Computer Vision 1: Filters

Bogdan Teleaga, 11413565
Mircea Mironenco, 11164182

February 27, 2017

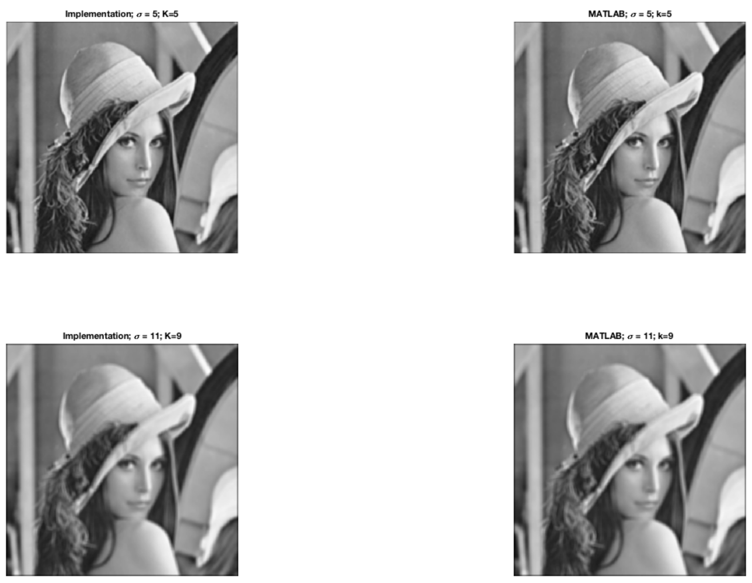
1 2D Gaussian Filter

We construct a manual implementation of a 2D Gaussian filter, and compare it to the standard MATLAB implementation.

$$G(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-(x^2+y^2)/(2\sigma^2)}$$

1.1 Visualize output

Figure 1: Comparison of Gaussian filter implementations.


Implementation; σ = 5; K=5
MATLAB; σ = 5; k=5
Implementation; σ = 11; K=9
MATLAB; σ = 11; k=9

2 More on filters

2.1 Gaussian versus Box

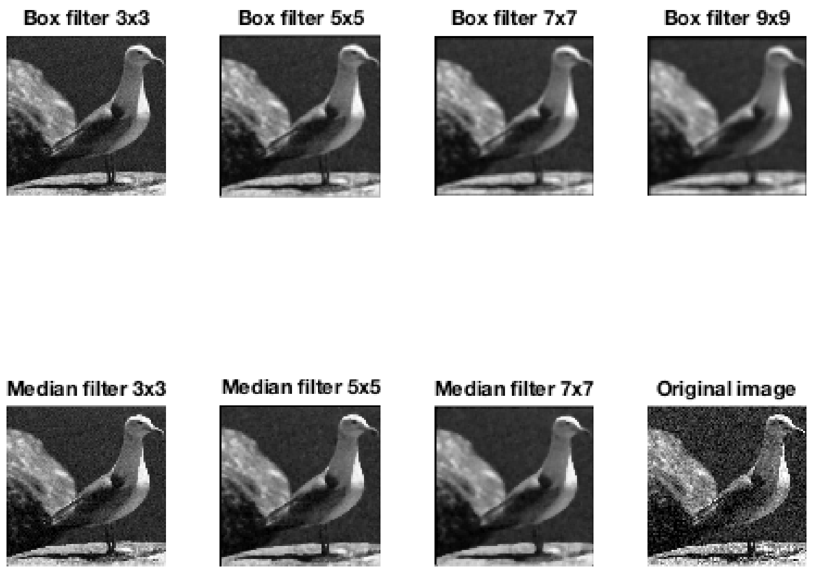
We implement both the both and median filter as a measure to denoise a given image.

(a) The results of both denoising filters with different hyper-parameters are presented in Figure 2, alongside the original image.

(b) As the kernel size is increased, more pixels are now a part of the convolution process. The filter will stop using only local information and take the global structure of the image into account. This will lead to more than just the surrounding center pixels being blurred, and small local noise details being stamped out. The entire picture will be smoothed accordingly.

(c) Because median filtering uses the median instead of the mean, compared to the box filter it is less prone to outliers. Outliers, such as noise of any kind, can be problematic when attempting to maintain an image's edges, which preserve the structural elements in the image.

Figure 2: Comparison of Gaussian filter implementations.



2.2 Histogram Matching

Histogram matching transforms the first image so that it has a histogram that is the same as the histogram of the second image. The implementation can be found in **myHistMatching.m**. We can see in Figure 4 how the final histogram roughly follows the one of the reference image. As a result, the image itself gets brighter.

