A Generalized Graph Reduction Framework for Interactive Segmentation of Large Images

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

École de technologie supérieure, 1100 Notre-Dame Ouest, Montreal (Québec), H3C 1K3 Canada

Abstract

The speed of graph-based segmentation approaches, such as random walker (RW) and graph cut (GC), depends strongly on image size. For high-resolution images, the time required to compute a segmentation based on user input renders interaction tedious. We propose a novel method, using an approximate contour sketched by the user, to reduce the graph before passing it on to a segmentation algorithm such as RW or GC. This enables a significantly faster feedback loop. The user first draws a rough contour of the object to segment. Then, the pixels of the image are partitioned into "layers" (corresponding to different scales) based on their distance from the contour. The thickness of these layers increases with distance to the contour according to a Fibonacci sequence. An initial segmentation result is rapidly obtained after automatically generating foreground and background labels according to a specifically selected layer; all vertices beyond this layer are eliminated, restricting the segmentation to regions near the drawn contour. Further foreground / background labels can then be added by the user to refine the segmentation. All iterations of the graph-based segmentation benefit from a reduced input graph, while maintaining full resolution near the object boundary. A user study with 16 participants was carried out for RW segmentation of a multi-modal dataset of 22 medical images, using either a standard mouse or a stylus pen to draw the contour. Results reveal that our approach significantly reduces the overall segmentation time compared with the status quo approach ($p < 0.01$). The study also shows that our approach works well with both input devices. Compared to super-pixel graph reduction, our approach provides full resolution accuracy at similar speed on a high-resolution benchmark image with both RW and GC segmentation methods. However, graph reduction based on super-pixels does not allow interactive correction of clustering errors. Finally, our approach can be combined with super-pixel clustering methods for further graph reduction, resulting in even faster segmentation.

Keywords:
Interactive segmentation, user study, graph-based segmentation, graph reduction, random walker, graph cuts

1. Introduction

Image segmentation consists of delineating specific (foreground) objects from the background of a given image. This task plays a crucial role in biomedical image analysis. Emerging applications, for example tumour measurement, 3D organ reconstruction or cell counting, typically require a segmentation step. This task can be achieved with varying degrees of user involvement, on a continuum from fully-manual to fully-automated. Manual approaches are time consuming and lack repeatability, whereas fully-automated approaches are not applicable in complex scenarios. A compromise between these extremes is interactive segmentation, where the user supervises and adjusts inputs in response to intermediate segmentation results. Because the user can modify inputs as long as the results are unsatisfactory, *human factors* (i.e., usability) are an important consideration.

Recently, graph-based approaches have gained popularity for interactive segmentation [2, 11, 30, 19, 26, 23, 29]. The idea is to represent the image as a graph, where vertices correspond to pixel locations and edges represent pixel adjacency. Edges are weighted as a function of their likelihood of crossing an object boundary. For interactive segmentation to be practical, the computation of the edge weights and segmentation must be fast, enabling a tight feedback loop. However, in graph-based segmentation, computation time increases with graph size, often precluding interactive segmentation of high-resolution images. The total time to perform a segmentation also depends on human factors, such as the input device used and the kind of input required. These challenges are addressed in this paper. We present a graph reduction approach that is guided by a rough drawing of the object boundary provided by the user. Our preliminary work [14] used automatically simulated input drawings to show that this approach speeds up computations for random walker segmentation. This paper further investigates the approach and extends our analysis to make the following additional contributions:

- A controlled experiment compared performance with two input techniques and two input devices (mouse and stylus pen).
- The graph reduction approach is extended to different interactive graph-based segmentations ensuring a precise, high-resolution segmentation.

*Corresponding author
E-mail: houssem-eddine.gueziri.1@ens.etsmtl.ca (H.-E. Gueziri)

Manuscript accepted in *Computer Vision and Image Understanding* *May 17, 2016*
DOI : 10.1016/j.cviu.2016.05.009

- We evaluate our approach alongside, and in combination with, graph reduction methods based on single and multi-resolution super-pixels [1] to benefit from further speed-ups.

Section 2 reviews work related to interactive graph-based segmentation. Section 3 describes our user-guided graph reduction approach, and Section 4 discusses some of its key properties. Section 5 presents the user study, and Section 6 presents benchmarks obtained by generalizing our approach to other graph-based segmentation methods and exploiting super-pixel-based reductions. Finally, Section 7 discusses the benefits and limitations of our approach, and future directions.

2. Related work

2.1. Interactive graph-based segmentation

To preserve the intuitive character of manual segmentation, Mortensen and Barrett [26] proposed *Intelligent Scissors* (IS), which define a cost function measuring the likelihood of graph edges crossing an object boundary. While the user draws a contour near the object boundary, this contour is adjusted on the fly using Dijkstra's algorithm so as to follow a minimum-cost path in the graph. Extensions of IS, including work by Mishra et al. [24] and the Magnetic Lasso offered in Adobe's commercial Photoshop software, were proposed to enhance segmentation flexibility. IS and its variants have two drawbacks: since the minimum-cost path must be computed in real-time during user interaction, the approach suffers from interaction feedback lags when applied to large images. Moreover, IS requires relatively high accuracy from the user when drawing the contour, which makes segmentation laborious [19].

In contrast to the *contour-based interaction* required by IS, *region-based interaction* involves drawing scribbles (labels) on a small set of pixels, in the foreground and/or background regions of the image. Boykov and Jolly [2]'s *graph cut* (GC) segmentation is a popular approach where pixels (vertices) are typically labeled in this manner. GC segmentation uses these foreground / background labels to remove edges to maximize flow [3], breaking the graph into two sub-graphs (foreground and background).

Variants of GC segmentation have reduced the required user interaction [15]. In *GrabCuts* [30], for example, the user first frames the object inside a bounding box, to reduce the search space. A Gaussian mixture model (GMM) is fitted to the cropped image intensities and labels are automatically generated according to the modes of the GMM. An initial segmentation result is then obtained using GC. The user can then add explicit foreground and background labels to adjust the segmentation. GrabCuts fails in the presence of weak boundaries, mostly because of the limited ability of the GMM to capture the true object intensity distribution. Our approach is similar to GrabCuts in that we exploit a user-drawn boundary to reduce the search space. However, our approach relies solely on the graph-based segmentation algorithm; no additional statistical model is required. Moreover, instead of confining the search space to the inside of a bounding box, the rough contour drawing is used to reduce the search space to pixels *near the object boundary*. This has three effects: (i) whereas the bounding box is constrained by object shape (e.g., a large bounding box is needed to surround a thin diagonally-oriented object), our approach is more flexible and optimizes graph reduction for complex shaped objects, (ii) our approach ignores pixels that are sufficiently *far inside* the drawn contour, resulting in a speed-up, and (iii) our approach allows more flexibility in the drawing, i.e., the drawn contour may lie slightly inside and/or outside the object to segment.

In the presence of weak boundaries, GC leads to the "small cuts" miss-segmentation problem. To address this, Grady [11] proposed random walker (RW) segmentation, wherein unlabeled pixels are assigned *probabilities* of belonging to each label category (foreground or background). Segmentation consists of selecting the most probable label for each pixel. In the absence of edges in the image, an unlabeled pixel is assigned equal probability of belonging to equidistant labels, thereby overcoming the small cuts problem.

Like GC, the RW method segments at interactive speeds for reasonably sized images [19], but is not fast enough for high resolution images. Grady and Sinop [13] proposed to pre-compute the eigen-decomposition of the image graph's Laplacian matrix off-line to accelerate the computation of RW probabilities. However, this solution is specific to RW segmentation, while our approach adapts to other graph-based segmentation methods. Moreover, the pre-computation itself is time and memory consuming and unfeasible for live applications.

GPU parallelization has also been used to accelerate RW [12] and GC [8], but is still constrained by hardware limitations because of the storage required for large datasets. Our approach is compatible with existing GPU approaches and reduces the amount of required GPU storage.

Yang et al. [34] proposed a *constrained RW* segmentation framework where the user may draw foreground and background labels, as well as *hard* constraint labels to enforce a particular boundary alignment, and *soft* constraint labels to indicate a region where the boundary is expected to pass. This combines *contour-based input* with the traditional *region-based input* of RW segmentation. We leverage a similar combination of user input methods, wherein the user draws a rough contour of the object and then provides additional foreground / background labels to refine the segmentation results. Unlike our approach, Yang et al. [34]'s approach does not address the issue of computation time. Moreover, their approach relies on additional energy functionals to force the boundary to pass through certain labels, corrupting the probabilities computed by RW segmentation, whereas our approach is entirely consistent with the probabilities generated with RW. Because our approach is solely based on user input to reduce graph size, it generalizes easily to other segmentation approaches that rely on foreground / background labeling, such as GC.

2.2. Interactive segmentation evaluation

Because of human factors, it is difficult to quantitatively assess interactive segmentation, especially when comparing methods involving different types of input. Olabarriaga and

Smeulders [28] enumerated three major criteria that an interactive segmentation should satisfy: (i) *Accuracy* is the degree of similarity between the segmentation and the ground truth. (ii) *Efficiency* relates to the amount of time required to perform segmentation. (iii) *Repeatability* is the extent to which similar results can be obtained from multiple segmentations of the same image. The three criteria are related. For example, given sufficient time, the user can often refine the segmentation to reach higher accuracy. This paper relies on these criteria to assess interactive segmentation through a user study. To assess repeatability, different users are asked to use different segmentation approaches under the same conditions. Because the interactive segmentation process depends on the user's assessment of the results, the trade off between efficiency and accuracy is left to user's discretion.

