Outlining Objects for Interactive Segmentation on Touch Devices

Matthieu Pizenberg
University of Toulouse
Toulouse, France
matthieu@pizenberg.fr

Axel Carlier
University of Toulouse
Toulouse, France
axel.carlier@enseeiht.fr

Emmanuel Faure
CNRS - IRIT
Toulouse, France
emmanuel.faure@irit.fr

Vincent Charvillat
University of Toulouse
Toulouse, France
vincent.charvillat@enseeiht.fr

ABSTRACT

Interactive segmentation consists in building a pixel-wise partition of an image, into foreground and background regions, with the help of user inputs. Most state-of-the-art algorithms use scribble-based interactions to build foreground and background models, and very few of these work focus on the usability of the scribbling interaction. In this paper we study the outlining interaction, which is very intuitive to non-expert users on touch devices. We present an algorithm, built upon the existing GrabCut algorithm, which infers both foreground and background models out of a single outline. We conducted a user study on 20 participants to demonstrate the usability of this interaction, and its performance for the task of interactive segmentation.

KEYWORDS

interactive segmentation; outlining; usability; user study

1 INTRODUCTION

The number of pictures that are captured, stored and shared online is growing everyday. In march 2017, Facebook reported that 300 millions pictures are uploaded each day on their website. These pictures are increasingly used by companies, or even by individual users, enabling new applications trying to improve everyday life. Object segmentation constitutes an important step towards automatic image understanding which is key to those smart applications.

Object segmentation in an image remains a challenging task. This process of assigning a label to each pixel is very sensitive to the classical difficulties encountered in computer vision: lightning conditions, occlusions, etc. Recent advances in deep learning have enabled researchers to obtain state-of-the-art results [14] by training on the PASCAL segmentation dataset [8]. Some other techniques learn to infer a pixel-wise segmentation out of weak annotations, i.e. bounding boxes around objects [18]. These methods are very promising but need huge amount of human labeled samples in order to train the deep neural network. Recent approaches have tried to overcome this issue, introducing active learning to train deep neural networks using a limited amount of selected samples [13], on the problem of image classification, but none of these methods have yet been applied on semantic segmentation.

Because automatic segmentation is still in many cases out of algorithms' reach, researchers have introduced the concept of interactive segmentation. This problem has often been approached with a task-driven point of view: what type of interaction may bring the necessary information to significantly help an algorithm achieve an acceptable segmentation?

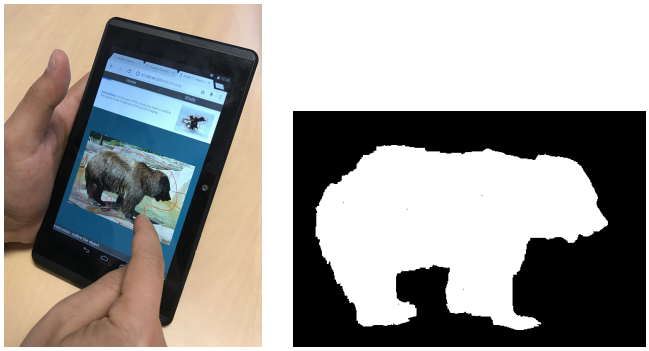


Figure 1: A user outlining an object on a touch device, and the resulting segmentation mask obtained with our method.

The users providing the interactions are always supposed to have a fair understanding of what segmentation is. This assumption is problematic, especially when putting into perspective the extraordinary amount of images to be annotated. That is why, in this paper, our target population is composed of non-expert users who are not knowledgeable about image processing and segmentation. As a consequence most of the existing work, which rely on foreground and background scribbles and require a high cognitive load from the users, are not suitable to our problem.

Instead, we propose to use an intuitive interaction, outlining (see Figure 1), that can be performed quickly and lead to good segmentation results while keeping users from entering a process of iterative segmentation refinement. This outlining interaction is particularly well suited for touch devices, which is appropriate considering the

MM'17, October 23–27, 2017, Mountain View, CA, USA

growing usage of tablets and smartphones compared to computers. All these properties make the outlining interaction very interesting for crowdsourcing segmentation annotations on thousands of images, with non-expert users.

We present two main contributions in this work: first, a modification of the GrabCut algorithm that takes as input an outlining interaction, instead of a bounding box. We take advantage of the free-form shape drawn by the users to extract information about foreground (using the Blum Medial Axis computation) out of a background annotation (the outline). The second contribution of this work is the usability comparison of various interactions that are used in interactive segmentation. We argue that the outline offers the advantage of being a quick, easy-to-understand and usable interaction while providing a high amount of supervision to obtain a good segmentation.

The rest of the paper is organized as follows: we first review the related work in Section 2, we then explain our method to compute segmentation masks out of outlining interactions in Section 3. Finally we present our experiments and the results we find that prove our simple interaction can lead to a good quality segmentation in Section 4.

2 RELATED WORK

2.1 Existing interactions for user-assisted segmentation

Many interactions have been explored in the literature to provide users with a way to bring semantic information to help a segmentation algorithm. We review the interactions in this paragraph and present briefly the algorithms that are attached to them.

