BME271D - Signals and Systems
Fourier-based Blur Detection for Digital Images

Juseong (Joe) Kim

28 April 2021

Duke University

Instructor: Dr. Michael Rizk

Section: BME271D-01D

I have adhered to the Duke Community Standard in completing this assignment.

Juseong (Joe) Kim

1 Introduction

1.1 Background

The Fourier transform enables the analysis of the frequency content of signals. In image processing applications, the transformation decomposes two-dimensional input into its sine and cosine components, translating information from the spatial to the Fourier, or frequency, domain. Each point in the Fourier representation of an image corresponds to a specific frequency from the spatial domain image. In essence, the higher frequencies form the details while the lower frequencies provide the structure to an image.
In the case of the discrete fourier transform (DFT), only a subset of the frequencies required to fully describe the spatial domain image are retained. The image in the spatial and Fourier domains are equal in size since the number of frequencies corresponds to the number of pixels in the spatial domain range [1]. The fast Fourier transform (FFT) is commonly utilized in lieu of the standard one-dimensional DFT due to its computational savings, reducing the complexity from $O(n^2)$ to $O(n \log n)$.
By manipulating and analyzing the frequency content of images, one can determine whether an image is blurry or not. This process of blur detection has a variety of practical applications, as will be discussed below.

1.2 Applications

The computational classification of images as blurred or not can benefit people in diverse settings. For example, blur detection algorithms can be utilized to automatically grade the quality of images. This may assist photographers of all skill levels in more efficiently sorting through photos and discarding those that are blurry or of poor quality. In videography, blur detection can help to extract the key, important frames from a video, facilitating the process of generating and suggesting thumbnails on video sharing platforms like YouTube. In QR code scanning applications, users can be notified of significant blur and be advised to reposition the camera. In the context of real-time optical character recognition (OCR) from video, blur detection can ensure the expensive OCR computations are only applied to non-blurry frames.

2 Methods

This blur detection algorithm, implemented in Python, is a modification of that provided by PyImageSearch[2] and inspired by the works of R. Liu et al.[3] Required packages include matplotlib, NumPy, OpenCV, imutils, and Pillow. See Appendix A to access the GitHub repo.

2.1 Preprocessing

The input image is converted to grayscale, such that only one channel is required to fully represent the image. Each pixel is specified by a single intensity value, as opposed to three intensities corresponding to red, green, and blue components for a full color image [4]. The grayscale intensities are sufficient to perform blur detection and significantly reduces the complexity of image processing.

2.2 Algorithm

Each step of the blur detection algorithm is summarized below.

1. Find dimensions of the input image and compute center (x,y)-coordinates
2. Perform DFT of image using FFT algorithm from NumPy
3. Shift the zero-frequency (DC) component to the center
4. Zero-out the central region of specified shape and size in the shifted FFT
5. Apply inverse FFT shift to reposition the DC component back to the corners of the two-dimensional magnitude spectrum
6. Perform inverse FFT
7. Compute magnitude spectrum of reconstructed image with DC values filtered out
8. Calculate the mean of the magnitudes of the reconstructed image
9. Compare the mean to a user-defined threshold value
10. Output a binary decision of whether or not blur was detected in the input image.

As discussed above, step 2 transforms the image from the spatial to the frequency domain. The FFT shift in step 3 facilitates manipulation of the frequency content of an image by relocating the lower frequency components from the corners to the center of the spectrum. This magnitude spectrum is primarily used in image processing, but the phase spectrum must also be preserved to re-transform the Fourier image, via the inverse FFT, back into the correct spatial domain [1], as in step 6. The removal of low frequencies in step 4 involves the use of an ideal high-pass filter [5] to zero a portion of the center of the shifted FFT. In computing the magnitude spectrum of the FFT for visualisation after filtering (step 4) and of the reconstructed image (step 7), the linear scale is transformed to a logarithmic scale since the dynamic range of Fourier coefficients is larger than can be displayed on the screen [6]. This is accomplished by the *log* function from NumPy. The *abs* function and the *where* parameter of *log* are used to avoid calculating the logarithm of negative values and zeroes.

2.3 Post-processing

In order to effectively display the result of the blur detection in color, two channels were added to the grayscale image. The mean of the magnitudes and indication of blur were overlaid on the image—in green for images with no blur and red for those with blur.

2.2 Data

All images used in the analysis of this algorithm were shot on the OnePlus 5T.

