JUNE 9, 2024

Bauhaus-Universität Weimar

ASSIGNMENT NO. 4

IMAGE ANALYSIS AND OBJECT RECOGNITION – GROUP 13

KLAURENT MADHI, BHAGATH MAMINDLAPELLY, ABDUL RAHMAN

Table of Contents

Table of Figures:

Task A Description:

The task was to reduce the noise of an image with fourier transformation. In this task, we apply gaussian noise on the image then transform it into frequency domain using fast fourier transformation (fft). We also calculate the gaussian filter mask, adjust its padding, and apply fft. Apply the gaussian mask on noisy image in frequency domain and calculating its inverse using inverse fft. This final step will provide the reduces the noise of the image.

The steps included:

- Reading and converting an input image to grayscale.
- Adding gaussian noise to the image.
- Applying FFT to the noisy image.
- Computing gaussian mask, adjusting its padding, and transforming with fft.
- Applying gaussian mask onto noisy image.
- Parametric study for better sigma to remove the noise.

(a) (b)

Figure 1:Original image with gray scale version. (a) Original Image, and (b) Same image in gray scale

Implementation:

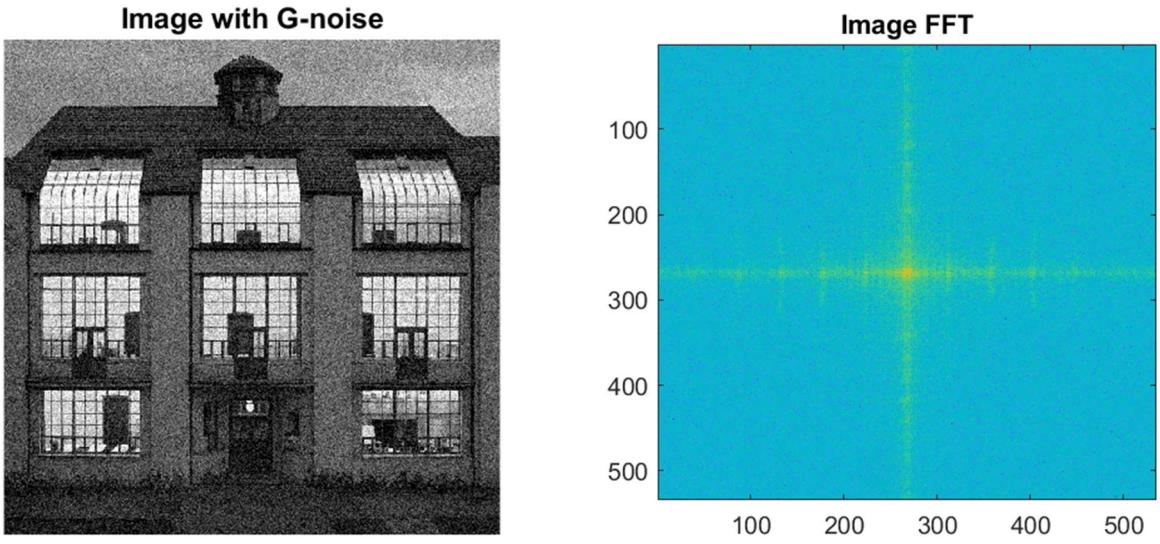
Step a: Reading and converting the image to grayscale

- The input image was read and converted to grayscale with pixel values scaled between 0 and 1. Original and grayscale images are presented in figure.1 (a) and(b) respectively.

