Supplementary Material to Technical Report
Low-Level Salient Region Detection Using Multi-Scale Filtering

1 Introduction

Here, we present further analysis on the multi-scale filtering. We first present another example (synthetic) to illustrate the multi-scale filtering. We then comment on the computational complexity, and finally show additional saliency maps.

2 The Frequency Domain Analysis of Multi-Scale Filtering

In Section 3 of the paper, we analyze the multi-scale filters we use for salient region detection. In addition, we show in Figure 2(a) of the paper that combining filters allows us to manipulate the bandwidth and the center frequency of a filter. Here, in order to visualize the filtering operation, we apply our filters to a frequency pattern (64×64) in Figure 1 (top-right corner). In the pattern, the spatial frequency is equal to zero at the image center and increases radially. As we can see from Figure 1, it is possible to adjust the frequency information that we want to extract from the image.

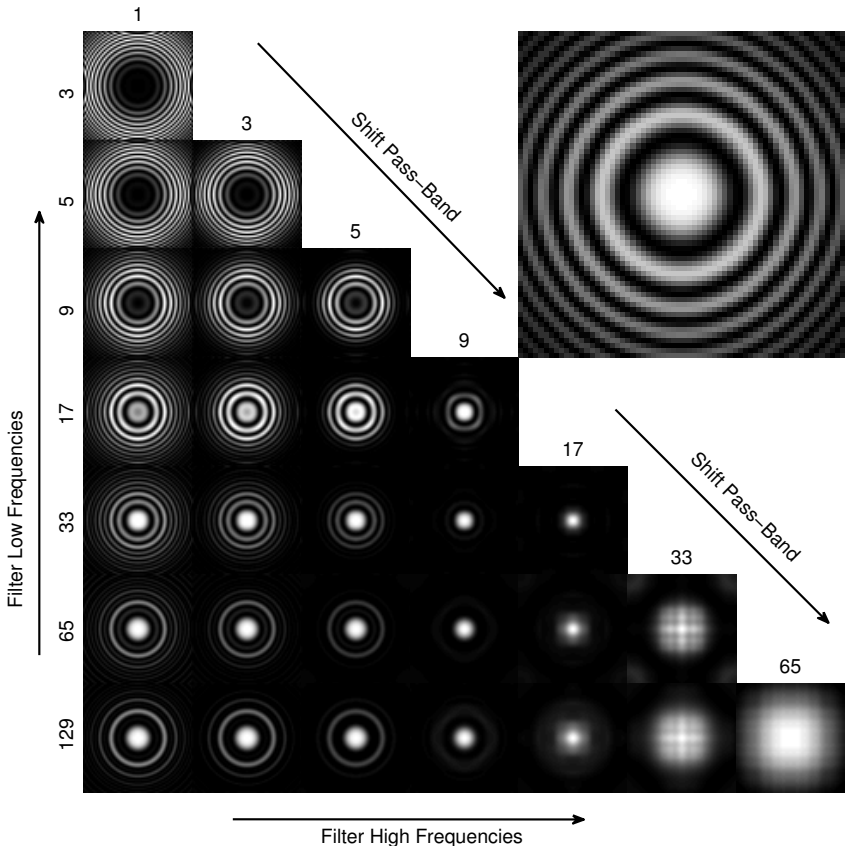


Figure 1: Multi-scale filtering on a frequency pattern (top-right corner). The numbers represent the size of the filters that are combined to get a specific band-pass filter. For example, the filter on the bottom-left cornered is obtained by taking the difference of the filters with size 1 and 129 (Best viewed via zooming on a computer screen to avoid aliasing).

