Saliency Detection within a Deep Convolutional Architecture

Yuetan Lin[1], Shu Kong[2,3], Donghui Wang[1,*], Yueting Zhuang[1]
[1]College of Computer Science, Zhejiang University, Hangzhou, P. R. China
{linyuetan,dhwang,yzhuang}@zju.edu.cn
[2]Hong Kong University of Science and Technology, Hong Kong, P. R. China
[3]Huawei Noah's Ark Lab, Hong Kong, P. R. China
aimerykong@ust.hk

Abstract

To tackle the problem of saliency detection in images, we propose to learn adaptive mid-level features to represent image local information, and present an efficient way to calculate multi-scale and multi-level saliency maps. With the simple k-means algorithm, we learn adaptive low-level filters to convolve the image to produce response maps as the low-level features, which intrinsically capture texture and color information simultaneously. We adopt additional threshold and pooling techniques to generate mid-level features for more robustness in image local representation. Then, we define a set of hand-crafted filters, at multiple scales and multiple levels, to calculate local contrasts and result in several intermediate saliency maps, which are finally fused into the resultant saliency map with vision prior. Benefiting from these filters, the resultant saliency map not only captures subtle textures within the object, but also discovers the overall salient object in the image. Since both feature learning and saliency map calculation contain the convolution operation, we unify the two stages into one framework within a deep architecture. Through experiments over challenging benchmarks, we demonstrate the effectiveness of the proposed method.

Introduction

Visual saliency is basically a process that detects scene regions different from their surroundings (Borji and Itti 2012). Saliency refers to the attraction of attention arising from the contrast between the feature properties of an item and its surroundings (Lin, Fang, and Tang 2010). Recently, the field of visual saliency has gained its popularity in computer vision, as it can be widely borrowed to help solve many computer vision problems, including object detection and recognition (Walther and Koch 2006; Shen and Wu 2012; Yang and Yang 2012), image editing (Cheng et al. 2011), image segmentation (Ko and Nam 2006), image and video compression (Itti 2004), etc.

There are many popular saliency models in literature. Itti, Koch, and Niebur propose a general model which uses colors, intensity and orientations as the features to represent image locals (1998). Lin, Fang, and Tang adopt the local entropy to deal with the specific situation that human's perceptual salient region has low local contrast values but a large contrast background (2010). Borji and Itti use two color models and two scales of saliency maps to boost the detection performance (2012). As all these models use hand-engineered features that separately consider colors or edge orientations, the local representation lacks adaptiveness. Recently, Yan et al. propose to fuse segmented image maps at three levels into the final saliency map to deal with a practical situation that the salient object has complex textures, facing which many methods merely focus on inner-object textures and fail in discovering the true salient object (2013). Although these methods consider multiple cues such as colors and edge orientations and multi-level salient regions, they are not generic enough due to both the hand-crafted features and the saliency maps at fixed scales.

In image classification literature, researchers begin to learn mid-level features for better representation of images. The so-called mid-level features remain close to image-level information without attempts at any need for high-level structured image description, and are generally built over low-level ones (Boureau et al. 2010). The low-level features can be either hand-engineered descriptors like SIFT (Lowe 2004) and HoG (Dalal and Triggs 2005), or the learned ones *e.g.* convolutional neural networks (LeCun et al. 1998) and deep belief networks (Hinton and Salakhutdinov 2006; Boureau, Cun, and others 2007). Amongst the feature learning methods, the simple k-means algorithm can be also used to generate mid-level features, which is empirically proven to be quite effective (Coates, Ng, and Lee 2011).

Inspired by feature learning methods and saliency detection community, in this paper, we propose a unified framework to tackle the saliency detection problem, as demonstrated by Figure 1 on the flowchart of our model. Our method first learns the low-level filters by the simple k-means algorithm, and uses the resultant filters to convolve the original image to get low-level features, which can intrinsically and simultaneously consider color and texture information. With additional techniques such as hard threshold and pooling at multiple scales, we generate mid-level features to robustly represent image locals. Then, we predefine some hand-crafted filters to calculate saliency maps at multiple levels. These multi-scale and multi-level saliency maps captures not only the salient textures within the object, but also the overall object in images. Finally, we fuse

