Latency management in scribble-based interactive segmentation of medical images

Houssem-Eddine Gueziri, Michael J. McGuffin and Catherine Laporte

***Abstract*—Objective: During an interactive image segmentation task, the outcome is strongly influenced by human factors. In particular, a reduction in computation time does not guarantee an improvement in the overall segmentation time. This paper characterizes user efficiency during scribble-based interactive segmentation as a function of computation time.**

Methods: We report a controlled experiment with users who experienced 8 different levels of simulated latency (ranging from 100 to 2000 milliseconds) with two techniques for refreshing visual feedback (either *automatic*, where the segmentation was recomputed and displayed continuously during label drawing, or *user-initiated*, which was only computed and displayed each time the user hits a defined button).

Results: For short latencies, the user's attention is focused on the automatic visual feedback, slowing down his/her labelling performance. This effect is attenuated as the latency grows larger, and the two refresh techniques yield similar user performance at the largest latencies. Moreover, during the segmentation task, participants spent in average $72.67\,\% \pm 2.42\,\%$ for automatic refresh and $96.23\,\% \pm 0.06\,\%$ for user-initiated refresh of the overall segmentation time interpreting the results.

Conclusion: The latency is perceived differently according to the refresh method used during the segmentation task. Therefore, it is possible to reduce its impact on the user performance.

Significance: This is the first time a study investigates the effects of latency in an interactive segmentation task. The analysis and recommendations provided in this paper help understanding the cognitive mechanisms in interactive image segmentation.

I. Introduction

IMAGE segmentation consists in extracting one or several objects of interest from a given image. The gold standard for performing segmentation is to manually delineate the object boundary. This is tedious, time consuming and subject to large inter-operator variability [23]. On the other hand, fully-automated methods provide fast and repeatable segmentation results, but are prone to failures, limiting their application in complex scenarios. A third approach is to *interactively assist* the user during the segmentation to reduce the user's workload and reduce variability in the results, while allowing the user to supervise the segmentation. The success of such approaches relies on: (i) the robustness of the algorithm to estimate the object boundary given the user inputs, and (ii) the ability of the user interaction mechanism (UIM) to interpret the user's intention efficiently. While the first criterion represents the computational efficiency of the algorithm, limitations in the second criterion are know as the *gulf of execution* [28] and represents the gap between the user's goal and what the system allows him/her to do.

H.-E. Gueziri and C. Laporte are with the Department of Electrical Engineering. École de technologie supérieure, Montreal, Canada.

M.J. McGuffin is with the Department of Software and IT Engineering. École de technologie supérieure, Montreal, Canada.

This work was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds de recherche du Québec - Nature et technologies (FRQNT).

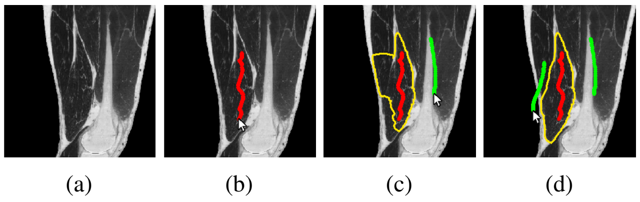


Fig. 1: Scribble-based segmentation example: (a) the original image; (b-d) the user-drawn foreground (red) and background (green) labels yielding the computed segmentation (yellow).

Previous evaluations of interactive segmentation have focused on algorithmic runtime, with few studies of human factors. One preliminary study [10] found that improvements in computational performance did not yield a commensurate improvement in *overall* segmentation performance, i.e, the total time including user actions. Although computation time was improved by a factor ~ 10 (from $1.23\,\mathrm{s}$ to $0.13\,\mathrm{s}$), overall segmentation time was only improved by a factor ~ 2 (from $25.25\,\mathrm{s}$ to $17.08\,\mathrm{s}$). Still, the approaches tested led to similar accuracy. These results point to the importance of assessing the user's role in interactive segmentation processes [30, 20]. In turn, the impact of the user depends on his/her degree of involvement required by the UIM during the segmentation task. The goal of this paper is to provide insight into the factors that affect the *user's performance* during an interactive segmentation task.

However, UIMs vary significantly from one method to the next. For example, in contour-based segmentation [24, 5, 6], the UIM focuses the user's actions on tracing the *boundary* of the object, while in region-based segmentation [1, 40, 27], the user's actions are mostly focused on identifying the *inside* of the object. This difference in the UIM does not necessarily induce a difference in the quality of the final segmentation result. Depending on the application, a given UIM can be more suitable than another. Moreover, different UIMs can be combined into a single segmentation framework [11, 36, 40]. Nevertheless, we can expect that approaches sharing similar UIMs lead to consistent and comparable reactions on part of the user. In this paper, we study the user's performance in *scribble-based* segmentation (Fig. 1), a popular paradigm for interactive image segmentation. This paradigm has been successfully used within many segmentation algorithms and applications [1, 9, 32, 34, 12]. In scribble-based segmentation, the user drags the mouse using one of two buttons to "paint" either foreground or background labels on pixels. In response, the system recomputes and displays the segmentation.

The motivation behind studying the scribble-based UIM is two-fold. First, it has shown promising results for a large number of interactive segmentation approaches, e.g., with graph-cuts [1] and random walker [9] algorithms. Second, the scribble-based paradigm allows a variety of valid segmentation inputs, which may have significant impacts on the user's behaviour depending on the shape, the position and the order in which the labels are drawn. In the context of scribble-based segmentation failure caused by a noisy/missing boundaries, e.g., often occurring in ultrasound images, the user's actions to correct the segmentation involve drawing additional foreground, background or both labels, whereas such difficulties would be addressed by manually tracing the boundary in a contour-based paradigm. In contrast to contour-based segmentation approaches that preserve the intuitive character of manual segmentation by focusing the user's attention on the object *boundary*, e.g., in live wire segmentation [24], the scribble-based paradigm is more permissive, focusing the user's attention on the object foreground and background *regions*. Therefore, when the user observes the updated results during the segmentation task, he/she may adapt the drawings differently, leading to very diverse scenarios. In other words, the conditions under which the user receives visual feedback strongly influence the scribble-based segmentation process.

There are multiple aspects that govern user behaviour during an interactive task. Focusing on the usefulness of the system, Nielsen [26, p. 26] proposed a model that identifies five attributes of usability in a user-centered system design: 1) is easy to learn, 2) is efficient to use, 3) is easy to remember, 4) produces low error rate and 5) is subjectively pleasing. Each attribute reflects a dimension of the user behaviour. However, when designing such systems, it is common to determine which attributes are key components to prioritize [15]. In this study, we are interested in segmentation efficiency. In particular, we report an experiment that manipulates the delay induced by computation time, i.e., the time elapsed between the drawings provided by the user and the response provided by the segmentation algorithm. This delay, referred to as the *feedback latency*, characterizes the computational efficiency of a segmentation method. Feedback latency is unavoidable in interactive applications and often has significant repercussions on the user's performance [22]. Previous work outside the context of image segmentation has studied the effect of latency in virtual reality [39], exploratory visual analysis [16] or touch-based interactive systems [4]. However, in an interactive image segmentation task, the effects of latency are unknown. For example, to what extent does the feedback latency decrease the user performance during a segmentation task? Is there a UIM design that reduces the effect of this latency on the user performance? What is the recommended latency, if it exists, below which the user interaction is no longer affected?

