Grabber: A Tool to Improve Convergence in Interactive Image Segmentation

Jordão Bragantini[b,*], Bruno Moura[a], Alexandre X. Falcão[b], Fábio A. M. Cappabianco[a]

[a]Group for Innovation Based on Images and Signals, Universidade Federal de São Paulo, Instituto de Ciência e Tecnologia, Av. Cesare Mansueto Giulio Lattes, 1201 - Eugênio de Mello, São José dos Campos, 12247-014, Brazil

[b]Laboratory of Image Data Science, University of Campinas, Institute of Computing, R. Saturnino de Brito, 573 - Cidade Universitária, Campinas, 13083-851, Brazil

Abstract

Interactive image segmentation has considerably evolved from techniques that do not learn the parameters of the model to methods that pre-train a model and adapt it from user inputs during the process. However, user control over segmentation still requires significant improvements to avoid that corrections in one part of the object cause errors in other parts. We address this problem by presenting *Grabber* — a tool to improve convergence (user control) in interactive image segmentation. Grabber is thought for integration with some other method. From a given segmentation mask, Grabber quickly estimates anchor points in one orientation along the boundary of the mask and delineates an optimum contour constrained to pass through those points. The user can control the process by adding, removing, and moving anchor points. Grabber can also incorporate extra object information from the main approach to improve boundary delineation. We integrate Grabber with two recent methods, a region-based approach and a pixel classification method based on deep neural networks. Extensive experiments with robot users on two datasets show in both cases that Grabber can significantly improve convergence, with faster delineation, higher effectiveness, and less user effort.

Keywords:
User Interaction in Image Segmentation, Contour-based Object Delineation, Image Foresting Transform, Deep Neural Networks

1. Introduction

Interactive image segmentation requires object localization, enhancement, delineation, and verification. Object localization may involve the indication of interior and exterior markers, points along the object boundary, a bounding box around the object, the information about its pose, and its approximate position in the image domain, for instance. Object enhancement improves the contrast between object and background to facilitate their separation (e.g., an object saliency map, a gradient image). Object delineation defines the precise spatial extension of the object in the image and it can usually improve when using a suitable object enhancement. Object verification may be done by the user or by optimizing some criterion function. As accurate segmentation rarely works from a single input action, the user usually verifies the result and takes actions for correcting errors, such that all previous steps can be improved along with multiple user interventions.

Object localization and verification are better performed by humans while machines usually outperform humans in object enhancement and delineation. An ideal method for interactive image segmentation should explore the complementary abilities of humans and machines to minimize user effort while maintaining complete user control over the segmentation process (Falcão et al.

*Corresponding author
Email address: jordao.bragantini@gmail.com (Jordão Bragantini)

(1998)). For that, the user's actions to correct segmentation should be effective, not causing errors in other parts of the object wherein the result is already correct.

Interactive image segmentation methods have considerably evolved in the last years. Initially, the methods did not learn the parameters of the segmentation model (Beucher (1979); Kass et al. (1988)). Many methods then started learning a model but fixing it during segmentation (Falcão et al. (1998); Cootes et al. (2001); Miranda et al. (2010); Xu et al. (2016); Maninis et al. (2018)). Other approaches can learn and update a model from user inputs during segmentation (Rother et al. (2004); Yang et al. (2010); Spina et al. (2012, 2014); Bragantini et al. (2018)) and several recent ones start from a pre-trained model that is adapted from user inputs during segmentation (Jang and Kim (2019); Sofiiuk et al. (2020); Kontogianni et al. (2019)).

However, while classical approaches, such as live-wire and live-lane (Falcão et al. (1998)), can provide considerable user control over the segmentation process at the cost of a higher user effort in more complex boundaries, the existing methods still lack user control during segmentation correction. See an example in Figures 1(a)-(d) for a recent method, named *feature Back-propagating Refinement Scheme* (fBRS), based on deep neural networks (Sofiiuk et al. (2020)).

In this paper, we address the above problem by presenting *Grabber* — a tool to improve convergence (user control) in interactive image segmentation. Grabber is inspired by the higher user control offered by live-wire (Falcão et al. (1998)), but it improves it in several aspects. First, Grabber can estimate anchor points from a segmentation mask and sort the points in one boundary orientation, rather than requiring the user to provide a sequence of anchor points in a given order along the boundary. This feature allows us to integrate Grabber with other automatic and interactive methods. Object delineation is obtained by an optimum contour constrained to pass through those anchor points, and Grabber can use extra object information from the main approach to further improve boundary delineation. Second, Grabber allows the user to control segmentation in any part of the boundary by adding, removing, and moving anchor points.

Although, it is not explored here, different definitions of connectivity strength between anchor points may be adopted in parts of the boundary or for distinct types of boundaries (Miranda et al. (2012); Barreto et al. (2016); Condori et al. (2017)). In order to demonstrate the impact of Grabber to increase convergence in interactive segmentation, we integrate it with two recent approaches, a pixel classification method based on deep neural networks, fBRS (Sofiiuk et al. (2020)), and a region-based method, named *Dynamic Trees* (DT, Bragantini et al. (2018); Falcão and Bragantini (2019)), that relies on combinatorial image analysis. Thus, Grabber is not meant to compete with other methods but to improve them, as soon as the user experiences difficulty in convergence or realizes that Grabber can complete segmentation faster than the main approach. Figures 1(e)-(f) illustrate its potential to improve interactive segmentation, when integrated with fBRS.

In summary, as main contributions, this paper presents a tool, named Grabber, to improve convergence in interactive segmentation, when integrated with other approaches; and two hybrid methods, fBRS-Grabber and DT-Grabber, that result from that integration.

