PixelGuard: A DCT-based Watermarking Tool for Image Protection

Gaurang Mathur PB11
Achintya Chaware PB39
Isha Talathi PB51

Watermarking

- Watermarking is the process of embedding an identifying image or pattern, usually imperceptible, into another image or video.
- Watermarking can be used for a variety of purposes, such as indicating ownership or copyright, providing proof of authenticity, or preventing unauthorized distribution or copying of digital media.
- Digital watermarking can be done using various techniques such as spread spectrum, frequency modulation, and wavelet transform.

Types of Watermarking

- There are two main types of watermarks: visible watermarks and invisible watermarks. Visible watermarks are typically added on top of the image or video and are clearly visible to the viewer. Invisible watermarks, on the other hand, are embedded within the image or video and are not visible to the viewer without special tools or software.

Discrete Cosine Transform (DCT)

- Converts a signal from the time or spatial domain to the frequency domain.
- DCT is used to decompose an image into a set of frequency components.
- DCT coefficients represent the amplitude of each frequency component in the image.
- DCT is similar to the Discrete Fourier Transform (DFT), but is more suited for images as it is less sensitive to small changes in the image.

$$C(u) = a(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

$$u = 0,1,\ldots,N-1$$

$$a(u) = \begin{cases} \sqrt{\frac{1}{N}} & u = 0 \\ \sqrt{\frac{2}{N}} & u = 1,\ldots,N-1 \end{cases}$$

Discrete Cosine Transform (DCT)

- DCT is used in watermarking because it is a lossy transform. That means, even small changes in the original image can cause significant changes in the DCT coefficients. Therefore, it is difficult to remove the watermark without significantly affecting the quality of the watermarked image.
- In watermarking, the watermark is embedded into the image by modifying the DCT coefficients of the image in the frequency domain. The watermark is usually embedded in the high-frequency coefficients of the image because they are less noticeable to the human eye. By modifying the high-frequency coefficients, the watermark can be embedded into the image without affecting its visual quality.

Code Function

- Import necessary libraries and modules: The code imports OpenCV, NumPy, and tkinter libraries, which are required to read, display, and manipulate images, and create GUI interfaces.
- Define the Embedder class: The Embedder class contains the functions to load the original and watermark images, and embed the watermark image into the original image.
- Initialize the GUI: The GUI window is created and initialized with the original and watermark image selection buttons, alpha scale for adjusting the watermark strength, and embed watermark button.

Code Function

- Select the original image: The user is prompted to select an original image using the filedialog, and the selected image is read, converted to grayscale, and resized to 512 x 512 pixels. The resized image is then displayed in a window using OpenCV's imshow() function.
- Select the watermark image: The user is prompted to select a watermark image using the filedialog, and the selected image is read, converted to grayscale, resized to 512 x 512 pixels, and displayed in a window using OpenCV's imshow() function.

Code Function

- Embed the watermark image: The watermark image is zero-padded to match the dimensions of the resized original image. Both the original and watermark images are then converted to their discrete cosine transforms (DCTs) using OpenCV's dct() function. The watermark image DCT is multiplied by an alpha value, which controls the strength of the watermark. The DCT of the original image and the adjusted DCT of the watermark image are added together to create the watermarked DCT. The watermarked DCT is then inverse transformed back to the spatial domain using OpenCV's idct() function, and the resulting watermarked image is displayed in a window using OpenCV's imshow() function.

Code Function

- Run the GUI: The mainloop() function of tkinter is called to start the GUI and wait for user input.
- Terminate the program: The program terminates when the user closes the GUI window.

File Edit Selection View Go Run Terminal Help — main.py - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

main.py 5

Syllabus College > 3rd Year (MITWPU) > DIP > Watermarking (DIP PBL) > main.py > Embedder > embed_watermark

