Member-only story

Fourier Transform for Image Processing in Python from scratch.

Raoof Naushad · Follow
Published in DataDrivenInvestor
8 min read · Oct 23, 2020

> *In this blog we are also implementing DFT , FFT and IFFT from scratch.*

This blog is previously published on

https://www.datadriveninvestor.com/2020/10/23/fourier-transform-for-image-processing-in-python-from-scratch/

First of all it is really interesting to work with mathematical problems. Right? I know the answer can be yes and no. Still applying maths on real world problems for optimisations, modelling will be really good. I started to do this recently and see how things works for me. So this blog is a part of my learning and it is to understand how computational complexity for convolution can be reduced using Fourier Transform techniques. I will try to go in detail.

Check out this repo for building Discrete Fourier Transform, Fourier Transform, Inverse Fast Fourier

Transform and Fast Fourier Transform from scratch with Python.

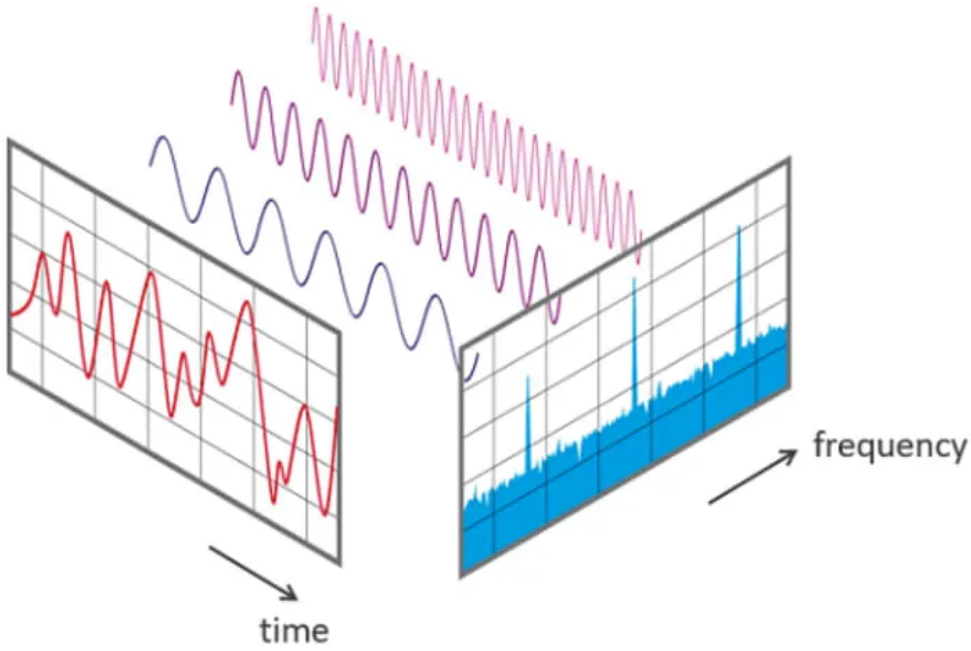
raoofnaushad/Fourier-Transformation-for-Image-Processing

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or...

github.com

Let's see what Wikipedia has to say about Fourier Transform.

> *Source-Wikipedia*
>
> *In mathematics, a* ***Fourier transform*** *(**FT**) is a mathematical transform that decomposes a function (often a function of time, or a signal) into its constituent frequencies, such as the expression of a musical chord in terms of the volumes and frequencies of its constituent notes. The term* Fourier transform *refers to both the frequency domain representation and the mathematical operation that associates the frequency domain representation to a function of time.*



Fourier Transform — Source

Is it confusing? If yes lets see how it helps us.

The algorithm helps in such a way that it allows us to split the input signal that is spread in time (Like in the image above) into the number of frequencies of length, amplitude and phase so that all these frequencies together can reform the original signal.

So it actually converts the data information of time domain into domain of frequencies and also backwards.

Let's work with an analogy. We usually make tea right. Ingredients of tea are milk, tea powder, sugar and water. Here fourier transform helps us to split out the ingredients to 4 different bottles with each in each one. This is what FT does — it splits the whole input into its ingredients.

$$X_k = \sum_{n=0}^{N-1} x_n \mathrm{e}^{-i2\pi kn/N} \quad k = 0, \ldots, N-1$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \mathrm{e}^{i2\pi kn/N} \quad n = 0, \ldots, N-1$$

