Computer Vision 1
Neighborhood Processing & Filters

Gongze Cao, Kai Liang, Xiaoxiao Wen and Zhenyu Gao
Faculty of Science, University of Amsterdam

1 Introduction

In this assignment, we cover some important topics of neighborhood processing for image processing. We begin with discussing the mathematical concepts of linear filters for neighborhood processing in section 2. Then, we implement and discuss some commonly used linear filters in neighborhood processing in section 3 (e.g. Gaussian and Gabor filters). At the end, we apply theories and perform edge detection, image denoising and image segmentation in section 4.

2 Neighborhood Processing

2.1 Question-1

1. The difference between the correlation and convolution operators is that their linear combination ways between $\mathbf{I}$ and $\mathbf{h}$ are exactly the opposite. For correlation, we just need to slide $\mathbf{h}$ over $\mathbf{I}$ and perform point-wise multiplication for each pixel, but for convolution, we first need to flip $\mathbf{h}$ with respect to its center point and then perform the steps of correlation.

2. Correlation and convolution operators are equivalent when the form of the mask $\mathbf{h}$ is symmetric. The reason is that when $\mathbf{h}$ is symmetric, $\mathbf{h}(k,l) = \mathbf{h}(-k,-l)$, so the following equation holds:

$$\sum_{k,l} \mathbf{I}(i-k, j-l)\mathbf{h}(k,l) = \sum_{k,l} \mathbf{I}(i-k, j-l)\mathbf{h}(-k,-l) = \sum_{k,l} \mathbf{I}(i+k, j+l)\mathbf{h}(k,l)$$

3 Low-level filters

3.1 Question-2

Due to the separability of 2D Gaussian kernels, a 2D Gaussian kernel can be defined as follows:

$$G_\sigma(x,y) = G_\sigma(x) * G_\sigma(y)$$

where $G_\sigma(x)$ and $G_\sigma(y)$ are the 1D Gaussian filters in the x- and y-direction respectively. When convolved with a 2D image $\mathbf{I}$, due to the associativity of convolutions, the output can be defined as:

$$\begin{aligned}\mathbf{I_{out}} &= \mathbf{I} * G_\sigma(x,y) \\ &= \mathbf{I} * (G_\sigma(x) * G_\sigma(y)) \\ &= (\mathbf{I} * G_\sigma(x)) * G_\sigma(y)\end{aligned}$$

thus, there is no difference between convolving an image with a 2D Gaussian filter and a 1D Gaussian filter in the x- and y-direction. Given the size of the image is $m \times n$ and the size of the 2D Gaussian filter is $h \times w$, when doing the convolution with the 2D Gaussian filter, the number of multiply-accumulate operations is approximately $m \times h \times n \times w$; when doing the convolution with the two 1D Gaussian filters separately, the number of multiply-accumulate operations is approximately $m \times h \times n + m \times n \times w$. Therefore, the computational complexity of doing the convolution separately is much less than doing it with the 2D Gaussian kernel.

3.2 Question-3

The second-order derivative of the Gaussian kernel, as shown in Figure 1, can be applied to images to either localize edges at zero-crossing points (which is more accurate than using the first-order derivative), or detect blob-shaped patterns. For edge detection, it can be observed that the difference in amplitude between the outer regions and the center is huge such that the gradient in the image (difference in pixels) can be significantly captured. For blob detection, with different values of σ, the spread of the central area can be controlled, and the difference between the outer rim of the blob and the central area is scaled and signified for the pixels in the central area in the output image.

