Blurred Image Detection using FFT

Nisha Kumari Barsha (2204101007)
Achint Kumar Kansal (2204101009)
Amit Raj (2204101010)

December 2022

Contents

1 Introduction

Blur detection is detecting whether an image is blurry or not.

1.1 Application of Blur detection

- Automatic image quality grading.
- Helping professional photographers sort through 100s to 1000s of photos during a photo shoot by automatically discarding the blurry/low-quality ones.
- Applying OCR to real-time video streams but only applying the expensive OCR computation to non-blurry frames.

Such a blur detection procedure could either automatically discard the poor-quality images or tell the end user, "Hey bud, try again. Let's capture a better image here."

1.2 Fast Fourier Transform

The Fast Fourier Transform(FFT) is a convenient mathematical algorithm for computing the Discrete Fourier Transform. It converts a signal from the spatial domain into the Fourier/frequency domain.
The FFT represents the image in both real and imaginary components.
By analyzing these values, we can perform image processing routines such as blurring, edge detection, thresholding, texture analysis, and blur detection.

2 Methodology

Implement our Fast Fourier Transform blur detector with OpenCV.

The method we'll cover is based on the following implementation from *Liu et al.'s 2008 CVPR publication, Image Partial Blur Detection and Classification.*

Our blur detector implementation requires both matplotlib and NumPy. We'll use a Fast Fourier Transform algorithm built-in to NumPy as the basis for our methodology.
In the code, we have implemented $detect_blur_fft()$ function, accepting four parameters:

- image: Our input image for blur detection
- size: The size of the radius around the centerpoint of the image for which we will zero out the FFT shift
- thresh: A value which the mean value of the magnitudes (more on that later) will be compared to for determining whether an image is considered blurry or not blurry.
- vis: A boolean indicating whether to visualize/plot the original input image and magnitude image using matplotlib.

First, we calculate the center location of points in the image. After that, we apply the FFT2 transform, then we compute the magnitude. After that, zero out the center of the FFT shift (i.e., remove low frequencies), apply the inverse shift such that the DC component once again becomes the top-left, and then apply the inverse FFT.

Compute the magnitude spectrum of the reconstructed image, then compute the mean of the magnitude values.

The image will be considered **blurry** if the mean value of the magnitudes is less than the threshold value.

3 Dataset Used

We have used the dataset from Kaggle.

Blur Dataset: www.kaggle.com/datasets/kwentar/blur dataset

4 Results


Test Image
Blurry (18.1882)


Blurry (7.7451)


Test Image
Blurry (2.6919)

