Graph-Based Image Segmentation Using Dynamic Trees

Jordão Bragantini(✉), Samuel Botter Martins, Cesar Castelo-Fernandez, and Alexandre Xavier Falcão

Laboratory of Image Data Science, University of Campinas, Campinas, Brazil
{jordao.bragantini,samuel,cesar.castelo,afalcao}@lids.ic.unicamp.br
https://lids.ic.unicamp.br

Abstract. Image segmentation methods have been actively investigated, being the graph-based approaches among the most popular for object delineation from seed nodes. In this context, one can design segmentation methods by distinct choices of the image graph and *connectivity function*—i.e., a function that measures how strongly connected are seed and node through a given path. The framework is known as *Image Foresting Transform* (IFT) and it can define by seed competition each object as one *optimum-path forest* rooted in its internal seeds. In this work, we extend the general IFT algorithm to extract object information as the trees evolve from the seed set and use that information to estimate arc weights, positively affecting the connectivity function, during segmentation. The new framework is named *Dynamic IFT* (DynIFT) and it can make object delineation more effective by exploiting color, texture, and shape information from those dynamic trees. In comparison with other graph-based approaches from the state-of-the-art, the experimental results on natural images show that DynIFT-based object delineation methods can be significantly more accurate.

Keywords: Image Foresting Transform · Multiple object delineation · Graph Cut · Image segmentation by seed competition

1 Introduction

Image segmentation is a challenging task that often requires user's assistance for corrections [11]. Deep neural networks can provide impressive object approximations [10], but object delineation is still not accurate, even when the user provides careful object localization [12] (Fig. 1). On the other hand, the combination of interactive object localization and graph-based object delineation may solve the problem in a few iterations of corrections with simple user effort (Fig. 1c).

The authors thank CNPq (302970/2014-2 grant), FAPESP (2014/12236-1, 2017/03940-5 and 2018/08951-8 grants) and the Department of Statistics of the University of Campinas for the financial support.

R. Vera-Rodriguez et al. (Eds.): CIARP 2018, LNCS 11401, pp. 470–478, 2019.
https://doi.org/10.1007/978-3-030-13469-3_55

Among many interesting approaches, graph-based object delineation has become quite popular with methods based on Random Walks [8], Graph Cuts [2], Watershed Cuts [6], and Image Foresting Transform (IFT) [7]. These frameworks interpret an image as a graph and, often from some hard constraints (e.g., seed nodes that were chosen by the user to locate the objects), the methods delineate the objects by optimizing some energy function [3,13].

In this work, we explore the optimum-path trees that dynamically evolve from seed nodes during the IFT algorithm for more effective object delineation. This defines a new framework, named *Dynamic* IFT (DynIFT), with methods that can estimate the arc weights in the graph during object delineation by exploiting the object knowledge that increases at each moment. The methods are compared with state-of-the-art graph-based delineation approaches [2,3,6] using a natural image dataset with two types of seed sets provided by users. The experimental results using color information only already show considerable effectiveness gains in object delineation using the DynIFT algorithm.

The next sections present the proposed framework, with algorithm and examples of dynamic arc-weight estimation methods, the experimental results, discussion, and conclusion.

(a) (b) (c)

Fig. 1. (a) Original image with four extreme points (magenta) for the method in [12] and orange and green markers for the proposed algorithm. (b) Result of the method in [12] with errors and (c) the desired segmentation using the proposed algorithm. (Color figure online)

2 Dynamic Image Foresting Transform

A 2D image is a pair $(D_I, \mathbf{I})$ in which $\mathbf{I}(p)$ assigns a set of local image features (e.g., color components) to each pixel $p \in D_I \subset \mathcal{Z}^2$. An image may be interpreted as a graph $(\mathcal{N}, \mathcal{A})$ in various distinct ways by defining the nodes in $\mathcal{N} \subseteq D_I$, for example, as pixels, superpixels, or pixel vertices, and using some *adjacency relation* $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$ in the image domain and/or feature space to define the arcs. For the sake of simplicity, we focus on pixels as nodes ($\mathcal{N} = D_I$), with $\mathbf{I}(\mathbf{p})$ being the CIELab color components of pixel p, and the 4-neighborhood relation to define the arcs.