To address these questions, we experimentally investigated the effects of the visual feedback latency and timing on the user's performance by slightly modifying the condition under which the user receives this feedback during a scribble-based segmentation task. Specifically, two techniques for refreshing visual feedback under different levels of latencies, from 100 ms to 2 s are tested. The first technique is an *automatic* refresh method in which the user is continuously shown segmentation results as soon as they are made available, during label drawing. In the second technique, the user manually initiates the segmentation each time he/she wants to visualize the results. Segmentation time, accuracy and drawing measurements are recorded during the segmentation. Our contributions are summarized in the following findings:

- Most of the segmentation time is allocated to the user's cognitive activity, i.e., assessing the segmentation result, thinking about where to draw next labels, etc.;
- The user-initiated refresh method reduces the effects of the latency on the user's performance;
- The user's drawing activity is sensitive to latency when the automatic refresh method is used. The effect decreases non-linearly as latency increases;
- No effect of latency was observed on other user performance measures such as the number of pixels labelled, mouse clicks, undo actions, updates requested, or stroke continuity;
- Around $\sim$2 s latency, user performance seems to converge towards similar behaviours for both refresh methods.

II. Background

A. Latency in interactive applications

The 2 s response time threshold suggested in the literature for efficient user-computer communication [22] is subject to change according to the nature of the task. The human visual system is highly efficient for cognitive tasks. It has been reported that the visual system can distinguish *comprehensive* content in images displayed for 13 ms [31]. However, even if the user is able to understand the visual content, the complexity of the mechanism involved during human interaction delays his/her reaction by 100 ms to 200 ms [18, p. 117–118]. Moreover, according to Miller [22], human activities are naturally organized into groups of actions, named *closures*, that are determined by the achievement of subjective purposes. The user is more sensitive to a latency occurring within the same group of actions, hereafter referred to as the *within-activity* latency, than to a latency occurring between two groups of actions, referred to as *between-activity* latency.

In the context of scribble-based segmentation, we typically identify two groups of activities[1] [30]: (1) The *query* actions, in which the user provides the inputs. This is represented by dragging the mouse to draw labels. (2) The *feedback* actions, which consist in the cognitive activity, where the user receives and interprets the results. Depending on when the latency occurs in the segmentation process, it may affect the user differently. In this paper, we designed two refresh methods that represent the two types of latency. In the first scenario (exemplified by the automatic refresh), the segmentation updates are automatically displayed on the screen as soon as

[1] Note that this applies to interactive segmentation in general. However, for concreteness, we restrict our definition to scribble-based segmentation.

they are available, i.e., while the user is still drawing. The query and the feedback actions are confounded, leading to the occurrence of within-activity latencies during the segmentation task. This type of mechanism can be found in medical platforms, such as 3D Slicer [7], in which during the segmentation one can check an “Auto-Update” button to activate the automatic refresh of the segmentation results while drawing labels. In the second scenario (exemplified by the user-initiated refresh), the segmentation updates are controlled by the user, dissociating the query and the feedback into two distinct groups corresponding to different actions. Hence, the delay caused by the computations is considered to be a between-activity latency.

The representation of the within- and between-activity latencies is not limited to the proposed automatic and user-initiated refresh methods. The aim of our study is to assess how these types of latencies can influence the user performance during a scribble-based segmentation task. Different UIMs with different input/output devices can be designed to experiment these types of latency. The proposed refresh methods have the following advantages for our study: (i) they keep the UIM minimal to reduce the influence of user interface and input/output devices on the user performance (details in Section III-D), and (ii) the UIM is based on intuitive mouse drawing, similar to the one used by McGuinness and O’Connor [20].

B. *Interactive segmentation assessment*

The user plays a significant role in the interactive segmentation process. Yet, when it comes to the assessment of interactive segmentation methods, it is common in the literature to emphasize the computational aspects of the process without regards to the user’s performance. However, to design a segmentation assessment framework, it is important to consider the context of the segmentation task [38]. In fact, the nature of the task, the type of images and the application domain influence the segmentation outcome and the user behaviour. In most cases, the goal is to compare different segmentation methods. Conducting a user study is the most common way to account for the user’s performance in the assessment of interactive applications [17]. User studies have been used in interactive segmentation contexts to evaluate the performance of different UIMs [37, 35]. McGuinness and O’Connor [20] proposed a unified platform for interactive segmentation assessment. However, heterogeneous mechanisms often involve significant changes in user interface, requiring a specific evaluation platform. Focusing on the UIMs in the context of 3D image segmentation, Ramkumar et al. [33] conducted a user study to compare contour-based and scribble-based approaches. Objective and subjective metrics were recorded, allowing the comparisons in terms of computational performance and user appreciation, respectively.

Objective metrics, such as computation time and accuracy of the segmentation results, are reliable tools to assess the computational aspects of a segmentation method. In contrast, subjective metrics, often obtained through forms filled by the participants at the end of the experiment, are typically used to assess the cognitive aspects. However, some dimensions of the user’s cognitive and behavioural performances during the segmentation can be measured objectively. Hebbalaguppe et al. [14] attempted to measure the attention effort produced during a segmentation task by analyzing the user’s electroencephalogram signal, recorded during the task. In order to assess the performance of a haptic interface in 3D segmentation, Harders and Szekely [13] conducted experiments by adapting a model based on Fitts’ law [8], which describes a formal relationship between speed and accuracy in aimed movements.

While the aforementioned works focus on comparing the performance of segmentation methods in terms of user interaction, this paper aims at investigating user behaviour under controlled response times using a standard user interface. Based on the suggestions documented by Udupa et al. [38] and Olabarriaga and Smeulders [30], we designed a user study to characterize user behaviour during the completion of a scribble-based segmentation task.

III. Experiment

To evaluate the effect of latency on a scribble-based interactive segmentation task, we carried out a controlled user study. This section details the study design for the experiment.

A. *Preparing the image dataset*

A dataset of 80 images (250×250 pixels) was prepared. The images were carefully selected from *the cancer imaging archive* public database [3], to which we added samples from our own collection. The database includes samples from computed tomography (CT), magnetic resonance (MR), X-ray and ultrasound (US) images. Because the segmentation task is sensitive to the medical application, the samples covered a broad range of anatomical structures: brain imaging (MR, CT), carotid imaging (US), abdominal imaging, e.g., kidney, bladder, prostate (US, MR and CT) and chest and pelvic imaging (X-ray). All the ground truth data were obtained by manually segmenting the images. The dataset was then partitioned into 8 non-overlapping subsets $D_{i=1\ldots 8}$ of 10 images each, to avoid the carryover effect caused by segmenting the same image multiple times. The order in which the images within a dataset appear to the user is randomly shuffled. An additional dataset, $D_{\textit{training}}$, of 21 images was similarly prepared to serve as a training dataset, such that $D_{\textit{training}} \cap D_i = \emptyset, \forall i = 1 \ldots 8$.