[*]Corresponding author

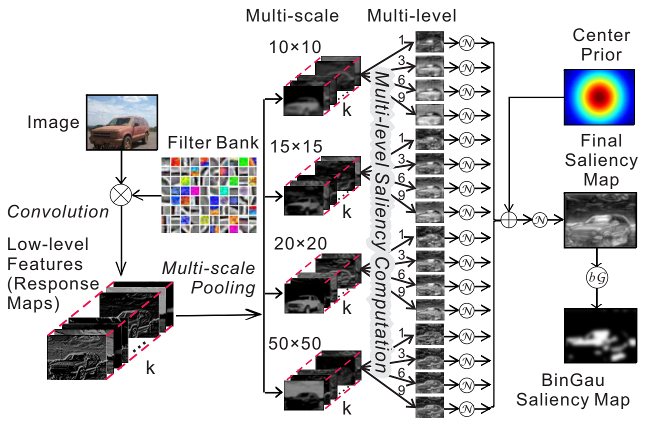


Figure 1: (Best view in color and zoom in.) Flowchart of the proposed framework. The filter bank produce k response maps via convolution with the k low-level filters. Multiple pooling scales ensure that saliency can be better captured in face of varying sized local contrasts. Multi-level saliency calculation produces more maps to capture both inner-object textures and the salient object itself. The final saliency map is the fusion of all these intermediate maps with the center prior. The $\mathcal{N}$ and $b\mathcal{G}$ mean the normalization operation, and Gaussian blur after binarization, respectively.

these intermediate maps into the final saliency map. As both our feature learning and saliency map calculation contain the common convolution operation, we unify the two stages into one framework within a deep architecture, which also mimics the human nerve/cognitive system for receiving and processing visual information (O'Rahilly and Müller 1983). Our model merely includes feed-forward computations through the architecture, thus parallel computing can be adopted to expedite the overall process.

The rest of this paper is organized as follows. We first briefly review several related works. Then, we elaborate our proposed saliency model. We present experimental comparisons with other popular methods to show the effectiveness of our model, before closing this paper.

Related Work

Different from popular methods for saliency detection, our method adaptively learns mid-level features to represent image locals. The so-called mid-level features remain close to image-level information without attempts at any need for high-level structured image description, and are generally built over low-level ones (Boureau et al. 2010). The mid-level features can be built in a hand-engineered manner over SIFT or HoG descriptors, *e.g.* concatenating several pooled vectors over sparse codes of these low-level descriptors in predefined pyramid grids (Yang et al. 2009). It can also be learned hierarchically in a deep architecture, *e.g.* summing up outputs of previous layer as the input of current layer through nonlinear operations such as max pooling and sigmoid function (Lee et al. 2009; Krizhevsky, Sutskever, and Hinton 2012). Recent studies demonstrate that the simple k-means method for convolution can also produce promising mid-level features w.r.t comparable performances (Coates, Ng, and Lee 2011; Coates and Ng 2012).

In literature, to boost performance of saliency detection, methods consistently consider either multiple cues like colors and edge orientations, or multiple scales of local contrast calculation. For example, Borji and Itti take advantage of two different color models (Lab and RGB) and two scales (local and global) of local contrast calculation (2012), and fuse these resultant saliency maps into the final one. But their method uses non-overlapping patches based on color information to represent image locals, so some valuable information is missed due to bound effect and lacks of texture/edge consideration. Moreover, their local and global scales need to be generalized for more subtle situations. Yan et al. explicitly consider the situation that the real salient object has complex inner textures, which lead popular methods to focusing on the trivial patterns rather than the object itself. For this reason, they propose to segment the image at three scales to generate intermediate saliency maps, and fuse them into the final one. Similarly, the preprocessing stage of image segmentation merely relies on color information, thus ignoring other valuable cues such as textures.

Saliency Detection within a Deep Convolutional Architecture

Overview of the Proposed Model

The deep architectural framework for saliency detection consists of two parts, mid-level feature generation and intermediate saliency map calculation at multiple scales and multiple levels, as demonstrated by Figure 1. The mid-level features are generated by threshold and pooling over low-level features, which are the response maps by convolving the image with a bank of low-level filters learned by the simple k-means algorithm. Here, our method uses a set of predefined filters to fast pool the mid-level features in an overlapping manner, therefore, subtle information can be captured such as inner-object textures and the salient object itself. Then, we define a bank of hand-crafted filters to calculate local contrast maps at multiple levels, *i.e.* the size of "local" is defined by our filters in this step. A benefit of this is to ensure the whole salient object can be detected. Finally, we normalize all the intermediate maps and fuse them into the final saliency map with the vision prior of center bias (Tatler 2007). In this section, we elaborate the important components in feature learning and intermediate saliency map calculation, and demonstrate how to unify these components into one deep architecture.