2. Related Works

Methods for interactive image segmentation may be divided into connectivity-based and classification-based approaches, being connectivity-based methods further divided into region-based and contour-based techniques. In region-based approaches (Udupa and Saha (2003); Grady (2006); Boykov and Funka-Lea (2006); Couprie et al. (2010); Bragantini et al. (2018)), the user may select internal and/or external seeds (markers, strokes), such that the object pixels can be connected to their internal seeds by sequences of adjacent elements. Several approaches interpret an image as a graph, whose vertices are the pixels and edges are defined by adjacent elements, such that the object may result from an optimum graph cut with seeds as hard constraints (Boykov and Funka-Lea (2006)), from an optimum-path forest rooted at internal markers (Bragantini et al. (2018)), from a random walker starting from the same initial pixels (Grady (2006)), etc. In contour-based approaches, a contour model may be deformed to define an object's boundary (Kass et al. (1988); Cootes et al. (2001)) or the boundary may result from a sequence of optimum segments (Falcão et al. (1998); Miranda et al. (2012); Barreto et al. (2016); Condori et al. (2017)). There are also hybrid approaches that combine

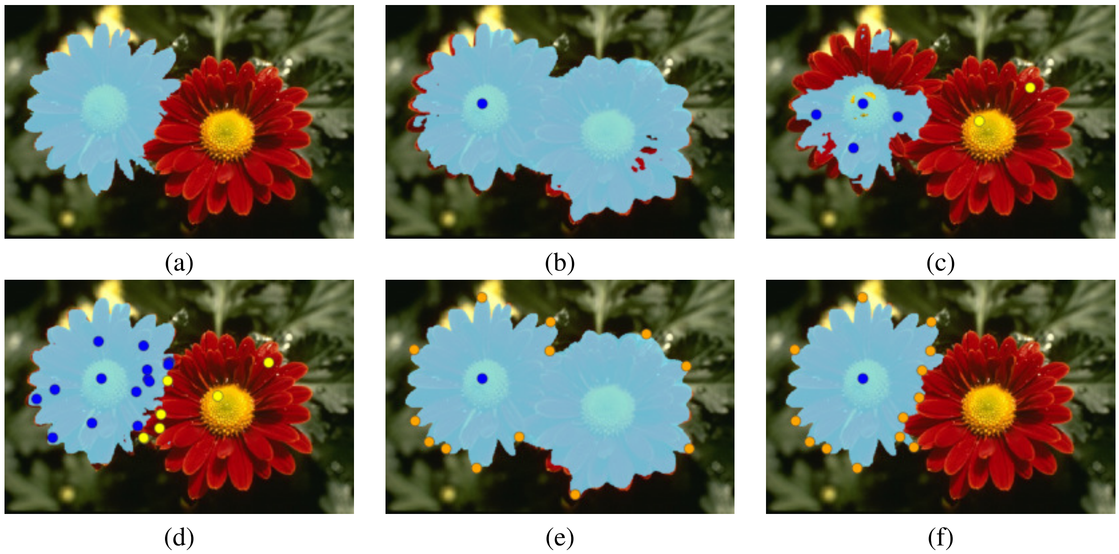


Figure 1: (a) Original image with ground-truth segmentation. (b), (c), (d) Results of fBRS (Sofiiuk et al. (2020)) after 1, 6, and 25 iterations, respectively, of an oracle user that inserts internal (blue) and external (yellow) points. One can see that corrections in (c) ruin correct parts of the result in (b). (e) Grabber can improve the result from iteration 1 of fBRS, delineating an optimum contour constrained to pass through estimated anchor points (orange). (f) By letting the user move, add, and remove anchor points, Grabber convergences faster with higher accuracy in 13 less user interventions.

advantages from contour-based and region-based methods (Yang et al. (2010); Spina et al. (2014)). In some approaches, object information is obtained from user-drawn markers (Miranda et al. (2010); Spina et al. (2012, 2014)) — a feature that relies on pixel classification to improve delineation when the image is interpreted as a graph. The classifier creates a *fuzzy object* (saliency) map, which enhances the edge weights that separate object and background.

Pixel classifiers are the core of classification-based methods. In this case, the object is usually defined regardless the connectivity among its pixels [1] and, due to that, several approaches adopt regularization procedures (Xu et al. (2017), Wang et al. (2019)), especially in automatic segmentation (Chen et al. (2017)). As a trend, these approaches normally use deep neural networks, pre-trained on a large set with previously segmented images, and let the user add markers (clicks) on new images, as constrains to correct segmentation (Xu et al. (2016); Maninis et al. (2018)). The object is usually defined by thresholding a fuzzy object map of the network, which explains the disconnected components in Figure 1(c). The most recent approaches can adapt the network from user inputs during segmentation (Jang and Kim (2019); Kontogianni et al. (2019); Sofiiuk et al. (2020)). In (Sofiiuk et al. (2020), the positions and distance maps of internal and external clicks are used to crop a region around the object and adapt the initial model for each specific segmentation task. This can improve the fuzzy object map but still does not solve the connectivity problem, as shown in Figure 1(c).

A main problem is the absence of user control during interactive segmentation (Figure 1). As a contour-based tool, Grabber can continue interactive segmentation from the mask obtained at any stage of another approach, with considerably higher user control, defining the object as a connected component, increasing effectiveness, and reducing user effort to complete the process, as we will show. It is worth noting that most contour-based methods cannot continue segmentation from another method (Falcão et al. (1998); Miranda et al. (2012);

[1] In some applications, such as land-use classification in remote sensing images, it is better to not force connectivity between markers and pixels, which favors the use of classification-based approaches.

Barreto et al. (2016); Condori et al. (2017)) and some provide very limited or no user control over delineation (Kass et al. (1988); Cootes et al. (2001)). User control is also addressed in hybrid connectivity-based approaches (Yang et al. (2010)) with different types of markers, but its effectiveness in boundary delineation with reduced user effort has not been validated.

3. Grabber

For a given segmentation mask, Grabber can estimate anchor points along the boundary of the mask and let the user manipulate them: add, remove, and move points. The initial anchor points are obtained in a desired order (clockwise or anti-clockwise) using the Douglas-Peucker algorithm (Douglas and Peucker (1973)) with a curvature-approximation threshold ϵ. It transforms the contour of the mask into line segments and uses their extreme points as anchors. Grabber then estimates the object boundary as an optimum contour constrained to pass through those anchor points. For that, Grabber uses the Image Foresting Transform (IFT) algorithm (Falcão et al. (2004)).