B. *Study design*

Two factors were investigated. The *Latency* factor, which is the time elapsed between the drawing query and the segmentation response, i.e., the delay before displaying the update on the screen. Eight Latency conditions were tested, $\mathcal{L} = \{100, 200, 350, 500, 750, 1000, 1500, 2000\}$ (ms). The second factor is the *Refresh* method and was designed to capture between- and within-activity latency types. Two conditions were tested $\mathcal{R} = \{\textit{Automatic}, \textit{User-initiated}\}$. In the automatic refresh condition, the updates were automatically displayed on the screen after the latency time elapsed. Here, the latency condition acted as within-activity latency. In the user-initiated refresh condition, the updates were displayed on the participant's demand by pressing a button on the keyboard. Once initiated by the user, the results were displayed after the latency time elapsed. Here, the latency condition acted as between-activity latency.

A total of eight participants were recruited from engineering undergraduate and graduate programs. Some of the participants had prior experience with scribble-based segmentation approaches. Every participant tested all the latency conditions and all the refresh conditions. The order in which the latency conditions were tested was counterbalanced according to a 8×8 Latin square design [17, p. 177], i.e., one latency condition, $l_i \in \mathcal{L}, \forall i = 1 \ldots 8$, was tested on a single dataset $D_{j=1 \ldots 8}$, such that all combinations $\{l_i, D_j\}$ were tested by the eight participants. Therefore, each participant tested all the latency conditions with the automatic refresh method in a first experiment. Then, he/she tested all the latency conditions once again with the user-initiated refresh method in a second experiment. However, since the same image datasets were used for both experiments, such an ordered design could bias the participant's performances. To reduce the risks that a participant remembers the images and their associated labels, the second experiment was conducted at least two weeks after the user's participation in the first experiment.

Each experiment of a given refresh condition involved eight rounds of two successive steps: a training step followed by an evaluation step. The participants performed the training and the evaluation with the same latency condition on the first dataset, then the training and the evaluation with the next latency condition on the second dataset, and so on. In total, there were 8 participants $\times$ 8 latencies $\times$ 2 refresh methods $\times$ 10 images per dataset $=$ 1280 segmentation trials.

C. *Experiment progress*

1) Training step: During the training step, no data were recorded. The goal of the training was two-fold. First, it aimed at preparing the participant to understand how the scribble-based segmentation approach works. Second, it acted as a buffer between two successive conditions to accustom the participant for the next latency condition. During the training step, 10 images were randomly selected from the training dataset $D_{training}$. The participant had to segment all the 10 images under a given latency condition before proceeding to the evaluation step under the same condition. For each segmentation trial, the ground truth was provided and displayed beside the original image. In order to instruct the participant on the amount of accuracy required for the segmentation, the accuracy score (see Section IV) of the current segmentation result was displayed on the top right of the screen, during the training step. An acceptable score was considered to be 0.90 or above, before ending the trial and moving on to the next image.

2) Evaluation step: During the evaluation step, the time, the accuracy and efficiency of the segmentation were measured according to the parameters described in Section IV. Similarly to the training step, the ground truth was provided to the participant, indicating the anatomical structure to segment.

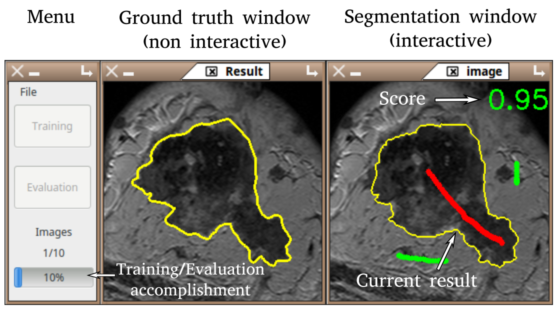


Fig. 2: The user interface in our study.

This information compensates for the lack of medical image interpretation skills of the participants. However, during the evaluation step, the segmentation accuracy score was not shown during the task. The participant was asked to perform the most accurate segmentation according to his/her appreciation with respect to the ground truth in the minimum possible time. The evaluation step ended when the 10 images of the dataset D_i were segmented.

D. *Interaction mechanism*

The user interface was designed based on a standard 2D window-icon-menu-pointer (WIMP) paradigm for minimal user-computer interaction (see Fig. 2). It involved two menu buttons to switch between training and evaluation steps, and two windows to display the image and its ground truth. The user's actions were restricted to: (i) clicking and dragging the mouse to draw foreground and background labels; and (ii) undoing the last drawings using a keyboard button. The number of undo actions was unlimited. For the user-initiated refresh condition, the user had to press a keyboard button, with his/her non-dominant hand, for every desired segmentation update. The automatic refresh condition did not require any additional interaction. Once the results were satisfactory, the user ended the segmentation using a dedicated keyboard button. The next image and its ground truth were then automatically loaded on the screen. Any additional user interaction, such as zooming/panning, loading images and resizing the thickness of the drawing brush, were prohibited, due to the unnecessary cognitive load they impose on the user.

During the segmentation, the user drew foreground and background labels and the result was updated on the screen. The segmentation computations were indicated using the mouse's waiting cursor. However, the participants were allowed to draw new labels while the computations were taking place. Hence, for both automatic and user-initiated refresh conditions, the drawing and the computations were separate processes. For example, for a 2000 ms latency, participants were able to anticipate the segmentation result by drawing additional labels before the computations were completed.

E. *Segmentation method and computations*

We considered a binary segmentation task of a single object per image, i.e., the user was allowed to draw only foreground and/or background labels. The minimum latency tested in

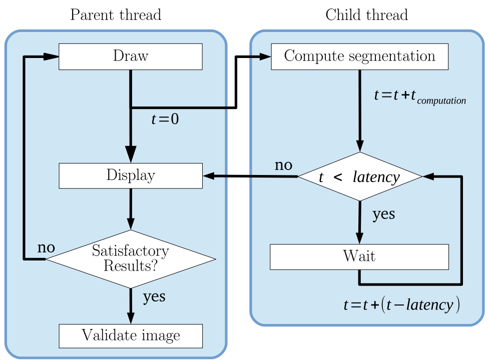


Fig. 3: Workflow of the interactive segmentation software used for experiments. The drawing and segmentation computation are processed in different threads, thereby allowing user interaction during computations. The elapsed time t is used to control the latency of the segmentation.

this study was 100 ms. Therefore, the segmentation method must satisfy this speed threshold, i.e., provide updates under 100 ms. To achieve this, we developed the FastDRaW[2] method [10], a fast adaptation of the random walker segmentation algorithm [9]. FastDRaW exploits a multi-scale framework combined with a dynamic region of interest (ROI) search to reduce the computation time. Prior to the segmentation, the image is rescaled using a nearest-neighbourhood down-sampling algorithm. The multi-scale framework consists in performing the computations on the reduced size of the image, then the results are *refined* on the original image. During the segmentation, the ROI is dynamically extracted based on the position of the drawn labels. Our assumption is that the object boundary is more likely to be located somewhere between the foreground and background labels. For each label category $class \in \{\text{Foreground}, \text{Background}\}$, a Euclidean distance map $Dist^{class}$ is computed and normalized such that $Dist^{class} \in [0,1]$. Then, a relevance map E is computed for every pixel p in the image

$$E(p) = 1 - \frac{1}{2}\left(Dist^{Foreground}(p) + Dist^{Background}(p)\right) \quad \in [0,1], \tag{1}$$

The pixel p belongs to the ROI if

$$E(p) \geq \bar{E} - k\sigma_E, \tag{2}$$

where $\bar{E}$ and σ_E are the mean and the standard deviation of the values in E, respectively, and $k \in \mathbb{R}$ is a constant parameter, such that increasing k reduces the number of pixels to be selected in the ROI. We empirically set $k = 1$. The segmentation then is performed only on pixels belonging to the ROI, reducing the computation time.