Step b: Adding gaussian noise to the image

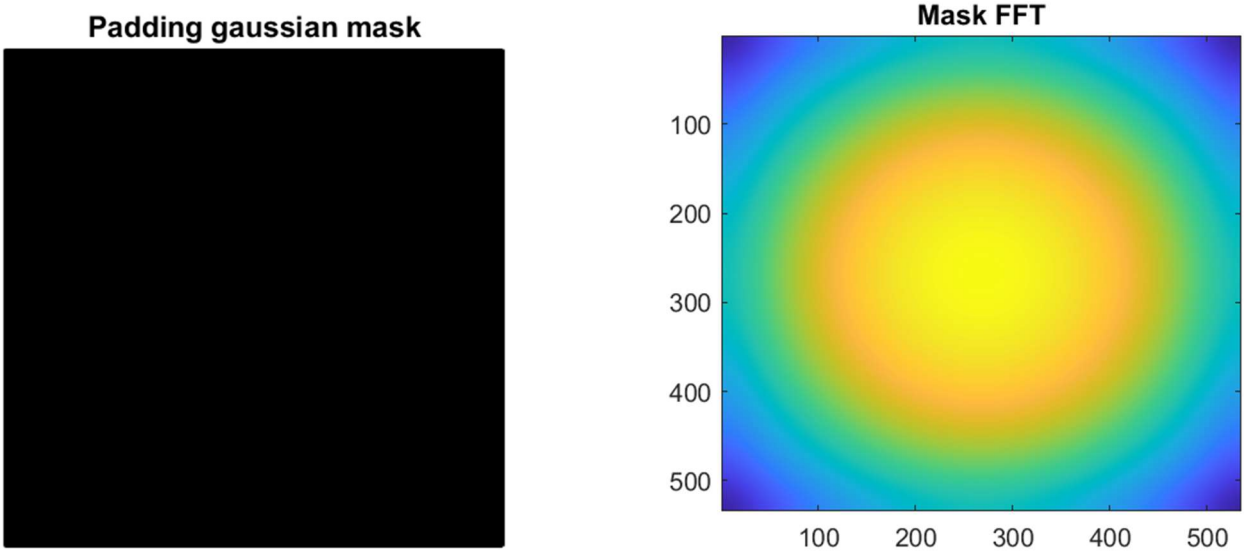
- Used *imnoise* function to apply the gaussian noise onto the image. The parameters used for noise application are M = 0, and V = 0.01.
- The obtained result is presented in figure.2 (a). Step c: Applying FFT to the noisy image

- fft2 is used to apply the Fast Fourier Transformation on the image. The obtained transform matrix is in the same pixel size of the image. The transform matrix values are then scaled on to logarithmic scale and shifted to the middle for visualization.
- *log(abs(fftshift(image_fft)))* These are the functions used to scale and shift the image. The result is presented in figure.2 (b).



(a) (b)

Figure 2: Noisy image and FFT (a) Noisy image (gaussian noise), (b) the logarithmic centred image spectra of the noisy image



a b

Figure 3: Gaussian mask with standard deviation sigma = 1.8. (a) Gaussian mask (b) the logarithmic centred image spectra of the noisy image

Step d: Computing gaussian mask, adjusting its padding, and transforming with fft.

- Now gaussian mask with standard deviation (sigma) is used to transform the image.
- Sigma is multiplied with 3 to adjust the mask radius. After computing gaussian filter mask, its padding is increased according to the input image (see figure.1) dimensions.
- Then *cricshift* function is used to translate the mask to each corner of the image. Each corner of the image is containing quarter portion of the mask.
- The obtained results are transformed using Fast Fourier Transformation. The transformed matrix is scaled to logarithmic scale and shifted to the centre for better visualization.
- The shifter gaussian mask and its fourier transformation are presented in figure.3 (a) and (b) respectively.

Step e: Applying gaussian mask onto noisy image

- The image and gaussian mask frequency domains are multiplied elementwise to reduce the noise on the image. The obtained result in frequency domain with its inverse fourier transformation is presented in figure 4.

