Saliency Segmentation based on Learning and Graph Cut Refinement

Paria Mehrani
pmehrani@uwo.ca
Olga Veksler
olga@csd.uwo.ca

Department of Computer Science
The University of Western Ontario
London, Canada

Abstract

Saliency detection is a well researched problem in computer vision. In previous work, most of the effort is spent on manually devising a saliency measure. Instead we propose a simple algorithm that uses a dataset with manually marked salient objects to learn to detect saliency. Building on the recent success of segmentation-based approaches to object detection, our saliency detection is based on image superpixels, as opposed to individual image pixels. Our features are the standard ones often used in vision, i.e. they are based on color, texture, etc. These simple features, properly normalized, surprisingly have a performance superior to the methods with hand-crafted features specifically designed for saliency detection. We refine the initial segmentation returned by the learned classifier by performing binary graph-cut optimization. This refinement step is performed on pixel level to alleviate any potential inaccuracies due to superpixel tesselation. The initial appearance models are updated in an iterative segmentation framework. To insure that the classifier results are not completely ignored during later iterations, we incorporate classifier confidences into our graph-cut refinement. Evaluation on the standard datasets shows a significant advantage of our approach over previous work.

1 Introduction

Humans are capable of understanding rather complex natural scenes rapidly. Most models of human visual system support the theory that to ease computational burden, only certain regions of the visual field are selected to be processed in detail [16, 19, 35]. These regions are typically called *focus of attention* or *salient*. In addition to computational advantages, selecting salient regions of an image is also useful for computer vision systems. Some applications include object recognition [30], image resizing [4, 11], image thumbnailng [23, 34], photo collages [28], small device displays [9], image retrieval [32], etc.

A salient region is generally understood as a part of an image that stands out from its surrounding and thus captures the attention of a human observer. However, there is no universally accepted definition of visual saliency. For example, in [1], they define as salient those regions that are visually more conspicuous due to their contrast with respect to their surroundings. In [18] a region is visually salient if it has unpredictable characteristics for different scales in some feature space. In [8], the saliency is approached in information theory framework, with saliency based on self-information of each local image patch. In [11], they develop a new interesting notion of saliency which is context-aware. Their aim is to also include as salient parts of the background that convey the dominant object context.

There are multiple saliency detection approaches [1, 2, 8, 14, 17, 18, 20, 21, 36]. Traditional ones [1, 2, 17, 36] consist of three main steps. First low level features, often inspired by biological visual systems [2, 17], are extracted. Then for each feature, a saliency map is computed, based, for example, on center-surround [17], or information maximization [8]. Last, the saliency maps for each feature are normalized and combined, usually linearly.

In [20], an approach different from the traditional is proposed. The problem is addressed in a supervised learning CRF framework and the weights for linear combination of features are learned. In addition, they develop regional and global features for salient object detection.

The approaches mentioned above spend a significant amount of effort to design features that are relevant for salient object detection. Instead we propose a simple supervised learning algorithm that learns to detect saliency based on generic features often used in computer vision. In recent years, many successful approaches to object detection rely on a general purpose image segmentation [13, 22, 25, 29]. The main idea is that image features should be extracted from image regions (or superpixels), as opposed to individual image pixels or rectangular patches. Superpixels contain more information than pixels, and therefore features extracted from them are likely to be more informative. Compared to rectangular patches, also popular in computer vision, superpixels align better with image edges, and therefore are more likely not to cross object boundaries.

In our approach we exploit features that are the standard ones often used in vision. They are based on color, texture, etc. These simple features, properly normalized, surprisingly have a performance superior to the methods with hand-crafted features specifically designed for saliency detection. In addition, we avoid a step that, somewhat arbitrarily, uses a linear combination to fuse the saliency maps obtained with each feature. Our features are combined with more general functions during the learning stage, in a manner that minimizes errors. We take the confidences obtained with the trained classifier as the saliency map.

Most previous approaches produce a real valued saliency map. Often an actual segmentation of a salient object is needed. One can threshold the saliency map to obtain a segmentation [39]. This method makes an independent decision at each pixel, which results in an incoherent segmentation. Another approach is to use a generic segmentation algorithm to partition an image into regions and then threshold the mean segment saliency [1]. The results are more coherent but not necessarily more accurate, since the state of the art in image segmentation is not advanced enough to ensure that segments do not cross object boundaries.

