UNIVERSITY OF AMSTERDAM

ASSIGNMENT 2

Neighborhood Processing & Filters

September 26, 2021

Students:
Victor Wernet
UvAnetID 13367366

Sameer Ambekar
UvAnetID 13616064

Apostolos Panagiotopoulos
UvAnetID 14021900

Tutor:
Dr. Shaodi You

Group:
12

Course:
Computer Vision 1

1 Introduction

In this assignment we explore kernels and their use as filters for various computer vision applications, such as denoising, edge detection and background separation. In more detail, in Section 2 we implement Gaussian and Gabor filters and provide insights on their parameter selection and usage. In section 3, we use Gaussian filtering, along with box and median filtering, to denoise `image1.jpg` when injected with salt-and-pepper and Gaussian noise. Section 4 utilizes Gaussian filters for edge detection. We also investigate how changing the hyper parameters, adding pre processing steps like Gaussian filters to remove noise brings changes to the outputs and to what to extent. Finally, Section 5 utilizes Gabor filters as a feature extraction method and uses k-means to separate the images in two parts, foreground and background.

2 Low-level filters

Question 1a: Difference between convolution and co relation filters: Both the filters are used in image operations to convolve a kernel over an image to produce an output correspondingly. But in convolution we transpose the kernel in every dimension and do the multiplication using point wise method. In co relation we make use of simple multiplication if the given kernel to the input image. Due to this property convolution is associative that is A*(B*C) can also be computed as (A*B)*C.

Question 1b: Yes both are equivalent when we have 'h'. We can observe this through the formula. This section describes Gaussian filters, in particular 1D-2D Gaussian filters, and 1D-2D Gabor filters. The Gaussian filter is an operator used for smoothing using a 2D convolution [21] operator that is used to blur images and remove detail and noise [5]. Unlike the mean filter, it uses a different kernel that represents the shape of a Gaussian (Bell-shaped) hump [15]. The Gabor filter, on the other hand, is used most often with texture analysis [6]. Essentially, the frequency of a given content is analyzed in an image in a specific direction, as such this technique is found to also share similarities with the visual cortex [14].

2.1 Question 3

By designing a second order kernel, we essentially use this to make line structures more pronoun, or extract more detail. The second order kernel will help us then in edge detection [7]. The

other reason for doing so is also due to the fact that Gaussian derivative kernels can also act as a bandpass filter [8]. In simple terms, this means that it can be used as a filter to attenuate signal frequencies outside the range (band). This is specifically useful in image analysis, since they can be used to denoise images while at the same time reducing low-frequency artifacts. In Figure 1 we can see how the different order of derivative behave with respect to the 1D Gaussian [18].

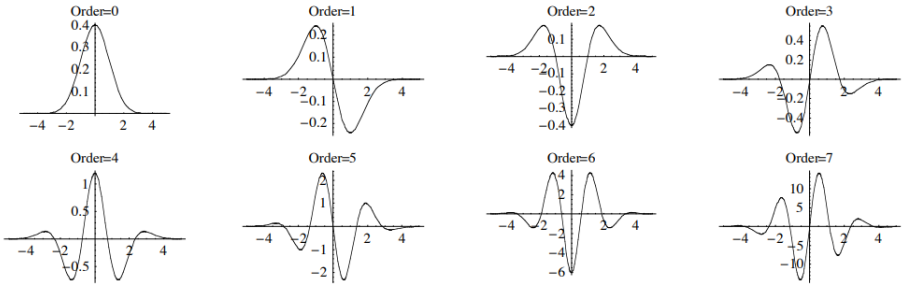


Figure 1: Plots of the 1D Gaussian derivative function for order 0-7

2.2 Question 4

In equations (1) and (2) we can see the 2D Gabor filters for the real and imaginary parts:

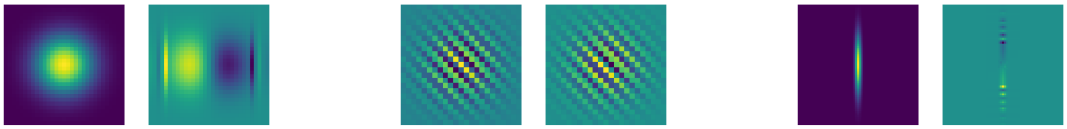
$$g_{\text{real}}(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2 * \pi \frac{x'}{\lambda} + \psi\right) \tag{1}$$

$$g_{\text{im}}(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \sin\left(2 * \pi \frac{x'}{\lambda} + \psi\right) \tag{2}$$

