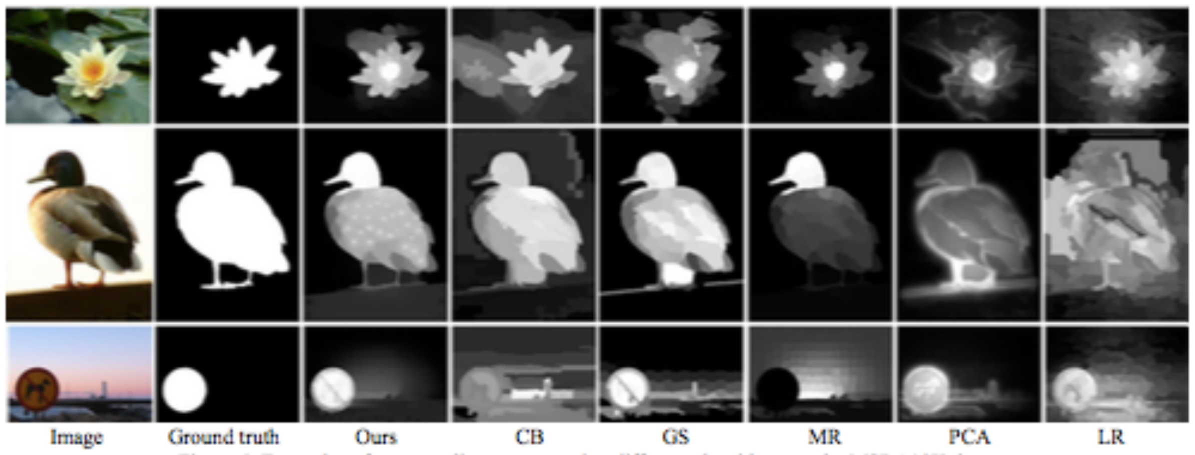
Final Report

Yunke Tian
109929662

Overview



My course project is to implement a paper “Robust Saliency Detection via Regularized Random Walk.

Saliency detection is ubiquitously needed for many purposes. However, some existing method is not accurate on the detected area. This problem is especially severe when superpixel segmentation itself is not precise. On the other hand, treating superpixel as unit will lose details within each superpixel.

The main contribution of this method is:

- 1. In the background saliency detection phase, use manifold ranking to remove one erroneous boundary of the four → Accurate & Robust

- 2. While measuring foreground saliency, use regularized random walks ranking to formulate pixel-wised saliency maps → accurate & detail preserved.

Basically there are 3 phases of the implementation: Background saliency detection, foreground saliency detection and saliency map formulation by regularized random walks ranking.

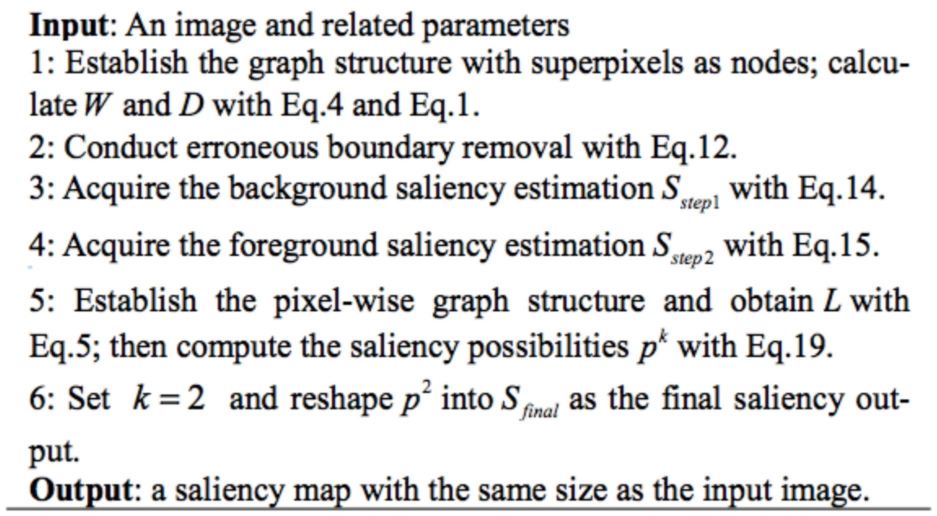
I implemented this method on Matlab. The input is a normal image with saliency, and then automatically output the robust saliency map with more precise details.