Our goal is an accurate segmentation of a salient object. Therefore, unlike previous work, we avoid simply thresholding a saliency map. Instead we refine the initial segmentation of the trained classifier with binary graph-cut optimization [6]. This refinement step is done on pixel level to obtain more accurate boundaries. Superpixel segmentation almost always contains inaccuracies, i.e. there are superpixels that straddle the boundary between the salient object and background. When optimizing on a pixel level, we have a chance to make corrections by encouraging salient object boundaries to align with image edges.

Our problem is an excellent fit for the iterative binary segmentation framework developed by [27]. In [27], initial appearance models for the object and the background are obtained with user interaction. Instead, we use the segmentation obtained by the trained classifier to initialize the color appearance models. These initial appearance models are iteratively updated in an EM-like framework of [27]. Appearance models initialized from the classifier results help as long as there is no significant overlap between the histogram of the salient object vs. background. If there is a large overlap, results can deteriorate. In this case

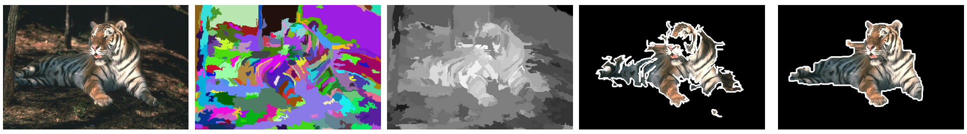


Figure 1: Summary of our approach. First the input image in (a) is segmented into superpixels, shown (b). Then the trained classifier is applied to each superpixel individually, classifier confidences are in (c). Based on confidences, superpixels are classified into salient/background, shown in (d). Finally the results in (d) are refined with graph-cut optimization, and the result after the refinement is in (e).

segmentation with graph cuts will deviate a lot from the initial classifier result. Therefore we found it helpful to incorporates classifier confidences into our iterative framework to insure that the final segmentation does not differ drastically from the initial one.

Out of the saliency segmentation algorithms, our work is perhaps closest to [20], which is also based on supervised learning. In [20], the features are linearly combined, what is learned is the coefficient for the linear combination. In our approach, a good non-linear way to combine features, including which features to combine, is learned automatically. Another difference is that in [20] a CRF framework is used, and learning is highly intractable in this framework. Our learning framework is based on a simpler modeling and is more tractable.

Our approach is more similar to segmentation-based object recognition methods such as in [25]. However, they use multiple segmentation to alleviate the problems due to instability of image segmentation. We use pixel level binary optimization. In addition, for our approach, iterative segmentation framework of [27] is an excellent fit and can improve results.

We evaluate our algorithm on the standard datasets and show a significant advantage of our approach over previous work. In [2], they introduce a new saliency segmentation method and show that it outperforms other state-of-the-art methods. We have performance significantly better than that of [2].

2 Our Approach

The overview of our method is in Fig. 1. First an input image 1(a) is oversegmented into regions 1(b), typically called superpixels. We use the efficient algorithm of [10] for this purpose. Next we use a trained classifier to output a confidence value, independently for each superpixel 1(c). The confidence map is also taken to be our saliency map. We threshold it at zero to get classification of the image into salient object and background, shown in 1(d). The classifier results are further refined in a graph-cut framework, where the problem is stated as binary segmentation. Our final results after refinement are in Fig. 1(e). Even though results in Fig. 1(d) are already rather accurate, graph-cut refinement shown in Fig. 1(e) fills missing parts of the salient object/background and has a more accurate boundary.

The two main steps are the confidence map computation 1(c) and the graph-cut refinement 1(e). We discuss them in sections 2.1 and 2.2, respectively.

2.1 Saliency Map

The main step of constructing a saliency map is training a saliency classifier from a dataset with ground truth. While the goal is to assign each pixel its saliency measure, a single pixel does not have enough reliable features for saliency classification. Therefore the classifier we use is based on “superpixels”, building on the recent encouraging results in scene labeling and object recognition [13, 22, 25]. A small image segment has more information than a single pixel and at the same time is more likely to belong to a single object, compared to a rectangular patch. Hence, the first step is to segment an image into “superpixels”. The second step is to extract features that may be useful for classifying a superpixel as salient/background. We have experimented with standard features used, for example, in [13, 22, 25] and hand-crafted features that seemed to be promising for salient object detection, such as those in [2]. However, we found that the standard features perform very well, provided that they are properly normalized, and there was no further benefit of including features specifically designed for saliency detection. Our set of features is based on color, size, location, and texture of superpixels, and is described in more details as follows.