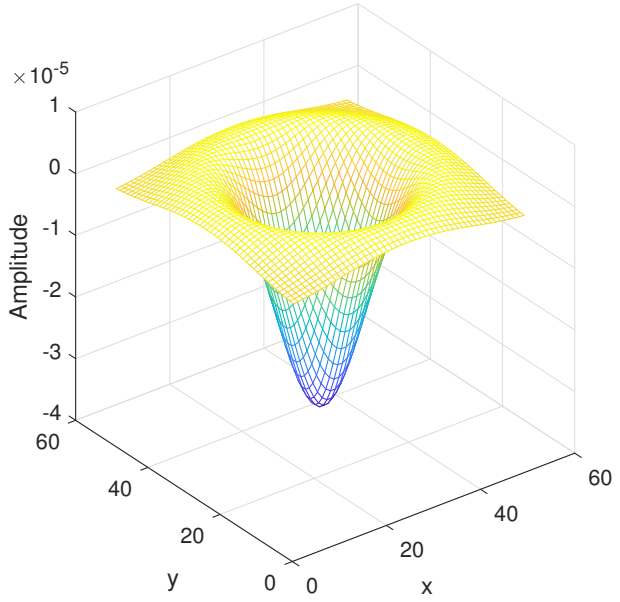

Figure 1: *Visualization of the second-order derivative of the Gaussian filter ($\sigma = 10$).*

3.3 Question-4

Without the loss of generality, we only consider the real part of the Gabor filter here:

- λ: λ is the wavelength of the cosine function and is measured in terms of pixels. It controls the spatial scale of one full period of the obtained kernel.
- θ: θ is the direction of the frequency of the cosine function. It controls in which direction we want to look for the carrier frequency in the given signal.
- ψ: ψ is the phase offset of the cosine function, it ranges between $[-\pi, \pi]$. When it is $-\pi$, π or 0, it remains the same and is symmetric with respect to the origin point. When it equals $\frac{\pi}{2}$ or $-\frac{\pi}{2}$, it is anti-symmetric with respect to the origin point.
- σ: σ is the scale parameter of the Gaussian kernel and determines the size of the non-zero part of the kernel. It can determine the size of the kernel pattern jointly with λ.
- γ: γ controls the spatial ratio of the kernel pattern to the support. When γ equals 1, the obtained pattern is square. It controls the support of the kernel in the same way as σ by projecting the unit Gaussian kernel to a warped one.

3.4 Question-5

As we can infer from Figure 2, changing θ does affect the pattern orientation of the filter, and larger σ indeed leads to a larger kernel which contains more stripe patterns. Moreover, when γ increases from 0.3 to 1, the kernel supports gradually change from a skew shape to a square shape, which also proves our interpretation of γ in subsection 3.3.

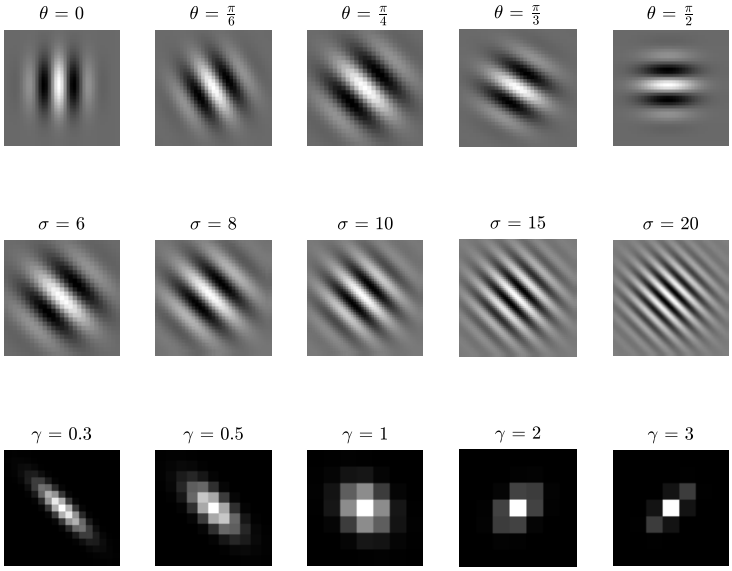

Figure 2: *The visualization of Gabor filter with different value of θ, σ and γ.*

4 Applications in image processing

4.1 Question-6

1. The PSNR between *image1.jpg* and *image1_saltpepper.jpg* is 16.1079 dB.
2. The PSNR between *image1.jpg* and *image1_gaussian.jpg* is 20.5835 dB.

4.2 Question-7

1. The denoised images obtained by applying box filtering and median filtering on *image1_saltpepper.jpg* and *image1_gaussian.jpg* with different filter sizes are shown from Figure 3 to Figure 6.

