CS 543 - A2: Scale-space Blob Detection

Yiwen Xu

March 7, 2016

Abstract

The goal of this assignment is to implement the Laplacian blob detection. In the project, two implementations of cascade Laplacian scale space with filtering have been used, namely using increase Laplacian kernel or using decrease image size. **See all results image in Appendix**.

1 Blob Detection Introduction

Blob Detection is used for detecting scale invariant regions in the image in which pixels share similar properties. The most common strategy for blob detection is to convolve gray scale image with Laplacian of Gaussian (LoG) with certain sigma value to attain Laplacian of Gaussian response under certain scale. The Local extrema in the response image shows the possible candidates of blob center.

The Blob Detection Algorithm mainly consist of two parts. First, iteratively apply LoG with different standard deviations to gray scale image in order to generate multi-layer scale space of LoG response. Next, given a stack of LoG response image where shows the different region of blob response, the work would be find the local extrema across space scale layer to locate the blob's center and radius.

2 Construct Scale Space

To construct Laplacian of Guassian response in different scale space, there are two distinguish way of doing so. One is to iteratively convolve grayscale image with LoG filter in bigger and bigger standard deviation. The other way is to down sample the grayscale image to make the LoG filter relatively bigger and then up sampling the response image to normal size.

Size of Pyramid: Assume the biggest size of blob regions we would like to detect has a radius of R, we would like to build a pyramid with N layers. Here is the basic idea of how we can build different size of LoG to convolve with different scales. Given a LoG with standard deviation of σ, the blob detected with max response has a radius of R_σ, where $R_\sigma = \sigma * \sqrt{2}$. As we plan to increase the sigma size each time by a factor of k, in order to build $N+1$ layers.

$$\sigma_0 * k^N = R_{max}/\sqrt{2}$$

Therefore,

$$k = \sqrt[N]{R_{max}/(\sqrt{2} * \sigma_0)}$$
$$\sigma_i = \sigma_0 * k^N, \text{ where } i = 0, 1, 2, \cdots, N$$

Scale-Normalized LoG: For the reason that we would like to find local extrema in space, not just in one layer of pyramid, we would like to have response comparable across different scales. However, the response of LoG decreases as the σ grows. Therefore we have to do something to the original LoG before apply to image to get normalized response. Given to normalized LoG kernel,

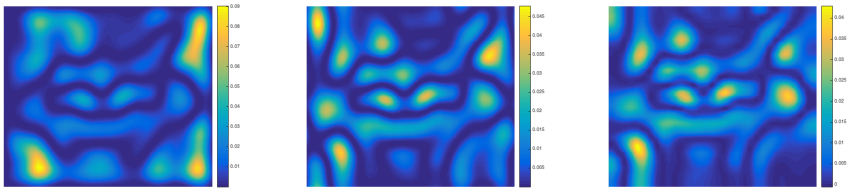
$$\nabla^2_{norm} g = \sigma^2(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2})$$

we multiple σ^2 before it convolve with image.

Find extrema: After convolve with image, we will get a response image range from negative to positive where negative minimum represents for blob of darker area and maximum represents for the brighter area. In fact we not only want to have white blob but also black blob therefore we squared the response image to attain maximum and minimum at the same time.

2.1 Up-sampling Kernels

For the method of up-sampling kernels, a new σ is calculated in each iteration and is based to generate a new Laplacian of Guassian filter. Then the filter is normalized and convolve to get blob response. **Note** that in order to convolve image, *imfilter* is used as well as its '*symmetric*' option to avoid edge problems.Shown in Figure 1, *symmetric* method has the most continuous and reasonable response across edges. As it is mentioned above, the response is then squared and stacked as pyramid. For response image pyramid using Up-sampling Kernels method, please refer to Section 4 for details.



