Journal of Applied Mathematics and Physics, 2025, 13(5), 1744-1761
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Super-Resolution Using Fourier Image Transformer

Leith Uwaydah

American Community School Beirut (ACS), Beirut, Lebanon
Email: uwaydahleith@gmail.com

How to cite this paper: Uwaydah, L. (2025) Super-Resolution Using Fourier Image Transformer. *Journal of Applied Mathematics and Physics*, **13**, 1744-1761.
https://doi.org/10.4236/jamp.2025.135097

Received: March 6, 2025
Accepted: May 23, 2025
Published: May 26, 2025

Copyright © 2025 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

The primary goal of Super-Resolution (SR) is to reconstruct an image of higher quality from an image of lower quality. The Fourier transform helps in this process by recovering lost, high-frequency details. Various applications, such as satellite imaging, forensics, and surveillance, require high resolution, particularly when zooming in on specific parts of an image. To address this challenge and enhance image reconstruction, this article explores the Fourier transform and the Fourier Inverse transform. We can predict missing high-frequency components, resulting in a higher-resolution output from a low-resolution input. This approach demonstrates the practicality of solving relevant image analysis tasks in Fourier space, a domain that is inherently inaccessible to traditional convolutional architectures.

Keywords

Super Resolution, Image, Fourier Transform, Frequency, Time, Domain

1. Introduction

When you look at your own skin, do you see a single color? Certainly not; you will notice subtle variations in color and texture. This variation becomes really clear when we look at things under different magnifications or resolutions. **Figure 1** shows an image of human skin at multiple resolutions [1]. It reveals a whole new world of texture and color variation. There's nothing smooth or solid about this. Indeed, because we observe different patterns at different resolutions, we can conclude that natural surfaces exhibit a multi-resolution structure [2].

Let's do another example of this: the snake skin in **Figure 2**. At low resolution it looks like a collection of solid white and black scales. But at higher resolutions we see new color details. The white scales contain tiny brown dots and the black scales have a light coloration around the edges. And if we keep zooming in things

Figure 1. Multiple resolution skin human hand.

get really crazy. It is fascinating how, the closer one examines, the more details emerge. Exactly, and that's the essence of multiresolution structures. The color and texture of an object is the result of adding all of the different details together.

Figure 2. Multiple resolution skin snake.

In the era of high-definition imaging and data-driven analysis, enhancing image resolution has become a fundamental challenge in various fields, including medical imaging, remote sensing, and computer vision. Super-resolution (SR) techniques aim to reconstruct high-resolution images from low-resolution inputs, improving visual quality and enabling finer details to be extracted.

To improve resolution, various mathematical techniques are employed, including deep learning, inverse Fourier methods, interpolation, and super-resolution reconstruction.

One of the most powerful mathematical tools used in super-resolution is the Fourier transform, which provides a frequency-domain perspective for analyzing and enhancing image structures.

The Fourier transform decomposes an image into its constituent frequency components, making it possible to manipulate and reconstruct details that may be blurred or lost in low-resolution images. Using Fourier-based techniques, super-resolution methods can amplify high-frequency components and recover fine details with greater accuracy. These approaches are particularly useful in overcoming the limitations imposed by optical and sensor constraints in imaging systems.

In this article, we explore the principles of super-resolution, the role of the Fourier transform in frequency-domain enhancement, and various methodologies that integrate Fourier-based techniques with modern machine learning approaches. First, we start by an overview about super resolution image illustrated in Section 2, followed by the Fourier Transform and its inverse in Section 3 and 4. Additionally, we discuss an application in this context in Section 5.2, concluding

in the last section, Section 6.

2. Super-Resolution Imaging (SR)

Super-resolution plays a crucial role in various domains by enhancing image quality beyond the limitations of imaging hardware. Enhanced visualization and interpretation of images are essential, making super-resolution a critical tool for applications requiring high detail. For example, in medical imaging, SR techniques help improve diagnostic accuracy by enhancing the resolution of MRI or CT scans. In remote sensing, they help analyze satellite images with greater precision, enabling better environmental monitoring and urban planning. In computer vision, SR improves facial recognition, object detection, and image restoration tasks as [3].

Beyond its practical applications, super-resolution also addresses fundamental challenges in signal processing, such as noise reduction and image reconstruction. The ability to recover fine details from low-resolution images benefits fields like astronomy, where telescopic images can be sharpened to reveal distant celestial objects more clearly. As computational methods advance, super-resolution continues to evolve, integrating deep learning techniques and frequency-domain approaches to achieve unprecedented levels of detail and accuracy.

Enhancing image resolution involves various mathematical techniques that aim to increase the detail and clarity of images. These methods range from traditional interpolation approaches to advanced deep learning algorithms. One of these methods is the Frequency Domain Methods like the Fourier Transform that allows the manipulation of specific frequency components. By enhancing high-frequency components, which correspond to edges and fine details, one can improve the perceived resolution of an image.

Typically, SR methods are evaluated using various metrics to gauge performance.

- **Use Standard Image Quality Metrics** like: Peak Signal-to-Noise Ratio (PSNR): Measures the similarity between the reconstructed image and the original high-resolution image. Higher PSNR indicates better quality [4].
- **Structural Similarity Index (SSIM)**: Measures perceived quality by comparing luminance, contrast, and structure between images. It tends to be more aligned with human perception [5].
- **Mean Squared Error (MSE)**: Quantifies the error between the predicted and ground truth images [6].
- **Learned Perceptual Image Patch Similarity (LPIPS)**: A more recent metric that measures perceptual similarity between images. Perform comparisons of your SR technique's results with these metrics, especially when comparing to other SR techniques [7].

