Hierarchical Interactive Image Segmentation using Irregular Pyramids

Michael Gerstmayer, Yll Haxhimusa, and Walter G. Kropatsch

Pattern Recognition and Image Processing Group **
Vienna University of Technology - Institute of Computer Graphics and Algorithms
{mig,yll,krw}@prip.tuwien.ac.at

Abstract. In this paper we describe modifications of irregular image segmentation pyramids based on user-interaction. We first build a hierarchy of segmentations by the minimum spanning tree based method, then regions from different (granularity) levels are combined to a final (better) segmentation with user-specified operations guiding the segmentation process. Based on these operations the users can produce a final image segmentation that best suits their applications. This work can be used for applications where we need accuracy in image segmentation and in annotating images and ground truth among others. *Keywords:* interactive, hierarchical image segmentation, irregular combinatorial pyramid, image annotation

1 Introduction

Image segmentation cannot produce a perfect final segmentation, only by using low-level visual cues. The reason is the *intrinsic ambiguity* in the exact location of region boundaries in digital images. In general, homogeneity of low-level cues will not map to the semantics [12], and the degree of homogeneity of a region is in general quantified by threshold(s) for a given measure [6]. To avoid problems with the automatic segmentation methods one can use human help to guide segmentation methods, producing results acceptable by users/practitioners. Most interactive or semi-automatic segmentation algorithms found in literature make use of this external knowledge, some of them e.g. Snakes [11], Live Wire (or Intelligent Scissors) [15] and recent approaches based on the Graph Cuts formalism [1, 16, 2] are well known. They are often used in e.g. medical image segmentation, image or video object extraction and to refine or improve results from automatic methods.

But the notion of 'interactive' is ambiguous and not very well defined. Some of the methods are initialized (e.g. statistic shape models, rule sets, training sets) others use seed points or strokes for guiding and limiting a segmentation process [9]. Besides initialization, existing methods can also be categorized either as optimizing (manually guided, influenced) or post-processing methods

** This paper has been supported by the Austrian Science Fund under grants FWF-P20134-N13.

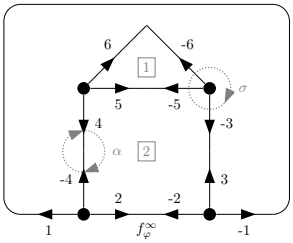
(manually corrected, modified) [10]. The work presented in this paper is located between the last two categories. It uses user-interaction to guide the minimum spanning tree (MST) based pyramid segmentation [8]. The MST-based segmentation method produces a stack of (dual) graphs (a graph pyramid) [8] on each level of the pyramid (Fig. 2). Each level of this hierarchy corresponds to one image segmentation. Regions from different levels of the segmentation pyramid are combined by user interaction resulting in a modified pyramid containing the 'final good' segmentation at top. In the regions where the user did not set any operation of modification, the algorithm will be guided automatically by a pairwise comparison of region similarity [5].

Meine et al [14] use the topological GeoMap representation and user interaction to guide a segmentation method in the medical image analysis. The use of a topologically correct representation has a major impact on the processing and its results. This motivated us to use combinatorial maps, as a topological representation. The authors in [13] interactively modify a hierarchical watershed segmentation, which is similar to our user modification(s) of the MST based segmentation hierarchy. In our case we can, if needed, access each pixel, which is not the case in the work of [13].

The paper is structured as follows. We first give a short overview of the combinatorial maps and combinatorial pyramids (Section 2). After that the operations that a user can set are presented in Section 3.1, in Section 4 we show some results and conclude the paper.