For a given seed set $\mathcal{S}$—e.g., labeled scribbles (markers) drawn by the user in each object (including background) for object localization and/or segmentation correction—we wish to partition the image into objects such that each

Fig. 2. (a) Original image with markers (yellow and blue circles) drawn by the user. (b–e) Dynamic tree evolution in some iterations (each color being one different tree). Notice that each marker has multiple trees (one for each root pixel), but some roots may conquer most pixels in the region dominated by the marker. (Color figure online)

object consists of the pixels more closely connected to its internal seeds than to any other. Each seed $p \in \mathcal{S}$ is then uniquely identified as belonging to one among c objects by a labeling function $\lambda_O(p) \in \{0, 1, 2, \ldots, c\}$, being 0 the background. Similarly, one can also identify the marker $\lambda_M(p) \in \{1, 2, \ldots, m\}$ among m markers that contains the seed p. This can be used, for instance, to control marker deletion and addition during segmentation correction. Therefore, a *connectivity function* f measures how closely connected are seed and node through any given path in the image graph from the former to the latter. A path $\pi_q = \langle p_1, p_2, \ldots, p_n = q\rangle$ with terminus q is a sequence of nodes, such that $(p_i, p_{i+1}) \in \mathcal{A}$, $i = 1, 2, \ldots, n-1$, being trivial when $\pi_q = \langle q\rangle$. A path π_q is *optimum* when $f(\pi_q) \leq f(\tau_q)$ for any other path τ_q, irrespective to its starting node. Defining Π as the set of all possible paths in the graph, the Image Foresting Transform (IFT) algorithm [7] minimizes a path cost map C,

$$C(q) = \min_{\forall \pi_q \in \Pi} \{f(\pi_q)\}, \tag{1}$$

by computing an *optimum-path forest* P—i.e., an acyclic predecessor map that assigns to every node q its predecessor $P(q) \in D_I$ in the optimum path π_q^P with terminus q or a marker $P(q) = nil \notin D_I$, when q is a *root* (starting node) of the map (i.e., $\pi_q^P = \langle q\rangle$ is optimum). The algorithm can also propagate to every node $p \in D_I$, the root $R(p) \in \mathcal{S}$ in the optimum-path forest, the object label map $L(p) = \lambda_O(R(p)) \in \{0, 1, 2, \ldots, c\}$ (resulting segmentation), and the marker label map $M(p) = \lambda_M(R(p))$. The roots of the map are drawn from $\mathcal{S}$, such that each object is defined by the optimum-path forest rooted in its internal seeds. Connectivity functions may be defined in different ways, which do not always guarantee the optimum cost mapping conditions [4], but can produce effective object delineation [14]. In this work, we explore the connectivity function

$$\begin{aligned} f_{\max}(\langle q\rangle) &= \begin{cases} 0 & \text{if } q \in \mathcal{S}, \\ +\infty & \text{otherwise,} \end{cases} \\ f_{\max}(\pi_p \cdot \langle p, q\rangle) &= \max\{f_{\max}(\pi_p), w(p, q)\}, \end{aligned} \tag{2}$$

where $\pi_p \cdot \langle p, q \rangle$ indicates the extension of path π_p by an arc $\langle p, q \rangle$ and $w(p, q)$ is an arc weight usually estimated from $\mathbf{I}$ (e.g., $w(p, q) = \|\mathbf{I}(q) - \mathbf{I}(p)\|$). The IFT algorithm with $f_{\max}$ computes optimum paths from $\mathcal{S}$ to the remaining nodes by growing one optimum-path tree $\mathcal{T}_r$ for each seed $r \in \mathcal{S}$.

The *dynamic* IFT essentially exploits the sets $\mathcal{T}_r$ to estimate the arc weights $w(p, q)$ as the costs of including q, through $\pi_p \cdot \langle p, q \rangle$, as part of the same object that contains p at the time the optimum path π_p is found (Fig. 2).

2.1 DynIFT Algorithm for $f_{\max}$

The dynamic IFT algorithm for $f_{\max}$ is a variant of the IFT algorithm, which maintains the dynamic trees $\mathcal{T}_r$, for all $r \in \mathcal{S}$, object label map L, path cost map C, marker label map M, predecessor map P, and root map R for possible use during the segmentation process, especially for arc weight estimation.

