Basic improvement of Selective Search [1]

Comp 417 at McGill University
Name: Zhaoqi Xu
Student ID: 260563752

Abstract

The original paper published in 2013 addressed to the problem of generating possible object locations for use in object recognition as well as using results from selective search to recognize objects with Bag-of-Words model [2]. From a large number of tests, I found out that selective search works well in most cases. However, under specific conditions, like the image with lots of noise, blur images, low contrast, black and white, the performance of selective search are relatively low. The accuracy of object recognition decreases and there are many meaningless boxes among the search results. This paper shows some basic attempts to address this kind of problem. The basic idea of my approach is filtering those images before they are searched. I implemented several methods for image sharpening, de-noising, contrast-stretching, and color image enhancing to filter the image first to get a better result from selective search. It turns out that my approach can actually improve the performance of selective search for those abnormal images. Detail will be discussed later in this paper.

Background

Before discussing my approach to the improvement, I want to restate the selective search. In my opinion, selective search can be implemented in three steps.

Step 1: Generate initial sub-segmentation
The goal is to generate many regions, each of which belongs to at most one object as showed in Figure 1. Using the method described by Felzenszwalb et al. [3 4]

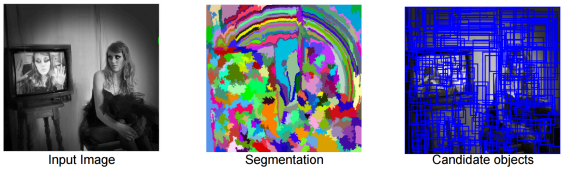


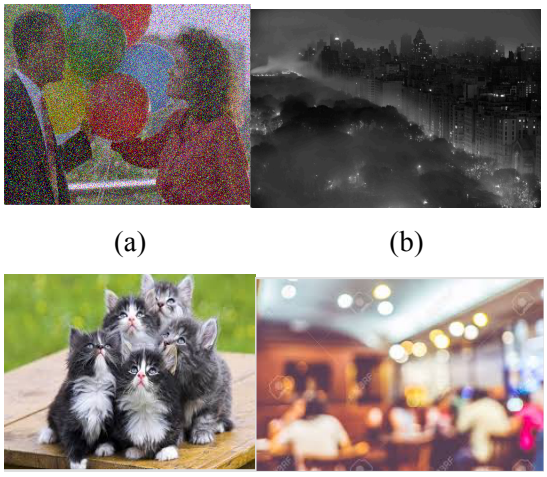
Figure 1: images above show the original input image (left most), the image after initial segmentation (middle) and the initial candidate objects.

Step 2: Recursively combine similar regions into larger ones. Here we use a greedy algorithm:

1.From set of regions, choose two that are most similar.
2. Combine them into a single, larger region.
3. Repeat until only one region remains.

This yields a hierarchy of successively larger regions, just like we want. In the original paper [1], the author used four complementary, fast-to-compute measures to decide which two

boxes should be combined in each step. s_{colour} measures colour similarity. $s_{texture}$ measures texture similarity. s_{size} encourages small regions to merge early and s_{fill} measures how well two regions fit into each other. These four measurements are crucial in selective search because the results largely depend on the performance of these measurements. However, from the results of tests on some images, I found out that some factors (in Figure 2) have a negative effect on the results. They make the results either lack of potential reasonable objects or full of meaningless objects. This is the reason that I implement image filters filtering out the negative factors to obtain better results.



(a) (b)

(c) (d)

Figure 2: Some negative factors affect the selective search performance. (a) image with lots of shot noise. (b) city at foggy night. (c) many objects with similar color and texture and stay really close. (d) blur image.

Step 3: Use the generated regions to produce candidate object locations for use in a practical object recognition framework.

To calculate the similarity of two regions, we should not only take those 4 similarity measurements into consideration, but also the color space. In the original paper, the author mentioned 8 different color spaces: (1) *RGB*, (2) the intensity (grey-scale image) *I*, (3) *Lab*, (4) the *rg* channels of normalized *RGB* plus intensity denoted as *rgI*, (5) *HSV*, (6) normalized *RGB* denoted as *rgb*, (7) *C* [5] which is an opponent color space where intensity is divided out, and finally (8) the Hue channel *H* from *HSV*. However, they always use a single color space throughout the algorithm.

By using 8 color spaces and 4 different similarity measures, they diversify selective search.

Image Filters

In this section we detail our filters for improving selective search. To deal with different abnormal images, I implement some different filters. I will discuss them one by one.