(a) Zero (b) Circular (c) Symmetric

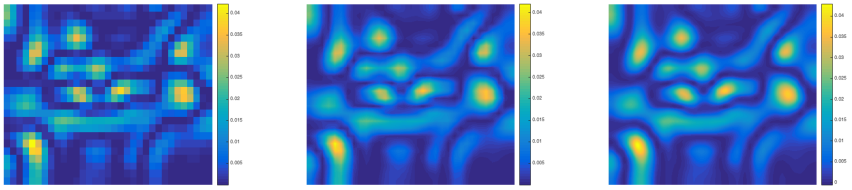
Figure 1: A comparison between different edge choice

2.2 Down-sampling Images

In method of down-sampling, the scale normalized Laplacian of Guassian with initial σ is generated once and applied several times on image of different scale. Since the LoG is actually same σ across different scales normalization does not really matter. During each iteration, grayscale image is subsampled by factor of k to convolve with and then Up sample to original size again.

There are two things need to be especially notice of. Firstly, we might run into **aliasing problem** as we sub-sample image and then recover. Fortunately,

imresize function do the Anti-aliasing by default. Secondly,when up-sample image to recover, the **interpolation quality** matters a lot. Matlab has three implementation of interpolation in *imresize* function,namely *nearestneighbor*, *bilinear* and *bicubic*. Different method varies a lot in quality as shown in Figure 2. Since response quality have huge weight on final detected position of blob, my choice is the most accurate one which is *bicubic* method.



(a) Nearest Neighbor (b) Bilinear (c) Bicubic

Figure 2: A comparison between different interpolation method

2.3 Performance Comparison

Running Time: Theoretically, there should be a huge difference between method of Up-sample Kernel and method of Sub-sample Image where Sub-sample image will be faster.Practically, the running time of my two implementation under different image shown in Table 1 below.

Table 1: Running time for Up-sample and Sub-sample method in 8 images.

Method	Running time in each image								Aver.
Up-sample	2.02	4.32	1.99	1.36	1.30	2.80	6.79	1.98	2.82
Sub-sample	0.08	0.13	0.06	0.08	0.08	0.10	0.09	0.09	0.09

Detect Quality:Although up-sample method performs poorly in running time it has higher position detection accuracy as the response reflects the actual position of a blob. In contrast, position in sub-sample varies for information lose while sub-sampling and up-recovering. When using a higher threshold,it can be clearly seen sub-sample method blob shifting towards to up-left corner shown in Figure 3.This is because *imresize* tends to keep first col or row when sub-sampling and while interpolation is depending on those survived elements thus blob has great chance of up-left shifting instead of down-right shifting. To slightly eliminate this bias I use *ceil*() to slightly expend the image while sub-sampling.

3 Non-maximum Suppression

We use non-maximum suppression to find the local extrema in image space. In order to identify the blob's 2D location as well as its most suitable scale. We should not only find the local maximum for 2D image, but should also be able to compare across neighboring layer.

(a) Up-sample (b) Sub-sample (c) Sub-sample with ceil()

Figure 3: Position accuracy between implementations

In order to do that, my implementation is to first assign the pixel with the max value in its 2D neighborhood and then compare if this value is larger than neighboring layer's corresponding position and also has original intensity that equals to local max value.

3.1 Performance Comparison

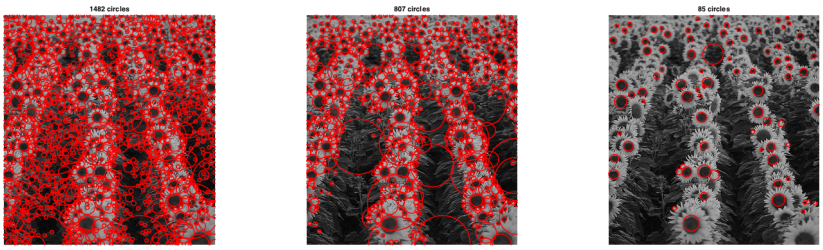
There are three implementation could do this job in Matlab, namely $nlfilter$, $colfilt$ or $ordfilt2$. After first try, it is obvious $nlfilter$ is much slower than the other two due to its interface so we just compare the running time for later two methods, shown in Table 2. As it is shown, $ordfilt2$ is faster.

Table 2: Running time for $colfilt$ and $ordfilt2$ in 8 images.

