**Create Any Image Using Only Sine Functions -  
2D Fourier Transform in Python**

**Report submitted in the fulfilment of**

**Project Work**

In

**Introduction to Data, Image and Signal Analysis(IDSA)**

By

**B-Tech Data Science (II Year)**

**J022 Rishi Gandhi**

Under the supervision of

**Prof. Jaykrishna Joshi**

**SVKM’s NMIMS University**

(Deemed-to-be University)



**MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING**

# TABLE OF CONTENTS:

|  |  |  |
| --- | --- | --- |
| **INDEX** | **TITLE** | **PAGE** |
| **1** | **Introduction** | **3** |
| **2** | **Objective** | **3** |
| **3** | **Methodology** | **4** |
| **4** | **Code** | **5-7** |
| **5** | **Working of code** | **8-9** |
| **6** | **Conclusion** | **10** |
| **7** | **References** | **10** |

# Introduction:

The aim of this project is to create any given image using only sine functions and implement the 2D Fourier Transform algorithm in Python. Fourier Transform is a mathematical technique used to analyse and transform signals into a frequency domain representation. It can be used to analyse and manipulate images by converting them from the spatial domain to the frequency domain.

# Objective:

The main objective of this project is to demonstrate the power of Fourier Transform in image processing and generate images using only sine functions. The specific objectives are as follows:

* Implement the 2D Fourier Transform algorithm in Python to convert an image from the spatial domain to the frequency domain.
* Generate sine functions with varying frequencies and amplitudes to approximate the image in the frequency domain.
* Use the inverse 2D Fourier Transform to convert the frequency domain image back to the spatial domain and create the final image.
* Test the method on a variety of images and compare the results with the original images to evaluate the accuracy of the generated images.

# Methodology:

The project will be implemented using the following steps:

* Load the desired image into Python and convert it to a grayscale image.
* Implement the 2D Fourier Transform algorithm using the numpy and scipy libraries in Python to convert the image from the spatial domain to the frequency domain.
* Generate a series of sine functions with varying frequencies and amplitudes to approximate the image in the frequency domain.
* Use the inverse 2D Fourier Transform to convert the frequency domain image back to the spatial domain and create the final image.
* Evaluate the accuracy of the generated image by comparing it with the original image using various performance metrics such as mean squared error, peak signal-to-noise ratio, and structural similarity index.

# Code:-

*# fourier\_synthesis.py*

**import** time

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

image\_filename **=** "Rishi.jpg"

start **=** time**.**time()

**def** calculate\_2dft(input):

ft **=** np**.**fft**.**ifftshift(input)

ft **=** np**.**fft**.**fft2(ft)

**return** np**.**fft**.**fftshift(ft)

**def** calculate\_2dift(input):

ift **=** np**.**fft**.**ifftshift(input)

ift **=** np**.**fft**.**ifft2(ift)

ift **=** np**.**fft**.**fftshift(ift)

**return** ift**.**real

**def** calculate\_distance\_from\_centre(coords, centre):

*# Distance from centre is √(x^2 + y^2)*

**return** np**.**sqrt(

(coords[0] **-** centre) **\*\*** 2 **+** (coords[1] **-** centre) **\*\*** 2

)

**def** find\_symmetric\_coordinates(coords, centre):

**return** (centre **+** (centre **-** coords[0]),

centre **+** (centre **-** coords[1]))

**def** display\_plots(individual\_grating, reconstruction, idx):

plt**.**subplot(121)

plt**.**imshow(individual\_grating)

plt**.**axis("off")

plt**.**subplot(122)

plt**.**imshow(reconstruction)

plt**.**axis("off")

plt**.**suptitle(f"Terms: {idx}")

plt**.**pause(0.01)

*# Read and process image*

image **=** plt**.**imread(image\_filename)

image **=** image[:, :, :3]**.**mean(axis**=**2) *# Convert to grayscale*

*# Array dimensions (array is square) and centre pixel*

array\_size **=** len(image)

centre **=** int((array\_size **-** 1) **/** 2)

*# Get all coordinate pairs in the left half of the array,*

*# including the column at the centre of the array (which*

*# includes the centre pixel)*

coords\_left\_half **=** (

(x, y) **for** x **in** range(array\_size) **for** y **in** range(centre**+**1)

)