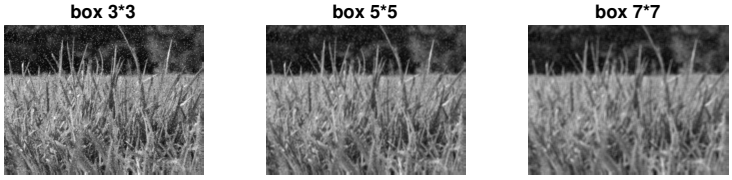

Figure 3: *Denoising saltpepper.jpg by applying box filtering of different sizes.*

Figure 4: *Denoising gaussian.jpg by applying box filtering of different sizes.*

2. The PSNR values for the denoised images are shown in Table 1, from which we can infer that the filter size negatively correlates with the PSNR. Specifically, the larger the filter size, the fuzzier the image and the worse the photo quality.

Figure 5: *Denoising saltpepper.jpg by applying median filtering of different sizes.*

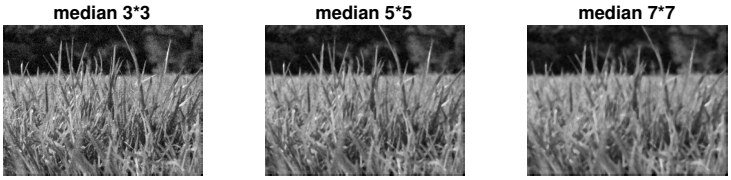

Figure 6: *Denoising gaussian.jpg by applying median filtering of different sizes.*

Table 1: *PSNR [dB] for the denoised images.*

	box 3×3	box 5×5	box 7×7	median 3×3	median 5×5	median 7×7
saltpepper.jpg	23.3941	22.6410	21.4220	**27.6875**	24.4957	22.3722
gaussian.jpg	**26.2326**	23.6610	21.9441	25.4567	23.7983	22.0765

3. The median filter is better for the salt-and-pepper noise for two reasons. First, the salt-and-pepper noise is approximately equal in magnitude but randomly distributed at different locations. There are both contaminated and clean points in the figure, therefore, the median filter can replace the contaminated points with clean points. Secondly, the mean value of the noise is not zero, so it is not suitable for mean filtering. On the other hand, the box filter is better for the Gaussian noise, as the probability density function of Gaussian noise is approximately a normal distribution with a mean of zero, so we can use the mean filter to process the noise. In addition, because all points are contaminated, the median filter cannot be selected to the correct clean point.

4. We choose different window sizes and standard deviations and we find that when choosing $\sigma = 0.83$ and window size = 3×3, we get the best denoising effect (see Figure 7). Under these parameters, the PSNR is 26.7004 dB.

Figure 7: *Denoising gaussian.jpg by applying Gaussian filtering of* $\sigma = 0.83$ *and window size* $= 3 \times 3$.

5. The PSNR values under different Gaussian filters with diverse sizes and standard deviations can be seen in Table 2. We can conclude that under the same window size, there is an optimal standard

deviation which can make the best denoising effect.

Specifically, if the standard deviation is small, the smoothing effect is not obvious because of the tall and narrow 2D Gaussian image (the value of the central element is much larger than those of other elements in the Gaussian function). Hence, it can be found that when $\sigma = 0.2$, the PSNR value is almost the same as the one of the image without denoising (i.e. 20.6 dB, see subsection 4.1). On the other hand, when the standard deviation becomes larger, the filtering effect is more obvious because of the shorter and wider 2D Gaussian kernel (similar to the average template). In fact, the best standard deviation should be neither small nor big, so the optimal standard deviation should fall in $[0.8, 0.9]$ in this case.

Table 2: *PSNR [dB] for the denoised images using Gaussian filter with different σ and window sizes.*

Size \ σ	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
3×3	20.5835	20.7109	22.0248	24.2888	25.8936	26.5417	**26.6963**	26.6748	26.6024
5×5	20.5835	20.7109	22.0249	24.2954	25.9307	26.5893	26.6624	26.4639	26.1597
7×7	20.5835	20.7109	22.0249	24.2954	25.9307	26.5894	26.6599	26.4446	26.0981