Method	Running time in each image								Aver.
colfilt	0.79	1.09	0.75	0.96	1.44	0.93	1.07	0.94	1.00
ordfilt2	0.43	0.74	0.40	0.55	0.85	0.58	0.54	0.49	0.57

3.2 Tuning Parameters

To get the best blob detection results, tuning parameters is necessary. The main two parameters I am playing around is the max blob radius R_{max} and threshold to ignore the blob response th. The max blob is easy to decide since it is visually connected with biggest area you would like to detect. The threshold, however, need several try to get a good results.Too low th cause blob detected too dense while too high is not good either. In this project, threshold is represent by percentage of peek response in image stack.



(a) th=0.01 (b) th=0.15 (c) th=0.5

Figure 4: Tuning threshold parameters for image No.4

References

[1] Lecture Notes, Svetlana Lazebnik, *http://slazebni.cs.illinois.edu/spring16/lec09_sift.pdf.*

[2] David A. Forsyth, Jean Ponce, *Computer Vision: A Modern Approach, 2nd Edition.*

[3] Source of images, No.5 - No.8,

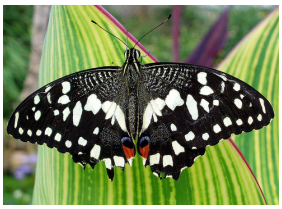
No.5 - *http://www.birds.com.*

No.6 - *http://www.telegraph.co.uk/news.*

No.7 - *http://www.zoo.org.au/melbourne/animals/zebra.*

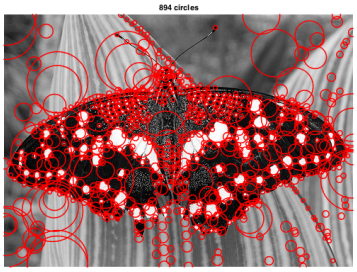
No.8 - *http://cdn.blisstree.com/files/2011/11/shutterstock_80802235.jpg.*