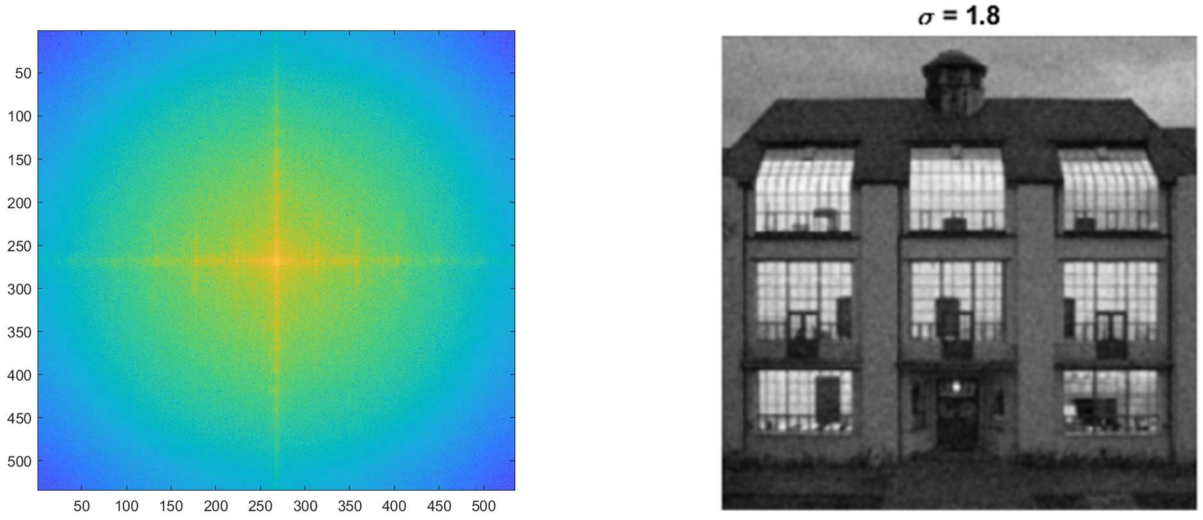


Figure 4: Multiplication of noisy image and mask in frequency domain.
(a) the logarithmic centred image spectra of the noisy image after filtering (sigma = 1.8).
(b) inverse fft of the same.

Step f: Parametric study for better sigma to remove the noise.

- Different sigma values ranging from 0.5 to 1.5 are considered for this parametric study. The obtained results are presented in figure XX. From these results, it is evident that for given problem sigma 1.8 is removing the noise well. It is providing good compromise between noise and blurriness.
- Higher sigma making image more blurred and lower sigma value not reducing the noise.