The most intuitive methods are the ones that require the user to manually designate the contours of the object. The LabelMe tool [20] (see Figure 2) is the most famous example of such an interface. The web-based interface developed by the authors allows users to draw a polygon around an object. The segmentation obtained with this technique is not necessarily precise at the pixel level, but is sufficient in many cases and has the advantage of being easy for users. In a variant of this technique called the Intelligent Scissors [16], the users click points on the contour of the object and a dynamic programming algorithm searches the optimal path that ties those points. There exists another variation of contour drawing called Soft Scissors [22]. One has to follow the contour using a soft constrained, size-adaptable thick contour brush, requiring less precision than exact polygon contour drawing.

A second possibility for interactive segmentation has been proposed by Rother et al. [19]. The user is only required to draw a bounding box around the object (see Figure 3), which is used to learn a background model. The foreground is then obtained using iterative graph-cut and alpha matting. This method works very well for object that distinctly emerge from a repetitive background. However in the case of complex scenes, the authors allow users to perform an additional refining step based on scribbles.

Scribbles form another category of interactions for segmentation, and are undoubtedly the most widely used (see Figure 3). Users can typically draw foreground and background scribbles on the image, and receive a feedback on the current state of the resulting segmentation mask. Boykov and Jolly [3] use this input to build a trimap,

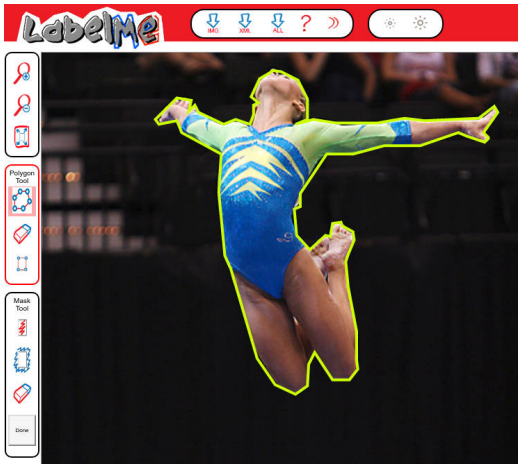


Figure 2: Visualization of an image annotated with the LabelMe tool [20]

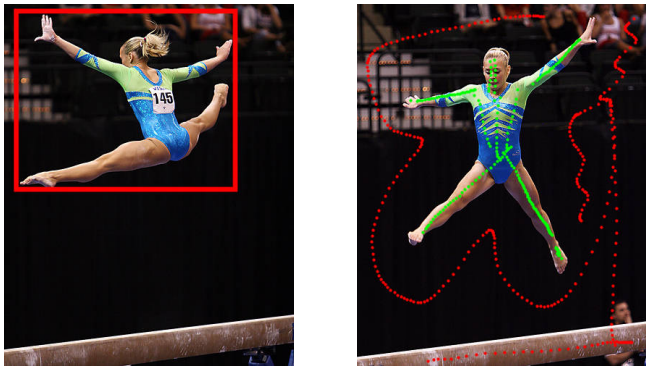


Figure 3: Example bounding box and scribbles interactions. In the left image, a user drew a bounding box around the gymnast. In the right image, a user drew green foreground scribbles on the gymnast and red background scribbles outside.

i.e. a partition of the image into hardly constrained foreground and background regions, and a softly constrained in-between region. They run a graph-cut algorithm to find the optimal object boundary on the softly constrained region. McGuinness and OâĂŹConnor [15] describe how to use scribbles to segment an image using a Binary Partition Tree (BPT) [21]. The BPT is a hierarchy of image segments that can be used to propagate the foreground and background inputs between similar regions. Scribbles have also been used in the context of image co-segmentation [1], to provide foreground and background information across a set of images depicting the same object. As an alternative to scribbles, single foreground and background points have been used as input to select the best masks among a set of object candidates [4].

The mouse is used in most of these work as interaction device, which probably explains why outlines are rarely studied in the literature. Outlines are indeed tedious to perform with a mouse. However, most of the literature algorithms can take outlines as an

input; in our work we choose to use GrabCut to obtain a segmentation out of the outlines.

2.2 Interactive segmentation for non-expert users

Although many work have studied how to incorporate human interactions into segmentation algorithms, there are only very few authors who studied interactive segmentation with a special focus on the interaction matter. In particular, the distinction between experts and non-expert users has rarely been made when studying the performance of interactive segmentation algorithms. Considering the tremendous number of ground truth masks needed to train deep learning algorithms to segment images, the human annotations necessarily have to be provided by non-expert users. Some experiments on crowdsourced segmentation [5] clearly show that non-expert users can misunderstand a simple segmentation HIT (Human Intelligence Task), mistake foreground and background colors, misunderstand the segmentation feedback, etc.

As a matter of fact, most of the crowdsourcing efforts in interactive segmentation have been performed with a LabelMe type of interface. In addition to the actual LabelMe project, some work on medical imaging have crowdsourced segmentation masks by asking users to draw a polygon around hip joints [6], muscle and melanoma cells [9], and nuclei [10].

However, those work do not specifically aim at the usability of their interfaces. Recent works by Korinke et al. [11, 12] give insights on the preferred user interactions for interactive image segmentation on mobile devices. Dividing the process into two steps, initialization and refinement, seems to be the preferred input method. The initial step can be either bounding box drawing or a simple outline.

Our work is largely inspired by these findings; since we target non-expert users, we want to provide the most natural interaction and choose outlining for our interactive segmentation algorithm.