After it, I used F-measure to test the accuracy of the result.

Currently the parameters are not adjusted to the best, so the system is not working too satisfying, but it works fine.

Process

Actually, the algorithm is already given in the paper:



Input: An image and related parameters
1: Establish the graph structure with superpixels as nodes; calculate W and D with Eq.4 and Eq.1.
2: Conduct erroneous boundary removal with Eq.12.
3: Acquire the background saliency estimation S_{step1} with Eq.14.
4: Acquire the foreground saliency estimation S_{step2} with Eq.15.
5: Establish the pixel-wise graph structure and obtain L with Eq.5; then compute the saliency possibilities p^k with Eq.19.
6: Set $k=2$ and reshape p^2 into S_{final} as the final saliency output.
Output: a saliency map with the same size as the input image.

However, the given algorithm is pretty rough, and there're many hidden details.

Generally, the whole process consists of 15 steps:

- **Foreground Saliency Detection**
- 1. Establish the graph structure with superpixels as nodes (SLIC: simple linear iterative clustering)
- 2. Compute weight matrix between superpixel nodes Wij
- 2.a. Decide connectivity between superpixel nodes

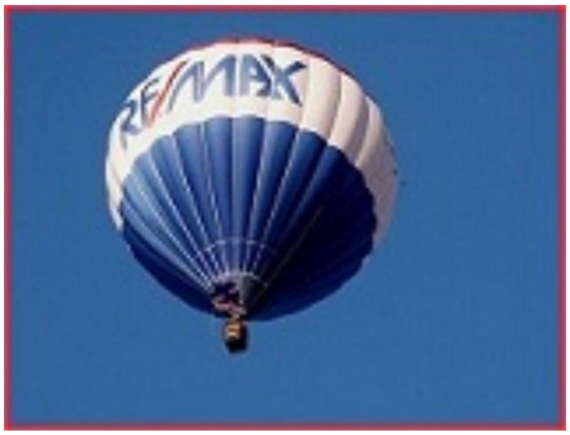
- 2.b. Convert image from RGB to Lab and compute Lab average for every node
- 2.c. Compute weight matrix
- 3. Sum Wij according to columns → di
- 4. Erroneous boundary removal
- 4.a Determine boundary superpixels
- 4.b Histogram boundary superpixels using 256 bins
- 4.c Compute Euclidean distance between histograms and remove the most faraway boundary.
- 5. Compute manifold ranking values of the 3 remaining boundaries, f
- 6. Compute S_step1 which indicate the background relevance of each node.
- **Foreground Saliency Detection**
- 7. Thresholding S_step1 and compute foreground saliency estimation S_step2.
- **Saliency Map Formulation by Regularized Random Walks Ranking**
- 8. Establish the pixel-wise graph structure and compute Wuv, Du as step (2) and (3)
- 9. Compute Laplacian matrix L
- 10. Differentiate Dirichlet integral
- 11. Use the S_step2 as seeds and categorize seeds into background and fore ground.
- 12. With regard to foreground seeds, perform random walk ranking to compute the possibility of each non-seed to be foreground.
- 13. Re-forming the result into saliency map and output.
- 14. refine the output.