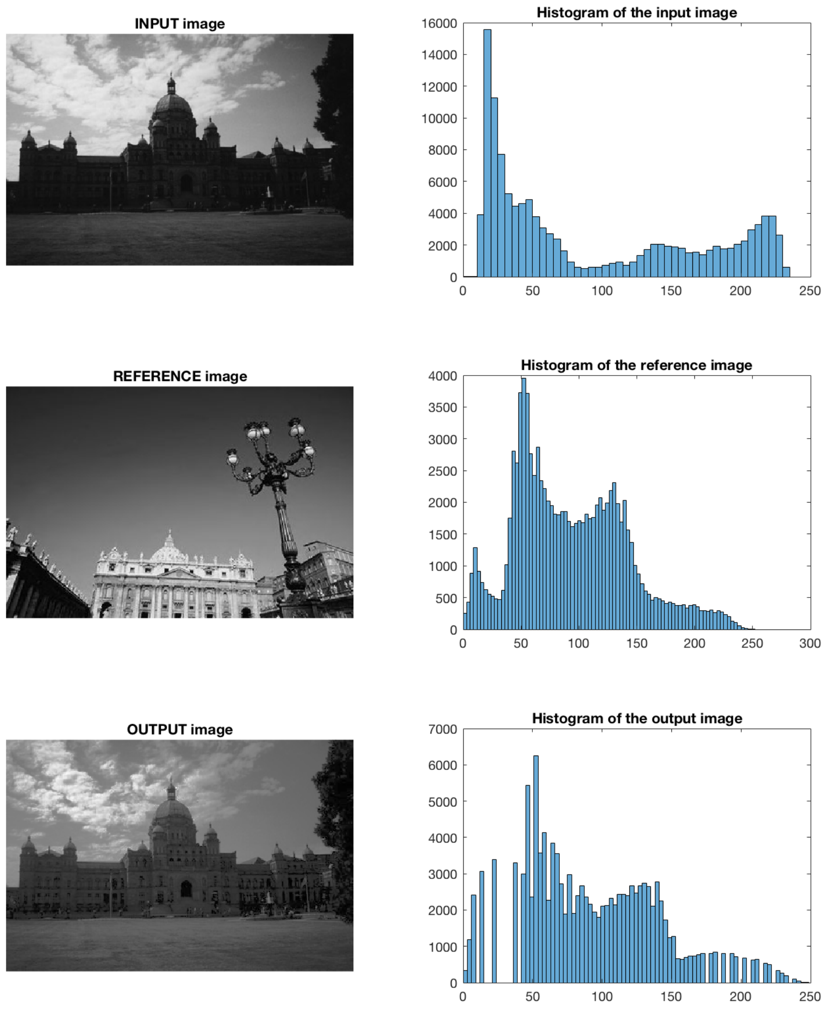


Figure 3: Histogram Matching

2.3 Gradient Magnitude and Direction

We have implemented **compute_gradient.m** which computes the magnitude and the direction of the gradient vector corresponding to a given image using the Sobel kernel to approximate the 2D derivative of the image. We found different definitions for the Sobel kernel online where the signs are sometimes flipped. We experimented with them, but in the end we stuck with the predefined one inside Matlab. When plotting the gradient direction the result is quite noisy, similar to `imgradient`'s output. To get a cleaner picture we multiplied it with a mask based that only keeps directions where the magnitude is greater than a threshold value.