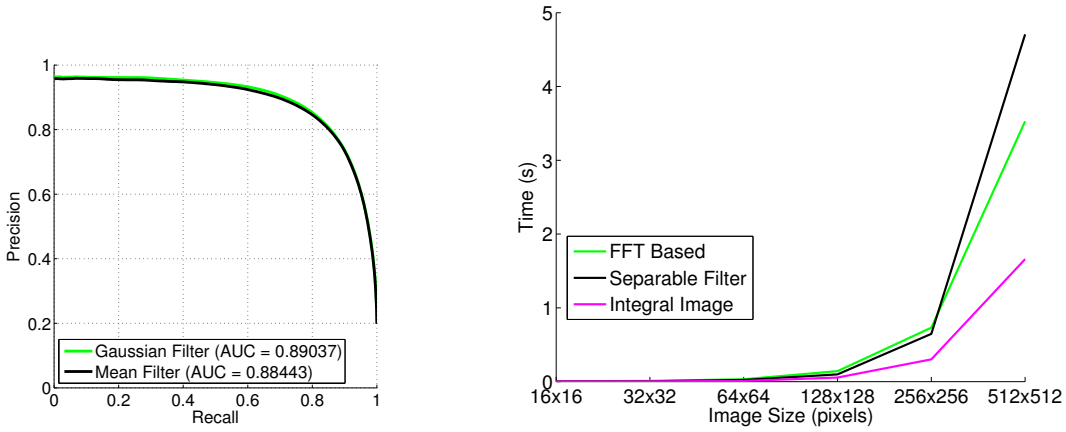
3 Computational Complexity

Our algorithm requires a series of filtering and combination operations. In addition, in Section 4.1 of our paper, we define the maximum filter size as the smallest filter that is larger than the image. Thus, the processing time of our method increases with increased resolution.

In order to assess the trade-off between speed and accuracy, we implement three different methods in MATLAB to compute the multi-scale filters. The first method is based on the Fast Fourier Transform (FFT) and it performs the filtering in frequency domain. The second method uses the separability property of the 2D Gaussian filters and operates in spatial domain. The final method employs mean filters (all of the filter coefficients are equal) instead of Gaussian filters, which can be efficiently computed using integral images.

As we can see from Figure 2(a), mean filtering is slightly worse than Gaussian filtering (the performances of the FFT-based and the separable filter methods are equal). As far as the processing time is concerned, the integral image approach is the fastest method among the three implemented methods. The FFT-based method is slightly slower than the separable filters for small images. However, it gets more efficient as the image size increases and becomes faster when the input image is larger than 256×256.

For an input image of size $N \times N$, the computational complexity of filtering in the frequency domain is $\mathbf{O}(N^2 \log_2(N)) + \mathbf{O}(N^2)$, where the first term is for the FFT, and the second term is for the element-wise multiplication for filtering. If we use the separable filters, the complexity becomes $\mathbf{O}(N^2 L)$, where L is the filter length. As we mention before $L_{\max}$ depends on the image size ($N < L_{\max} \leq 2N$). Thus, the FFT-based method works faster for large images. Consequently, in order to balance between accuracy and speed, we use the FFT-based method in our paper.



(a) (b)

Figure 2: (a) The performance comparison of Gaussian and mean filters using the MSRA-1000 dataset [1] (b) The processing time comparison of three filtering methods (one image, in MATLAB).

The comparison in terms of the processing time of each method is given in Table 1. The processing time of other algorithms are presented in [3] and are computed using an Intel Core i7-920 2.6 GHz with 3GB RAM. For our method, we use an Intel Core i7 2.0 GHz with 8GB RAM. As our code runs in MATLAB, it appears to be the slowest. However, optimizing the algorithm in C++ can dramatically reduce the processing time, due to the simple filtering structure.

Table 1: Processing time comparison of different methods.

Method	RC [2]	SF [1]	LR [4]	FT [1]	Ours
Time (s)	0.144	0.153	NA	0.012	3.112
Code	C++	C++	-	C++	MATLAB

4 Additional Saliency Maps

In Figure 10 of the paper, we compare our performance with the state-of-the-art algorithms [1, 2, 3, 4] using 9 different images from the MSRA-1000 dataset [1]. We explain the differences in Section 5 of our paper. Here, we provide additional comparisons in Figure 3-7.

The examples in Figure 3 show that our method is not affected by the segmentation errors we mention in Section 1 of our paper, and it computes relatively uniform saliency maps.

In Figure 4, due to clear color contrast, all of the methods are accurate. In Figure 5, the foreground objects have similar colors with the corresponding backgrounds. This problem creates false negatives and positives on the final saliency maps for all low-level methods. In the first row of Figure 6, high-level object information is required to understand that the signpost is the salient object. Thus, all of the algorithms fail to detect the salient regions. In the second row, all methods detect the whole tower as salient. However, the annotation of the ground truth only shows the top of the tower, which illustrates that ground truth annotation can be subjective. The examples in Figure 5 and 6 summarize the limitations of low-level saliency detection methods.

In the first row of Figure 7, due to the multi-colored pattern on the balloon, our algorithm estimates a non-uniform saliency map. This problem arises when average local color is not correctly estimated by multi-scale filters. This problem can be solved by incorporating high-level *objectness* information to the detected salient region. In the remaining rows, our method produces false positives in the background. This is due to the *leakage* of weak saliency maps that are not properly eliminated by our compactness measure defined in Section 4.2 of the paper.

