the two projects of this thesis. The charts in figure 26 show the work plan that was followed in these projects, reflecting my contribution in each one of the tasks using a color legend. The following list contains the correspondence between the numbers in the charts and the task description:

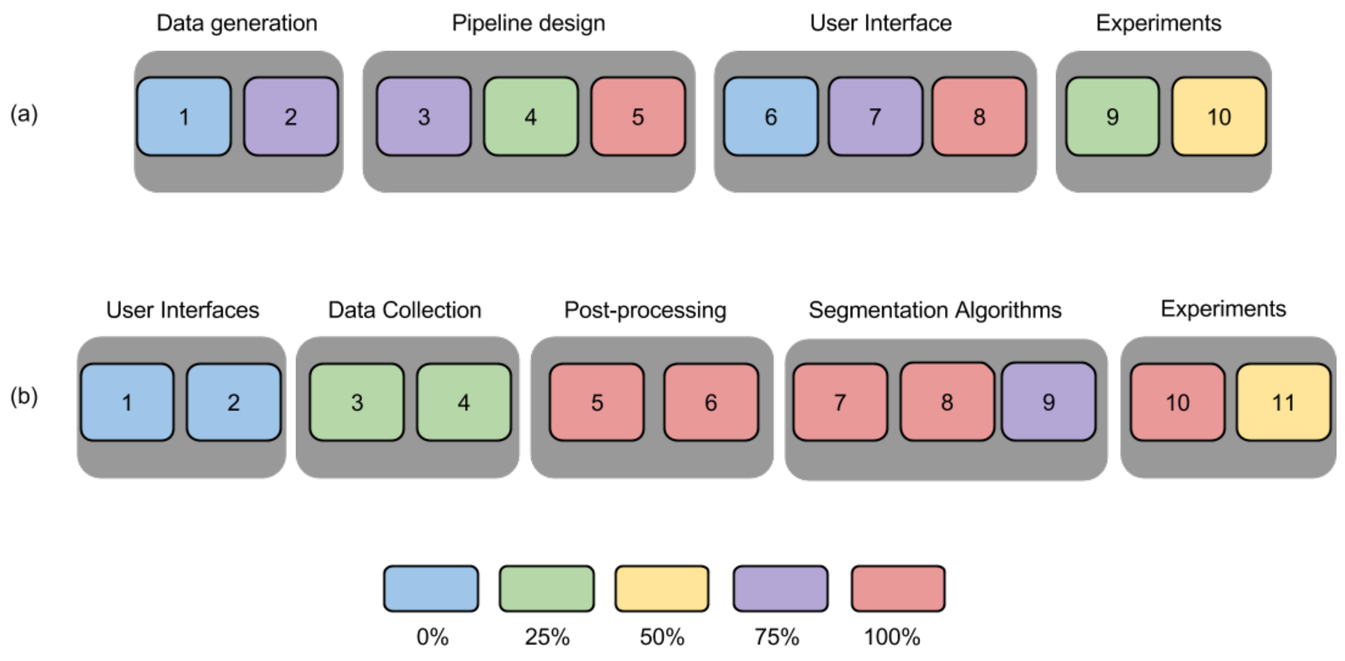


Fig. 26: Work plan of the collaborative projects of this thesis.

(a) Instance Search

1) Keyframe extraction from TRECVID videos.
2) Object Candidates computation for target database.
3) Implementation of baseline code for retrieval.
4) Feature extraction.
5) Implementation of relevance feedback strategies.
6) User interface template design.
7) Integration of the retrieval system to the user interface.
8) Data collection.
9) Experiments on local approaches.
10) Relevance feedback simulation.

(b) Interactive Object segmentation

1) Ask'nSeek implementation and design.
2) Click'n'Cut implementation and design.
3) Data collection with Ask'nSeek.
4) Data collection with Click'n'Cut.
5) Cleanup of users traces.
6) Bounding box generation from users traces.
7) Design of GrabCut Segmentation algorithm.
8) Design of Algorithm 1 for Object Candidates.
9) Design of Algorithm 2 for Object Candidates.
10) Experiment on the comparison of segmentation algorithms.
11) Experiments with Click'n'Cut.

Both projects introduced in this thesis have been developed collaboratively with researchers from Dublin, Toulouse, Barcelona and Florida. Tools such as SVN, GIT, Google Hangouts and Google Drive have been used extensively to organize the work plan and to communicate with the members of the team. The programming languages and packages used for these projects are MATLAB, Python, HTML/AngularJS and MongoDB.