User studies are a common way to evaluate different interaction techniques [21]. McGuinness and O'Connor [23] presented an experimental user study comparing four interactive segmentation approaches according to Olabarriaga and Smeulders [28]'s criteria. Results revealed better performance by methods using foreground / background labeling. Hebbalaguppe et al. [16] investigated the influence of various types of user scribbling input on GC segmentation performance. The study considered: (i) basic foreground / background *scribbling*, as originally introduced by Boykov and Jolly [2], (ii) *bounding box with scribbling*, where the user frames the object in a bounding box to focus segmentation, with optional foreground and background scribbles to adjust the results, and (iii) *outline with scribbling*, where the user initially draws the object boundary and uses scribbling for segmentation refinement. This last user interaction method was essentially manual segmentation. As mentioned before, our approach requires the user to draw a rough contour of the object boundary. However, compared with the *outline with scribbling* method of Hebbalaguppe et al. [16], ours requires far less accuracy in the drawing, making it closer in spirit to scribbling rather than manual segmentation. In addition to time and accuracy, Hebbalaguppe et al. [16] monitored user brain activity during segmentation with electroencephalography (EEG) as a measure of *user-effort*. Assuming that less user-effort reflects more efficient interaction, the study highlights the benefits of using scribbles over a manual segmentation. The study also shows a time reduction when using the *bounding box with scribbling* compared to *outline with scribbling* and *scribbling* approaches. This could be because the user is allowed to focus more exclusively on foreground object labelling, since background labels are automatically generated outside the bounding box. A similar effect is exploited in our approach. Once the rough contour is drawn, foreground and background labels are automatically generated inside and outside the object, respectively, allowing the user's attention to focus near the object boundary.

2.3. Graph reduction

Graph-based segmentation extends naturally to three-dimensional images. Unfortunately, computation time increases with image size, significantly impacting interaction. To reduce segmentation time, adjacent vertices can be grouped together according to a homogeneity criterion to form a single vertex called a *super-pixel*. This is typically performed before user interaction as a pre-segmentation step. Several approaches have been proposed to extract super-pixel structures from an image, such as normalized cuts [25], TurboPixels [18] and simple linear iterative clustering (SLIC) [1]. For interactive graph-based segmentation, each super-pixel forms a single vertex in a new graph of smaller size. Super-pixels have been used with a GPU implementation of RW segmentation [10], with watershed clustering and RW segmentation [7], using hierarchical graph clustering and GC for video segmentation [9], and using random seed generation with a lazy RW strategy [32]. However, the super-pixel extraction step affects the quality of the segmentation results provided by the *main segmentation* algorithm (e.g., RW or GC). If super-pixel extraction fails to detect weak boundaries, the final segmentation inherits these errors and, more importantly, these cannot be corrected through user interaction. Moreover, super-pixels effectively reduce spatial resolution over the entire image, impacting the segmentation.

Lermé et al. [17] proposed a different method to reduce graph size for GC segmentation while preserving high resolution in parts of the image graph where maximum flow is high. During construction, vertices are discarded if they do not contribute significantly to max-flow computation [3]. Unfortunately, for highly textured images, the graph is only reduced slightly and the time spent on graph reduction may not be compensated by the time gained during segmentation.

To our knowledge, very little work has investigated the relevance of user drawings for graph reduction. Such a drawing hints as to where the object boundary is likely to be. Although the bounding box method used by Hebbalaguppe et al. [16] also leverages user input, this is basically to crop the image, and the user still needs to scribble the foreground. GrabCut [30] also uses a bounding box, but fails in images with low contrast due to GMM fitting. Our approach maintains the same quality as the primary segmentation algorithm (e.g., GC or RW) used. Moreover, bounding boxes lack flexibility for dealing with objects whose dimensions are not aligned with the pixel grid or whose shape is not convex.

3. Proposed graph-reduction method

Fig. 1 illustrates our approach. First, the user roughly draws the object boundary. The object of interest is not required to fit inside the contour, making our approach more flexible than bounding box approaches. Starting from this user-drawn contour, a distance map is computed containing the distance from each pixel to the contour. Then, the distance map is used to partition the pixels (vertices) into *layers*, whose thicknesses increase according to the Fibonacci sequence (see Fig. 1.b). Foreground and background labels are then automatically generated on two selected layers, called the *detail significance layers* (DSLs), and vertices beyond the DSL are eliminated from the graph. Finally, a segmentation algorithm (e.g., RW or GC) is run on the reduced graph, thereby accelerating computation. Further benefits of our approach are: (i) it easily extends to

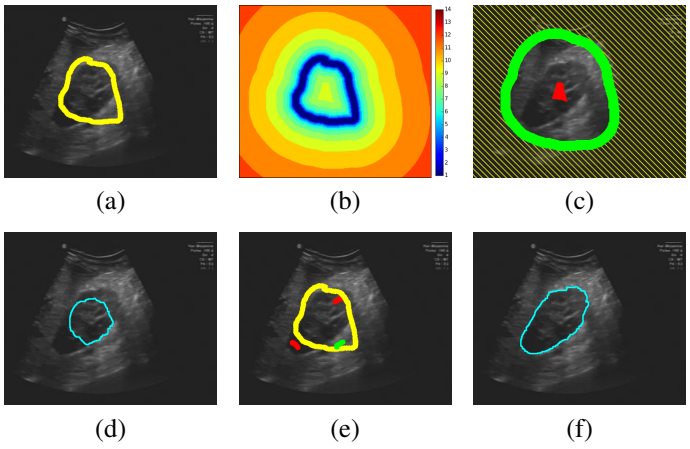


Figure 1: Example of a segmentation using our graph reduction approach : (a) Rough boundary quickly drawn by the user; (b) Layer thicknesses increasing according to the Fibonacci sequence; (c) Seed generation in the inner (red) and outer (green) regions, corresponding to the detail significance layers (*DSLs*); the hatched region (yellow) contains ignored vertices; (d) initial RW segmentation result; (e) refinement by the user with foreground (red) and background (green) labels and (f) final RW segmentation result.

super-pixel representations [10, 7, 9, 32] for further graph reduction; (ii) it is parallelizable using a GPU implemention of the distance transform [31], so that the entire segmentation can be run on a GPU [12, 8]; and (iii) unlike super-pixel-based graph reduction, our approach preserves full resolution near the boundary and only one segmentation algorithm is required (e.g., RW or GC), thereby preserving the performance and homogeneity of the approach.

3.1. Random walker segmentation

Our approach applies to most interactive graph-based segmentation methods using foreground / background labels (e.g., GC [2], GrabCuts [30], and RW [11]). For concreteness, we focus on RW, which we now review.

First, the graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is built from the image, where $v \in \mathcal{V}$ are the vertices corresponding to pixels of the image and $e \in \mathcal{E} \subseteq \{\{u, v\} : u, v \in \mathcal{V}\}$ are the edges connecting each pair of adjacent pixels. A weight w_{ij} is assigned to the edge e_{ij} that connects vertices v_i and v_j such that

$$w_{ij} = \exp(-\beta(g_i - g_j)^2), \quad \forall i, j = 1, .., N, \; i \neq j, \tag{1}$$

where g_i and g_j are the pixel intensities at vertices v_i and v_j, respectively, N is the total number of pixels in the image, and β is a user-supplied constant. A large β results in high sensitivity to weak boundaries.

Assume that the user has manually labeled sets of foreground and background pixels (seeds), partitioning the vertices $\mathcal{V}$ into a set S of seeds and a set U of unlabeled vertices. For each label category (foreground or background), and for each unlabeled vertex $v_i \in U$, the RW algorithm computes the probability that a random walk starting at v_i will reach a seed of the label category in question before reaching a seed of the opposite category. Let $\mathbf{x}$ be an $N \times 2$ matrix such that each column contains the probabilities of the vertices belonging to one of the two label categories. We can represent $\mathbf{x}$ as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x_S} \\ \mathbf{x_U} \end{bmatrix},$$

where $\mathbf{x_U}$ is the $|U| \times 2$ probability matrix of the unlabeled vertices and $\mathbf{x_S}$ is the $|S| \times 2$ probability matrix of the seeded vertices. Then, the graph's Laplacian matrix $\mathbf{L}$ is given by [11]

$$L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -w_{ij} & \text{if } v_i \text{ and } v_j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where $d_i = \sum_j w_{ij}$ is the degree of the vertex i.

$\mathbf{L}$ can be decomposed as

$$\mathbf{L} = \begin{bmatrix} \mathbf{L_S} & \mathbf{B} \\ \mathbf{B^T} & \mathbf{L_U} \end{bmatrix},$$

where the subscript S (resp. U) denotes the seeded (resp. unlabeled) components of the Laplacian matrix $\mathbf{L}$, and $\mathbf{B}$ is the remainder submatrix of $\mathbf{L}$. The unknown probabilities $\mathbf{x_U}$ are obtained by solving

$$\mathbf{L_U x_U} = -\mathbf{B^T x_S}. \tag{3}$$

$\mathbf{L_U}$ is a sparse matrix and is easily inverted. The speed of the algorithm depends on how efficiently Eq. 3 is solved, i.e., $O(|U|)$. Generally, $|S| \ll |U|$, so the segmentation time is strongly dependent on the image size $N = |\mathcal{V}|$.