Color: By itself, color might look like a weak feature since one object with a specific color might be salient in one image and be background in another one. Yet, some colors, like red, are naturally salient and some are more likely to be in the background, including blue, which is usually the color for sky or sea, and green, which is often the grass color. We make no assumptions on the salient colors and so it is the classifier that decides on the saliency of colors. The color features extracted from superpixels include mean of RGB and HSV values, hue histogram, and saturation histogram. Also, the wider a color is distributed in the image, the less possible the salient object contains that color. Therefore, the probability of the superpixel color given the color distribution of in the whole image is computed.

Location and Size: Location is a surprisingly good feature. Indeed, salient objects are usually located close to the central part of an image. Also, large segments are highly likely to belong to the background. Therefore, the mean of normalized x and y coordinates for pixels in a superpixel and also the normalized area of a superpixel are computed.

Texture: Texture is a good cue for many vision tasks and for saliency detection as well. For example, animals tend to have an interesting texture that draws attention and therefore makes regions with this texture salient. Although many animals developed texture for camouflage, most photographs taken by humans and containing animals actually feature these animals as the subject of focus, or, in other words, salient. Textures such as that of sand or grass tend to be more often in the background. In addition, usually a human photographer focuses on a salient object. Therefore salient objects have sharper texture edges, and the background has weaker ones. This property lead us to use different subsets of the texture filters [26]. The texture features include: mean and standard deviation of the absolute responses, histogram of maximum responses, and the maximum responses. Moreover, some features based on the texton [33] channels such as the most frequent texton channel, texton histogram and the number of different texton channels in each superpixel were extracted.

An important step of our approach is our feature normalization. Without normalization, the training does not work well at all, and the accuracy drops by a significant factor. Typically a given feature is normalized across all images simultaneously. We normalize a given feature of a superpixel only with the superpixels that came from the same image. That is features of a single image are normalized separately from features in all the other images. Indeed, a feature such as edge sharpness is relative to the image content. A superpixel that appears to have sharp texture edges in image A (and therefore likely to be part of a salient object in image A) may appear to have weak texture edges in image B, when compared against the superpixels in image B. Our normalization measures the strength/weakness of features in one superpixel with respect to the other superpixels in the same image.

After the features are normalized, we use a boosting algorithm with small decision trees [31] implemented with [37] for training. The problem is formulated as a two-class classification with classes being "salient" and "background". At the testing stage, a new input image is segmented into superpixels. Given the feature vector computed for each superpixel, the trained classifier outputs the class label together with the confidence for that superpixel. Then the saliency map is formed by assigning the confidence of the superpixel to all pixels in that superpixel. Thresholding the confidence map at zero gives classification of image pixels into the salient object and the background.

2.2 Refinement with Graph-Cuts

The results obtained with a saliency classifier, even when reasonable, contain holes and inaccurate boundaries. This is especially true since we classify on superpixel level. Superpixel boundaries often do not coincide with salient object boundaries. To improve the boundary and coherence of the classifier results, we use binary graph cut optimization [5] for refinement. The problem is formulated as binary salient object/background segmentation. Our energy function is of the standard form used in the graph cut framework:

$$E(f) = \sum_{p \in \mathscr{P}} D_p(f_p) + \lambda \sum_{\{p,q\} \in \mathscr{N}} V_{\{p,q\}}(f_p, f_q). \tag{1}$$

In the equation above, $\mathscr{P}$ is the set of image pixels, f_p is the binary label assigned to pixel p, f is collection of pixel-label assignments, $\mathscr{N}$ is a 4-neighborhood system, D_p is the data term, and $V_{\{p,q\}}(f_p, f_q)$ is the smoothness term for two neighboring pixels.

The smoothness term is the one from [6] which is

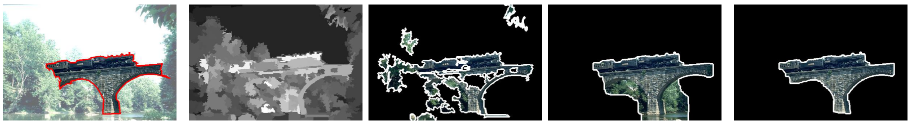
$$V_{\{p,q\}}(f_p, f_q) \propto \exp(-\frac{\Delta I^2}{2\sigma^2}) \cdot \delta(f_p \neq f_q) \tag{2}$$

Where ΔI denotes the intensity difference of two neighboring pixels, σ^2 is the variance of intensity difference of pixels, and $\delta(\cdot)$ is 1 if its argument is true and 0 otherwise. This $V_{\{p,q\}}$ encourages the boundary between labels to align with significant image edges.

