**Chapter 1**

**Introduction**

**1.1 Existing System**

The existing digital image management and protection systems rely heavily on traditional methods like copyright notices, digital rights management (DRM), and visible watermarks. While these methods provide a basic level of protection, they have significant limitations:

**Digital Rights Management (DRM):** DRM technologies involve encrypting digital content and controlling access through licensing. While DRM is effective in preventing unauthorized access, it can be inconvenient for legitimate users and does not protect the content once it is accessed legally. For instance, an image can be easily copied and redistributed after it has been decrypted.

**Visible Watermarks:** Visible watermarks, such as logos or text overlays, provide immediate visual recognition of ownership. However, they can detract from the aesthetic quality of the image and are often easy to remove or alter using basic image editing tools. This is particularly problematic for high value images, such as professional photography and digital art, where visual integrity is crucial.

**Software Tools:** Various software tools like Adobe Photoshop, GIMP, and specialized watermarking software offer features for embedding watermarks. These tools typically use simple overlay techniques, which are not robust against tampering. Additionally, they often do not provide comprehensive solutions for tracking and verifying image ownership across different platforms.

**1.2 Problem Statement**

Digital images are vulnerable to unauthorized use and distribution, posing significant challenges in maintaining intellectual property rights. Traditional methods like DRM and visible watermarks are often inadequate for robust protection. There is a critical need for a more sophisticated system that can embed ownership information directly into the images in a way that is both imperceptible to viewers and resilient against tampering. This project aims to explore and implement advanced digital watermarking techniques to address these challenges.

Digital watermarking involves embedding information into digital media such that it is difficult to remove or alter without affecting the quality of the media. The embedded information, known as a watermark, can be used for various purposes, including verifying the authenticity of the media, protecting copyright, and tracking distribution. The challenge lies in developing watermarking techniques that are robust, imperceptible, and capable of embedding a sufficient amount of information without degrading the media quality.

**1.3 Objectives**

The primary objectives of this project are:

**To study different techniques for watermarking digital images:** Conduct a thorough review of existing watermarking techniques, including spatial domain techniques, frequency domain techniques, and hybrid techniques. This review will identify the strengths and weaknesses of each method and help determine the most suitable techniques for various applications.

**To implement a watermarking system that is both robust and imperceptible:** Develop algorithms for embedding and extracting watermarks that are resilient to common attacks, such as compression, cropping, and noise addition. The system should ensure that the watermarks remain imperceptible to human viewers, preserving the visual quality of the images.

**To analyze the effectiveness of the implemented watermarking system:** Conduct experiments to evaluate the robustness, imperceptibility, and capacity of the watermarking system. Compare the results with existing techniques to assess the effectiveness of the proposed methods.

**To explore methods for removing watermarks and their implications:** Investigate techniques for detecting and removing watermarks, and analyze the legal and ethical implications of watermark removal. Develop strategies for enhancing the robustness of watermarks to counteract these techniques.

**1.4 Scope**

This project covers the following aspects:

**A detailed survey of existing watermarking techniques:** Review the current state of digital watermarking, including different techniques and their applications. Identify gaps in existing research and areas for further investigation.

**Implementation and analysis of watermarking algorithms:** Develop and test various watermarking algorithms, focusing on techniques that offer a good balance between robustness, imperceptibility, and capacity. Implement these algorithms using programming languages like Python or MATLAB, with the help of image processing libraries such as OpenCV and NumPy.

**Exploration of watermark removal techniques:** Investigate methods for detecting and removing watermarks, evaluate their effectiveness, and understand their implications. Study different types of attacks on watermarks and develop strategies to counteract them.

**Examination of legal and ethical considerations in digital watermarking:** Address legal and ethical issues related to digital watermarking, including copyright protection, digital rights management, and compliance with industry standards and best practices.

**Chapter 2**

**Literature Survey**

Digital watermarking has been extensively researched, resulting in various techniques developed over the years. This chapter reviews the literature on digital watermarking, covering different techniques, their applications, and the challenges they address.

**Visible Watermarking:** Techniques that overlay visible information onto images to indicate ownership. Common methods include placing a semi transparent logo or text over the image. While effective in asserting ownership, visible watermarks can be distracting and are susceptible to removal or alteration. Studies show that visible watermarks are often used in scenarios where immediate visual recognition of ownership is necessary, such as in stock photography and online image galleries.

