IMAGE PROCESSING USING DFT

A MINI PROJECT REPORT

Submitted by

Anjanay Khare [RA2011032010007]
Paras Sharma [RA2011032010049]

Under the guidance of

Mrs. Sai Santhiya D
(Assistant Professor, Department of Networking and Communications)

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING
With specialization in Internet of Things

S.R.M.Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this project report **"Image Processing Using DFT"** is the bonafide work of **"Anjanay Khare (RA2011032010007) and Paras Sharma (RA2011032010049)"** of III Year/VI Sem B.tech(CSE) who carried out the mini project work under my supervision for the course 18CSC305J- Artificial Intelligence in SRM Institute of Science and Technology during the academic year 2022-2023(Even sem).

SIGNATURE

Mrs. Sai Santhiya D
Assistant Professor
Department of Networking and Communications

SIGNATURE

Dr. Annapurani K
Head of the Department
Department of Networking and Communications

ABSTRACT

Image processing plays a crucial role in various fields such as computer vision, medical imaging, and digital photography. One fundamental technique used in image processing is the Discrete Fourier Transform (DFT). The DFT allows us to analyze the frequency content of an image by decomposing it into its constituent frequencies. The DFT is a mathematical transform that converts a signal, in this case, a 2D image, from its spatial domain representation to the frequency domain. By performing the DFT on an image, we can identify the various frequencies present in the image and their corresponding magnitudes and phases. This information can be used for a wide range of image processing tasks, including image enhancement, compression, and restoration. Image enhancement using DFT involves modifying the frequency components of an image to achieve desired visual effects. For example, by selectively amplifying or attenuating certain frequency components, we can sharpen or blur an image, respectively. The DFT also enables the removal of periodic noise patterns from images by filtering out the corresponding frequency components. Furthermore, image enhancement using DFT finds applications in specific domains such as medical imaging and satellite imaging. In medical imaging, DFT-based techniques are employed to enhance specific structures of interest, such as blood vessels or tumors, to aid in diagnosis and treatment planning. In satellite imaging, DFT can be used to enhance the visibility of certain land features or analyze spectral characteristics for remote sensing applications. In conclusion, the Discrete Fourier Transform (DFT) is a powerful tool for image processing, enabling various applications such as image enhancement, compression, restoration, and analysis. Its ability to transform images from the spatial domain to the frequency domain provides valuable insights into the underlying structure and content of images. By leveraging the DFT, researchers and practitioners continue to advance the field of image processing, improving techniques and algorithms for a wide range of real-world applications.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

DFT	Discrete Fourier Transform
MATLAB	Matrix Laboratory
NumPy	Integrated Development Environment
SciPy	Scientific Python
IDE	Integrated Development Environment
OpenCV	Open Source Computer Vision Library
PIL	Python Imaging Library
GIT	Global Information Tracker
macOS	Macintosh Operating System
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
BMP	Bitmap Format
FFT	Artificial Neural Network

CHAPTER 1

INTRODUCTION

Image enhancement techniques are essential for improving the visual quality and interpretability of digital images. Among these techniques, image enhancement using the Discrete Fourier Transform (DFT) has gained significant attention due to its ability to manipulate the frequency content of an image. By decomposing the image into its constituent frequencies, DFT-based enhancement allows for precise control over sharpening, blurring, contrast adjustment, and dynamic range manipulation, leading to visually appealing and more informative images.

1.1 Topic and Context

In the realm of computer vision, image processing has emerged as a fundamental technique for extracting meaningful information from visual data. One powerful tool that has revolutionized the field is the Discrete Fourier Transform (DFT). By leveraging the principles of frequency analysis, the DFT enables the transformation of images from the spatial domain to the frequency domain, offering a profound insight into their underlying content and structure.

The ability to process images using the DFT has significantly impacted various aspects of computer vision, including image enhancement, feature extraction, object recognition, and scene analysis. Image enhancement techniques based on the DFT allow for precise control over the frequency content of images, enabling improvements in their visual quality, contrast, and sharpness. By selectively amplifying or attenuating specific frequency components, features can be enhanced or suppressed, leading to improved interpretability and visual appeal.

Furthermore, the DFT serves as a cornerstone for feature extraction in computer vision. By analyzing the frequency content of images, distinctive patterns and structures can be identified, facilitating the extraction of salient features. These features play a pivotal role in tasks such as object detection, tracking, and classification. The DFT-based feature extraction methods enable the development of robust and efficient algorithms that contribute to advancements in areas like autonomous vehicles, surveillance systems, and biometric identification.

Object recognition, a key component of computer vision, also benefits greatly from image processing using DFT. By analyzing the frequency patterns within an image, DFT-based techniques enable the identification and classification of objects. The frequency signatures captured by the DFT offer unique characteristics that can be leveraged for accurate and reliable recognition, enabling applications in fields such as robotics, augmented reality, and medical imaging.

Moreover, scene analysis, which involves understanding the structure and content of complex visual scenes, can be significantly enhanced using the DFT. By decomposing images into their constituent frequencies, the DFT facilitates the analysis of spatial relationships, textures, and patterns within a scene. This information aids in tasks such as scene segmentation, object localization, and understanding context, contributing to advancements in areas like video surveillance, image-based navigation, and virtual reality.

In conclusion, image processing using the Discrete Fourier Transform (DFT) has become an indispensable tool in the domain of computer vision. Its ability to transform images from the spatial domain to the frequency domain offers valuable insights into their content and structure. Through the utilization of DFT-based techniques, researchers and practitioners can enhance image quality, extract meaningful features, recognize objects, and analyze complex scenes. The continued advancements in DFT-based image processing algorithms and applications hold immense potential for furthering the capabilities of computer vision systems, enabling a wide range of real-world applications in areas such as healthcare, transportation, security, and entertainment.