Our data term consists of two parts. First of all, the classifier segmentation (Fig. 1(d)) is used to construct a color appearance model for the salient object and the background. This appearance model can help to fill the missing holes by reassigning pixels similar in appearance to the salient object (or background) to the appropriate label. To reduce dimensionality, we quantize the color space into 8^3 bins, 8 bins for each color channel. Then we compute the normalized histogram in this quantized color space.

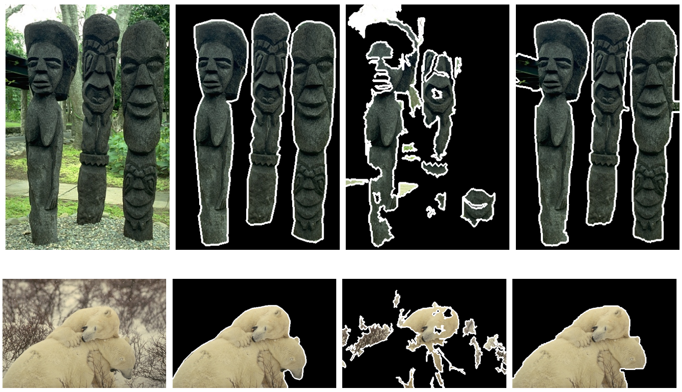
If we use only the appearance models for graph cut segmentation, the final segmentation can completely disregard the classifier results. In case when the appearance models between the salient object and the background overlap, this can cause serious errors. Therefore, to make sure that the refinement with graph cuts does not differ drastically from the classifier results, in the second part of our data term D_p we include the magnitude of the saliency map. Our full data term is as follows:

$$\begin{aligned} D_p(1) &= -\ln Pr(C_p|1) - \gamma \cdot \ln(m_p) \\ D_p(0) &= -\ln Pr(C_p|0) - \gamma \cdot \ln(1 - m_p) \end{aligned} \tag{3}$$



(a) ground truth (b) saliency map (c) classifier result (d) GC refinement without confidences (e) GC refinement with confidences

Figure 2: Excluding vs. including confidences in Graph-cut segmentation



(a) original image (b) ground truth (c) classification (d) final results

Figure 3: Graph cut refinement

where 1 is the salient object and 0 is the background, C_p is the quantized color of pixel p, m_p is the confidence of pixel p, and γ controls the relative importance of the two parts. We also assume that confidences m_p are renormalized to be in the range $[0,1]$ with $[0,0.5]$ indicating the preference for the background and $(0.5,1]$ indicating preference for the salient object.

Fig. 2 illustrates some examples of GC optimization with and without the magnitude of saliency map term. Fig. 2(a) shows the ground truth and Fig. 2(b) the saliency map. Fig. 2(c) shows the classifier result. Notice that several pieces of the background are classified as salient. Therefore the initial model for the salient object is not very accurate. If the magnitude of the salience map (i.e. classifier confidences) are not incorporated, the result is as in Fig. 2(d). Even larger part of the background is misclassified as salient because it fits into the somewhat inaccurate appearance model for the salient object. With inclusion of the confidences, however, we get results in Fig. 2(e). The results are improved because the pieces of the background erroneously added in Fig. 2(d) have low confidence for being salient. The other parts are cleaned up too because of the constraint of image edge alignment.

Our problem is an excellent for the iterative EM-style framework of [18]. The initial appearance model is provided by the classifier results, as opposed to user interaction. Optimization is performed with graph cuts and the appearance model is re-estimated from the new segmentation. Iterative segmentation usually improves the color appearance models of the salient object and the background. Confidences in Eq. (3) are never updated, it is just the appearance models that get updated. The energy in Eq. (1) is optimized by computing a minimum graph cut on a certain graph [5], which we compute with [7].