In this article, we do not make this comparison but will do it in future work.

Overall, super-resolution imaging represents a significant advancement in both scientific research and practical applications, providing deeper insights into mi-

croscopic structures and enhancing the quality of digital images [8] and [9].

3. Fourier Transform

Frequency domain analysis and Fourier transforms are a cornerstone of signal and system analysis. These concepts are also foundational pillars within electrical engineering. Among all of the mathematical tools utilized in electrical engineering, frequency domain analysis is arguably the most far-reaching. In fact, these ideas are so important that they are widely used in many fields-not just in electrical engineering, but in practically all branches of engineering and science, and several areas of mathematics.

The time and frequency domains are alternative ways of representing signals. The Fourier transform is the mathematical relationship between these two representations. If a signal is modified in one domain, it will also be changed in the other domain, although usually not in the same way [10].

In 1822, Joseph Fourier published his work on heat flow, in which he showed that the functions of a variable can be converted into a series of sinusoidal functions. This was originally used to convert periodic functions to a sum called the Fourier series. Later, it was generalized to non-periodic functions, using the Fourier transform (FT). FT is an integral transform that decomposes a signal into its constituent components and frequencies. FT is capable of decomposing a complicated waveform into a sequence of simpler elemental waves (more specifically, a weighted sum of sines and cosines). This is analogous to how an image is represented by FT. The image is a sum of complex exponential of varying magnitudes, frequencies, and phases.

The Fourier transform plays a critical role in a broad range of image processing applications, including enhancement, analysis, restoration, and compression. It breaks down an image into its basic frequency components. They employ methods such as self-attention to assess the significance of various sections of the input data, enhancing their ability to comprehend context and patterns. The result of the transformation depicts the image in the Fourier or frequency domain, whereas the input image corresponds to the spatial domain version. In the Fourier domain image, every point signifies a specific frequency found in the spatial domain image.

The inclusion of Fourier transforms into transformer models could help establish effective communication between the image frequency components and image description so that the model captures such features of the image remarkably well. In turn, it will improve the performance of image classification, segmentation, and generation tasks. The technique of processing an image to isolate or obtain the more useful information, and to remove unwanted information, such as the background noise and then will obtain clearer image as in **Figure 3**.

The frequency in images is closely related to the concept of spatial frequency, which describes how the intensity of the image changes across space. Low frequency represents smooth variations in intensity, such as large objects or gradual shading, while high frequency corresponds to rapid changes in intensity, such as

edges, textures, and fine details. Here is a picture taken from Foundations of Vision by Brian Wandell, who is in the Psychology Department at Stanford.

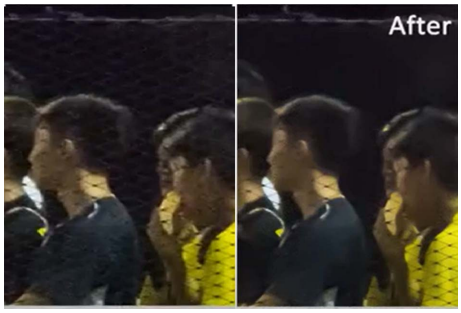

Figure 3. Result of Fourier in image after removing noise.

Figure 4. Frequency in image.

The shades of blue and yellow in **Figure 4** are the same in the two pictures, the only change is in the frequency. The closer spacing "mixes" the blue and yellow to give a greenish cast.

Mathematically, when we apply the Fourier transform to our image, we can decompose it into the various frequencies that it was originally composed of. Then, it will be easy to isolate the frequencies that are undesirable and apply our filter to remove them. Once the filter is applied, the inverse Fourier transform is applied, and we get back our image as a function of time with a super resolution one. **Figure 5** shows the application of the Fourier transform when removing noise from a signal.

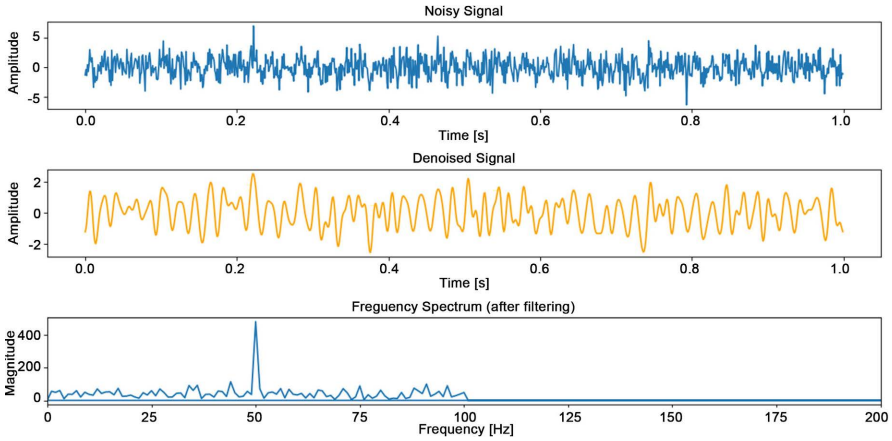

Figure 5. Denoise signal using Fourier.

4. Mathematical Aspect

The Fourier transform can be applied in both continuous and discrete contexts, each with its own specific mathematical treatment. In this article we work with Discret Fourier Transform.