4 Appendix: Collection of Results

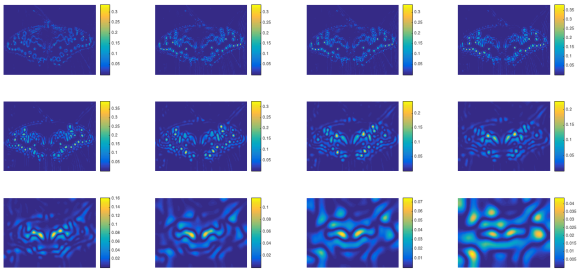


(a) Original Image

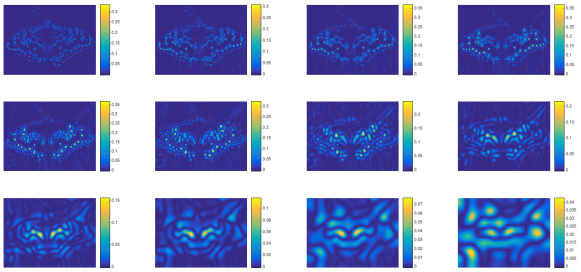
(b) Upsample Kernel



(c) Subsample Image

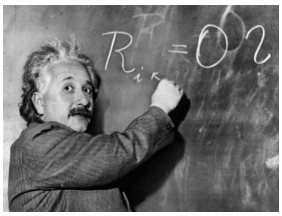


(d) Pyramid of Upsample



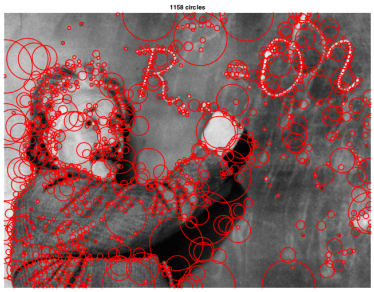
(e) Pyramid of Subsample

Figure 5: Blob Detection of Image No.1

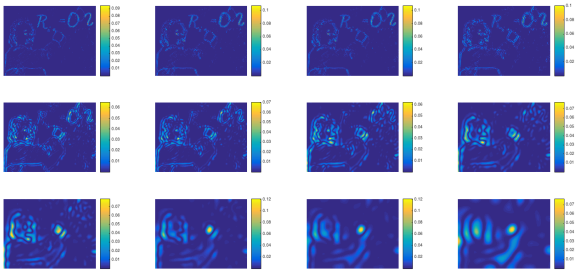


(a) Original Image

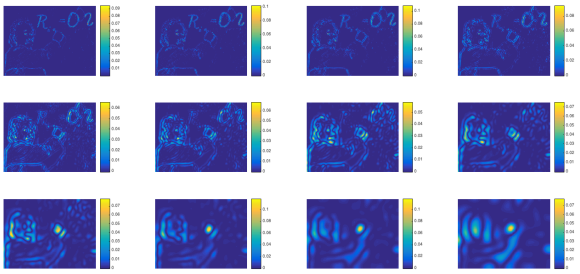
(b) Upsample Kernel



(c) Subsample Image



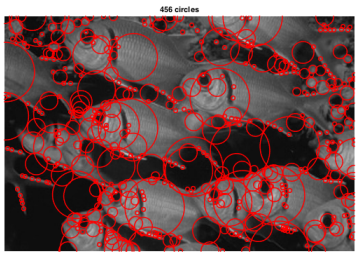
(d) Pyramid of Upsample



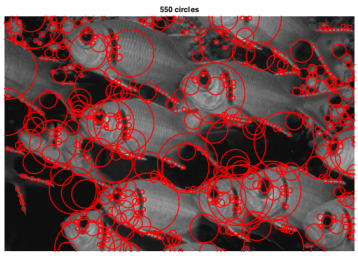
(e) Pyramid of Subsample

Figure 6: Blob Detection of Image No.2

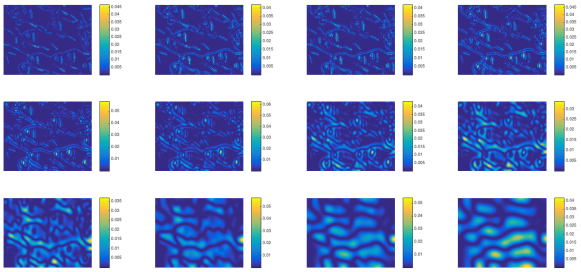
(a) Original Image



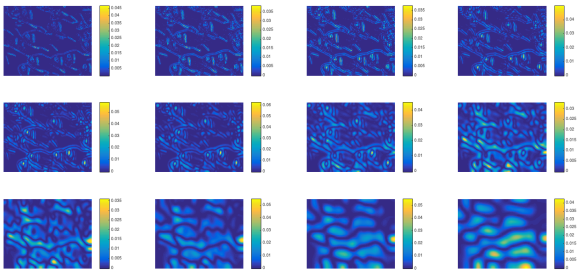
(b) Upsample Kernel



(c) Subsample Image