3 OUTLINING OBJECTS FOR INTERACTIVE SEGMENTATION

In this section we explain why we use outlining annotations, and our method to compute segmentation masks out of outlining interactions.

As stated in the previous section, most of prior crowdsourcing campaigns in image segmentation have asked users to draw a polygon around the object. This interaction has some merit in terms of usability: it is very straightforward to understand, and does not require iterative refinement from the user. In addition, the user does not have to evaluate the quality of the produced segmentation mask to know when to stop interacting. When the polygon is drawn, the segmentation is over.

However, we have two main concerns with this interaction. First, it is tedious and time consuming. It requires users' full attention, in order to precisely click on the object boundary. It also requires users to implicitly determine the number of edges of the polygon they should draw. A second limitation of this interaction is the quality of the segmentation mask obtained. Shape details and curved boundaries can only be approximated by a polygon, and their quality is correlated with the time the human annotator is willing to spend annotating.

Outlining an object has the same merits than drawing a polygonal shape around the object: the task is easily defined, and it is easy for a user to assess the quality of an outline. It also adresses the first limitation of the polygons: since it requires less precision in following the object boundaries, it is less tedious and time consuming. It has nevertheless an important drawback: it does not provide an accurate segmentation.

In order to address this problem, we choose to rely on the popular Grabcut algorithm [19]. The original GrabCut takes a bounding box as an input. It considers every pixel outside of the bounding box as fixed background, and aims at separating foreground from background inside the bounding box. To this end, a background model is estimated from the fixed background, and a foreground model is estimated from the pixels inside the bounding box. The likelihood of each pixel inside the bounding box to be foreground or background is then estimated, and graph-cut is applied to obtain a temporary segmentation mask. This mask is then used to update the foreground and background models, and the process is iterated until convergence.

In our implementation, we slightly alter the GrabCut algorithm to take into account a major difference between outlines and bounding boxes: we can make stronger assumptions on the foreground positions from an outline than from a bounding box, by looking at the general shape of the outline. We restrict the initial foreground model computation to the pixels that are most likely to be foreground, which decreases the number of iterations needed for convergence and improves the segmentation quality.

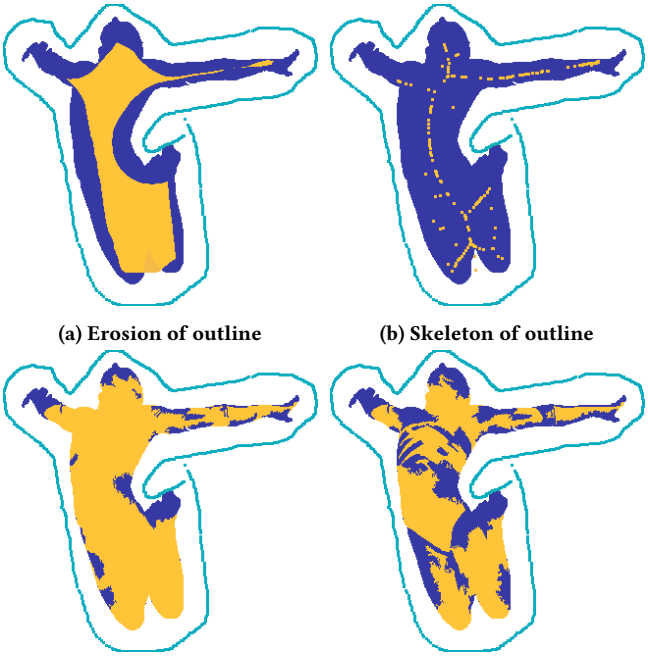
In the rest of the section, we explain two different methods to infer foreground out of the ouline shape: the first method consists in eroding the outline, and the second is based on the Blum medial axis computation. We then post-process the foreground pixels using superpixels.

3.1 Outline erosion

The simplest method to obtain points that are likely to be foreground from an outline is to apply morphological erosion of a mask representing the inside points of the outline. We use a disk as a structuring element for the erosion, and the only parameter of this method is the radius of the disk.

In our implementation, the disk radius is specific to each user and computed by studying the outline performed by the user on a Gold Standard image. We compute the mean m_d and standard deviation s_d of the distance d from each outline point to the ground truth mask. Assuming the user consistently outlines all images, i.e. the mean distance of the user outline to an object is more or less constant across all images, a disk radius equal to $m_d + 2 \cdot s_d$ should produce an eroded outline that is almost certainly completely foreground.

An example of this process can be visualized on Figure 4a. The eroded outline (yellow) is almost entirely contained in the ground truth mask (dark blue).



(a) Erosion of outline (b) Skeleton of outline

(c) Erosion of outline extended with superpixels (d) Skeleton of outline extended with superpixels

Figure 4: Different foreground inferring methods from a user outline. The ground truth mask is in dark blue. The user outline is in cyan. The inferred foreground is in yellow.

3.2 Blum medial axis algorithm

In shape analysis and model animation, the Blum medial axis transform [2] is one of the most popular tools. The Blum medial axis of a shape is composed of the centers of the circles that are tangent to the shape in at least two points. It is especially appropriate to compute skeletons, composed of the medial axis points inside the shape.