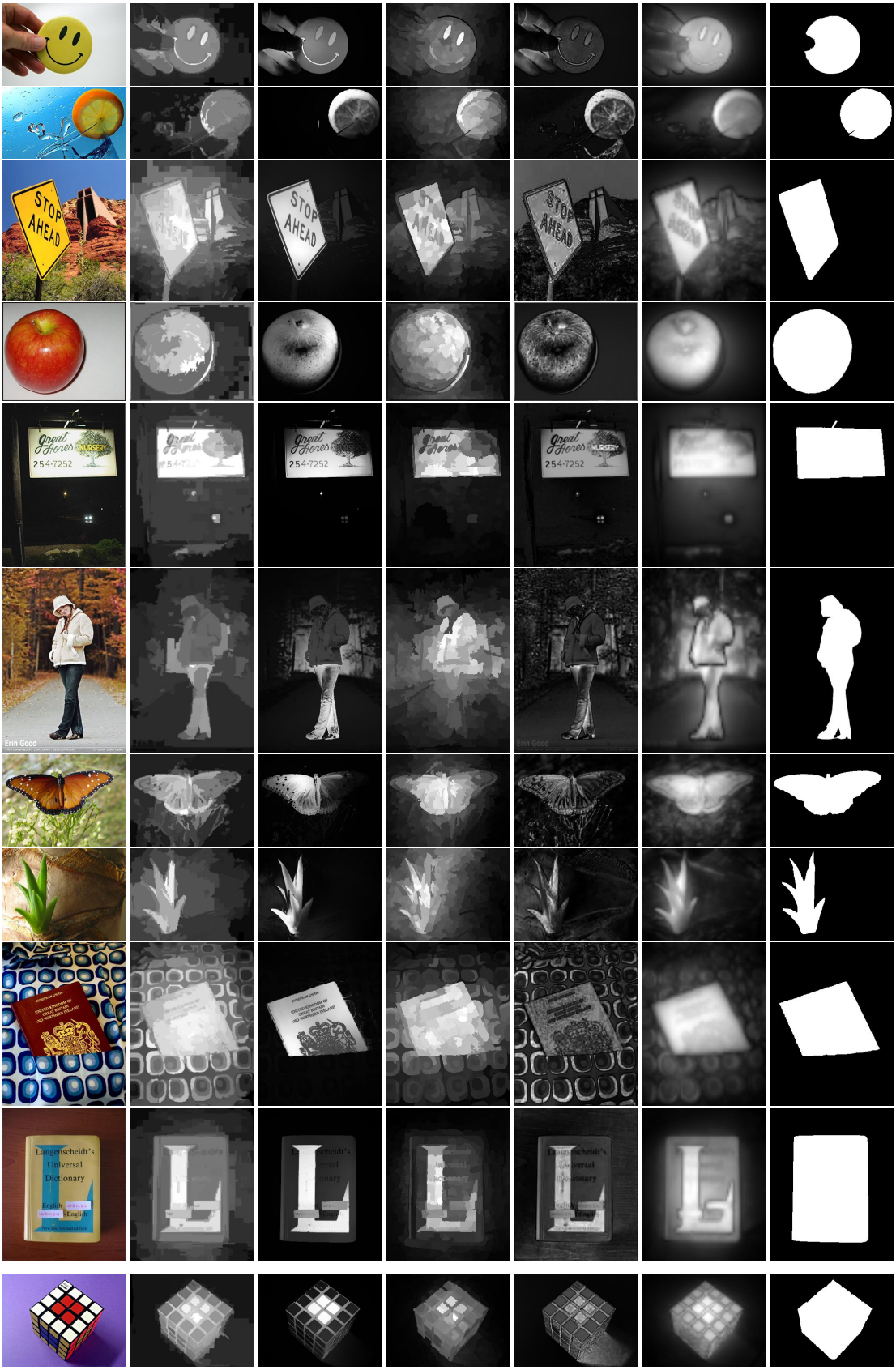
References

[1] R. Achanta, S. Hemami, F. Estrada, and S. Süsstrunk. Frequency-tuned Salient Region Detection. In *Proceedings of IEEE CVPR*, pages 1597 – 1604, 2009.

[2] M. Cheng, G. Zhang, N. J. Mitra, X. Huang, and S. Hu. Global contrast based salient region detection. In *Proceedings of IEEE CVPR*, pages 409–416, 2011.