2 Combinatorial Image Pyramids

In this section a short overview of the most important concepts of combinatorial image pyramids are given. Combinatorial maps and generalized combinatorial maps define a general framework which allows to encode any subdivision on nD topological spaces orientable or non-orientable with or without boundaries [3]. Using $2D$ images, combinatorial maps may be understood as a particular encoding of a planar graph, where each edge is split into two half-edges called darts. Since each edge connects two vertices, each dart belongs to only one vertex. A $2D$ combinatorial map is formally defined by the triplet $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ [4] where $\boldsymbol{\mathcal{D}}$ represents the set of darts and $\sigma(d)$ is a permutation on $\boldsymbol{\mathcal{D}}$ encountered when turning clockwise around each vertex. Finally $\alpha(d)$ is an involution on $\boldsymbol{\mathcal{D}}$ which maps each of the two darts of one edge to the other one. Given a combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, its dual is defined by $\overline{G} = (\boldsymbol{\mathcal{D}}, \varphi, \alpha)$, with $\varphi = \sigma \circ \alpha$. The cycles of permutation φ encode the faces of the combinatorial map. In what it follows, the cycles of α, $\sigma(d)$ and φ contain g a dart d will be respectively denoted by $\alpha^*(d)$, $\sigma^*(d)$ and $\varphi^*(d)$ (an example of a combinatorial map is shown in Fig. 1). Thus all graph definitions used in irregular pyramids are analogously defined. A *combinatorial pyramid* is a stack of combinatorial maps successively reduced by the set of contraction and removal operations, that is, $(G_0, ..., G_k)$, where k represent the levels of the pyramid. Each map $k+1$ is build from the one below, k, by selecting a set of contraction kernels $K_{k,k+1}$ and applying it to a given



$\boldsymbol{\mathcal{D}}$ $(1,-1,2,-2,3,-3,4,-4,5,-5,6,-6)$:

$$\alpha = (1,-1)(2,-2)(3,-3)(4,-4)(5,-5)(6,-6)$$
$$\sigma = (1,-4,2)(-2,3,-1)(-3,-5,-6)(4,6,5)$$
$$\varphi = (5,-6)(2,3,-5,4)(-1,-4,6,-3)(-2,1)$$

Fig. 1: An example of a combinatorial map. The image of the house and its relations are described with permutations on the dartset $\boldsymbol{\mathcal{D}}$. The infinite face f_φ^∞ is encoded with the orbit $\varphi^*(-2) = (-2,1)$.

combinatorial map G_k to get the reduced $G_{k+1} = \mathcal{C}[G_k, K_{k,k+1}] = G_k \setminus K_{k,k+1}$. More on removal of the redundant edges can be found in [3].

Region adjacency graphs (RAG), dual graphs [8], combinatorial maps [7], and GeoMap [14] have been used before [3] to represent the partitioning of $2D$ space. From these structures, we use the combinatorial maps because RAGs cannot correctly encode multiple boundaries and inclusions. Dual graphs lack the explicit encoding of edge orientation around vertices, which is present in a combinatorial map [3](e.g. Fig. 1). Moreover with combinatorial maps, its dual must not be explicitly represented because one combinatorial map is enough to fully characterize the partition and can be easily deduced.

3 Interactive Operations on Pyramids

Usually, automatic segmentation methods will not be able to deliver a final segmentation that is acceptable by the users (see Fig. 3). Thus there is a need to perform a user interaction such that one can produce a better image segmentation. We have chosen a (hierarchical) pyramid based segmentation method where

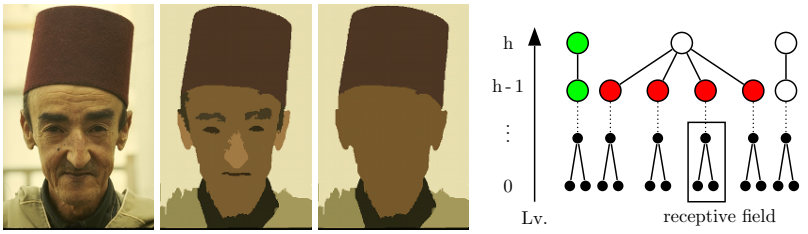


Fig. 2: *Left:* Hierarchical Segmentation of person; *Right:* Discrete levels with the merging tree. Interactive operations combine regions from different segmentations in different levels of the pyramid. Green vertex represent a region inhibited from merging in further processing, and red vertices are chosen to get merged.

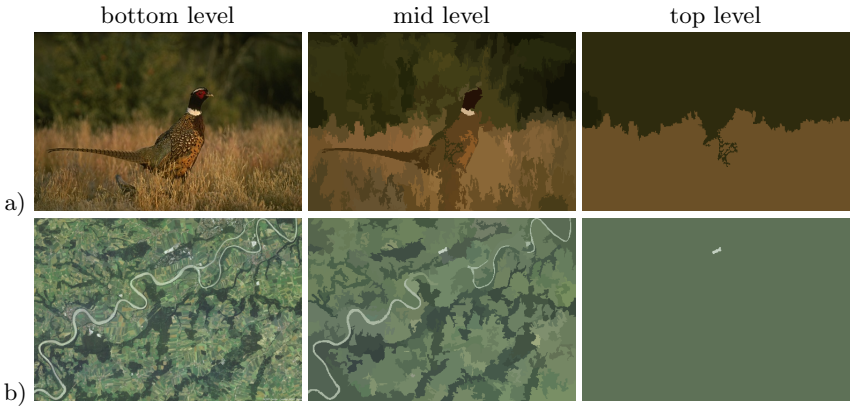


Fig. 3: Image segmentation with automatic segmentations method [8]; a) Due to lack of contrast bird is merged with meadow; b) Due to the thin structure the Danube river disappears.