3.2. Interactive graph reduction

In our approach, the initial user input is a rough drawing of the object boundary. This section shows how multi-scale layers are built from the user's drawing. Then, we describe how to use the layers to automatically generate foreground and background seeds and reduce the graph for segmentation. After the initial segmentation results are obtained, the user can add foreground and background labels for segmentation refinement.

3.2.1. Layer construction

Assuming a cooperative user, the true object boundary is most likely to be near the drawn contour. To focus the search for the boundary near the drawn contour and ignore details in distant regions, we adaptively reduce image resolution according to the distance from the drawn contour. A Euclidean distance map D is computed as [22], such as,

$$D(p) = \sqrt{\sum_i^d (p_i - l_i)^2}, \tag{4}$$

where the subscript i indicates the i^{th} coordinate in a d dimensional space, and l is the labeled pixel with the smallest Euclidean distance to the unlabeled pixel p. Thus, D is the distance from each pixel to the drawn contour. Pixels are then grouped into layers which quantify the significance, or relative *scale*, of the information contained in the image, based on the distance map. This notion of scale is naturally embedded in layers whose thicknesses increase multiplicatively with distance.

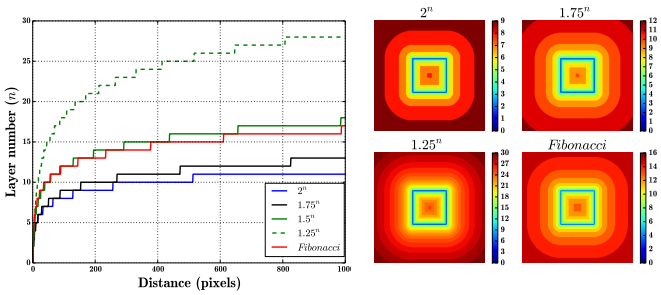


Figure 2: Effect of thickness function on layer generation: (a) plot of number of layers generated according to the distance from the drawn contour using different thickness functions $t(n) = ka^n$ with $k = 1$ and $a = 2$, $a = 1.75$, $a = 1.5$ and $a = 1.25$ and the Fibonacci function $t(n) = t(n-1) + t(n-2)$, (b) results of layer generation using a square drawing (blue).

Thus, the thickness $t(n)$ of the n^{th} layer is given by the exponential relationship

$$t(n) = ka^n, \tag{5}$$

where $a > 1$ is a constant representing the thickness ratio between layers n and $n+1$, and $k > 0$ is a multiplicative constant representing the thickness assigned to the contour drawing itself. For example, with a constant $a = 2$ each layer n is twice as thick as the previous layer $n-1$. We want to find, for each pixel p, the index (or scale) n that corresponds to the highest $t(n)$ that is lower than its distance $D(p)$ to the drawn contour. Hence, we assign a layer number, $\mathcal{L}(p)$, to each pixel p in the image such that

$$\mathcal{L}(p) = \left\lfloor \frac{\log(D/k)}{\log(a)} \right\rfloor. \tag{6}$$

In the particular case where

$$t(n) \simeq \frac{1}{\sqrt{5}}\phi^n,$$

where the constant $\phi = (1+\sqrt{5})/2 \simeq 1.618$ is the *golden ratio*, thickness grows according to the Fibonacci sequence:

$$t(n) = \begin{cases} n, & n \in \{0,1\} \\ t(n-1) + t(n-2), & \forall n =\geq 2. \end{cases} \tag{7}$$

Fig. 2 shows examples of layer maps generated using different $t(n)$. Experimentally, we observe that the Fibonacci sequence provides a reasonable trade-off between the goals of maintaining high resolution near the drawing and rapidly decreasing resolution far away from it. For the remainder of this paper, we choose the Fibonacci sequence to build the layers. However, a and k can be adjusted to adapt layer growth depending on the application.

In practice, Eq. 7 is computed using Binet's formula [33]

$$t(n) = \frac{\phi^n - (-\phi)^{-n}}{\sqrt{5}}. \tag{8}$$

Since

$$\left| -\frac{1}{\phi} \right|^n < \frac{1}{2}, \quad \forall n > 1, \tag{9}$$

Eq. 6 can be rewritten as

$$\mathcal{L}(p) = \left\lfloor \frac{\log(\sqrt{5}D(p) + \frac{1}{2})}{\log(\phi)} \right\rfloor, \tag{10}$$

were $\frac{1}{2}$ is added for rounding convenience based on Eq 9.

Note that the distance map D in Eq. 10 need not be Euclidean; different distance metrics may suit different applications or imaging modalities. For example, a gradient-based geodesic distance would grow slowly in homogeneous regions, increasing the thickness of the associated layers. On the other hand, highly textured regions would be associated with a locally fast increasing distance map, leading to thinner layers. Therefore, image information can be integrated to the distance map to facilitate layer growth according to local image characteristics, e.g., with texture-based geodesic distance map [29]. For the remainder of this paper, we consider the Euclidean distance metric in Eq. 4.

3.2.2. Segmentation

We assume that the user roughly labeled only one object boundary, thereby defining an *inner* R_{in} region and *outer* R_{out} region[1]. Recall that the flexibility of the contour drawing paradigm allows the true object boundary to lie on both the inner and the outer regions, which is not the case for previously proposed user-selected bounding box [16]. Next, pixels are assigned to their respective layers using Eq. 10 and the most distant layer for each region is computed:

$$L_{in} = \max(\mathcal{L}(p)), \quad \forall p \in R_{in}$$

and

$$L_{out} = \max(\mathcal{L}(p)), \quad \forall p \in R_{out}.$$

The smaller of these two numbers determines the index of the *detail significance layers (DSLs)*

$$DSL = \min(L_{in}, L_{out}). \tag{11}$$

Pixels lying on the *DSLs*, inside and outside the user-drawn contour, are automatically labeled as foreground and background seeds, respectively, and all vertices beyond the *DSLs* are discarded from the graph[2]. The runtime of RW segmentation is $O(|U|)$. Thus reducing the the number $|U|$ of unlabeled vertices improves computation time.

The proposed layer formulation naturally lends itself to a user-defined multi-resolution representation of the image. This can be obtained using super-pixel clustering, where the size of the super-pixels grows according to the layer where they reside. This highly general approach will be illustrated in Section 6.3. For the moment, we consider the binary case where

[1] In a multi-label segmentation, the user must create separate objects by roughly marking their boundaries. This is the only constraint imposed on the user.

[2] We assume that no background regions (holes) are contained inside the target object to segment. This limitation can be addressed by drawing multiple contours inside the object to separate the background regions.

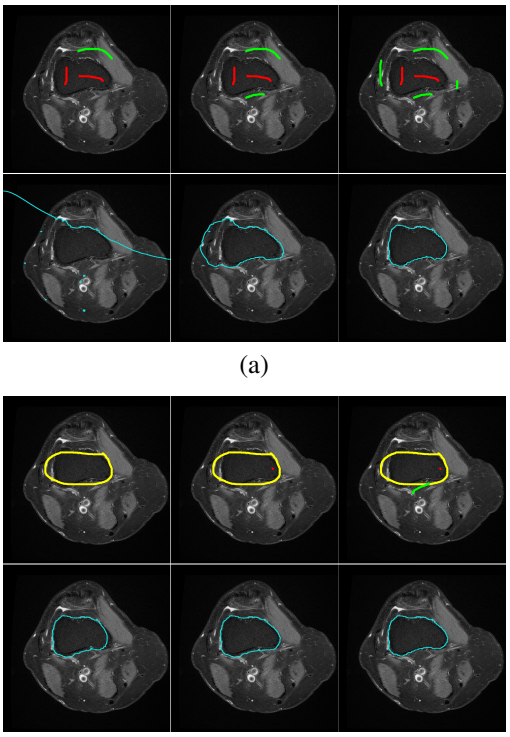


Figure 3: Step-by-step RW segmentation example showing the geometric constraints of label positioning : (a) conventional RW segmentation approach; (b) our segmentation approach. Top rows show foreground (red) and background (green) seeding and rough contour drawing (yellow), respectively. Bottom rows show results (blue). Label positioning constraints force the user to correct the segmentation through multiple iterations, surrounding the object with background labels. This is explicitly expressed with the rough contour drawing in our approach.

the image is represented using only two resolution categories: (i) a *pixel-resolution* below the *DSLs*, and (ii) a *one-region-resolution* above the *DSLs*. This amounts to treating pixels lying above the *DSL* as a single vertex in the graph, leading to an inner vertex inside the object and an outer vertex outside the object. This particular case, achieved by thresholding of the distance map according to Eq. 11, allows us to validate our hypothesis that pixels far from the contour drawing (i.e., above the *DSL*) do not contribute to the segmentation.

The *DSL* can also be selected manually. This controls how loosely the user-drawn contour can fit the true contour. Increasing the *DSL* reduces the number of ignored vertices (the dashed yellow area in Fig. 1.c), providing a larger unlabeled area where the contour is sought. Decreasing the *DSL* leads to a smaller graph and, therefore, to fewer computations, but may require a more accurate drawing to achieve good results. The *DSL* computed automatically with Eq. 11 provides a reasonable compromise between these two extremes.