In this study, all processing was done on an Intel© Core i7-2630QM 2GHz×4 machine with 4Gb of RAM. To avoid holding the drawing resources during computations, all image processing operations were run in a separate thread (see Fig.3). For images of size 250×250, the FastDRaW method provides segmentation results in $\sim$ 90 ms which is sufficiently fast for the purpose of the study. Larger latencies were simulated by constraining the processing thread to wait the remaining amount of time before displaying the result on the screen.

[2]Available at https://github.com/hgueziri/FastDRaW-Segmentation

IV. Measures

This section describes the metrics used to capture the user's performance during the evaluation step of the experiment.

Overall time - t_Ω: The overall *time* elapsed during the segmentation of each image was measured. We use the average time required to perform one image segmentation over a dataset D_i under a given latency condition.

Labelling time - t_Λ: During the segmentation, the time taken to draw the labels was recorded. This measure involves the sum of elapsed times between the moment the user presses then releases the mouse buttons to draw a foreground or background label, for a single image. The labelling time is the average time recorded per image over each dataset D_i under a given latency.

Drawing speed - υ: The user labels the image by drawing scribbles using the mouse. A scribble is drawn by pressing and holding the mouse's button down while moving the mouse through the image. The speed at which the user moves the mouse between the moment the button is pressed and released indicates how fast the scribbles are drawn. This drawing speed was computed by dividing the distance travelled by the time elapsed between these two moments, and is given in $pixel/s$. The goal of this metric is to observe how the user draws the scribbles. In fact, we hypothesize that a large cognitive load slows down the user's drawings. The drawing speed is given by the average speed recorded while segmenting a dataset D_i.

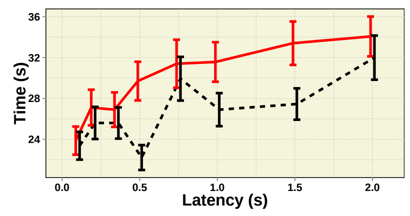
Accuracy - $\mathcal{A}$: The Dice index (DI) is commonly used to assess the accuracy of a binary segmentation result. The DI measures the agreement between two samples of binary data. To measure segmentation accuracy, interactive segmentation results were compared to their respective ground truth segmentations, such that

$$DI = \frac{2TP}{2TP + FP + FN} \quad \in [0,1], \tag{3}$$

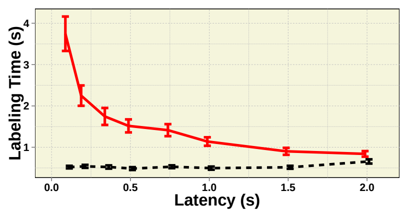
where TP, FP and FN are the true positive (object surface), false positive and false negative scores, respectively. The DI tends towards 1 as the segmentation result approaches the ground truth.

Continuity of the strokes - $\mathcal{C}$: The continuity is the number of labelled pixels per mouse click in one segmentation trial. Because the drawing is performed by maintaining the button of the mouse pressed, this measure expresses the average number of pixels labelled in one mouse stroke. Therefore, a large value indicates that the labels were drawn continuously, i.e., in the form of long strokes. A discontinuous drawing can be caused by two events. Either the user draws multiple strokes of the same label category, or the user alternates between drawing foreground and background labels. We hypothesize that a continuous drawing is performed during the same action, and may be associated to a single thinking process on the part of the user. The continuity measure characterizes the way the drawings were accomplished.

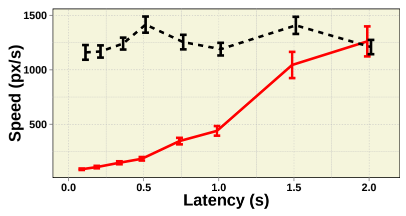
Number of labels - $\mathcal{N}$: When a segmentation trial was completed, the total number of labelled pixels was computed.



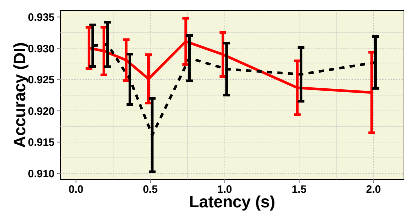
(a)



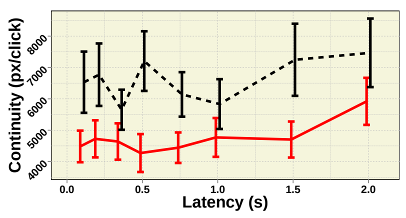
(b)



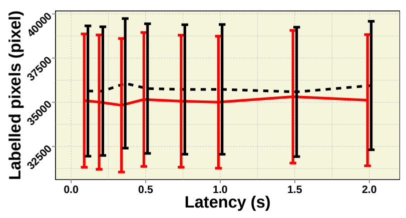
(c)



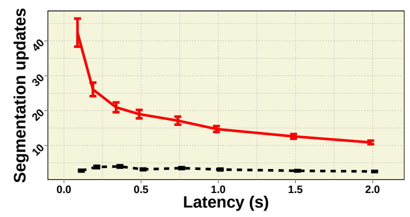
(d)



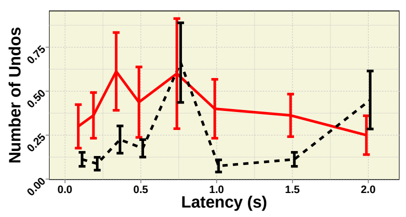
(e)



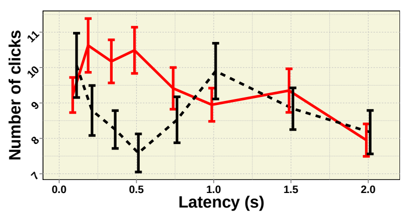
(f)



(g)



(h)



(i)

Fig. 4: Summary of the results obtained with the automatic (solid red line) and user-initiated (dashed black) refresh methods: (a) average time to complete one image segmentation trial, (b) average time to label an image (c) average drawing speed per image, (d) average Dice segmentation score per image (bigger is better), (e) average strokes continuity per image (bigger means longer strokes), (f) average number of labelled pixels per image, (g) average number of segmentation update per image, (h) average number of undo action performed per image and (i) average number of mouse click performed per image.

Complex segmentation tasks would require more labels than simple ones. The number of pixels labelled is an indicator of the amount of effort produced by the user during the segmentation task.