A control description of the parameters passed, can be described as follows: x' and y' are the first and second rows of the 2D rotation matrix $[\mathbf{x} \cdot \mathbf{cos}(\theta) + \mathbf{y} \cdot \mathbf{sin}(\theta)]^\intercal$ and y' is the second row, $[-\mathbf{x} \cdot \mathbf{sin}(\theta) + \mathbf{y} \cdot \mathbf{cos}(\theta)]^\intercal$. The λ controls the wavelength of the sinusoidal factor, the θ controls the orientation of the normal to the parallel stripes of the Gabor function. The ψ controls the phase offset, which essentially boils down to overlaying two instances of a periodic waveform on top of each other. The σ controls the standard deviation of the Gaussian envelope, where you can image for bigger values of σ means a less steep bell shape, and the opposite a sharper bell shape. The γ controls the spatial aspect ratio. In other words, it specifies the ellipticity of the Gabor function [10].

2.3 Question 5

Below we see 3 different plotted images using different values, in essence to verify our notion of what the parameters θ, σ, and γ control. Here can be seen visually the result for different values.



(a) $\theta = 0$, $\sigma = 5$, $\gamma = 1$ (b) $\theta = \pi/4$, $\sigma = 5$, $\gamma = 1$ (c) $\theta = \pi/2$, $\sigma = 10$, $\gamma = 10$

Figure 2: Different Gabor Values

3 Image Denoising

3.1 Quantitative evaluation

PSNR is a metric to quantitatively measure the discrepancy between an original image and a modified version of it. In the context of this assignment, we use PSNR to measure the discrepancy between an image and its enhanced counterpart.

Question 6. By definition, PSNR is the ratio of the maximum pixel intensity to the root mean square error (RMSE) between the original and the enhanced image, on the decibel scale. RMSE is a metric to measure the absolute discrepancy between the original and the enhanced image, by considering the difference of intensity values at every pixel. However, as a standalone metric, it can be misleading when comparing images with different peak (or mean) intensities. For instance, if a bright and a dark image have the same RMSE with their enhanced versions, the discrepancy in the bright pictures is going to be less apparent than in the dark ones, since the relative difference is smaller. Using the ratio between the maximum intensity and the RMSE is a good way to overcome this problem. Finally, it has been shown that the human eye perceives light in a logarithmic manner [19, 9], so measuring this ratio on the decibel scale expresses the image discrepancy more accurately from a human perspective.

Equation (18) implies that higher PNSR means lower RMSE, given the same reference image. Hence, higher PSNR means higher similarity between the enhanced and the original photo, which is of course a desired property. However, we should be cautious to use the same reference image and denoising algorithm when comparing noise effects using this metric, otherwise PSNR is not a good quality indicator [11]. For example, in Figure 3 we see `image1.jpg` injected with Salt-and-pepper and Gaussian noise, which gives a PSNR of 16.1 dB and 20.58 dB respectively. We can't infer thatthe second image is less noisy than the first, since we're talking about two different type of noises.

To reproduce results, run the `myPSNR.py` script.

```
cd path-to-lab2/code/Image_enhancment & python myPSNR.py
```

(a) No noise
PSNR = ∞

(b) Salt-and-pepper noise
PSNR = 16.10 dB

(c) Gaussian noise
PSNR = 20.58 dB

Figure 3: `image1.jpg` injected with noise.

3.2 Neighborhood processing for image denoising

We aim to denoise the images of Figure 3b and 3c using box, median and Gaussian filtering.

Question 7. First, we use box and median filtering for various kernel sizes. We present the original and denoised images in Figures 4 and 5, and the PSNR of the denoised images in Tables 1 and 2. We notice that PSNR decreases as the kernel size increases for both injected noises and both denoising algorithms. This implies that increasing the kernel size decreases the denoising performance, which is reasonable given the over-smoothed effect we see for 7x7 kernels in Figures 4a, 4b, 5a and 5b.

By closely inspecting Figure 4, we observe that, for the salt-and-pepper noise, median filtering provides the cleanest looking denoised image. This is also evident by the PSNR in Tables 1 and 3. This happens because the salt-and-pepper noise injects high noise in only some pixels, and median filtering replaces each pixel's value with the median value of the neighboring pixels. This completely removes the noisy pixels, since the noise is always to high to be the median value, and gives a crisp output. On the contrary, both box and Gaussian filtering average each pixel's neighborhood, which results in the noise to be attenuated locally but distributed to the neighborhood.