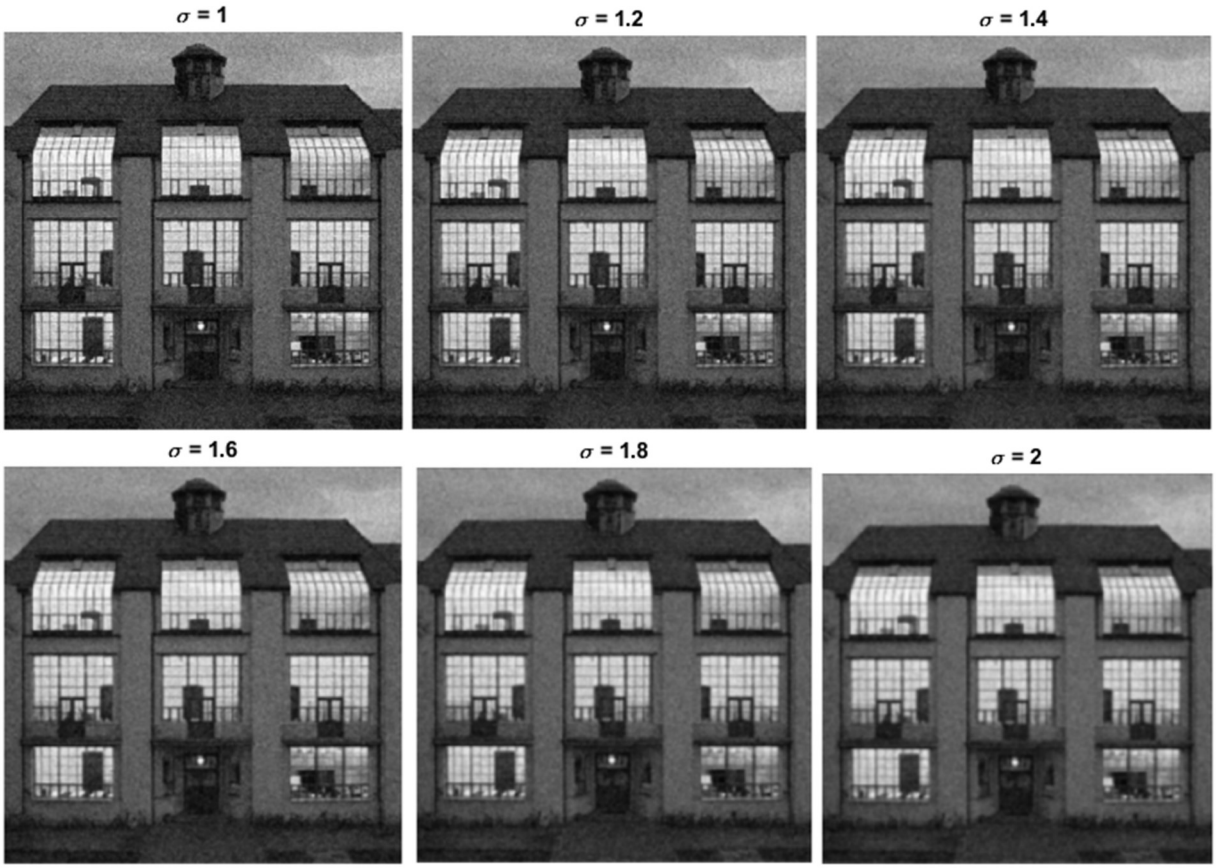


Figure 5 : Paramateric study of images with different simga values from 1 to 2. Lower values not removing noise and higher values are blurring the image.

Results Discussion:

The implemented code for reducing noise in frequency domain yields promising results.

1. Robust Parameter Settings:

- The code's robustness to parameter settings allows for fine-tuning based on image characteristics.

2. Real-World Applicability:

- The successful detection of straight lines showcases practical applicability in robotics, navigation, and industrial automation.

3. Room for Improvement:

- While results are promising, further optimization could enhance performance, especially in challenging environments.
- Even detailed parametric study can be done to find out the best sigma value.

Task B Description:

The task is to find out the shape given in trainB.jpg in other three test images. We need to calculate the boundary of the image and calculate its fourier descriptor of the shape in train image and compare all the shapes in test images to find out if the shape exists in them or not.

Steps included:

- Reading and calculating boundary pixels of the shape in train image.
- Calculating first 24 fourier descriptors and making affine invariant.
- Reading test images and calculating their fourier descriptors.
- Finding train image shape in test images.

Implementation:

Step a: Reading and calculating boundary pixels of the shape in train image.

- The input image first converted into grayscale and then converted into binary image form.
- The binary image is used to calculate the boundary pixels of the shapes in the image.

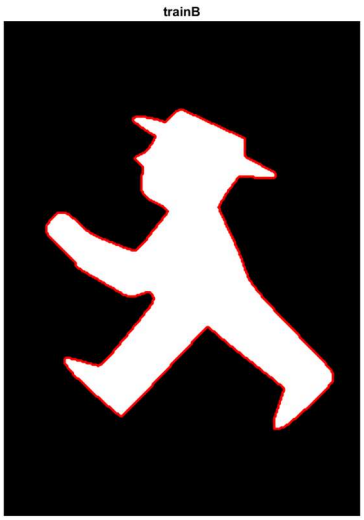


Figure 6: Train image in binary form and with its boundary highlighted.

Step b: Calculating first 24 fourier descriptors and making affine invariant.

- Since the train image has only once shape in it. The pixel information x, y are used to construct its complex number for in $x + iy$ form.
- The complex number list is used to calculate fourier descriptors using *fft* function in MATLAB.
- The first 24 descriptors are considered for comparing the image boundary.
- We neglected first descriptor to make it position invariant

- We divided all descriptor by first to make it scale invariant. Since it contains scale information.
- We only considered absolute values of make the descriptor list invariant of rotation.

Step c: Reading test images and calculating their fourier descriptors.

- All the test images are converted into grayscale. Calculated their fourier descriptors as mentioned in the above step.
- In the test images we calculated the fourier descriptors of all the shapes that are present in the image.
- We only considered the shaped with 25 pixels to ensure we get at least 24 descriptors.

Step d: Finding train image shape in test images.

- We subtracted train shape fourier descriptor array with fourier descriptor array of all the shapes in test images.
- Then the norm of the difference is calculated and checked if the norm is less than 0.09. if yes, we found our shape and the shape is marked with red contour in the test images.
- The obtained results are provided in figure.7, figure.8, and figure.9.