The research line on object segmentation was sponsored by a Research grant from the Spanish Ministry (COLAB). The work introduced in this part of the thesis has been accepted to the CrowdMM Workshop on ACM Multimedia 2014. The instance search part of this thesis was developed during a research internship in Dublin City University, funded by a grant from La Generalitat de Catalunya (MOBINT). This work will be published in the Work Notes for TRECVID 2014.

Acknowledgments

I would like to thank my supervisors Xavier Giró-i-Nieto and Kevin McGuinness for their support and assistance in this project. I also wish to acknowledge the help and contributions provided by Carles Ventura and Eva Mohedano to the TRECVID Instance Search submission on 2014. I would also like to express my appreciation to the advice given by Professor Noel O'Connor, which has been a great help during this project. Finally, I want to thank Axel Carlier, Oge Marques and Vincent Charvillat for their guidance, steadiness and persistance that led to our recent joint contribution to the Crowdsourcing for Multimedia Workshop 2014.

References

[1] Paul Viola and Michael Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. IEEE, 2001, vol. 1, pp. I–511.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2274–2282, 2012.

[3] Pedro F Felzenszwalb and Daniel P Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[4] Jianbo Shi and Jitendra Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.

[5] Dorin Comaniciu and Peter Meer, "Mean shift: A robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[6] Luc Vincent and Pierre Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE transactions on pattern analysis and machine intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[7] Philippe Salembier and Luis Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *Image Processing, IEEE Transactions on*, vol. 9, no. 4, pp. 561–576, 2000.

[8] Pablo Arbelaez, "Boundary extraction in natural images using ultrametric contour maps," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*. IEEE, 2006, pp. 182–182.

[9] J. Carreira and C. Sminchisescu, "Constrained parametric min-cuts for automatic object segmentation," in *CVPR*, 2010.

[10] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum, "Learning to detect a salient object," *PAMI*, vol. 33, no. 2, 2011.

[11] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik, "Multiscale combinatorial grouping," in *CVPR*, 2014.

[12] João Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu, "Semantic segmentation with second-order pooling," in *Computer Vision–ECCV 2012*, pp. 430–443. Springer, 2012.

[13] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik, "Simultaneous detection and segmentation," in *Computer Vision–ECCV 2014*, pp. 297–312. Springer, 2014.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv preprint arXiv:1311.2524*, 2013.

[15] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *IEEE CVPR*, 2014.

[16] C Lawrence Zitnick and Piotr Dollár, "Edge boxes: Locating object proposals from edges," in *Computer Vision–ECCV 2014*, pp. 391–405. Springer, 2014.

[17] JRR Uijlings, KEA van de Sande, T Gevers, and AWM Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[18] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid, "Segmentation driven object detection with fisher vectors," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2968–2975.

[19] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin, "Regionlets for generic object detection," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 17–24.

[20] Vicente Ordonez, Jia Deng, Yejin Choi, Alexander C Berg, and Tamara L Berg, "From large scale image categorization to entry-level categories," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2768–2775.

[21] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, Aug. 2004.

[22] Pablo Arbeláez and Laurent Cohen, "Constrained image segmentation from hierarchical boundaries," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[23] Kevin McGuinness and Noel E O'connor, "A comparative evaluation of interactive segmentation algorithms," *Pattern Recognition*, vol. 43, no. 2, pp. 434–444, 2010.

[24] Suyog Dutt Jain and Kristen Grauman, "Predicting sufficient annotation strength for interactive foreground segmentation," in *ICCV*, 2013.

[25] Xavier Giró-i Nieto, Manuel Martos, Eva Mohedano, and Jordi Pont-Tuset, "From global image annotation to interactive object segmentation," *MTAP*, vol. 70, no. 1, 2014.

[26] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE T. on Image Processing*, vol. 9, no. 4, 2000.

[27] Eva Mohedano, Graham Healy, Kevin McGuinness, Xavier Giro-i Nieto, Noel E O'Connor, and Alan F Smeaton, "Object segmentation in images using eeg signals," *arXiv preprint arXiv:1408.4363*, 2014.

[28] Nima Bigdely-Shamlo, Andrey Vankov, Rey R Ramirez, and Scott Makeig, "Brain activity-based image classification from rapid serial visual presentation," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 16, no. 5, pp. 432–441, 2008.

[29] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft coco: Common objects in context," *CoRR*, 2014.

[31] Luis Von Ahn, Ruoran Liu, and Manuel Blum, "Peekaboom: a game for locating objects in images," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 55–64.

[32] Luis Von Ahn and Laura Dabbish, "Labeling images with a computer game," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2004, pp. 319–326.

[33] Jeroen Steggink and Cees Snoek, "Adding semantics to image-region annotations with the name-it-game," *Multimedia Systems*, vol. 17, 2011.