4.1. Discrete Fourier Transform (DFT)

The DFT is one of the most powerful tools in digital signal processing, allowing us to calculate the spectrum of a finite-duration signal as in [10]. For image, it is a representation of an image as a sum of complex exponential of varying magnitudes. DFT transform Time Domain function $x(t)$ to Frequency Domain function $X(w)$ as in [11] [12] and [13].

In particular, DFT creates the relationship between sampled signals in the time range and their representation in the frequency domain. The DFT is widely used in the fields of spectral analysis, applied mechanics, acoustics, medical imaging, numerical analysis, instrumentation, and telecommunications. The following figure shows how to use the DFT to transform data from the time domain into the frequency domain (**Figure 6**).

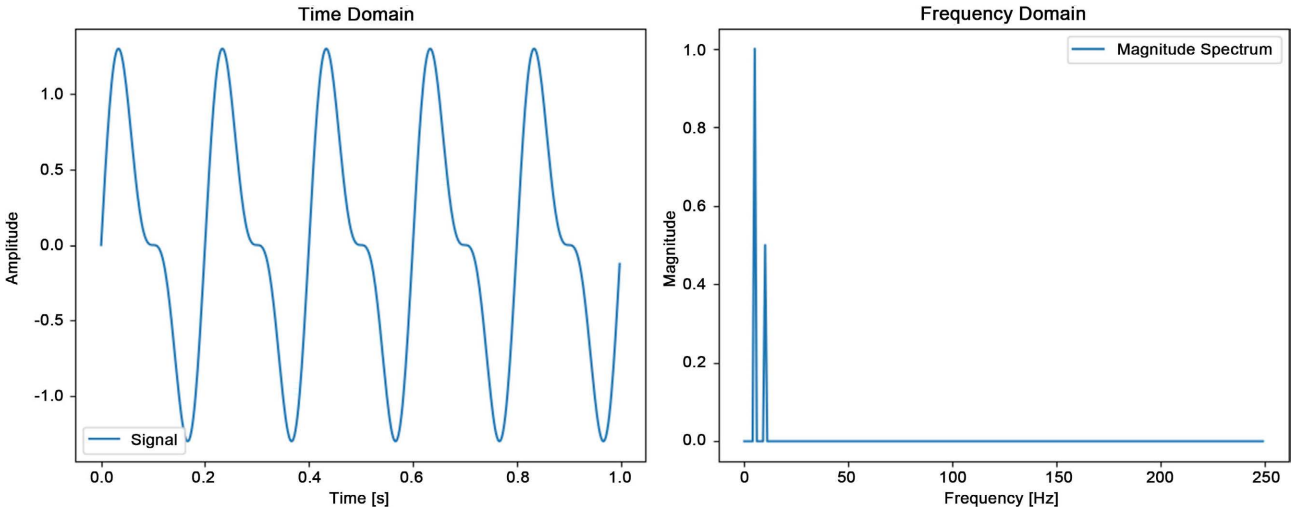

Figure 6. Fourier in time and frequency domain.

The mathematical equation for DFT is given by

$$\mathcal{F}(\omega) = X(k) = \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-j2\pi kn}{N}}$$

and the inverse function IDFT is given by:

$$f(t) = x(n) = \sum_{k=0}^{N-1} X[k] \cdot e^{\frac{j2\pi kn}{N}}$$

N samples of the input signal result in N samples of the discrete Fourier transform (DFT). That is, the number of samples in both the time and frequency representations is the same. The following equation shows that regardless of whether

the input signal $x(n)$ is real or complex, $X(k)$ is always complex, although the imaginary part may be zero. In other words, every frequency component has a magnitude and phase.

Properties of DFT

1) **Linearity:** The addition of two functions corresponding to the addition of the two frequency spectrum is called linearity. If we multiply a function by a constant, the Fourier transform of the resultant function is multiplied by the same constant. The Fourier transform of the sum of two or more functions is the sum of the Fourier transforms of the functions. The Fourier transform is a linear operator, meaning that for any functions $f(x)$ and $g(x)$, and constants *a* and *b*:

$$\mathcal{F}\{af(x)+bg(x)\}=a\mathcal{F}\{f(x)\}+b\mathcal{F}\{g(x)\}$$

This property allows for the decomposition of complex signals into simpler components, facilitating analysis and processing.

2) **Scaling:** Scaling is the method that is used to change the range of the independent variables or features of data. If we stretch a function by the factor in the time domain then squeeze the Fourier transform by the same factor in the frequency domain. Scaling the time variable by a factor aa affects both the amplitude and width of the Fourier transform:

$$\mathcal{F}\{f(ax)\}=\frac{1}{|a|}\mathcal{F}\left\{f\left(\frac{\xi}{a}\right)\right\}$$

Time compression ($|a|>1$) leads to frequency domain expansion, while time expansion ($|a|<1$) results in frequency domain compression.

3) **Differentiation:** Differentiating function with respect to time yields to the constant multiple of the initial function.

To find the Fourier transform of the derivative of $f(t)$, we use the property of derivatives under the Fourier transform:

$$\mathcal{F}\left\{\frac{\mathrm{d}}{\mathrm{d}t}f(t)\right\}=i\omega\mathcal{F}(\omega)$$

For higher-order derivatives, this relationship generalizes as:

$$\mathcal{F}\left\{\frac{\mathrm{d}^n}{\mathrm{d}t^n}f(t)\right\}=(i\omega)^n\mathcal{F}(\omega)$$

Thus, the Fourier transform of the *n*-th derivative of $f(t)$ is simply $(i\omega)^n$ times the Fourier transform of $f(t)$.