2.3 Determining the threshold

To facilitate the process of determining the threshold value for blur classification, the Gaussian kernel, which acts as a low-pass filter, was used to artificially introduce increasing degrees of blur into the images [7]. The GaussianBlur method from OpenCV was utilized with radii of odd numbers increasing from 1 to 15. The FFT-based blur detection algorithm was then executed on each image.

It is important to note that the threshold value for classifying blur varies with the use case of the detection algorithm. For natural scene images, the amount of blur may not necessarily correlate with lower quality images. For example, photographers may

intentionally place backgrounds out of focus, an optical effect termed the bokeh effect, for aesthetic appeal. In addition, in the application of OCR on document images, a higher sensitivity to blur is required, since the accuracy of OCR decreases significantly with increasing degrees of blur. This necessitates an increased threshold value such that the algorithm tolerates less blur and identifies higher quality images.

3 Results and Discussion

3.1 Effect of blur on mean magnitude of reconstructed image

The result of the blur detection algorithm for varying levels of blur in an image is shown below in Figure 1. The threshold is set at 1.0 and a circular high-pass filter of radius 40 pixels is used. As the extent of blur increases, the mean magnitude decreases.

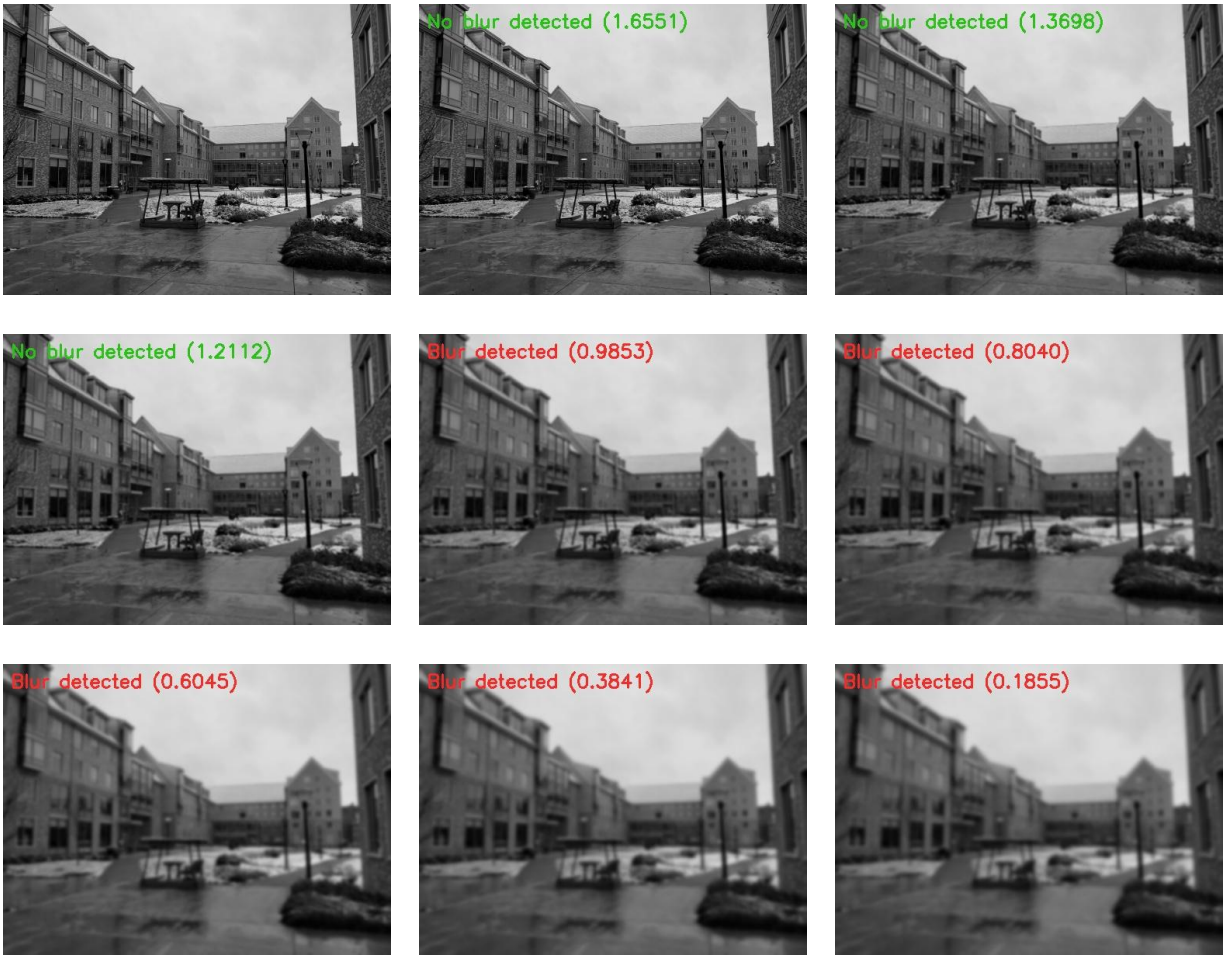

Figure 1 Images blurred with odd Gaussian kernel sizes 1 through 15 *(left to right, top to bottom)*. Grayscale original image shown in top-left.

