Computer Vision Homework 2, Part2
Context-Aware Saliency Detection Implementation

Jiawei Hou
Shanghaitech University
houjw@shanghaitech.edu.cn

Abstract

I read the context-aware saliency algorithm [1], which detects the salient regions based on four principles in psychological evidence, and use Matlab to implement the algorithm. In this report, I will describe the algorithmic process and discuss the parameters setting. Finally, I will show that my program runs well results.

1. Introduction

The context-aware saliency algorithm follows four basic principles supported by psychological evidence [4, 5, 2, 3]:

- Local low-level considerations, including factors such as contrast and color.
- Global considerations, which suppress frequently occurring features, while maintaining features that deviate from the norm.
- Visual organization rules, which state that visual forms may possess one or several centers of gravity about which the form is organized.
- High-level factors, such as human faces.

For the first principle, it needs to split the image into pieces, then calculate the color distances and position distances between patches. With these two important factors, we can estimate the saliency of each patch by calculating the ratio of these two distances as the dissimilarity between patches.

For the second principle, we change the scale of patches, and calculate the mean of saliency for every pixel. Under a big scale, background patches are easier to find similar patches while object patches are hard to be similar to others. For the third principle, we set a threshold for getting foci region, then re-compute the saliency of each pixel by weighting their original saliency according to its Euclidean distance to the closest attended pixel.

Finally, for the last principle, I use a face detector to improve the performance.

2. Program analysis

In this section we analyze the program step by step, and discuss the related parameters setting.

2.1. Split the image

Most of the calculations in this algorithm are based on pieces. So our first task is to divide the picture into pieces. One easy method is dividing the image into neat little squares directly, which is easy but the saliency pixel outline is not exactly the object's outline. The program use the super pixel segmentation to split the image into super pixels. This part just uses a mex file wrote by others.

Image segmentation results as follows:

Figure 1. Example: splits into 200 pieces

2.2. Estimate saliency of each patch

First, we calculate the Euclidean distance between the patches in CIE L*a*b color space, and normalized to the

range [0,1]:

$$d_{color}(p_i, p_j) = \sqrt{(L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2} \tag{1}$$

$$d_{position}(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

where $d_{color}(p_i, p_j)$ is the color distance in L*a*b* space between patch i and j, and $d_{position}(p_i, p_j)$ is the Euclidean distance between patch i and j.

Then we calculate the dissimilarity between patch i and j, which is proportional to the difference in appearance and inverse proportional to the positional distance:

$$d(p_i, p_j) = \frac{d_{color}(p_i, p_j)}{1 + d_{position}(p_i, p_j)} \tag{3}$$

where c = 2 in the program.

If the dissimilarity of patch i is high for all patch j, it must be salience. In the algorithm, we only choose K most similar patch to patch i to calculate its single-scale saliency (under the scale r):

$$S_i^r = 1 - exp\{-\frac{1}{K}\sum_{k=1}^{K} d(p_i^r, p_k^r)\} \tag{4}$$

where the parameter K is not a fixed number in the program, but changed according to the scale, which effectively improves the detection accuracy.

2.3. Multi-scale saliency enhancement

We use 4 scales in the program to calculate the saliency: $R = r, \frac{1}{2}r, \frac{1}{4}r, \frac{1}{8}r$. Saliency of patches under different scales are shown in Figure 2 and 3.

Then take the mean of the saliency at pixel i at different scales:

$$\bar{S}_i = \frac{1}{M}\sum_{r \in R} S_i^r \tag{5}$$

where M is the number of scales.

2.4. Saliency weighted by distance

According to the third principle, visual forms may possess one or several centers of gravity about which the form is organized [11]. That areas that are close to the foci of attention should be explored significantly more than faraway regions.

First, we find the foci by setting a saliency threshold T, which is also not a fixed parameter, but according to the scale. If the mean saliency of a pixel larger than T, then we think the pixel as a attended pixel.

Then, we give pixel saliency weight according to their distance to the foci to get the final computed saliency :