4. Interaction constraints and segmentation behavior

In this section, we highlight the explicit and implicit constraints imposed by the contour drawing paradigm compared to standard foreground-background seeding (FBS) input. Then, we discuss the sensitivity of our approach to inaccuracies in the contour drawing.

4.1. User interaction constraint

Most cases require to approximately surround the object with the contour drawing to obtain a satisfactory segmentation. In the presence of weak boundaries, the FBS approach implicitly embeds a similar spatial constraint on seed positioning. Fig. 3 shows an example of a RW segmentation using FBS interaction. Due to the initial seed positions, the user is forced to correct the segmentation with background labels through several feedback exchanges with the segmentation algorithm. For high resolution images, segmentation feedback can take a longer time, rendering this interaction tedious. It is possible to anticipate the behavior of FBS segmentation by labeling the potential segmentation overflow areas surrounding the object. This very nearly amounts to a rough contour drawing.

Fig. 4 shows more examples of RW segmentation using our approach. In most cases, the contour drawing is sufficient to correctly segment the object. For other complex scenarios where the boundary is weak or missing, few additional foreground-background labels are required (e.g., the second row of Fig. 4).

4.2. Sensitivity of the contour drawing

To measure the sensitivity of our method to the drawing, we systematically evaluated the quality of the segmentation as this drawing departs from a ground truth segmentation. That is, the drawing was iteratively shrunk inward (respectively expanded outward) the object. At each iteration, 50 contour drawings were generated using a random path that roughly follows the shrunk or expanded curve [14]. Then, the segmentation result associated to each trial is evaluated using the harmonic mean of precision and recall (also known as the Dice index), denoted

$$F1\text{-}Score = \frac{2TP}{2TP + FP + FN} \in [0, 1], \qquad (12)$$

where TP, FP and FN are the true positive (object surface), false positive and false negative scores, respectively. The *F1-Score* tends to 1 as the segmentation result approaches the ground truth. Fig. 5.a shows the *F1-Score* as a function of a drawing's distance from the true boundary. The results prompt two important observations. First, note that the curve is skewed in the outward (negative) direction meaning that the interaction is slightly more robust to errors when the contour is drawn outside the object boundary. Second, significant cyclic drops of the *F1-Score* can be observed. These are related to the quantization of the layers generated using the Fibonacci sequence (see Fig. 6). In fact, Eq. 11 selects the largest layer number. However, the layer with the largest number in the inner region can occasionally contain too few pixels, leading to segmentation errors. Examples of segmentation failure are shown in Fig. 5.b-c. Note that the location of the *F1-Score* valleys are specific to the image and the considered object. In practice, this does not affect the quality of segmentations obtained interactively, as rapid

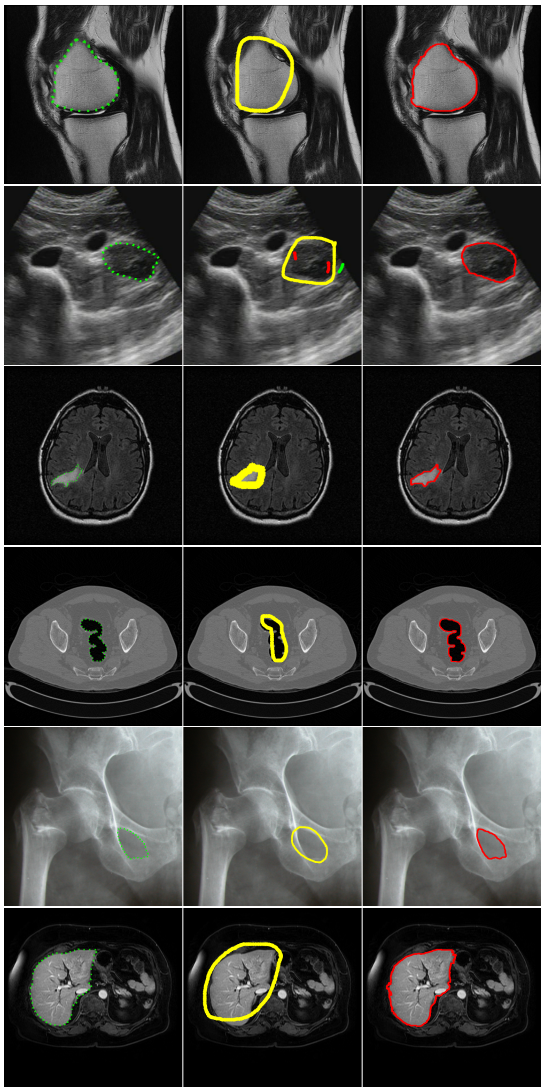
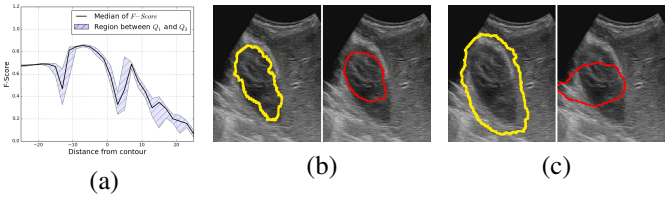


Figure 4: Examples of segmentation using the random walker with our approach: (left) ground truth image, (middle) rough contour drawn by the user, and (right) segmentation results. For most cases, a rough contour drawing is sufficient to obtain a satisfactory segmentation. Note that for the complex segmentation of the ultrasound image of kidney (second row), a few additional foreground-background labels are required.

response from the segmentation algorithm allows for a quick interactive correction with few additional labels, and all corrections benefit from the initial speed-up. This is demonstrated experimentally in Section 5.

5. User study

We conducted a controlled experiment to compare our rough contour drawing (RCD) method against the conventional foreground-background seeding (FBS) approach for segmentation. RCD has the advantage of reduced computation time for each iteration of segmentation, but incurs the up-front cost of drawing a contour, hence the need for an experimental comparison. The following questions are addressed: (i) Does RCD provide satisfactory results in terms of segmentation quality?



(a) (b) (c)

Figure 5: Sensitivity of the algorithm to the accuracy of the contour drawing: (a) plot of median value of *F1-Score* and range between first and third quartiles, respectively Q_1 and Q_3 as function of drawing distance to the true object boundary; positive (resp. negative) distance indicates an inward (resp. outward) drawing distance. Examples of segmentation failure at distance 3 pixels (b) and -13 pixels (c).

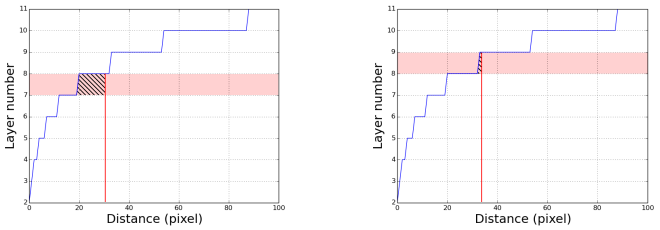


Figure 6: Illustration of the quantization effect of the Fibonacci sequence on seed generation: The left image represents a distance of 31 pixels generating a $DSL = 8$, the number of seeds generated is related to the size of the dashed area. On the right image, a higher $DSL = 9$ is generated with a distance of 36 pixels. However, the number of generated seeds is smaller.

(ii) Does RCD's reduced computation time help the user reduce the overall segmentation time? (iii) Is performance affected by the input device used? We assessed performance by measuring the quality of the final segmentation, the overall time to complete a segmentation, and the number of labels drawn by the user.

5.1. Study design

With the FBS drawing technique, the user labels the foreground object and the background as is conventionally done [11]. With the RCD drawing technique, the user draws a rough contour of the object to reduce the graph and later uses foreground / background labeling to refine the segmentation. Two input devices were used: a standard mouse, and a "Grip Pen" on a Wacom Cintiq Companion Hybrid graphics tablet connected to the desktop PC. The Wacom device allows the user to physically draw on the tablet screen using a handheld stylus pen. There were thus two experimental factors crossed to form four main conditions: *Device* (Mouse or Pen) × *Drawing* (FBS or RCD) yielding the combinations ***M+FBS***, ***P+FBS***, ***M+RCD*** and ***P+RCD***. Eq. 1 with $\beta = 300$ was used in all cases, and all processing was done on an Intel© Core i7-2630QM 2GHz × 4 machine.

Sixteen participants (13 male, 3 female), primarily undergraduate and graduate students with no particular expertise in medical imaging, were recruited. Some had prior experience with interactive segmentation using FBS with GC and/or RW.

An initial dataset of 22 images was prepared, ranging from 256 × 256 to 1348 × 1101 pixels. Images comprised computed tomography (CT), magnetic resonance (MR) and X-ray

images from the cancer imaging archive database [5], to which we added ultrasound (US) images acquired with an Ultrasonix SonixTablet. Images were selected to cover a broad range of medical applications: 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 images were manually segmented to generate ground truth data. This dataset of 22 images was then divided into two subsets, dataset 1 (*DS1*) and dataset 2 (*DS2*), of 11 images each.