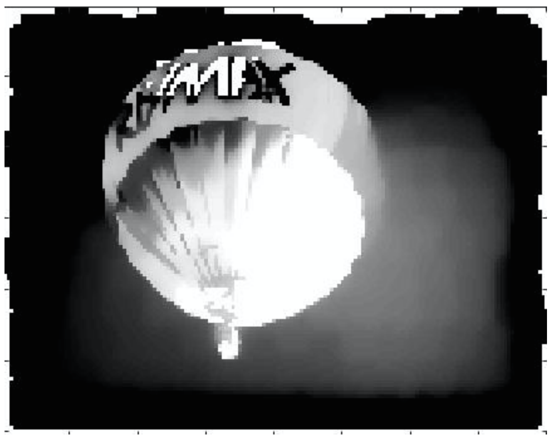
- 15. evaluate accuracy

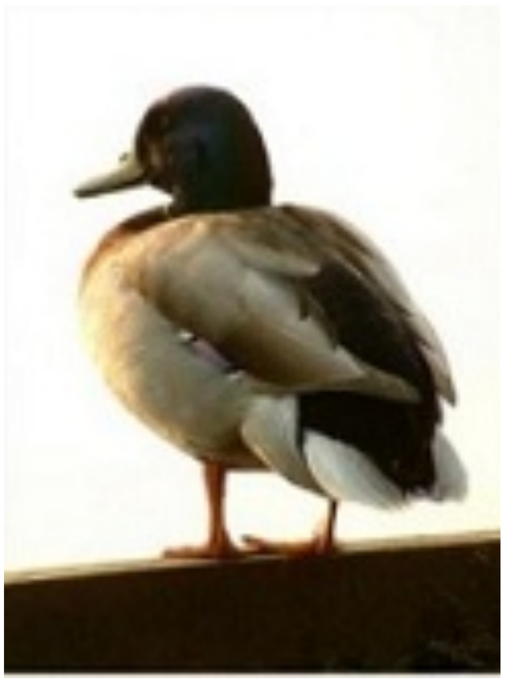
Outcome

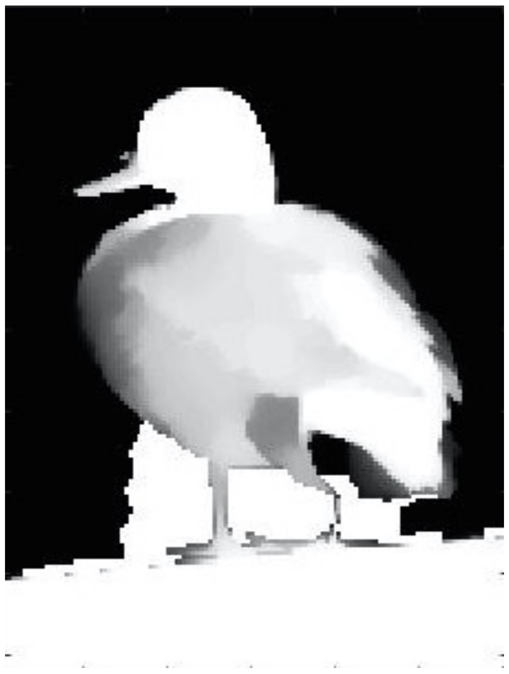
After all of steps, we can get the saliency map.

Below are two examples:









Though some small parts are not satisfying, it still shows the robust saliency map with some extent of details.

Evaluation is needed, so I chose the bird picture as the sample to evaluate. F-measure is used (set significance factor β^2 = 0.3):

$$precision = \frac{\sum_{i=1}^{N} G(i) \cdot S_{final}(i)}{\sum_{i=1}^{N} S_{final}(i)}$$

$$recall = \frac{\sum_{i=1}^{N} G(i) \cdot S_{final}(i)}{\sum_{i=1}^{N} G(i)},$$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

and computed precision = 0.5276, recall = 0.7638 and F = 0.5642.

F = 0.5642 means that the saliency detection method works just satisfactory under my version of parameters.