**Invisible Watermarking:** Methods that embed information imperceptibly within an image's data. Invisible watermarks are designed to be undetectable to the human eye but can be extracted or detected using appropriate algorithms. These watermarks are often used for tracking and verification purposes. Research focuses on developing algorithms that can embed watermarks in a way that is robust to various attacks while maintaining the visual quality of the image.

**Spatial Domain Techniques:** Embedding watermarks by directly modifying the pixel values of an image. Techniques such as Least Significant Bit (LSB) modification fall under this category. Spatial domain techniques are generally simpler and faster but may be less robust against image processing operations like compression and filtering. Studies show that LSB embedding is effective for applications where simplicity and speed are more important than robustness.

**Frequency Domain Techniques:** Using transformations like Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT), and wavelet transforms to embed watermarks in the frequency domain. These techniques are generally more robust against image processing operations and attacks. For example, embedding a watermark in the DCT coefficients of an image can make it more resistant to JPEG compression. Research focuses on developing algorithms that can embed watermarks in a way that is robust to various attacks while maintaining the visual quality of the image.

**Robust Watermarking:** Techniques designed to withstand various attacks such as compression, cropping, and noise addition. Robust watermarks are critical for applications where the image is likely to undergo transformations or be subjected to tampering attempts. Studies show that robust watermarking techniques are effective for applications where the integrity of the watermark is more important than its imperceptibility.

The literature survey also covers various applications of digital watermarking, including copyright protection, authentication, and data hiding. Each application has its own set of requirements and challenges, and the choice of watermarking technique depends on the specific needs of the application.

**Chapter 3**

**Domain Analysis**

Digital watermarking can be applied in various domains, each with its own set of requirements and challenges. This chapter provides an in depth analysis of the different domains where digital watermarking is used and the factors that need to be considered in each domain.

**Photography:** Protecting the rights of photographers by embedding ownership information directly into their digital images. This helps prevent unauthorized use and ensures that photographers receive proper attribution for their work. The primary challenge in this domain is to develop watermarking techniques that are imperceptible and do not degrade the visual quality of the images. Additionally, the watermarks need to be robust against common image processing operations such as resizing, cropping, and compression.

**Publishing:** Preventing unauthorized reproduction of digital publications, such as e books and digital articles. Watermarking can help track the distribution of digital content and identify the source of unauthorized copies. The main challenge in this domain is to develop watermarking techniques that can embed a large amount of information without affecting the readability of the text. Additionally, the watermarks need to be robust against common text processing operations such as copying and pasting, printing, and scanning.

**Digital Media Distribution:** Ensuring that music, videos, and other media are not distributed illegally. Watermarking can be used to embed information about the purchaser or distributor, helping to track and prevent unauthorized sharing. The primary challenge in this domain is to develop watermarking techniques that are robust against common media processing operations such as compression, transcoding, and editing. Additionally, the watermarks need to be imperceptible and should not affect the quality of the media.

**Medical Imaging:** Protecting patient information and ensuring the integrity of medical images. Watermarking can be used to embed patient information and other metadata into medical images, helping to prevent unauthorized access and tampering. The main challenge in this domain is to develop watermarking techniques that are imperceptible and do not degrade the quality of the medical images. Additionally, the watermarks need to be robust against common image processing operations such as compression and noise addition.

**Geographic Information Systems (GIS**)**:** Protecting the integrity of digital maps and other geographic data. Watermarking can be used to embed ownership information and other metadata into digital maps, helping to prevent unauthorized use and ensure the accuracy of the data. The primary challenge in this domain is to develop watermarking techniques that are robust against common image processing operations such as compression, cropping, and filtering. Additionally, the watermarks need to be imperceptible and should not affect the quality of the maps.

**Scientific Data:** Ensuring the integrity and provenance of scientific data, including images and other digital assets. Watermarking can be used to embed information about the source and authenticity of the data, helping to prevent unauthorized use and tampering. The main challenge in this domain is to develop watermarking techniques that are imperceptible and do not degrade the quality of the data. Additionally, the watermarks need to be robust against common data processing operations such as compression and noise addition.

For each domain, the analysis includes a detailed examination of the specific requirements and challenges, along with a review of the existing watermarking techniques and their effectiveness in addressing these challenges.