4) **Convolution:** It includes the multiplication of two functions. The Fourier transform of a convolution of two functions is the point-wise product of their respective Fourier transforms.

Convolution in the Time Domain:

The convolution of two functions $f(t)$ and $g(t)$ in the time domain is defined as:

$$\int_{-\infty}^{\infty}f(\tau)g(t-\tau)\mathrm{d}\tau$$

where:

- $f(t)$ and $g(t)$ are two functions (signals).
- τ is a dummy variable of integration.
- $*$ represents the convolution operator.
- The result is a new function, $(f*g)(t)$, which is the combined effect of $f(t)$ and $g(t)$.

Convolution in the Frequency Domain:

One of the most important properties of the Fourier transform is that convolution in the time domain corresponds to multiplication in the frequency domain.

Specifically:

$$\mathcal{F}\{(f*g)(t)\}(\omega)=\mathcal{F}\{f(t)\}(\omega)\cdot\mathcal{F}\{g(t)\}(\omega)$$

That is:

$$F(\omega)\cdot G(\omega)=\mathcal{F}\{f*g\}(\omega)$$

where:

- $\mathcal{F}\{f(t)\}(\omega)$ and $\mathcal{F}\{g(t)\}(\omega)$ are the Fourier transforms of $f(t)$ and $g(t)$, respectively.
- $F(\omega)$ and $G(\omega)$ are the frequency-domain representations of the functions $f(t)$ and $g(t)$, respectively.
- The convolution $(f*g)(t)$ in the time domain corresponds to the pointwise multiplication of their Fourier transforms in the frequency domain.

5) **Frequency Shift and Time Shift:** Frequency is shifted according to the coordinates. There is a duality between the time and frequency domains and frequency shift affects the time shift. The time variable shift also affects the frequency function. The time-shifting property concludes that a linear displacement in time corresponds to a linear phase factor in the frequency domain.

Shifting a function in time corresponds to a phase shift in the frequency domain:

$$\mathcal{F}\{f(x-x_0)\}=\mathrm{e}^{-i2\pi x_0\xi}\mathcal{F}\{f(x)\}$$

here, x_0 is the shift in the time domain, and $\mathrm{e}^{-i2\pi x_0\xi}$ represents the phase shift in the frequency domain.

Modulating a function by a complex exponential in the time domain results in a shift in the frequency domain:

$$\mathcal{F}\{\mathrm{e}^{i2\pi\xi_0 x}f(x)\}=\mathcal{F}\{f(x)\}(\xi-\xi_0)$$

This property is fundamental in frequency modulation and demodulation processes.

Normally the magnitude of the spectrum is displayed. The magnitude is the square root of the sum of the squares of the real and imaginary parts as follows:

$$\text{magnitude}(\mathcal{F})=\sqrt{\text{real}(\mathcal{F})^2+\text{imaginary}(\mathcal{F})^2}$$

$$\text{phase}(\mathcal{F})=\tan^{-1}\left(\frac{\text{imaginary}(\mathcal{F})}{\text{real}(\mathcal{F})}\right)$$

Briefly, the magnitude indicates the amount of a specific frequency component present, determining the relative presence of a sinusoid $e^{\frac{j2\pi kt}{N}}$ in $f(t)$. The phase tells "where" the frequency component is in the image, more it determines how the sinusoids line up relative to one another to form $f(t)$ or it determines the shift in the sinusoid components of the image.

Graphically, Figure 7 shows us an image in term of magnitude and phase.

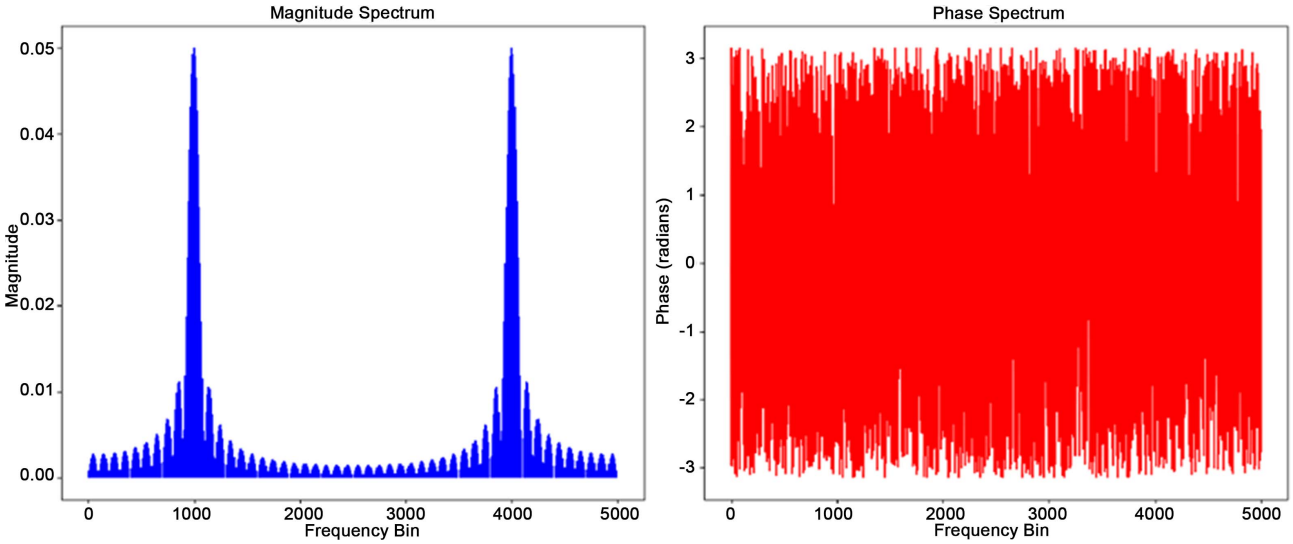

Figure 7. Fourier in magnitude and phase.