6. (a) Both the box filter and the Gaussian filter are linear filters. As discussed earlier, the box fitter is better for the Gaussian noise compared to the median filter. Further, the Gaussian filter uses a weighted average of the neighbour pixels instead of the mean average, which can improve the effect of denoising the Gaussian noise. When the size of filers are all 3×3, we have:

$$PSNR_{Gaussian}(26.7004) > PSNR_{Box}(26.2326) > PSNR_{Median}(23.3941)$$

For the median filter, it is a non-linear filter. It replaces the value of each pixel with the median of the gray-scale values of all pixels in the neighborhood of that point. It is very effective in smoothing impulse noise (e.g. salt-and-pepper noise) and it protects the sharp edges of the image. When the size of filers are all 3×3, we have:

$$PSNR_{Median}(27.6875) > PSNR_{Box}(23.3941)$$

(b) When white noises are added to high-, medium- and low-frequency parts of an image separately, the image quality with high-frequency noises is better than the other two cases, but the PSNR values of the three are the same because human eyes are more sensitive to low-frequency noise. For example, the PSNR values for Figure 8a and Figure 8b are both around 25.46 dB, but Figure 8b has a better quality because most of its noises are of high frequency.

(a) **(b)**

Figure 8: *Denoising Gaussian noise by Median filter with size $= 3 \times 3$ (a) and by Gaussian filter with $\sigma = 0.565$ and size $= 3 \times 3$ (b).*

4.3 Question-8

The visualizations of the gradient in the x- and y-direction, the gradient magnitude and gradient direction of *image2.jpg* are all displayed in Figure 9. As could be inferred from the top-left figure, the gradient

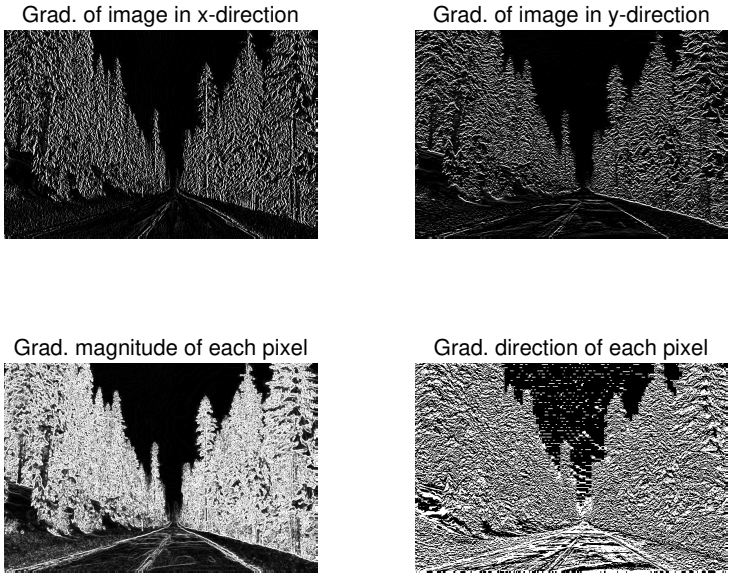

Figure 9: *Gradient in x- and y-direction, gradient magnitude and gradient direction of image2.jpg by applying the Sobel kernels.*

of the (input) image in the x-direction conveys information about the vertical edges in the image, while the top-right figure showing the gradient of the image in the y-direction conveys information about the horizontal edges. As for the bottom-left figure showing the gradient magnitude of each pixel, it concatenates the information from the x- and y-direction and hence strengthens the edges from both directions. Finally, the bottom-right figure reflects the gradient direction of each pixel in the image, and it conveys information about the orientation of edges with respect to the pixels.