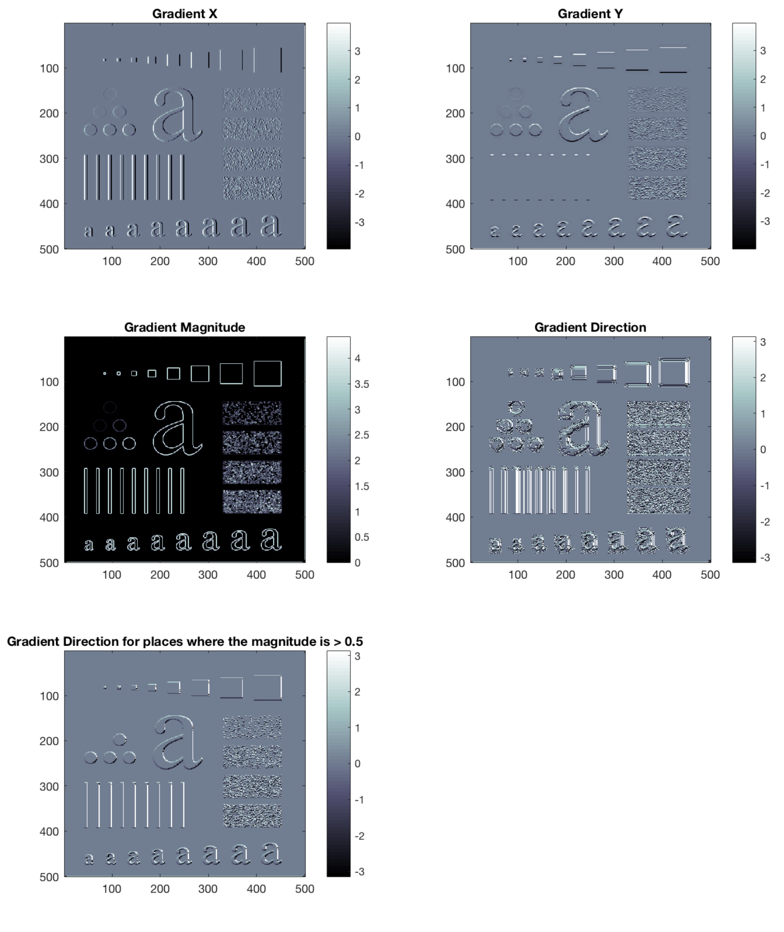


Figure 4: Image gradient

2.4 Unsharp masking

We have have implemented **unsharp.m** which highlights the image for finer details.

(a) Using *image4.jpg* we can visualize the results in Figure 6.

(b) K is a weighting factor that affects the level of contrast added to the image. This in turn affects how much the image is sharpened. With k = 1 we can observe an initial emphasis on the edges and borders. With k > 1 this effect is amplified and differences are accentuated further.

(c) As mentioned previously, kernel size affects the amount of pixels that are part of any convolution process. The kernel radius is increased, and more are used in the transformation. In this case a smaller radius would focus on sharpening smaller detail structures, and as expected when increasing the size global structures are affected (e.g. contrast around the entire moon, etc.).

(d) Image details and resolution can be improved using unsharp masking. It can potentially be used to discover less obvious patterns in the image; making it appear to have a higher quality. It can be used as pre-processing step for further digital image processing techniques, or for scanning the image.

2.5 Laplacian of Gaussian

We have implemented three variants of computing the Laplacian of Gaussian:

- Smoothing the image with a Gaussian operator, then taking the Laplacian of the smoothed image.
- Convolving the image directly with LoG operator.
- Taking the difference between two Gaussians (DoG) computed at different scales σ 1 and σ 2.

(a) The image results of the three methods can be seen in Figure 5.

(b) The first method first applies a Gaussian operator/filter to reduce the noise, and then takes the Laplacian of the resulting image.

Second method is a hybrid filter, which precomputes the LOG kernel, using the fact that LOG operators are second order derivatives operators, and the convolution operation is associative. Since it precomputes the filter, it is the least costly method.

The third method computes a difference of Gaussians at different blur levels by controlling the σ parameter. If the Gaussian operator is highly optimized or implemented in hardware it is the desired approach.

(c) Since the Laplacian filter essentially approximates second order derivatives it inherits their properties such as being very sensitive to noise. Applying a Gaussian filter as a preprocessing step smoothens the image, removing a significant amount of high frequency noise making it easier to detect changes in intensity.

(d) According to [1], the best ratio to approximate the LoG is $\frac{\sigma_2}{\sigma_1} = 1.6$.

(e) Since the operator mainly detects changes in intensity and as can be seen in Figure 5 the LoG is a great edge detector. In some cases it is also used as a blob detector.

