Computer Vision 1 Assignment 2

Dmitrii Krasheninnikov
University of Amsterdam
11719230
dmkr0001@gmail.com

Akash Raj Komarlu Narendra Gupta
University of Amsterdam
11617586
akashrajkn@gmail.com

1 Introduction

This report describes completion of the tasks of the second assignment of the Computer Vision 1 course at the University of Amsterdam. First, we examine the theory of neighborhood processing and low-level filters, covering convolutions, correlations, Gaussian and Gabor filters in detail. Next, we apply the theory of low-level filters to image denoising and edge detection. Finally, we apply the theoretical knowledge from previous sections to background segmentation.

2 Neighborhood Processing

Question 1

1. Correlation expresses similarity of two signals as they are shifted by one another. (It is maximum when both the signals are identical.) Convolution expresses the amount of overlap of one signal as it is shifted over another. The basic difference between the two is that the kernel is flipped horizontally and vertically in convolution compared to correlation (and vice versa).

2. Correlation and Convolution are equivalent when the kernel **h** is symmetric. An example is the Gaussian kernel. In the case of symmetric kernels, a combination of horizontal and vertical flips will result in the same kernel.

3 Low-level filters

Question 2

Convolution with a 2D Gaussian filter is equivalent to convolving the image with 1D Gaussian filter along x-axis and then along y-axis (derivation shown below). The result will be the same in both the cases. In terms of computational complexity, convolution with 2D filter is generally less efficient than convolving twice with 1D filters, since the former requires n^2 multiplications and the latter requires $2n$ multiplications, where n is the kernel dimension along one axis.

$$\begin{aligned} G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2}) \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-x^2}{2\sigma^2}) * \frac{1}{\sqrt{2\pi}\sigma}\exp(\frac{-y^2}{2\sigma^2}) \\ &= G_\sigma(x) * G_\sigma(y) \end{aligned}$$

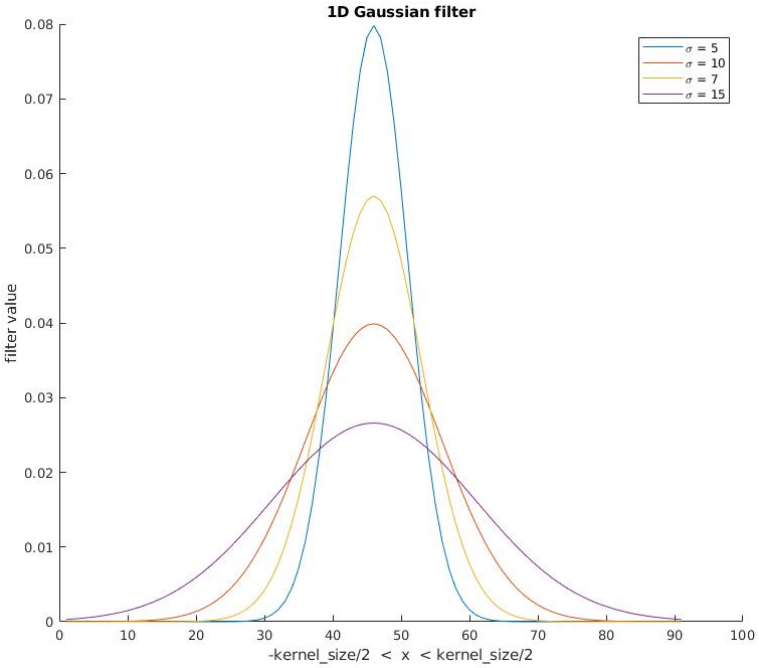

Figure 1: 1D Gaussian filter for different σ values.

Question 3

The second derivative filters are generally called Laplacian filters. The Second derivative of Gaussian filter helps in finding edges (rapid intensity changes). It **reduces its sensitivity to noise**. The filter will be zero at points away from edges, positive on one side of the edge and negative on the other side of the edge. All of these properties make it attractive to use for edge detection. The laplacian is given by,

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

And the Laplacian of Gaussian (second derivative filter) is given by,

$$\text{Laplacian of Gaussian} = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Question 4

For the sake of comparison, we use a default set of parameters, $[\sigma, \theta, \lambda, \psi, \gamma] = [10, 0, 100, 0, 1]$. The filter generated is shown in the leftmost plot in Figure 2