Mid-level Feature Learning

***K*-means for Low-level Filters.** We use the simple k-means algorithm to learn low-level filters instead of handcraft ones, such as SIFT and HoG. The filters are learned from a large set of patches, which are randomly extracted from web downloaded images, as demonstrated by Figure 2. We can see in this figure, different from existing methods that use hand-crafted features such as colors and edge orientations separately, our learned filters simultaneously capture

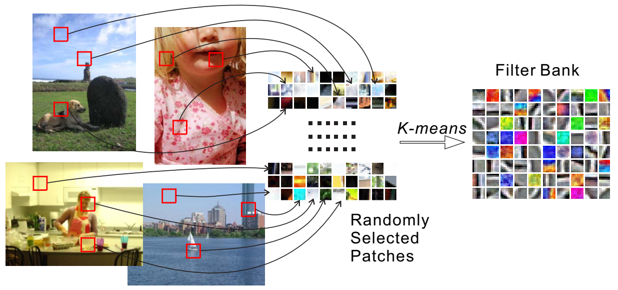


Figure 2: Illustration of low-level filters learning by k-means algorithm. Color patches of size $8 \times 8 \times 3$ are randomly extracted from images that are downloaded from the internet. Over these patches, the k-means simply learns $k = 80$ filters throughout our work, and we store them in a filter bank.

both color and edge information. Throughout our work, we learn $k = 80$ filters and store them in a filter bank, which is kept unchanged for extracting low-level features for all images. The filter bank is used to generate low-level features as described in the following.

Response Maps as Low-level Features. Over the bank of k filters, we convolve the input image with them, and produce k response maps, as shown in the left panel of Figure 3. In addition, we adopt a thresholding operation and normalization along the third mode of these maps to produce sparse responses. Thresholding produces sparse responses, while normalization makes all the low-level features have unit Euclidean length. A benefit of these operations is that small response values can be removed, which merely reflect trivial information in the image; meanwhile, high responses are highlighted that capture more meaningful local information. Empirically, we find that an effective way to threshold the response maps is to keep first 50 largest elements in the $k = 80$ ones at the same position and zero the others. We define the hard threshold parameter to be the ratio of the number of these retained largest elements to the number of response maps; here we denote the thresholding parameter as $5/8$. After this process, we generate the low-level features and feed them into the next step to produce mid-level ones.

Mid-level Features Generation. Over the low-level features in the response maps, we turn to generate mid-level features to represent image locals. The mid-level features are the spatially pooled vectors in local patches of predefined sizes, as described by the right panel of Figure 3. The pooling operation will capture local information of each pixel in the original image, thus leading to a better local representation. As for the pooling technique, average and max pooling are two popular choices. But through experiments, we find average pooling is better than max pooling. The reason is that average pooling aggregates local statistics information by preventing large response values taking over and small ones being removed out. In contrast, max pooling always captures the larger response values, which can be outliers regions in the image rather than human perceptual saliency. As well, it is worth noting that average pooling generate the same results as mean filtering (Jahne 2002).

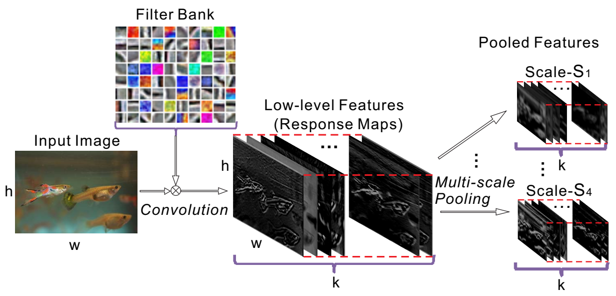


Figure 3: Convolved response features for multi-scale pooling. The convolved low-level response maps are pooled with filters at multiple scales from $S_1 \times S_1$ to $S_4 \times S_4$.