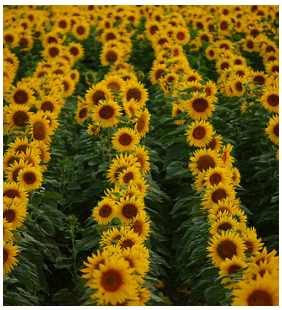


(d) Pyramid of Upsample

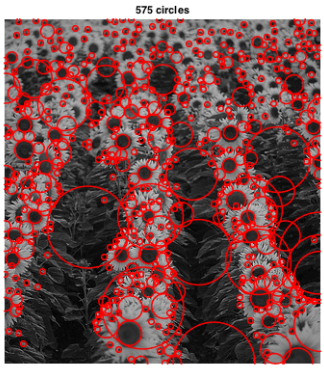


(e) Pyramid of Subsample

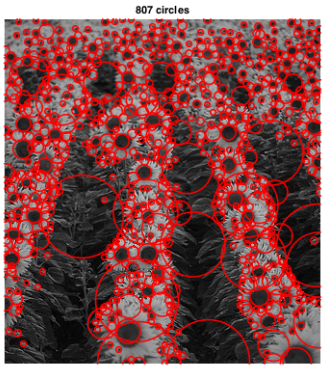
Figure 7: Blob Detection of Image No.3



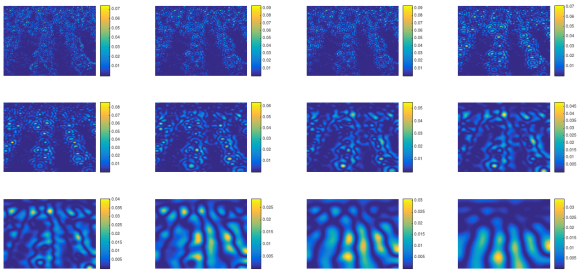
(a) Original Image



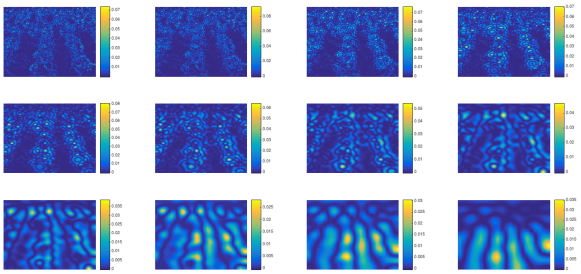
(b) Upsample Kernel



(c) Subsample Image



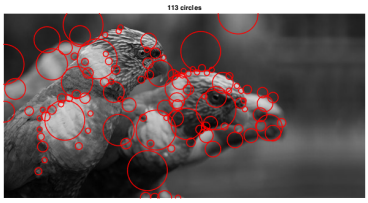
(d) Pyramid of Upsample



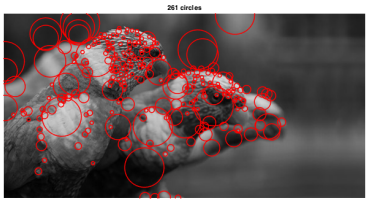
(e) Pyramid of Subsample

Figure 8: Blob Detection of Image No.4

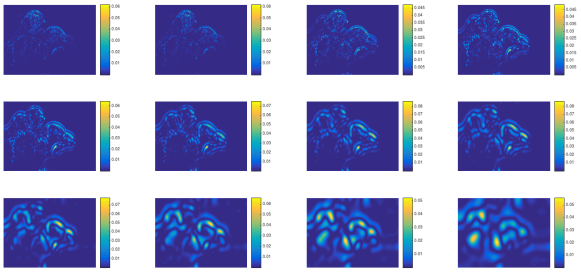
(a) Original Image



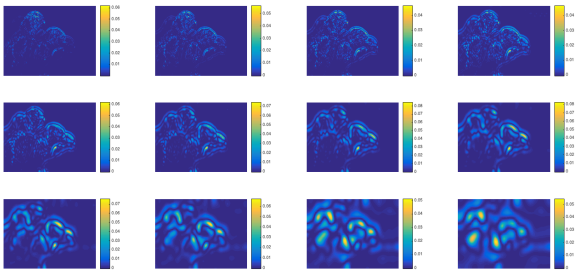
(b) Upsample Kernel



(c) Subsample Image



(d) Pyramid of Upsample



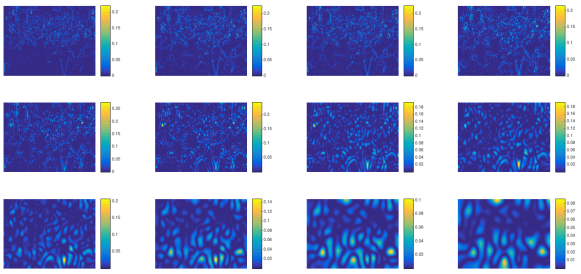
(e) Pyramid of Subsample

Figure 9: Blob Detection of Image No.5