The image is interpreted as a graph $G = (V, E, I)$, where the set V of vertices contains the image pixels, $I : V \rightarrow \mathcal{R}^k$ is the image function that maps each vertex to a feature vector in $\mathcal{R}^k$ space (e.g., the L*a*b* color space), and the edge set E with pairs of adjacent pixels (e.g., 8-neighbors). A path π_u in G is a sequence of adjacent vertices with terminus u, being $\pi_u = \langle u \rangle$ a trivial path and $\pi_u \cdot \langle u, v\rangle$ the extension of π_u by an edge $\langle u, v\rangle$. For a given path-cost function $f(\pi_u)$, the IFT algorithm minimizes a path-cost map $C(u)$ by assigning to every $u \in V$ the cost of an optimum path with terminus u. In this process, it creates an optimum-path forest P — an acyclic map that assigns to every $u \in V$ a predecessor $P(u)$ in the optimum path $P^*(u)$ with terminus u or a marker $nil \notin V$, when u is a *root* (starting vertex) of the map. Note that the vertices of the path $P^*(u)$ can be accessed backwards by following the predecessors recursively until its root vertex.

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_n\} \subset V$ be the set of anchor points in a given order, such that $s_i \prec s_{i+1}$, $i = 1, 2, \ldots, n-1$. The IFT algorithm should compute one optimum contour $\mathcal{L}$ constrained to pass through the anchor points by following their order in $\mathcal{S}$. This will require n executions of the IFT algorithm to find minimum-cost segments from s_i to s_{i+1}, $i = 1, 2, \ldots, n-1$, and then from s_n to s_1. The set $\mathcal{L}$ is empty before the first execution. After each execution $1 \leq i < n$, $\mathcal{L}$ returns with the vertices of the optimum path $P^*(s_{i+1})$ with terminus s_{i+1} and root s_1. For the last execution n, s_1 is removed from $\mathcal{L}$ in order to close the contour as an optimum path $P^*(s_1)$ with root in a vertex s whose $P(s) = s_1$. The path-cost function f may be defined as

$$
\begin{aligned}
f(\langle u \rangle) &= \begin{cases} 0 & \text{if } u = s_1 \text{ and } \mathcal{L} = \emptyset, \\ +\infty & \text{if } u \in V \setminus \mathcal{L}, \end{cases} \\
f(\pi_u \cdot \langle u, v \rangle) &= f(\pi_u) + w_I(u, v), \qquad (1) \\
w_I(u, v) &= \exp\left(-\frac{\|I(l_{u,v}), I(r_{u,v})\|_2}{\sigma_I}\right) \qquad (2)
\end{aligned}
$$

where $\|., .\|_2$ is the Euclidean norm, $l_{u,v}$ and $r_{u,v}$ are the left and right vertices of an edge $(u, v) \in E$, respectively, and $\sigma_I > 0$ is a constant. It is expected lower values of $w_I(u, v)$ on the object's border than in the immediate neighborhood inside and outside it. The parameter σ_I controls the importance of color differences between the left and right sides of the edges.

Algorithm 1 presents the contour-based object delineation of Grabber. In Lines 1-2, it executes the trivial path-cost initialization of Equation 1, being all paths initially trivial. The main loop (Lines 3-5) computes the minimum-cost segments from s_i to s_{i+1}, $i = 1, 2, \ldots, n-1$. It then removes s_1 from $\mathcal{L}$ in Line 6, updating its trivial path cost according to Equation 1, to close the contour in Line 7. Every time an optimum segment is computed, it is added to the contour in $\mathcal{L}$.

Algorithm 1. – Contour-based segmentation algorithm

Input:	Graph $G = (V, E, I)$ and anchor-point set $\mathcal{S}$.
Output:	Optimum object contour $\mathcal{L}$.
Auxiliary:	Path-cost map C and predecessor map P.

1. ***For each*** $u \in V$ ***do*** $C(u) \leftarrow +\infty$ *and* $P(u) \leftarrow u$.
2. $\mathcal{L} \leftarrow \emptyset$, $i \leftarrow 1$, *and* $C(s_1) \leftarrow 0$.
3. ***While*** $i < n$ ***do***
4. $\quad (C, P, \mathcal{L}) \leftarrow$ *Optimum-Segment*$(G, s_i, s_{i+1}, C, P, \mathcal{L})$.
5. $\quad i \leftarrow i + 1$
6. $\mathcal{L} \leftarrow \mathcal{L} \setminus \{s_1\}$ *and* $C(s_1) \leftarrow +\infty$.
7. $(C, P, \mathcal{L}) \leftarrow$ *Optimum-Segment*$(G, s_n, s_1, C, P, \mathcal{L})$
8. *Return* $\mathcal{L}$.

Algorithm 2 shows how to compute and concatenate optimum segments in $\mathcal{L}$. Line 1 inserts the source s in a priority queue $\mathcal{Q}$ and in an auxiliary set $\mathcal{U}$. Set $\mathcal{U}$

must contain all vertices inserted in Q during the algorithm in order to restore the costs of trivial paths, according to Equation 1, for the subsequent execution. Note that the source s is always inserted in Q with optimum cost $C(P^*(s))$, as obtained in the previous execution. The optimum segments already in $\mathcal{L}$ have lower costs than that and the remaining vertices are available to be conquered by a new segment from s. The main loop (Lines 2-17) removes the vertex u with minimum path cost from Q in Line 3. If u is the destination point t (Line 4), the algorithm concatenates the previous segments to the optimum segment from s to t (Lines 5-7), restores the trivial paths and their costs according to Equation 1 (Lines 8-9), and resets Q and $\mathcal{U}$ to end the loop. While u is not the destination point, the algorithm visits its adjacent vertices (Lines 12-17) to which the extended paths $\pi_u \cdot \langle u, v\rangle$ with cost $tmp = f(\pi_u \cdot \langle u, v\rangle)$ (Line 13) may be lower than the cost $C(v) = f(\pi_v)$ of the current path π_v. If this is the case, then path π_v is replaced by $\pi_u \cdot \langle u, v\rangle$ when $P(v)$ is set to u in Line 16. The cost $C(v)$ and status of v in Q and $\mathcal{U}$ are updated accordingly (Lines 15-17).