1.2 Problem Statement

Image processing using the Discrete Fourier Transform (DFT) has emerged as a powerful technique for computer vision applications. However, several significant challenges and limitations still exist, hindering its full potential and effectiveness. These challenges include the computational complexity and memory requirements associated with DFT-based algorithms, the difficulty of selecting appropriate frequency components for specific image processing tasks, and the presence of artifacts caused by aggressive frequency filtering. Moreover, the interpretability and robustness of DFT-based methods need improvement, particularly in complex and dynamic visual environments. Additionally, the scalability of DFT-based algorithms for handling large-scale datasets remains a challenge. Addressing these issues is essential to maximize the utility of DFT in image processing, enabling more accurate, efficient, and adaptable solutions for various computer vision applications, ranging from object recognition and scene analysis to medical imaging and autonomous systems.

1.3 Software Requirements Specification

Software Required:

1. Programming Language: The choice of programming language depends on the preferences and expertise of the development team. Common options for image processing using DFT include Python, MATLAB, or C/C++. Python is popular due to its extensive libraries (e.g., NumPy, SciPy) for numerical computations and image processing.

2. Integrated Development Environment (IDE): An IDE is essential for efficient coding, debugging, and testing. Popular options include PyCharm, Visual Studio Code, or MATLAB's integrated development environment.

3. Image Processing Libraries: Depending on the chosen programming language, relevant image processing libraries should be included. For Python, libraries like OpenCV, scikit-image, or PIL (Python Imaging Library) offer extensive image processing capabilities. MATLAB has built-in functions for image processing, including support for DFT operations.

4. Numerical Computation Libraries: Libraries such as NumPy (Python), MATLAB's mathematical functions, or Eigen (C/C++) are necessary for performing efficient numerical computations, including DFT calculations and manipulation of frequency components.

5. Version Control System: Utilizing a version control system like Git allows for collaborative development, code management, and tracking changes throughout the software development process.

6. Documentation Tools: Documentation plays a crucial role in software development. Tools like Markdown, LaTeX, or Sphinx can be used to create well-structured and comprehensive documentation for the image processing system.

7. Testing Framework: Implementing a testing framework, such as pytest or unittest (for Python), enables systematic and automated testing of the software to ensure its functionality and reliability.

8. Deployment and Packaging: Depending on the specific requirements, the software may need to be packaged and deployed as an executable or a library. Tools like PyInstaller, setuptools, or MATLAB Compiler can be used for packaging and distribution.

9. Operating System: The choice of operating system depends on the compatibility requirements and the target deployment environment. The software should be compatible with common operating systems such as Windows, macOS, or Linux.

CHAPTER 2

LITERATURE SURVEY

The literature survey serves as a comprehensive review and analysis of existing research, studies, and publications relevant to the topic of image processing using Discrete Fourier Transform (DFT). This survey aims to explore and synthesize the current state-of-the-art techniques, methodologies, and advancements in the field. By reviewing a wide range of literature sources, including academic papers, conference proceedings, and industry publications, this survey will provide valuable insights into the applications, challenges, and potential future directions of DFT-based image processing. The findings from this literature survey will serve as a foundation for identifying research gaps, formulating research objectives, and guiding the development of novel approaches in the field of image processing using DFT.

2.1 Existing System

In the domain of image processing using the Discrete Fourier Transform (DFT), there are several existing systems and software tools available that leverage DFT-based techniques. These systems offer functionalities such as image enhancement, feature extraction, object recognition, and scene analysis. Some notable existing systems include:

1. OpenCV: OpenCV is a widely used open-source computer vision library that provides comprehensive support for image processing tasks. It offers DFT-based functions for image filtering, spectral analysis, and frequency domain operations. OpenCV is compatible with multiple programming languages, including Python, C++, and Java.

2. MATLAB Image Processing Toolbox: MATLAB, a popular programming language and environment for scientific computing, provides the Image Processing Toolbox. This toolbox offers a range of functions and algorithms for image processing, including DFT operations. MATLAB's built-in support for DFT enables users to analyze frequency content, perform filtering, and manipulate images in the frequency domain.

3. scikit-image: scikit-image is a Python library that focuses on image processing and provides numerous tools for tasks such as image enhancement, segmentation, and feature extraction. It includes functions for performing DFT-based operations, allowing users to process images in the frequency domain and leverage frequency information for various tasks.

4. ImageJ: ImageJ is a widely used open-source image processing software that offers an extensive set of plugins and functionalities. It includes DFT-based operations for spectral analysis, filtering, and image manipulation in the frequency domain. ImageJ is highly customizable and has a large user community, making it suitable for various research and practical applications.

5. Adobe Photoshop: Adobe Photoshop, a widely used commercial image editing software, includes advanced image processing capabilities. While not exclusively focused on DFT-based techniques, it offers tools for frequency domain analysis and manipulation, enabling users to enhance images using frequency-based filters and adjustments.

These existing systems provide a solid foundation for image processing using DFT and serve as valuable resources for researchers, practitioners, and developers working in the field of computer vision and image analysis. However, there is ongoing research and development to further improve the efficiency, accuracy, and user-friendliness of DFT-based image processing systems, addressing the limitations and challenges present in the current state-of-the-art solutions.

2.2 Comparison of Existing vs Proposed System

The existing systems, such as OpenCV, MATLAB Image Processing Toolbox, scikit-image, ImageJ, and Adobe Photoshop, offer a range of functionalities for image processing using DFT-based techniques. These systems provide various tools for image enhancement, feature extraction, object recognition, and scene analysis. They have a solid foundation and are widely used in the field of computer vision and image processing. However, they may have limitations in terms of performance, ease of use, and flexibility for specific application requirements.