**Chapter 4**

**Requirements Specification**

**4.1 Hardware Requirements**

**Computer Specifications:** A computer with at least 8GB RAM and a multi core processor is essential for processing high resolution images and implementing complex watermarking algorithms. A multi core processor helps in parallelizing tasks and speeding up the processing time. Additionally, a high resolution monitor is necessary for visually inspecting the quality of watermarked images.

**Storage Capacity:** A storage capacity of at least 500GB is required to store high resolution images and various versions of watermarked images generated during experimentation. This ensures that there is sufficient space for storing large datasets and for performing extensive testing and analysis.

**Graphics Processing Unit (GPU):** A GPU can significantly accelerate image processing tasks and is recommended for implementing and testing advanced watermarking algorithms. GPUs are particularly useful for deep learning based watermarking techniques and for handling large batches of images.

**4.2 Software Requirements**

**Operating System:** The system should be compatible with major operating systems, including Windows, macOS, and Linux. This ensures that the system can be used on a wide range of platforms and can leverage the strengths of each operating system.

**Programming Languages:** Python and MATLAB are the preferred programming languages for implementing watermarking algorithms. Python is widely used for image processing due to its extensive libraries and ease of use. MATLAB is also popular in academic and research settings for its powerful image processing capabilities.

**Libraries and Tools:** Essential libraries include OpenCV, NumPy, SciPy, and scikit image for image processing tasks. Additional tools such as TensorFlow or PyTorch may be used for implementing deep learning based watermarking techniques. These libraries provide a comprehensive set of functions for image manipulation, transformation, and analysis.

**4.3 Functional Requirements**

**Embedding Visible and Invisible Watermarks:** The system should provide options for embedding both visible and invisible watermarks, depending on the user's needs. This ensures flexibility and allows users to choose the appropriate watermarking technique for their specific application.

**Support for Different Watermarking Techniques:** The system should support both spatial domain and frequency domain techniques, allowing users to choose the most suitable method for their requirements. This ensures that the system can handle a wide range of applications and provides users with the flexibility to select the best technique for their needs.

**Customizable Watermark Information:** Users should be able to specify the watermark information, such as text or logo, and customize the appearance and embedding parameters of the watermark. This ensures that the watermarks are tailored to the specific needs of the user and provides a high level of customization.

**4.4 Non Functional Requirements**

**Efficiency :** The watermarking process should be efficient and not significantly increase image processing time. The system should be optimized to handle large batches of images and process them in a reasonable amount of time. This ensures that the system is practical and can be used in real world applications.

**Robustness and Imperceptibility :** The system should ensure the robustness and imperceptibility of watermarks. The watermarks should be resilient to various attacks and transformations while remaining invisible to the human eye. This ensures that the watermarks provide effective protection without affecting the visual quality of the images.

**User Interface :** The user interface should be intuitive and user friendly. The system should provide a simple and easy to use interface for embedding, detecting, and removing watermarks, with clear instructions and feedback for the user. This ensures that the system is accessible to a wide range of users and provides a positive user experience.

**Chapter 5**

**Proposed Methodology**

**5.1 Overview**

The proposed methodology involves designing and implementing a comprehensive watermarking system using various techniques. The system will be evaluated for robustness, imperceptibility, and capacity. The goal is to develop a flexible and effective watermarking solution that can meet the requirements of different applications and domains.

**5.2 System Design**

**Architecture :** The system architecture will follow a modular design, where each component (embedding, detection, removal) is independently developed and tested. This approach allows for easy modification and extension of individual components. The architecture will include separate modules for embedding watermarks, detecting watermarks, and removing watermarks, with well defined interfaces and communication protocols between the modules.

**Workflow :** The workflow will involve user inputs for image and watermark information, selection of the embedding technique, and output of the watermarked image. The system will also provide options for detecting and removing watermarks. The workflow will be streamlined and user friendly, with clear instructions and feedback for each step of the process.

**5.3 Watermarking Techniques**

**Spatial Domain Techniques :** These techniques involve direct modification of pixel values. For example, Least Significant Bit (LSB) embedding modifies the least significant bits of the pixel values to embed the watermark. This technique is simple and fast but may be less robust against image processing operations. The implementation will include algorithms for LSB embedding and extraction, with options for customizing the embedding parameters.