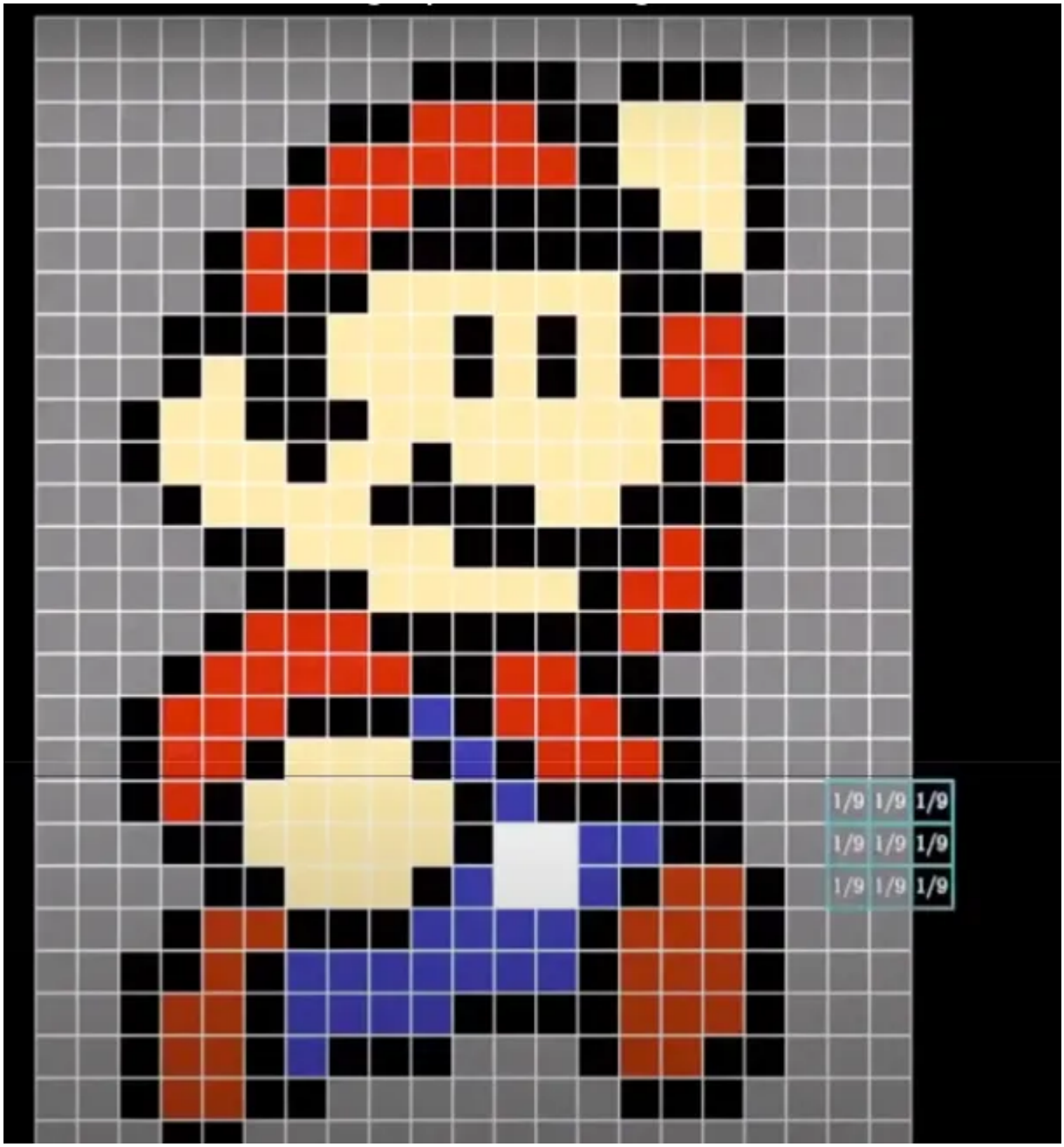
Fourier Transform equation — Source

let's start working on it.

Our main aim is to use fourier transform to reduce the computational complexity for convolution. There are lots and lots of applications in computing, physics, sound mixing etc..

You know how convolution works, we take a filter(kernel) and we will be having an image so what happens is we hover the filter above the image pixels and multiply and sum them up. After that the filter move to the next set of pixels and multiply then sum.

Woah! There will be lots of multiplication and computational complexity.



Convolving through an image — Source

How can we solve this using Fourier Transform?

There is a nice and awesome property of Fourier transform related to convolution. This mentions that convolution of two signals is equal to the multiplication of their Fourier transforms. Yeah! So instead of multiplying throughout the image with the kernel we could take the Fourier transform of it and just get a bit wise multiplication. This can even be applied in convolutional neural networks also. Before the convolutional layer transform the input and kernel to frequency domain then multiply

then convert back. Even though it deals with transforming and reverse transforming still it is computationally less expensive.

Convolution:

$$x[n] * y[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \overset{DTFT}{\leftrightarrow} X(e^{j\omega})Y(e^{j\omega})$$

Property — Source

Before moving onto the implementation and application let us first see and understand what is the difference of **Fourier Transform** and **Fast Fourier Transform** and also why we prefer to do Fast Fourier Transform?.

This is the equation of Fourier Transform.

$$x[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}$$

DFT Equation — Source

In `Fourier Transform` we multiply each of the signal value `[n]` with `e` raised to some function of `n`. So here comes N (multiplications) x N (additions) thus the `computational complexity` in Big-O notation is `O(N²)`

`Fast Fourier transform` is a method to find Fourier transform in a way that `minimise this complexity` by a strategy called `divide and conquer` because of this the computation complexity will be reduced to `O(NlogN)`. Check the below table how it helps us in finding fourier transform.

N	1000	10^6	10^9
N^2	10^6	10^{12}	10^{18}
$Nlog_2N$	10^4	$20\,x\,10^6$	$30\,x\,10^9$

Computation Complexity of FFT & FT — Source

So lets consider a case where it take `1 nano second for a operation`. It would take `Fast Fourier Transform` to complete it in `30s`. However regular algorithm will need more than `10¹⁸ => 31.2 years.`

Introduction to Time Series Forecasting of Stock Prices with Python | Data Driven Investor

In this simple tutorial, we will have a look at applying a time series model to stock prices. More specifically, a...

www.datadriveninvestor.com

If you want to read more about the Fast Fourier Transform computation complexity and simple implementations, check out the below link.