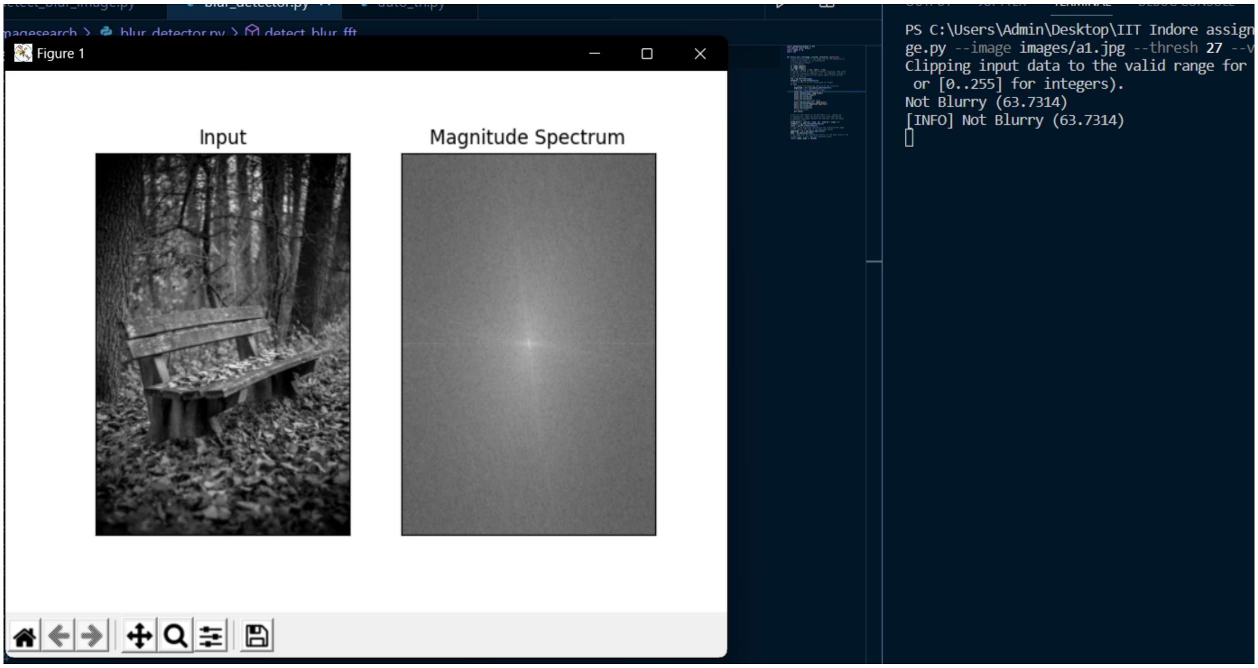
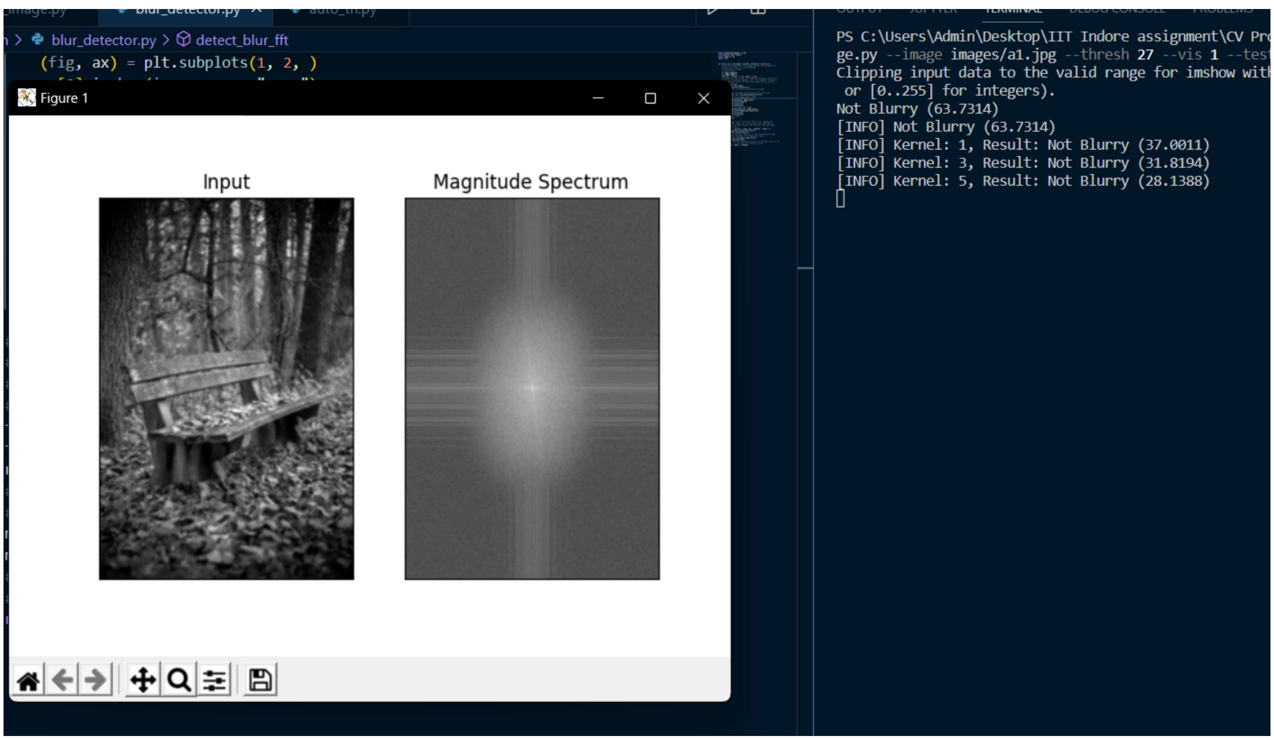
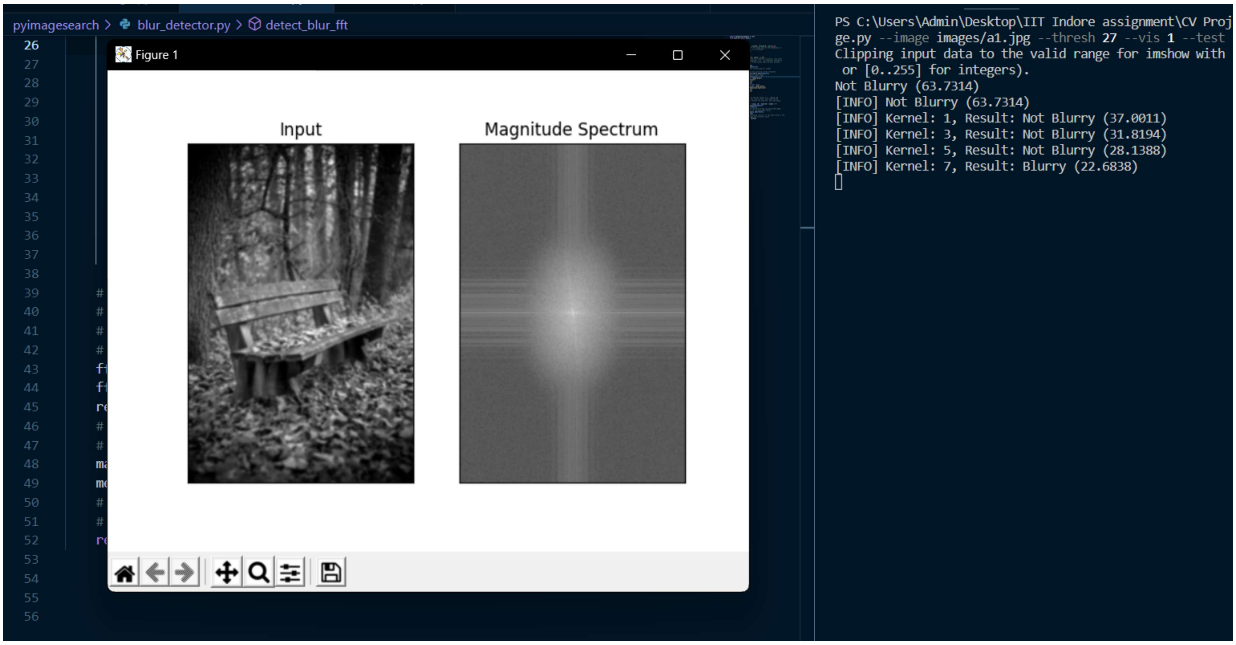


Figure 1
Input
Magnitude Spectrum
PS C:\Users\Admin\Desktop\IIT Indore assign
ge.py --image images/a1.jpg --thresh 27 --v
Clipping input data to the valid range for
or [0..255] for integers).
Not Blurry (63.7314)
[INFO] Not Blurry (63.7314)


blur_detector.py > detect_blur_fft
(fig, ax) = plt.subplots(1, 2,)
Figure 1
Input
Magnitude Spectrum
PS C:\Users\Admin\Desktop\IIT Indore assignment\CV Pr
ge.py --image images/a1.jpg --thresh 27 --vis 1 --tes
Clipping input data to the valid range for imshow wit
or [0..255] for integers).
Not Blurry (63.7314)
[INFO] Not Blurry (63.7314)
[INFO] Kernel: 1, Result: Not Blurry (37.0011)
[INFO] Kernel: 3, Result: Not Blurry (31.8194)
[INFO] Kernel: 5, Result: Not Blurry (28.1388)



5 Conclusion

We can detect blurred images with the help of a threshold set for an image, and we also introduced in the image gaussian blurriness with the help of loops to check for blurriness detection.

6 Future Work

This project can also be extended for video blur detection i.e., using Fast Fourier Transform to detect video blur detection.