Noise Removal:

When the selective search is processing images with noise, it returns many meaningless partial patterns which will make the object recognition take more time than it should. So we want to remove image noise before it is fed into selective search. To remove or reduce noise, we have to first think about what the noise is in digital images. Let's imagine that an area with uniform blue in an image might have a very small white part. If this is a single pixel, it is

likely to be spurious and noise; if it covers a few pixels in an absolutely regular shape, it may be a defect in a group of pixels in the image-taking sensor; if it is irregular, it may be more likely to be a true feature of the image. So we want to use this feature to detect noise spots and change the color of those pixels to the color of surrounding pixels. Two methods are helpful to achieve this.

Moving Window Operations:

The form that low-pass filters usually take is as some sort of moving window operator. The operator usually affects one pixel of the image at a time, changing its value by some function of a "local" region of pixels ("covered" by the window). The operator "moves" over the image to affect all the pixels in the image. Both methods below are based on moving window operation.

Average filter:

The basic idea behind average filter is replacing each pixel by the average of the pixels in a square window surrounding this pixel.

Average filter algorithm:

For each pixel in the image

1.extract numbers of a window from (i, j) of Im1 and multiply with the window

2.calculate the sum of the pixels in the window

3.calculate the average and assign the value to the center of the window

---

The output *y(i,j)* of WA filters can be calculated as

$$y(i,j) = \sum_{P=-P}^{P} \sum_{Q=-Q}^{Q} 1 \cdot x(i+p, j+q) \Big/ \sum_{P=-P}^{P} \sum_{Q=-Q}^{Q} 1$$

where-$P \leq p \leq P$,-$Q \leq q \leq Q$(i.e., $(2P+1) \times (2Q+1)$ window size). $x(i+p, j+q)$ is a signal within the window. The reason we normalize is to keep the image pixel values between 0 and 255.

There is also an improved averaging filter called weighted averaging filter. Instead of averaging all the pixel values in the window, give the closer-by pixels higher weighting, and far-away pixels lower weighting. The weight of weighted average (WA) filters *W(p,q)* is assigned a real value. The output *z(i,j)* of WA filters can be calculated as

$$z(i,j) = \sum_{P=-P}^{P} \sum_{Q=-Q}^{Q} W(p,q) \cdot x(i+p, j+q) \Big/ \sum_{P=-P}^{P} \sum_{Q=-Q}^{Q} W(p,q)$$

If all weights *W(p,q)* are equal values, WA filters are equivalent to simple average filters.

Using average filters, we re-calculate every pixel in this image like Figure 3, if there is a big difference between a pixel and its surroundings, it will be updated by the average value of its pixels. Usually, larger window leads to more effective noise removal, but also blur the details. So there is always a trade-off between noise removal and detail preserving with average filter.

100	100	100	100	100
100	200	205	203	100
100	195	200	200	100
100	200	205	195	100
100	100	100	100	100

original

100	100	100	100	100
100	144	167	145	100
100	167	200	168	100
100	144	166	144	100
100	100	100	100	100

average filter

100	100	100	100	100
100	156	176	158	100
100	174	201	175	100
100	156	175	156	100
100	100	100	100	100

weighted average filter

Figure 3: this figure shows the original pixel value and how it is changed after filtering through average filter and WA filter.

Median filter [6]***:***

The problem with average filter is the blur edges and details in the filtered image. And it is also not effective for impulse noise (Salt-and-pepper). So I move on to median filter. Median filter is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing noise while preserving edges. The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighboring pixels. The window shape does not need to be a square and special shapes can preserve line structures. Figure 4 shows the pixel value after median filtering. Compared to average filter and WA, median filter has a better performance on speckle noise and salt and pepper noise. But it will cost more time than average filter and WA, because you need to sort every time to get the median.

100	100	100	100	100
100	100	200	100	100
100	200	200	200	100
100	100	195	100	100
100	100	100	100	100

Figure 4: this figure shows how pixels change from original value in Figure 3 filtered by median filter.

Sharpening:

Sharpening is to enhance line structures or other details in an image. To improve the accuracy of $s_{texture}$ measurements of blur image, we need to take more details into consideration, so we want to obtain more details and line structures by sharpening the image. This is why I tried to use image sharpening to improve selective search.

Enhanced image is obtained by using original image + scaled version of the line structures and edges in the image. And line structures and edges can be obtained by applying a difference operator on the image. In this paper, I use spatial operation which takes difference between current and averaging (weighted averaging) of nearby pixels. Two examples are showed in Figure 6.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 6: Two discrete approximations to the Laplacian filter. Note that all coefficients sum to 0.