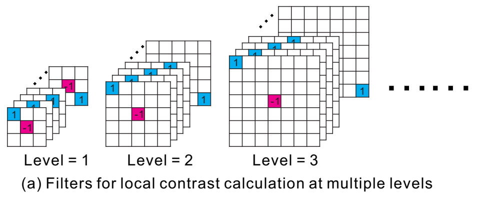
To facilitate the average pooling operation, we construct several filters with different sizes. Specifically, a filter of $s \times t$ can be represented by a square matrix $\mathbf{s} = \frac{\mathbf{1}}{st} \in \mathbb{R}^{s \times t}$, where $\mathbf{1}$ is a matrix with all the elements equaling to one with appropriate size. Then, convolving each low-level response map with the filter $\mathbf{s}$ will lead to the mid-level feature map accordingly. There are two advantages of the convolution operation for average pooling. First, convolution means every possible local patch is considered for calculating local contrast. Second, convolution at multiple scales can be done in a parallel manner, thus fast processing can be expected.

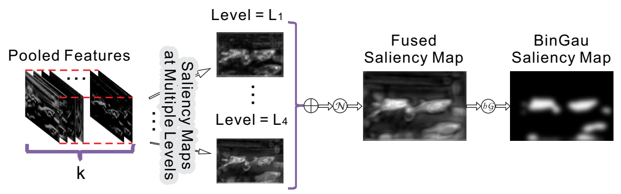
In our work, we explore more filters with varying sizes, so that both the inner-object textures and the object itself can be captured as salient regions. In this work, we choose four scales, $S_1 = 10$, $S_2 = 15$, $S_3 = 20$ and $S_4 = 50$, to produce four filters; then, for an image of $h \times w$-pixel resolution, specifically, each grid size in scale S_1 has size of $\frac{h}{S_1} \times \frac{w}{S_1}$. Therefore, for the filter of the first scale, we have $s = \frac{h}{S_1}$ and $t = \frac{w}{S_1}$. We use these four filters to convolve each of the $k = 80$ low-level response maps to produce 320 mid-level feature maps in total, as illustrated by Figure 3. Note that the resultant maps produced by the sizing filters have different sizes and thus produce bound effects, so we pad the smaller maps with zero values around the map bound.

Multi-level Saliency Calculation

Using the mid-level feature maps, we now calculate intermediate saliency maps at multiple levels. The multi-level saliency maps mean that the local contrast is defined at regions with different sizes. Each level is defined as the number of surrounding laps. For instance, we calculate the local contrast between 2 laps of surrounding patches with the center patch when the level is 2. This can also help to capture both the inner-object saliency and the salient object itself.

To this end, we also define a set of filters of different sizes for calculating local contrast values. Convolving these filters with the mid-level feature maps results in a set of intermediate saliency maps at multiple levels of local contrasts. In this paper, we define four levels of filters, which are demonstrated by Figure 4 (a). When convolving the mid-level feature maps at a lower level, *i.e.* a small region is compared within a small neighborhood, subtle inner-object saliency


Level = 1
Level = 2
Level = 3
(a) Filters for local contrast calculation at multiple levels



(b) Saliency Map Calculation at Multiple Local Contrast Levels

Figure 4: (a) Some predefined filters for local contrast calculation at multiple levels; (b) Calculation of intermediate saliency maps over the mid-level feature maps at multiple local contrast levels. Note that in (a), the square grids are not pixels, and the size of grids at higher level is larger than that at lower level. Moreover, "1" and "-1" means the two local areas/grids are used to calculate contrast values. The final saliency map is obtained by summing up all the intermediate maps at all the pooling scales and all the local contrast levels. We also present the map after binarization and Gaussian blurring.

can be detected, such as characters in a vehicle plate. While at a higher level, *i.e.* a local region has a larger surrounding neighborhood for contrast calculation, the overall salient object can be detected, such as a vehicle plate placed on a table as the background. Thus, all these intermediate saliency maps will contribute to detecting both inner-object saliency and the salient object itself.

Mathematically, with a defined filter $\mathbf{p}_l$ to calculate local contrast values of mid-level feature maps at a specific level denoted by l. Specifically, we convolve all the m feature maps $\mathbf{X}_i^{(s)}$ $(i = 1, \ldots, m)$ at a particular scale s with $\mathbf{p}_l$ to obtain the corresponding saliency map $\mathbf{Y}_l^{(s)}$:

$$\mathbf{Y}_l^{(s)} = \sum_{i=1}^{m} \mathbf{p}_l * \mathbf{X}_i^{(s)}, \qquad (1)$$

where $*$ indicates the convolution operation.