$$\hat{S}_i = \bar{S}_i(1 - d_{foci}(i)) \tag{6}$$

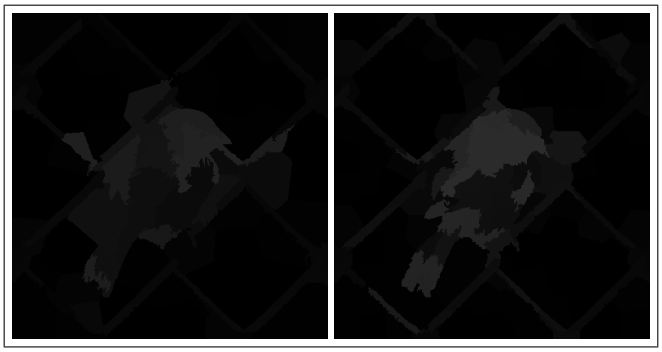

Figure 2. Left: scale=r; Right: scale=r/2

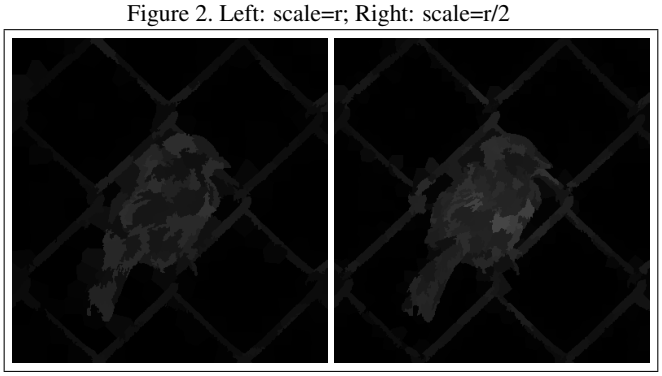

Figure 3. Left: scale=r/4; Right: scale=r/8

where $d_{foci}(i)$ is the Euclidean distance between pixel i and the closest foci pixel. This step takes a very long time.

2.5. High-level detection

As we all known, human face have much higher saliency, so the program use a face detector to improve the saliency of the face areas by setting their $d_{foci}(i) = 0$.

Here we compare the result using and without using face detector:

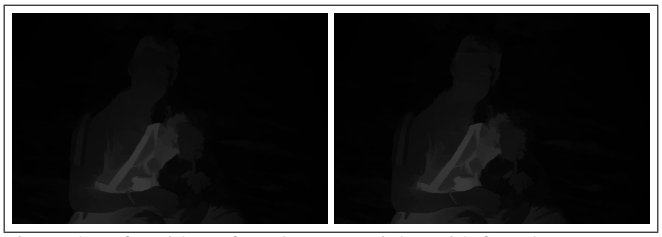

Figure 4. Left: without face detector; Right: with face detector

3. Result analysis

Here we use four images as inputs to test the code performance.

Figure 5 parameters: Size of image: 400*388; Number of patches: 100, 200, 400, 800, that $M = 4, R = lr$, where $l = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$; Coefficient before $d_position(i, j)$: $c = 2$;

Figure 5. Left: input; Right: my result

number of similar patches for calculating S_i: $K = 120/l$; Threshold: $T = 0.14 + 0.02 * l$.

Figure 6. Left: input; Right: my result

Figure 6 parameters: Size of image: 600*400; Number of patches: 100, 200, 400, 800, that $M = 4, R = lr$, where $l = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$; Coefficient before $d_position(i, j)$: $c = 2$; number of similar patches for calculating S_i: $K = 120/l$; Threshold: $T = 0.14 + 0.02 * l$.

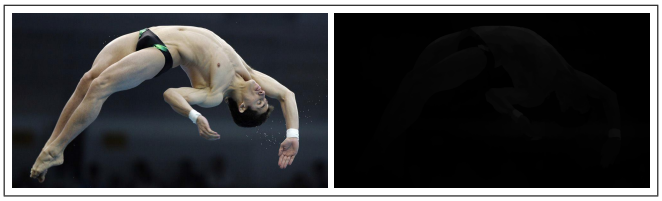

Figure 7. Left: input; Right: my result

Figure 7 parameters: Size of image: 547*992; Number of patches: 200, 400, 800, 1600, that $M = 4, R = lr$, where $l = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$; Coefficient before $d_position(i, j)$: $c = 2$; number of similar patches for calculating S_i: $K = 120/l$; Threshold: $T = 0.05 + 0.02 * l$.

Figure 8 parameters: Size of image: 547*992; Number of patches: 200, 400, 800, 1600, that $M = 4, R = lr$, where $l = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$; Coefficient before $d_position(i, j)$: $c = 2$; number of similar patches for calculating S_i: $K = 120/l$; Threshold: $T = 0.08 + 0.02 * l$, the performance has almost no difference compared with $T = 0.08$.

Figure 8. Left: input; Right: my result

References

[1] S. Goferman, L. Zelnik-Manor, and A. Tal. Context-aware saliency detection. In *Computer Vision and Pattern Recognition*, pages 2376–2383, 2010.

[2] C. Koch and T. Poggio. Predicting the visual world: silence is golden. *Nature Neuroscience*, 2(1):9, 1999.

[3] K. Koffka. Principles of gestalt psychology. *American Journal of Psychology*, 48(3):527, 1935.

[4] A. Treisman and G. A. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97–136, 1980.

[5] J. M. Wolfe. Guided search 2.0 a revised model of visual search. *Psychonomic Bulletin & Review*, 1(2):202–238, 1994.