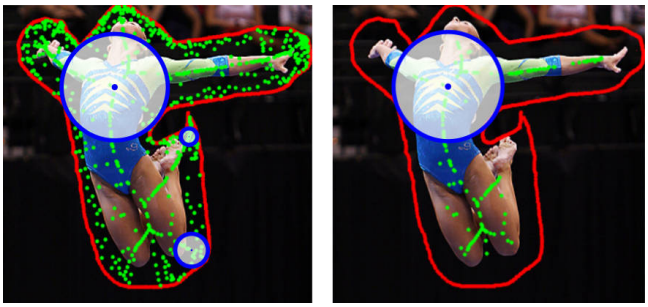


Figure 5: Skeleton (in green) computed using the Blum medial axis algorithm from an outline (in red). Few example disks are shown in blue. In the image on the left, all disks centers (green points) are kept, generating a very noisy skeleton. In the image on the right the skeleton is pruned, by filtering out centers of small disks.

One of the problems of the medial axis algorithm is its stability when the shape frontier is noisy. It tends to create a high number of branches (see Figure 5), which deteriorates the simplicity of the skeleton, and incidentally the comprehension of the shape. In our case, this is rather an advantage. Indeed more ramifications lead to a higher number of points inside the shape for our foreground scribbles. However, we need to filter the inside points, since those close to the outline have a high probability of being outside of the object to segment. Radius of the inside circles of medial axis points constitute a good filter option, because the medial axis points with the smaller radius are typically close to the outline and appear as a consequence of the outline noise. In our implementation, we choose to keep only centers with a radius higher than half the larger radius. Figure 4b depicts a ground truth mask in dark blue, a user outline in cyan and the filtered medial axis points in yellow. Most of the yellow points fall inside the ground truth mask, thus making it a good starting point to learn the foreground model.

3.3 Enhancing foreground with superpixels

These two methods, Blum medial axis and outline erosion, allow to select foreground points that make a valuable input to the GrabCut algorithm. However, we add a post-processing step to (i) extend this foreground information and (ii) filter as much false foreground points as possible.

To do so, we compute a superpixels segmentation of the image, i.e. an oversegmentation that groups neighbouring pixels with similar colorimetric properties. We (i) extend the foreground labels from pixels to the superpixels they belong to. This considerably increases the surface of the foreground region. In addition, we (ii) handle conflicting superpixels, which contain both pixels denoted as foreground and a piece of the outline, by removing them from the foreground mask. An example of the result can be seen on Figure 4c and Figure 4d. Note that the errors arising from the first step (between the knees in Figure 4a and Figure 4b) have successfully been removed in the post-processed inferred foreground mask.

We choose to use the Mean-Shift superpixels [7] because no compacity constraint is used in their computation. As a consequence, a superpixel can cover a large area (especially in the case of similar background regions, such as an homogeneous sky) and will more likely correct wrongly inferred foreground points.

4 EXPERIMENTS

In this section we describe the setup of our experiments and analyze the outcome of the study.

4.1 Experimental setup

Interactions Since the purpose of the study is interactive segmentation on touch devices, we choose to compare only three annotations: outlines, scribbles, and bounding boxes. We do not include polygon drawing because it is clearly not adapted to a touch device. Indeed, fingers are too big to precisely touch the boundary of an object, they would hide the area where the user would try to place the vertex on.

The interfaces are kept as simple as possible. The user is shown an image and has to provide a valid input to be allowed to move on to the next image.

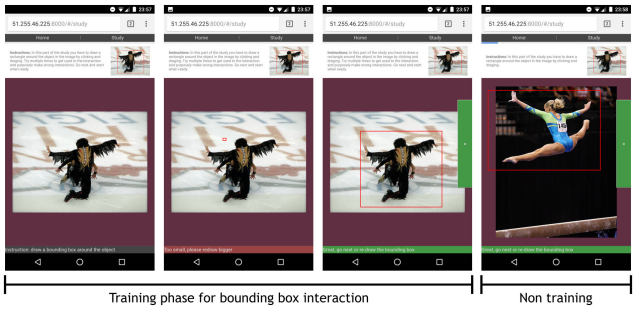


Figure 6: Several screenshots of the bounding box interface during training phase (left), and during the study (right).

The bounding box interface allows the user to draw a rectangle over the image using a touch and drag interaction. If the user is not satisfied with their previous attempt, they can start over, which will replace the former rectangle with a new one. The user can only move on to the next image when the current rectangle is of sufficient size (we discard rectangles that are too small to avoid common mistouch issues).

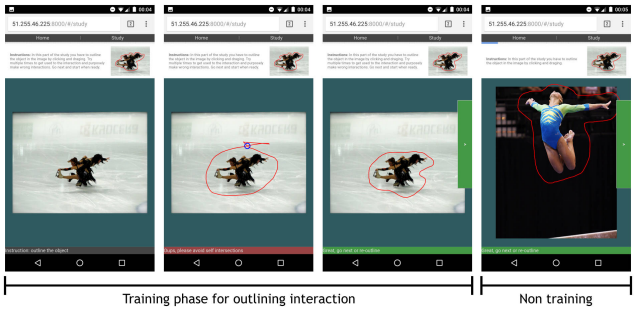


Figure 7: Several screenshots of the outlining interface during training phase (left), and during the study (right).