4.4 Question-9

1. The visualizations of the results using the three methods are shown in Figure 10.

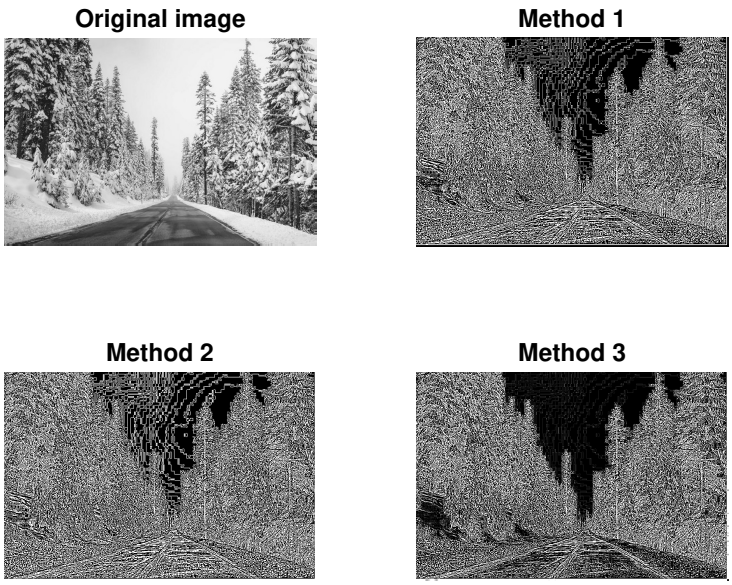

Figure 10: *Comparison of the three methods for computing the Laplacian of Gaussian.*

2. Among the three methods, the third method performs the best as it is least affected by noises in the image and the detected edges are very clear. The first method is worse than the third method because there are more noises in the output, which is obvious particularly in the sky and on the

road of the original image. The second method is the worst since it is most affected by the noises and there is no clear boundary between the road and the forest in the image.

3. For the first method, it is vital to first convolve the image with a Gaussian filter to smooth the image and mitigate the effect of the noises, otherwise, the noises would further affect the performance of the edge detection via applying the Laplacian filter.

4. According to [1], the optimal ratio between σ_2 and σ_1 is 1.6, where the optimal trade-off is made to minimize the bandwidth and maximize the peak sensitivity. Using two different standard deviations for the two Gaussian filters results in a band-pass filter which can preserve the interesting spatial information and simultaneously suppress the high-frequency spatial noise.

5. As shown in Figure 10, even though the third method has the best performance among the three methods, it is still difficult to distinguish between the forest and the road. To improve performance, supplementary processing should be done (e.g. lane detection), since the two edges of the road consist of (semi-)straight lines which can be used to further isolate the road.

4.5 Question-10

1. The results of the algorithm running over all test images with the provided parameter settings are shown from Figure 11a to Figure 11f. From the results, we can see that in the provided parameter settings, simple images like *robin-1*, *robin-2* and *polar-bear* work well enough, but for *Kobi* which has a complex background and *cows* whose foreground contains different patterns, the algorithm does not work well. Moreover, when it comes to *science-park*, we can readily see that the model can discern the blob pattern from the stripe-like pattern in the upper part, but does not generalize well where the blob pattern is not present. This indicates that we might need to tune the parameters to suit for complex situations in order to get the correct segmentation.