Algorithm 2. – Optimum-segment finding algorithm

Input: Graph $G = (V, E, I)$, source $s \in V$, destination $t \in V$, previous path-cost map C, predecessor map P, and contour set $\mathcal{L}$.
Output: Updated path-cost map C, predecessor map P, and contour set $\mathcal{L}$.
Auxiliary: Priority queue $Q = \emptyset$ and set $\mathcal{U} = \emptyset$.

1. $Q \leftarrow Q \cup \{s\}$, *and* $\mathcal{U} \leftarrow \mathcal{U} \cup \{s\}$.
2. ***While*** $Q \neq 0$ ***do***
3. $\quad Q \leftarrow Q \setminus \{u\}$ *such that* $u = arg\min_{v \in Q}\{C(v)\}$.
4. $\quad$ ***If*** $u = t$ ***then***
5. $\quad\quad$ ***While*** $u \neq s$ ***do***
6. $\quad\quad\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{u\}$ *and* $u \leftarrow P(u)$.
7. $\quad\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{s\}$.
8. $\quad\quad$ ***For each*** $u \in \mathcal{U} \setminus \mathcal{L}$ ***do***
9. $\quad\quad\quad C(u) \leftarrow +\infty$ *and* $P(u) \leftarrow nil$.
10. $\quad\quad Q \leftarrow \emptyset$ *and* $\mathcal{L} \leftarrow \emptyset$
11. $\quad$ ***Else***
12. $\quad\quad$ ***For each*** $v \mid (u, v) \in E$ *and* $C(v) > C(u)$ ***do***
13. $\quad\quad\quad tmp \leftarrow C(u) + w_I(u, v)$
14. $\quad\quad\quad$ ***If*** $tmp < C(v)$ ***then***
15. $\quad\quad\quad\quad$ ***If*** $v \in Q$ ***then*** $Q \leftarrow Q \setminus \{v\}$.
16. $\quad\quad\quad\quad C(v) \leftarrow tmp,\ P(v) \leftarrow u,$
17. $\quad\quad\quad\quad Q \leftarrow Q \cup \{v\}$, *and* $\mathcal{U} \leftarrow \mathcal{U} \cup \{v\}$.
18. *Return* $(C, P, \mathcal{L})$.

4. Integration and Usage of Grabber

As a contour-based tool, Grabber can be integrated with other methods to speed up convergence. The idea is to start interactive segmentation with one main approach and use Grabber to conclude the process by the time the user observes that either the main method is requiring much user effort to achieve the desired result or Grabber could conclude the process faster from the current output of the main approach. This integration also allows us to explore object information from the main approach, making Grabber more effective.

Figure 2 illustrates one example where the user draws internal and external markers (Figure 2(b)) to separate the object in Figure 2(a) from the background by using *Dynamic Trees* (DT, Bragantini et al. (2018); Falcão and Bragantini (2019)). DT is a region-based method that can compute optimum-path forests rooted at markers in multiple objects, such that each object is defined by the forest of its internal markers. The main difference between DT and other IFT-based methods (e.g., a watershed transform) is that the edge weights of its path-cost function are computed on-the-fly, based on mid-level properties of the growing trees. Its advantages over region-based and classification-based methods have been demonstrated in (Bragantini et al. (2018); Falcão and Bragantini (2019)). We demonstrate in Section 5 that the integration of Grabber and DT can improve convergence. Figure 2(c) shows the result of Grabber when the algorithms of Section 3 are used as explained. The segmentation mask, however, allows to compute Gaussian Mixture Models (GMMs) from the interior (Figure 2(d)) and exterior (Figure 2(e)) of the mask, and use this information to replace the edge-weight function $w_I(u, v)$ in Equation 2 by the one below.

$$w_G(u, v) = w_I(u, v) \exp\left(-\frac{\sum_{\lambda \in \{0,1\}} |D_\lambda(r_{u,v}) - D_\lambda(l_{u,v})|}{\sigma_G}\right) \tag{3}$$

where $D_\lambda(.)$ are the Gaussian density maps for each label $\lambda \in \{0, 1\}$ (i.e., background and object), and $\sigma_G > 0$ is a constant. Figure 2e shows the final result of Grabber using Equation 4 and after manipulating anchor points. The density maps can be updated after each iteration of Grabber. The choices of σ_I in w_I and σ_G are important to control the balance between local image and object information.

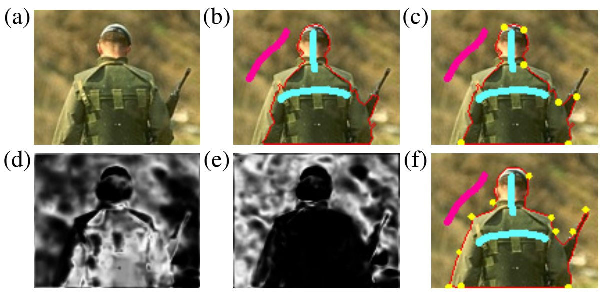


Figure 2: Grabber: (a) input image; (b) initial segmentation with DT (Bragantini et al. (2018)) from object (blue) and background (pink) markers; (c) the resulting object contour from Grabber; Gaussian density maps for the (d) object and (e) background; (f) final segmentation with Grabber, after anchor-point manipulation.

The initial number of anchor points in Grabber is still an empirical parameter, but one can explore gradient and salience information along the border of the mask to estimate more effective points. We are choosing points based on a simplification degree of the original contour. When the user clicks on one anchor point v_k, the paths from the previous point v_{k-1} and to the next point v_{k+1} are redefined by running Algorithm 2 on the affected segments (Figure 3). Therefore, Grabber can limit user interaction to a segment of the object border providing the desired control over the correction process.