Algorithm 1 . DYNAMIC IFT FOR $f_{\max}$

INPUT: Image $(D_I, \mathbf{I})$, adjacency relation $\mathcal{A}$, and seed set $\mathcal{S}$ with labeling functions λ_O and λ_M.
OUTPUT: Object label map L.
AUXILIARY: Priority queue $Q = \emptyset$, dynamic sets $\mathcal{T}_r = \emptyset$, $\forall r \in \mathcal{S}$, maps C, R, P, and M, and variable *tmp*.

1. **For each** $p \in D_I$
2. $\quad$ $C(p) \leftarrow +\infty$, $R(p) \leftarrow p$, and $P(p) \leftarrow nil$
3. $\quad$ **If** $p \in \mathcal{S}$
4. $\quad\quad$ $C(p) \leftarrow 0$, $L(p) \leftarrow \lambda_O(p)$, and $M(p) \leftarrow \lambda_M(p)$
5. $\quad$ Insert p in Q
6. **While** $Q \neq \emptyset$
7. $\quad$ Remove p from Q, such that $p = \arg\min_{\forall q \in Q}\{C(q)\}$ and $\mathcal{T}_{R(p)} \leftarrow \mathcal{T}_{R(p)} \cup \{p\}$
8. $\quad$ **For each** $(p, q) \in \mathcal{A} \mid q \in Q$
9. $\quad\quad$ Estimate $w(p, q)$ as described in Section 2.2
10. $\quad\quad$ $tmp \leftarrow \max\{C(p), w(p, q)\}$
11. $\quad\quad$ **If** $tmp < C(q)$
12. $\quad\quad\quad$ $C(q) \leftarrow tmp$, $R(q) \leftarrow R(p)$
13. $\quad\quad\quad$ $L(q) \leftarrow L(p)$, $M(q) \leftarrow M(p)$, and $P(q) \leftarrow p$

Lines 1–5 of the DynIFT algorithm initialize the maps, being all pixels $p \in D_I$ defined as trivial paths $\langle p \rangle$ in P and inserted in Q. The main loop (Lines 6–13) computes in P optimum paths from the minima of the cost map (i.e., the seeds in $\mathcal{S}$) to the remaining pixels in $D_I \backslash \mathcal{S}$. When a pixel p is removed from Q in Line 7, the current path π_p^P, that can be obtained backward in P, is optimum and p is inserted in the dynamic tree $\mathcal{T}_{R(p)}$ of the root of p. The internal loop (Lines 8–13) considers only the adjacent pixels $q \in Q$ that does not belong to any dynamic set yet. It can estimate the arc weight $w(p, q)$ by extracting object information from the maps and dynamic sets (Sect. 2.2). The remaining lines compute the cost of the extended path $\pi_p^P \cdot \langle p, q\rangle$ and if this cost is lower than the current path cost $C(q)$, it updates the maps and the path π_q^P becomes $\pi_p^P \cdot \langle p, q\rangle$ in P.

Next we explore simple and yet effective ways to estimate the arc weights.

2.2 Dynamic Arc Weight Estimation

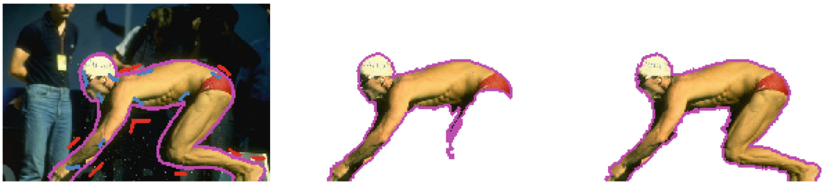
Algorithm 1 executes in $|D_I|$ iterations of the main loop. By the time a pixel p is removed from Q in Line 7, the tree $\mathcal{T}_{R(p)}$ contains information about the region conquered by the root $R(p) \in \mathcal{S}$ (which includes p), the map P contains the optimum path π_p^P, the map M contains the union $\bigcup_{\forall r \in \mathcal{S} | M(r)=M(p)} \mathcal{T}_r$ of trees rooted in each marker $M(p) \in \{1, 2, \ldots, m\}$, and the map L contains the union $\bigcup_{\forall r \in \mathcal{S} | L(r)=L(p)} \mathcal{T}_r$ of trees rooted in each object $L(p) \in \{1, 2, \ldots, c\}$. Therefore, color, texture, and shape information about the object or its regions can be explored for dynamic arc weight estimation—i.e., to estimate the cost of including q as part of the object that contains p. We then evaluate the following dynamic arc weight functions based on the tree mean color $\mu_{\mathbf{R(p)}}$ of the pixels $p \in \mathcal{T}_{R(p)}$ and the object mean color $\mu_{\mathbf{L(p)}}$ of the pixels $p \in \bigcup_{\forall r \in \mathcal{S} | L(r)=L(p)} \mathcal{T}_r$.