Each participant completed tasks under the four main conditions (*M+FBS*, *P+FBS*, *M+RCD*, *P+RCD*) whose order was counterbalanced according to a 4×4 Latin square (i.e., each quarter of the participants went through the conditions in an order given by one row of the Latin square). For each participant, two of the conditions were performed using *DS1*, and the other two were performed using *DS2*. Further counterbalancing ensured that half of the participants started with *DS1*, the other half started with *DS2*. In total, there were 16 participants $\times$ 2 Drawing conditions (FBS and RCD) $\times$ 2 Devices (Mouse and Pen) $\times$ 1 data set (*DS1* or *DS2*) $\times$ 11 images per data set = 704 interactive segmentation trials.

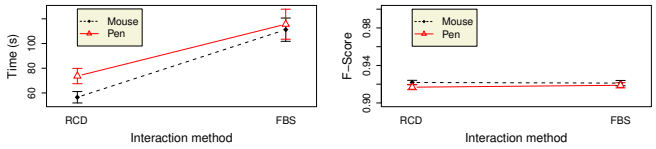
In each trial, participants were shown the ground truth segmented image and were asked to reproduce a similar result. The ground truth segmentation was provided because the purpose of the study was to assess segmentation performance, not the medical image interpretation skills of the participants. Participants assessed the accuracy qualitatively by visualization of the ground truth image. No time limit was set for the segmentation task and the accuracy was left to user satisfaction. This reflects the trade-off between ease of use and the time required to segment the image. For each main condition (*M+FBS*, *P+FBS*, *M+RCD*, or *P+RCD*), participants were first introduced to the segmentation method through a training session using 13 images belonging to neither *DS1* nor *DS2*. No data were recorded during training. Then, during the recorded session, the participants were asked to perform the most accurate segmentation they could with respect to the ground truth in the shortest time possible for each image. A trial consists of segmenting one image. During each trial, editing (where the user positions the labels) and processing (where the segmentation results are computed) phases alternated.

With interactive segmentation, any additional user interface features, such as ability to zoom or undo, would affect performance, however these are not the focus of our study. We therefore controlled for such differences between user interfaces by keeping the user interface as simple as possible, restricting user actions to (i) drawing foreground, background and contour (for RCD) labels, (ii) resizing the drawing brush, and (iii) erasing seeds. Participants could not zoom or undo.

At the end of each trial, when a satisfactory segmentation result for the image was obtained, we recorded the time required to perform the segmentation and the accuracy of the final result using Eq. 12. Because user performance varies substantially between images, the overall segmentation performance for the whole dataset was considered, rather than for each individual image. In other words, the time reported in our results section is the total time required for a participant to segment all 11 images, and the accuracy score is the average *F1-Score* obtained over the 11 images.

Criterion	Results
Time	The fastest interactive segmentations are achieved using **our approach** (RCD), irrespective of the device (mouse or pen) that is used.
Accuracy	Our graph reduction approach does not sacrifice accuracy in any significant way, irrespective of the device (mouse or pen) that is used.
Human effort	Overall, interactive segmentation that is initiated using our RCD approach does not require significantly more manual labeling than segmentation initiated with the conventional FBS approach.

Table 1: Key conclusions of the user study.



(a) (b)

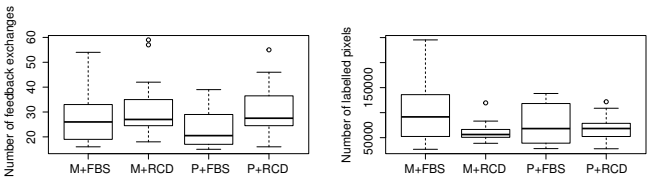
Figure 7: (a) The mean time for each condition and (b) the *F1-Score* (a larger score means better accuracy) for each condition.

5.2. Results

Table 1 summarizes our conclusions. The details of the analysis supporting these conclusions are provided in the next two subsections, with important results in bold.

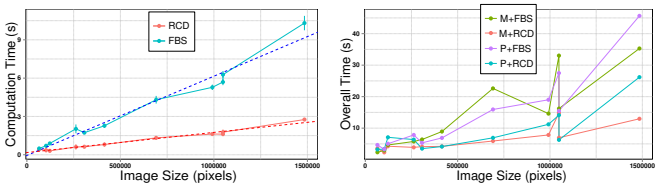
5.2.1. Interaction

Fig. 7 shows average time and accuracy for the main conditions. A Shapiro-Wilk normality test revealed that the time taken to perform segmentations was not normally distributed [$p < 0.01$]. Therefore, a non-parametric ANOVA-type statistic (ATS) test [4] was considered. Segmenting using the RCD ($mean_{RCD}(Time) = 65.09 \pm 3.98 sec$) was significantly faster than using FBS ($mean_{FBS}(Time) = 113.43 \pm 6.92 sec$) [$p < 0.01$]. Qualitatively, the time reduction was more substantial for larger images. These results support our initial hypothesis that **the time spent drawing the rough contour results in a significant gain in the overall segmentation time**. Segmentation using the pen ($mean_{pen}(Time) = 94.65 \pm 6.67 sec$) was significantly slower than using the mouse ($mean_{Mouse}(Time) = 83.86 \pm 7.42 sec$) [$p < 0.01$]. Considering the intuitiveness of the tablet and pen for drawing, this was unexpected. This result could be explained by the low drawing accuracy required by RW segmentation to obtain satisfactory results. The *F1-Score* results show that both devices provide sufficient control to perform a good segmentation. Indeed, no significant difference was found between the four conditions regarding the *F1-Score* ($mean(F1\text{-}Score) = 0.919 \pm 0.001$) with [$p = 0.70$] and [$p = 0.14$] for the *Drawing* and the *Device* factors, respectively ; meaning that **using the rough contour drawing (RCD) method, users achieved the same segmentation accuracy in shorter time with both devices**.



(a) (b)

Figure 8: Box plot of (a) number of segmentation feedback exchanges and (b) number of labeled pixels. Note that in the latter, FBS shows larger variations than RCD.



(a) (b)

Figure 9: Results of the segmentation time according to the image size: (a) Computation time excluding user interaction time, dashed lines represent the linear regression of the data for both FBS (blue) and our approach RCD (red) and (b) overall segmentation time including user interaction time.

To evaluate user effort, we analyzed the number of pixels that were labeled by the participants and the number of segmentation feedback exchanges required to achieve the final segmentation (i.e., the number of times the user pushed the segmentation button to view an intermediate result), shown in Fig. 8. Tukey contrast tests [27] on the normalized number of labeled pixels and the number of segmentation feedback exchanges reveal no significant differences between the four conditions, suggesting that **all the approaches required similar amounts of effort**, on average.

However, a Fligner-Killeen test [6] reveals a significant inhomogeneity of variances in the number of labeled pixels [$\chi^2 = 21.95, df = 3, p < 0.01$]. This reflects the larger inter-quartile range displayed for the FBS approaches in Fig. 8.b. In other words, **the conventional labelling approach allows larger variation in the drawings**. This result is related to the implicit label positioning constraints in the FBS approaches, discussed in Section 4.1. The FBS approach gives the user more freedom in drawing, but the positions of truly useful labels are actually constrained by the algorithm. Therefore, the number of labeled pixels varies significantly from one user to the next, depending on the usefulness of their drawings. In contrast, the RCD approach is explicitly constrained, thereby facilitating the seeding process.

5.2.2. Computation time

From the previous study, we extracted the final labels generated by the participants for each image. We also recorded the average time required to compute the segmentation on each image, ignoring the time required by participants for drawing. Fig. 9.a shows that the average computation time grows linearly with image size. This is because the size of the linear system of Eq. 3 is proportional to the size of the image. The computation time trend with respect to image size is estimated with a linear regression giving 6.203 $s/Mpixel$ for the FBS and 1.577 $s/Mpixel$ for the RCD. Therefore, **our approach is on average 3.93 times faster than the conventional RW segmentation**.

Fig. 9.b shows average overall segmentation time (including user interaction time) as function of image size. First, we note that the time required for the user to perform segmentation increases with image size. Second, the most significant benefits of the contour-based graph reduction approach occur for large images. For the largest image with size 1348×1101, the average participant performed the segmentation in 35.28 ± 8.42 s for *M+FBS*, 12.96 ± 4.56 s for *M+RCD*, 45.64 ± 20.22 s for *P+FBS* and 26.18 ± 8.26 s for *P+RCD*. In addition to reducing segmentation time, our graph reduction approach leads to the highest repeatability between participant performances. This is due to the explicit label positioning constraint that forces participants to focus the drawing while using our segmentation approach.

6. Extension to other segmentation algorithms

Having demonstrated the benefits of our approach in the context of RW segmentation through a user study, we now show how these extend to other graph-based approaches and how they compare and can be combined with super-pixel clustering for further improvements in efficiency. A benchmark segmentation is experimented and three key features of our method are highlighted: (i) the benefits of combining our approach to super-pixel-based graph reduction, for example using simple linear iterative clustering (SLIC) [1], are shown in Section 6.1, (ii) the independence of our graph reduction approach with respect to the choice of segmentation algorithm is shown in Section 6.2 by extending our approach to the GC [2] and Lazy Snapping [19] segmentation algorithms, and finally (iii) multi-resolution graph segmentation using multiple super-pixel resolutions. Table 2 summarizes the segmentation time obtained for each method. For all the experimented algorithms, using our graph reduction, the segmentation performs faster than the conventional super-pixel graph reduction. Furthermore, our approach achieves slightly better *F1-Score*s, due to the preservation of full pixel resolution around the boundary (which is not possible with super-pixels). When combined with super-pixel reduction, both RW and GC are accelerated. However, this is not the case for Lazy snapping segmentation approach, due to the on-line k-means clustering. The remainder of this section provides further details about the experiment results.