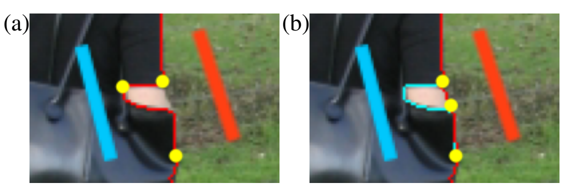


Figure 3: Contour-based correction of a border section. (a) Initial segmentation with DT (Bragantini et al. (2018)) followed by Grabber. (b) The middle anchor point is moved to the object's border and Grabber fixes delineation in red. The previous segment (error) is shown in cyan.

When a new anchor point is added by the user, the contour is not immediately redefined since the user might want to move the point to a better location. A good practice is showed in Figure 4. If the user wants to improve a contour section and keep the rest unchanged, it is best to add points at the extremities of that section and then make further point manipulation inside the section.

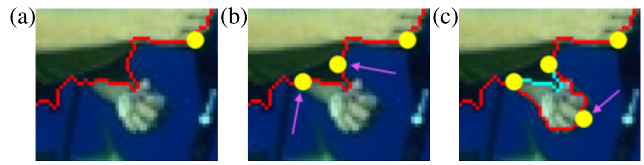


Figure 4: A good practice in Grabber: (a) initial contour for correction; (b) points are added to both extremities of a contour section; (c) a new point is added in that section and moved to a better location; (d) the resulting correction in red with the previous wrong segment in cyan. Note that the rest of the contour remains unchanged.

Object information may be obtained by different strategies. In Spina et al. (2012), the authors present an intelligent approach to select the pixels from internal and external markers that best discriminate the object and background for image enhancement as an object saliency map. Live Markers (Spina et al. (2014)) uses this method for edge weight estimation in the IFT algorithm. It also allows the user to draw live-wire segments on the border of the object, creating internal and external markers around them, but the segmentation is always a watershed transform from those markers. Another possibility is to use the object map that is improved along with internal and external clicks during segmentation by fBRS (Sofiiuk et al. (2020)). Figure 5 illustrates the object maps obtained by the method in (Spina et al. (2012)), when running Live Markers, by GM models from the segmentation mask of Live Markers (map D_1 in Equation 4), and by the network in fBRS. In general, Grabber can improve contour-based delineation when it incorporates object information in edge-weight estimation. Assuming that $O(u)$ is the object map value from some of these approaches, Equation 2 may be replaced by

$$w_C(u, v) = w_I(u, v) \exp\left(-\frac{|O(r_{u,v}) - O(l_{u,v})|}{\sigma_O}\right) \quad (4)$$

5. Results and Discussion

This section evaluates Grabber associated with two methods, fBRS (Sofiiuk et al. (2020)) and DT (Bragantini et al. (2018)). The resulting approaches are called fBRS-Grabber and DT-Grabber. We indicate the edge weight

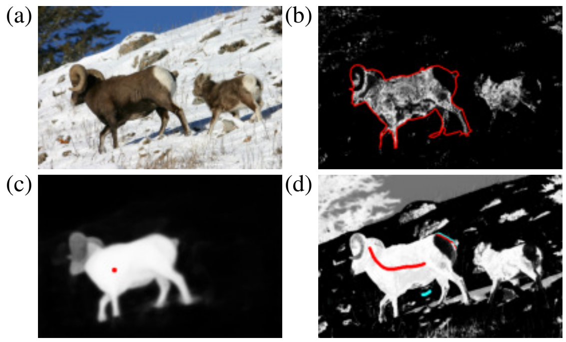


Figure 5: (a) Original image. Object saliency maps by (b) GM model from a large sheep mask (red boundary), (c) the network in fBRS from a single click (red dot), and (d) the method in Spina et al. (2012) from background (cyan) and foreground (red) markers.

function used in each case — e.g., DT-Grabber-w_I when Grabber uses Equation 2.

The experiments used an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz CPU and a Titan X with 12 GB of memory. We adopted a stress experiment similar to the convergence analysis from Sofiiuk et al. (2020). It measures the number of interactions required to achieve a fixed threshold of Intersection over Union (IoU) (Xu et al. (2016); Maninis et al. (2018); Jang and Kim (2019); Sofiiuk et al. (2020)) — i.e., the number of clicks/anchor manipulations to achieve 0.95 IoU is denoted by NoC@0.95 and we count the number of images to which the methods could not achieve 0.95 IoU in 50 interactions. The experiments used the testing set of Berkeley (Martin et al. (2001)) from McGuinness and O'connor (2010) and a subset provided by Jang and Kim (2019), with 10 percent of the images (video frames) from DAVIS (Perazzi et al. (2016)). The ground-truths of the datasets where pre-processed, removing disconnected objects and holes to facilitate the experiments with a proposed robot for Grabber (Section 5.1).

ResNet101, the network used in fBRS, was pre-trained on SBD (Hariharan et al. (2011)), and it is used as provided by Sofiiuk et al. (2020). The robot user for fBRS and DT is the same (Xu et al. (2016)). As it adds internal and external clicks, fBRS can crop a region around the object and use distance maps of the clicks to improve an object saliency map, solving segmentation by thresholding the map. In DT, the clicks define the object as an optimum-path forest rooted at the internal ones.

We switched from fBRS (or DT) to Grabber when the IoU changed less than 1% for three consecutive interactions and Grabber started without resetting the number of interactions. Its parameters were optimized by grid-search on 150 random images from SBD (Hariharan et al. (2011)), resulting $\epsilon = 100$, $\sigma_I = 0.5$, and $\sigma_O = 0.25$.