*# Sort points based on distance from centre*

coords\_left\_half **=** sorted(

coords\_left\_half,

key**=lambda** x: calculate\_distance\_from\_centre(x, centre)

)

plt**.**set\_cmap("gray")

ft **=** calculate\_2dft(image)

*# Show grayscale image and its Fourier transform*

plt**.**subplot(121)

plt**.**imshow(image)

plt**.**axis("off")

plt**.**subplot(122)

plt**.**imshow(np**.**log(abs(ft)))

plt**.**axis("off")

plt**.**pause(2)

*# Reconstruct image*

fig **=** plt**.**figure()

*# Step 1*

*# Set up empty arrays for final image and*

*# individual gratings*

rec\_image **=** np**.**zeros(image**.**shape)

individual\_grating **=** np**.**zeros(

image**.**shape, dtype**=**"complex"

)

idx **=** 0

*# All steps are displayed until display\_all\_until value*

display\_all\_until **=** 10

*# After this, skip which steps to display using the*

*# display\_step value*

display\_step **=** 100

*# Work out index of next step to display*

next\_display **=** display\_all\_until **+** display\_step

*# Step 2*

**for** coords **in** coords\_left\_half:

*# Central column: only include if points in top half of*

*# the central column*

**if** **not** (coords[1] **==** centre **and** coords[0] **>** centre):

idx **+=** 1

symm\_coords **=** find\_symmetric\_coordinates(

coords, centre

)

*# Step 3*

*# Copy values from Fourier transform into*

*# individual\_grating for the pair of points in*

*# current iteration*

individual\_grating[coords] **=** ft[coords]

individual\_grating[symm\_coords] **=** ft[symm\_coords]

*# Step 4*

*# Calculate inverse Fourier transform to give the*

*# reconstructed grating. Add this reconstructed*

*# grating to the reconstructed image*

rec\_grating **=** calculate\_2dift(individual\_grating)