[34] Cees GM Snoek, Marcel Worring, Ork de Rooij, Koen EA van de Sande, Rong Yan, and Alexander G Hauptmann, "Videolympics: Real-time evaluation of multimedia retrieval systems," *MultiMedia, IEEE*, vol. 15, no. 1, pp. 86–91, 2008.

[35] Klaus Schöffmann and Werner Bailer, "Video browser showdown," *ACM SIGMultimedia Records*, vol. 4, no. 2, pp. 1–2, 2012.

[36] Paul Over, George Awad, Martial Michel, Jonathan Fiscus, Greg Sanders, Wessel Kraaij, Alan F. Smeaton, and Georges Quéenot, "Trecvid 2013 – an overview of the goals, tasks, data, evaluation mechanisms and metrics," in *Proceedings of TRECVID 2013*. NIST, USA, 2013.

[37] Joseph John Rocchio, "Relevance feedback in information retrieval," 1971.

[38] Eleanor Ide, "New experiments in relevance feedback," *The SMART retrieval system*, pp. 337–354, 1971.

[39] Giorgio Gia, Fabio Roli, et al., "Instance-based relevance feedback for image retrieval," in *Advances in neural information processing systems*, 2004, pp. 489–496.

[40] Cees Snoek, Kvd Sande, OD Rooij, Bouke Huurnink, J Uijlings, M Liempt, M Bugalhoy, I Trancosoy, F Yan, M Tahir, et al., "The mediamill trecvid 2009 semantic video search engine," in *TRECVID workshop*, 2009.

[41] Yan-Tao Zheng, Shi-Yong Neo, Xiangyu Chen, and Tat-Seng Chua, "Visiongo: towards true interactivity," in *Proceedings of the ACM International Conference on Image and Video Retrieval*. ACM, 2009, p. 51.

[42] Robin Aly, Relja Arandjelovic, Ken Chatfield, Matthijs Douze, Basura Fernando, Zaid Harchaoui, Kevin McGuinness, Noel E O'Connor, Dan Oneata, Omkar M Parkhi, et al., "The axes submissions at trecvid 2013," 2013.

[43] Kevin McGuinness, Noel E O'Connor, Robin Aly, Franciska De Jong, Ken Chatfield, Omkar M Parkhi, Relja Arandjelovic, Andrew Zisserman, Matthijs Douze, and Cordelia Schmid, "The axes pro video search system," in *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*. ACM, 2013, pp. 307–308.

[44] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez, "Aggregating local descriptors into a compact image representation," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3304–3311.

[45] Carles Ventura, Manel Martos, Xavier Giró-i Nieto, Verónica Vilaplana, and Ferran Marqués, *Hierarchical navigation and visual search for video keyframe retrieval*, Springer, 2012.

[46] Graham Healy and Alan F Smeaton, "Optimising the number of channels in eeg-augmented image search," in *Proceedings of the 25th BCS Conference on Human-Computer Interaction*. British Computer Society, 2011, pp. 157–162.

[47] Ashkan Yazdani, Jean-Marc Vesin, Dario Izzo, Christos Ampatzis, and Touradj Ebrahimi, "Implicit retrieval of salient images using brain computer interface," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 3169–3172.

[48] Axel Carlier, Oge Marques, and Vincent Charvillat, "Ask'nseek: A new game for object detection and labeling," in *ECCV Workshops*, 2012.

[49] Amaia Salvador, Axel Carlier, Xavier Giro-i Nieto, Oge Marques, and Vincent Charvillat, "Crowdsourced object segmentation with a game," in *ACM CrowdMM*, 2013.

[50] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM Transactions on Graphics (TOG)*. ACM, 2004, vol. 23, pp. 309–314.

[51] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222–1239, 2001.

[52] G. Bradski, ," *Dr. Dobb's Journal of Software Tools*.

[53] Kevin McGuinness and Noel E. O'Connor, "A comparative evaluation of interactive segmentation algorithms," *Pattern Recognition*, vol. 43, no. 2, 2010.

[54] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV*, 2001.

[55] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, vol. 88, no. 2, 2010.

[56] Tomasz Adamek, *Using contour information and segmentation for object registration, modeling and retrieval*, Ph.D. thesis, Dublin City University, 2006.

[57] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3384–3391.

[58] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," *arXiv preprint arXiv:1403.6382*, 2014.

[59] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, "Deep fisher networks for large-scale image classification," in *Advances in neural information processing systems*, 2013, pp. 163–171.

[60] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," 2014.

[61] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "Imagenet large scale visual recognition challenge," 2014.

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[63] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky, "Neural codes for image retrieval," *arXiv preprint arXiv:1404.1777*, 2014.