5.1. Proposed Robot User for Grabber

Grabber's robot starts locating a reference pixel for each anchor. The reference pixel is the closest one to the anchor on the ground-truth border. It then locates the closest point to each reference pixel on the segmentation's border. If an anchor is not the closest point to its own reference pixel, the robot removes that anchor (Figure 6(a)-(b)). This behavior imitates the action of a user removing anchors from regions where Grabber should update the contour without adding new anchors. The distance is the length of a geodesic path in the component between the GT and the segmentation mask, thus avoiding tangling the contours in non-convex shapes.

After that, the robot lists all pixels at which the GT and segmentation overlap. For every pair of consecutive intersection points between the overlapping GT and segmentation borders, it computes the area between the borders and try to correct these regions in a decreasing order of area.

Given the current largest and incorrect region, the robot adds a pair of anchors on its extremities to limit the changes to the local contour, if needed, just as the strategy shown in Figure 4. This is a key factor to maintain user control over the segmentation process. Then, the robot moves the furthest anchor from its reference pixel onto the GT border (Figure 6(b)-(c) and (d)-(e)). If there are no anchors on the segment, the robot adds one anchor and moves it (Figure 6(c)-(d)).

To better emulate a real user, the anchor target is around the middle of the GT segment on the pixel with the weakest gradient. Thus, imposing a constraint where the difference between foreground and background could be nonexistent, Figure 6(d).

5.2. Results

Tables 1-3 show that the integration of fBRS and DT with Grabber can improve convergence and, at the same

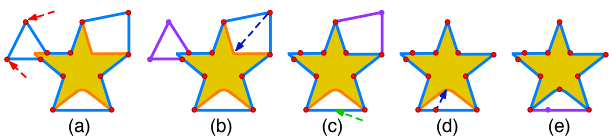


Figure 6: Example of robot iterations. The blue, orange and purple lines represent current, desired, and removed contour segments, respectively, and the red dots are anchors. Red, blue, and green arrows point to positions where anchors should be deleted, moved, or added. (a) Initial segmentation; (b) after deleting two anchors; (c) after moving an anchor; (d) after adding an anchor; (e) after moving the anchor added in the previous iteration.

time, reduce the average response time of the interactions, and increase the mean IoU. Table 1 shows that Grabber can considerably reduce to almost half the number of images to which the main methods could not achieve 0.95 IoU in 50 interactions. It can also reduce the average number of interactions to achieve 0.95 IoU. Grabber was more required for DT than for fBRS, which was expected since DT is simpler and this version does not use object saliency map learned from long strokes (Falcão and Bragantini (2019)). However, fBRS-Grabber-w_C performed worse than fBRS-Grabber-w_I in DAVIS and did not make much difference in Berkeley with respect to fBRS-Grabber-w_I.

Method	Dataset	# Images $\geq$ 50	NoC@0.95	Grabber (%)
fBRS	Berkeley	23	16.77	-
fBRS-Grabber-w_I	Berkeley	**12**	14.53	42.0
fBRS-Grabber-w_C	Berkeley	**12**	14.02	43.0
DT	Berkeley	31	27.71	-
DT-Grabber-w_I	Berkeley	**22**	26.77	71.0
fBRS	DAVIS	133	24.83	-
fBRS-Grabber-w_I	DAVIS	**93**	24.49	65.8
fBRS-Grabber-w_C	DAVIS	100	24.74	65.8
DT	DAVIS	163	39.07	-
DT-Grabber-w_I	DAVIS	**139**	37.85	91.6

Table 1: For each method and dataset, the number of images to which the method could not achieve 0.95 IoU in 50 interactions (bold is best), average NoC@0.95, and percentage of images that required Grabber.

Table 2 shows that Grabber can reduce the average response time for the interactions in both methods. Indeed, when fBRS stops, the response time of Grabber is negligible (real time). One may have an idea about that by observing the response times of DT-Grabber-w_I.

Table 3 shows that Grabber can increase effectiveness of the main approach, especially in Berkeley. DAVIS is a more challenging dataset and the proposed robot for

Method	Dataset	Time per Inter.	Dataset	Time per Inter.
fBRS-Grabber-w_I	Berkeley	0.218	DAVIS	0.457
fBRS-Grabber-w_C	Berkeley	0.225	DAVIS	0.458
fBRS	Berkeley	0.738	DAVIS	1.798
DT	Berkeley	0.111	DAVIS	0.287
DT-Grabber-w_I	Berkeley	0.078	DAVIS	0.200

Table 2: For each method and dataset, the average response time in seconds per interaction.

Grabber requires improvements. It is also worth noting that the IoU of fBRS may be 5% overestimated since the same robot is used for training and testing (Benenson et al. (2019)).

Method	Dataset	IoU	Dataset	IoU
fBRS	Berkeley	0.938 ± 0.023	DAVIS	0.915 ± 0.071
fBRS-Grabber-w_I	Berkeley	0.946 ± 0.014	DAVIS	0.916 ± 0.122
fBRS-Grabber-w_C	Berkeley	0.946 ± 0.012	DAVIS	0.922 ± 0.105
DT	Berkeley	0.925 ± 0.077	DAVIS	0.913 ± 0.098
DT-Grabber-w_I	Berkeley	0.941 ± 0.027	DAVIS	0.910 ± 0.125

Table 3: For each method and dataset, the mean and standard deviation of IoU over the image cases that required Grabber.

Finally, Figures 7(a)-(b) show that the Grabber's contour has higher adherence to the object's border than the segmentation of fBRS, and Figures 7(c)-(d) show the same for Grabber-DT.

5.3. Discussion

The experiments have been enough to demonstrate the advantages of integrating Grabber with fBRS and DT. Similar results should be observed with other methods. We expected better results with fBRS-Grabber-w_C than with fBRS-Grabber-w_I, because the use of object saliency maps can usually improve segmentation, as observed in Falcão and Bragantini (2019). It is possible that our decision to change from fBRS to Grabber was premature and Grabber could not benefit of the improvements in the object saliency map of fBRS.