The outlining interface is very similar to the bounding box interface. The user can draw the outline using a touch and drag interaction; the system automatically draws the closing segment between the ending and starting points when the user releases their finger. The user can also start over if not satisfied with the current outline. The system allows the user to move on to the next image if the outline is of sufficient area. In addition, for the training image only, the system checks the absence of loops in the outline path (see Figure 7), for they may reveal incorrect usage. This loop detection feature is deactivated for the other images to limit its impact on the interaction and user frustration.

The scribbling interface displays three buttons: one to select the foreground scribbles, which are drawn in green, one to select the background scribbles, which are drawn in red, and one to remove the last drawn scribble. Users are required to provide a certain scribble length to be allowed to move on to the next image.

Device and software We used a regular 8" android tablet, for which the buttons appeared large enough to be easily clickable. The user study being a web application, it was conducted in the chrome browser for android. The code for this study (web client and server), as well as the results presented here are all available online (github.com/mpizenberg/otis).

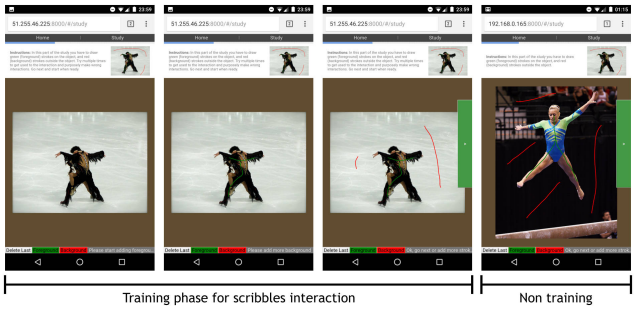


Figure 8: Several screenshots of the scribbling interface during training phase (left), and during the study (right).

Images We select 36 images from the iCoseg dataset [1], which we divide into 3 groups of 12 images. We want the segmentation results to be comparable between different interactions, but since each user tests the three interfaces, we do not want to use the same images for the three phases of the study. This would indeed risk biasing the results: users might get annoyed of annotating three times the same images and this could affect the quality of their annotation, for example.

The iCoseg dataset provides multiple images depicting the same object in different situations. Examples of these images can be seen on Figure 9.

Figure 9: Some images from the iCoseg dataset

Methodology The protocol of the study is as follows.

The users are not explained the concept of segmentation, we tell them that we require annotations on images, and that we wish to compare three interactions to provide those annotations.

The study is composed of three steps, one step per interaction. For each step, the evaluator first explains the user how the interaction works, and demonstrates it on a training image. The evaluator demonstrates good and bad examples of interactions. Then the user tests the interaction on the same training image. The evaluator can correct the user and criticize or validate the users interactions. Once the user completely understands the tool, the eleven other images are proposed for interaction. Finally, at the end of each step, the user answers two questions about the interaction.

In order to limit bias, the order of the interactions is randomized, as well as the order of appearance of each image during each step.

Among the eleven images annotated by the user, one is considered Gold Standard. It is introduced to (i) check whether the user is performing the task correctly (this is particularly useful in a crowdsourcing context), and (ii) to learn the radius of the erosion disk (see Section 3.1).

The two questions asked at the end of each step of the study are:

- Overall, I am satisfied with the ease of completing the tasks in this scenario
- Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario

The users can answer on a scale from 1 (strongly agree) to 7 (strongly disagree). We choose to ask only these two questions because we are not trying to assess the usability of a whole system, but only of an interaction. A standard usability questionnaire, such as SUS (used in [12]), was not really adapted to our use case and instead we extracted these two questions from a popular post-task questionnaire (ASQ, After Scenario Questionnaire).

Finally at the end of the study, we ask users to rank the three interactions in their order of preference.

Participants Twenty users (10 Male, 10 Female) participated to this study, with ages ranging from 25 to 55 years old. Most users have no experience in image segmentation, and some of them are familiar with the concept.

4.2 Usability metrics

Among the criteria stated by Nielsen [17] as defining the usability of a system, we are able to evaluate efficiency, errors, and user satisfaction.

Efficiency designates the swiftness with which users were able to complete the tasks once they learned how to interact with the system. We evaluated this criteria both subjectively, by asking users about their perception of the time they spent on the task (table 1), and objectively by measuring the time it took them to complete their interactions on each image (see Figure 10).

User satisfaction is measured through our questionnaire, both by the question on the perceived task easiness and the interaction ranking.

Finally, errors are measured by counting the number of times the users redrew a bounding box or an outline around the object (for the bounding box and outline interactions), and by the number of clicks on the *Undo last scribble* button in the case of the scribbling interaction (see Figure 11).

Method	Bounding box	Outline	Scribble
Ease	2.1 ± 0.62	2.65 ± 0.74	2.1 ± 0.61
Time	2.35 ± 0.69	2.5 ± 0.67	2.6 ± 0.70
Rank	1.95 ± 0.43	1.90 ± 0.32	2.15 ± 0.37

Table 1: Results of the questionnaire with a 95% confidence interval

Overall, the questionnaire results can not allow us to conclude on the superiority of one interaction method over the others. Although

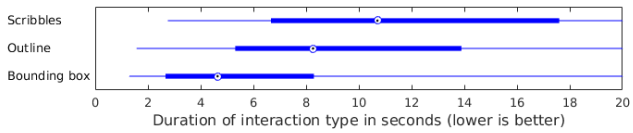


Figure 10: Duration of interactions, on all images and all users. The dots are the median durations, and the thick blue line limits the first and third quartiles.