Number of segmentation updates - $\mathcal{R}$: The number of times the user receives segmentation feedback is recorded. Automatic refresh typically provides a larger number of updates, which induces a bias when compared to user-initiated refresh. This measure is used primarily with user-initiated refresh and indicates the presence/absence of penalties caused by the latency that can influence the user behaviour. In some applications, e.g., puzzle solving [29], it has been reported that additional latency can make the user change his/her strategy to achieve the task. We wanted to see whether this was the case for interactive image segmentation.

Number of undo actions - $\mathcal{U}$: The number of undo actions counts the average number of times the user performs an undo action, i.e., he/she deletes the last label drawn. Similar to the number of segmentation updates, $\mathcal{U}$ expresses the penalty induced by the *time recovering from errors* described by Mayhew [19]. In the case of large latencies, the user may prefer saving time over accuracy.

Number of mouse click - ***clk***: The average number of mouse click performed by the user during the segmentation of one image. This reflects one aspect of the user activity during the segmentation task.

V. Results

Fig.4 summarizes the results obtained by all the participants for both the automatic and user-initiated refresh experiments.

A. Overall time

The overall segmentation time results are shown in Fig. 4a. A repeated-measure analysis of variance (ANOVA) [2] reveals a statistically significant effect of the latency on the overall segmentation time ($F_{7,49} = 3.35, p < 0.01$) for both refresh methods. For the automatic refresh method, the overall segmentation time increases non-linearly with the latency. The impact of the latency on the segmentation time becomes less significant as the latency increases. In fact, for latencies between 100 ms and 500 ms, the segmentation performance slows down quickly, from an overall segmentation time of $t_{\Omega}^{100} = 23.86\,\text{s} \pm 1.37\,\text{s}$ to $t_{\Omega}^{500} = 29.69\,\text{s} \pm 1.89\,\text{s}$ per image, respectively, which represents a slope of 12.46 ± 0.48. This impact is attenuated for latencies between 750 ms and 2000 ms to reach a slope of 2.79 ± 0.19.

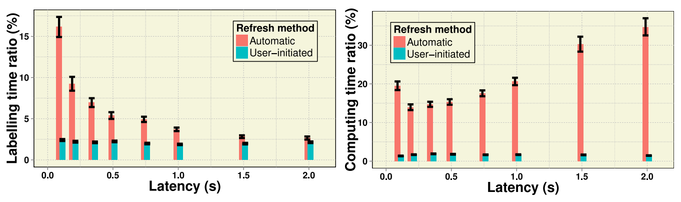


Fig. 5: Labelling time (left) and computation time (right) as fractions of the overall time, using the automatic refresh (red) and user-initiated refresh (blue) methods. Note that these two processes are performed in parallel as shown in Fig.3.

Compared to the automatic refresh method, the segmentation task using the user-initiated refresh method was completed slightly faster, but no statistically significant difference was found between both refresh methods ($F_{1,7} = 0.91, p = 0.37$). We draw the reader's attention to the shorter segmentation time obtained using the user-initiated refresh method at 500 ms latency. This score is similar to the one obtained with 100 ms latency for the same refresh condition. However, the corresponding accuracy score shows one of the worst performances with an average Dice index of $\mathcal{A} = 0.916 \pm 0.005$ (see Fig. 4d). This value may represent an outlier.

B. Labelling time and drawing speed

The average time required to label an image is shown in Fig. 4b. The user's drawing performances are clearly affected by latency when the automatic refresh method is used ($F_{7,49} = 7.13, p < 0.01$). For a latency of 100 ms, $t_{\Lambda}^{100} = 3.74\,\text{s} \pm 0.41\,\text{s}$, representing $16.19\% \pm 1.22\%$ of the overall segmentation time. Then, a rapid decrease in the labelling time occurs between 100 ms and 500 ms latency, before reaching a stable value around $t_{\Lambda} = 1.16\,\text{s} \pm 0.11\,\text{s}$ in the 500 ms to 2000 ms latency interval. Here, the labelling time represents $3.89\% \pm 0.25\%$ of the overall segmentation time (Fig.5 (left)). Using the user-initiated refresh method, the latency condition seems to have little effect on the labelling time ($F_{7,49} = 2.02, p = 0.07$). The average performance for all the participants was $0.53\,\text{s} \pm 0.03\,\text{s}$, i.e., $2.18\% \pm 0.10\%$ of the overall segmentation time. In comparison, the fraction of the computation time relative to the overall time is shown in Fig.5 (right). While the fraction of the computation times seems to be stable for all latency conditions using the user-initiated refresh method, it is larger and varies significantly with the latency when using the automatic refresh method. This is expected, due to the large number of updates induced by the automatic refresh. Because the segmentation is performed in a separate thread (see Fig. 3), the overall segmentation performance is not represented by the exact sum of the labelling performance and computation performance. However, in terms of user performance, the extreme scenario would be to consider both the labelling and the computation processes to be performed serially. In this case, they can be summed up to represent an indicator for the user performance during the segmentation task (as used in Section VI-B).

The average drawing speed is shown in Fig.4c. The results are consistent with the labelling time. Using the automatic refresh method, the latency has significant effect on the drawing speed ($F_{7,49} = 8.62, p < 0.001$). The drawing speed increases with latency to reach its highest value of $\upsilon = 1261.15\,\text{pixel/s} \pm 137.73\,\text{pixel/s}$ at 2 s latency. Using the user-initiated refresh method, the drawing speed is not significantly affected by latency ($F_{7,49} = 1.14, p = 0.35$), with an average speed of $\upsilon = 1255.80\,\text{pixel/s} \pm 101.57\,\text{pixel/s}$. This value is most likely the upper speed limit achievable while maintaining reasonably careful drawings in the tested scribble-based segmentation approach.

C. Segmentation accuracy

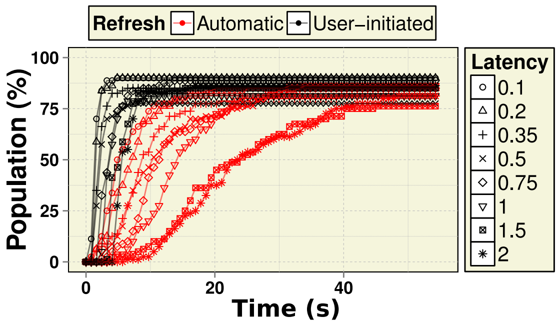
The average accuracy obtained per image is plotted in Fig. 4d. In this study, the participants were allowed to adjust the drawings until a satisfactory segmentation result was obtained, without a time limit. Having been shown the Dice score during the training step, the participants were visually habituated to match the segmentation result and the ground truth with a sufficient similarity ($DI \geq 0.90$). Therefore, it is not surprising to observe high Dice indices (0.927 ± 0.036 for automatic refresh and 0.926 ± 0.037 for the user-initiated refresh methods) with small variations between participants. The ANOVA test revealed no statistically significant difference between the accuracy obtained using both refresh methods ($F_{1,7} = 0.65, p = 0.44$). To gain more insight into the accuracy performance, we recorded the evolution of the Dice index during the segmentation task for all latency conditions. Fig. 6a shows the cumulative fraction of all the trials that reached a Dice index of 0.90 or above over time. Using the user-initiated refresh method, users rapidly achieved satisfactory segmentation results. In contrast, using the automatic refresh method, satisfactory results depend on the latency, as they take longer to be achieved under long latencies. Fig.6b shows similar cumulative fraction of Dice index according to the number of segmentation updates provided to the user. Under short latencies, the automatic refresh provides more updates to the user. Therefore, the evolution of the cumulative fraction is slower. However, for user-initiated refresh, the effect of the latency is not observable.