Saliency Maps Fusion with Priors

Since we obtain a set of intermediate saliency maps at different pooling scales and different local contrast levels, we now turn to fuse them into the final saliency map with vision prior. In particular, we merely consider the simple center-bias prior (Tatler 2007), which can be seen in Figure 1. Please note that other high-level priors are used in literature, such as warm colors (Rother, Kolmogorov, and Blake 2004), face priors (Judd et al. 2009), etc. But we do not adopt these principles in our current work.

There are several ways to fuse the intermediate maps, such as "add", "product" and "max". But empirically we find "add" fusion generates the best result, so we choose "add" fusion throughout our work.

Moreover, we incorporate the center-bias prior in this fusion stage. In particular, the specific prior is generated by a 2D Gaussian heat map $\mathbf{W}$ as shown by the upper-right panel in Figure 1. Mathematically, the final saliency map $\mathbf{Z}$ is obtained by the following operation:

$$\mathbf{Z} = \mathcal{N}(\mathbf{W} \odot \sum_{l=1}^{m} \sum_{s=1}^{n} \mathcal{N}(\mathbf{Y}_l^{(s)})), \qquad (2)$$

where $\odot$ means Hadamard product between matrices, and $\mathcal{N}$ denotes the normalization operation (Itti, Koch, and Niebur 1998) that transforms all the values in the fused map to lie in the range of $[0, 1]$. Besides, we also get the map after binarization and Gaussian blurring.

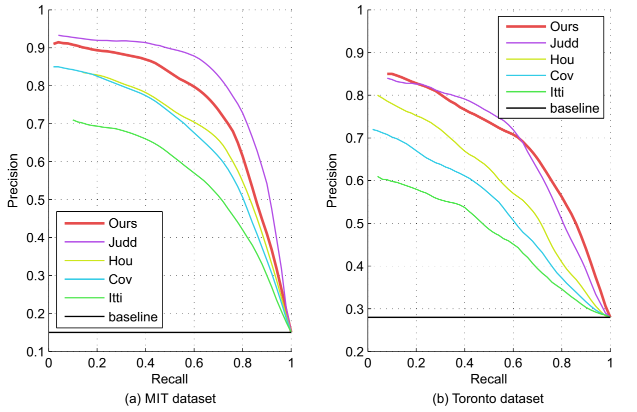


Figure 5: Smoothed precision-recall curve on MIT (left) and Toronto (right) datasets.

Experiments

In this section, we evaluate out method through experiments, and all the methods are carried out in MATLAB platform.

Datasets and Evaluation Metric

Datasets. The datasets used in our experiments are the MIT dataset (Judd et al. 2009) and the Toronto dataset (Bruce and Tsotsos 2005).

The MIT dataset contains 1003 images (resolution from 405×1024 to 1024×1024 pixels), including landscape, portrait and synthetic data. It is a challenging large dataset containing complex foreground and background scenes with very high resolutions.

Another challenging dataset is the Toronto dataset which is widely used. It contains 11 subjects with indoor and outdoor scenes. It includes 120 color images (resolution of 511×681 pixels) with complex foreground objects of different sizes and positions, of which a large portion do not contain particular regions of interest.

Precision-Recall Curve. Based on information retrieval, the precision-recall curve (PRC) has become a widespread conceptual basis for assessing classification performance

Table 1: Average AUC Score w.r.t. different saliency methods on MIT and Toronto datasets

datasets	**Ours**	Judd	Hou	Cov	Itti
MIT	**85.0%**	**86.9%**	80.4%	78.1%	61.7%
Toronto	**77.5%**	75.3%	69.7%	64.9%	58.0%

and saliency model evaluation (Brodersen et al. 2010). In PRC, the human fixations for an image are regarded as the positive set and some points randomly selected from the image are considered as the negative set. The saliency map is treated as a binary classifier to separate the positive set from the negatives (Borji and Itti 2012). The precision-recall curve is a relation of the positive predictive value and the true positive rate.