Figure 8 and Figure 9 show how Fourier transform divides an image to magnitude and phase. The initial figure in time domain for two persons with their magnitudes and phases.

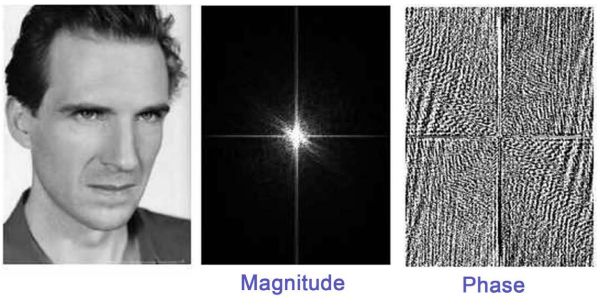

Figure 8. Fourier in magnitude and phase.

The phase values determine the shift in the sinusoid components of the image.

With zero phase, all sinusoids are centered at the same location, resulting in a symmetric image that bears no real correlation with the original image. Figure 10 shows the image of Ralph with phase equal zero and same magnitude as Figure 8.

The phase-only reconstruction preserves features because of the principle of phase congruency. At the location of edges and lines, most of the sinusoid components have the same phase see [11]. This properly alone can be used to detect

lines and edges without regard to magnitude. So you can see that the phase information is most important.

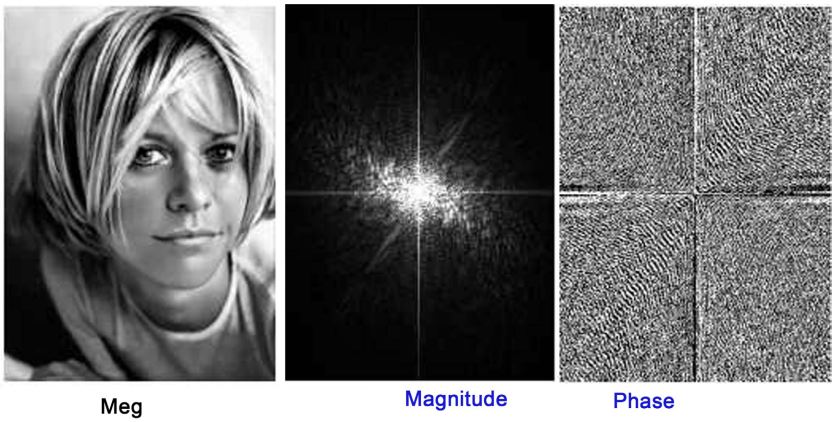

Figure 9. Fourier in magnitude and phase.

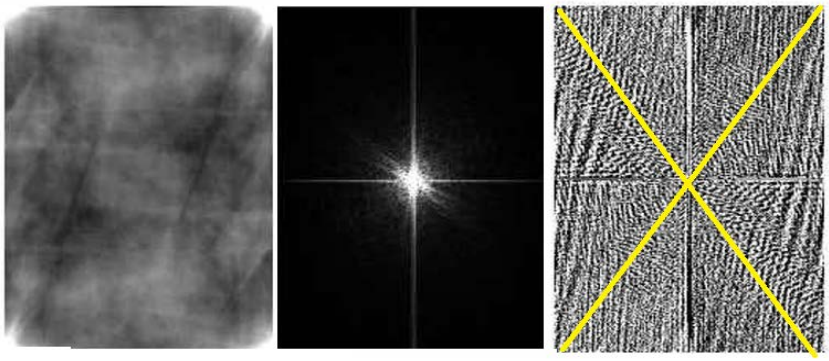

Figure 10. Ralph image with phase equal zero and same magnitude.

Changing the magnitude of the various component sinusoids changes the shape of the feature. When you do a phase-only reconstruction, you set all the magnitudes to one, which changes the shape of the features, but not their location. Figure 11 shows the image of Ralph with the same phase as in Figure 8 and magnitude equal to one.

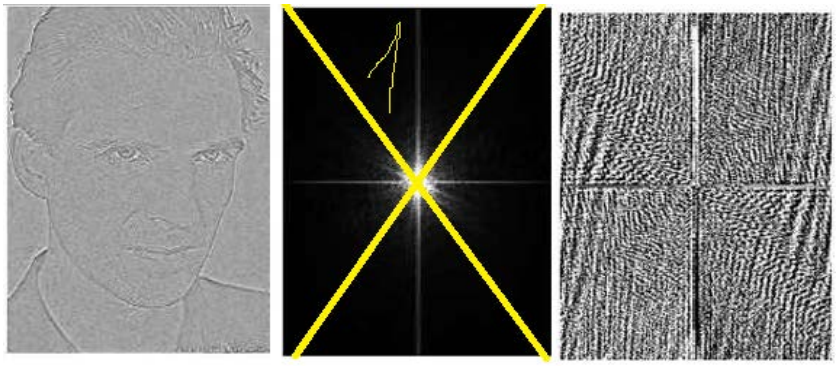

Figure 11. Ralph image with same phase and magnitude = 1.