Figures 8 (a)-(f) show the progress of the object saliency map along with a few interations in fBRS. Note in Figure 8(c) that the map of fBRS is biased to enhance people. Due to crop and optimization based on the distance maps of the clicks, fBRS indeed improves the map for the object of interest — the shirt of a person. Nevertheless, even the initial map in Figure 8(c) affects Grabber positively since its results are better with than without the map, for distinct values of ϵ (Figures 8(g)-(l)). This example also shows that a higher value of ϵ would improve convergence in fBRS-Grabber-w_C and fBRS-Grabber-w_I, when compared to the best value of ϵ estimated in SBD.

In summary, the user experience can be considerably improved when using Grabber to speed up convergence in interactive segmentation. The decision of when to switch from the main method to Grabber, its parameters, and anchor-point manipulation should be better managed by the user, upon the visual analysis of the process, edge weights, object maps, etc.

6. Conclusions

We presented Grabber, a tool to improve convergence in interactive segmentation by integration with any other method. We evaluated fBRS-Grabber and DT-Grabber, showing improvements in user control, effectiveness, and efficiency. We intend now to investigate saliency map improvement from user markers for Grabber segmentation (Spina et al. (2012)).

Acknowledgments

The authors thank CNPq (303808/2018-7), FAPESP (2014/12236-1, 2019/11349-0, 2016/21591-5) and Sofiiuk et al. (2020) for making their implementation publicly available.

References

Barreto, T.L.M., Almeida, J., Cappabianco, F.A.M., 2016. Estimating accurate water levels for rivers and reservoirs by using sar products: A multitemporal analysis. Pattern Recognition Letters 83, 224–233.

Benenson, R., Popov, S., Ferrari, V., 2019. Large-scale interactive object segmentation with human annotators, in: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 11700–11709.

Beucher, S., 1979. Use of watersheds in contour detection, in: Proceedings of the Int. Workshop on Image Processing, pp. 17–21.

Boykov, Y., Funka-Lea, G., 2006. Graph cuts and efficient nd image segmentation. Int. Journal of Computer Vision 70, 109–131.

Bragantini, J., Martins, S.B., Castelo-Fernandez, C., Falcão, A.X., 2018. Graph-based image segmentation using dynamic trees, in: Iberoamerican Congress on Pattern Recognition, pp. 470–478.

Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. on Pattern Analysis and Machine Intelligence 40, 834–848.

Condori, M.A., Mansilla, L.A., Miranda, P.A., 2017. Bandeirantes: A graph-based approach for curve tracing and boundary tracking, in: Int. Symp. on Math. Morph. and Its App. to Signal and Image Processing, pp. 95–106.

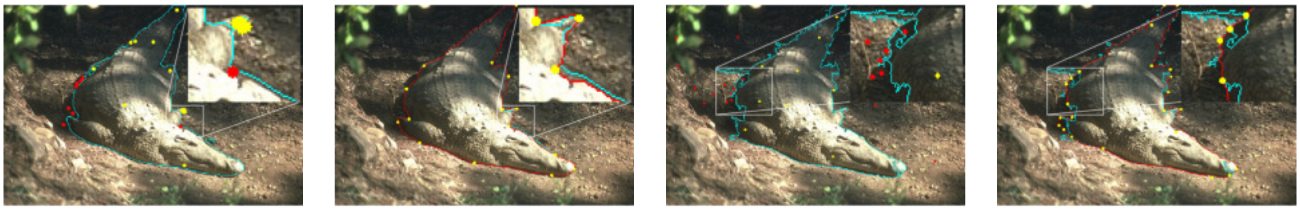
Cootes, T.F., Edwards, G.J., Taylor, C.J., 2001. Active appearance models. IEEE Trans. on Pattern Analysis and Machine Intelligence , 681–685.

Couprie, C., Grady, L., Najman, L., Talbot, H., 2010. Power watershed: A unifying graph-based optimization framework. IEEE Trans. on Pattern Analysis and Machine Intelligence 33, 1384–1399.

Douglas, D.H., Peucker, T.K., 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: the Int. Journal for Geographic Info. and Geovisualization 10, 112–122.

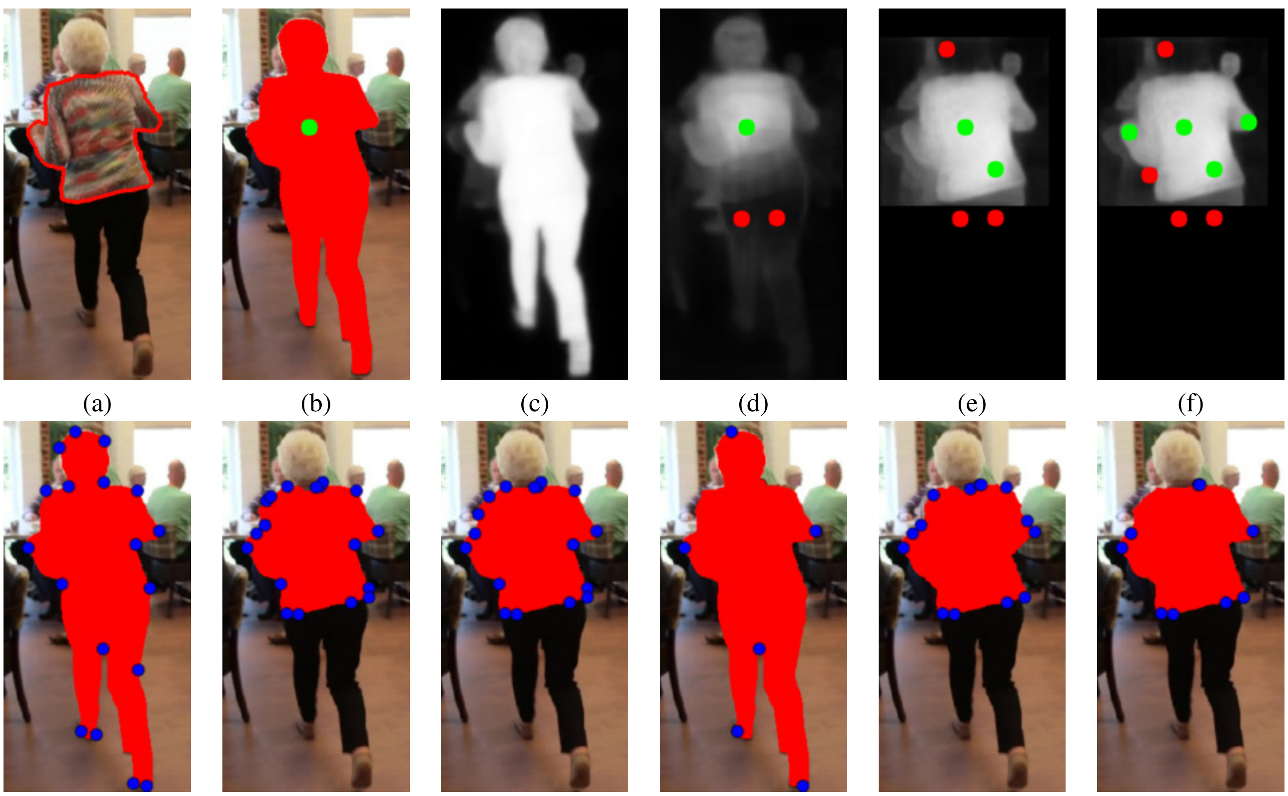
Falcão, A., Bragantini, J., 2019. The role of optimum connectivity in image segmentation: Can the algorithm learn object information during the process?, in: Int. Conf. on Discrete Geometry for Computer Imagery, pp. 180–194.