we define user operation which will guide the merging and division by using regions in different level of the pyramid to a final acceptable image segmentation. The irregular (combinatorial) pyramid [7] produces automatically a stack of image segmentations (only some levels are shown in Fig. 3). The segmentation results are produced automatically by merging processes that take low-level cues (in this example RGB color values) into consideration [5]. At the beginning, the user will choose one of the levels of the pyramid called working-level that suits her requirement the best, and start the process of manually changing the image segmentation by merging and/or division operations. Note that in our pyramid all these manual operations will change the merging tree. In this work the user can set focus on region(s) lying in different levels of the pyramid (having different granularities). On these regions the user can define modifying operation(s) that will guide the processes in changing the merge tree in the pyramid . This results in a stack of segmentation images where the final (wished) segmentation is at top of the pyramid. Because we keep the hierarchy it is always possible to decompose the object into its subparts or restart the process for further refinement. Instead of doing this with unpredictable result (e.g. effect of a stroke in Graph Cuts) we explicitly can address each region in the merging tree (up to the pixel level if needed) while non restricting the flexibility of the algorithm in merging (other) non selected (not in the focus) regions automatically.

3.1 Modifying Operations

The manual image segmentation process consists of two parts:

- building the irregular pyramid based on the MST [8, 7], and
- a user interface where the user places its modification operations.

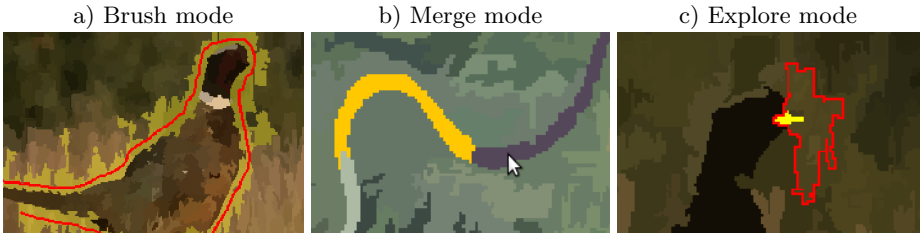


Fig. 4: User interactions. a) Using brush mode (red line) for effectively applying operations (encircle bird), b) Merging regions by selection (yellow - first selected region, violet - region to be selected), c) Dividing the region into its component and restoring the birds beak (indicated with yellow).

The framework uses as input the original image, its stack of automatically generated segmented images at different levels (e.g. images in Fig. 3), their labeling and the hierarchical information (merging tree) of the regions. Images and its structure are represented as (attributed) combinatorial maps.

Modifying relations between regions, requires the creation of a correspondence between the user-operations based on regions in the user interface (visual representation) and their combinatorial map correspondent. This information is given by the structural description of the image relations, inherently encoded in its primal and dual combinatorial maps. To get the interpixel-boundary e.g. between the roof (region 1) and the wall (region 2) in Fig. 1 in the primal or the edge representing the adjacency in its dual is calculated through permutations on the darts in $\boldsymbol{\mathcal{D}}$. Each region is represented in the primal by a dart $\in \boldsymbol{\mathcal{D}}$ and its orbit φ^* describing the boundary. These darts (aligned around a vertex corresponding to this region) encode also the links to the neighbouring regions in the dual. Therefore the joining edge for e.g. merging two regions is calculated through iterating the φ^* permutations each until both darts are on the same edge $\in \alpha$. This transformation from regions to edges is necessary because the operations placed in the user interface implicitly describe what to do with the available region information (labeling). The representation in terms of combinatorial maps explicitly defines what to be modified. In the presented work we need only the concept of regions' boundary in order to define the user operations. Thus, the idea is general and can be implemented also on other topological representations like region adjacency graphs, dual graphs [8] etc. Therefore this approach is not limited to the image representation with combinatorial maps even though it strongly benefits from them.