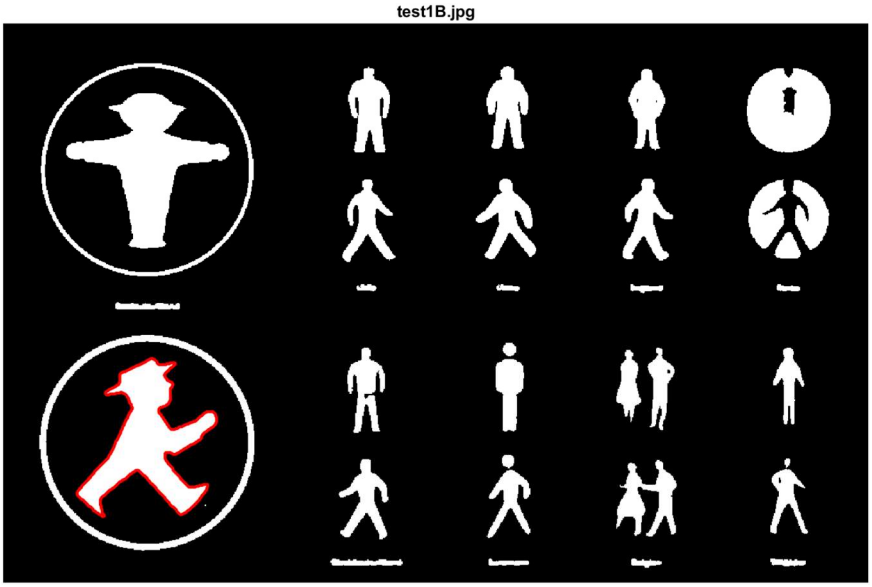


Figure 7: First test image with train shape highlighted.

Results Discussion:

- This code recognizes the train shape fast. This is because of avoiding all the images with less than 25 pixels. Otherwise, it would take 1-2 minutes detect the shape.
- This works well with images with complexity like provided train and test images.
- We can make more robust by considered all the edges to make it generalized implementation.

Figure 8: Second test image with train shape highlighted.

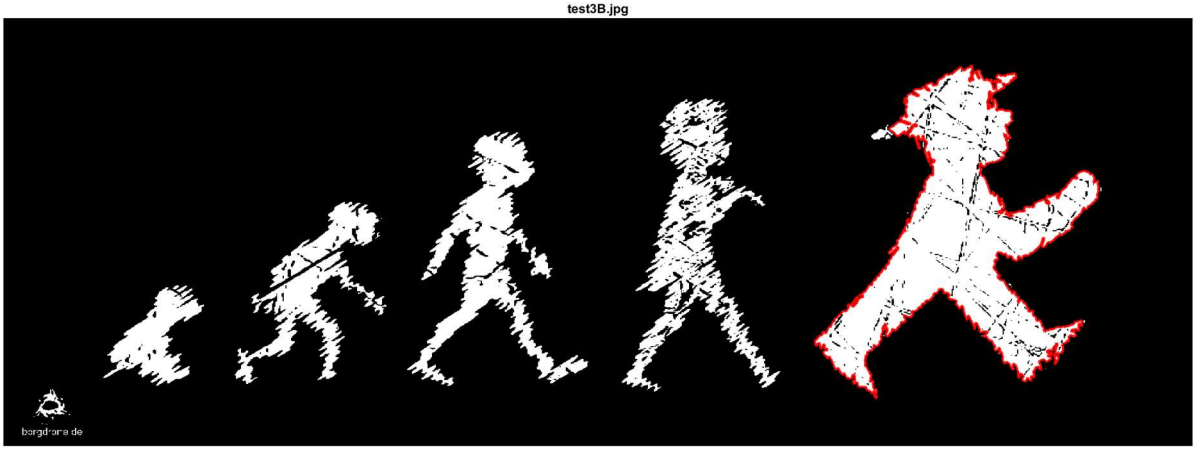


Figure 9: Third test image with train shape highlighted.

References:

1. https://de.mathworks.com/help/ for MATLAB functions.
2. ChatGPT for code debugging.