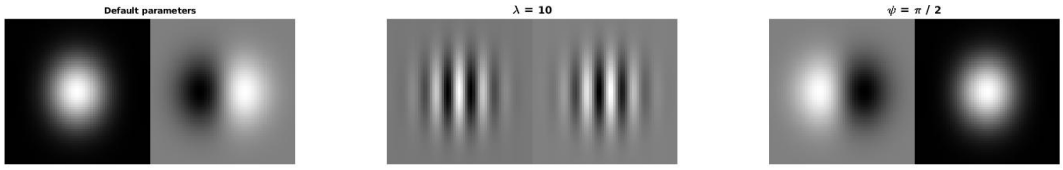

Figure 2: The first figure on the left shows a Gabor filter with (default) parameters $[\sigma, \theta, \lambda, \psi, \gamma] = [10, 0, 100, 0, 1]$. The middle figure shows the filter with $\lambda = 10$. The rightmost figure shows the filter with $\psi = \pi/2$. Note that in the third figure, a phase offset of $\pi/2$ results in the inversion of real and imaginary components.

- λ is the wavelength of the carrier signals (their central frequency). The value of λ establishes the kind of sinusoidal wave the filter will respond best. As λ value increases, the ellipses (of the gabor filter) widens and vice versa. As the frequency bandwidth increases, more frequencies are captured by the filter.
- θ dictates the orientation of the (Gaussian) envelope. A value of $\theta = 0$ indicates no rotation and the ellipses are vertical.
- ψ denotes the phase offset of the carrier signal. It controls how much the ellipses have to be shifted with respect to the center. A value of $\psi = \pi/2$ results in inversion of the real and imaginary components. This is shown in the rightmost plot of Figure 2.
- σ is the standard deviation of the Gaussian envelope. It denotes the spread of the envelope.
- γ decides how elongated the ellipses need to be, i.e., it controls the aspect ratio of the Gaussian envelope. As γ decreases, the ellipses are elongated.

Question 5

Figure 3 shows the real and imaginary components of the Gabor filter for different values of Orientation. As θ is varied, the direction (orientation) changes.

Figure 4 shows the real and imaginary components for different values of standard deviation. As σ increases, it filter becomes finer.

Figure 5 shows the real and imaginary components for different values of the aspect ratio. As γ decreases, the ellipses are elongated.

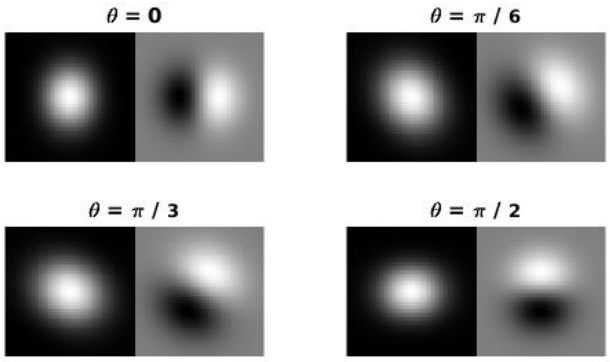

Figure 3: Figure shows the real and imaginary components of the Gabor filter as the orientation, θ is varied.

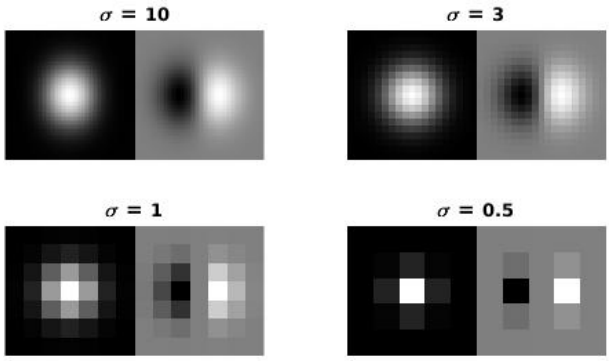

Figure 4: Figure shows the real and imaginary components of the Gabor filter as the standard deviation, σ is varied.

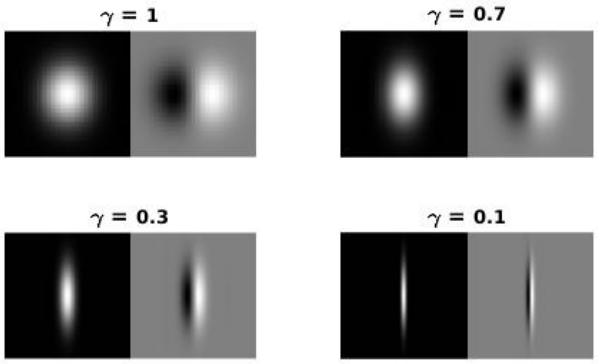

Figure 5: Figure shows the real and imaginary components of the Gabor filter as the aspect ration, γ is varied.