rec\_image **+=** rec\_grating

*# Clear individual\_grating array, ready for*

*# next iteration*

individual\_grating[coords] **=** 0

individual\_grating[symm\_coords] **=** 0

*# Don't display every step*

**if** idx **<** display\_all\_until **or** idx **==** next\_display:

**if** idx **>** display\_all\_until:

next\_display **+=** display\_step

*# Accelerate animation the further the*

*# iteration runs by increasing*

*# display\_step*

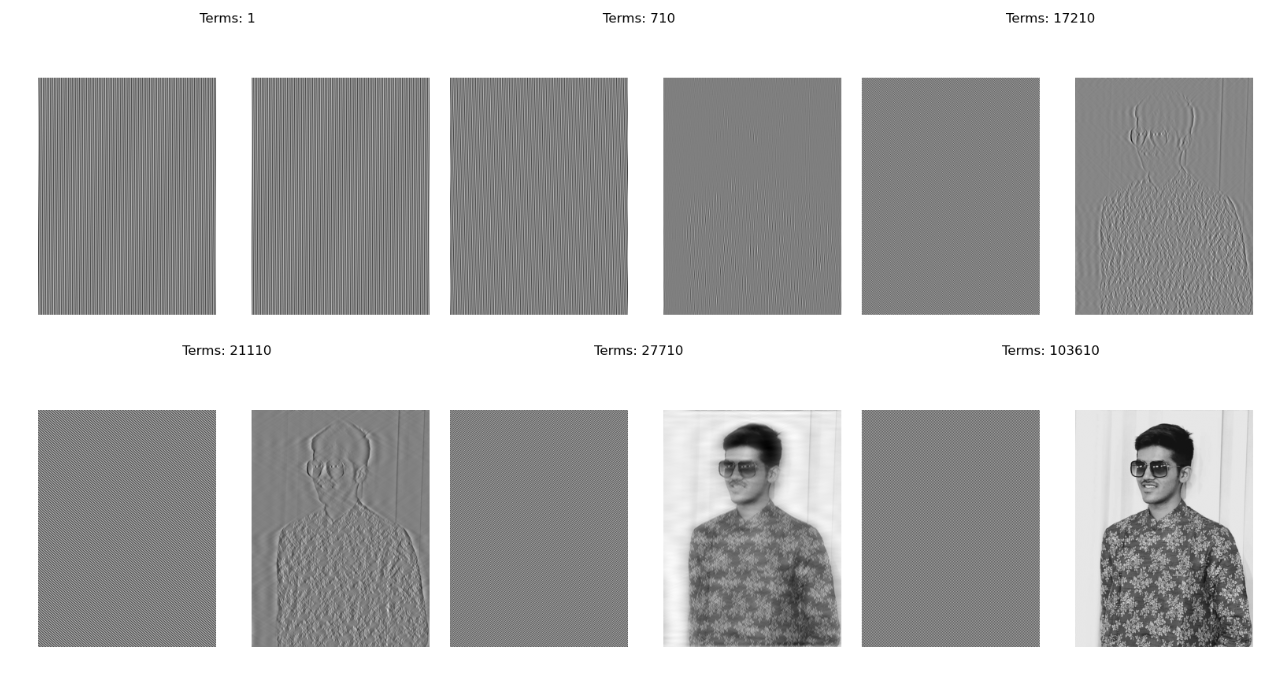
display\_step **+=** 100

display\_plots(rec\_grating, rec\_image, idx)

plt**.**show()

print("Time taken :",time**.**time()**-**start)

Output:-



# Working of code:-

The project focuses on demonstrating how sine functions can be used to create any image through the 2D Fourier Transform in Python. The project contains two Python scripts, "gratings.py" and "fourier\_synthesis.py". The "gratings.py" script generates an image of a sine wave, calculates its Fourier Transform and displays the inverse Fourier Transform of the image. The "fourier\_synthesis.py" script reads an image, calculates its Fourier Transform, applies Fourier Synthesis to the image, and then displays the reconstructed image.

Gratings.py

The "gratings.py" script generates an image of a sine wave, calculates its Fourier Transform and displays the inverse Fourier Transform of the image. The following steps are performed in the script:

1. Create a one-dimensional array "x" containing values ranging from -500 to 500.
2. Create a two-dimensional array "X" and "Y" using numpy's meshgrid function.
3. Set the wavelength of the sine wave to 100 and angle to pi/9 radians.
4. Use the sine function to generate the sine wave.
5. Display the image using matplotlib's imshow function.
6. Calculate the Fourier Transform of the image.
7. Use matplotlib's imshow function to display the absolute value of the Fourier Transform.
8. Use numpy's ifftshift and ifft2 functions to calculate the inverse Fourier Transform of the Fourier Transform.
9. Display the reconstructed image using matplotlib's imshow function.

Fourier\_synthesis.py

The "fourier\_synthesis.py" script reads an image, calculates its Fourier Transform, applies Fourier Synthesis to the image, and then displays the reconstructed image. The following steps are performed in the script:

1. Read the image using matplotlib's imread function.
2. Convert the image to grayscale by taking the average of the RGB values of each pixel.
3. Calculate the dimensions of the image and the centre pixel.
4. Create a generator object containing all coordinate pairs in the left half of the array, including the column at the centre of the array.
5. Sort the coordinate pairs based on their distance from the centre.
6. Display the original image and its Fourier Transform.
7. Create an empty array for the final reconstructed image and an empty array for the individual gratings.
8. Iterate over each coordinate pair in the left half of the array.
9. If the current coordinate pair is not in the central column or in the bottom half of the central column, continue to the next iteration.
10. Calculate the symmetric coordinate pair of the current coordinate pair.
11. Copy the values from the Fourier Transform array into the individual grating array for the current and symmetric coordinate pairs.
12. Calculate the inverse Fourier Transform of the individual grating and add it to the final reconstructed image.
13. Clear the values from the individual grating array for the current and symmetric coordinate pairs.
14. Display the individual grating and the reconstructed image if the current iteration is less than a certain value or if the current iteration is a multiple of a certain value.

# Conclusion :-

The project titled "Create Any Image Using Only Sine Functions -2D Fourier Transform in Python" successfully demonstrates the use of sine functions to create any image through the 2D Fourier Transform in Python. The project contains two Python scripts, "gratings.py" and "fourier\_synthesis.py", which perform the necessary operations to generate and reconstruct the images. The project is useful in understanding the mathematical concepts behind the Fourier Transform and how they can be applied in image processing. The scripts can be further developed to generate more complex images and to perform more advanced image processing tasks

# References :-

This project was referred from the below website :-

* [2D Fourier transform in Python: Create any image using only sine functions](https://thepythoncodingbook.com/2021/08/30/2d-fourier-transform-in-python-and-fourier-synthesis-of-images/)