Now let's discuss more detail about sharpening. The unsharp filter is a simple sharpening operator which derives its name from the fact that it enhances edges via a procedure which subtracts an unsharp, or smoothed, version of an image from the original image. However, there is a more common way of implementing the unsharp mask is by using the negative Laplacian operator to extract the highpass information directly. See Figure 7.

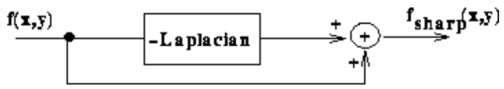


Figure 7: Spatial sharpening.

This is the procedure that I implemented image sharpening. After convolving an original image with a kernel such as one of these in Figure 6, it need only be scaled and then added to the original.

However, image sharpening always brings image with more noise, so it seems that it is a conflict between smoothing the noise without blurring the details and enhancing edges without amplifying noise. I believe it is still an active research area.

Brightness

I expected that increasing the brightness would give a better performance on selective search because from the experiment results I observed that usually the performance on dark image is worse than the brightness. Brightness can be simply increased by simple addition to the image matrix. For example, in Figure 8, we consider a black image. Since the whole matrix is filled with zero, and the image is very much darker. Now what we will do, is that we will add 50 to each of the matrix value of the image 1 and see what the image has become, then we get image 2. We can easily observe that image 2 is brighter than image 1.

image 1 image 2

Figure 8

Contrast

I also observed from experiment results that selective search sometimes can only tell part of the objects when many similar objects with the same color as well as the same texture. It has trouble on giving back the whole figure of one of those objects. For example, the selective

search cannot tell the left most cat in Figure 2 (c). Selective search uses s_{colour} , $s_{texture}$, s_{size} , s_{fill} to decide the similarity. However, in Figure 2 (c), all these 4 measurements do not work well since the cats have similar color, texture, size and since they stay very close, no gap between them so that they can fill well with each other. Since contrast is the difference in luminance or color that makes an object (or its representation in an image) distinguishable, I think adjusting contrast might help to distinguish the objects.

To adjust contrast, I simply use Matlab built-in function

$$\boldsymbol{J = imadjust(I, [low_{in}\ high_{in}], [low_{out}\ high_{out}])}$$

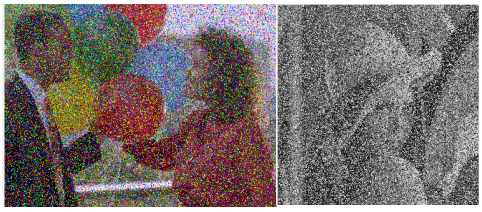
The first vector passed to *imadjust()* specifies the low and high intensity values that you want to map. The second vector specifies the scale over which you want to map them.

Testing

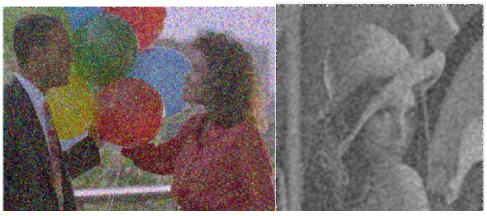
Testing of this paper is done in two parts. For each filter, I first test how the image filters work, then compare the results of selective search between original images and filtered images.

Noise removal:

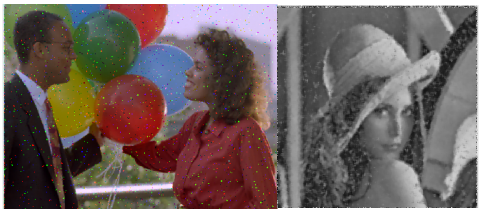
I first tested the average filter and median filter for noise removal. It turns out that the median filter works better than the average filter on impulse noise(salt and pepper). Also, median filter preserves much more detail and edges than average filter. See Figure 9. So in the next step, I only test selective search on image filtered by median filter. The result is definitely better than average filter.



original



average filter



median filter

Figure 9

As I mentioned earlier in this paper, selective search for images with noise will return tons of meaningless partial objects which are not what we want. So lets' test on filtered images.

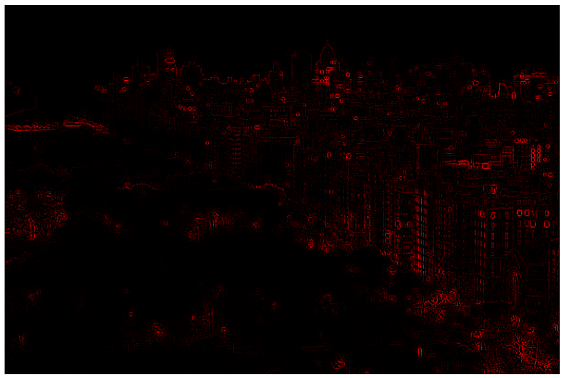
To show the improvement of selective search, I take the image with a couple and balloons as an example. The selective search code returned 25 * 39 = 975 boxes as potential objects for the original image while it only returned 25 * 26 = 650 boxes for the de-noised image. (975 – 650) / 975 = 33.3% of the boxes are eliminated due to removal of noise. And most boxes among the 33.3% are meaningless. Therefore, I can conclude that the noise removal filter not only speeds up the running time of selective search, but also the accuracy.