We also try denoising both images with Gaussian filtering and present denoised images in Figures 4c and 5c. PSNR is reported in Table 3. We chose the kernel size to be $k = 4\sigma + 1$, where σ is the standard deviation, since the kernel elements outside that range are either zero or negligible [2, 1]. Again we observe that as the standard deviation σ increases, the PSNR decreases. This implies worse denoising, which is evident by the over-smoothing for $\sigma = 3$ in Figures 4c and 5c.

These experiments indicate the choice of the denoising filter is sensitive to the injected noise. Median filter is more suitable for the salt-and-pepper noise since it removes outliers, while the gaussian filter is more suitable for the Gaussian noise since it averages the noise around a region. Box filter could be considered as a simple and chunky version of the gaussian filter, since it average neighboring pixels as well, but places the same weight on all neighbor pixels, resulting in artifacts on the output image.

To reproduce results, run the `denoise.py` script.

```
cd path-to-lab2/code/Image_enhancment & python denoise.py
```

	Box filtering	**Median filtering**
3	23.39	27.86
5	22.63	24.67
7	21.41	22.55

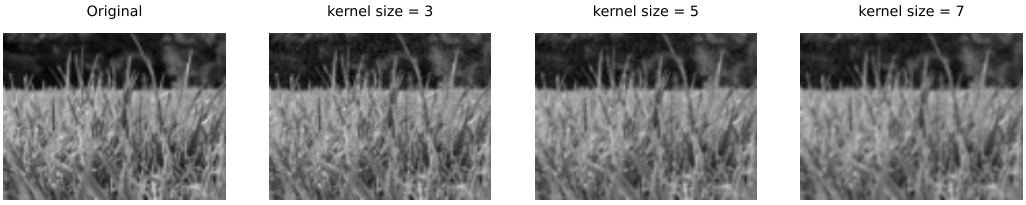
Table 1: PSNR after denoising `image1_peppersalt.jpg`

	Box filtering	**Median filtering**
3	26.22	25.53
5	23.65	23.94
7	21.93	22.24

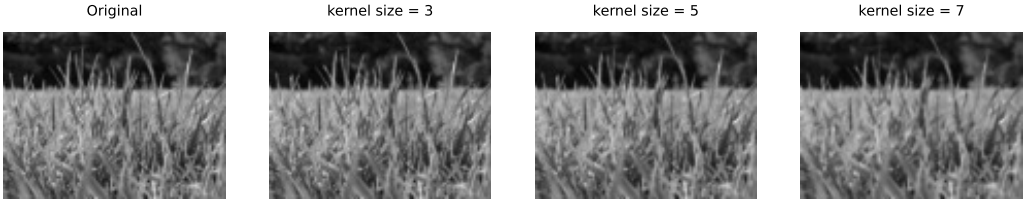
Table 2: PSNR after denoising `image1_gaussian.jpg`

	Salt-and-pepper noise	**Gaussian noise**
1	23.88	26.41
2	22.34	23.07
3	20.93	21.31

Table 3: PSNR after denoising both images with Gaussian filtering



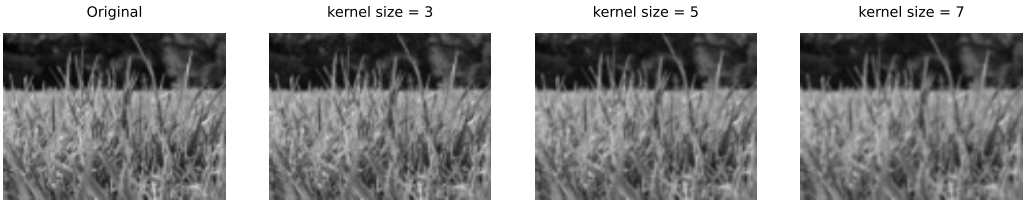
(a) Box filtering



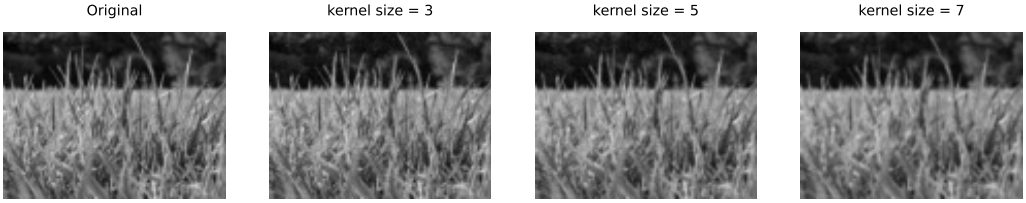
(b) Median filtering

(c) Gaussian filtering

Figure 4: Denoising `image1_saltpepper.jpg`



(a) Box filtering



(b) Median filtering

(c) Gaussian filtering

Figure 5: Denoising `image1_gaussian.jpg`

4 Edge Detection

The derivative of a gray image can be used to calculate edges. Since it as the edges where the change of intensity takes place and derivative helps us to capture that change of values.