The blurring of images can be described as the application of a low-pass filter, which attenuates higher frequencies. The loss of high frequency content decreases the mean of magnitudes in the reconstructed image, especially for sharp images that are composed of more high frequency components. Since the amount of blur and mean magnitude are

indirectly related, mean values above and below a threshold value appropriately correspond to non-blurry and blurry images, respectively.

3.2 Effect of filter shape

The shape of the ideal high-pass filter used in step 4 of the algorithm was varied to assess its effect on the image characteristics and detection performance. Figure 2 shows the original image before filtering, the non-shifted magnitude spectrum and reconstructed image after filtering for square and circular filters with side length 20 pixels and radius 10 pixels, respectively.

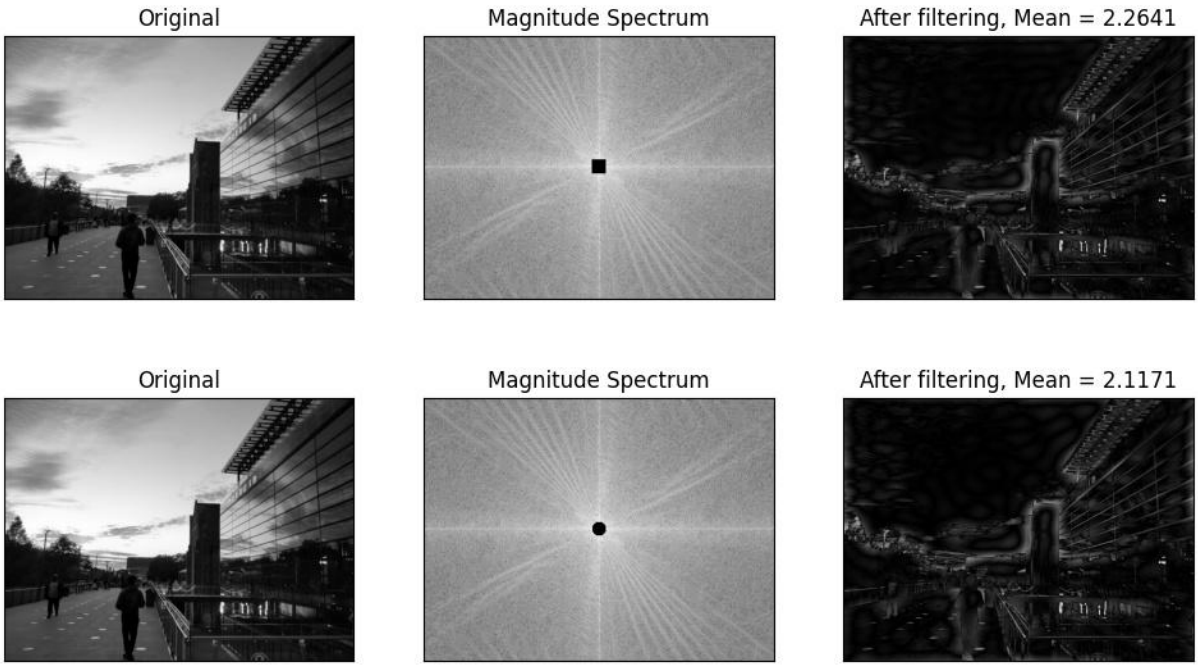

Figure 2 Effect of high-pass filter shape *(top: square, bottom: circle)* on image.

The mean magnitude of the square-filtered image was 2.2641 while that of the circle-filtered image was 2.1171. For various blurry and non-blurry input images, the mean magnitude for the circle-filtered image was observed to be less than that of the square-filtered image. Note that for the given sizes, the square covers more area than the circle, meaning more low frequencies are zeroed by the square. The effect of such a negligible difference in area is difficult to observe in the reconstructed image. Thus, the higher mean magnitude for the square-filtered image may be explained by the shape of the cut off of the filter. The linear nature of the square filter cut off regions may potentially remove frequencies of lower intensity, or, conversely, preserve frequencies of higher intensity. Due to the abrupt frequency cut-off of the ideal high-pass filter, some noise and ringing can be observed in both of the reconstructed images (Figure 2). This distortion can be reduced by using the Butterworth or Gaussian filters, which smoothly block information within a certain radius from the origin point of the kernel[5].