Fast Fourier Transform

How to implement the Fast Fourier Transform algorithm in Python from scratch.

towardsdatascience.com

Applying Fourier Transform in Image Processing.

We will be following these steps.

1) Fast Fourier Transform to transform image to frequency domain.

2) Moving the origin to centre for better visualisation and understanding.

3) Apply filters to filter out frequencies.

4) Reversing the operation did in step 2

5) Inverse transform using Inverse Fast Fourier Transformation to get image back from the frequency domain.

Some Analysis

- Digital images are not continuous so we use DFT instead of Fourier transform.

Time Duration		
Finite	Infinite	
Discrete FT (DFT)	Discrete Time FT (DTFT)	discr.
$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\omega_k n}$	$X(\omega) = \sum_{n=-\infty}^{+\infty} x(n)e^{-j\omega n}$	time
$k = 0, 1, \ldots, N-1$	$\omega \in [-\pi, +\pi)$	n
Fourier Series (FS)	Fourier Transform (FT)	cont.
$X(k) = \frac{1}{P}\int_0^P x(t)e^{-j\omega_k t}dt$	$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt$	time
$k = -\infty, \ldots, +\infty$	$\omega \in (-\infty, +\infty)$	t
discrete freq. k	continuous freq. ω	

Discrete and Continuous Fourier Transformation — Source

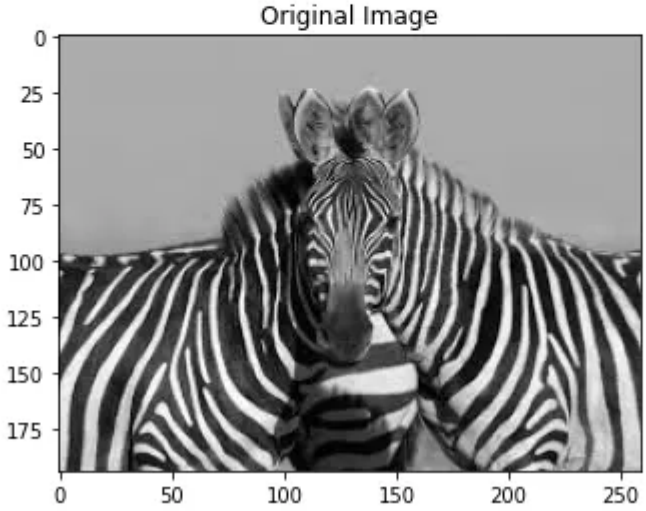
- Ordinary DFT is slow so we chose FFT. (reason mentioned above)

Let's try converting the image into frequency domain and get it back to its original form.

1. First we have to read the image. Here we are using CV package to read the image. Now the image is loaded in grey scale format.

```
In [4]: img_c1 = cv2.imread("zebra.jpeg", 0)
        plt.imshow(img_c1, "gray"), plt.title("Original Image")

Out[4]: (<matplotlib.image.AxesImage at 0x7f59415c2e10>,
         Text(0.5, 1.0, 'Original Image'))
```

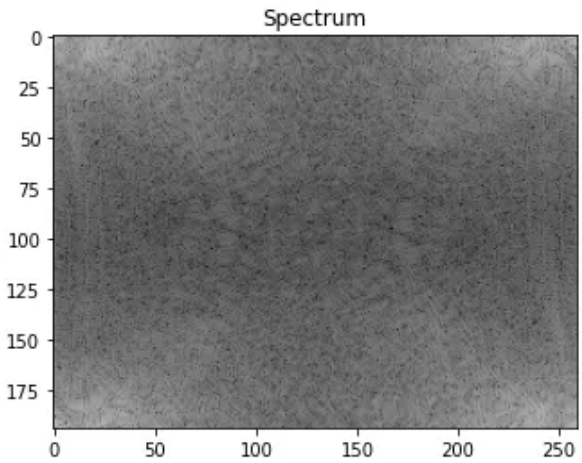


Generated By Author

2. Now we are going to apply FFT from numpy package. This means that we are taking the discrete image pixel values and are transforming it to a frequency domain and visualising it. The whole information is reserved but transformed to another domain.

```
In [5]: img_c2 = np.fft.fft2(img_c1)
        plt.imshow(np.log(1+np.abs(img_c2)), "gray"), plt.title("Spectrum")

Out[5]: (<matplotlib.image.AxesImage at 0x7f5940344630>, Text(0.5, 1.0, 'Spectrum'))
```



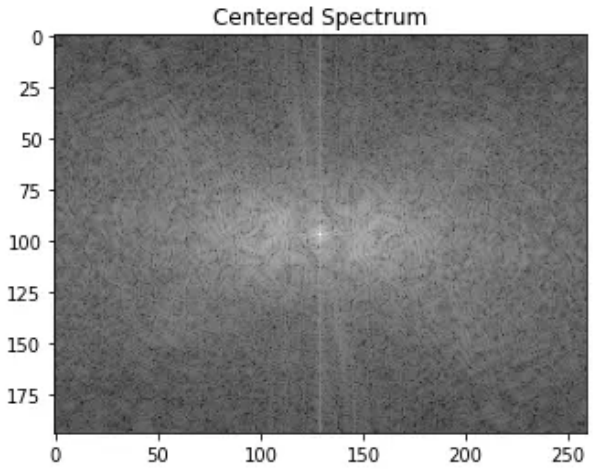
Generated By Author.