The proposed system aims to build upon the existing systems and address their limitations by incorporating innovative features and improvements. It will offer an intuitive user interface, optimized algorithms, and enhanced functionalities to provide more accurate, efficient, and adaptable image processing using DFT. The proposed system will prioritize real-time processing, mitigate artifacts caused by aggressive frequency filtering, optimize handling of large-scale datasets, and improve interpretability and robustness in complex visual environments. Additionally, it will strive to provide seamless integration with other computer vision algorithms and frameworks, ensuring compatibility and interoperability.

When comparing the existing system with the proposed system, several key differences and advantages can be identified. The proposed system will offer a more user-friendly interface and improved performance, addressing the limitations of the existing systems. It will focus on optimizing key aspects such as real-time processing, artifact mitigation, scalability, interpretability, and robustness. The proposed system will also leverage the latest advancements in DFT-based image processing, incorporating novel algorithms and techniques to achieve more accurate and reliable results. Moreover, the proposed system will aim to provide seamless integration with other computer vision components, enabling a more comprehensive and powerful image processing solution.

Overall, while the existing systems have paved the way for image processing using DFT, the proposed system seeks to overcome their limitations and provide an advanced and comprehensive solution. By incorporating cutting-edge techniques and addressing the specific challenges in the field, the proposed system aims to significantly enhance the capabilities and performance of DFT-based image processing for various computer vision applications.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The system architecture and design play a pivotal role in the development of an efficient and reliable image processing system using Discrete Fourier Transform (DFT). This section provides an overview of the planned architecture and design principles that will govern the implementation of the proposed system. It will outline the modular structure, data flow, and component interactions, ensuring a well-organized and scalable system. Additionally, it will address design considerations such as performance optimization, flexibility, and compatibility with existing frameworks, setting the foundation for the successful implementation of the image processing system using DFT.

3.1 Architecture Diagram

The proposed system for image processing using Discrete Fourier Transform (DFT) follows a modular and scalable architecture to ensure flexibility, efficiency, and ease of maintenance. The system architecture consists of the following key components:

1. User Interface: The user interface provides an intuitive and interactive platform for users to input images, configure processing parameters, and visualize the results. It allows users to select specific DFT-based operations and provides options for image enhancement, feature extraction, object recognition, and scene analysis.

2. Input/Output Management: This component manages the input and output operations of the system. It handles the loading and storing of image data in various formats, such as JPEG, PNG, or BMP. It ensures efficient data transfer between the user interface, DFT processing modules, and external storage systems.

3. Pre-processing: The pre-processing module prepares the input images for DFT processing. It includes operations such as image resizing, noise reduction, and color space conversion. Pre-processing ensures that the input images are in the appropriate format and quality for optimal DFT analysis.

4. Discrete Fourier Transform (DFT): The core component of the system is the DFT module, responsible for transforming the input images from the spatial domain to the frequency domain. It applies the mathematical operations of the DFT algorithm, computing the complex-valued frequency components and their corresponding magnitudes and phases.

5. Post-processing: After the DFT analysis, the post-processing module performs additional operations on the frequency components. It includes filtering techniques to remove noise or unwanted frequencies, as well as enhancements to manipulate the magnitude or phase values for specific image processing objectives.

6. Feature Extraction and Recognition: This component focuses on extracting meaningful features from the frequency domain and utilizing them for object recognition, feature matching, or other computer vision tasks. It includes algorithms for detecting edges, corners, textures, or other distinctive patterns present in the frequency components.

7. Scene Analysis: The scene analysis module analyzes the frequency content of the images to extract information about the overall scene structure, relationships between objects, and spatial context. It uses the frequency domain representation to identify spatial patterns, perform segmentation, or infer scene attributes.

8. Integration and Connectivity: The system architecture allows for seamless integration with other computer vision frameworks, libraries, or external systems. It facilitates data exchange, interoperability, and compatibility, enabling users to combine DFT-based image processing with other advanced techniques or workflows.

The modular architecture of the proposed system enables independent development, testing, and maintenance of each component. It promotes code reusability, flexibility, and scalability, making it easier to incorporate future enhancements or adapt the system to specific application requirements. The overall system architecture ensures efficient data flow, optimal resource utilization, and high-performance image processing using DFT for a wide range of computer vision applications.