3.3 Effect of filter size

For increasing circle radii of the ideal high-pass filter, the blur detection algorithm was executed on non-blurry and blurry input. The mean magnitude of the reconstructed image was recorded. As the filter size increased, the mean decreased. With greater amounts of low frequency components being removed, the image loses more of its frequency content, and it follows that the mean magnitude in the reconstructed image must decrease.

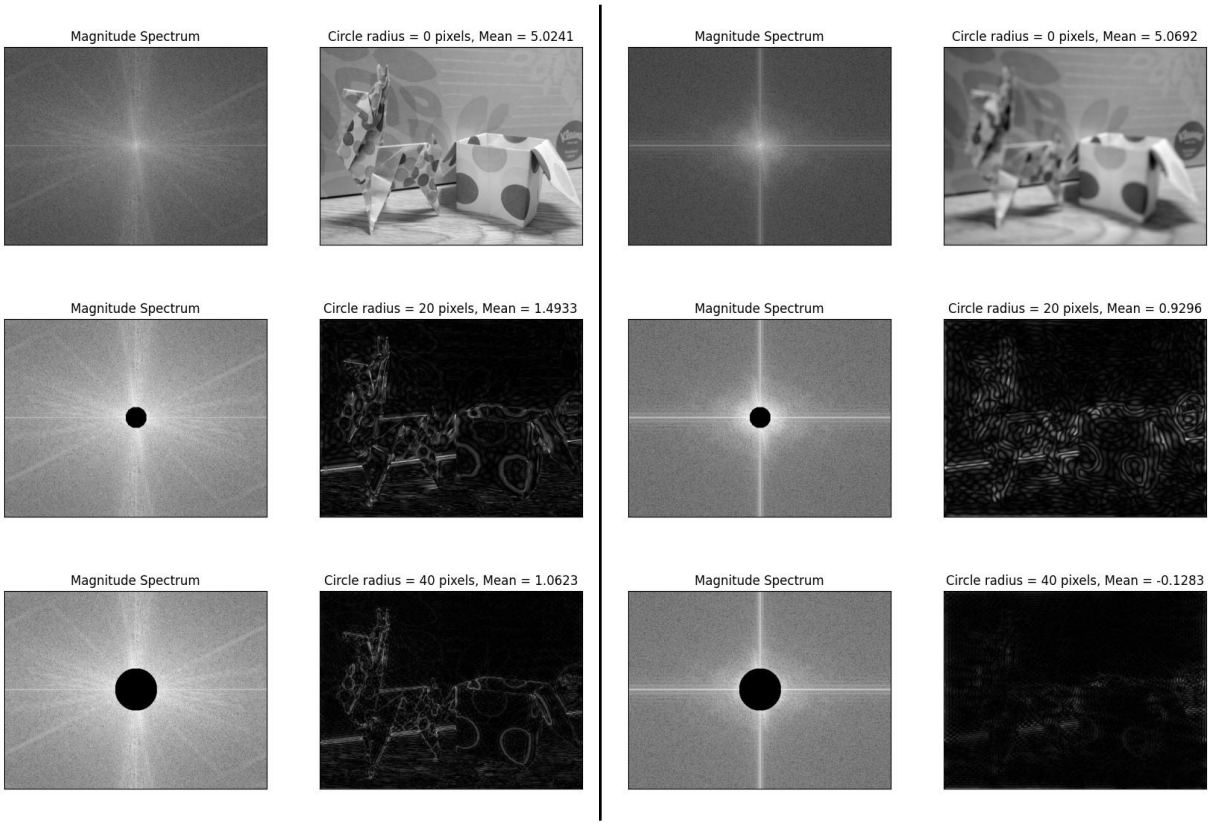

Figure 3 Effect of high-pass filter size on image (*left: non-blurry, right: blurry)*.

The differing effect of filter size on input images without and with blur can be observed in Figure 3. At a circular filter radius of 20 pixels, the edges and outlines of the sharp image (*left*) are better retained than those of the blurry image (*right*). When the filter radius is increased to 40 pixels, the fine details of the sharp image are preserved while the blurry image exhibits minimal retention of its original information. This indicates that much of the information in the blurry image is contained in the lower frequencies, while the sharp image has more information in the higher frequencies. This observation forms the basis of this FFT-based blur detection algorithm. After high-pass filtering of any size, the mean magnitude of sharper images is greater than that of blurry images, since there is more content in the higher frequencies of non-blurry images. Therefore, a lower mean corresponds to a blurry image.