Here you can see that there are white spots over corners which represents the low frequency components.

3. In this step we take the origin from corner to centre. This results in such a way that the centre part contains the low frequency components where as other contains high frequency.

```
In [6]: img_c3 = np.fft.fftshift(img_c2)
        plt.imshow(np.log(1+np.abs(img_c3)), "gray"), plt.title("Centered Spectrum")

Out[6]: (<matplotlib.image.AxesImage at 0x7f59403249b0>,
         Text(0.5, 1.0, 'Centered Spectrum'))
```

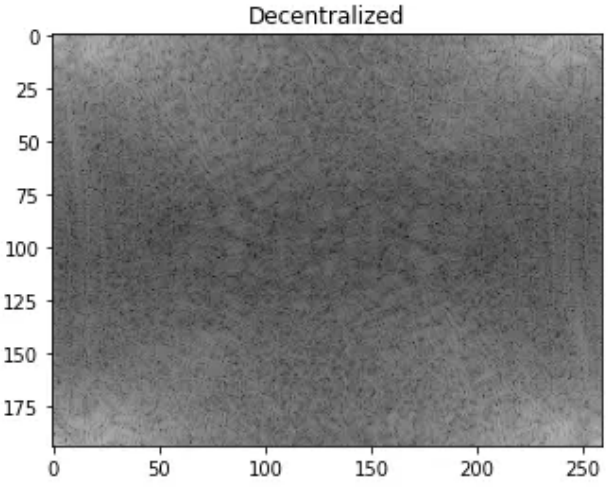


Generated By Author.

4. As the next step we are going to decentralise the origin back to where it belong to so that we can convert the image in frequency domain back to what we had.

```
In [8]: img_c4 = np.fft.ifftshift(img_c3)
        plt.imshow(np.log(1+np.abs(img_c4)), "gray"), plt.title("Decentralized")

Out[8]: (<matplotlib.image.AxesImage at 0x7f59402cd630>,
         Text(0.5, 1.0, 'Decentralized'))
```

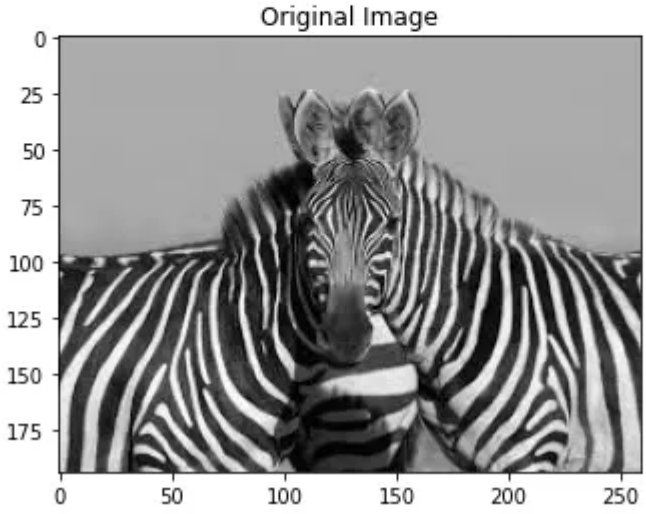


Generated By Author.

5. Final step, here we reverse back the image from the frequency domain by using inverse fourier transformation.

```
In [4]: img_c1 = cv2.imread("zebra.jpeg", 0)
        plt.imshow(img_c1, "gray"), plt.title("Original Image")

Out[4]: (<matplotlib.image.AxesImage at 0x7f59415c2e10>,
         Text(0.5, 1.0, 'Original Image'))
```



Generated By Author.

Image Processing.

Low frequencies in images are pixel values that are changing slowly that means the smooth areas with slightly color changing. Similarly high frequencies are pixels whose values are changing fast. This include edges with rapid changes in pixel values. So while we need to process the images in various methods we need to apply various filters mask etc in applications like edge detection, smoothing, removing noise etc.. Common filters that we use are High Pass filter, Low Pass filter, Ideal filter, Butterworth filter etc..

Let's try some processing..

We are going to work on a Gaussian Filter now.