Fig. 3 illustrates the improvements that can be achieved with graph cut refinement. Columns (a)-(c) show original images, ground truth, and classifier results, respectively. Col-

Dataset	Saliency Map Error				Graph Cut Refinement Error			
	Itti	Hou	Achanta	**Ours**	Itti	Hou	Achanta	**Ours**
BSD	25.01	26.24	26.32	**21.25**	25.47	26.64	30.63	**20.12**
ASD	17.63	19.38	12.15	**10.57**	14.38	17.68	14.60	**7.58**
SED	22.00	25.71	25.17	**12.99**	21.48	25.81	25.85	**15.56**

Table 1: Performance comparison of different saliency detection methods. Columns 2-5 show percentage errors obtained from saliency map. Columns 6-9 show percentage errors after utilizing graph cut refinement.

umn (d) shows the improvements with graph cut refinement.

3 Experimental Results

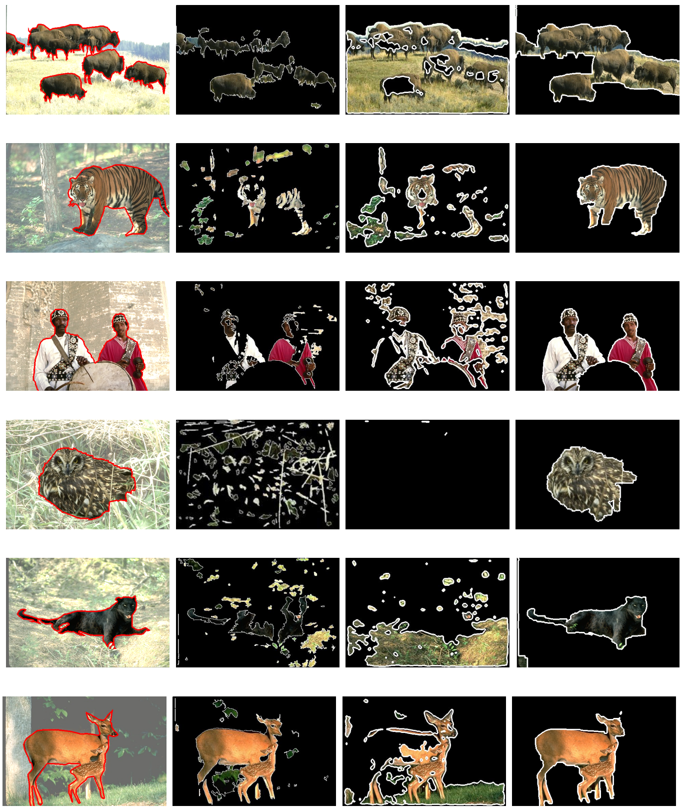


Figure 4: Comparison of the performance for different methods.

The method proposed by Achanta *et al.* [2] gives a final labeling of the image by applying an adaptive threshold on the saliency map. However, Hou *et al.* [15] and Itti *et al.* [17] give only the saliency map as the final result. Therefore, for Hou *et al.* and Itti *et al.*, to

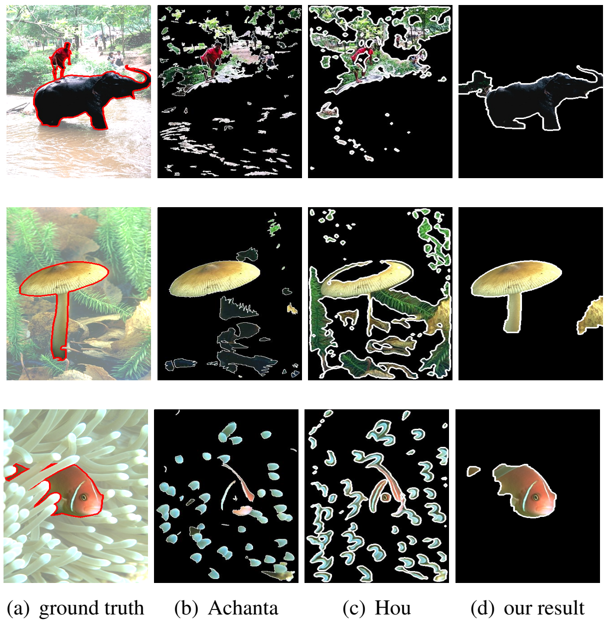


Figure 5: Comparison of the performance for different methods.

get a segmentation, we applied different thresholds on their saliency maps and selected the threshold that gives the smallest average error rate for each dataset.

