Yue Pan
yuepan@student.ethz.ch

Assignment 2

Image Interpretation
October 5, 2019

Exercise 1

Topic: Gaussian Smoothing

For this exercise, please refer to *ii_gaussian.m*.

There are two key parameters with regard to gaussian filter, namely the gaussian kernel size s and the standard deviation σ. Filtering result under different s and σ are shown in Fig.1-3. The goal of gaussian smoothing on target image is to filter the noise but still keep the general information of the image. My conclusion is: the larger the kernel size s and the standard deviation σ are, the greater the degree of smoothing and blurring is. The default $\sigma = 0.5$ in matlab's *imgaussfilt* function, which is too small for this task. After comparison experiments, I used $\sigma = 2$ instead. In my opinion, the parameter settings for gaussian filter should be conducted according to actual image interpretation problem.

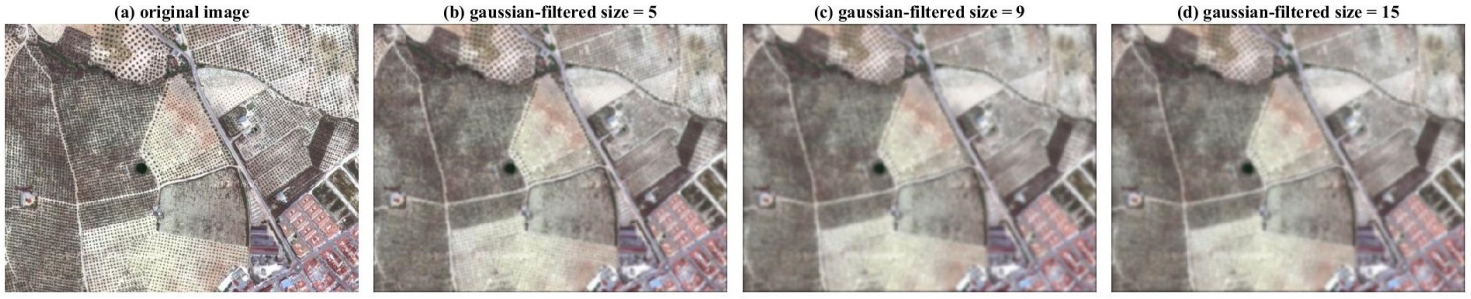

Figure 1: Guassian filter result for different gaussian kernel size while $\sigma = 2$

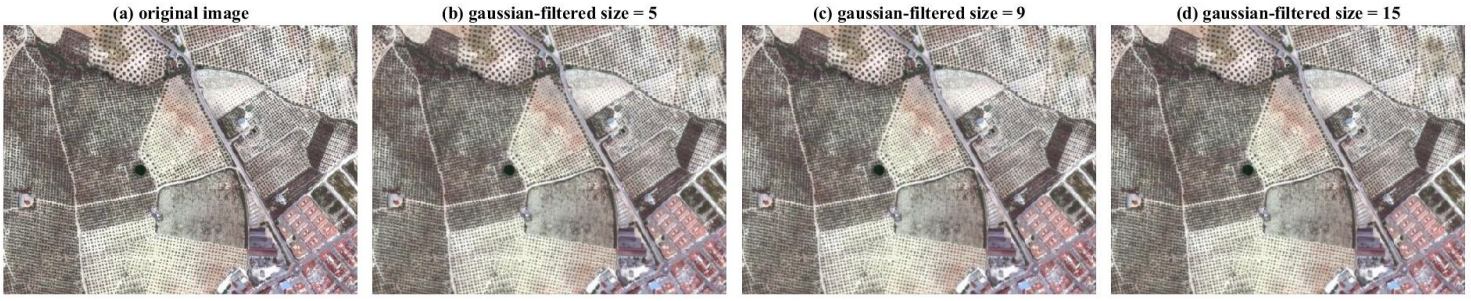

Figure 2: Guassian filter result for different gaussian kernel size while $\sigma = 0.5$

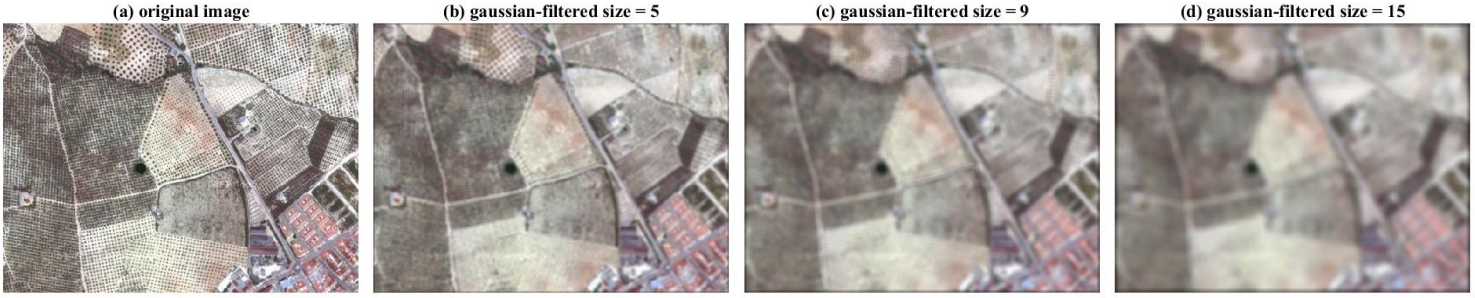

Figure 3: Guassian filter result for different gaussian kernel size while $\sigma = 5$