Removal of noise obviously makes an improvement on selective search for images with impulse noise.

Sharpening:

The sharpening Matlab code can actually make the image looks sharper than before. To see how it works clearly, I output the intermediate edges and line structures. See Figure 10.

Original image



Filtered image

Sharpened image

Figure 10

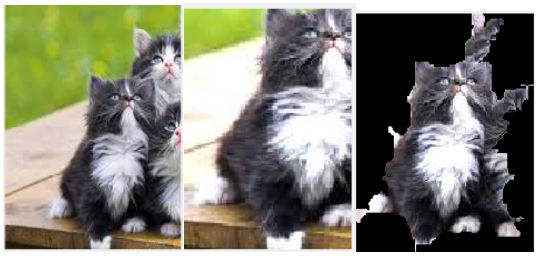
As you can see from Figure 10, compared to the original image, sharpened image does show sharper details in the fog. Then let's compare them in selective search.

The selective search can find 115 possible locations from original image while it can find 499 potential locations from sharpened image. It is apparent that by sharpening the image, the selective search can obtain much more details about the image. After carefully comparing those boxes, I found out that boxes obtained from sharpened image even cover details on the building, like a small window in the dark as well as the far building in the fog. So we can conclude that image sharpening can definitely improve performance of selective search.

Contrast:

I expected that increasing image contrast can bring a better performance in the same way as image sharpening. Because both modify images with more details. This time I want to try to address the problem mentioned in the background section. In Figure 2(c), the leftmost cat cannot be completely figured out by selective search. So I want to see whether it can be recognized after increasing image contrast.

Actually, it works as what I expected at the very beginning. See Figure 11. (a) is the best pattern I can get from selective search using the original image. Though it includes the whole figure of the leftmost cat, it also includes two faces of other cats which means that it can not precisely figure out the leftmost one. While (b) and (c) are the patterns obtained from the contrast enhanced image.



(a) (b) (c)

Figure 11

Figure 11 (b) and (c) obviously focus on the leftmost cat. In (c), selective search even takes other patterns away except for the cat which indicates that it completely recognize that lovely cat. Therefore, enhancing image contrast can actually bring the performance of selective search to a higher level.

Brightness:

To compare the effect of brightness and contrast on selective search, I tested brightness on the same cats' image again. In my test, I increased the value of all pixels by 20, and it turned out that increasing brightness can improve performance but not as much as increasing contrast. See Figure 12.

Figure 12

Figure 12 is the best pattern I can get for the leftmost cat by just adjusting brightness. I tried to increase brightness in different value, but it seems that it cannot work as well as contrast enhancing.

Conclusion

This paper gives a summary of selective search [1], then introduces four methods based on image processing to improve the performance of selective search for some different abnormal images. After testing, I can conclude that all these four image processing methods can improve performance under different situations in different levels. Also, there are some more sophisticated algorithm for noise removal, image sharpening, enhancing contrast as well as brightness. I believe that the more sophisticated the algorithm is, the better performance one can get from it. Besides the negative factors that I mentioned in this paper, there are also some other factors that affect the selective search performance in a bad way like blur image. Hopefully, I can address this problem in the future.

References

[1] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective Search for Object Recognition. In IJCV, 2013.

[2] Harris, Zellig. Distributional Structure. 1954.

[3] P. F. Felzenswalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. In IJCV, 2004.

[4] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. TPAMI, 24:603–619, 2002. 1, 3

[5] J. M. Geusebroek, R. van den Boomgaard,

A. W. M. Smeul- ders, and H. Geerts. Color invariance. TPAMI, 23:1338–1350, 2001. 4

[6] Jayaraman; et al. Digital Image Processing. Tata McGraw Hill Education. p. 272.

[7] R. Haralick and L. Shapiro *Computer and Robot Vision*, Addison-Wesley Publishing Company, 1992.

[8] B. Horn *Robot Vision*, MIT Press, 1986, Chap. 6.

[9] A. Jain *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989, Chap. 7.

[10] R. Schalkoff *Digital Image Processing and Computer Vision*, John Wiley & Sons, 1989, Chap. 4.