$$
\begin{aligned}
w_1(p,q) &= \|\mu_{\mathbf{R(p)}} - \mathbf{I(q)}\|, && (3)\\
w_2(p,q) &= \min_{\forall r \in \mathcal{S} | L(r)=L(p)} \{\|\mu_{\mathbf{r}} - \mathbf{I(q)}\|\}, && (4)\\
w_3(p,q) &= \|\mu_{\mathbf{L(p)}} - \mathbf{I(q)}\|, && (5)\\
w_4(p,q) &= w_1(p,q) + \|\mathbf{I(q)} - \mathbf{I(p)}\|, && (6)\\
w_5(p,q) &= w_2(p,q) + \|\mathbf{I(q)} - \mathbf{I(p)}\|, && (7)\\
w_6(p,q) &= w_3(p,q) + \|\mathbf{I(q)} - \mathbf{I(p)}\|. && (8)
\end{aligned}
$$

DynIFT with w_1 assumes that the mean color of the region of the object that contains p (i.e., the dynamic tree $\mathcal{T}_{R(p)}$) is more representative than $\mathbf{I(p)}$. However, it also assumes that each seed can only conquer pixels q whose color is similar to the mean color of that region. The purpose of w_2 is to relax this criterion by considering the closest mean color of all dynamic trees rooted in the same object. This allows, for instance, to delineate object regions not necessarily connected to their most similar seeds (Fig. 3). Function w_3 extends the concept of w_1 for the entire object, which should not be a good idea since the object may be represented by different parts. The remaining functions essentially add the local arc weight $\|\mathbf{I(q)} - \mathbf{I(p)}\|$ to the previous ones in order to evaluate the importance of the local contrast between regions.

3 Experimental Results

For comparison, we use the *power watershed* ($PW_{q=2}$) algorithm [5] (i.e., image segmentation based on minimum spanning forest and random walk), the IFT algorithm for $f_{\max}$ with arc weight function $w(p,q) = \|\mathbf{I(q)} - \mathbf{I(p)}\|$ (i.e., a



(a) (b) (c)

Fig. 3. (a) Original image with markers (red and blue) and ground-truth delineation (magenta). Segmentation results using arc weight functions (b) w_4 and (c) w_5. Even without object markers on the swimmer's legs, w_5 is still able to delineate it, because of the global similar tree search. (Color figure online)

watershed transform [7]), and the min-cut/max-flow algorithm [2,15]. The IFT-based watershed transform provides a cut in the graph given hard constraints (seed set) such that the lowest arc weight in the cut is maximum (i.e., it is an energy maximizer, $GC_{\max}$, as defined in [3] or a watershed cut as defined in [6]). Its counterpart is the energy minimizer, GC_{sum}, as defined in [3], which uses the min-cut/max-flow algorithm [2,15] and obtains a cut in the graph given the seed set such that the sum of the arc weights in the cut is minimum. For that, one can simply use the normalized complementary arc weight function $\bar{w}(p,q) = |\frac{w_{\max}-w(p,q)}{w_{\max}}|^{\alpha}$, where $\alpha \geq 1$ and $w_{\max}$ is the maximum value of $w(p,q)$ in the graph, in the min-cut/max-flow algorithm with source and sink connected to the seed set.

As proposed by Andrade and Carrera [1], our experiments run on two predefined sets of markers to avoid bias of prior knowledge of the process of segmenting with each algorithm. The first is the dataset from [9], in which about four mark-

Table 1. Experimental results.