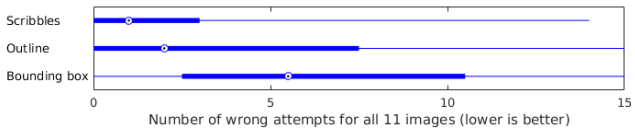


Figure 11: Number of errors per interaction and per user on all images. The dots are the median number of errors, and the thick blue line limits the first and third quartiles.

slightly in favor of the bounding box interaction, the perceived ease and time are not statistically better for any of the three interactions. However the results are all between 2 and 3 (on a scale from 1 to 7), which mean users were mostly satisfied with all three interactions. We can note that the time perception results (table 1) are correlated with the objective duration of interaction (Figure 10), measured during the experiments. The bounding box is obviously the quickest interaction, while the scribbles suffer from the time needed to switch between foreground and background scribbling.

Surprisingly, the outline ranks first in the users preference (although not significantly), ahead of the bounding box interaction. The reason of this observation, as explained by many of the participants during the experiment, is due to to the frustration that can arise when trying to draw a bounding box around a non-convex object. Users who were trying to draw the bounding box as close as possible from the object boundary often had to use several attempts, because of the difficulty to position the first bounding box corner. This problem is very visible on Figure 11, which shows the high number of errors made by users when drawing the bounding box. The errors that occurred during the outline study were mostly due to a too high interaction speed, or to the users hand masking the object during interaction, for users who were the less familiar with touch devices.

4.3 Interactions informativeness

We define the background area of user inputs as follows. For a bounding box (resp. outline), the background area is composed of all pixels outside of the bounding box (resp. outline). For scribbles, the background area is the union of the superpixels annotated as background (containing part of a background scribble).

Looking at the precision of background user inputs (Figure 12) we see that more than 75% of user annotations are perfect (a precision score of 1). This means that 75% of user inputs do not intersect at all the object of interest. We can conclude that users understand well the tasks they are given.

In order to estimate the informativeness of an interaction, we also measure the recall index (Figure 13). It indicates the percentage

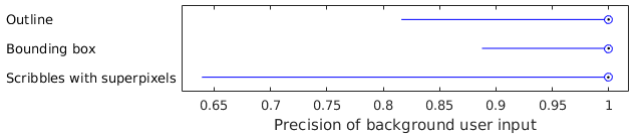


Figure 12: Precision of background user input

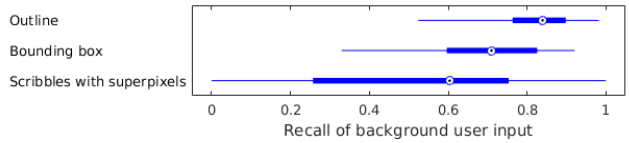


Figure 13: Recall of background user input

area of all background that is annotated by an interaction. With no surprise, outlining is the more informative since it is often very close to the boundary of the object (see Figure 18) and thus, the outside of the outline covers most of the image. Background (red) scribbles are the least informative here since only superpixels that are scribbled over count as background information.

Except for the foreground (green) scribble interaction, we do not have raw foreground annotations. We thus define the foreground input area as the inferred foreground (through erosion or medial axis computation, extended by superpixels as explained previously).

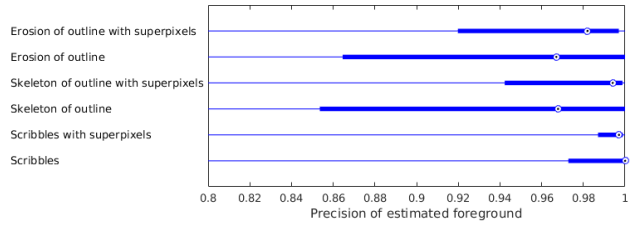


Figure 14: Precision of foreground user input

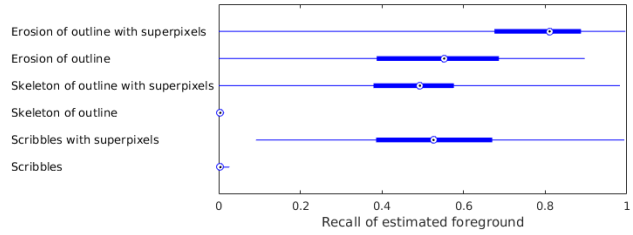


Figure 15: Recall of foreground user input

The precision of foreground area is given in Figure 14, relatively to the ground truth masks. We can observe that more than 75% of foreground (green) scribble inputs are over the 0.97 index. This means that the task of scribbling inside the objects is globally well performed but still slightly harder than background (red) scribbles. It is explained by the fact that objects can have thin shapes and thus not precisely locatable under the finger during the touch interaction.

Using the superpixels extension of the scribbles, we observe that the smart background correction mentioned in Section 3.3, enhances the 75% index to a precision of 0.99. With the two foreground inference techniques (erosion and skeleton), the improvement provided by the superpixels extension is obvious.