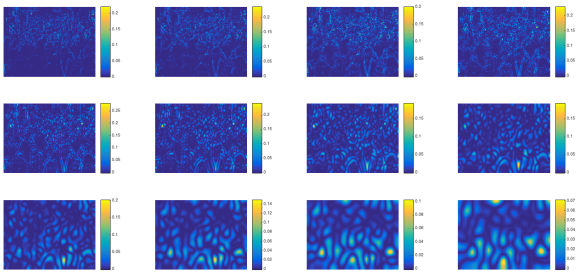
(a) Original Image

(b) Upsample Kernel

(c) Subsample Image



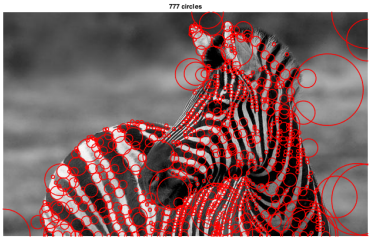
(d) Pyramid of Upsample



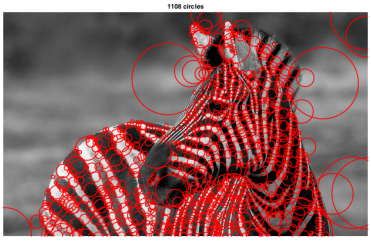
(e) Pyramid of Subsample

Figure 10: Blob Detection of Image No.6

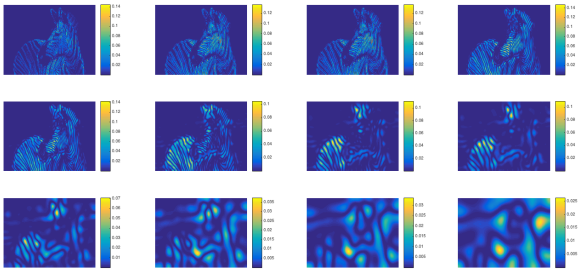
(a) Original Image



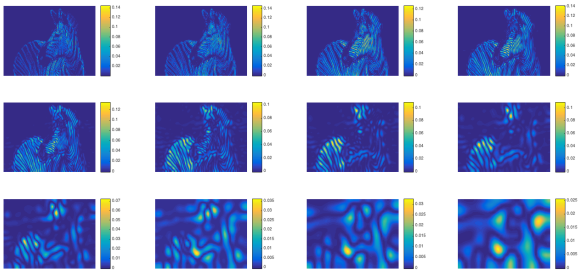
(b) Upsample Kernel



(c) Subsample Image



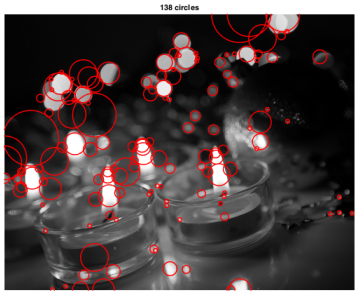
(d) Pyramid of Upsample



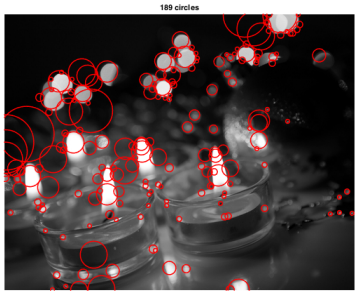
(e) Pyramid of Subsample

Figure 11: Blob Detection of Image No.7

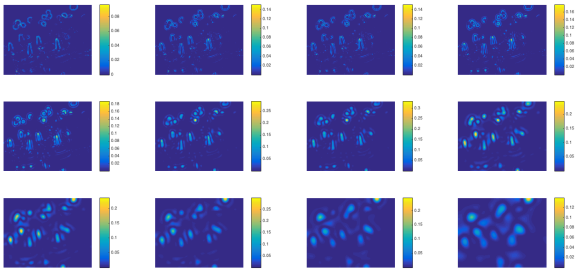
(a) Original Image



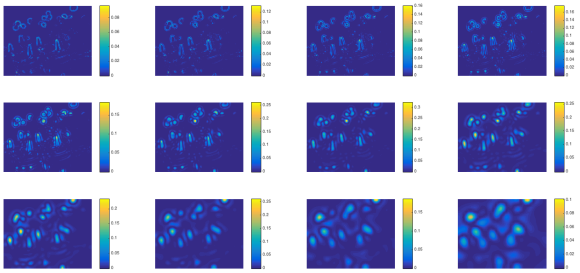
(b) Upsample Kernel



(c) Subsample Image



(d) Pyramid of Upsample



(e) Pyramid of Subsample

Figure 12: Blob Detection of Image No.8