In many images, low-frequency components have higher magnitudes than high-frequency components, making phase-only reconstruction resemble a high-pass filter. In short, phase contains the information about the locations of features. You cannot add the phase-only and magnitude-only images to get the original. You can multiply them in the Fourier domain and transform back to get the original. Fourier transforms and their general applications in image processing can be explored in recent advancements in Fourier transform techniques or provide comparative studies with other methods. For instance:

- **Comparative Analysis of Image Processing Techniques**: A study compared the Discrete Fourier Transform (DFT), K-Means clustering, and Artificial Neural Networks (ANN) for quality control in assembled tires, highlighting the strengths and weaknesses of each method as in [14].
- **Fourier vs. Wavelet Transform in Image Compression**: Research comparing Fourier Transform and Wavelet Transform for image compression found that Wavelet Transform generally offers better compression quality, though with increased computational complexity as in [15].

5. Application

5.1. Image in Gray Scale for Fourier Transform

The image is the raw input, as shown in **Figure 13**. It shows the original resolution and color information. Details and texture in the image appear as captured by the camera.

In this image, each color channel (R, G, B) has been independently upsampled using Fourier transform zero-padding. The result is a larger image with increased resolution. Although the zero-padding method doesn't create new details, it smooths and interpolates between existing ones. You should notice that the overall image looks smoother and may have less visible pixelation compared to simply stretching the image. This version converts the upsampled true-color image to grayscale. The grayscale image is useful for checking the structural and edge details. Sometimes, subtle features become more apparent in grayscale because the influence of color is removed. Comparing the grayscale result with the true-color one can help you evaluate whether the Fourier upsampling preserves important details. Note that in the first column of this figure we use samples = 4 while in the second one we have 8 samples. By running the following code with your fig.jpg file, you will be able to compare the original grayscale image with the Fourier-upsampled, red-tinted version (**Figure 12**).

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, img_as_float

def fourier_upsample(image, upsample_factor):
```

```
    """
    Upsample a 2D image using Fourier-domain zero-padding.

    Parameters:
        image (2D ndarray): Grayscale image.
        upsample_factor (int): Factor by which to upscale the image.

    Returns:
        upsampled_img (2D ndarray): Upsampled image.
    """
    orig_rows, orig_cols = image.shape
    new_rows, new_cols = orig_rows * upsample_factor, orig_cols * upsample_factor

    # Compute the 2D Fourier Transform and shift zero-frequency to the center.
    F = np.fft.fft2(image)
    F_shifted = np.fft.fftshift(F)

    # Create a zero-padded Fourier-domain array.
    F_padded = np.zeros((new_rows, new_cols), dtype=complex)

    # Calculate insertion indices to center the original Fourier data.
    row_start = (new_rows - orig_rows) // 2
    col_start = (new_cols - orig_cols) // 2

    F_padded[row_start:row_start+orig_rows, col_start:col_start+orig_cols] = F_shifted

    # Inverse shift and compute the inverse FFT.
    F_unshifted = np.fft.ifftshift(F_padded)
    upsampled_img = np.fft.ifft2(F_unshifted).real

    # Normalize to the range [0, 1].
    upsampled_img = (upsampled_img - upsampled_img.min()) / (upsampled_img.max() - upsampled_img.min())
    return upsampled_img

# ------------------------------
# Main Processing Workflow
# ------------------------------

# 1. Load the image (fig.jpg).
image_file = "fig.jpg"
```

```
image = io.imread(image_file)

# Convert to grayscale if the image is in color.
if image.ndim == 3:
    image = color.rgb2gray(image)
image = img_as_float(image)

# 2. Upsample the image using Fourier transform zero-padding.
upsample_factor = 4    # Adjust this value as needed.
upsampled_gray = fourier_upsample(image, upsample_factor)

# 3. Convert the upsampled grayscale image to a red-tinted image.
# Create an RGB image with red channel = upsampled grayscale and other channels
# zero.
red_image = np.zeros((upsampled_gray.shape[0], upsampled_gray.shape[1], 3))
red_image[:, :, 0] = upsampled_gray    # Assign grayscale to the red channel.

# ------------------------------
# Display the Results
# ------------------------------
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Grayscale Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(red_image)
plt.title(f'Upsampled Red-Tinted Image (x{upsample_factor})")
plt.axis("off")