We evaluate our algorithm on three datasets: from Berkeley (BSD) [24], Achanta *et al.* (ASD) [1], and Alpert *et al.* (SED) [3]. There is also a saliency dataset by Liu *et al.* [20]. However, their ground truth is a rectangular box, which is not accurate enough for our purposes since we are after pixel-precise salient object segmetnation. The ground truth for all of these datasets is somewhat subjective, since it is labeled by human subjects manually. ASD dataset contains manually labeled salient object segmentation, constructed by [1]. BSD and SED sets contain manual segmentations into multiple regions for BSD and two segments for SED. There are several labelings for each image for both of these datasets. We chose as salient the segment for SED (and possibly multiple segments for BSD) that appears most appropriate. The quality of our choices can be judged from Figs. 4 and 5 and the supplementary material. The SED database is gray scale, therefore features based on color were eliminated. The BSD, ASD, SED datasets contain 300, 1000, and 100 images, respectively. For evaluation purposes, we applied 5-fold cross validation to ASD and 10-fold cross validation to BSD and SED datasets. Since ASD is a much larger dataset, less folds were used.

We directly compare the performance of our algorithm with Achanta [2], Itti [17], and Hou [15], since their code is easily available. However, in Achanta [2] they compare their performance to five other saliency detection algorithms, namely [1, 12, 15, 17, 21], and show that their new method developed in [2] has superior performance over all five of them.

Table 1 shows the error rates for different algorithms on three datasets. Columns (2)-(5) show performance obtained without graph-cut refinement for all methods, including ours. Our method, even without graph-cut refinement, significantly outperforms all the others. Notice that the method of Achanta *et al.* is clearly superior to Itti *et al.* and Hou *et al.* only on their own dataset (ASD). This suggests that perhaps their parameter settings were overfitted for their database, the only one they used for experiments. Notice also that all algorithms

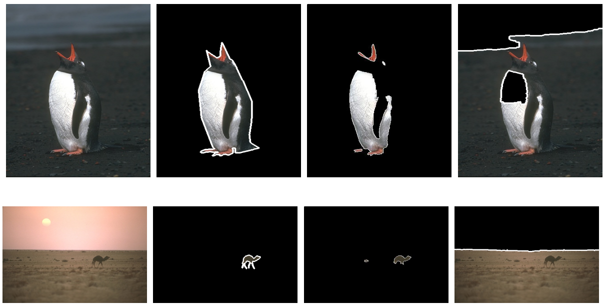

(a) original (b) ground truth (c) Achanta (d) our result

Figure 6: Worst failure images of our method compared to Achanta *et al.* [2]. From left to right: original image, ground truth, results from Achanta *et al.* [2], results of our method.

perform much better on ASD set. These images are of very high quality, the salient object is often of distinct color and surrounded by sharp edges. BSD and SED datasets have images of standard quality and more challenging appearance for the salient object.

We should mention that the straightforward comparison of our method to previous work is somewhat unfair, since we use the ground truth for training (using cross validation) and the other methods are not based on supervised learning. However, it is not clear how to do a more fair comparison, since our method inherently needs to train on labeled data. More importantly, we show the point that learning saliency measure from manually labeled set give better performance than trying to design such a measure by hand.

In addition to performing graph cut refinement on our own saliency map, we performed this step on the saliency maps from all the other methods. Table 1 has a comparison after graph cut refinement in columns (6)-(9). For our saliency maps, graph cuts improve results significantly on ASD dataset, slightly for the BSD dataset, and worsen the results on SED dataset. The worsening of results on SED dataset is probably due to SED containing grayscale images. Appearance models based on grayscale values are likely to highly overlap for the foreground and the background, making graph cut segmentation less reliable.

Graph cut refinement mostly worsens the results of the other methods (Itti *et al.*, Hou *et al.*, Achanta *et al.*), sometimes significantly. Their saliency maps are less accurate. Therefore it seems important to start with a reasonable saliency map to get an improvement.

Fig. 4 and 5 show several examples of successful salient object segmentation of our method, and compare them to Achanta's *et al.* [1] and Hou's *et al.* [15] methods. There are, of course, cases when our results are inferior. Fig. 6 depicts two worst detection results by our algorithm. The images in Figs. 4, 5, and 6 are from BSD dataset.

References

[1] R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk. Salient region detection and segmentation. In *International Conference on Computer Vision Systems*, pages 66–75, 2008.