AUC Score. Another evaluation metric in our experiments is the area under the receiver operator curve (AUC) score (Bruce and Tsotsos 2005). The well-known receiver operating characteristic (ROC) curve is obtained by thresholding over the saliency map and plotting true positive rate vs. false positive rate. It often provides a useful alternative to the precision-recall curve. The AUC concentrates the performance of the two-dimensional ROC by a single scalar, as implied by its name; and the value of AUC is the area under the ROC curve. AUC is a good evaluation method for saliency models whose main advantages over other evaluation methods are its insensitiveness to unbalanced datasets and better insight into how well the saliency model is.

Experimental Setup

Size of Low-level Filters. We randomly extract 10000 patches from web downloaded images to train low-level filters. In view of the image size involved, the size of each sampling color patch is set to $8 \times 8 \times 3$. We perform k-means on the preprocessed sampling patches. The resultant filter bank is composed of a set of filters each of length $192 = 8 \times 8 \times 3$. The number of k-means filters is 80, thus we can get 80 response maps after convolution.

Threshold parameter. Thresholding on the response maps greatly improve the saliency performance. Here we set the threshold parameter to be $5/8$. In other words, when the length of the response maps is 80, we zero out the smallest $80 * (1 - 5/8) = 30$ feature values of each convolved feature vector.

Pooling Grid Scales. When we summarize features over regions of the convolved features, we use multi-scale average pooling to capture salient objects of different scales. The pooling grid scales consist of four versions, namely, 10×10, 15×15, 20×20 and 50×50.

Saliency Levels. We perform saliency computation on the mid-level feature maps on multiple numbers of surrounding laps, namely, the saliency computational levels, which are set to 1, 3, 6 and 9.

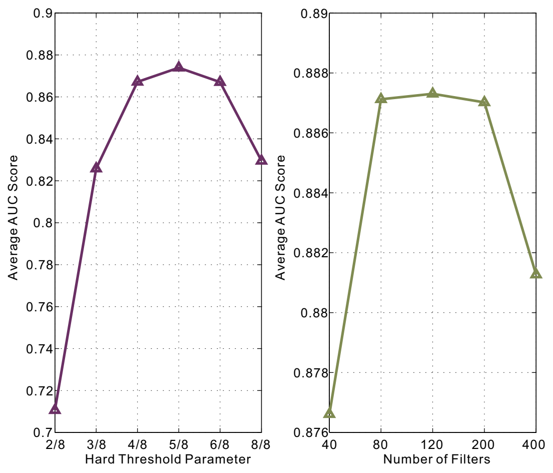


Figure 7: Left: Average AUC score with respect to different threshold parameters. The x-axis is the threshold parameter and the y-axis is the corresponding average AUC score on some randomly selected images. Right: Average AUC score with respect to different numbers of filters. The x-axis is the number of filters.

Experimental Results

The saliency models we compare with are: (1) Judd's model (Judd et al. 2009) is a supervised learning model which combines low-level local energy features, mid-level gist features and the high-level face and person detectors for saliency computation. (2) Hou's model (Hou, Harel, and Koch 2012) uses "image signature" as an image descriptor which contains information about the foreground of an image. (3) Cov's model (Erdem and Erdem 2013) uses covariance matrices of simple image features extracted from local image patches as meta-features. Region covariances as low-dimensional representations of image patches capture local image structures. (4) Itti's model (Itti, Koch, and Niebur 1998) uses colors, intensity and orientations as the features to represent image locals.

Figure 5 plots the performances of all the compared methods and ours based on the precision-recall curve (PRC). The results in Figure 5 show that Judd's and our models show higher performance than the rest on MIT and Toronto dataset, respectively. Besides, we compare the saliency performance using the AUC score. The average AUC score with respect to these saliency models is shown in Table 1. Among the compared saliency models, our models lead to higher performance than others although it is not perfect. Because our model uses adaptive mid-level features based on low-level filters which are obtained by unsupervised learning k-means and capture the texture and color information simultaneously, while the mid-level features of other models are not adaptive. Our model computes saliency through a unified multi-scale and multi-level framework, thus capturing both inner-object textures and the salient object itself, while others compute saliency on limited scales.