Exercise 2

Topic: Edge Detection

For this exercise, please refer to *ii_deriv.m, ii_deriv_1d.m, ii_sobel.m.*

In order to accomplish derivation calculation for different order and different orientation in one function, a first-order, one-dimensional derivation calculation unit function (*ii_deriv_1d.m*) is created. Then *ii_deriv.m* uses while loop to conduct first-order derivation m and n times for x and y orientation to get the result of mth-order x-derivative and nth-order y-derivative. The final result is shown in Fig.4. It's clear that by applying x (y) -derivation filter, the vertical (horizontal) boundaries are highlighted. Besides, second order derivative can achieve more meticulous profile than first order derivative.

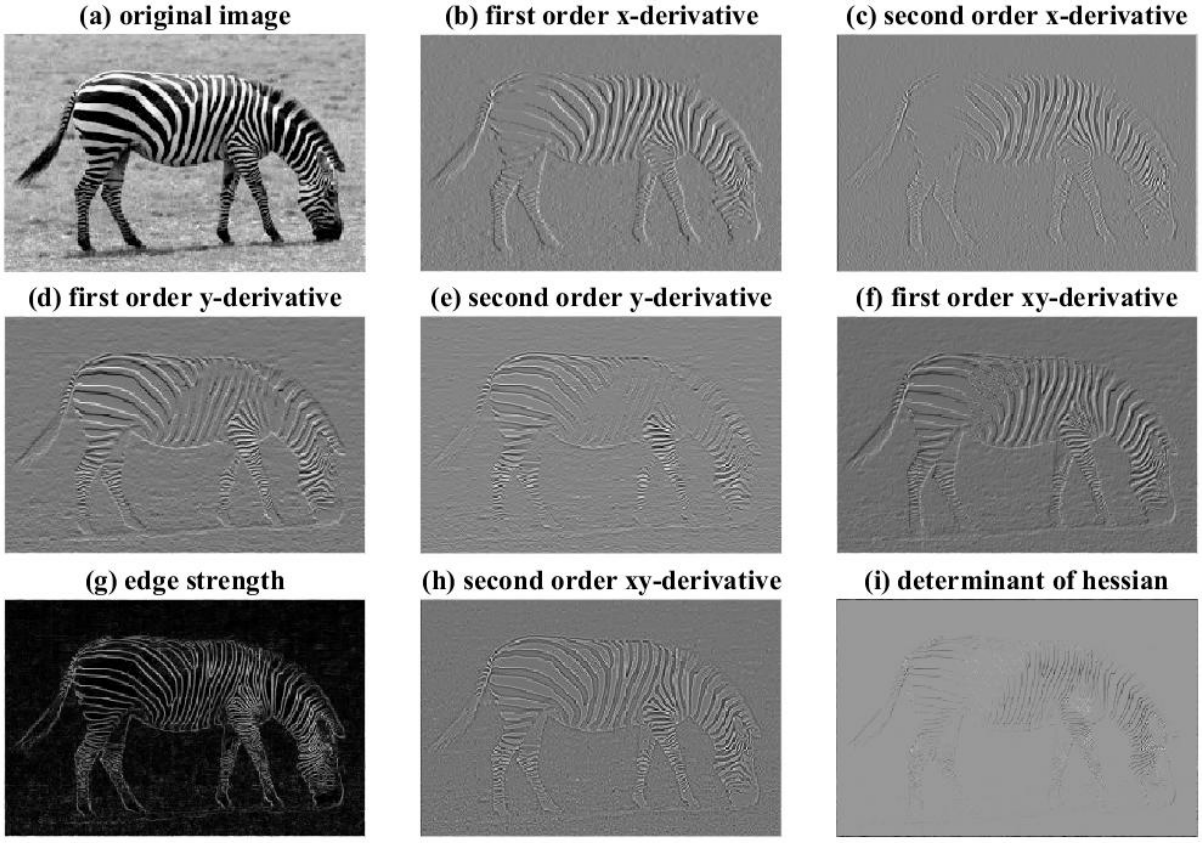

Figure 4: Image derivation and edge detection result for different order and orientation

According to Fig.5, it seems that it's better to apply the gaussian filter on the derivative function to extract edges. Since the original derivative image has a lot of noise, the gaussian filter smooths it and achieves finer edges. This is the exact effect achieved by the well-konwn Sobel operator.