The recall of foreground area (Figure 15) provided by these interactions, extended through superpixels is also coherent with what we observe in Figure 4. Skeleton and scribbles recall values are almost 0 since they are of dimension 0/1 (points/lines) for a measure of surfaces (dimension 2). Erosion provides the most foreground information, but has the lower precision rate (Figure 15). We will show in the next section that this trade-off is worth exploring.

4.4 Segmentation quality

We computed the resulting segmentation of images using five different methods. As a reference method, the mean Jaccard index obtained with foreground and background scribbles is 0.79 (see table 2). When using the bounding boxes, determining a clear background model input for the GrabCut algorithm, the mean Jaccard increases to 0.82. As expected, it increases even more when using outlining interaction inputs, providing richer inferred initial foreground models to the GrabCut algorithm. The higher scores (0.88 and 0.89) are respectively obtained when using the erosion and skeleton processing of the outline. The best performance is achieved using the skeleton processing, which tends to show that for the results presented in the previous section, the precision of the foreground user input is more important than its recall.

Method	Scrib.	B. Box	Outl.	Outl. + er.	Outl. + BMA
Mean Jaccard	0.79	0.82	0.86	0.88	0.89

Table 2: Mean Jaccard index obtained on all images for all users for each interaction

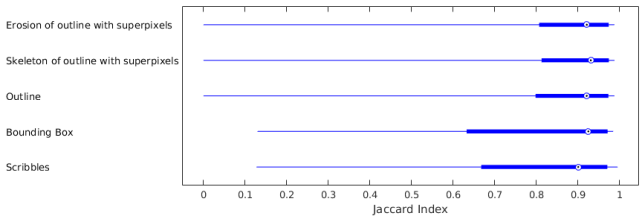


Figure 16: Jaccard index obtained on all images for all users for each interaction type. The dot represents the median value, and the thick blue line limits the first and third quartiles.

Perhaps more importantly, the outlining interaction enables reaching consistently higher Jaccard index than the other techniques. In Figure 16, we observe that the first quartile is always higher than 0.8 with variants of the outlining interaction. Some final segmentation results are visible in Figure 17 and show the clear improvement brought by an outline over a bounding box.

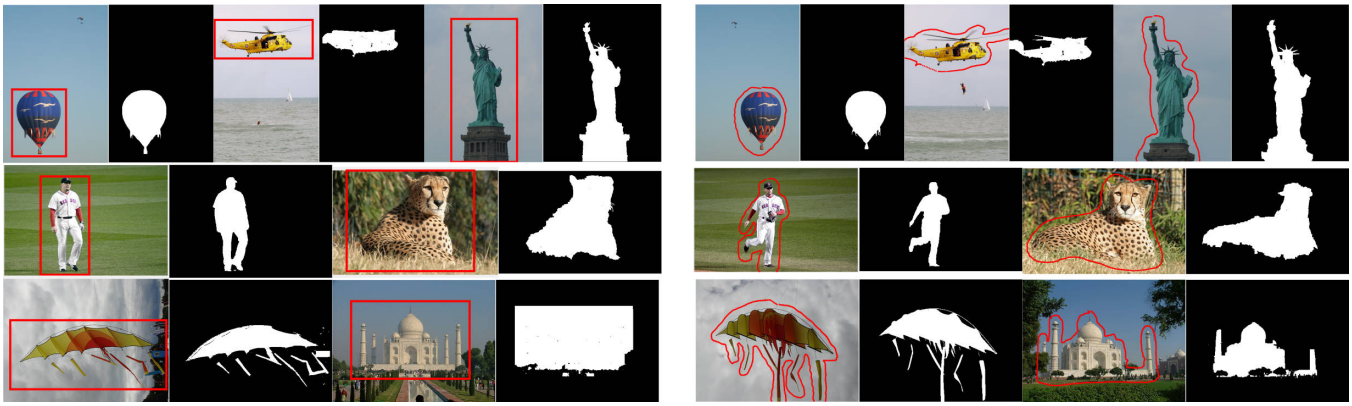


Figure 17: Segmentation results for bounding box and outlining interactions from a single user.

4.5 Discussion

All the results we obtained confirm the good properties of the outlining interaction, in the perspective of being used in a segmentation crowdsourcing campaign.

First, it is a very straightforward interaction. One of the users explained it in these terms: *outlining is easier since you do not need to think, just trace the object. Bounding boxes are tougher, particularly in determining a correct size, and scribbles is too much thinking and a bit more time consuming.* Another user said: *It's actually more fun to draw around object and would seem to me less tiring than the other methods.* The usability criteria point out that outlining might be a little less usable than drawing a bounding box, and comparable to scribbling, but remains a very usable interaction.

Another interesting property of the outlining interaction is the speed at which it can be performed. Figure 10 shows that most of the outlines were produced in less than 10 seconds, which is very reasonable considering some of the images we chose have complex shapes (see Figure 18).

Figure 18: Outlines drawn by the third user on three images with complex shapes.

The quantity of information brought by outlines is also very good, as discussed in the previous section, especially when compared with the interaction usability. This information is of course less than a polygon drawn on the boundary of the object (such as in LabelMe), but can be augmented using computer vision techniques (Blum medial axis, superpixels, GrabCut, etc.) and lead to very good segmentation masks. The average Jaccard index value of 0.89 obtained with the outlines is particularly impressive considering it was obtained without any refinement, and in less than 10 seconds in average (see for example the comparison with Jaccard index vs. time curves described in [4]).