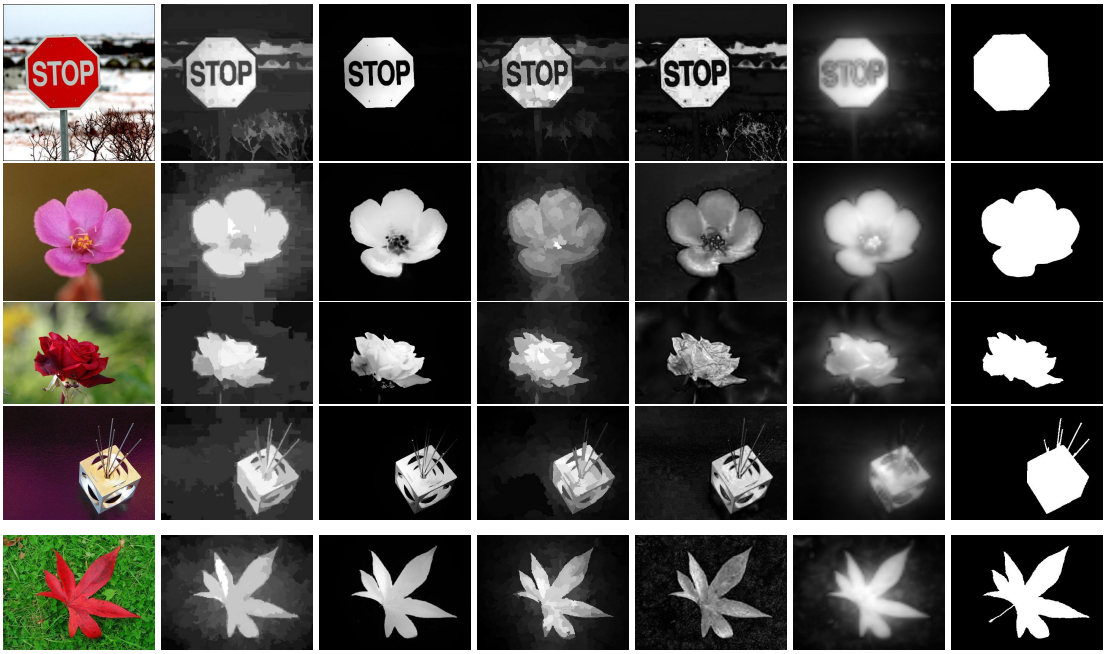
[3] F. Perazzi, P. Krahenbuhl, Y. Pritch, and A. Hornung. Saliency filters: Contrast based filtering for salient region detection. In *Proceedings of IEEE CVPR*, pages 733–740, 2012.

[4] X. Shen and Y. Wu. A unified approach to salient object detection via low rank matrix recovery. In *Proceedings of IEEE CVPR*, pages 853–860, 2012.



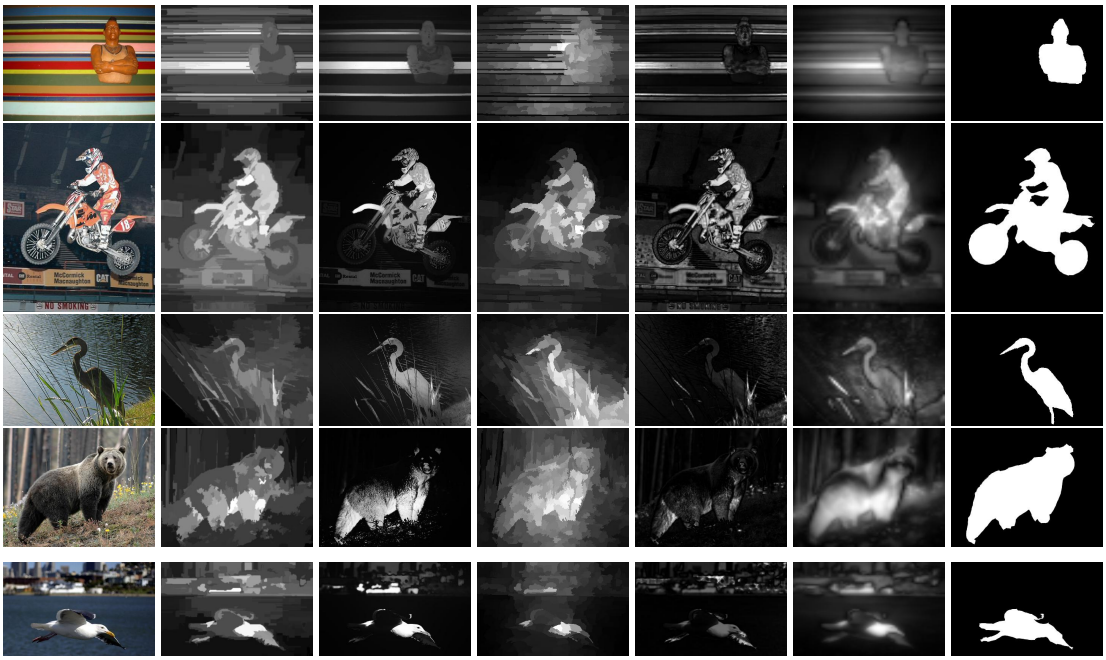
(a) Original (b) RC [2] (c) SF [3] (d) LR [4] (e) FT [1] (f) Ours (g) Ground Truth

Figure 3: Example images from MSRA-1000 dataset [1] and corresponding saliency maps for different methods, where our algorithm produces accurate and uniform saliency maps.