D. Continuity of the strokes

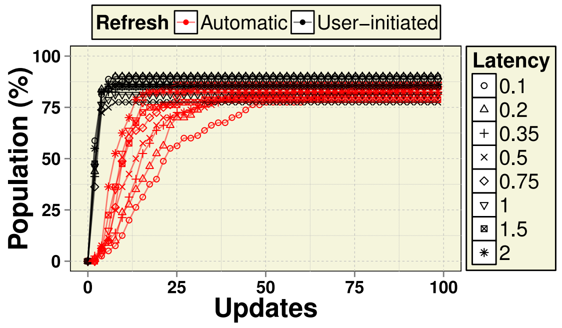
The stroke continuity results are shown in Fig.4e. Recall that the continuity measure indicates the average length of strokes per click and gives insight into how the labels were drawn. Using the user-initiated refresh method, the strokes produced by the participants appear to be longer, i.e., with larger values of $\mathcal{C}$. However, the ANOVA test reveals no statistically significant difference between the refresh methods ($F_{1,7} = 1.48, p = 0.26$). Moreover, no statistically significant effect of the *latency* was found in the stroke continuity ($F_{7,49} = 0.62, p = 0.73$).

E. Number of labels

The average number of labelled pixels required to segment all the images for each latency condition is shown in



(a)



(b)

Fig. 6: The cumulative fraction of trials having a Dice score of 0.9 or above as a function of time (a) and number of updates (b), using the automatic refresh (red) and user-initiated refresh (black) methods.

Fig. 4f. The latency has no statistically significant effect on the number of pixels labelled during the segmentation task ($F_{7,49} < 0.001, p \simeq 1$). However, the automatic refresh method required significantly fewer labels than the user-initiated method ($F_{1,7} = 7.17, p = 0.03$).

F. Number of segmentation updates

The average number of segmentation updates $\mathcal{R}$ for each image is shown in Fig. 4g. It is expected that $\mathcal{R}$ is large for short latencies when using the automatic refresh, with a maximum value of $\mathcal{R}^{100} = 42.37 \pm 4.02$ updates and a minimum value of $\mathcal{R}^{2000} = 10.86 \pm 0.51$ updates. Therefore, the ANOVA test is performed for the user-initiated refresh method analysis only, revealing no significant effect of the latency on the number of updates performed by the participants during the segmentation task ($F_{7,49} = 1.90, p = 0.089$). In other words, slow computation time did not yield statistically significant changes in user behaviour regarding the number of times he/she requested a segmentation update.

G. Number of undo actions

Fig.4h shows the average number of undo actions per image. Few undo actions were performed during the segmentation task, regardless of the refresh method used and the latency. The ANOVA test reveals no statistically significant effect of the latency condition on either of the refresh methods ($F_{7,49} = 1.18, p = 0.33$). Moreover, no statistically significant difference in the number of undo actions was found between the refresh methods ($F_{1,7} = 1.96, p = 0.20$).

H. Number of mouse clicks

Fig. 4i shows the average number of clicks performed by the user per image. No statistically significant difference was found between the automatic and the user-initiated refresh methods ($F_{1,7} = 0.16, p = 0.69$), nor between the latency conditions ($F_{7,49} = 0.68, p = 0.68$) regarding the number of clicks performed during the segmentation task.

VI. Discussion

A qualitative summary of the results is shown in Table I

TABLE I: Qualitative summary of the experimental results.

Refresh method		Observations
Automatic	t_Ω:	non-linear increase with latency (rapidly then slowly)
	t_Λ:	non-linear decrease with latency (rapidly then slowly)
	υ:	linear increase with latency
	$\mathcal{A}$:	slow convergence to satisfactory results
	$\mathcal{C}$:	no observed effect
	$\mathcal{N}$:	fewer labels using automatic refresh
	$\mathcal{R}$:	–
	$\mathcal{U}$:	no observed effect
	clk:	no observed effect
User-initiated	t_Ω:	increase with latency (no observed pattern)
	t_Λ:	no observed effect
	υ:	no observed effect
	$\mathcal{A}$:	fast convergence to satisfactory results
	$\mathcal{C}$:	no observed effect
	$\mathcal{N}$:	more labels using user-initiated refresh
	$\mathcal{R}$:	no observed effect
	$\mathcal{U}$:	no observed effect
	clk:	no observed effect

A. Automatic vs. user-initiated refresh method

Under the automatic refresh condition, the user is made subject to a within-activity latency. Any delay occurring during computations would be associated with the drawing actions, confounding between the cognitive task of drawing and interpreting results. On the other hand, a user-initiated refresh of the results allows the user to explicitly separate the drawing task from the interpretation of the results. The user is then subject to a between-activity latency. In our experimental results, the user's performance under the user-initiated refresh condition was less sensitive to latency than under the automatic refresh condition. In fact, using the user-initiated method, no significant effect was observed regarding the way the user draws labels (i.e., labelling time, drawing speed, strokes continuity and number of labels) or the way he/she interacts with the algorithm (i.e., number of updates, number of undo actions and number of clicks). Moreover, using the automatic refresh method, users seemed to tend towards similar behaviour when the latency was large (~ 2 s).

B. Cognition and drawing efficiency

Providing labels during the segmentation task involves an underlying effort of understanding the image content and the current results, i.e., in our case, evaluating the current segmentation contour and deciding where to draw the next labels. This effort is referred to as the *thinking task* and reflects *user scan/read* time, *user think* time and *user response* time as described by Mayhew [19, p. 508]. This task is different from *drawing* the actual labels, which involves locating and tracking the position of the cursor and manipulating the mouse. In fact, our experiment revealed that most of the segmentation time is allocated to the thinking task (see Fig. 5). In the extreme case where the drawing task and the computation task are taken to be performed serially, the average thinking time for automatic refresh was $72.67\,\% \pm 2.42\,\%$ and for user-initiated $96.23\,\% \pm 0.06\,\%$ of the overall segmentation time. Note that these results include idle time.

In what follows, we refer to *the cognitive process* as mental preparation of the user to perform a labelling task, i.e., involving the combination of both the drawing and the thinking tasks. Using the definition given by Newell [25, p. 261-268] in the context of immediate response tasks, the cognitive process requires longer processing time as the task increases in complexity. In our experiment, this can be observed when using the automatic refresh method, where the thinking task is confounded with the drawing task. In this scenario, the effect is particularly noticeable with short latencies. In fact, the participants required more cognitive processing time to label the image, slowing down their labelling performance. The effect is attenuated as the latency increases, making the refresh less frequent. Here, the gap between the drawing task (achieved in constant real-time during the experiment) and the thinking task (occurring less frequently as the latency increases) is larger when the latency is large ($\sim 2000\,\text{ms}$).