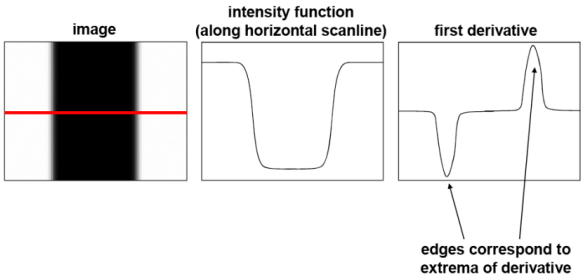
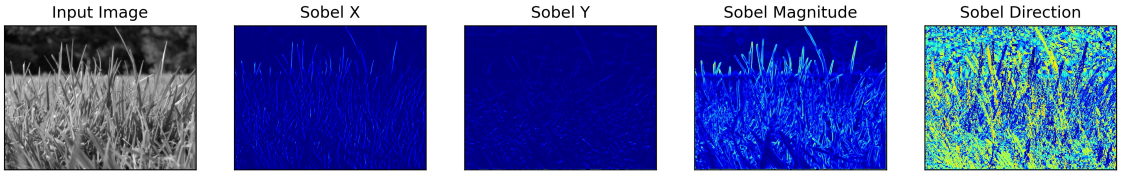


Figure 6: Representation of an edge and how derivative of an image helps us to capture the edge related information. [1]

However, edge detectors are very sensitive to noise hence they pick up any changes that are observed through the derivative function. Usually Edge detection consist of 3 steps namely:
Smoothing: where the given input image is smoothed using a filter such as Gaussian kernel so as to remove small noise elements that might be sensed by the derivative function. However, while smoothing we should should be careful of not using 'high' intensity kernels/functions that might eventually remove the edge information from the given image. Differentiation: We take the derivative of the image in this step to extract the edge information from the given smoothed image. Different edge detectors like 1st order derivative or 2nd order derivatives edge detectors can be used for this step. Finalizing the image: The obtained edge information can be passed through functions such image thresholding, image binarizer so as to extract only the useful information/remove noise to some extent and also to represent the image in a better form.

4.1 Question 8: Sobel Filter

Sobel belongs to 1st derivative filters and It finds the local extrema of the gradient, then it makes use of thresholding to filter the image and result of threshold image is considered as edges.



(a) Sobel Filter using given Input Image

(b) Sobel Filter using additional image

Figure 7: Sobel filter and its outputs

We present our results obtained from sobel filter for 2 images.Initially, we start with 2 kernels one

in 'X' direction and the other one in 'Y' direction. Then, we convolve the images using the given kernels using the scipy convolve 2d function. The second image from Fig 7a and 7b denotes the image obtained by convolving from x-direction (Sobel X) and similarly third image from Fig 7a and 7b display the image obtained from Y-direction kernel (Sobel Y). We use the following formula to compute the Magnitude of pixels along the given input images. $\sqrt{(sobelx^2 + sobely^2)}$. where sobelx, sobely denotes the outputs obtained from the previous step. Since the values obtained are not scaled from 0-255 (standard OpenCV values), we convert them using mix-max scaling. Sometimes, when the input image is not in regular shape we make use of padding functions to pad the image with 0s at the borders.

Since gradient also gives the direction of change along with the change, we computer Sobel Direction using np.arctan that is using and display it in Fig 7a and 7b titled as 'Sobel Direction'.

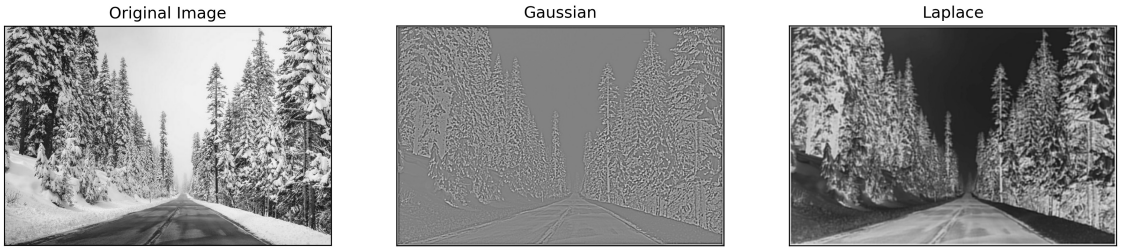
We observe from the outputs of Fig. 7 that Although Sobel uses two kernels in the x and y direction not all the edge information has been captured and it has also picked up noise from the image. In computer vision applications like self driving where edge information is extremely important using traditional computer vision image filters such as Sobel may not give us expected results.