Method	Gulshan's markers dataset		Andrade's markers dataset	
	Dice Acc. (%)	Time (secs)	Dice Acc. (%)	Time (secs)
GC_{sum}	76.2 ± 1.6	0.230 ± 0.167	90.6 ± 0.08	0.123 ± 0.075
$GC_{\max}$	75.6 ± 1.6	0.038 ± 0.012	89.5 ± 0.09	0.039 ± 0.012
$PW_{q=2}$	72.3 ± 1.7	0.966 ± 0.300	89.9 ± 0.08	1.015 ± 0.299
DynIFT w_1	84.0 ± 1.3	0.042 ± 0.016	92.1 ± 0.08	0.046 ± 0.018
DynIFT w_2	81.9 ± 1.8	4.633 ± 2.573	$\mathbf{95.4 \pm 0.04}$	8.460 ± 3.802
DynIFT w_3	74.7 ± 2.2	0.035 ± 0.012	82.8 ± 0.16	0.036 ± 0.013
DynIFT w_4	83.1 ± 1.4	0.047 ± 0.017	91.7 ± 0.08	0.048 ± 0.017
DynIFT w_5	$\mathbf{84.3 \pm 1.7}$	4.538 ± 2.484	95.3 ± 0.04	8.030 ± 3.485
DynIFT w_6	74.4 ± 2.1	0.039 ± 0.013	82.9 ± 0.16	0.037 ± 0.013

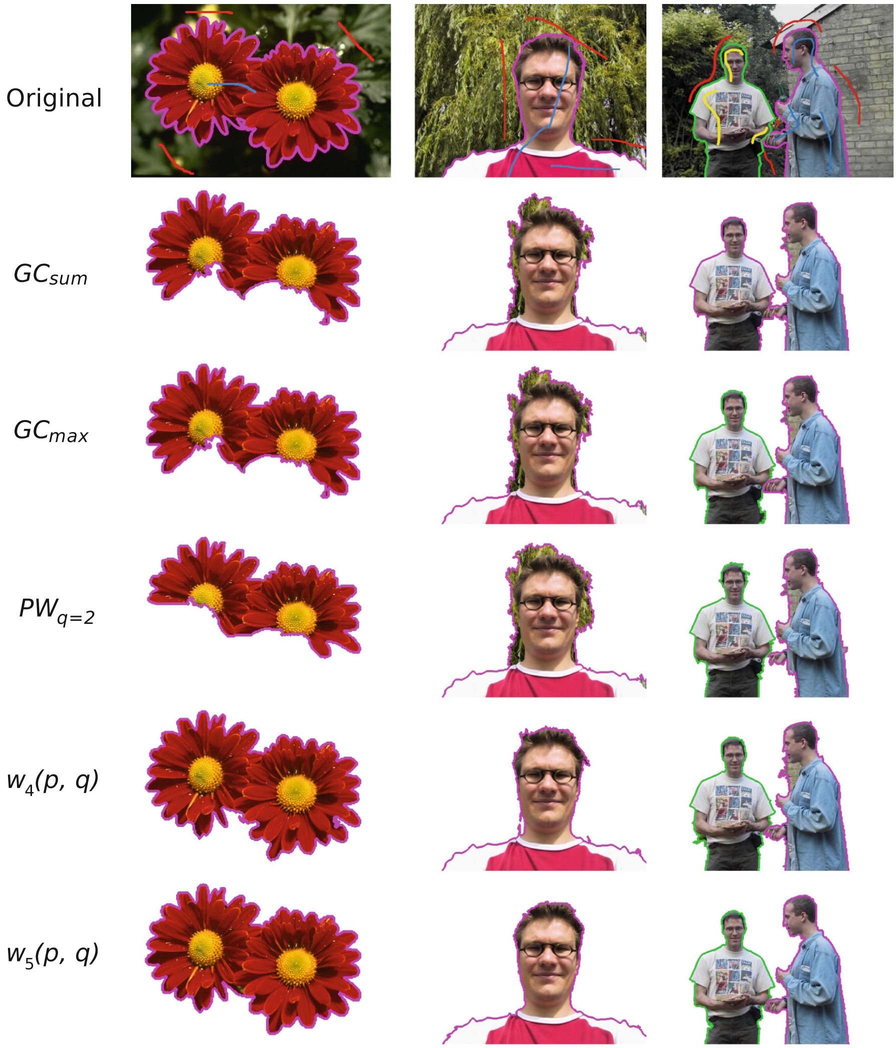


Fig. 4. Examples of segmentation results using DynIFT and the baselines. First row shows the images with the chosen markers for the objects (blue and yellow) and background (red), and the borders of the ground-truth segmentations (magenta and green). The remaining rows show the segmentations from the considered methods for the chosen markers. Note that GC_{sum} is only able to produce binary segmentations. (Color figure online)

ers cover a small area of both the background and foreground on each image. The second dataset contains a more carefully selected set of scribbles [1].