$$H(u, v) = e^{-D^2(u, v)/2D_0^2}$$

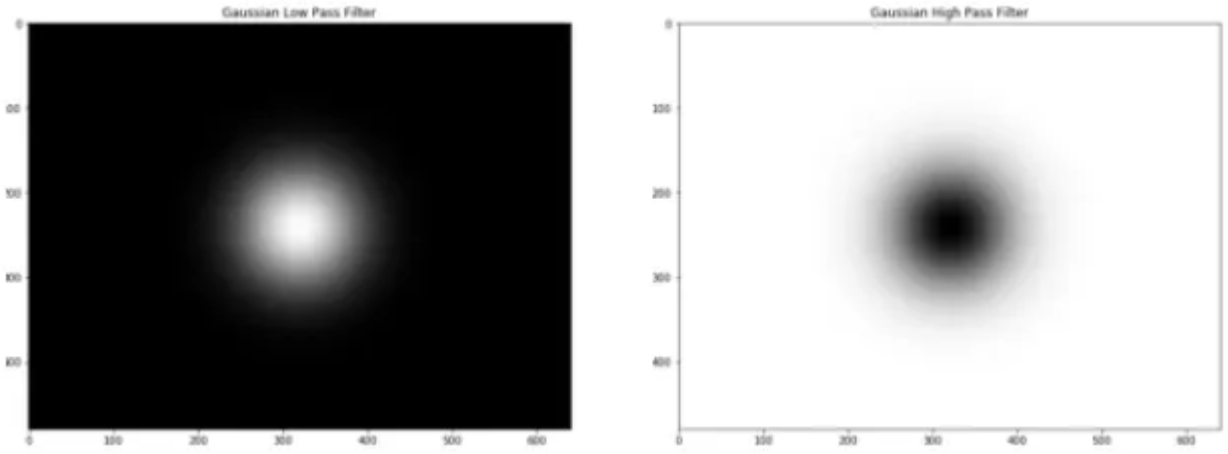
Gaussian Low Pass Filter — Source

$$H(u, v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

Gaussian High Pass Filter -Source

In this case formula for Gaussian low pass filter where D_0 is a positive constant and D(u, v) is the distance between a point (u, v) in the frequency domain and the center of the frequency rectangle. Formula for Gaussian high pass filter where D_0 is a positive constant and D(u, v) is the distance between a point (u, v) in the frequency domain and the center of the frequency rectangle. These equations are shown above.

How do these filters works?



Gaussian High and Low pass filter — Source

So in low pass filter only the centre portion has high values which diminishes going beyond centre. As we have already seen the centre contains low frequency components. Thus it removes high frequency component when we multiply and keep low frequency. The opposite happens in the other case.

Using this filters now our process is to multiply these filters along with image instead of convolving the filter to the entire matrix. (This is how Fourier Transform helps us in computation)

```
: ## Gaussian Filters

def distance(point1,point2):
    return np.sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2)

def gaussianLP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = np.exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base

def gaussianHP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1 - np.exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base
```

Function for Gaussian Filter

Now lets do the process again.

1. Convert image to Discrete Fourier Transform here we use Fast Fourier Transform.
2. Shift the origin to centre.
3. Apply filter by multiplying filter with fourier representation of image.
4. Reverse the shift.
5. Inverse fourier transform for image.

```
In [17]: img = cv2.imread("zebra.jpeg", 0)
         original = np.fft.fft2(img)
         center = np.fft.fftshift(original)

         plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

         plt.subplot(131), plt.imshow(img, "gray"), plt.title("Original Image")

         LowPassCenter = center * gaussianLP(50,img.shape)
         LowPass = np.fft.ifftshift(LowPassCenter)
         inverse_LowPass = np.fft.ifft2(LowPass)
         plt.subplot(132), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Gaussian Low Pass")

         HighPassCenter = center * gaussianHP(50,img.shape)
         HighPass = np.fft.ifftshift(HighPassCenter)
         inverse_HighPass = np.fft.ifft2(HighPass)
         plt.subplot(133), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Gaussian High Pass")

         plt.show()
```

Generated By Author

In the above code snippet we do the steps above and we got the result as observed.

Genearated By Author

Checkout the Github repo below.

raoofnaushad/Fourier-Transformation-for-Image-Processing

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or...

github.com

Help me to educate the children of rural India with dreams. Let us make some world leaders from them. https://www.buymeacoffee.com/raoofnaushA

Thanks for reading and for your time.

Gain Access to Expert View — Subscribe to DDI Intel

Image Processing | Computer Vision | Computational Thinking | Fourier Transform | Fast Fourier Transform

Follow

Written by Raoof Naushad

406 Followers · Writer for DataDrivenInvestor

Keep it Simple, but Significant.

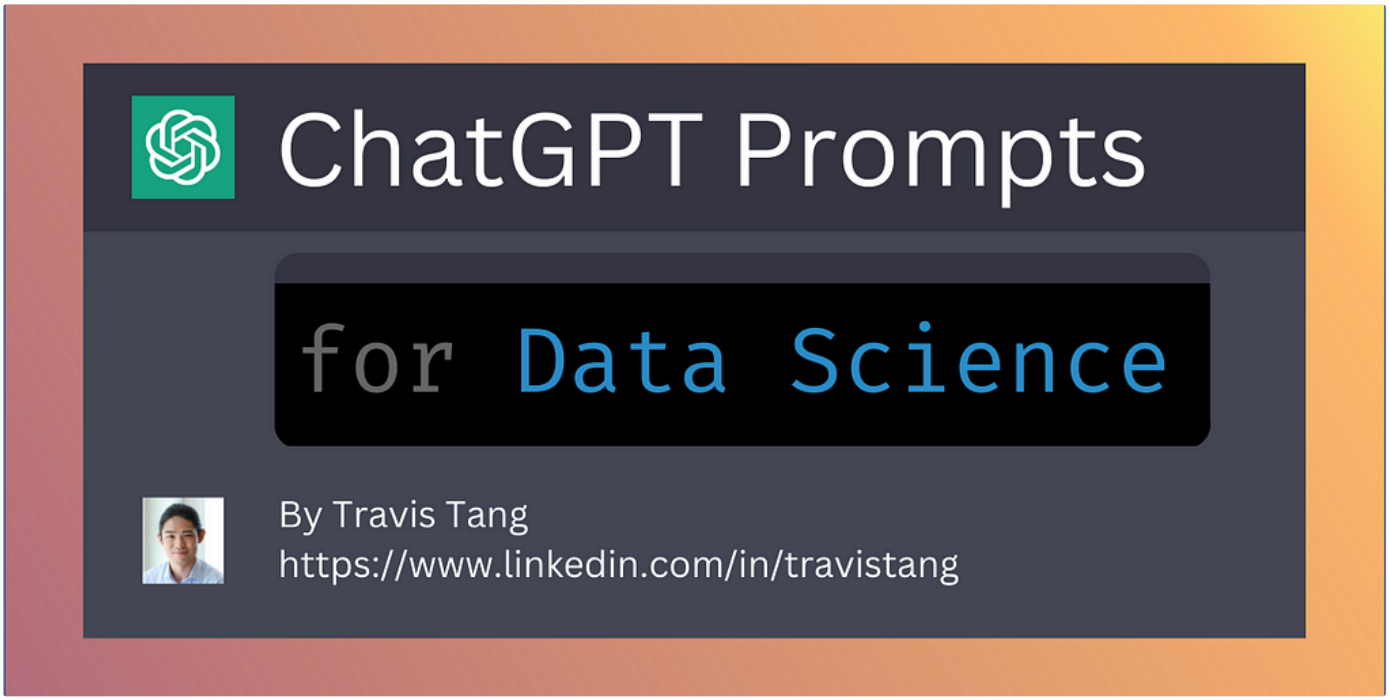
More from Raoof Naushad and DataDrivenInvestor