Falcão, A.X., Stolfi, J., de Alencar Lotufo, R., 2004. The image foresting transform: Theory, algorithms, and applications. IEEE Trans. on Pattern Analysis and Machine Intelligence 26, 19–29.



(a) (b) (c) (d)

Figure 7: Crocodile image segmented with (a) fBRS (cyan contour) from internal (yellow) and external (red) clicks. Grabber provides (b) higher boundary adherence with less effort (red contour, previous contour in cyan). (c) Segmentation with DT from object (yellow) and background (red) markers. (d) Correction with Grabber (red contour, previous contour in cyan).



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Figure 8: (a) Original image with ground-truth contour. (b) fBRS's segmentation with one click and (c) its saliency map biased to people. (d-f) Saliency map progress for additional clicks. (g) Grabber's anchors (blue), for $\epsilon = 100$, and its results (h) without and (i) with (1 less interaction) the saliency map in (c). (j) Grabber's anchors (blue), for $\epsilon = 1000$, and its results (k) without and (l) with (5 less interactions) the saliency map in (c).

Falcão, A.X., Udupa, J.K., Samarasekera, S., Sharma, S., Hirsch, B.E., Lotufo, R.d.A., 1998. User-steered image segmentation paradigms: Live wire and live lane. Graphical models and image processing 60, 233–260.

Grady, L., 2006. Random walks for image segmentation. J_PAMI , 1768–1783.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J., 2011. Semantic contours from inverse detectors, in: 2011 International Conference on Computer Vision, IEEE. pp. 991–998.

Jang, W.D., Kim, C.S., 2019. Interactive image segmentation via backpropagating refinement scheme, in: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 5297–5306.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. Int. Journal of Computer Vision 1, 321–331.

Kontogianni, T., Gygli, M., Uijlings, J., Ferrari, V., 2019. Continuous adaptation for interactive object segmentation by learning from corrections. arXiv preprint arXiv:1911.12709 .

Maninis, K.K., Caelles, S., Pont-Tuset, J., Van Gool, L., 2018. Deep extreme cut: From extreme points to object segmentation, in: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 616–625.

Martin, D., Fowlkes, C., Tal, D., Malik, J., 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: IEEE Int. Conf. on Computer Vision, IEEE. pp. 416–423.

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

Miranda, P.A., Falcão, A.X., Spina, T.V., 2012. Riverbed: A novel user-steered image segmentation method based on optimum boundary tracking. IEEE Trans. on Image Processing 21, 3042–3052.

Miranda, P.A., Falcão, A.X., Udupa, J.K., 2010. Synergistic arc-weight estimation for interactive image segmentation using graphs. Computer Vision and Image Understanding 114, 85 – 99. doi:10.1016/j.cviu.2009.08.001.

Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A., 2016. A benchmark dataset and evaluation methodology for video object segmentation, in: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 724–732.

Rother, C., Kolmogorov, V., Blake, A., 2004. Grabcut: Interactive foreground extraction using iterated graph cuts, in: C_ACM, pp. 309–314.

Sofiiuk, K., Petrov, I., Barinova, O., Konushin, A., 2020. f-brs: Rethinking backpropagating refinement for interactive segmentation. arXiv preprint arXiv:2001.10331 .

Spina, T., De Miranda, P.A.V., Falcão, A.X., 2012. Intelligent understanding of user interaction in image segmentation. International Journal of Pattern Recognition and Artificial Intelligence 26, 1265001. doi:10.1142/S0218001412650016.

Spina, T., de Miranda, P., Falcão, A.X., 2014. Hybrid approaches for interactive image segmentation using the live markers paradigm. IEEE Trans. on Image Processing 23, 5756–5769.

Udupa, J.K., Saha, P.K., 2003. Fuzzy connectedness and image segmentation. Proceedings of the IEEE 91, 1649–1669.

Wang, Z., Acuna, D., Ling, H., Kar, A., Fidler, S., 2019. Object instance annotation with deep extreme level set evolution, in: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 7500–7508.

Xu, N., Price, B., Cohen, S., Yang, J., Huang, T.S., 2016. Deep interactive object selection, in: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 373–381.

Xu, N., Price, B., Cohen, S., Yang, J., Huang, T.S., 2017. Deep grabcut for object selection, in: 28th British Machine Vision Conference, BMVC 2017, BMVA Press.

Yang, W., Cai, J., Zheng, J., Luo, J., 2010. User-friendly interactive image segmentation through unified combinatorial user inputs. IEEE Trans. on Image Processing 19, 2470–2479.