Most importantly, for the algorithm to correctly identify images as blurry or not blurry, an appropriate threshold value must be set manually. This threshold value is dependent on

the filter size and the nature of the image. For a filter radius of 20 pixels, a threshold value of 1.2 would correctly identify the left photo as non-blurry and the right as blurry (Figure 3). However, for a filter radius of 40 pixels, the same threshold value would result in the incorrect classification of both images as blurry. Given a circular high-pass filter of radius 40 pixels, a threshold value of 1.2 appears to suffice for long shots like in Figure 1, but fail for close-up shots like that in Figure 3. Therefore the threshold must be appropriately adjusted for varying filter sizes and shot types.

3.4 Limitations

The aforementioned need to manually determine the threshold mean value of magnitudes in the filtered image is less than ideal. Though the introduction of increasing levels of Gaussian blur (see section 2.3) can assist the user with this process, due to the large variance in the optimal threshold value for images in different contexts, redefining a threshold for each use case is inefficient and time-consuming. The incorporation of machine learning may allow for the threshold value to be set automatically. Additionally, replacing the ideal high-pass filters with the Gaussian filter, for example, can reduce the effect of waves and ringing in the filtered image. This may improve the consistency and accuracy of mean magnitude calculations.

Another limitation of this blur detection algorithm is that classification is strictly binary: blur is detected or not. Since the quality of an image is not solely defined by the amount of blur, defining a quantitative measure of image quality such as the cumulative probability of blur detection (CPBD), Just Noticeable Blur (JNB), or the frequency domain image blur measure (FM) proposed by De et al. (2013) may be beneficial for image quality assessment applications[8]. Information from the exposure histograms of photos can also aid in the evaluation of image quality.

4 Conclusion

This blur detection algorithm was implemented by utilizing the Fourier transform of two-dimensional input images. After removing lower frequency components that largely correspond to blur, calculation of the mean magnitude of the reconstructed image provides information about the relative frequency composition of the original image. Higher mean magnitudes indicate that the original image was made up of higher frequency components overall, meaning the edges and details of the image were more prominent than the smoother, blur-like regions. Thus, comparing the mean magnitude to a user-defined threshold value allows for the binary classification of an image as blurry or non-blurry. In the case of ideal high-pass filters, a square shape resulted in higher mean magnitudes as compared to a circular shape, while both shapes introduced distortion. As the size of the high-pass filter was increased, the mean magnitude value of the filtered image was observed to decrease. Due to this variation of the mean value, which depends on numerous factors including filter size and image type, setting an optimal threshold value that works for any image is difficult. Thus, this algorithm can be further improved by incorporating machine learning methods, kernel filters, and alternative blur evaluation metrics.

References

[1] Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2003a). Fourier Transform. Hypermedia Image Processing Reference. https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm

[2] Rosebrock, A. (2020, June 15). OpenCV Fast Fourier Transform (FFT) for blur detection in images and video streams. PyImageSearch. https://www.pyimagesearch.com/2020/06/15/opencv-fast-fourier-transform-fft-for-blur-detection-in-images-and-video-streams/

[3] Renting Liu, Zhaorong Li and Jiaya Jia, "Image partial blur detection and classification," 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1-8, doi: 10.1109/CVPR.2008.4587465.

[4] Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2003). Grayscale Images. Hypermedia Image Processing Reference. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm

[5] Chen, C. (2020, February 16). Digital Image Processing using Fourier Transform in Python. Medium. https://hicraigchen.medium.com/digital-image-processing-using-fourier-transform-in-python-bcb49424fd82

[6] OpenCV. (2021). Discrete Fourier Transform. https://docs.opencv.org/3.4/d8/d01/tutorial_discrete_fourier_transform.html

[7] Bobick, A. (2014). Frequency and Fourier Transforms [Slides]. Georgia Tech College of Computing. https://www.cc.gatech.edu/~afb/classes/CS4495-Fall2014/slides/CS4495-Frequency.pdf

[8] De, K., & Masilamani, V. (2013). Image Sharpness Measure for Blurred Images in Frequency Domain. Procedia Engineering, 64, 149–158. https://doi.org/10.1016/j.proeng.2013.09.086

Appendix

Appendix A Source code

The repository containing the Python source code for this blur detection algorithm can be accessed via this link: https://github.com/juseong-kim/blur_detection.