6.1. Combination with super-pixels

Our graph reduction approach is independent of the image graph. Therefore, super-pixel clustering approaches can be used alongside it to further reduce the graph. To illustrate this, we choose the simple linear iterative clustering (SLIC) super-pixel method, which provides satisfactory results in terms of under-segmentation error [1]. A cryosectional image of human anatomy of size 2048×1216 [20] was used for this experiment.

		Off-line	Column 1 **Random Walker**	Column 2 **Lazy Snapping**	Column 3 **Graph Cuts**
Conventional		–	26.401 ± 0.364	13.235 ± 1.726	3.861 ± 0.167
(i)	- Graph building	–	0.188 ± 0.006	1.880 ± 0.069	1.693 ± 0.140
	- k-means	–	–	1.114 ± 0.033	–
	- Eq. 3/max-flow	–	25.813 ± 0.356	9.726 ± 1.726	1.595 ± 0.070
Conventional w/ super-pixels		39.391	1.789 ± 0.028	2.671 ± 0.091	1.068 ± 0.035
(ii)	- Super-pixels	39.391	–	–	–
	- Graph building	–	0.205 ± 0.004	0.604 ± 0.023	0.600 ± 0.020
	- k-means	–	–	1.573 ± 0.077	–
	- Eq. 3/max-flow	–	1.168 ± 0.029	0.003 ± 0.002	0.00071 ± 0.0
Our approach		**0.551**	1.352 ± 0.023	**1.541 ± 0.068**	1.004 ± 0.046
(iii)	- Layers	0.551	–	–	–
	- Graph building	–	0.704 ± 0.019	0.567 ± 0.049	0.530 ± 0.030
	- k-means	–	–	0.530 ± 0.039	–
	- Eq. 3/max-flow	–	0.038 ± 0.009	0.011 ± 0.0	0.024 ± 0.002
Our approach w/ super-pixels		39.942	**0.822 ± 0.026**	3.588 ± 0.189	**0.901 ± 0.029**
(iv)	- Layers	0.551	–	–	–
	- Super-pixels	39.391	–	–	–
	- Graph building	–	0.417 ± 0.021	0.512 ± 0.020	0.514 ± 0.023
	- k-means	–	–	2.683 ± 0.193	–
	- Eq. 3/max-flow	–	0.000447 ± 0.0	0.000039 ± 0.0	0.00004 ± 0.0

Table 2: Computation time comparison (mean ± one standard deviation) of the conventional segmentation approaches to our graph reduction approach with and without super-pixel pre-segmentation. The *off-line* time indicates the required pre-processing time to build the graphs (i.e., layers and/or super-pixels). Columns represent the method used to solve the segmentation problem, i.e., Random Walker, Lazy Snapping and Graph cuts. Rows represent the graph reduction approaches used during the segmentation, i.e., (i) *conventional* approaches as described in [11], [19] and [2] respectively, with no graph reduction (ii) conventional approaches used with SLIC super-pixel graph reduction, (iii) *our approach* as described in Section 3, and (iv) *our approach* combined with SLIC super-pixel graph reduction. Shaded rows represent the total segmentation time. Best results are showed in bold characters.

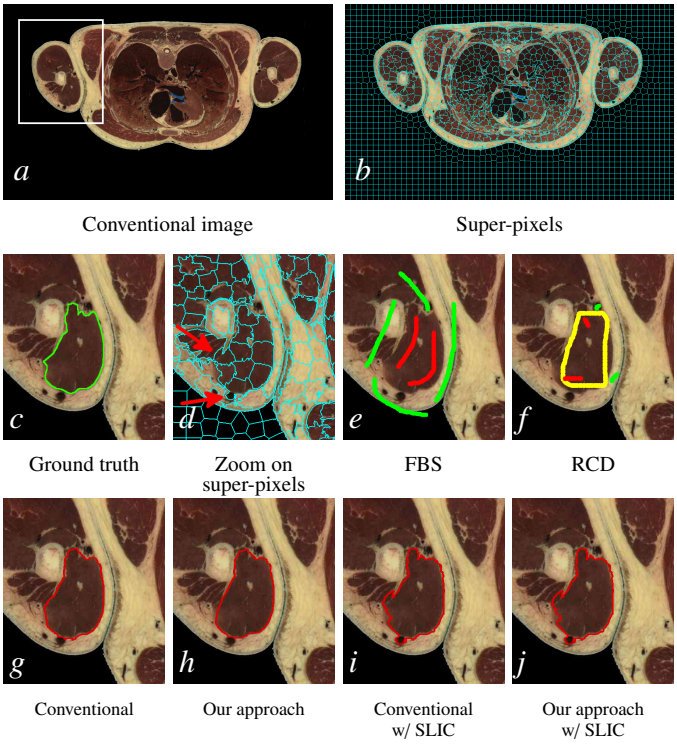


Figure 10: Random walker segmentation of the right biceps of a high-resolution cryosectional image: (a) Conventional image, the white rectangle is the region represented in figures (c-j), (b) super-pixel clustering using SLIC, (c) ground truth segmentation, (d) zoom on the super-pixel clustering, the red arrows indicates segmentation errors, (e) foreground and background drawings for RW approaches with and without SLIC super-pixels, (f) rough contour drawings approaches with and without SLIC super-pixels, (g-j) random walker segmentation results using conventional approach, our approach, conventional approach with super-pixels, and our approach with super-pixels, respectively. When using super-pixels, the final segmentation inherits the SLIC segmentation errors.

The task was to segment the right hand biceps using identical input labels for RW with and without SLIC (resp. for our approach with and without SLIC) approaches (see Fig. 10.e-f). Table 2 (column 1) shows the computation time results of the four segmentation approaches. The experiment was repeated 100 times using the same labels, to account for time lags from external factors. The time to generate 2000 super-pixels using SLIC was 39.391 *s*. Using our approach, the time required to build the layers from the RCD was 0.551 *s*. Note that the layers are computed immediately after the user draws the contour. These two computations are only carried out once. However, the time required for super-pixel clustering renders the segmentation inefficient for live applications. The computational bottleneck for RW segmentation is solving Eq. 3 (25.813 ± 0.356 *s*). Using our approach, this computation is reduced to 1.352 ± 0.023 *s*, outperforming RW with SLIC. Moreover, our approach can be combined with super-pixels to further reduce the computation time to 0.822 ± 0.026 *s*. However, using super-pixels can lead to segmentation errors in the case of weak boundaries, as illustrated in Fig. 10.d. Our approach, preserves high-resolution segmentation results near the object boundary (0.968 using our reduction approach vs. 0.938 using SLIC reduction). This is important for interactive segmentation, since the user cannot possibly correct segmentation errors that originate in super-pixel creation.

6.2. *Extensions to graph cut and lazy snapping segmentation algorithms*

To show the generalizability of our approach, we applied our graph reduction approach to lazy snapping segmentation [19] (see Fig. 11). Lazy snapping segmentation is based on interactive GC [2]. Briefly, the GC approach is based on the minimization of a Gibbs energy function

$$E = \sum_{v_i v_j} E_{binary}(v_i, v_j) + \lambda \sum_{v_i} E_{unary}(v_i), \qquad (13)$$

where in the case of the lazy snapping segmentation, E_{binary} represents similarity between pairs of adjacent vertices and E_{unary} represents similarity to the labeled vertices. The same energy functions as proposed in the original algorithm [19] were used. Hence, E_{unary} is calculated using the minimum distance between an unlabeled vertex and the k clusters of the k-means algorithm applied to the labeled vertices. The parameter λ balances the two energy terms. A large λ enables treating non contiguous regions as belonging to the same object. We empirically set $\lambda = 0.01$ for our experiment. Similarly to Section 6.1, we experimented four algorithms based on the lazy snapping segmentation: (i) and (ii) using FBS respectively without and with SLIC super-pixel pre-segmentation, and (iii) and (iv) using our graph reduction respectively without and with SLIC super-pixels. Table 2 (column 2) shows the computation time results of the four approaches averaged over 100 repetitions using the same labels. Using our graph reduction approach we achieve a faster segmentation with a better *F1-Score* (0.976 using our approach without SLIC) than the approaches using