4 Applications in Image processing

Question 6

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

Question 7

7.1 Figure 6 shows the results of denoising an image with salt-and-pepper noise with Box and Median filters with kernel sizes 3x3, 5x5, 7x7. The results of denoising an image with Gaussian noise with similar filters are shown in Figure 7.

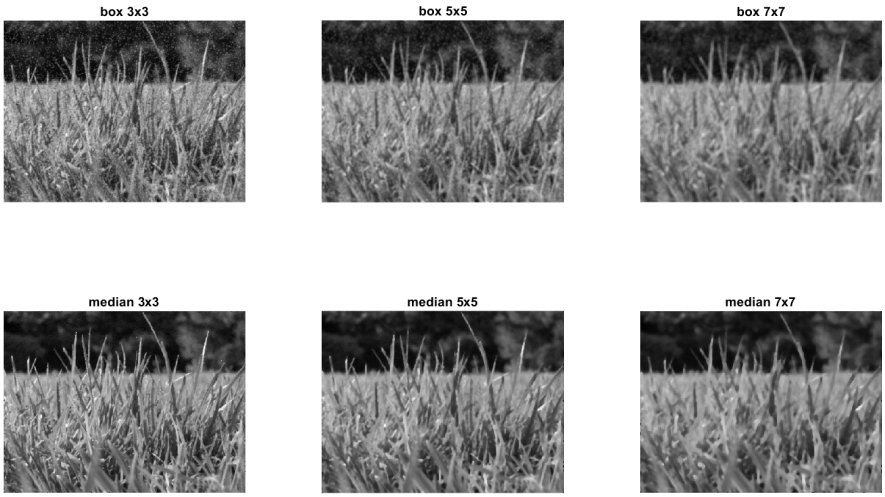

Figure 6: An image with salt-and-pepper noise denoised with Box and Median filters with kernel sizes 3x3, 5x5, 7x7.

7.2 Table 1 shows PSNR for every denoised image from Figures 6 and 7. We observe that as the filter size increases, PSNR decreases for both Box and Median filters, and both images being denoised.

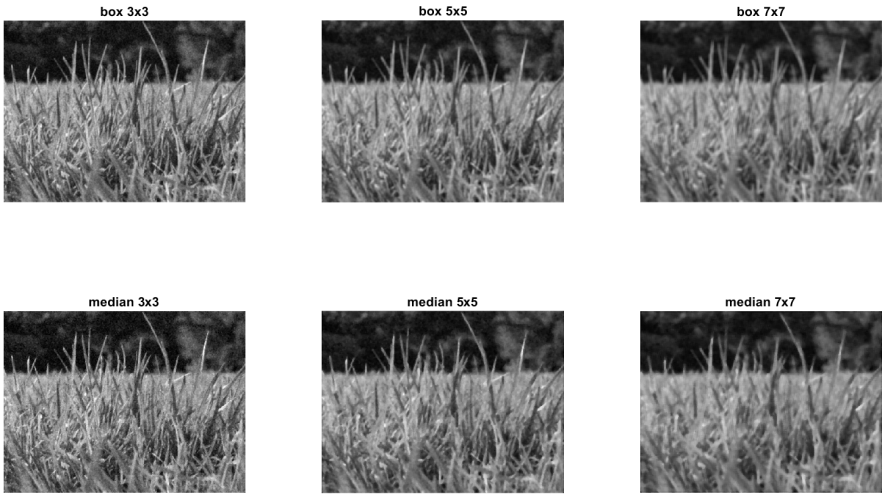

Figure 7: An image with Gaussian noise denoised with Box and Median filters with kernel sizes 3x3, 5x5, 7x7.

	Image with salt-and-pepper noise	Image with Gaussian noise
Box 3x3	23.3952	26.2351
Box 5x5	22.6421	23.6620
Box 7x7	21.4224	21.9448
Median 3x3	27.6875	25.4567
Median 5x5	24.4957	23.7983
Median 7x7	22.3722	22.0765

Table 1: PSNR of the images denoised with Box and Median filters

This is expected, since as the filter size increases, RMSE increases as well since the pixel values are less correlated in larger image patches than in smaller patches.