**Frequency Domain Techniques :** These techniques involve embedding watermarks in the transformed domain. For example, using Discrete Cosine Transform (DCT) to modify frequency coefficients. This approach is generally more robust against image processing operations and attacks. The implementation will include algorithms for DCT based embedding and extraction, with options for customizing the embedding parameters and selecting the appropriate frequency coefficients for embedding.

**Hybrid Techniques :** Combining spatial and frequency domain techniques to leverage the strengths of both approaches. Hybrid techniques can provide a good balance between robustness and imperceptibility. The implementation will include algorithms that combine spatial and frequency domain techniques, with options for customizing the embedding parameters and selecting the appropriate regions of the image for embedding.

**5.4 Algorithm Implementation**

**Embedding Algorithm :** Steps to insert the watermark into the image. This involves selecting the appropriate technique (spatial or frequency domain), determining the embedding parameters, and modifying the image accordingly. The algorithm will be optimized for efficiency and robustness, with options for customizing the embedding parameters and selecting the appropriate regions of the image for embedding.

**Detection Algorithm :** Steps to extract or verify the presence of a watermark. This involves analysing the watermarked image using the corresponding detection algorithm to retrieve the embedded information. The algorithm will be robust to various attacks and transformations, with options for customizing the detection parameters and selecting the appropriate regions of the image for analysis.

**Removal Algorithm :** Techniques to attempt removing the watermark. This involves applying various image processing operations to degrade or remove the watermark and evaluating the effectiveness of these techniques. The algorithm will include options for customizing the removal parameters and selecting the appropriate regions of the image for processing.

**5.5 Evaluation Metrics**

**Robustness :** Evaluated by subjecting the watermarked images to various attacks (e.g., compression, cropping) and checking if the watermark remains intact. Robustness is a critical measure of the watermark's ability to withstand tampering attempts. The evaluation will include a comprehensive set of attacks and transformations, with detailed analysis of the impact on the watermark.

**Imperceptibility :** Measured using metrics like Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM). These metrics assess the visual quality of the watermarked images compared to the original images. The evaluation will include a detailed analysis of the visual quality of the watermarked images, with comparisons to the original images and analysis of the impact of the watermark on the image quality.

**Capacity :** The amount of information that can be embedded without degrading image quality. Capacity is an important consideration for applications that require embedding large amounts of data. The evaluation will include a detailed analysis of the embedding capacity, with analysis of the impact on image quality and robustness as the amount of embedded information increases.

**Chapter 6**

**Source Code**

**Code for Adding Watermark**

#Adding Watermark

import cv2

def add\_watermark(input\_image\_path, output\_image\_path, watermark\_text, position):

# Load the original image

image = cv2.imread('C:/Users/Sagar H S/Documents/mini project/watermarked.jpg')

# Set the font and get the size of the text

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_scale = 1

font\_thickness = 2

text\_size = cv2.getTextSize(watermark\_text, font, font\_scale, font\_thickness)[0]

# Calculate the position

if position == 'bottom-right':

text\_x = image.shape[1] - text\_size[0] - 10

text\_y = image.shape[0] - text\_size[1] - 10

elif position == 'top-left':

text\_x = 10

text\_y = text\_size[1] + 10

else:

raise ValueError("Position must be 'bottom-right' or 'top-left'")

# Add the text to the image

cv2.putText(image, watermark\_text, (text\_x, text\_y), font, font\_scale, (255, 255, 255), font\_thickness)

# Save the output image

cv2.imwrite('C:/Users/Sagar H S/Documents/mini project/output1.jpg', image)

# Example usage

add\_watermark('input.jpg', 'output\_with\_watermark.jpg', 'sagar', 'bottom-right')

Fig: Code of adding watermark

**Code For Removing Watermark**

#Removing Watermark

import cv2

import numpy as np

def add\_watermark(input\_image\_path, output\_image\_path, watermark\_text, position):

# Load the original image

image = cv2.imread(input\_image\_path)

# Set the font and get the size of the text

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_scale = 1

font\_thickness = 2

text\_size = cv2.getTextSize(watermark\_text, font, font\_scale, font\_thickness)[0]