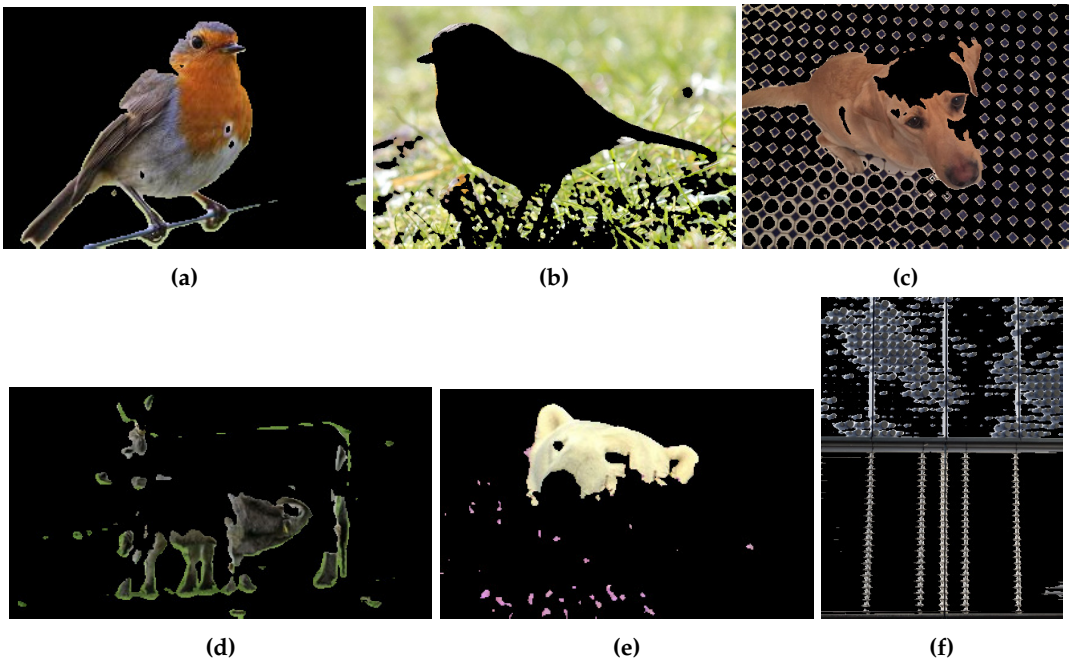

Figure 11: *The segmentation results of robin-1 (a), robin-2 (b), Kobi (c), cows (d), polar-bear (e) and science-park (f) with the provided parameter settings.*

2. After a lot of tuning works, we find out that λ and θ have basically no contribution in improving the final results, but they do have an non-trivial effect on the segmentation results. For example, when we set the upper bound of θ from $\frac{\pi}{2}$ to π, the circle in the background of *Kobi* will be captured in the full direction. Thus, we focus on tuning σ on the *Kobi*, *cows* and *science-park* images. We set σ to be $[1, 2, 3, 4]$ and σ of the smoothing Gaussian kernel to be 2 for *Kobi*, and set σ to be $[0.5, 1, 1.5, 2]$ and σ of the smoothing Gaussian kernel to be 7 for *cows*. For *science-park*, we set σ to $[4, 6]$ to avoid

over-emphasis on the small patterns appeared in both the foreground and background. The kernel size and the standard deviation of Gaussian smoothing are also changed to 7 and 5 respectively for the same purpose. The comparison of segmentation results can be seen in Figure 12.