7.3 Median filters are useful in case of presence of salt-and-pepper noise in an image. As the true pixel values in an image patch are correlated, and the value of a damaged pixel is typically very far and is not correlated with the true pixel value, replacing the pixel value with the median of the pixel values in the image patch results in removal of the extreme value.

Box filters are well-suited to deal with stationary zero mean Gaussian noise, since for small image patches we expect that variance of the true pixel values is smaller than variance of the Gaussian noise. Thus applying the box filter moves the added noise towards its zero mean while only weakly affecting the true pixel value.

7.4 Figure 8 shows the results of denoising an image with Gaussian noise with Gaussian filters with kernel size 3x3 and standard deviations of 0.5, 1, and 2. The 3x3 kernel size was chosen for two reasons: first, kernels of this size qualitatively worked best for Box and Median filters; second, the image is fairly low-res at 512x384 and is high-detail, so the "pixels in the image patch are strongly correlated" assumption would have likely been broken for larger patch sizes.

7.5 Table 2 shows PSNR for every denoised image in Figure 8. We observe that when σ increases from .5 to 1, PSNR increases, meaning that RMSE decreases. This is likely due to us getting closer to optimal denoising by putting slightly more weight on the surrounding pixels – a Gaussian filter with $\sigma = .5$ does very little denoising (weights of pixels near the center are 0.0838, and weights of pixels in the corners of the filter are 0.0113).

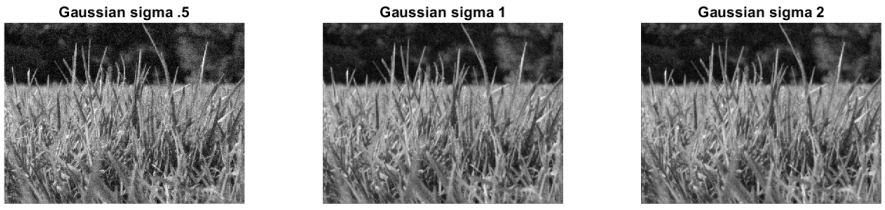

Figure 8: An image with Gaussian noise denoised with Gaussian filters with kernel size 3x3 and standard deviations of 0.5, 1, and 2.

	Image with Gaussian noise denoised with Gaussian filtering
Gaussian 3x3, $\sigma = .5$	24.2902
Gaussian 3x3, $\sigma = 1$	26.6050
Gaussian 3x3, $\sigma = 2$	26.1496

Table 2: PSNR of the images denoised with Gaussian filtering

However, when σ increases from 1 to 2, PSNR decreases, meaning that RMSE increases. A 3x3 Gaussian filter with $\sigma = 2$ is very similar to a 3x3 box filter, and we observe similar PSNR and image quality. The increase in RMSE is likely due to reasons similar to the reasons of RMSE increase when the box filter kernel size is increased (listed above).

7.6 Box and Gaussian filtering are useful for denoising images with static zero mean noise such as Gaussian noise. A Gaussian filter with a very large σ is basically equivalent to a box filter. Median filtering is useful for handling noise with outlier values, such as images with salt-and-pepper noise. One can often see the difference between the images with similar PSNR denoised with different filters. For example, the image with salt-and-pepper noise denoised with a 3x3 box filter and the image with Gaussian noise denoised with 5x5 median filter give similar PSNRs (23.395 and 23.798) but the resulting images are very different qualitatively (this is largely due to incorrect noise type and filter type match).

Question 8

Figure 9 shows the gradients of *image2.png* in x and y directions, and gradient direction and magnitude for each pixel.

Question 9

9.1 Figure 10 shows the results of applying LoG filters computed with 3 different methods to *image2.jpg*.

9.2 Each of the 3 methods successfully detects edges and corners; the results of the first two methods are indistinguishable on a max-min normalized plot; the results of the third method seem slightly less noisy, which is noticeable on the road edges and at the places without edges.

Mathematically the first and the second methods are equivalent (up to hyperparameter values), except the first method uses a more computationally inefficient algorithm, since the image is sequentially convolved with two kernels, as opposed to being convolved with one kernel which itself is a combination of two kernels. The third method is not mathematically equivalent to the other two methods, but is an approximation of them.

9.3 The Laplacian filter is extremely sensitive to noise, so it is important to remove noise before convolving the image with Laplacian. Convolving the image with the Gaussian filter achieves this.