# Calculate the position

if position == 'bottom-right':

text\_x = image.shape[1] - text\_size[0] - 10

text\_y = image.shape[0] - text\_size[1] - 10

elif position == 'top-left':

text\_x = 10

text\_y = text\_size[1] + 10

else:

raise ValueError("Position must be 'bottom-right' or 'top-left'")

# Add the text to the image

cv2.putText(image, watermark\_text, (text\_x, text\_y), font, font\_scale, (255, 255, 255), font\_thickness)

# Save the output image

cv2.imwrite(output\_image\_path, image)

def create\_watermark\_mask(image\_path, watermark\_text, position):

# Load the image

image = cv2.imread(image\_path)

# Create a blank mask with the same dimensions as the image

mask = np.zeros\_like(image[:, :, 0])

# Set the font and get the size of the text

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_scale = 1

font\_thickness = 2

text\_size = cv2.getTextSize(watermark\_text, font, font\_scale, font\_thickness)[0]

# Calculate the position

if position == 'bottom-right':

text\_x = image.shape[1] - text\_size[0] - 10

text\_y = image.shape[0] - text\_size[1] - 10

elif position == 'top-left':

text\_x = 10

text\_y = text\_size[1] + 10

else:

raise ValueError("Position must be 'bottom-right' or 'top-left'")

# Add the text to the mask

cv2.putText(mask, watermark\_text, (text\_x, text\_y), font, font\_scale, (255, 255, 255), font\_thickness)

return mask

def remove\_watermark(image\_path, mask\_path, output\_image\_path):

# Load the original image and the mask

image = cv2.imread(image\_path)

mask = cv2.imread(mask\_path, 0)

# Apply inpainting to remove the watermark

result = cv2.inpaint(image, mask, inpaintRadius=3, flags=cv2.INPAINT\_TELEA)

# Save the output image

cv2.imwrite(output\_image\_path, result)

# Example usage

# Paths

input\_image\_path = 'C:/Users/Sagar H S/Documents/mini project/watermarked.jpg'

watermarked\_image\_path = 'C:/Users/Sagar H S/Documents/mini project/output\_with\_watermark.jpg'

watermark\_mask\_path = 'C:/Users/Sagar H S/Documents/mini project/watermark\_mask.jpg'

output\_image\_path = 'C:/Users/Sagar H S/Documents/mini project/output\_without\_watermark.jpg'

# Add a watermark

add\_watermark(input\_image\_path, watermarked\_image\_path, 'sagar', 'bottom-right')

# Create a mask for the watermark

mask = create\_watermark\_mask(watermarked\_image\_path, 'sagar', 'bottom-right')

cv2.imwrite(watermark\_mask\_path, mask)

# Remove the watermark

remove\_watermark(watermarked\_image\_path, watermark\_mask\_path, output\_image\_path)

**Chapter 7**

**Experimental Results**

**7.1 Experimental Setup**

**Hardware and Software Environment :** A detailed description of the hardware and software environment used for testing, including computer specifications, operating system, programming languages, and libraries used. The setup will be designed to ensure reproducibility of the results and provide a detailed description of the experimental environment.

**Dataset of Images :** Details on the dataset of images used for experimentation, including information on the number of images, their resolution, and any preprocessing steps applied to the images. The dataset will be representative of the different domains and applications of digital watermarking, with a diverse set of images for comprehensive testing.

**7.2 Results**

**Robustness Testing :** Results showing how well the watermark withstands various attacks. This includes analysis of the watermarked images after being subjected to operations like compression, cropping, and noise addition. The results will include detailed analysis of the impact of each attack on the watermark, with comparisons to the original and watermarked images.

**Imperceptibility Testing :** Results showing the visual quality of watermarked images compared to original images. This includes metrics like PSNR and SSIM, along with visual inspection of the images. The results will include detailed analysis of the visual quality of the watermarked images, with comparisons to the original images and analysis of the impact of the watermark on the image quality.

**Capacity Testing :** Results showing the amount of information that can be embedded without significantly degrading image quality. This includes analysis of the embedding capacity for different watermarking techniques and the impact on image quality. The results will include detailed analysis of the embedding capacity, with analysis of the impact on image quality and robustness as the amount of embedded information increases.