The operations for the purpose of modifying the relations between the regions are placed in the user interface either by separate selection or brushing over them (e.g. Fig. 4). The two fundamental operations that can be applied on adjacent regions r_1 and r_2 for guiding the segmentation process are:

– $\texttt{mrg}(\texttt{r}_1, \texttt{r}_2)$: merging regions r_1 and r_2

– `imrg`($\mathtt{r}_1$): inhibition of merging r_1 with other regions in the segmentation levels above

One can define other operations as a combinations of these two basic modifying operations. Since we exert influence on the merging tree, merging solely does not implicate that the resulting region will be inhibited automatically since the algorithm can decide to merge it in higher levels of the pyramid. Through nesting other combinations of operations are possible:

– `imrg`(`mrg`($\mathtt{r}_1, \mathtt{r}_2$)): merge and inhibit the resulting region,
– `imrg`($\mathtt{r}_1, \mathtt{r}_2$): inhibit both from merging with other regions, etc.

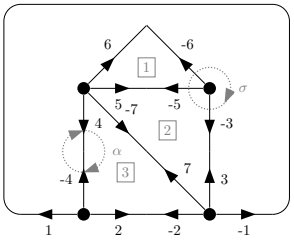
Since we use a hierarchical representation, it is also possible to select regions at different granularity from multiple levels. The basic way of doing this is traversing through all the pyramid, the other one is the so called 'explore' mode, intended to traverse down only within the receptive field of a single region. This can be used as a way of applying operations in higher levels on previously merged regions (at lower level of granularity). Its main purpose is splitting them up again while e.g. inhibiting the current state (Fig. 4, right). It is also possible that some regions (or their receptive fields) overlap which can lead to conflicts e.g. merging two regions and one of them is inhibited at the same time, which is not possible to achieve. We resolve these conflicts by analyzing the combinatorial map and the relations of region(s) under operation i.e. the affected edges (see Section 3.2 for more details). Thus we avoid cases where an edge must be deleted because of merge operation of two regions and at the same time preserved by inhibition operation.

Some more combination of operations and input methods are possible (see Section 5), but the listings above should be understood as an outline of the versatility of the framework. Finally a set of operations, each entry looking like the pattern $\mathtt{Lv.}, \mathtt{Op.}, \mathtt{R}_1[, \mathtt{R}_2]$ describes Level, Operation, affected Region 1 and Region 2 and is passed on to the segmentation framework.

3.2 Building Segmentation

To make the operations take effect on the merging tree it is fundamental to explain that they are not simply propagated upwards. The pyramid is recalculated including the changes and therefore we need to start from a level where the candidate-edges for a contraction are determined. In the automatic run (without user operations) we always start from the input image and at the base level. In the segmentation framework first the smallest weighted edges are collected for building a Minimum Spanning Tree (using Borůvka's algorithm) [8]. The decision whether two components (vertices connected by edge) are merged is guided by comparison of region similarity [5].

For retrieving the edges affected by the interactive operations this selection level mentioned before is determined by the lowest one among all operation levels. Reason for that is because everything above in the pyramid (each combinatorial



$R_1 \xrightarrow{split} (1)$

$R_2 \xrightarrow{split} (2,3)$

Darts for $\mathbf{R_1} = \varphi(5,-6)$

Darts for $\mathbf{R_2} = \varphi(3,-5,-7) \wedge \varphi(2,7,4)$

Fig. 5: The combinatorial map from a segmentation in a lower level (than Fig. 1) of the hierachy.

map and level in the hierarchy) has to be recalculated and therefore deleted. As a consequence of this, the affected regions of the operations placed in levels above the starting level in the merging tree have to be processed be equal.

This in turn implies because of former contractions within the receptive field of the original region that it may split up again into smaller parts (Fig. 5). The recalculation of the operations to an equivalent instruction set is achieved with the hierarchical information, through permutations on $\boldsymbol{\mathcal{D}}$ with φ and set operations and shown in the following example:

By comparison between Fig. 1 representing a higher level in the hierarchy and Fig. 5 the common starting level below the wall of the house region 2 splits up again into region 2 and 3 whereas the boundary of the roof remains the same. When applying e.g. the operation $\texttt{imrg}(\texttt{mrg}(1,2))$, meaning 'merge region 1 (roof) and 2 (wall) in Fig. 1 and inhibit the resulting region from merging' we can see that this operation leads to a different/non corresponding result when applied on the combinatorial map in Fig. 5.