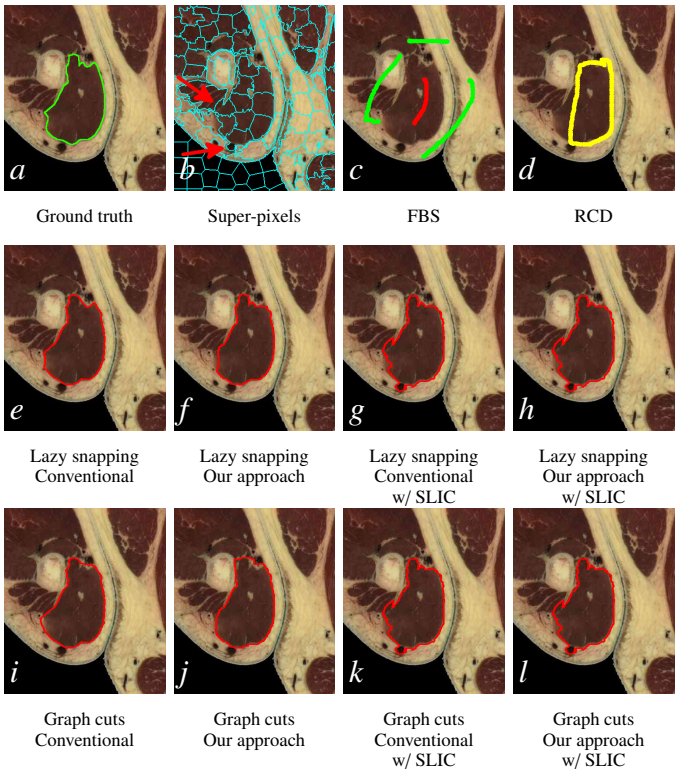


Figure 11: Graph cuts and Lazy snapping segmentation of the right biceps of a high-resolution cryosectional image (zoomed): (a) Ground truth image, (b) SLIC super-pixels, (c) Foreground and background drawings for conventional approaches with and without super-pixels, (d) Contour drawings and labels for our segmentation approach with and without super-pixels, (e-h) segmentation results for lazy snapping, (i-l) segmentation results for graph cut (GC). Note that using super-pixels the final segmentation inherits the SLIC segmentation errors.

super-pixels (0.939 using the conventional approach with SLIC and 0.939 using our approach with SLIC). Although the segmentation performance depends on the labels that are used, we can observe in Fig. 11.g-h that the segmentation errors originate from super-pixel clustering. Therefore, they cannot be corrected through user input. When combined to super-pixels, our approach achieved the fastest max-flow computation (row iv-column 2 in Table 2). However, the total segmentation time is slower than the conventional approach with SLIC (row ii-column 2 in Table 2). This is due to the E_{unary} computation using the k-means clustering. Indeed, the more vertices are labeled, the more vertices are involved in the k-means clustering, and the longer k-means takes to converge. Because our approach labels the super-pixels all around the object, more pixels are labeled than using a simple FBS rendering our approach less efficient when combined to super-pixels in this context.

To compare standard GC (without k-means) [2] performance between the four approaches, we ignore the unary energy term of the lazy snapping approach; i.e., we set $\lambda = 0$ in Eq. 13. Table 2 (column 3) shows the computation time results of the four approaches averaged over 100 repetitions using the same labels. A slightly better computation time is achieved using our approach at both the super-pixel (0.901 ± 0.029 s) and the pixel resolutions (1.004 ± 0.046 s) than the conventional segmentation using only super-pixels (1.068 ± 0.035 s). However, using pixel resolution achieves a better *F1-Score* with both the conventional GC (0.968) and our graph reduction approach (0.970).

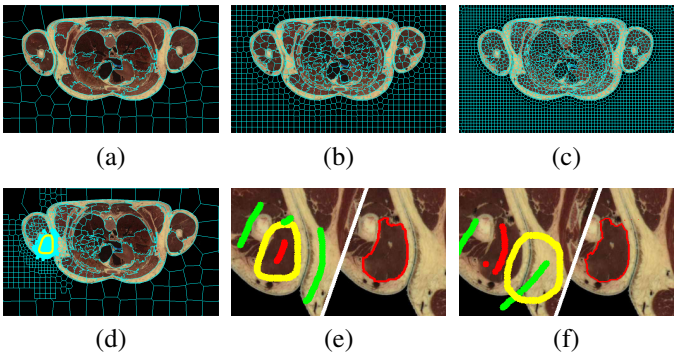


Figure 12: Multi-scale graph generation example using super-pixel images: (a-c) super-pixels generated using SLIC with 100, 1000 and 3000 blocks, respectively, (d) multi-scale graph image generated from the contour drawing (yellow) using combination of super-pixel images a-c, (e-f) examples of segmentation using the multi-scale graph.

6.3. Adaptive multi-scale super-pixels

Rather than simply choosing a *DSL*, the layers described in Section 3.2.1 can be used to combine super-pixel image decompositions at multiple resolutions to build an adaptive multi-scale graph. Fig. 12 shows an example of a multi-scale graph built from three SLIC super-pixel decompositions, R_1, R_2 and R_3 (containing 100, 1000 and 3000 blocks, respectively), and a full resolution image R_4, using the pseudo-code from Appendix B. This requires pre-computing each super-pixel image decomposition offline. For the image of size 1024×608 pixels shown in the example, $1.58s$, $7.15s$ and $20.45s$ were required to compute R_1, R_2 and R_3, respectively. The multi-scale graph was constructed in $0.773s$.

This representation provides: (i) a straightforward reduction in graph size, and (ii) an image resolution that gracefully adapts with distance from the boundary. Indeed, compared with conventional super-pixel clustering, the multi-scale graph prevents resolution loss near the object boundary, e.g., errors due to super-pixel segmentation discussed in Section 6.1, since the user can interactively adjust the segmentation with a pixel-resolution accuracy. Compared to our initial approach, where pixels beyond the *DSL* are completely ignored, the segmentation is slower, i.e., $0.88s$ using the multi-scale graph vs. $0.41s$ using our initial approach. However, because the information far from the drawings is not completely discarded (it is still available at a coarse resolution), the segmentation is less sensitive to contour drawing positioning (see Fig.12.f).

7. Conclusion

In this paper, we proposed a novel graph reduction method based on user drawings for high-resolution interactive image segmentation. In this approach, the user draws a rough contour of the object. Based on a distance map with respect to the drawn contour, image layers are computed such that a pixel far

from the contour is assigned to a thicker layer than a pixel near the contour. Then, foreground and background labels are automatically generated, ignoring pixels on the farthest layers. Our approach is based on a hybrid interaction approach combining rough contour drawing and foreground-background seeding, benefiting from fast computation and intuitive labeling. Finally, a graph-based segmentation method such as random walker segmentation, is applied on the reduced graph, thereby improving the computation time while preserving full resolution near the object boundary.

The user study reported in this paper showed that the amount of time the user spends to draw a rough contour leads to a significant gain in overall segmentation time. This is due to the fact that after the initial segmentation, the user focuses his/her effort on small adjustments, i.e., only few foreground-background labels are required to obtain a satisfactory segmentation. Moreover, the segmentation was experimented with two different devices: a mouse and a tablet with a stylus pen. Surprisingly, although drawing labels using a stylus pen should be more intuitive, segmentation using the mouse is faster. This is probably because labeling-based interactivity for graph-based segmentation requires little drawing accuracy, which is easily reached with the familiar mouse.

Further experiments showed the benefits of our approach both over and combined with graph reduction based on super-pixels and demonstrated its generalization to a variety of graph-based segmentation approaches. Using our graph reduction approach, segmentation is achieved in a time comparable to that achieved with super-pixel graph reduction. However, because our approaches preserves full pixel resolution, the user can interactively correct segmentation errors that cannot be corrected using a super-pixel resolution. Moreover, our approach can be combined to super-pixels and achieve even faster segmentation. This is useful for applications where time is more critical than accuracy. Finally, using our approach with super-pixel decompositions at multiple resolutions, we proposed a multi-scale graph construction that adapts to the user drawings. The multi-scale graph segmentation ensures a more flexible user interaction at the expense of slightly more computation time.

Future work will investigate two main aspects of our approach. First, the distance map used to generate layers does not consider image intensities. Using a gradient-based geodesic distance may provide better information about how to cluster pixels using the layers. Second, it is important to consider the effect of advanced editing and visualization methods on interactive segmentation, especially for 3D segmentation where drawing is not necessarily an intuitive task. Because user interaction is a time consuming part of the interactive segmentation process, reducing the amount of required interaction is of great interest to improve segmentation efficiency.

Acknowledgement

This work was supported by grants from the Natural Science and Engineering Research Council of Canada and the Fonds de Recherche Québécois sur la Nature et les Technologies.

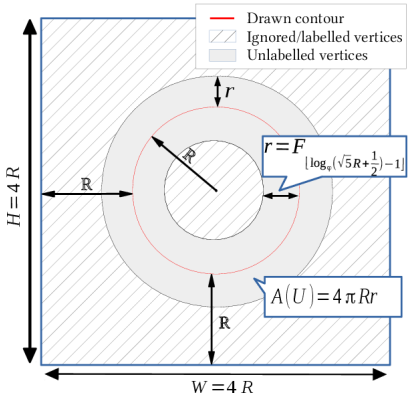


Figure A.13: Representation of the unlabeled pixels given a circle drawing with a radius of R

Appendix A. Computational complexity

Suppose that Eq. 10 is used to generate the layers and Eq. 11 to select the *DSL*. We consider the segmentation of a circular object in the middle of the image, and assume that the user draws a circle with radius R in the center of an image of size $(4R)^2$. This is the worst case circular contour for this image size, as any other size would increase the number of labeled vertices. The complexity of RW segmentation is $O(|U|)$, and $|U|$, the number of the unlabeled pixels, is given by

$$\mathcal{A}(U) = 4\pi Rr, \tag{A.1}$$

where

$$r = t\left(\left\lfloor \log_\phi(\sqrt{5}(R-1)+\frac{1}{2})\right\rfloor\right) \tag{A.2}$$

is the minimum distance of the layer generated from the circular drawing (see Fig. A.13). Using Binet's formula,

$$\mathcal{A}(U) = 4\pi R\frac{\phi^{\left\lfloor \log_\phi(\sqrt{5}(R-1)+\frac{1}{2})\right\rfloor} - (-\phi)^{-\left\lfloor \log_\phi(\sqrt{5}(R-1)+\frac{1}{2})\right\rfloor}}{\sqrt{5}}. \tag{A.3}$$

Noting that $(-1)^{-\left\lfloor \log_\phi(\sqrt{5}(R-1)+\frac{1}{2})\right\rfloor} = \pm 1$,

$$\mathcal{A}(U) \simeq 4\pi R^2 + R(\frac{2\pi}{\sqrt{5}} - 4\pi) \pm \frac{4\pi R}{5(R-1)+\frac{\sqrt{5}}{2}}. \tag{A.4}$$

Then, under our simplifying assumptions, the complexity of our approach is $O(R^2)$.