Raoof Naushad in Analytics Vidhya

Difference between WSGI and ASGI ?

This is a short story which will give you an idea about what is WSGI and ASGI.

· 3 min read · Jul 24, 2020



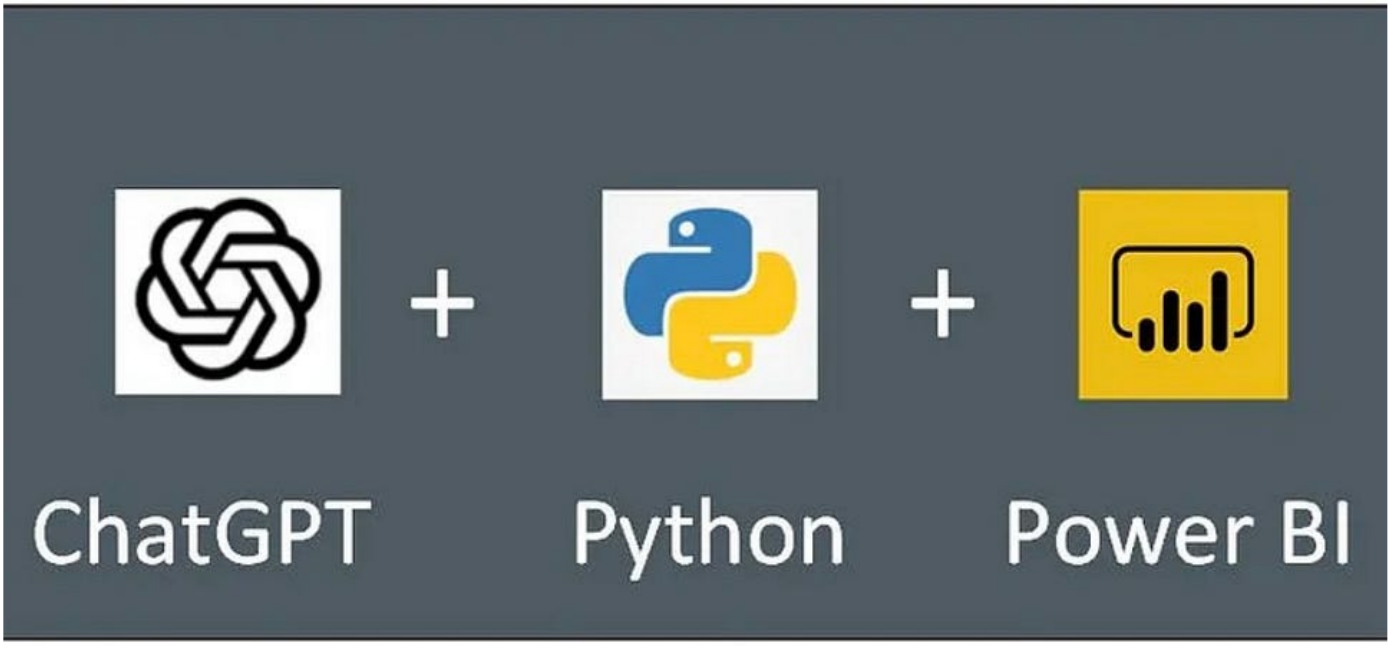
Travis Tang in DataDrivenInvestor

60 ChatGPT Prompts for Data Science (Tried, Tested, and Rated)

Automate data science tasks with ChatGPT

27 min read · Apr 11

1.4K 15



Gabe Araujo, M.Sc. in DataDrivenInvestor

ChatGPT + Python+ Power BI

You don't need to be an expert to integrate ChatGPT with Power BI. By following the steps outlined and leveraging the provided code...