Starting from the region/subregion relation and its φ permutation, the corresponding darts (or edges) can be reconstructed this way:
There are three different categories (enumeration below) and two collections (inhibition and removal) necessary to process the operation correctly:

1. **Former Contractions:** in each set of R_1, R_2 some darts are one the same edge (identified with permutation α and therefore adjacent in its dual graph). These will be remembered for **removal** e.g. $(7,-7)$
2. **User Operations:**
   - all remaining represent the outer borders for each region R_1, R_2, these will be remembered for **inhibition**, e.g. the darts $(3,-3)(5,-5)$,$(2,-2)(4,-4)$ and $(6,-6),(5,-5)$
   - now take the darts/edges from the outer borders of each region. Again applying a comparison with α yields again two darts occurring in each subset and lying on the same edge. This is the one we contract and will be remembered for **removal**, e.g. $(5,-5)$

As observed the edge $(5,-5)$ is assigned to both instruction sets and would lead to a conflict. Because we intended to merge, a rule decides that this one will be

a) Focus is set on the bird b) Focus is set on the river

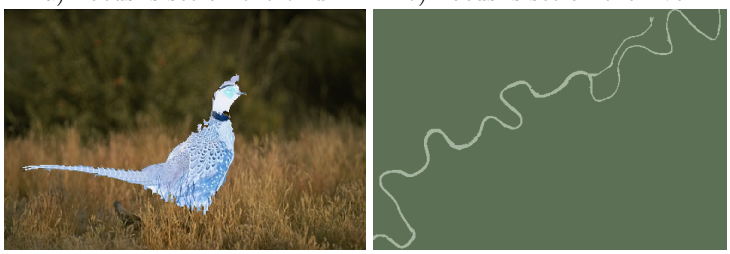


Fig. 6: Final segmentation after user interaction on the segmentation hierarchy.

removed. For the other operations the assignment to the collections is also well defined. On rebuilding the edges marked for removal are processed in separate levels not to interfere with the existing procedure (MST method) and stepwise removed by the contraction kernel until the next selection phase is initiated.

The remaining are throughout the levels inhibited from merging. At this stage it is required to recalculate the remaining inhibition operations since their region index has changed due to merging or internal renumbering. The upward propagation and the selection of edges is executed as long as the segmentation framework decides (based on image data) to remove some.

The resulting segmentation pyramid considers the selected regions and operations, resulting in a new final segmentation at top. Now a final state is reached, however this level can also be used to build a new hierarchy e.g. for creating a nested image annotation, by locking all levels below the top of the pyramid and iteratively merge the remaining regions. In contrast to other tools (e.g. labelme[17]) where each level of abstraction has to be created separately, this is an enormous simplification.

4 Segmentation Results

We show by means of some images the usage of the framework. We had first to define what is the focus and which objects are of interest.

To produce the result in Fig. 6a two rebuilds of the pyramid were necessary. A detailed set of user operations applied are shown in Table 1. The brush was used to encircle the bird (Fig. 4a) in a level of the pyramid with a fine segmentation. This causes that a limiting boundary is created. Inside of the area of interest and outside of it the algorithm merges the regions (i.e. removes the edges) automatically. In the second, the bird region is inhibited from merging, whereas the rest of the images is merged guided by the automatic segmentation to get a 'clean' background. In an aerial image of the Danube river (Fig. 3), we were looking to segment the river properly. Some parts of the river were correctly segmented in higher levels, thus these region are inhibited in our first manual interaction. In the thinner branches correction at pixel level was necessary (Fig.

Image	Run	Input	# Op.	Lv.#	Clicks	Time
1, bird	1.	brush	72 `imrg(mrg)`	1	1	
	2.	select	1 `imrg(mrg)`		1	
2, river	1.	select	24 `imrg(mrg)`	8	48	
		select	6 `imrg`		1	
	2.	select	3 `imrg`	1	1	

Table 1: Segmentation results in Fig. 6. Lv. # stands for number of different levels modified, # Op. is the overall sum of operations.

3). Thus we needed 48 clicks (third entry in Table 1). The final segmentation result is shown in Fig. 6. The final segmentation delivered by this framework can be easily annotated with labels. Since we have a hierarchical representation (the merging tree) one gets a nested annotation tree from the framework very easily.