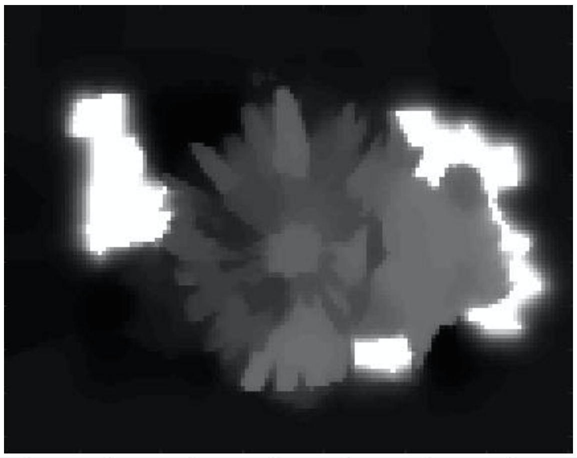
Notes

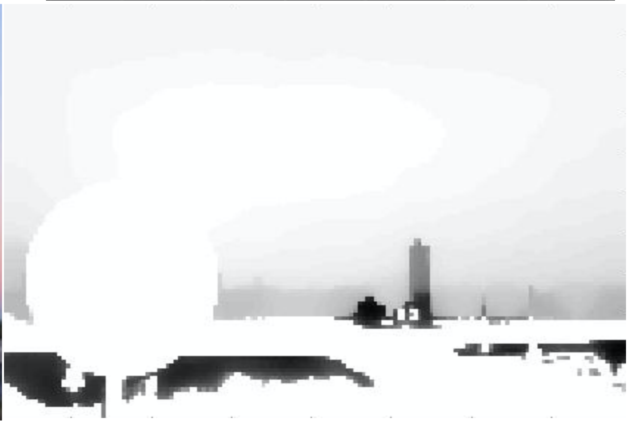
Difficulties I came across:

1. Equations given in the paper are not concrete enough. Therefore, there are variation for some equations. I have to fix them with experiments one by one. One formula is even wrong:

$$F_\beta = \frac{(1 + \beta^2 precision \cdot recall)}{\beta^2 precision + recall}$$

2. Tuning parameters. When I first finish the code, it could never generate good results.



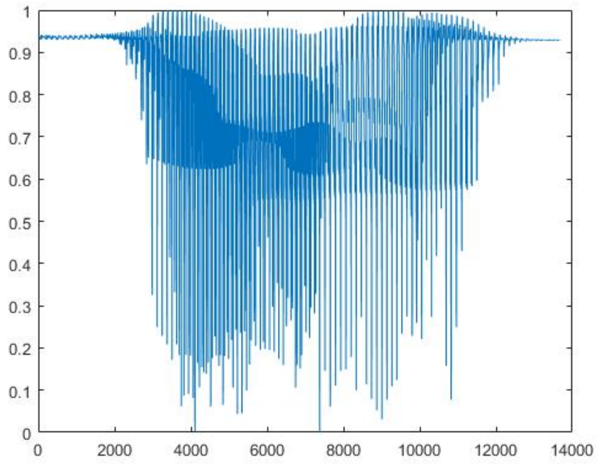


I have to tune the 2 parameters for the final saliency map function:

$$p_U^k = \left(L_U + \mu I\right)^{-1}\left(-B^T p_M^k + \mu Y_U^k\right)$$

Moreover, because of uncertainties of other equations, actually here I need to set 4 parameters and tune them. Luckily, I got satisfactory results after many tries.

3. Output adjustment. Even strictly follow the instructions in the paper, still the output image is far different from those in the paper. The problem is that the background part is not dark at all and saliency part is not bright enough.
   Thus I generate the brightness distribution plot: (for the balloon picture)



I should normalize the brightness value to t_min ~ t_max, which depends on the feature of the plot. (t_min = 0.5 and t_max = 0.9 in this case)

Other notes:

1. The size of the input image is really restricted: no more than 150*150. Otherwise, my laptop with 16GB RAM is not enough to run. The most costly codes are those pixel-wise matrix with the size of numOfPixels × numOfPixels.
2. With 150*150 image, one run takes about 5 minutes.
3. In this method, superpixel segmentation (vl_slic) and rgb2Lab are from libraries.
4. Actually there are existing matlab codes already implemented:
   http://sydney.edu.au/engineering/it/~yy ua4798/cvpr2015/
   This was pointed out in my midterm presentation, and I hadn't noticed this link before that presentation. I haven't read this implementation, and I think my code should be totally different from theirs.
5. 350-400 lines of Matlab code.

Reference

[1] Changyang Li, Yuchen Yuan, Weidong Cai, "Robust Saliency Detection via Regularized Random Walks Ranking " CVPR, 2015.