9.4 The best ratio between σ_1 and σ_2 is approximately 1.6 [1]. The following parameters are used in our implementation: $\sigma_1 = .8$, $\sigma_2 = .5$.

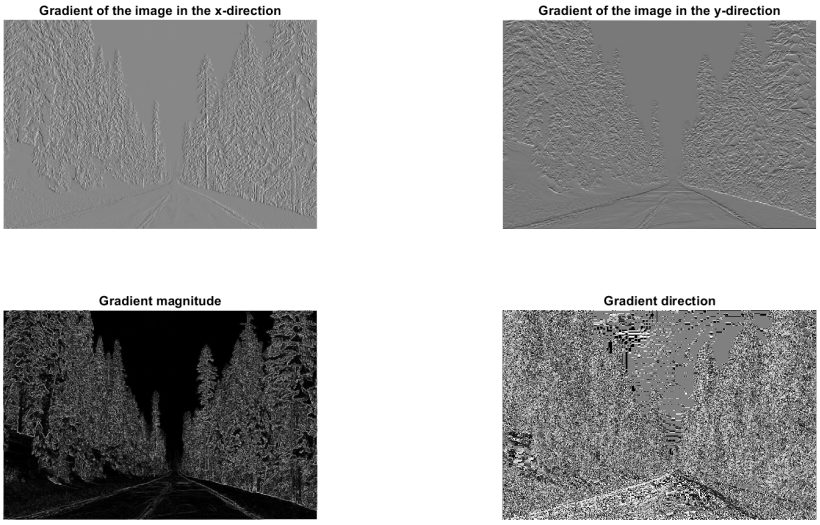

Figure 9: Gradients of *image2.png* in x and y directions, and gradient direction and magnitude.

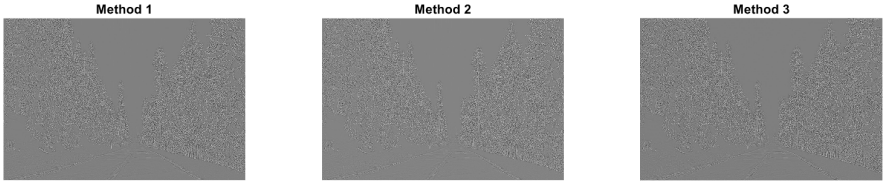

Figure 10: Results of applying LoG filters computed using 3 different methods to *image2.jpg*.

9.5 Visually the first-order method provides gradients of larger magnitudes; this visual intuition is confirmed by looking at the top-10 largest values of gradient magnitudes of the first order method and comparing them with the top-10 largest values of the results of the second order methods.

9.6 Adding filters that respond to diagonal edges and corners would likely improve the result of edge detection even further: filters detecting diagonal edges would be very useful for improving performance in the road region, and filters detecting corners would be helpful for the trees and snow by the road.

Question 10

10.1 We observe that the provided parameter settings work well for some input images, but work badly for others. Segmentation of kobi image (Figure 11) works well with the provided parameter settings. We can see the two different segments almost exactly corresponding to the dog and the floor. However, the given parameter settings do not work well with *Cows.png* – segmentation is poor as seen in Figure 12.

10.2 The optimal parameter settings change with respect to difference in textures – high frequency textures with small elements like grass in the cow image require small λ and small σ Gabor filters, and low frequency textures with larger elements like the floor in the kobi image require filters larger λ and σ.

For *kobi.png*, the provided parameter settings work well. For *cows.png*, we tried different parameter settings. After exhaustive search, we could only find a parameter setting which worked better than the default settings. Figure 13 shows the segmentation using the new parameter settings. For this particular output, we used $\sigma = [0.5, 2]$. We reduced the standard deviation of one of the filters in the filter bank. Our intuition is that as the spread of the Gaussian envelope decreases, it penalizes the difference in textures.

Other parameter settings included decreasing the step size of θ. On *kobi.png*, we did not find any significant improvement.

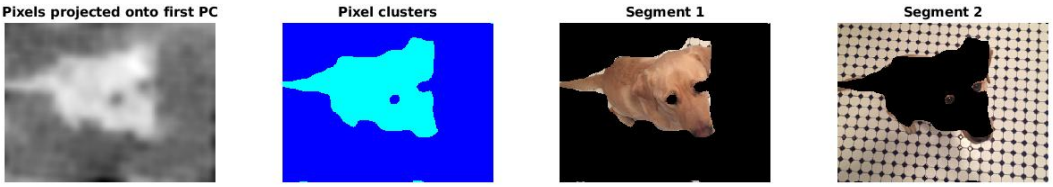

Figure 11: Foreground background segmentation on *kobi.png* using the default parameters.