Exercise 3

Topic: Variance in neighborhood

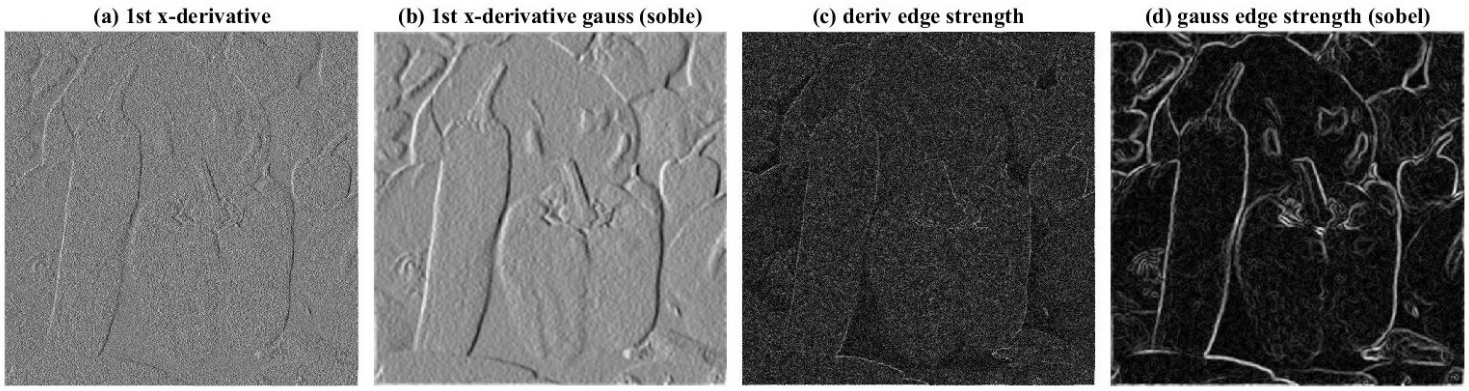

Figure 5: Image derivation result with and without gaussian filtering

For this exercise, please refer to *ii_variance.m*.

To avoid the size change of the image and the boundary artifacts, I firstly used *padarray* function with parameter *'symmetric'* and *'both'* to expand the image by adding s rows or columns to each boundary. Then for each pixel of the original image, the variance is calculated from its $(2s+1) \times (2s+1)$ neighborhood using *colfilt* function. At last, only the original size of the image is kept and output. The result highlights the urban part of the satellite image as shown in Fig.6 and Fig.7.

Here, I firstly compute the variance on the grayscale image converted from the original rgb image. Besides, I also compute the variance on each of the color channel and then convert the variance rgb image to gray image. Their results are respectively shown in Fig.6(b) and (c). According to Equ.1-3, it's clear that $D_{\text{gray2rgbfirst}}$ and $D_{\text{gray2rgblast}}$ should have similar trend but also some difference. For $D_{\text{gray2rgbfirst}}$, the green channel contributes more to the gray value compared with the blue channel. Since the image is normalized to fill its range of grayscale, it seems that there's only a bit difference between Fig.6(b) and (c). As for me, it may be better to calculate the variance on each channel and then convert the rgb variance image to gray image because it maintained the composition ratio of r, g, b channel.

By changing the size of the neighborhood, the result of variance image would also change. The larger the neighborhood size is, the coarser the boundary of the urban area is, as shown in Fig.8. Nevertheless, urban area and roads are correctly detected for all the tested neighborhood size.

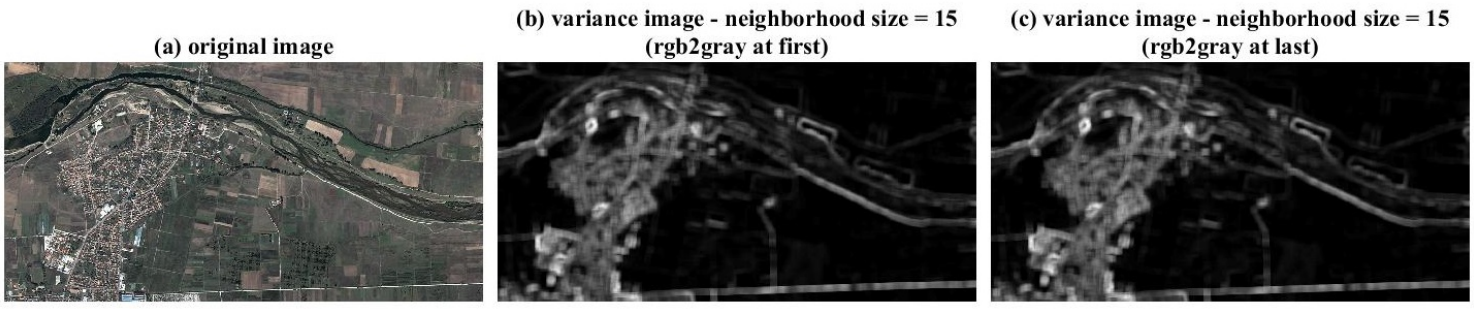

Figure 6: Satellite image neighborhood variance calculation result: (a) original image, (b) the image is transformed to gray image and then filtered, (c) the image is firstly filtered on each color channel and then transformed to gray image.