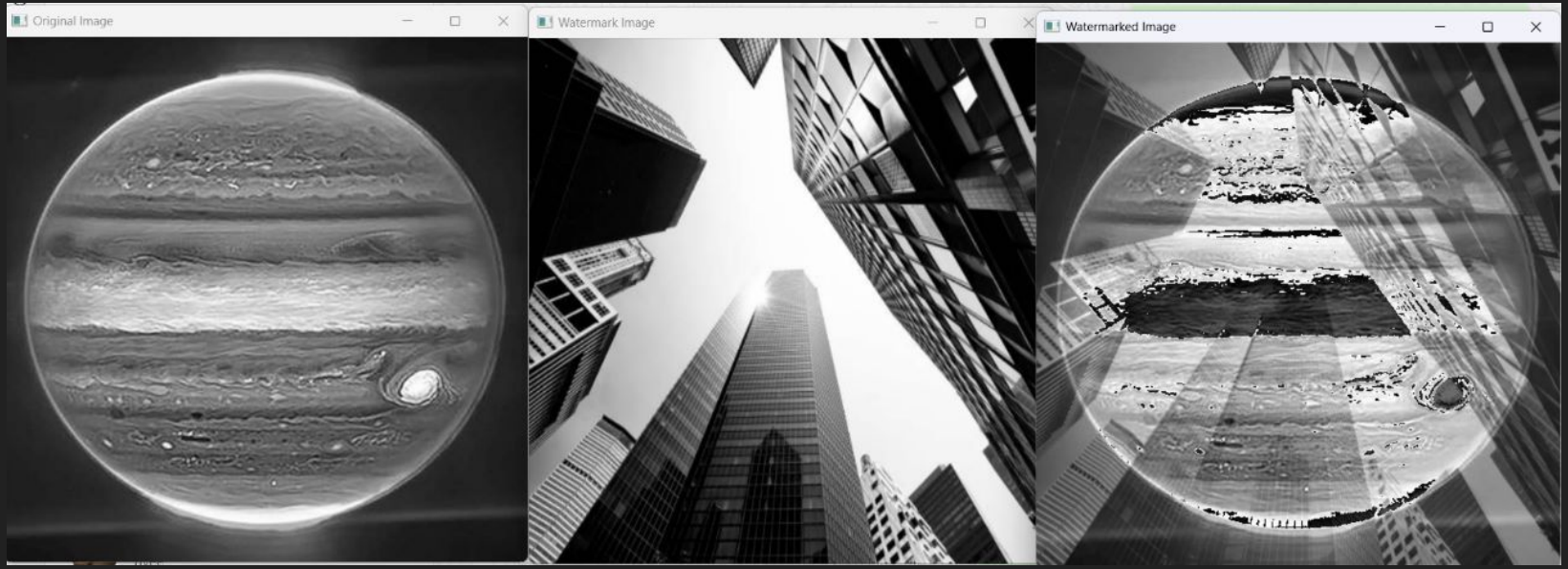
```python
import cv2
import numpy as np
import tkinter as tk
from tkinter import filedialog
from PIL import ImageTk, Image

class Embedder:
    def _init_(self):
        self.original_image_path = ""
        self.original_image = None
        self.watermark_image_path = ""
        self.watermark_image = None
        self.watermarked_image = None
    def select_original_image(self):
        path = filedialog.askopenfilename(filetypes=[("Image File", ".jpg .pn
        if path:
            self.original_image_path = path
            self.original_image = cv2.imread(path)
            cv2.imshow("Original Image", self.original_image)
    def select_watermark_image(self):
        path = filedialog.askopenfilename(filetypes=[("Image File", ".jpg .pn
        if path:
            self.watermark_image_path = path
            self.watermark_image = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
            cv2.imshow("Watermark Image", self.watermark_image)
    def embed_watermark(self):
        if self.original_image is None or self.watermark_image is None:
            return

        # Resize watermark image to fit into original image's bottom right co
        height, width = self.original_image.shape[:2]
        watermark_height, watermark_width = self.watermark_image.shape[:2]
        resized_watermark_height = min(int(height * 0.25), watermark_height)
        resized_watermark_width = min(int(width * 0.25), watermark_width)
        watermark_resized = cv2.resize(self.watermark_image, (resized_waterma

        # Define region of interest (ROI) as bottom right corner of original
        roi = self.original_image[height - resized_watermark_height:height, w
```

main.py 5

C: > Users > gmath > My Drive > Syllabus College > 3rd Year (MITWPU) > DIP > Watermarking (DIP PBL) > main.py > E

```python
        resized_watermark_width = min(int(width * 0.25), watermark_width)
        watermark_resized = cv2.resize(self.watermark_image, (resized_waterma

        # Define region of interest (ROI) as bottom right corner of original
        roi = self.original_image[height - resized_watermark_height:height, w

        # Add watermark to ROI
        watermark_mask = cv2.threshold(watermark_resized, 127, 255, cv2.THRES
        watermark_mask = cv2.merge((watermark_mask, watermark_mask, watermark
        bg_mask = cv2.bitwise_not(watermark_mask)
        watermark_colored = cv2.merge((watermark_resized, watermark_resized,
        watermark_bg = cv2.bitwise_and(roi, bg_mask)
        watermark_fg = cv2.bitwise_and(watermark_colored, watermark_mask)
        watermark_added = cv2.add(watermark_bg, watermark_fg)
        self.watermarked_image = self.original_image.copy()
        self.watermarked_image[height - resized_watermark_height:height, widt

        # Display watermarked image
        cv2.imshow("Watermarked Image", self.watermarked_image)
    def run(self):
        root = tk.Tk()
        root.title("Watermark Embedder")
        root.geometry("500x300")

        original_image_button = tk.Button(root, text="Select Original Image",
        original_image_button.pack()

        watermark_image_button = tk.Button(root, text="Select Watermark Image
        watermark_image_button.pack()

        embed_button = tk.Button(root, text="Embed Watermark", command=self.e
        embed_button.pack()

        root.mainloop()

embedder = Embedder()
embedder.run()
```

Restricted Mode | ⊗ 0 ⚠ 5 | Ln 29, Col 1 | Spaces: 4 | UTF-8 | LF | Python


Original Image
Watermark Image
Watermarked Image

Application

- Copyright protection: Watermarking is commonly used to protect copyrighted material such as images, videos, and audio files. By embedding a unique identifier into the content, it becomes more difficult for others to claim ownership of the material and reduces the likelihood of unauthorized use.
- Authentication: Watermarking is also used for authentication purposes to verify the authenticity of a document or an image. This is especially important in fields such as finance and law, where the integrity of the documents is crucial.
- Digital forensics: Watermarking can be used as a tool for digital forensics to track the origin and distribution of a digital asset. By embedding a unique identifier into the asset, it can be traced back to its source and any unauthorized copies can be identified.
- Branding: Watermarking is also used for branding purposes. Companies can embed their logo or brand name into images and videos to promote their brand and prevent unauthorized use of their content.
- Data hiding: Watermarking can be used to hide data within an image or a video. This technique is often used in steganography, where the goal is to hide the existence of the data rather than to protect it.

Conclusion

- Watermarking using DCT is an effective method for embedding a watermark in an image. The watermarked image appears to be visually similar to the original image, with the watermark being barely noticeable at lower alpha values.
- At higher alpha values, the watermark becomes more visible, but it still does not significantly alter the visual appearance of the original image.