5 Discussion

There is no straightforward strategy implied by the framework, on how to proceed with user-defined modification operations. Depending on the object(s)/region of interest where the user wants to set the focus on, the following ways can lead to the same segmentation result:

- select different regions from different levels,
- encircle object of interest with the brush (or selection),
- limit/fill-out object of interest using the brush (or selection),
- start from a coarse segmentation of the object and then split up, etc.

One can choose to combine one of the above to have a hybrid approach. Explicitly defining what to be done might cause that an object not denoted, but correctly segmented in the initial fully automatic run, will get lost.

It is also possible to start from or correct an existing segmentation, produced by other segmentation methods. We need only the border of the regions and establish a mapping to the corresponding combinatorial edges. A solution for this problem would be to reconstruct the merging tree with use of the labeling of the input and segmentation image given by a different segmentation method. Out of the receptive fields of each region, operations (e.g. $\texttt{imrg}(\texttt{mrg}(\texttt{r}_1, \texttt{r}_2))$) can be created to recalculate the intermediate levels.

The processing takes around a minute on images with 500×500 in a PC (2Gz processor with 2GB memory). We will evaluate this framework in the terms of user usability. A straightforward comparison with other frameworks is difficult since each approach is different and intended for different applications. Nevertheless the framework can be used easily for image/video annotation as well as for making ground truth.

5.1 Conclusion

The approach of interactively modifying an irregular pyramid by guiding it with user-specified operations is introduced. In contrast to a post-processing approach, the presented framework delivers in combination with the operations guided by the user good final segmentation results. The various strategies of interactions and the solutions developed for effective processing has been discussed.

References

1. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on PP(99), 1 (2010)
2. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In: Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. vol. 1, pp. 105 –112 vol.1 (2001)
3. Brun, L., Kropatsch., W.: Contraction kernels and combinatorial maps. Pattern Recognition Letters 24(8), 1051–1057 (2003)
4. Brun, L., Kropatsch, W.G.: Dual contraction of combinatorial maps. Tech. Rep. PRIP-TR-54, Institute of Computer Graphics and Algorithms 186/3, Pattern Recognition and Image Processing Group, TU Wien, Austria (1999)
5. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision 59(2), 167–181 (2004)
6. Fuh, C.S., Cho, S.W., Essig, K.: Hierarchical color image region segmentation for content-based image retrieval system. Image Processing, IEEE Transactions on 9(1), 156 –162 (Jan 2000)
7. Haxhimusa, Y., Ion, A., Kropatsch, W.G., Brun, L.: Hierarchical image partitioning using combinatorial maps. In: Chetverikov, D., Czuni, L., Vincze, M. (eds.) In Proceeding of the Joint Hungarian-Austrian Conference on Image Processing and Pattern Recognition. pp. 179–186. Hungary (May 2005)
8. Haxhimusa, Y., Kropatsch, W.G.: Segmentation graph hierarchies. In: SSPR/SPR. pp. 343–351 (2004)
9. Heimann, T., et al.: Comparison and evaluation of methods for liver segmentation from ct datasets. Medical Imaging, IEEE Transactions on 28(8), 1251 –1265 (aug 2009)
10. Hug, J.M.: Semi-Automatic Segmentation of Medical Imagery. Ph.D. thesis, Swiss Federal Institute of Technology Zürich (2000)
11. Kass, M., Witkin, A.P., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision 1(4), 321–331 (1988)
12. Keselman, Y., Dickinson, S.: Generic model abstraction from examples. IEEE Transactions on PAMI 27(5), 1141–1156 (2005)
13. Klava, B., Sumiko, N., Hirata, T.: Interactive image segmentation with integrated use of the markers and the hierarchical watershed approaches. In: VISSAPP (1). pp. 186–193 (2009)
14. Meine, H., Köthe, U., Stiehl, H.: Fast and accurate interactive image segmentation in the geomap framework. In: Tolxdorff, T. (ed.) Bildverarbeitung für die Medizin 2004. pp. 60–65. Springer (2004)

15. Mortensen, E.N., Barrett, W.A.: Interactive segmentation with intelligent scissors. Graphical Models and Image Processing 60(5), 349–384 (1998)
16. Rother, C., Kolmogorov, V., Blake, A.: "grabcut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. 23(3), 309–314 (2004)
17. Torralba, A., Russell, B., Yuen, J.: Labelme: Online image annotation and applications. Proceedings of the IEEE 98(8), 1467 –1484 (8 2010)