· 9 min read · May 12

730

```python
import tkinter as tk

class App(tk.Tk):
    def __init__(self):
        super().__init__()

        ## Setting up Initial Things
        self.title("Sample Tkinter Structuring")
        self.geometry("720x550")
        self.resizable(True, True)
        self.iconphoto(False, tk.PhotoImage(file="assets/title_icon.png"))



if __name__ == "__main__":
    app = App()
    app.mainloop()

```

Raoof Naushad in DataDrivenInvestor

How to Create & Structure a Complex Tkinter Application [2022]

Structuring Tkinter Object Oriented Application

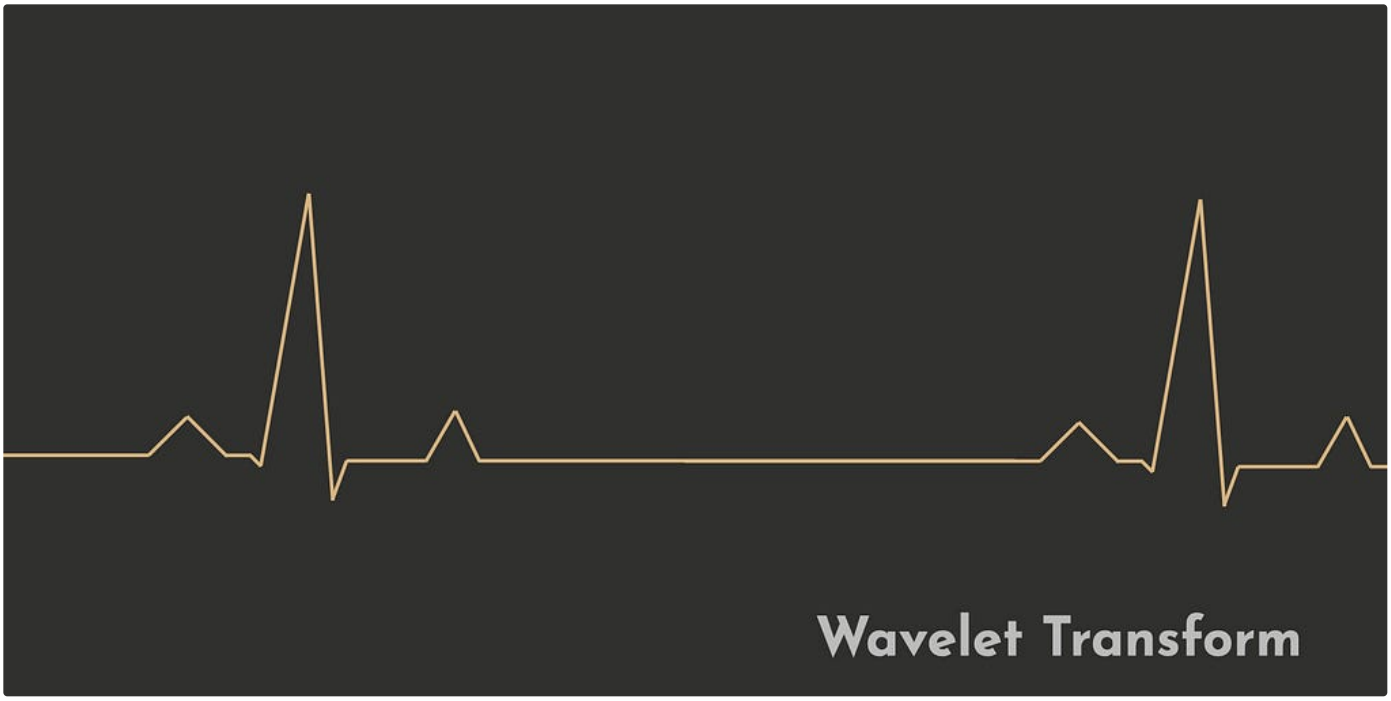
· 4 min read · Feb 6, 2022

119 1

See all from Raoof Naushad

See all from DataDrivenInvestor

Recommended from Medium

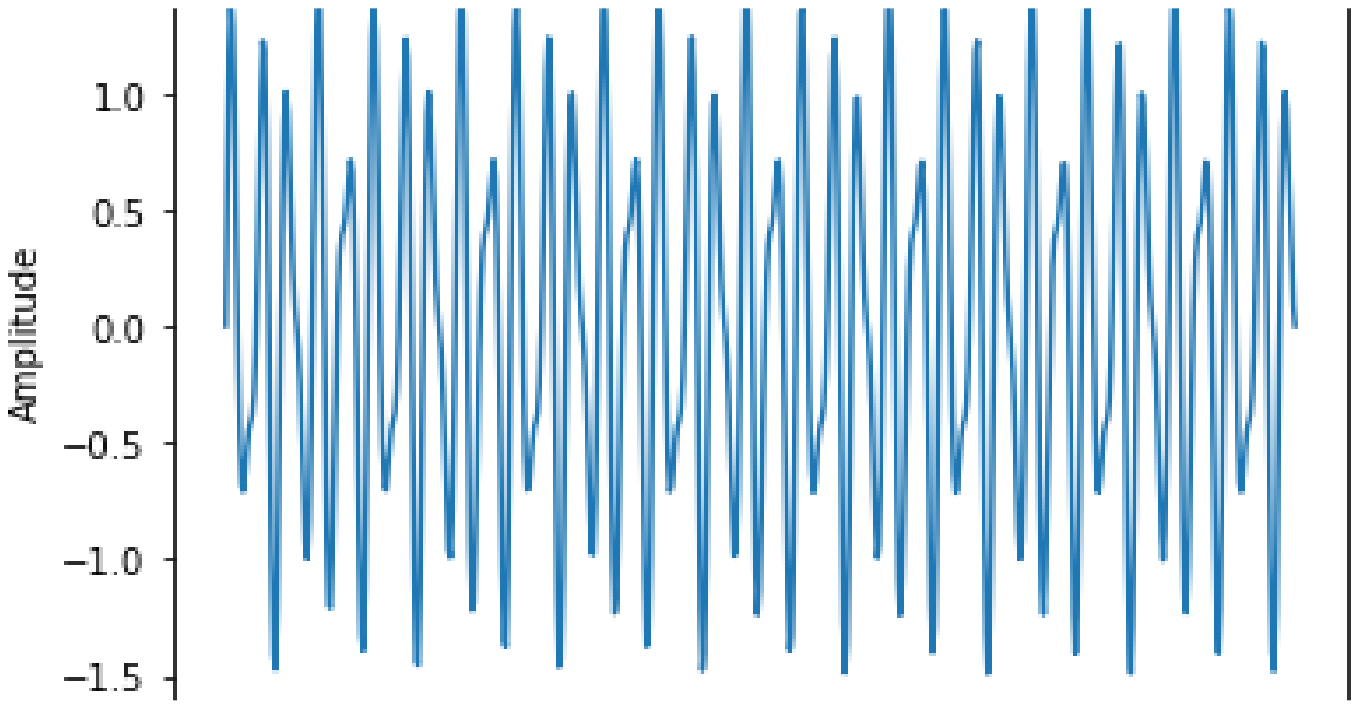


Shawhin Talebi in Towards Data Science

The Wavelet Transform

An Introduction and Example

· 6 min read · Dec 21, 2020



Nita Ghosh

Short Time Fourier Transform

The Short Time Fourier Transform (STFT) is a special kind of time-frequency analysis (others are wavelet analysis etc.) used in cases where...