Table 1 shows the mean and standard deviation of the results over the 50 images of the *GrabCut* dataset and their respective markers, two baselines (with

$\alpha = 100$ for the GC_{sum} algorithm), and the six proposed arc weight functions. Computations were performed on an Intel Core i7-7700 CPU 3.60 GHz.

In all cases, except for w_3 and w_6, as predicted, the DynIFT-based methods can considerably improve the accuracy of object delineation in comparison with the baselines. Note that the relaxed versions, represented by w_2 and w_5, can obtain better results than using the local mean color only of the tree $\mathcal{T}_{R(p)}$, represented by w_1 and w_4. Figure 4 illustrates these results on a few examples.

4 Conclusion

In this paper, we present a new framework, named *Dynamic* IFT (DynIFT), that explores the object knowledge from the evolution of optimum-path trees during the IFT algorithm for more effective object delineation. We evaluated the DynIFT with some arc-weight estimation methods using color information. Experimental results show that DynIFT attains considerable accuracy gains in object delineation when compared to three well-established baselines. As future work, we intend to explore the dynamic proprieties of the optimum-path forest and its combination with pattern recognition algorithms to better understand how image delineation can be improved.

References

1. Andrade, F., Carrera, E.V.: Supervised evaluation of seed-based interactive image segmentation algorithms. In: Symposium on Signal Processing, Images and Computer Vision, pp. 1–7 (2015)
2. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
3. Ciesielski, K.C., et al.: Joint graph cut and relative fuzzy connectedness image segmentation algorithm. Med. Image Anal. **17**(8), 1046–1057 (2013)
4. Ciesielski, K.C., et al.: Path-value functions for which Dijkstra's algorithm returns optimal mapping. J. Math. Imaging Vis., 1–12 (2018)
5. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watershed: a unifying graph-based optimization framework. IEEE Trans. Pattern Anal. Mach. Intell. **33**(7), 1384–1399 (2011)
6. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: thinnings, shortest path forests, and topological watersheds. IEEE Trans. Pattern Anal. Mach. Intell. **32**(5), 925–939 (2010)
7. Falcão, A.X., Stolfi, J., de Lotufo, R.A.: The image foresting transform: theory, algorithms, and applications. IEEE Trans. Pattern Anal. Mach. Intell. **26**(1), 19–29 (2004)
8. Grady, L.: Random walks for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **28**(11), 1768–1783 (2006)
9. Gulshan, V., et al.: Geodesic star convexity for interactive image segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3129–3136 (2010)
10. Guo, Y., et al.: A review of semantic segmentation using deep neural networks. Int. J. Multimed. Inf. Retr. **7**(2), 87–93 (2018)

11. Kronman, A., Joskowicz, L.: Image segmentation errors correction by mesh segmentation and deformation. In: Mori, K., Sakuma, I., Sato, Y., Barillot, C., Navab, N. (eds.) MICCAI 2013. LNCS, vol. 8150, pp. 206–213. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40763-5_26
12. Maninis, K.K., et. al: Deep extreme cut: from extreme points to object segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
13. Miranda, P.A.V., Falcão, A.X.: Elucidating the relations among seeded image segmentation methods and their possible extensions. In: Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 289–296 (2011)
14. Miranda, P.A.V., Mansilla, L.A.C.: Oriented image foresting transform segmentation by seed competition. IEEE Trans. Image Process. **23**(1), 389–398 (2014)
15. Rother, C., Kolmogorov, V., Blake, A.: GrabCut: interactive foreground extraction using iterated graph cuts. In: ACM Transactions on Graphics, vol. 23, pp. 309–314 (2004)