$$D_{\text{gray2rgbfirst}} = D\left(rR + gG + bB\right) = r^2 D\left(R\right) + g^2 D\left(G\right) + b^2 D\left(B\right) \tag{1}$$

$$D_{\text{gray2rgblast}} = rD\left(R\right) + gD\left(G\right) + bD\left(B\right) \tag{2}$$

$$r = 0.299,\ g = 0.587,\ b = 0.114 \tag{3}$$

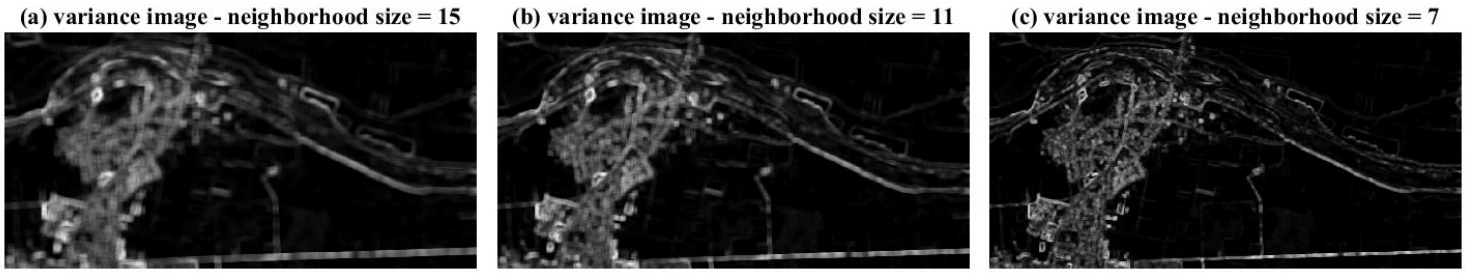

Figure 7: Satellite image neighborhood variance calculation result for different neighborhood size

Exercise 4

Topic: Image (patch) histogram

For this exercise, please refer to *ii_hist.m*.

As shown in Fig.8, the histogram with 16 gray scale levels for two iamge patches are generated.

Suppose the required bin number is s, I firstly use *linspace(0,1,s+1)* function to get a $s+1$ equally divided vector. Then the first and last element of the vector are deleted. The rest $s-1$ elements act as the dividing threshold of the processed image. *imquantize* function is adopted to get the image with s levels. At last, the histogram is generated by counting the pixel number of each gray level.

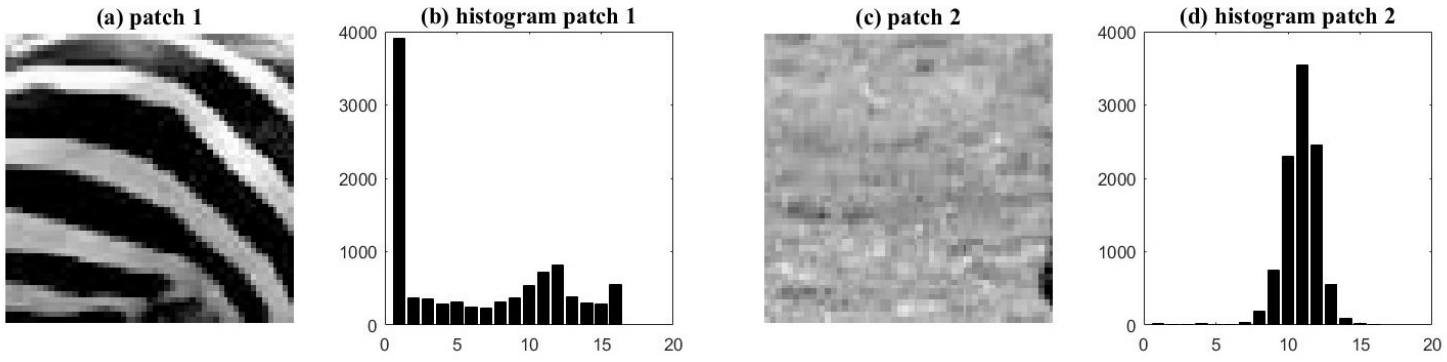

Figure 8: Histogram with 16 levels for two patterns in a image