This gap reaches its upper limit when the drawing and the thinking tasks are considered as totally separate tasks, as illustrated in the user-initiated refresh method. The cognitive process is exclusively restricted to the drawing task or thinking task, therefore reducing complexity to performing a single task at a time. While the experimental results do not allow a direct observation of the time allocated to the thinking task, participants provided a much more *efficient* labelling performance in this scenario.

C. The "no effect" results

Because the latency condition was the only variable parameter experimented with respect to the refresh method, it is important to report the observations where the latency had little or no effect on the user performance. To this end, in our study, we use the number of segmentation updates, the number of undo actions and the number of clicks performed during the segmentation task as surrogates to measure the user activity. Interestingly, the latency seems to have little effect on the user activity in terms of these three metrics when using the user-initiated refresh. In particular, the number of segmentation updates did not yield a significant change in user behaviour despite the additional cost of computation time induced by the latency (dashed line in Fig.4.g). This may be due to the fact that the maximum latency experimented is still an *acceptable* response time for interactive communication [22]. For the automatic refresh method, the number of segmentation updates is not an appropriate measure to observe the user behaviour. However, the number of undos and clicks revealed similar user behaviours, regardless of the latency.

Moreover, the study revealed that given sufficient time, latency does not affect segmentation accuracy, regardless of the refresh method. All participants were able to achieve satisfactory segmentation results. However, results were achieved faster with user-initiated refresh.

D. To what extent does the study apply to different UIMs?

It is most likely that important changes in the UIM of the segmentation approach would induce substantial changes in the user's behaviour. In this section we discuss the following questions: (i) what is the validity of our analysis for interactive segmentation approaches with different UIMs, e.g., in the case of a live-wire [24, 5] or a hybrid [36, 40, 11] UIM? and (ii) will the user's performance be similar given the same scribble-based UIM combined with a different segmentation algorithms, e.g. graph-cut [1] or random walker [9]?

Regarding question (i), our analysis is based on the query-feedback loop model [30]. For real-time segmentation approaches, the speed at which the loop cycles is very high, giving the impression of processing a single task. For example, this is the case for the live-wire segmentation approach [5], where the contour updates need to be provided in real-time, while the user is hovering the mouse. In this case, the latency range $[100\,\text{ms}, 2\,\text{s}]$ tested in our experiment would be inappropriate, necessitating tighter response times, e.g., $[10\,\text{ms}, 200\,\text{ms}]$, to evaluate a specific user behaviour for this approach. Nevertheless, because of the query-feedback loop, the between- and within-activity latency categorization is still valid, i.e., different UIMs can be designed for live-wire segmentation based on these types of latency. The choice of the UIM depends on the properties and the user performance that one needs to leverage during the segmentation task. Moreover, a interesting property of live-wire-based UIMs is that segmentation is performed locally, i.e., the approach provides a series of *final* segmentation results on *small sections* of the object boundary. In contrast, scribble-based UIMs provide a series of *intermediate* segmentation result for the *whole* object boundary. This conceptual difference may be worth studying to explore how the user performs under local/global interactive segmentation mechanisms. Specifically, it would be interesting to understand the organization of the cognitive task in this context, e.g., is the user's idle time uniformly distributed along his/her performance on each contour section? does it require less time to interpret the segmentation result on a smaller section of the object boundary or is it easier if he/she observes the whole object shape at each interaction?

Regarding question (ii), given the exact same inputs and UIMs for image segmentation, different computation algorithms may produce significantly different results. In this case, the user would react differently according to the received feedback, yielding variable segmentation performances. However, McGuinness and O'Connor [21] showed that it is possible to simulate the user behaviour, in terms of foreground/background labelling, as demonstrated in their experiment comparing four scribble-based segmentation methods. The authors proposed four *strategies* to automatically produce user scribbles, in a *realistic* way, in which the scribbles were generated by comparing the segmentation update obtained in each iteration to the ground truth segmentation result. Interestingly, although the segmentation algorithms tested produced different results, the strategies were able to simulate a coherent user drawing by following constant rules. This was possible because the strategies assumed: (i) a cooperative user behaviour, (ii) knowing what is the goal to achieve, i.e., being able to add meaningful labels at each iteration, and (iii) the consistency of the segmentation response regarding the input data, i.e., the algorithm produces the same results given the same inputs. In our study and in practice, the first assumption is met based on the user's willingness to achieve a satisfactory segmentation. The second assumption is met by showing the ground truth to the participants during the segmentation task, and is met in practice by the user's expertise (e.g. anatomy) related to the task. The third assumption is met through the probabilistic consistency of the random walker segmentation approach [9, 10]. Therefore, we expect the overall user behaviour to be comparable regardless of the intermediate results.

VII. Conclusions

Interactive scribble-based image segmentation is highly permissive in terms of user performance. Similar segmentation results can be achieved with different label drawings and efforts produced by the user during the segmentation task. Therefore, the condition under which the user receives the segmentation feedback affects his/her performance. In this study, we investigated the user's performance under two conditions of latency variation: (i) the latency level: ranging from 100 ms to 2 s, and (ii) the type of latency: within-activity and between-activity latencies represented by an automatic refresh and a user-initiated refresh of the segmentation updates.

Increasing the latency has negative impacts on the overall performance. However, this affects the user performance differently depending on the latency and the refresh method. For the automatic refresh method, we observed a non-linear variation of the user's behaviour, with a slower drawing performance for short latencies. The user is paying attention to the segmentation updates as they are refreshed frequently. For longer latencies, the overall segmentation time seems to increase slowly with the latency. The user is less attentive to the updates, resulting in faster drawing performance.

For the user-initiated refresh method, increased latency has less significant impacts on the user's behaviour. In this case, the drawing and the interpretation of the result updates are dissociated into two separate processes. The attention of the user is focused on a single task at a time, which improves his/her performance. In terms of the labelling time, the user-initiated refresh method showed better performances. The automatic and user-initiated refresh methods tested in our study exemplify the within- and between-activity types of latency, respectively. The results obtained are consistent with the theoretical characterization of latency discussed by Miller [22]. We observe a convergence of the user behaviour as the latency approaches 2 s, which is the suggested threshold for an effective response time in interactive applications.

Future work will include investigating the effects of the latency on UIMs based on the object contouring paradigm, e.g., live-wire [5], as they are substantially different from the scribbling paradigm. We believe these approaches require a shorter response time than scribble-based approaches to be effective. Therefore, more attention has to be allocated to the range of the feedback latency for these UIMs. Another interesting study would also involve investigating a different dimension of usability, e.g., how easy it is to learn the UIM and how an experienced user would interact with the UIM? This can be achieved by comparing ways to make interactive segmentation techniques self-revealing, e.g., by comparing the effectiveness of written instructions, tooltips that pop-up in context, step-by-step interactive help, or automatic detection of times when the user may need help.

References

[1] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images," in *IEEE International Conference on Computer Vision*, vol. 1, 2001, pp. 105–12.

[2] J. M. Chambers and T. J. Hastie, *Statistical models in S*. Wadsworth & Brooks, 1992.