Figure 12: Foreground background segmentation on *cows.png* using the default parameters.

Figure 13: Foreground background segmentation on *cows.png* using the adjusted parameters.

10.3 Smoothing helps treating textures as single pixel clusters. When smoothing is applied (by using a first order Gaussian filter), it blurs the parts of texture that should not be their own pixel clusters. This produces pixels with intermediate values between cluster means. When smoothing is not applied to the magnitude images we observe that segmentation fails due to pixel clusters being decided primarily based on absolute color intensity and not texture, since small patches of very high or low intensity do not get smoothed and get prioritized in pixel cluster formation. For example, Figure 18 demonstrates how on *kobi.png* the (yellowish) dog and the (yellowish) parts of the floor are clustered together, and each black dot on the floor becomes its own pixel cluster.

Conclusion

In this report we reviewed the theoretical approaches in computer vision related to low-level filters and their applications to image denoising, edge detection and background segmentation. We enjoyed the vast majority of the assignment, except for hand-tuning parameters in Question 10.

Figure 14: Foreground background segmentation on *polar.png* using the default parameters.

Figure 15: Foreground background segmentation on *robin-1.png* using the default parameters.

References

[1] David Marr and Ellen Hildreth. Theory of edge detection. In *Proc. R. Soc. Lond. B*, volume 207, pages 187–217. The Royal Society, 1980.

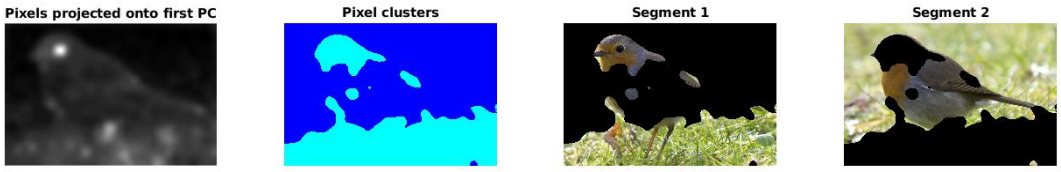

Figure 16: Foreground background segmentation on *robin-2.png* using the default parameters.

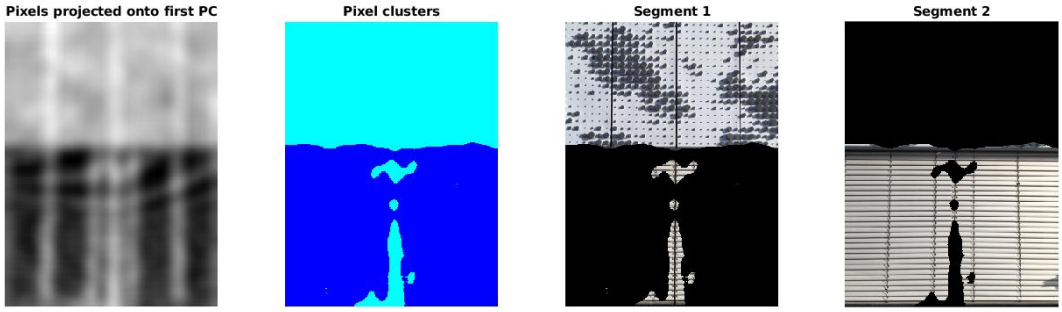

Figure 17: Foreground background segmentation on *kobi.png* using the default parameters.

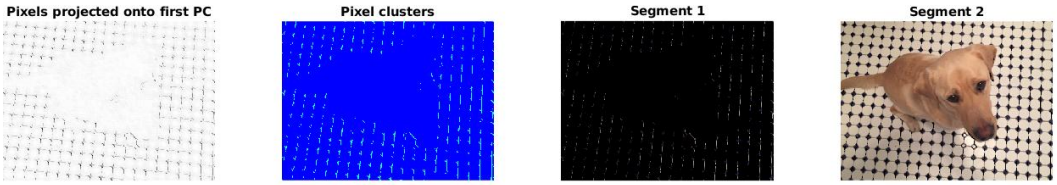

Figure 18: Foreground background segmentation on *kobi.png* without smoothing.