(a) Original (b) RC [2] (c) SF [3] (d) LR [4] (e) FT [1] (f) Ours (g) Ground Truth

Figure 4: Example images from MSRA-1000 dataset [1] and corresponding saliency maps for different methods, where the performances of all methods are comparable.

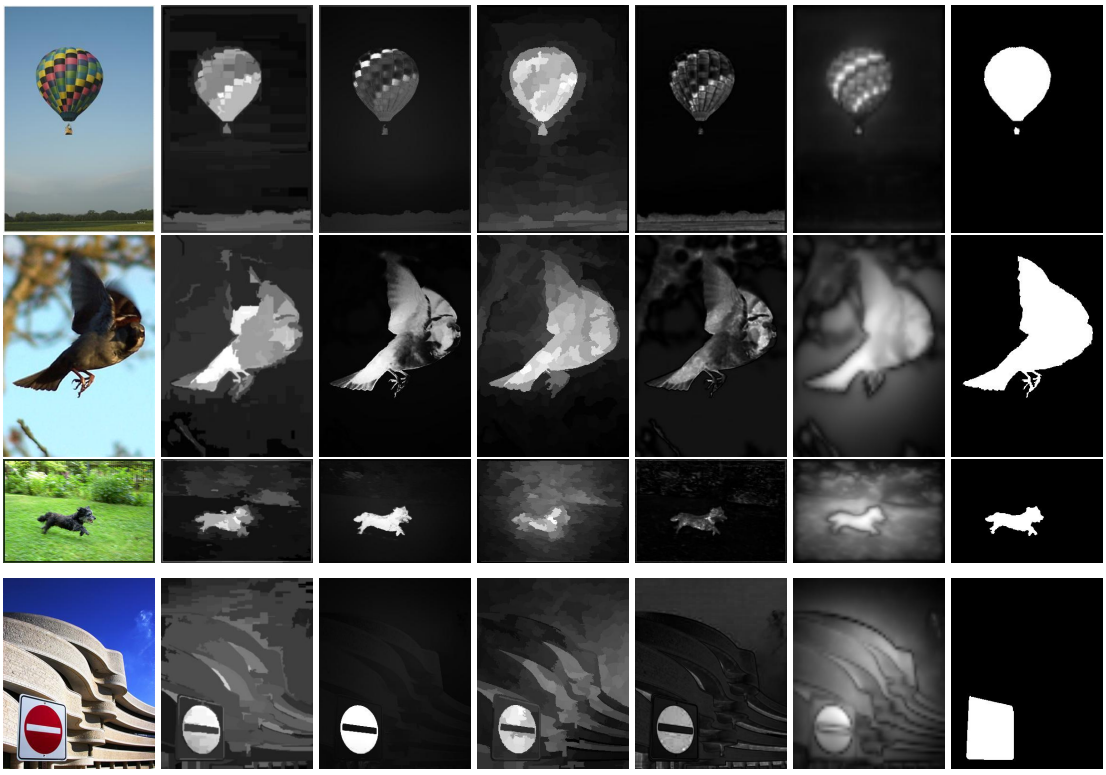


(a) Original (b) RC [2] (c) SF [3] (d) LR [4] (e) FT [1] (f) Ours (g) Ground Truth

Figure 5: Example images from MSRA-1000 dataset [1] and corresponding saliency maps for different methods, where saliency detection is difficult due to the color similarity between the foreground objects and the backgrounds.

(a) Original (b) RC [2] (c) SF [3] (d) LR [4] (e) FT [1] (f) Ours (g) Ground Truth

Figure 6: Example images from MSRA-1000 dataset [1] and corresponding saliency maps for different methods, where high-level information is required (the first row) or ground truth annotation is subjective (the second row).



(a) Original (b) RC [2] (c) SF [3] (d) LR [4] (e) FT [1] (f) Ours (g) Ground Truth

Figure 7: Example images from MSRA-1000 dataset [1] and corresponding saliency maps for different methods, where our algorithm produces relatively inaccurate saliency maps.