4.2 Question 9: Laplacian of Gaussian filter

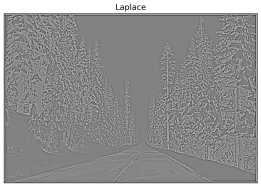
Laplacian filters are second derivative filters that can be used to detect rapid intensity changes in the image. Unlike other filters it varies by the way it operates:
Wherever there is no change (no change of intensity) the laplace is zero, but where there is change of intensity, the value of laplace is positive on the darker side and negative where the region is ligther. That is why we observe uneven surface for the edge detection outputs whenever laplace is applied.

Question 9a and 9b: We generate the results as shown in Fig. 8 Initially, we convert the given image to gray channel for the subsequent steps. Then for the Fig. 8a we make use of Gaussian filter to smoothen the image. This step removes some noise from the amount, which could be possibly have been detected as an edge by the Laplacian filter. We then construct/design a laplace kernel and then use it convolve over the image. The image obtained is then scaled/normalized so as to convert to format of OpenCV.

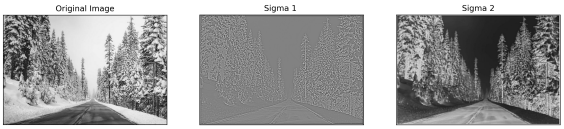
Question 9c: We convolve the image with a gaussian kernel so as to remove the noise from the image, as it eases out the subsequent process of edge detction as the noise could have been pickup up as edges by the Laplacian filter. Gaussian filter does try to eliminate noise to some extent but it may also remove some useful information if used extensively with large kernels and many usage over many iterations.



(a) Sobel Filter using given image using gaussian filter



(b) Sobel Filter using given Input Image



(c) Sobel Filter using given Input Image using varied sigma kernels

Figure 8: Results obtained by Sober Filter for given image

Question 9d: Similarly, for the Fig. 8c we convolve the image with different kernels with varied sigma so as to visualize how change in sigma alters the output. We observe that the change in sigma actually makes the edges look more brighter/evident and reduces the number of false positives. Similarly, in Fig. 9 we get the result by the increasing the sigma till 5.5 and observe that higher sigma doesn't guarantee better edge detection, instead it reduces the precision for true positive edges.

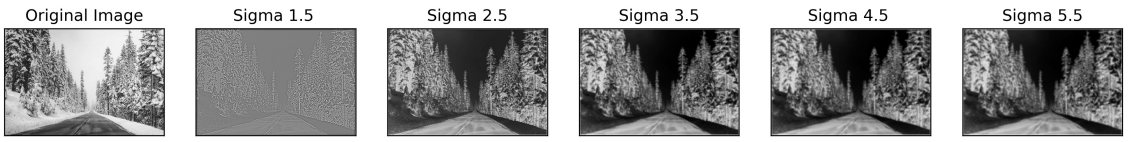


Figure 9: Input image of Road and its corresponding outputs when sigma is increased by 1 every time.

Question 9e: We observe from the results from above filters such as canny, sobel, Laplacian of Gaussian that they alone cannot be used for segmenting the road, Since, they pick up the environment as edge information along with road and also pick up many false positives from the environment, like small leaves, birds etc. Although, one may think of use of using Gaussian Noise to filter the noise since the method we make use of uses values that should be changed while traversing the road, there is perfect hyper-parameter value of kernel size that can be used for the noise removal or even for the case for edge detection there doesn't exist a perfect value of hyper-parameters made use of such as kernel size, standard deviation etc. Hence, we make use of modern methods such as Holistic Edge detection, Domain Adaptation, Transfer learning etc to achieve better road segmentation.