**Results:**

Adding Watermark

****

Fig: Input Image Fig: Output Image

Removing Watermark

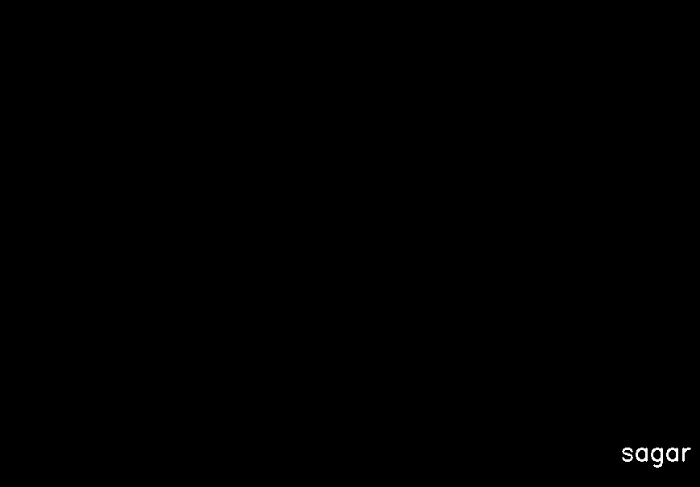
****

Fig: Input Image Fig: Output Watermask Image

****

****

Fig: Output Watermask Image Fig: Final output Image

**Chapter 8**

**References**

A comprehensive list of all sources cited in the report, including books, academic papers, articles, and online resources. This section ensures proper attribution of all referenced material and provides additional resources for further reading. The references will include detailed citations for all sources, with complete and accurate information for each reference.

1. **Digital Watermarking and Steganography (Second Edition) by Cox, Miller, Bloom, Fridrich, and Kalker**:

Link to Book

1. **Introduction to Digital Watermarking by Berghel and O'Gorman**:

[IEEE Xplore](https://ieeexplore.ieee.org/document/494206)

1. **Information Hiding Techniques for Steganography and Digital Watermarking by Katzenbeisser and Petitcolas**:

Link to Book

1. **Reversible Data Hiding by Celik et al.**:

[IEEE Xplore](https://ieeexplore.ieee.org/document/1034258)

1. **Improved Wavelet-Based Watermarking through Pixel-Wise Masking by Barni et al.**:

[IEEE Xplore](https://ieeexplore.ieee.org/document/918592)

1. **Official Website for PhpSpreadsheet**:

[PhpSpreadsheet](https://phpspreadsheet.readthedocs.io/en/latest/)

1. **Digital Image Watermarking: An Overview**:

[ResearchGate](https://www.researchgate.net/publication/220723670_Digital_Image_Watermarking_An_Overview)

1. **A Survey of Digital Watermarking Techniques for Multimedia Data**:

[ScienceDirect](https://www.sciencedirect.com/science/article/pii/S1570870518302582)

**Online Tutorials and Documentation**

1. **PhpSpreadsheet Documentation**:

[PhpSpreadsheet Docs](https://phpspreadsheet.readthedocs.io/en/latest/)

1. **Digital Image Processing Using Matlab by Gonzalez, Woods, and Eddins**:

MathWorks

1. **OpenCV Python Documentation**:

OpenCV Documentation

**Conclusion**

**9.1 Conclusion**

**Summary of Key Findings :** A comprehensive summary of the key findings from the project, including the effectiveness of different watermarking techniques, the robustness and imperceptibility of the watermarks, and the capacity of the watermarking system. The summary will provide a detailed analysis of the key findings, with insights into the strengths and weaknesses of the different techniques and methods.

**Reflection on Objectives :** An evaluation of the project's success in achieving its stated objectives, with analysis of any areas where the project fell short and suggestions for improvement. The reflection will include detailed analysis of the objectives, with insights into the successes and challenges of the project.

**9.2 Future Scope**

**Suggestions for Improvement :** Recommendations for enhancing the robustness, imperceptibility, and capacity of the watermarking system, with analysis of the current techniques and methods and insights into potential areas for improvement and future research. The suggestions will include detailed analysis of the current techniques and methods, with insights into potential areas for improvement and future research.

**Emerging Technologies :** Exploration of emerging technologies and methodologies, such as deep learning for watermarking, and their potential impact on the field. The future scope will include detailed analysis of the potential areas for research, with insights into the current trends and developments in the field.