[3] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle, L. Tarbox, and F. Prior, "The cancer imaging archive (TCIA): maintaining and operating a public information repository," *Journal of digital imaging*, vol. 26, no. 6, pp. 1045–57, 2013.

[4] J. Deber, R. Jota, C. Forlines, and D. Wigdor, "How much faster is fast enough?: User perception of latency & latency improvements in direct and indirect touch," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. ACM, 2015, pp. 1827–1836.

[5] A. X. Falcao, J. K. Udupa, and F. K. Miyazawa, "An ultra-fast user-steered image segmentation paradigm: live wire on the fly," *IEEE Transactions on Medical Imaging*, vol. 19, no. 1, pp. 55–62, 2000.

[6] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. de A. Lotufo, "User-steered image segmentation paradigms: Live wire and live lane," *Graphical Models and Image Processing*, vol. 60, no. 4, pp. 233–60, 1998.

[7] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J.-C. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, J. Buatti, S. Aylward, J. V. Miller, S. Pieper, and R. Kikinis, "3D Slicer as an image computing platform for the Quantitative Imaging Network," *Magnetic Resonance Imaging*, vol. 30, no. 9, pp. 1323 – 1341, 2012.

[8] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," pp. 381–91, 1954.

[9] L. Grady, "Random walks for image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–83, 2006.

[10] H.-E. Gueziri, L. Lakhdar, M. McGuffin, and C. Laporte, "FastDRaW – fast delineation by random walker: application to large images," in *MICCAI workshop on Interactive Medical Image Computing*, 2016.

[11] H.-E. Gueziri, M. J. McGuffin, and C. Laporte, "A generalized graph reduction framework for interactive segmentation of large images," *Computer Vision and Image Understanding*, vol. 150, pp. 44–57, 2016.

[12] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman, "Geodesic star convexity for interactive image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3129–36.

[13] M. Harders and G. Szekely, "Enhancing human-computer interaction in medical segmentation," *Proceedings of the IEEE*, vol. 91, no. 9, pp. 1430–42, 2003.

[14] R. Hebbalaguppe, K. McGuinness, J. Kuklyte, G. Healy, N. O'Connor, and A. Smeaton, "How interaction methods affect image segmentation: User experience in the task," in *1st IEEE Workshop on User-Centered Computer Vision*, Jan 2013, pp. 19–24.

[15] L. Leventhal and J. Barnes, *Usability Engineering: Process, Products and Examples*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007.

[16] Z. Liu and J. Heer, "The effects of interactive latency on exploratory visual analysis," *IEEE Trans. on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2122–31, 2014.

[17] S. I. MacKenzie, *Human-Computer Interaction, An Empirical Research Perspective*. Burlington, MA, USA: Elsevier Morgan Kaufmann, 2013.

[18] ——, "Fitts' law as a research and design tool in human-computer interaction," *Human-Computer Interaction*, vol. 7, pp. 91–139, 1992.

[19] D. J. Mayhew, *Principles and Guidelines in Software User Interface Design*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[20] K. McGuinness and N. E. O'Connor, "A comparative evaluation of interactive segmentation algorithms," *Pattern Recognition*, vol. 43, pp. 434–44, 2010.

[21] ——, "Toward automated evaluation of interactive segmentation," *Comput. Vis. Image Underst.*, vol. 115, no. 6, pp. 868–884, 2011.

[22] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, 1968, pp. 267–77.

[23] J. H. Moltz, S. Braunewell, J. Rühaak, F. Heckel, S. Barbieri, L. Tautz, H. K. Hahn, and H. O. Peitgen, "Analysis of variability in manual liver tumor delineation in ct scans," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2011, pp. 1974–77.

[24] E. N. Mortensen and W. A. Barrett, "Interactive segmentation with intelligent scissors," *Graphical Models and Image Processing*, vol. 60, no. 5, pp. 349–84, 1998.

[25] A. Newell, *Unified theories of cognition*. Harvard University Press, 1994.

[26] J. Nielsen, *Usability engineering*. Elsevier, 1994.

[27] G. Noris, D. Sýkora, A. Shamir, S. Coros, B. Whited, M. Simmons, A. Hornung, M. Gross, and R. Sumner, "Smart scribbles for sketch segmentation," *Computer Graphics Forum*, vol. 31, no. 8, pp. 2516–27, 2012.

[28] D. A. Norman, "Cognitive engineering," *User centered system design*, vol. 31, p. 61, 1986.

[29] K. P. O'Hara and S. J. Payne, "The effects of operator implementation cost on planfulness of problem solving and learning," *Cognitive Psychology*, vol. 35, no. 1, pp. 34–70, 1998.

[30] S. Olabarriaga and A. Smeulders, "Interaction in the segmentation of medical images: A survey," *Medical Image Analysis*, vol. 5, no. 2, pp. 127–42, 2001.

[31] M. C. Potter, B. Wyble, C. E. Hagmann, and E. S. McCourt, "Detecting meaning in rsvp at 13 ms per picture," *Attention, Perception, & Psychophysics*, vol. 76, no. 2, pp. 270–79, 2014.

[32] A. Protiere and G. Sapiro, "Interactive image segmentation via adaptive weighted distances," *IEEE Trans. on Image Processing*, vol. 16, no. 4, pp. 1046–57, 2007.

[33] A. Ramkumar, J. Dolz, H. A. Kirisli, S. Adebahr, T. Schimek-Jasch, U. Nestle, L. Massoptier, E. Varga, P. J. Stappers, W. J. Niessen, and Y. Song, "User interaction in semi-automatic segmentation of organs at risk: a case study in radiotherapy," *Journal of Digital Imaging*, vol. 29, no. 2, pp. 264–77, 2016.

[34] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. on Graphics*, vol. 23, no. 3, pp. 309–14, 2004.

[35] M. Sadeghi, G. Tien, G. Hamarneh, and M. S. Atkins, "Hands-free interactive image segmentation using eyegaze," in *Proc. SPIE Medical Imaging: Computer-Aided Diagnosis*, vol. 72601H, 2009.

[36] T. V. Spina, P. A. V. de Miranda, and A. X. Falcão, "Hybrid approaches for interactive image segmentation using the live markers paradigm," *IEEE Trans. on Image Processing*, vol. 23, no. 12, pp. 5756–69, 2014.

[37] A. Top, G. Hamarneh, and R. Abugharbieh, "Active learning for interactive 3d image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2011, pp. 603–10.

[38] J. K. Udupa, V. R. LeBlanc, Y. Zhuge, C. Imielinska, H. Schmidt, L. M. Currie, B. E. Hirsch, and J. Woodburn, "A framework for evaluating image segmentation algorithms," *Computerized Medical Imaging and Graphics*, vol. 30, no. 2, pp. 75–87, 2006.

[39] C. Ware and R. Balakrishnan, "Target acquisition in fish tank VR: The effects of lag and frame rate," in *Proceedings of Graphics Interface*. Canadian Human-Computer Communications Socity, 1994, pp. 1–7.

[40] W. Yang, J. Cai, J. Zheng, and J. Luo, "User-friendly interactive image segmentation through unified combinatorial user inputs," *IEEE Trans. on Image Processing*, vol. 19, no. 9, pp. 2470–79, 2010.