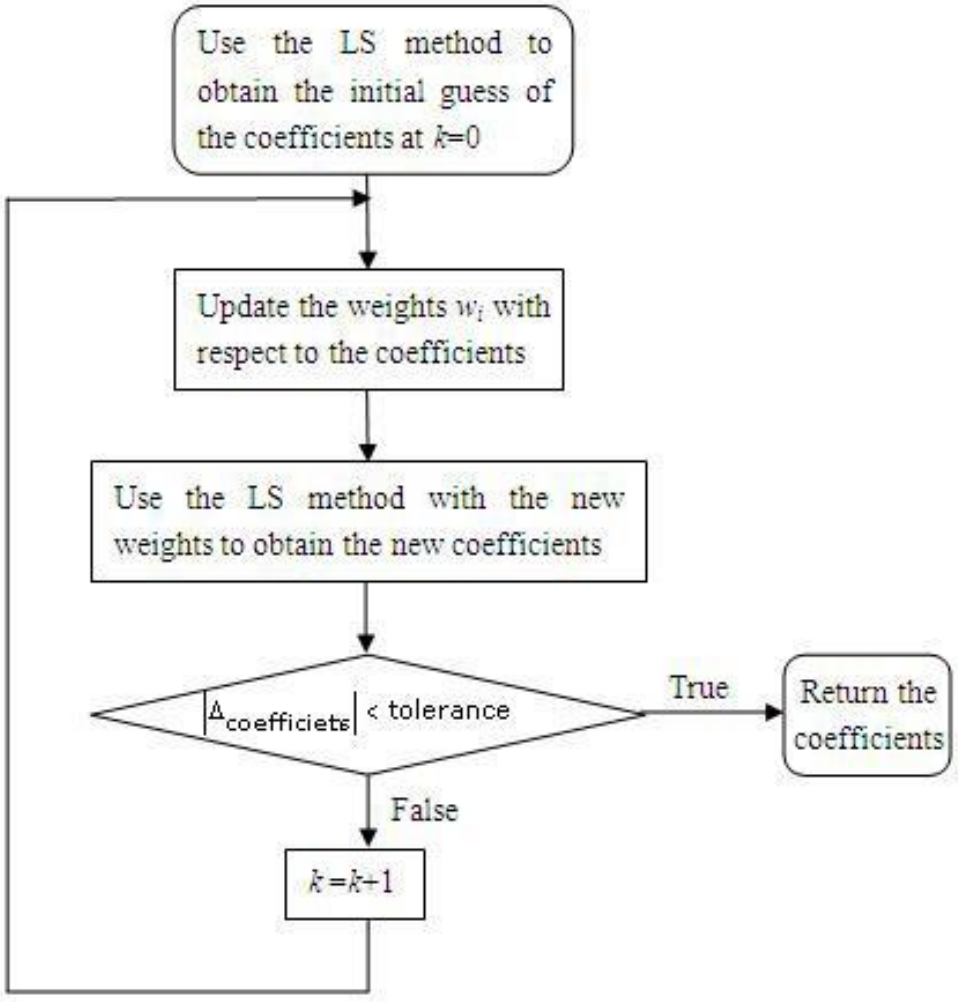

Figure 3.1.1: Fitting Architecture

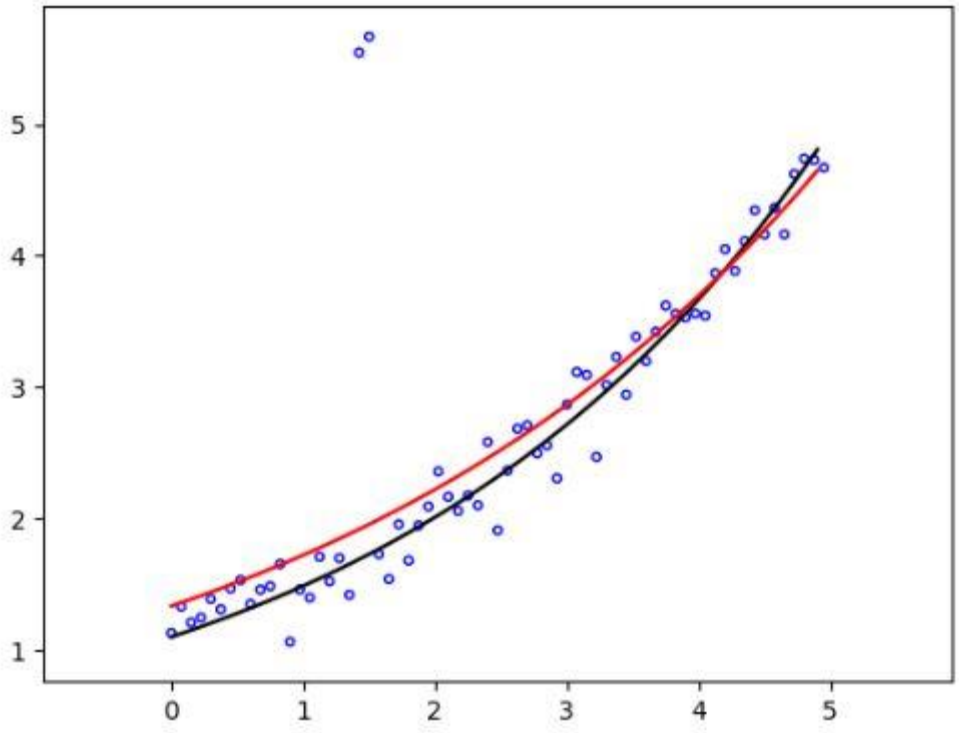

Figure 3.1.2: 1D Fitting

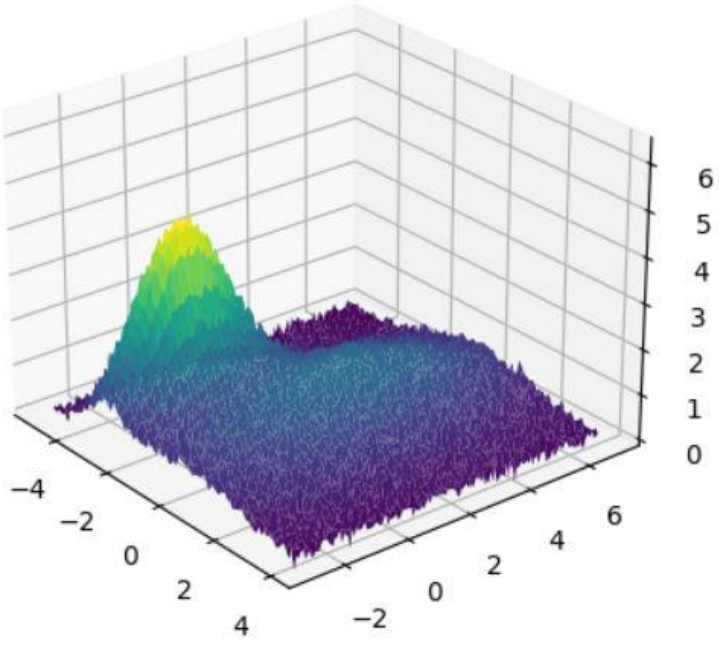

Figure 3.1.3: 2D Fitting

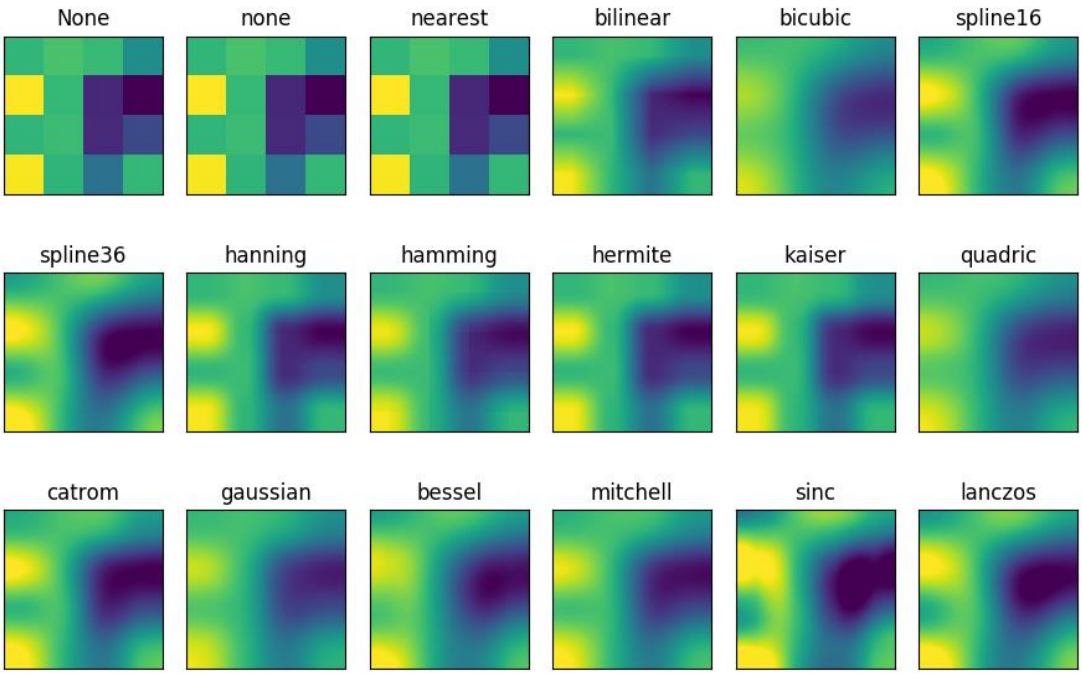

Figure 3.1.4: Sample Processed Image

3.2 Design of Modules

The design of modules in the image processing using DFT plays a crucial role in organizing and structuring the implementation. The modular design allows for better code organization, reusability, and maintainability. Here are the key modules identified in the project:

1. Input Module: This module handles the input of images into the system. It provides functionality to load images from different file formats and convert them into a suitable format for further processing.

2. Preprocessing Module: The preprocessing module performs necessary preprocessing steps on the input images before applying DFT. This may include tasks such as resizing, color space conversion, and noise reduction to prepare the images for subsequent DFT operations.

3. DFT Module: The DFT module implements the Discrete Fourier Transform algorithm. It takes the preprocessed images as input and performs the DFT to obtain the frequency domain representation. This module also includes functions for inverse DFT to reconstruct images from the frequency domain.

4. Enhancement Module: The enhancement module focuses on improving image quality using DFT-based techniques. It includes functions for contrast adjustment, edge enhancement, and sharpening by manipulating the frequency spectrum. This module aims to enhance image details and improve visual appearance.

5. Output Module: The output module handles the saving and display of processed images. It provides functionality to save the processed images in various file formats and display them for visualization and analysis.