We make use of Holistic Edge detection [24] paper to segment the road from the environment as shown in Fig. 10

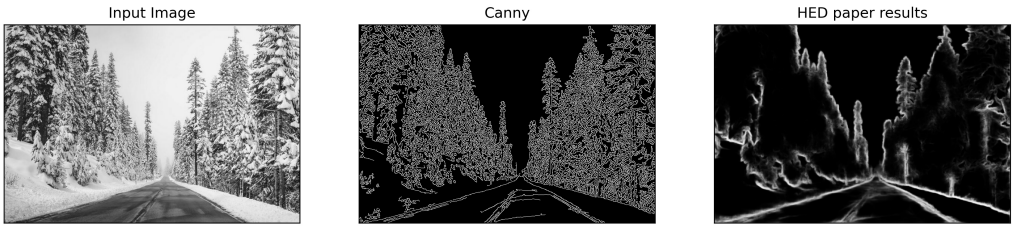


Figure 10: Input image of Road and its corresponding outputs when Traditional Canny edge detection is used and HED Holistically-Nested Edge Detection [24] method is used

We observe from Fig. 10 HED method that it performs better than mentioned edge detection by reducing the number of false positives like the leaves, many trees, snow surrounding, tiny noise from image, without the usage of any noise removal filters. However we still see that there are few false positives that are present like the outer border of the tree etc. These can be removed if we do transfer learning on our road dataset and fine tune using several labelled samples of our road dataset and optimize it under some metric.

In real world, problems like these where we are supposed to segment roads on our dataset (TARGET dataset) from the given image we can make use of Datasets like GTA [16], Synthia [17] (game datasets) (SOURCE datasets) where we already have labelled datasets)in the form of semantic segmentation) of a simulated environment and allow a Deep Neural network attached to a segmentation network to train it in a supervised way. Once we obtain a trained network on the simulated game dataset like GTA[16] we can make use of domain adaptation techniques like Advent [22], AdaptsegNet, Depth Aware Donain Adapation (DADA) [23], etc which propose to project the data points into 'bias-neutral' space using adversarial methods, Translation GAN methods. From Fig.

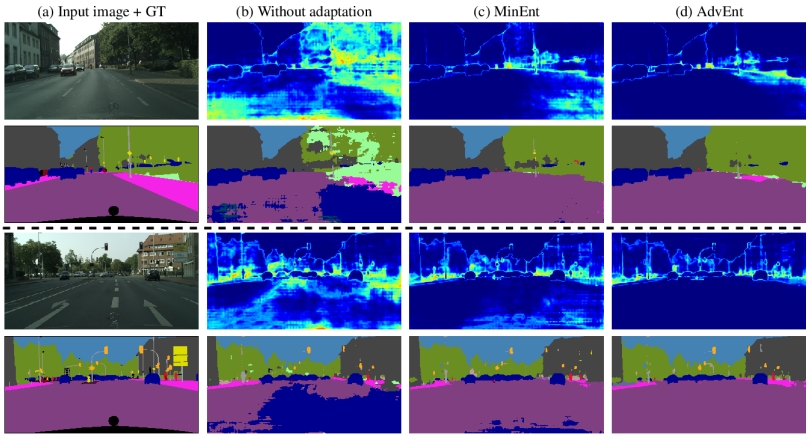


Figure 11: Input image [22] of Road image from Cityscapes [4] dataset and corresponding results

11 Visually we can see that these methods like Advent [22] and other methods are able to segment the road from the environment and also separate other objects such as poles, cycles, humans, pillars, buildings, bikes etc on Cityscapes [4] dataset. Interestingly, Although the source dataset does not contain diverse samples the methods are able to segment the roads on different countries/cities in an unsupervised way, that is they do not make use of any target labels from the target domain.

4.3 Foreground-Background separation

In this task, we aim to separate the foreground from background using a method called Gabor segmentation. The Gabor segmentation relies on Gabor filters, which can be described as a linear filter for image analysis. A Gabor filter directly relates to Gabor wavelets, which serves the basis