Figure 6 shows the visual saliency maps on the MIT dataset generated by different models. Ground truth saliency

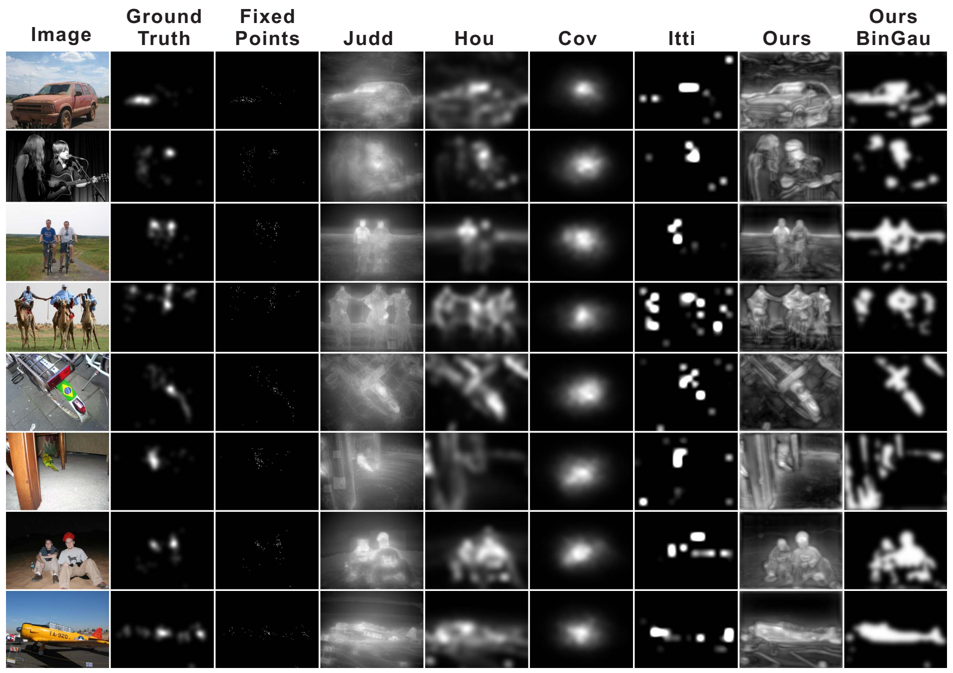


Figure 6: Saliency maps of different models on MIT dataset. Each row corresponds to an example. The columns are the original images, ground truth images, human fixation points and saliency maps with different models, respectively.

maps and human fixation points are also provided. The saliency maps of our model and the Gaussian smoothed binary saliency maps are given in the last two columns. The rest four columns are saliency maps obtained by other models, *i.e.* Judd, Hou, Cov and Itti. White regions indicate intense saliency of that patch, while black regions indicate low saliency. The white regions in our saliency maps are exactly the noticeable parts which draw our attention. Among these saliency maps generated by different models, we can see that our model captures salient parts in the image, including heads or bodies of people, without additional efforts such as human face detection, object recognition, etc. The Gaussian smoothed binary saliency maps of our model explicitly accord with the ground truth and human fixation points and these maps highlight the important areas of our attention.

Parameter Analysis

Here we discuss two important parameters, namely, the threshold parameters and the numbers of k-means filters.

Threshold has a notable influence on the performance of our saliency computation while it is not widely used in saliency detection. Through our experiments, we find that hard thresholding on the response maps indeed boosts the performance. We run experiments with different threshold parameters on randomly selected images on MIT and Toronto datasets. The hard threshold parameters we compare with are $2/8$, $3/8$, $4/8$, $5/8$, $6/8$ and $8/8$. We run five times to calculate the average and the experimental results are shown in the left panel of Figure 7, where the x-axis is the threshold parameter value and the y-axis is the average AUC score obtained on these thresholds. It is clear that the best saliency performance is achieved when the threshold parameter is $5/8$. No thresholding (threshold parameter equals $8/8$) and excessive thresholding (threshold parameter equals $2/8$) give worse results.

The performances on different numbers of k-means filters are also compared. As is shown in the right panel of Figure 7, the highest performance is achieved when the number k is between 80 and 200. For sake of efficiency, we should use as few filters as possible to represent a patch in the image, and we adopt $k = 80$. This also mimics the human nerve/cognitive system for receiving and processing visual information (O'Rahilly and Müller 1983).