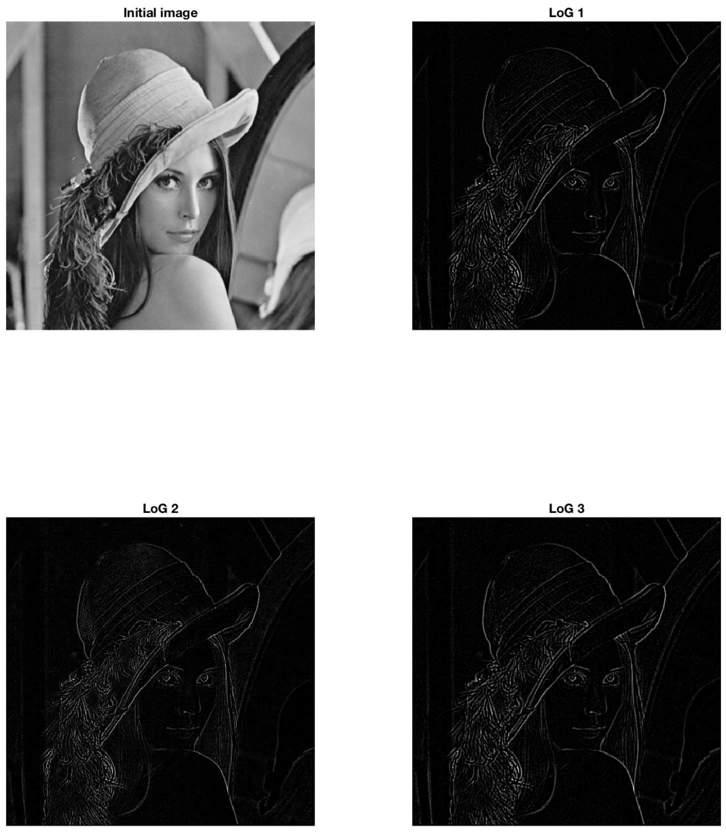


Figure 5: Different Laplacian of Gaussian methods

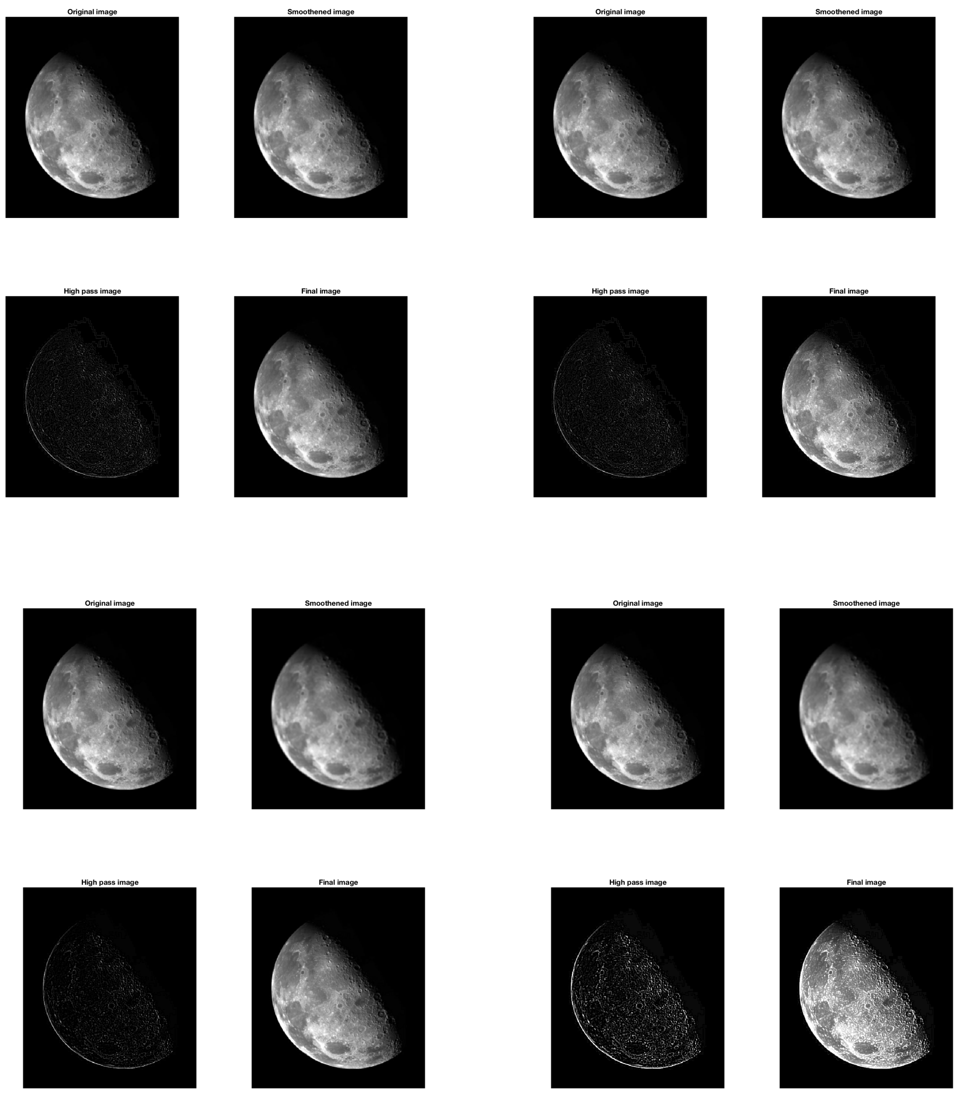


Figure 6: Left to right, top to bottom: k = 1 & $\sigma = 0.5$, k = 10 & $\sigma = 10$, k = 1 & $\sigma = 10$, k = 10 & $\sigma = 10$

References

[1] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London B: Biological Sciences*, 207(1167):187–217, 1980.