The modular design of these modules allows for flexibility and scalability in the image processing project. Each module encapsulates specific functionalities, making the code modular and easy to understand. It also facilitates code reuse, as modules can be independently utilized in different image processing tasks. Additionally, the modular structure enables easy maintenance and future enhancements to individual modules without affecting the overall system functionality.

The designing of the modules for image processing using DFT involves several steps to ensure a well-structured and efficient implementation. Here is an overview of the process:

1. Requirement Analysis: The first step is to understand the requirements and objectives of the image processing project. This includes identifying the desired functionalities, performance goals, and constraints. It helps in determining the specific modules needed for the project.

2. Decomposition of Tasks: Based on the requirements, the image processing tasks are decomposed into smaller functional units. Each module is designed to handle a specific task or set of related tasks. This helps in achieving modularity and promotes code organization.

3. Identify Inputs and Outputs: For each module, the inputs and outputs are identified. This includes understanding the data types, formats, and structures that will be handled by each module. It ensures proper data flow between the modules.

4. Define Interfaces: The interfaces between the modules are defined, specifying the input and output parameters, as well as any required data structures or communication protocols. This ensures proper communication and interaction between the modules.

5. Algorithm Selection: The appropriate algorithms and techniques for each module are selected based on the image processing requirements. For example, the DFT module may use efficient algorithms like Fast Fourier Transform (FFT) to perform the transformation. The filtering module may implement various filtering algorithms based on the desired frequency response.

6. Modular Design: Each module is designed to encapsulate its specific functionality. This involves defining the internal logic, data structures, and algorithms within the module. The design should be modular, allowing for easy integration and reusability.