Conclusion and Future Work

We propose a model for saliency detection which uses mid-level features on the basis of low-level *k*-means filters within a unified deep framework in a convolutional manner. This saliency model calculates the multi-scale and multi-level saliency maps and takes the "add" fusion operation to obtain final saliency map. Our model can be applied in a parallel manner, which makes it possible for fast computation. Through experimental validations, we demonstrate that the mid-level features are adaptive and the saliency model exhibits high performance.

Our saliency model merely consider the low-level *k*-means filters and mid-level features. Possible future work includes more proper features design or the combination of them. Image matching based on mid-level features which are generated by our multi-scale deep framework and benefit from the properties of adaptiveness and robustness can be further studied.

Acknowledgments

This work is partly supported by the 973 Program (No.2010CB327904) and National Natural Science Foundations of China (No.61071218) as well as CKCEST project.

References

Borji, A., and Itti, L. 2012. Exploiting local and global patch rarities for saliency detection. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 478–485. IEEE.

Boureau, Y.-L.; Bach, F.; LeCun, Y.; and Ponce, J. 2010. Learning mid-level features for recognition. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2559–2566. IEEE.

Boureau, Y.-l.; Cun, Y. L.; et al. 2007. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems*, 1185–1192.

Brodersen, K. H.; Ong, C. S.; Stephan, K. E.; and Buhmann, J. M. 2010. The binormal assumption on precision-recall curves. In *Proceedings of the Twentieth International Conference on Pattern Recognition (ICPR)*, 4263–4266. IEEE.

Bruce, N., and Tsotsos, J. 2005. Saliency based on information maximization. In *Advances in Neural Information Processing Systems*, 155–162.

Cheng, M.-M.; Zhang, G.-X.; Mitra, N. J.; Huang, X.; and Hu, S.-M. 2011. Global contrast based salient region detection. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 409–416. IEEE.

Coates, A., and Ng, A. Y. 2012. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*. Springer. 561–580.

Coates, A.; Ng, A. Y.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, 215–223.

Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 886–893. IEEE.

Erdem, E., and Erdem, A. 2013. Visual saliency estimation by nonlinearly integrating features using region covariances. *Journal of Vision* 13(4).

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Hou, X.; Harel, J.; and Koch, C. 2012. Image signature: Highlighting sparse salient regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(1):194–201.

Itti, L.; Koch, C.; and Niebur, E. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(11):1254–1259.

Itti, L. 2004. Automatic foveation for video compression using a neurobiological model of visual attention. *IEEE Transactions on Image Processing* 13(10):1304–1318.

Jahne, B. 2002. *Digital Image Processing*. Springer.

Judd, T.; Ehinger, K.; Durand, F.; and Torralba, A. 2009. Learning to predict where humans look. In *Proceedings of the 2009 IEEE Twelfth International Conference on Computer Vision (ICCV)*, 2106–2113. IEEE.

Ko, B. C., and Nam, J.-Y. 2006. Object-of-interest image segmentation based on human attention and semantic region clustering. *Journal of the Optical Society of America A* 23:2462–2470.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1106–1114.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lee, H.; Grosse, R.; Ranganath, R.; and Ng, A. Y. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 609–616. ACM.

Lin, Y.; Fang, B.; and Tang, Y. 2010. A computational model for saliency maps by using local entropy. In *Proceedings of the 2010 AAAI Conference on Artificial Intelligence*.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2):91–110.

O'Rahilly, R., and Müller, F. 1983. *Basic human anatomy: a regional study of human structure*. Saunders.

Rother, C.; Kolmogorov, V.; and Blake, A. 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, 309–314. ACM.

Shen, X., and Wu, Y. 2012. A unified approach to salient object detection via low rank matrix recovery. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 853–860. IEEE.

Tatler, B. W. 2007. The central fixation bias in scene viewing: Selecting an optimal viewing position independently of motor biases and image feature distributions. *Journal of Vision* 7(14).

Walther, D., and Koch, C. 2006. Modeling attention to salient proto-objects. *Neural Networks* 19(9):1395–1407.

Yan, Q.; Xu, L.; Shi, J.; and Jia, J. 2013. Hierarchical saliency detection. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Yang, J., and Yang, M.-H. 2012. Top-down visual saliency via joint crf and dictionary learning. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2296–2303. IEEE.

Yang, J.; Yu, K.; Gong, Y.; and Huang, T. 2009. Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1794–1801. IEEE.