This result assumes that a distance R separates the drawing from the image boundaries (see Fig. A.13). Hence, by construction, we have $R = \frac{1}{4}\sqrt{N}$. The complexity is $O(4\pi R^2) = O(N)$, with a constant factor reduction of $\frac{\pi}{4} \simeq 0.785$. Fig. A.14 shows N, $\mathcal{A}(U)$ and $4\pi R^2$ as a function of R. As R grows towards N,

$$r = \min\left(\frac{H}{2} - R, \frac{W}{2} - R\right) < R.$$

Thus, the *DSL* is selected based on the outer region, R_{out}, and $O(Rr) < O(R^2)$. This represents the worst scenario, where the object size nearly equals the image size. This is rare in practice. The main advantage of our approach is that it reduces the order of complexity to the size of the area enclosed by the user drawing $R^2 \ll N$. Thus, for a fixed foreground object size and a growing image size, the complexity is constant.

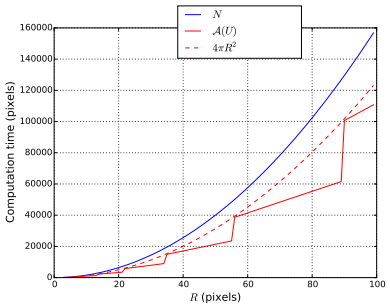


Figure A.14: Plot of N, $A(U)$, and $4\pi R^2$ as a function of the radius R in pixels. Note that $4\pi R^2 = O(A(U))$.

Appendix B. Pseudo-code for multi-resolution image generation

Algorithm 1 Multi-resolution super-pixels

Input: $(R_1, \ldots, R_i, \ldots R_N)$: N super-pixel images /* $R_i < R_{i+1}$ */
L: Hierarchical layer map /* see Fig. 1.b */
Output: M: Multi-resolution image

Initialize matrix M with -1
$labelCount \leftarrow 0$
$level \leftarrow DSL$ /* from Eq. 11 */
for $i \leftarrow 0$ **to** N **do**
  for each label $l \in R_i$ **do**
    /* all pixels beyond DSL are assigned to the lowest resolution */
    if $(\exists p \in l / L(p) \geq level)$ **and** $(i = 0)$ **then**
      $\forall p \in l / M(p) \leftarrow labelCount$
    end if
    /* pixels lying on level l are assigned to the same resolution */
    if $(\exists p \in l / L(p) = level)$ **then**
      $\forall p \in l / M(p) \leftarrow labelCount$
    end if
    $labelCount \leftarrow labelCount + 1$
  end for
  $level \leftarrow level - 1$
end for
/* assign the remainder pixels to pixel-resolution */
for $p \in M$ **do**
  if $M(p) = -1$ **then**
    $M(p) \leftarrow labelCount$
    $labelCount \leftarrow labelCount + 1$
  end if
end for

References

[1] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., 2012. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (11), 2274–2282.

[2] Boykov, Y., Jolly, M.-P., 2001. Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In: IEEE International Conference on Computer Vision. Vol. 1. pp. 105–112.

[3] Boykov, Y., Kolmogorov, V., 2004. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (9), 1124–1137.

[4] Brunner, E., Domhof, S., Langer, F., 2002. Nonparametric Analysis of Longitudinal Data in Factorial Experiments. Wiley series in probability and statistics, New York, NY, USA.

[5] Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., Moore, S., Phillips, S., Maffitt, D., Pringle, M., Tarbox, L., Prior, F., 2013. The cancer imaging archive (TCIA): maintaining and operating a public information repository. Journal of digital imaging 26 (6), 1045–1057.

[6] Conover, W. J., Johnson, M. E., Johnson, M. M., 1981. A Comparative Study of Tests for Homogeneity of Variances, with Applications to the Outer Continental Shelf Bidding Data. Technometrics 23 (4), 351–361.

[7] Couprie, C., Grady, L., Najman, L., Talbot, H., Sept 2009. Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In: IEEE International Conference on Computer Vision. pp. 731–738.

[8] Delong, A., Boykov, Y., 2008. A scalable graph-cut algorithm for n-d grids. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8.

[9] Galasso, F., Keuper, M., Brox, T., Schiele, B., 2014. Spectral graph reduction for efficient image and streaming video segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition.

[10] Gocławski, J., Węgliński, T., Fabijańska, A., 2015. Accelerating the 3D random walker image segmentation algorithm by image graph reduction and gpu computing. In: Image Processing & Communications Challenges 6. Vol. 313 of Advances in Intelligent Systems and Computing. pp. 45–52.

[11] Grady, L., 2006. Random walks for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11), 1768–1783.

[12] Grady, L., Schiwietz, T., Aharon, S., Westermann, R., 2005. Random walks for interactive organ segmentation in two and three dimensions: Implementation and validation. In: International Conference on Medical Image Computing and Computer Assisted Intervention. pp. 773–780.

[13] Grady, L., Sinop, A., 2008. Fast approximate random walker segmentation using eigenvector precomputation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8.

[14] Gueziri, H.-E., McGuffin, M., Laporte, C., 2015. User-guided graph reduction for fast image segmentation. In: IEEE International Conference on Image Processing. pp. 286–290.

[15] Gulshan, V., Rother, C., Criminisi, A., Blake, A., Zisserman, A., June 2010. Geodesic star convexity for interactive image segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3129–3136.

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

[17] Lermé, N., Malgouyres, F., Letocart, L., Sept 2010. Reducing graphs in graph cut segmentation. In: IEEE International Conference on Image Processing. pp. 3045–3048.

[18] Levinshtein, A., Stere, A., Kutulakos, K., Fleet, D., Dickinson, S., Siddiqi, K., 2009. TurboPixels: Fast Superpixels Using Geometric Flows. IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (12), 2290–2297.

[19] Li, Y., Sun, J., Tang, C.-K., Shum, H.-Y., 2004. Lazy snapping. ACM Transactions on Graphics 23 (3), 303–308.

[20] Lister Hill National Center for Biomedical Communications, 2004. Anat-Line. Accessed: May 2015.
URL `http://anatline.nlm.nih.gov/Anatline/index.html`

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

[22] Maurer, Jr., C. R., Rensheng, Q., Raghavan, V., 2003. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2), 265–270.

[23] McGuinness, K., O'Connor, N. E., 2010. A comparative evaluation of interactive segmentation algorithms. Pattern Recognition 43, 434–444.

[24] Mishra, A., Wong, A., Zhang, W., Clausi, D., Fieguth, P., Aug 2008. Improved interactive medical image segmentation using enhanced intelligent scissors (EIS). In: International Conference of the IEEE Engineering in Medicine and Biology Society. pp. 3083–3086.

[25] Mori, G., 2005. Guiding model search using segmentation. In: IEEE International Conference on Computer Vision. Vol. 2. pp. 1417–1423.

[26] Mortensen, E. N., Barrett, W. A., 1998. Interactive segmentation with intelligent scissors. Graphical Models and Image Processing 60 (5), 349–384.

[27] Munzel, U., Hothorn, L. A., 2001. A unified approach to simultaneous rank test procedures in the unbalanced one-way layout. Biometrical Journal 43 (5), 553–569.

[28] Olabarriaga, S., Smeulders, A., 2001. Interaction in the segmentation of medical images: A survey. Medical Image Analysis 5 (2), 127–142.

[29] Protiere, A., Sapiro, G., 2007. Interactive image segmentation via adaptive weighted distances. IEEE Transactions on Image Processing 16 (4), 1046–1057.

[30] Rother, C., Kolmogorov, V., Blake, A., 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics 23 (3), 309–314.

[31] Schneider, J., Kraus, M., Westermann, R., 2009. GPU-based real-time discrete euclidean distance transforms with precise error bounds. In: International Conference on Computer Vision Theory and Applications (VISAPP). pp. 435–442.

[32] Shen, J., Du, Y., Wang, W., Li, X., April 2014. Lazy random walks for superpixel segmentation. IEEE Transactions on Image Processing 23 (4), 1451–1462.

[33] Stakhov, A. P., 2009. The mathematics of harmony: from Euclid to contemporary mathematics and computer science. Vol. 22 of Series on Knots and Everything. World Scientific, Hackensack, NJ, USA.

[34] Yang, W., Cai, J., Zheng, J., Luo, J., 2010. User-friendly interactive image segmentation through unified combinatorial user inputs. IEEE Transactions on Image Processing 19 (9), 2470–2479.