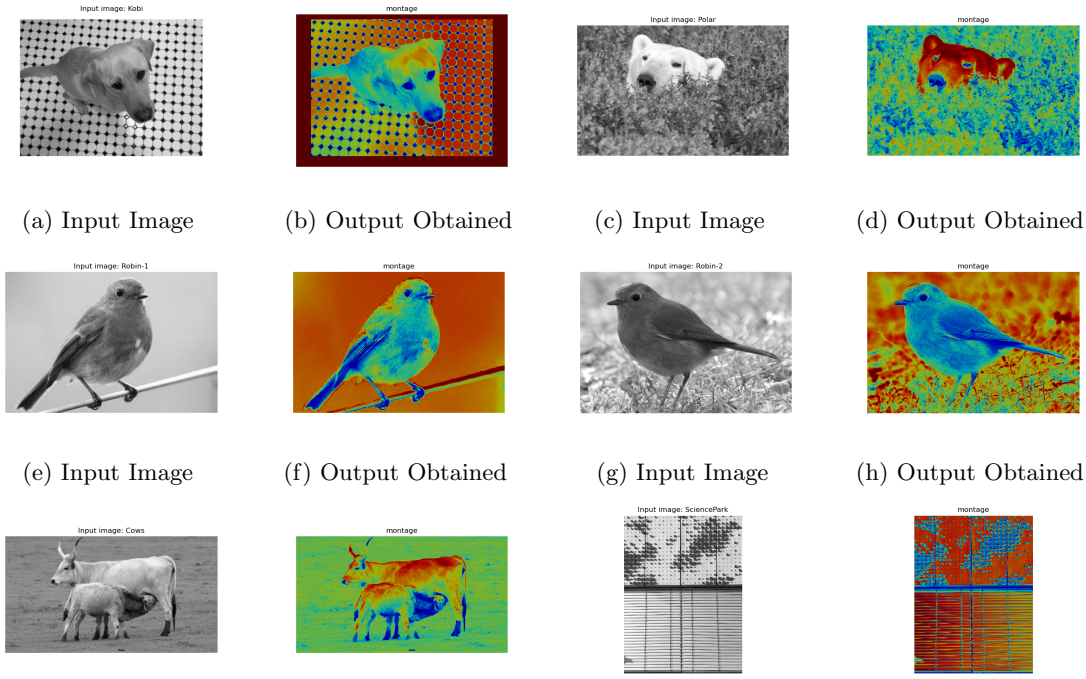
for Fourier transforms in information theory applications. Though, in practice calculating Gabor wavelets are computationally time-consuming, and hence why we opt for a predefined filter bank consisting of various Gabor filters [12]. The parameters for a Gabor filter, essentially controls the shape, form and wavelength of the Gabor filter. More of this can be read in section 2.2. This allows us to apply image segmentation, edge detection, texture analysis, and feature extraction using Gabor filters. In simple terms, when an input image is convolved with the Gabor filters, the patterns are easily highlighted, or essentially it gives the highest response at edges and at points where texture changes or extract features aligned at a particular angle, hence it's useful in image segmentation. The drawbacks of a Gabor filter is its high dimensionality and high redundancy, while having a maximum variance of features, as the feature vectors are very long [20]. Nonetheless, its use ranges wide from face recognition, fingerprint enhancement, and biomedical imaging. Due to its high sensitivity to orientation [13].

Question 10a: When we try to run the algorithm on all the images, we observe that the default values do not work on all the images. The algorithm is able to segment the images by some extent but not completely. There are artifacts that need fixing using custom hyper parameter tuning. The algorithm does provide decent outputs for few images like Robin1 and Polar images. It is because in the Polar image the skin of the bear stands out from the pixel values of its background. In Robin1 images similarly, the background is very different from the object itself due to which the algorithm performs well. Also, both of these images contain noise to a minimal extent that actually interfere with the algorithm.

Question 10b: However, For the Kobi image we observe that the pixel values for the skin of the dog and the background due to which the algorithm doesn't do a good job in differentiating between the skin and the white background. Similarly for the Robin2 image we observe that the original image and background image share similar spectrum of pixel values. In the Science Park image we observe that the image although looks like two different colors but when converted to grayscale the pixel values of two objects are very close to each other due to which the algorithm doesn't perform well through default parameter values.

We experiment with different values of sigma and gamma since theta empirically doesn't give any better outputs. We generate outputs and display in Fig. 12 using our own hyper-paramter values. For Kobi we use lambda values [5,13] and sigma [5 6 7] since the object lies on the centre of the image and the background is very similar to the output we want the algorithm to work in such a way that focuses more towards the central part of the image. Similarly for Robin we use similar values Sigma [5 6] and Lambdas [2,9] because the scene is similar (object centered image). However, when we use similar values for Robin2 although its a object centered image the values do not seem to give good outputs. Due to presence of noise in the background we make use of Gaussian filter here with a kernel of 15 size and remove the noise first and then perform the Gabor segmentation methods. Therefore we use values Sigma [5 6] and Lambdas [2,3,4,5,6]. For the cows image we make use of gaussian filter so as to focus on the object. however although by trying many empirical values the algorithm doesnt perform well possibly because of similarity of pixel intensity values, we use Sigma [1,2,3,4,5,6,7] and Lambdas[2,3,4,5,6]. In the Science park image we observe that the image is not really distinctive from the background/another object in gray scale in terms of visual sigh and also because of similar pixel intensity values. Although we try to empirical values such as Sigma[5,6,7,8] and Lambda higher values [7,8,9,10], the algorithm doesn't perform well.