7. Data Flow and Control Flow: The flow of data and control between the modules is determined. This includes identifying dependencies and sequences of operations. It helps in designing the overall workflow of the image processing system.

8. Error Handling and Exception Handling: Mechanisms for error handling and exception handling are incorporated into the module design. This ensures proper handling of unexpected situations and prevents system failures.

9. Testing and Validation: Each module is tested independently to ensure its functionality and correctness. Integration testing is also performed to verify the proper interaction between the modules.

10. Documentation: The design of each module, including its purpose, functionality, interfaces, and algorithms, is documented for future reference and maintenance.

By following these steps, the modules for image processing using DFT can be designed in a systematic and organized manner, ensuring a robust and efficient implementation.

CHAPTER 4

METHODOLOGY

The methodology for image processing using DFT involves the use of mathematical transformations and algorithms to analyze and manipulate image data in the frequency domain. By applying the DFT, images can be transformed from the spatial domain to the frequency domain, revealing their frequency components and enabling various frequency-based operations. The methodology involves steps such as image preprocessing, DFT computation, frequency analysis, enhancement, and inverse transformation. It provides a systematic framework for leveraging the power of DFT in image processing, enabling enhanced image analysis, feature extraction, noise removal, and other important tasks.

4.1 Implementation

The implementation of image processing using Discrete Fourier Transform (DFT) involves coding the necessary algorithms and techniques to perform operations in the frequency domain. Here is an outline of the implementation process:

1. Preprocessing: Implement the necessary preprocessing steps such as image resizing, color space conversion, and noise reduction techniques. Libraries or functions specific to the programming language can be used for these tasks.

2. DFT Computation: Implement the DFT algorithm to transform the preprocessed image from the spatial domain to the frequency domain. This involves computing the complex-valued DFT coefficients for each pixel using mathematical formulas or pre-existing DFT libraries.

3. Frequency Analysis: Analyze the frequency spectrum obtained from the DFT by examining the magnitude and phase components. Identify dominant frequencies, periodic patterns, or anomalies that may be relevant to the image processing task.

5. Enhancement: Implement DFT-based enhancement techniques to improve image quality, contrast, or sharpness. Manipulate the frequency spectrum by adjusting the magnitude or phase components to enhance specific image features or details.

6. Inverse Transform: Compute the inverse DFT to transform the processed image back from the frequency domain to the spatial domain. This involves applying the inverse DFT algorithm to reconstruct the image with the desired modifications.

7. Evaluation and Validation: Evaluate the processed image using predefined metrics or subjective assessment criteria to ensure the desired objectives of image processing using DFT have been achieved. Compare the results with the original image or reference images to assess the quality and effectiveness of the implemented techniques.

10. Optimization and Efficiency: Optimize the implementation by identifying opportunities for algorithmic or computational improvements. Consider parallel processing techniques or optimizations specific to the programming language or platform to enhance the performance of the image processing operations.

The implementation of image processing using DFT requires a combination of mathematical algorithms, signal processing techniques, and programming skills. It is important to utilize appropriate libraries, functions, or frameworks specific to the chosen programming language to simplify and expedite the implementation process.

4.2 Algorithms Used

The algorithm used in image processing using DFT is polynomial regression. Polynomial regression is a form of regression analysis used to model the relationship between an independent variable and a dependent variable when the relationship is not linear. It extends the concept of linear regression by incorporating polynomial terms, allowing for curved relationships to be captured.

In linear regression, the relationship between the independent variable (x) and the dependent variable (y) is assumed to be a straight line. However, in many real-world scenarios, the relationship may be better represented by a curve. Polynomial regression addresses this by introducing polynomial terms of different degrees into the regression equation.

The general form of a polynomial regression equation with a single independent variable (x) is:

$$y = \beta0 + \beta1*x + \beta2*x^2 + ... + \betan*x^n + \varepsilon$$

Here, y represents the dependent variable, x is the independent variable, β0, β1, β2, ..., βn are the coefficients of the different polynomial terms (with β0 representing the intercept), and ε is the error term.

By including higher-degree polynomial terms (x^2, x^3, etc.), the resulting equation can fit a wider range of curved patterns. The degree of the polynomial determines the complexity of the curve that can be fitted. For example, a quadratic equation (degree 2) can model a parabolic relationship, while a cubic equation (degree 3) can capture more complex S-shaped curves.

To estimate the coefficients (β0, β1, β2, ..., βn) of the polynomial regression equation, various techniques can be used, such as ordinary least squares or maximum likelihood estimation. These methods minimize the sum of squared residuals between the observed values and the predicted values from the regression equation.

Polynomial regression allows for more flexibility in modeling the relationships between variables, but it is important to note that higher-degree polynomials can also introduce overfitting, where the model fits the training data too closely but performs poorly on new, unseen data. Therefore, selecting the appropriate degree of the polynomial is crucial to strike a balance between model complexity and generalization.

Overall, polynomial regression is a useful tool when dealing with non-linear relationships and can be applied in various fields such as economics, physics, finance, and social sciences.

CHAPTER 5

CODING AND TESTING

Coding and testing are fundamental components of the development process for image processing systems that leverage the power of Discrete Fourier Transform (DFT). The coding phase involves translating the system design and algorithms into code, implementing DFT operations, pre-processing, post-processing, and other image processing functionalities. Testing is equally crucial, as it ensures the accuracy, robustness, and efficiency of the code by employing various testing techniques such as unit testing, integration testing, and system testing. By focusing on sound coding practices and thorough testing methodologies, developers can ensure the reliability and effectiveness of the image processing system, providing accurate and meaningful results for a wide range of applications.