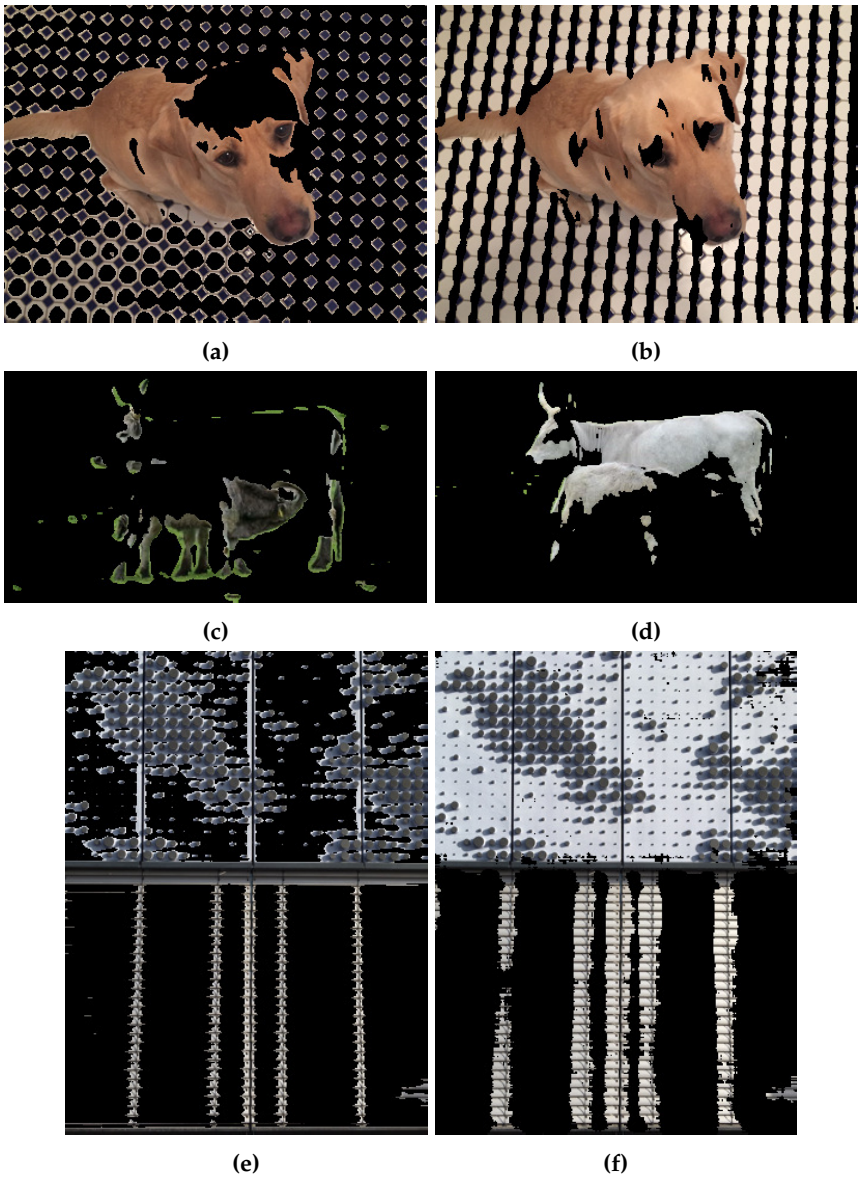

Figure 12: *The comparison of the segmentation results of provided settings (a, c, e) and the tuned settings (b, d, f).*

3. We presents three example results of *Robin-1* and *Polar* with or without smoothing in Figure 13. It is clear that smoothing makes the results better as it smooths away many unusual local variations and thus increase the objectiveness of the final results. The reason that smoothing works is because it neutralizes the extreme noises in the feature vector, and makes most of the features that are next to each other fall in the same cluster of the k-means clustering.

5 Conclusion

In previous sections we study on several image processing filters. The box filter is easy to understand and implement, and it is good at denoising the Gaussian noises. The median filter is a non-linear filter which is very effective in processing the salt-and-pepper noise. The Gaussian and Gabor filters are also important in image processing. We first lay the mathematical foundation of these two filters in section 3,

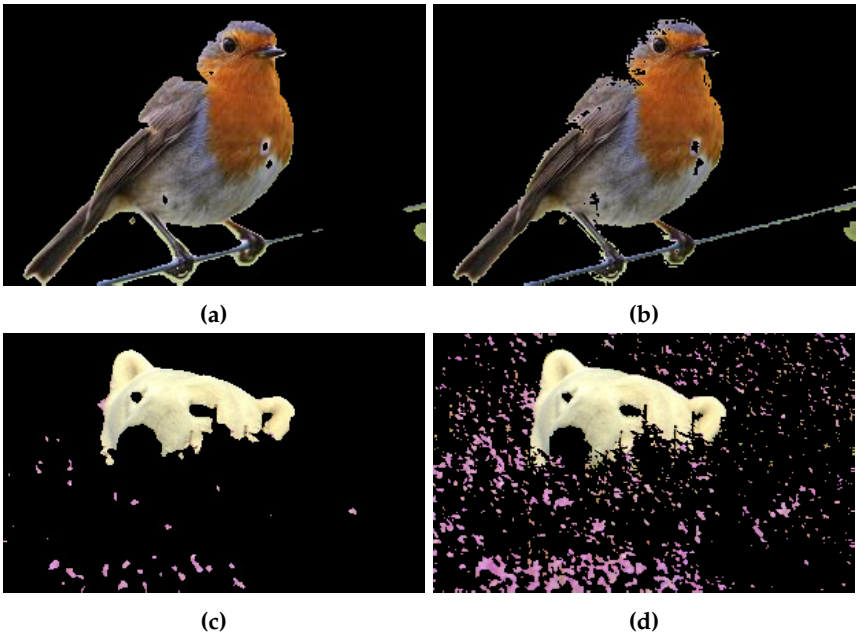

Figure 13: *The comparison of the segmentation results with smoothing (a, c) and without smoothing (b, d).*

then we use them to perform image denoising, edge detection, and foreground-background separation and discussed how to get a better result under different parameters.

References

[1] D. Marr and E. C. Hildreth, "Theory of edge detection.," *Proceedings of the Royal Society of London. Series B, Biological sciences*, vol. 207 1167, pp. 187–217, 1980.