plt.tight_layout()
plt.show()
```

5.2. Image in Red Scale for Fourier Transform

The final red-tinted image in **Figure 13** will have the same upsampled resolution as the grayscale image, but with a distinct red color. This can be useful if you want to visually differentiate the upsampled version from the original. Displays the red-tinted, upsampled image. You should notice that this image is larger (by a factor of 4 in both dimensions), and although it appears smoother, it retains the structure of the original image with a red color cast. Comparing the two images

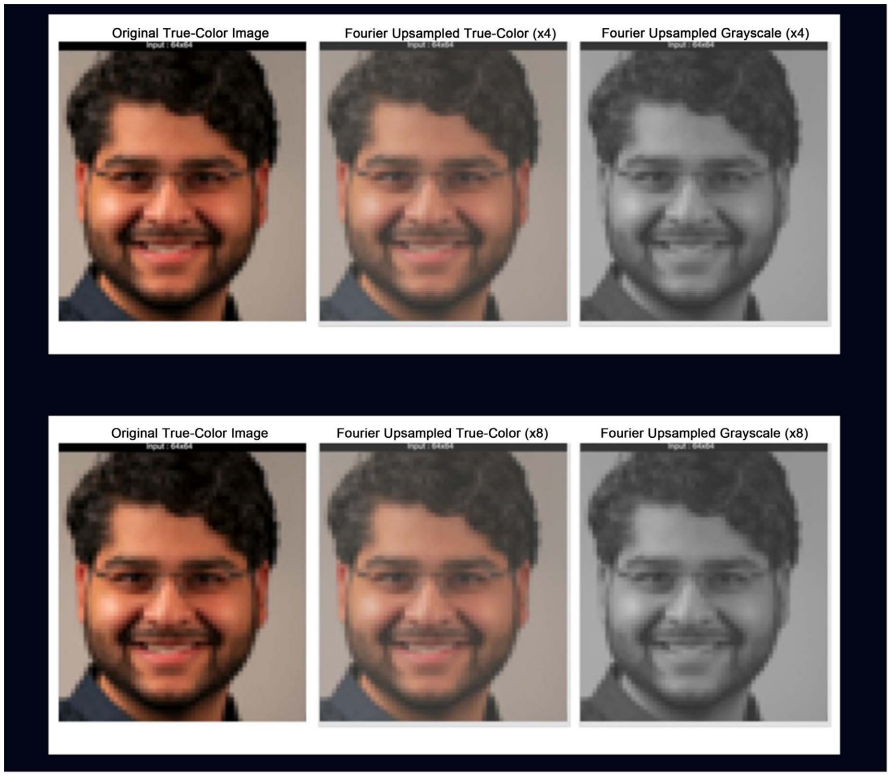

Figure 12. Resolution using Fourier 4 samples and 8 samples.

side-by-side allows you to see how the Fourier transform method increases resolution. Such techniques are often used in image processing for interpolation, frequency analysis, and sometimes for preparing images for further enhancement steps. By running the following code with your fig.jpg file, you will be able to compare the original grayscale image with the Fourier-upsampled, red-tinted version.

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, img_as_float

def fourier_upsample(image, upsample_factor):
    """
    Upsample a 2D image using Fourier-domain zero-padding.

    Parameters:
        image (2D ndarray): Grayscale image.
        upsample_factor (int): Factor by which to upscale the image.

    Returns:
        upsampled_img (2D ndarray): Upsampled image.
    """
    orig_rows, orig_cols = image.shape
```

```
    new_rows, new_cols = orig_rows * upsample_factor, orig_cols * upsample_factor

    # Compute the 2D Fourier Transform and shift zero-frequency to the center.
    F = np.fft.fft2(image)
    F_shifted = np.fft.fftshift(F)

    # Create a zero-padded Fourier-domain array.
    F_padded = np.zeros((new_rows, new_cols), dtype=complex)

    # Calculate insertion indices to center the original Fourier data.
    row_start = (new_rows - orig_rows) // 2
    col_start = (new_cols - orig_cols) // 2

    F_padded[row_start:row_start+orig_rows, col_start:col_start+orig_cols] = F_shifted

    # Inverse shift and compute the inverse FFT.
    F_unshifted = np.fft.ifftshift(F_padded)
    upsampled_img = np.fft.ifft2(F_unshifted).real

    # Normalize to the range [0, 1].
    upsampled_img = (upsampled_img - upsampled_img.min()) / (upsampled_img.max() - upsampled_img.min())
    return upsampled_img

# ------------------------------
# Main Processing Workflow
# ------------------------------

# 1. Load the image (fig.jpg).
image_file = "fig.jpg"
image = io.imread(image_file)

# Convert to grayscale if the image is in color.
if image.ndim == 3:
    image = color.rgb2gray(image)
image = img_as_float(image)

# 2. Upsample the image using Fourier transform zero-padding.
upsample_factor = 4    # Adjust this value as needed.
upsampled_gray = fourier_upsample(image, upsample_factor)

# 3. Convert the upsampled grayscale image to a red-tinted image.
```

```
# Create an RGB image with red channel = upsampled grayscale and other chan-
nels
# zero.
red_image = np.zeros((upsampled_gray.shape[0], upsampled_gray.shape[1], 3))
red_image[:, :, 0] = upsampled_gray    # Assign grayscale to the red channel.

# ------------------------------
# Display the Results
# ------------------------------
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Grayscale Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(red_image)
plt.title(f"Upsampled Red-Tinted Image (x{upsample_factor})")
plt.axis("off")

plt.tight_layout()
plt.show()
```

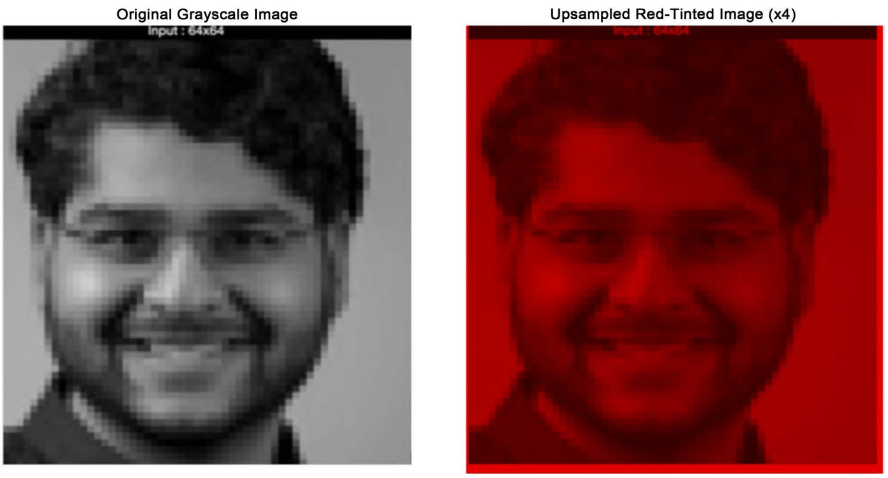

Figure 13. Resolution using Fourier and 4 samples.