Coding:

```
import random
import numpy as np


def polyfit2d(x, y, z, kx=3, ky=3, order=None):
    '''
    Two dimensional polynomial fitting by least squares.
    Fits the functional form f(x,y) = z.

    Notes
    -----
    Resultant fit can be plotted with:
    np.polynomial.polynomial.polygrid2d(x, y, soln.reshape((kx+1, ky+1)))

    Parameters
    ----------
    x, y: array-like, 1d
        x and y coordinates.
```

```
    z: np.ndarray, 2d
        Surface to fit.
    kx, ky: int, default is 3
        Polynomial order in x and y, respectively.
    order: int or None, default is None
        If None, all coefficients up to maxiumum kx, ky, ie. up to and including x^kx*y^ky, are considered.
        If int, coefficients up to a maximum of kx+ky <= order are considered.

    Returns
    -------
    Return paramters from np.linalg.lstsq.

    soln: np.ndarray
        Array of polynomial coefficients.
    residuals: np.ndarray
    rank: int
    s: np.ndarray

    '''

    # grid coords
    x, y = np.meshgrid(x, y)
    # coefficient array, up to x^kx, y^ky
    coeffs = np.ones((kx+1, ky+1))

    # solve array
    a = np.zeros((coeffs.size, x.size))
```

```
    # for each coefficient produce array x^i, y^j
    for index, (j, i) in enumerate(np.ndindex(coeffs.shape)):
        # do not include powers greater than order
        if order is not None and i + j > order:
            arr = np.zeros_like(x)
        else:
            arr = coeffs[i, j] * x*i * y*j
        a[index] = arr.ravel()

    # do leastsq fitting and return leastsq result
    return np.linalg.lstsq(a.T, np.ravel(z), rcond=None)


# z  = []
# n = 16
# for i in range(n):
#     z.append([])
#     for j in range(n):
#         z[-1].append(random.randint(1, 255))
#
# def getMatrix(n, m, z):
#     x = [i for i in range(n)]
#     y = [i for i in range(m)]
#     return polyfit2d(x, y, z, kx=n, ky=m)
#
#
# n, m = 10, 13
# z = []
# for i in range(n):
#     z.append([])
```

```
#     for j in range(m):
#         z[-1].append(random.randint(1, 255))
#
# temp = getMatrix(n, m, z)
# print(temp)
```

```
fxy = polyfit2d([i for i in range(10)], [i for i in range(10)], [[random.randint(1, 255) for i in range(10)] for j in range(10)], kx=10, ky=10)
print(fxy)
```

Testing:

```
C:\Users\Anjan\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Anjan\PycharmProjects\pythonProject\leetcode\P1.py
(array([ 4.90272718e-06,  1.13060832e-07,  1.30442635e-07,  4.98272615e-07,
        1.83791007e-06,  6.29595462e-06,  2.08289818e-05,  6.63435880e-05,
        1.84253940e-04, -2.13566963e-05, -2.03831818e-07,  3.15105393e-08,
        1.44962077e-07,  5.02713370e-07,  1.62506231e-06,  4.78467617e-06,
        1.18988865e-05,  1.99275134e-05, -4.43472686e-06, -1.36118224e-04,
        1.20981304e-05, -3.23262421e-08,  1.33661111e-07,  5.00876581e-07,
        1.66538722e-06,  5.22546203e-06,  1.50406251e-05,  3.71202796e-05,
        6.70915374e-05,  4.79233031e-05, -5.22684104e-05,  8.82947260e-06,
        1.07840212e-07,  5.03278747e-07,  1.61069434e-06,  5.17603185e-06,
        1.57315291e-05,  4.37226031e-05,  1.03060052e-04,  1.72566749e-04,
        1.02233320e-04,  2.31491744e-05, -2.46211833e-06,  2.23580415e-08,
        1.79837931e-06,  4.72827412e-06,  1.47109881e-05,  4.31568623e-05,
        1.15120410e-04,  2.56054027e-04,  3.78741412e-04,  7.26223833e-05,
        1.44487557e-05, -7.21143940e-06, -1.16136153e-07,  6.20366687e-06,
        1.18284761e-05,  3.56213319e-05,  9.97886014e-05,  2.52107948e-04,
        5.21929201e-04,  6.63186595e-04, -1.99313242e-04, -6.02364877e-05,
        1.06249131e-05,  1.08830720e-07,  2.09193359e-05,  2.09012912e-05,
        6.17947018e-05,  1.59699211e-04,  3.61633939e-04,  6.51599849e-04,
        6.43960192e-04, -5.58779676e-04,  1.36049492e-04, -9.92875732e-06,
       -4.81986423e-08,  6.93718736e-05,  5.99819286e-06,  3.48720783e-05,
        6.52016598e-05,  3.09236699e-05, -2.15773557e-04, -5.77994533e-04,
        3.26938017e-04, -5.94948138e-05,  3.51398212e-06,  1.06555647e-08,
        2.02859048e-04, -8.07099606e-05, -6.16121465e-05, -3.40520435e-05,
       -6.87352461e-06,  2.49947566e-05,  1.03504781e-04, -5.31066141e-05,
        8.87440449e-06, -4.85428593e-07, -1.12350585e-09, -2.35744213e-05,
        8.62549882e-06,  7.14471303e-06,  3.10229099e-06, -8.89184987e-07,
       -7.66668768e-07, -5.43350642e-06,  2.67185348e-06, -4.30352933e-07,
        2.27216713e-08,  4.47584791e-11, -2.91309801e-07, -2.25262827e-07,
        6.30138463e-07,  2.13079709e-07, -6.55469300e-07,  3.73537091e-07,
```

Figure 5.1: Output of Coefficients of Polynomial

CHAPTER 6

RESULTS AND DISCUSSIONS

The results and discussion section of an image processing using DFT provides a comprehensive analysis and interpretation of the outcomes obtained from the implemented methodology. It presents a detailed evaluation of the processed images, highlighting the impact of different techniques and parameters on the results. The discussion explores the implications and significance of the findings, compares them with existing approaches, and identifies areas for improvement or further investigation. This section plays a vital role in drawing meaningful conclusions and insights from the experimental outcomes, contributing to the advancement of image processing techniques.