Question 10c: By turning off the noise elimination filters such as Gaussian filters we observe the segmentation network doesn't perform well because inherently the image contains noise that is picked up in the subsequent process of the gabor segmentation, such as steps where we project using PCA and assign clusters to the image as either 'background', 'foreground'. The cluster assignment does not work well in this case since it has noise elements around the object itself and it assumes the noise as a part of the object/background, As a result of which the boundary between the background and foreground is not clearly separable, it is only separable where there are different clusters with a good separation distance between them.



(a) Input Image (b) Output Obtained (c) Input Image (d) Output Obtained

(e) Input Image (f) Output Obtained (g) Input Image (h) Output Obtained

(i) Input Image (j) Output Obtained (k) Input Image (l) Output Obtained

Figure 12: Given image and the output image obtained from the Gabor segmentation algorithm through hyper-parameter tweaking.

5 Conclusion

From the results of this report, it is apparent that filters are very useful for many computer vision problems. Although in Section 2 we tried to give a intuition about the parameters selection of the filters used, the experiments on Sections 3-5 indicate that applications are sensitive on parameters selection. What is more, these parameters are application specific, which makes the use of filters limited for real-world applications nowadays. We observe through results from Edge detection that simple 1st and 2nd derivative are able to extract edge information from the given input image and how tweaking the parameters involved can vary the outputs and may give better results. However, since the parameters involved are moistly not general in nature and can only be obtained by empirical ways. Therefore, we can make use of modern computer vision methods to segment, track the edges for better performance visually and empirically. In the Foreground-Background segmentation we makes use of gabor filter over given set of images and observe that the default parameters don't always work on every image. So me make use of preprocessing filters such as gaussian noise removal and also adjust the hyper-parameter values such that the objects are now distinct from the background so that the Gabor filter perform better. By Empirical methods we obtains several values that help us achieve the segmentation in the best way possible. For images such as cow, Robin2, Sciencepark where the gabor filter doesnt perform well we investigated them using a pretrained Deeplab v3+ [3] and observe the outputs visually.

References

[1] Gaussian blur - standard deviation, radius and kernel size.

[2] Parameters of gaussian kernel in the context of image convolution.

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In

ECCV, 2018.

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[5] E Roy Davies. *Machine vision: theory, algorithms, practicalities*. Elsevier, 2004.

[6] Itzhak Fogel and Dov Sagi. Gabor filters as texture discriminator. *Biological cybernetics*, 61(2):103–113, 1989.

[7] J-M Geusebroek, Arnold WM Smeulders, and Joost Van De Weijer. Fast anisotropic gauss filtering. *IEEE transactions on image processing*, 12(8):938–943, 2003.

[8] Richard A Haddad, Ali N Akansu, et al. A class of fast gaussian binomial filters for speech and image processing. *IEEE Transactions on Signal Processing*, 39(3):723–727, 1991.

[9] Randall Melville Hanes. The construction of subjective brightness scales from fractionation data: a validation. *Journal of Experimental Psychology*, 39(5):719, 1949.

[10] Jesper Juul Henriksen. 3d surface tracking and approximation using gabor filters, Mar 2007.

[11] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[12] Tai Sing Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on pattern analysis and machine intelligence*, 18(10):959–971, 1996.

[13] Javier R Movellan. Tutorial on gabor filters. *Open source document*, 40:1–23, 2002.

[14] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[15] A. Walker R. Fisher, S. Perkins and E. Wolfart. Gaussian smoothing, 2003.

[16] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[17] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.

[18] Spock. Gaussian derivatives, a difference which makes no difference is not a difference.

[19] Stanley S Stevens and Eugene H Galanter. Ratio scales and category scales for a dozen perceptual continua. *Journal of experimental psychology*, 54(6):377, 1957.

[20] Deepak Verma, Vijaypal Dhaka, and Shubhlakshmi Agrwal. An improved average gabor wavelet filter feature extraction technique for facial expression recognition. *International Journal on Innovations in Engineering and Technology*, 2:1058–2319, 2013.

[21] David Vernon. *Machine vision: Automated visual inspection and robot vision*. Prentice-Hall, Inc., 1991.

[22] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2517–2526, 2019.

[23] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Dada: Depth-aware domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7364–7373, 2019.

[24] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.