6. Conclusions

The Discrete Fourier Transform (DFT) is a cornerstone of digital signal processing and has proven to be a powerful tool in image analysis and enhancement. Here are some key takeaways:

Frequency Domain Analysis: DFT allows you to transform spatial (or time-do-

main) data into the frequency domain, where different frequency components can be analyzed separately. This is crucial for understanding the underlying structures in an image or signal.

Resolution Enhancement via Zero-Padding: By zero-padding the Fourier spectrum, we effectively increase the sampling grid in the frequency domain. When the inverse DFT is applied, this process interpolates the spatial data, resulting in a higher-resolution image. However, it's important to note that this method only smooths and interpolates between existing data-it does not create new, previously unseen details.

Color vs. Grayscale Processing: When working with color images, processing each channel separately via Fourier techniques can preserve true color information while enhancing resolution. On the other hand, converting to grayscale can sometimes reveal more structural details, free from the complexity of color.

Practical Applications: DFT-based methods are widely used in various applications, from image upsampling and super resolution to noise reduction and frequency-based filtering. They provide an efficient way to analyze and manipulate images based on their spectral content.

Limitations: While DFT and zero-padding can improve visual resolution, they do not add new information beyond what is contained in the original image. This highlights the trade-off between interpolation and the recovery of lost details.

In summary, the DFT offers a robust framework for both analyzing and processing images. Its ability to bridge the spatial and frequency domains makes it indispensable for many image processing tasks, although its interpolation methods are inherently limited by the original data's content.

The Fourier Discrete Transform (DFT) serves as a powerful tool in super-resolution imaging by enabling frequency-based analysis and reconstruction of high-resolution images. While challenges remain, advancements in computational techniques and machine learning continue to push the boundaries of what is possible in image enhancement. By integrating Fourier-based approaches with modern AI techniques, super-resolution imaging will continue to evolve and find applications in medical imaging, remote sensing, and beyond.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Arun, P.V., Buddhiraju, K.M., Porwal, A. and Chanussot, J. (2020) CNN-Based Super-Resolution of Hyperspectral Images. *IEEE Transactions on Geoscience and Remote Sensing*, **58**, 6106-6121. https://doi.org/10.1109/tgrs.2020.2973370

[2] Lei, S., Shi, Z. and Zou, Z. (2020) Coupled Adversarial Training for Remote Sensing Image Super-Resolution. *IEEE Transactions on Geoscience and Remote Sensing*, **58**, 3633-3643. https://doi.org/10.1109/tgrs.2019.2959020

[3] Zalevsky, Z., Mendlovic, D. and Lohmann, A.W. (2000) IV Optical Systems with Improved Resolving Power. In: *Progress in Optics*, Elsevier, 271-341. https://doi.org/10.1016/s0079-6638(00)80032-3

[4] Hamid, R.A. and Hassan, M.I.D. (2009) Image Compression Using Wavelets Based on Peak Signal-to-Noise Ratio. *International Journal of Computer Science and Network Security*, **9**.

[5] Wang, Z., Bovik, A.C., Sheikh, H.R. and Simoncelli, E.P. (2004) Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, **13**, 600-612. https://doi.org/10.1109/tip.2003.819861

[6] Parker, J.R. (2010) Algorithms for Image Processing and Computer Vision. Wiley-Interscience.

[7] Zhang, D., Isola, P. and Efros, A.A. (2020) A Perceptual Similarity Metric for Image Generation. *IEEE Transactions on Image Processing*. https://hal.science/hal-04605516v1/document

[8] Bartelt, H. and Lohmann, A.W. (1982) Optical Processing of One-Dimensional Signals. *Optics Communications*, **42**, 87-91. https://doi.org/10.1016/0030-4018(82)90371-6

[9] Fixler, D., Garcia, J., Zalevsky, Z., Weiss, A. and Deutsch, M. (2007) Speckle Random Coding for 2D Super Resolving Fluorescent Microscopic Imaging. *Micron*, **38**, 121-128. https://doi.org/10.1016/j.micron.2006.07.008

[10] Thomas, R. (2017) Fourier Transforms of Images. https://plus.maths.org/content/fourier-transforms-images

[11] Peters, T.M. and Bates, J.H.T. (1998) The Discrete Fourier Transform and the Fast Fourier Transform. In: Peters, T.M. and Williams, J., Eds., *The Fourier Transform in Biomedical Engineering*, Birkhäuser Boston, 175-194. https://doi.org/10.1007/978-1-4612-0637-8_6

[12] https://www.robots.ox.ac.uk/~sjrob/Teaching/SP/l7.pdf

[13] Digital Image Processing, 4e. https://dl.icdst.org/pdfs/files4/01c56e081202b62bd7d3b4f8545775fb.pdf

[14] Massaro, A., Dipierro, G., Cannella, E. and Galiano, A.M. (2020) Comparative Analysis among Discrete Fourier Transform, K-Means and Artificial Neural Networks Image Processing Techniques Oriented on Quality Control of Assembled Tires. *Information*, **11**, Article 257. https://www.mdpi.com/2078-2489/11/5/257?utm_source=chatgpt.com

[15] Kan, Y. (2024) A Comparison of Application of Fourier Transform and Wavelet Transform on Image Compression. *Applied and Computational Engineering*, **37**, 149-154. https://www.ewadirect.com/proceedings/ace/article/view/10123?utm_source=chatgpt.com