5 CONCLUSION

In this paper, we studied the outlining interaction on touch devices for interactive segmentation. We found that outlining is a simple and natural interaction, that allows to quickly obtain a good information on the location of an object. This information can be augmented with foreground inference, and then used to compute a segmentation mask. The segmentation masks obtained with this method reach an average Jaccard index of 0.89, which is a very good result considering the interaction does not require any knowledge on image processing or computer vision from the user.

Because of all these good properties (simplicity, swiftness, accuracy), outlining seems an interesting avenue to explore for the gathering of large datasets of image segmentation masks. Those datasets are crucial to bring the automatic image segmentation algorithms, today mostly based on deep learning techniques, to a new level of effectiveness. It is our intention to pursue this goal as a future work and launch a crowdsourcing campaign to build such datasets.

REFERENCES

[1] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. 2010. iCoseg: Interactive co-segmentation with intelligent scribble guidance. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3169–3176. DOI: http://dx.doi.org/10.1109/CVPR.2010.5540080

[2] Harry Blum and Roger N Nagel. 1978. Shape description using weighted symmetric axis features. *Pattern recognition* 10, 3 (1978), 167–180.

[3] Y. Y. Boykov and M. P. Jolly. 2001. Interactive graph cuts for optimal boundary amp; region segmentation of objects in N-D images. In *Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Proceedings*, Vol. 1. 105–112 vol.1. DOI: http://dx.doi.org/10.1109/ICCV.2001.937505

[4] Axel Carlier, Vincent Charvillat, Amaia Salvador, Xavier Giro-i Nieto, and Oge Marques. 2014. Click'N'Cut: Crowdsourced Interactive Segmentation with Object Candidates. In *Proceedings of the 2014 International ACM Workshop on Crowdsourcing for Multimedia (CrowdMM '14)*. ACM, New York, NY, USA, 53–56. DOI: http://dx.doi.org/10.1145/2660114.2660125

[5] Axel Carlier, Amaia Salvador, Ferran Cabezas, Xavier Giro-i Nieto, Vincent Charvillat, and Oge Marques. 2016. Assessment of crowdsourcing and gamification loss in user-assisted object segmentation. *Multimedia tools and applications* 75, 23 (2016), 15901–15928.

[6] Alberto Chávez-Aragón, Won-Sook Lee, and Aseem Vyas. 2013. A crowdsourcing web platform-hip joint segmentation by non-expert contributors. In *Medical Measurements and Applications Proceedings (MeMeA), 2013 IEEE International Symposium on*. IEEE, 350–354.

[7] Dorin Comaniciu and Peter Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence* 24, 5 (2002), 603–619.

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

[9] Danna Gurari, Diane Theriault, Mehrnoosh Sameki, Brett Isenberg, Tuan A Pham, Alberto Purwada, Patricia Solski, Matthew Walker, Chentian Zhang, Joyce Y Wong, and others. 2015. How to collect segmentations for biomedical images? A benchmark evaluating the performance of experts, crowdsourced non-experts, and algorithms. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE, 1169–1176.

[10] Humayun Irshad, Laleh Montaser-Kouhsari, Gail Waltz, Octavian Bucur, JA Nowak, Fei Dong, Nicholas W Knoblauch, and Andrew H Beck. 2014. Crowdsourcing image annotation for nucleus detection and segmentation in computational pathology: evaluating experts, automated methods, and the crowd.. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*. NIH Public Access, 294–305.

[11] Christoph Korinke. 2015. Intuitive Input Methods for Interactive Segmentation on Mobile Touch-Based Devices. In *Proceedings of the 23rd ACM International Conference on Multimedia (MM '15)*. ACM, New York, NY, USA, 645–648. DOI: http://dx.doi.org/10.1145/2733373.2807993

[12] Christoph Korinke, Nils Steffen Worzyk, and Susanne Boll. 2015. Exploring Touch Interaction Methods for Image Segmentation on Mobile Devices. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia (MUM '15)*. ACM, New York, NY, USA, 84–93. DOI: http://dx.doi.org/10.1145/2836041.2836049

[13] Peng Liu, Hui Zhang, and Kie B Eom. 2017. Active deep learning for classification of hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10, 2 (2017), 712–724.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

[15] Kevin McGuinness and Noel E OâĂŹconnor. 2010. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition* 43, 2 (2010), 434–444.

[16] Eric N Mortensen and William A Barrett. 1995. Intelligent scissors for image composition. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 191–198.

[17] Jakob Nielsen. 1994. *Usability engineering*. Elsevier.

[18] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. 2015. Weakly-and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 1742–1750.

[19] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. In *ACM SIGGRAPH 2004 Papers (SIGGRAPH '04)*. ACM, New York, NY, USA, 309–314. DOI: http://dx.doi.org/10.1145/1186562.1015720

[20] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. 2008. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision* 77, 1 (2008), 157–173.

[21] Philippe Salembier and Luis Garrido. 2000. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE transactions on Image Processing* 9, 4 (2000), 561–576.

[22] Jue Wang, Maneesh Agrawala, and Michael F Cohen. 2007. Soft scissors: an interactive tool for realtime high quality matting. In *ACM Transactions on Graphics (TOG)*, Vol. 26. ACM, 9.