· 3 min read · Dec 28, 2022

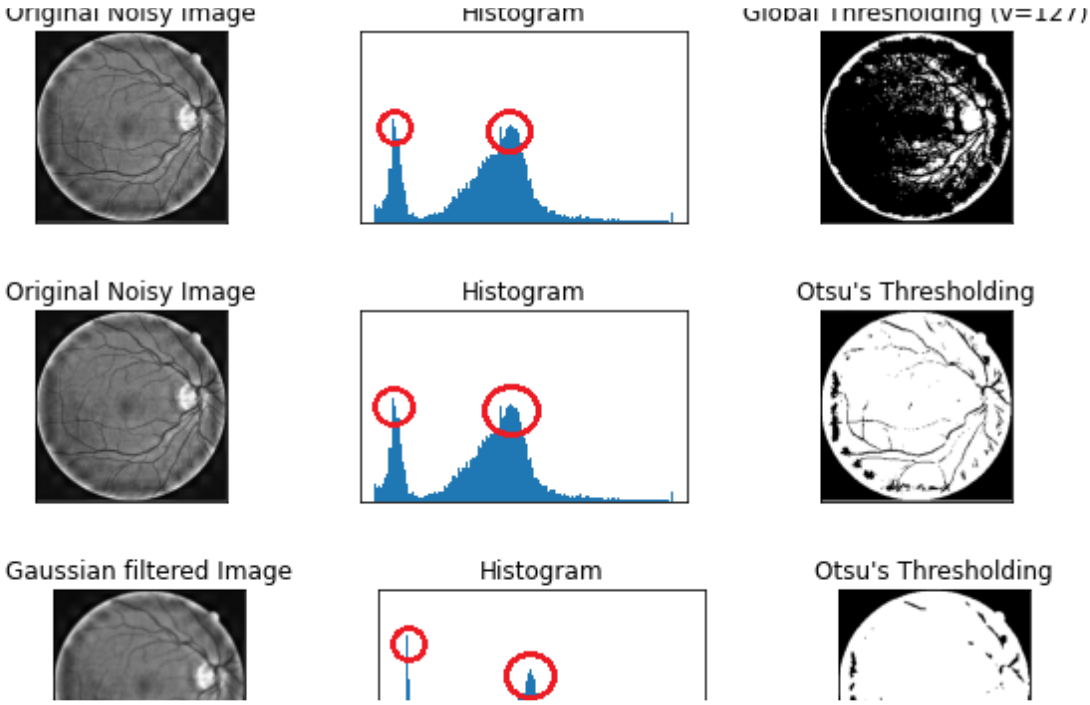
Dmitrii Eliuseev in Dev Genius

5 Open-Source Deep Learning Tools for Imaging Super-Resolution

Almost everybody has photos or videos made 10–20 years ago. These pictures were made in a time when 1600×1200 or even 640×480 image...

· 10 min read · Oct 14, 2022

67 4



Amit Chauhan

OpenCV: Adaptive and Otsu Threshold in Image Processing with Python

Image pre-processing techniques in artificial intelligence

· 3 min read · Mar 20

12

Sagar Shrestha in Level Up Coding

Learn How to Turn your Image into a Cartoon Using Python

This article will cover the various methods of turning a normal image into a cartoon version using Python. We will be using Python...

· 4 min read · Feb 8

200 2

Evgenii Munin in Better Programming

Camera Calibration on a Chessboard With Python and OpenCV

Estimate the intrinsic camera matrix and evaluate its precision

· 4 min read · Feb 22

156