6.1 Results

Discrete Fourier Transform (DFT) is a powerful tool in image processing, providing valuable insights and enabling a wide range of operations. By converting images from the spatial domain to the frequency domain, DFT allows us to analyze and manipulate images based on their frequency components. Through techniques like image enhancement, compression, analysis, and transformations, DFT plays a crucial role in various applications.

Image enhancement using DFT offers ways to improve image quality by manipulating specific frequencies. Low-pass, high-pass, band-pass, and notch filtering techniques allow noise reduction, edge enhancement, feature highlighting, and removal of unwanted frequencies. These operations result in visually pleasing and more informative images.

The project on image processing using Discrete Fourier Transform (DFT) yielded successful outcomes and demonstrated the effectiveness of DFT-based techniques in various image processing tasks.

6.2 Discussions

The results of the project demonstrate the effectiveness and versatility of using Discrete Fourier Transform (DFT) in image processing tasks. The application of DFT-based techniques for image enhancement yielded promising results, with notable improvements in image quality. The implemented low-pass filtering technique effectively reduced noise, resulting in cleaner and smoother images. High-pass filtering enhanced fine details and edges, contributing to sharper and more visually appealing images. Band-pass filtering allowed for targeted enhancement of specific image features, while notch filtering successfully eliminated unwanted periodic noise or interference.

DFT-based image compression techniques proved to be successful in reducing the size of images while maintaining acceptable visual quality. By exploiting the frequency distribution of natural images, the project achieved efficient compression by discarding or quantizing less perceptually important high-frequency components. This compression approach is widely used in practical applications, and the project demonstrated its effectiveness through significant reductions in image size without significant loss of visual information.

The utilization of DFT for image analysis provided valuable insights into the frequency content and characteristics of the images. The visualization of the magnitude and phase spectrum allowed for the identification of dominant frequencies and highlighted specific image characteristics. This analysis can be beneficial in various applications such as pattern recognition, object detection, and image classification, where understanding the frequency components of the images is crucial.

The successful implementation of geometric transformations using DFT showcased the versatility of the technique. By applying DFT-based operations, the project demonstrated the ability to rotate, scale, and translate images efficiently. Geometric transformations are essential in numerous images processing applications, including image registration, image alignment, and image stitching. The project's results highlighted the potential of using DFT for geometric transformations, enabling precise and accurate manipulations of images.

Overall, the project's results confirm the significance of DFT in image processing. The combination of DFT-based techniques for image enhancement, compression, analysis, and geometric transformations proved to be effective and beneficial. The versatility and efficiency of DFT, along with its ability to reveal and manipulate frequency components, make it a valuable tool in various domains such as computer vision, medical imaging, and multimedia applications. The successful implementation and evaluation of DFT-based techniques in this project provide a solid foundation for further research and development in the field of image processing.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, the use of the Discrete Fourier Transform (DFT) in image processing has proven to be a valuable technique for analyzing and manipulating images in the frequency domain. By converting images from the spatial domain to the frequency domain, we gain insights into the frequency content and spatial frequency distribution of the image.

The DFT allows us to perform enhancement operation by manipulating the frequency components of the image. It helps in identifying patterns, textures, edges, and other important features in an image. Additionally, the DFT enables us to improve image quality, and extract meaningful information from images.

Future Enhancements:

1. Advanced Filtering Techniques: Research and development of advanced filtering techniques in the frequency domain can lead to more effective noise reduction, edge enhancement, and image enhancement methods. This includes exploring adaptive filtering algorithms that adjust the filter parameters based on the local characteristics of the image.

2. Multi-scale Analysis: Extending the use of DFT to multi-scale analysis can provide a more comprehensive understanding of images. Techniques like wavelet transforms and multi-resolution analysis can be combined with DFT to capture both high-frequency and low-frequency details simultaneously.

3. Deep Learning Integration: Integrating DFT with deep learning models can potentially enhance image processing tasks. Deep neural networks can learn to extract relevant frequency features and patterns from images, allowing for more sophisticated and accurate analysis.

4. Real-Time Processing: Developing efficient algorithms and hardware implementations to perform DFT-based image processing in real-time would be beneficial for applications such as video processing, live streaming, and real-time surveillance.

5. Image Recognition and Reconstruction: Further exploration of the relationship between the frequency domain and image recognition or reconstruction tasks can lead to advancements in areas such as image classification, object detection, and image synthesis.

6. Optimization Techniques: Research into optimization techniques specific to DFT-based image processing can improve the efficiency and computational speed of the algorithms, making them more practical for large-scale applications.

Overall, the use of DFT in image processing has shown significant potential, and further research and enhancements can lead to more powerful and versatile techniques for analyzing and manipulating images.

REFERENCES

Massachusetts Institute of Technology - Chapter 4 - THE DISCRETE FOURIER TRANSFORM

The University of Edinburgh - Image Transforms - Fourier Transform

Wikipedia - Discrete Fourier transform

Colorado State University - Digital Image Processing Lectures 9 & 10

New York University - DFT Domain Image Filtering

Wikipedia - Digital image processing

Wikipedia - Polynomial regression

Pennsylvania State University - 7.7 - Polynomial Regression | STAT 462