[2] R. Achanta, S. Hemami, F. Estrada, and S. Süsstrunk. Frequency-tuned salient region detection. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[3] "Sharon Alpert, Meirav Galun, Ronen Basri, and Achi Brandt". "image segmentation by probabilistic bottom-up aggregation and cue integration.". In *"Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition"*, "2007".

[4] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Transactions on Graphics*, 26(3):10, 2007.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.

[6] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient n-d image segmentation. *International Journal Computer Vision*, 70(2):109–131, 2006.

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

[8] N. Bruce and J. Tsotsos. Attention based on information maximization. *Journal of Vision*, 7(9):950–950, 2007.

[9] L.Q. Chen, X. Xie, X. Fan, W.Y. Ma, H.J. Zhang, and H.Q. Zhou. A visual attention model for adapting images on small displays. *Multimedia Systems*, 9(4):353–364, 2003.

[10] P. F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[11] S. Goferman, L. Zelnik-Manor, and A. Tal. Context-aware saliency detection. *Conference on Computer Vision and Pattern Recognition*, 2010.

[12] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Advances in Neural Information Processing Systems*, pages 545–552, 2007.

[13] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *ACM Transactions on Graphics*, 24(3):577–584, 2005.

[14] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[15] X. Hou and L. Zhang. Dynamic visual attention: searching for coding length increments. In *Advances in Neural Information Processing Systems*, pages 681–688. 2009.

[16] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40:1489–1506, 2000.

[17] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[18] T. Kadir and M. Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[19] C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology*, 4:219–227, 1985.

[20] Tie Liu, Jian Sun, Nan-Ning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[21] Y.F. Ma and H.J. Zhang. Contrast-based image attention analysis by using fuzzy growing. In *ACM international conference on Multimedia*, pages 374–381, 2003.

[22] Tomasz Malisiewicz and Alexei A. Efros. Improving spatial support for objects via multiple segmentations. In *British Machine Vision Conference (BMVC)*, 2007.

[23] L. Marchesotti, C. Cifarelli, and G. Csurka. A framework for visual saliency detection with applications to image thumbnailing. In *International Conference on Computer Vision*, 2009.

[24] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference Computer Vision*, volume 2, pages 416–423, July 2001.

[25] C. Pantofaru, C. Schmid, and M. Hebert. Object recognition by integrating multiple image segmentations. In *European Conference on Computer Vision*, pages 481–494, 2008.

[26] X. Ren and J. Malik. Learning a classification model for segmentation. In *International Conference on Computer Vision*, volume 1, pages 10–17, 2003.

[27] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[28] C. Rother, L. Bordeaux, Y. Hamadi, and A. Blake. Autocollage. *ACM Transactions on Graphics*, 25(3):847–852, 2006.

[29] B.C. Russell, A.A. Efros, J. Sivic, W.T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Conference on Computer Vision and Pattern Recognition*, 2006.

[30] U. Rutishauser, D. Walther, C. Koch, and P. Perona. Is bottom-up attention useful for object recognition? *Conference on Computer Vision and Pattern Recognition*, 2:37–44, 2004.

[31] Robert E. Schapire. The boosting approach to machine learning: An overview, 2002.

[32] L. Shao and M. Brady. Invariant salient regions based image retrieval under viewpoint and illumination variations. *Journal of Visual Communication and Image Representation*, 17(6):1256–1272, 2006.

[33] J. Shotton, J.M. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. pages 1–15, 2006.

[34] B. Suh, H. Ling, B.B. Bederson, and D.W. Jacobs. Automatic thumbnail cropping and its effectiveness. In *ACM symposium on User interface software and technology*, pages 95–104, 2003.

[35] J.K. Tsotsos, S.M. Culhane, W.Y.K. Wai, Y. Lai, N. Davis, and F. Nuflo. Modeling visual attention via selective tuning. *Artificial Intellllligence*, 78(1-2):507–545, 1995.

[36] R. Valenti, N. Sebe, and T. Gevers. Image saliency by isocentric curvedness and color. In *International Conference on Computer Vision*, 2009.

[37] Alexander Vezhnevets. http://graphics.cs.msu.ru/en/science/research/machinelearning/adaboosttoolbox.

[38] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Joint optimization of segmentation and appearance models. In *International Conference on Computer Vision*, 2009.

[39] L. Zhuang, K. Tang, N. Yu, and Y. Qian. Fast salient object detection based on segments. *Measuring Technology and Mechatronics Automation*, 1:469–472, 2009.