Homework #2 – 2D-DCT

312553024 江尚軒

Code

Load the image and convert to grayscale

```python
# 1. Load the image and convert to grayscale
print("1. Load the image and convert to grayscale...")
image = cv2.imread('image/lena.png', cv2.IMREAD_GRAYSCALE)
image = cv2.resize(image, (256, 256))
print("image.shape:", image.shape)
```

Apply one 2D-DCT

```python
# 2. Apply one 2D-DCT
print("2. Apply one 2D-DCT...")
start_time = time.time()
dct_coefficients_2d = dct_2d(image)
end_time = time.time()
print(f'One 2D-DCT runtime: {end_time - start_time:.4f} seconds')
```

```python
# Implement 2D-DCT
Tabnine | Edit | Test | Explain | Document | Ask
def dct_2d(image):
    M, N = image.shape
    dct = np.zeros((M, N))

    for u in tqdm(range(M), desc='2D-DCT'):
        for v in range(N):
            sum_val = 0
            for x in range(M):
                for y in range(N):
                    sum_val += image[x, y] * np.cos((2 * x + 1) * u * np.pi / (2 * M)) * np.cos((2 * y + 1) * v * np.pi / (2 * N))

            c_u = 1 / np.sqrt(2) if u == 0 else 1
            c_v = 1 / np.sqrt(2) if v == 0 else 1
            dct[u, v] = (2 / N) * c_u * c_v * sum_val

    return dct
```

Visualize the 2D-DCT coefficients in the log domain

```python
# 3. Visualize the 2D-DCT coefficients in the log domain
print("3. Visualize the 2D-DCT coefficients in the log domain...")
log_dct_2d = np.log(np.abs(dct_coefficients_2d) + 1)
plt.imshow(log_dct_2d, cmap='gray')
plt.title('2D-DCT Coefficients')
plt.savefig('image/DCT_coefficients_2D.png')
# plt.show()
```

Apply one 2D-IDCT

```python
# 4. Apply one 2D-IDCT
print("4. Apply one 2D-IDCT...")
reconstructed_image_2d = idct_2d(dct_coefficients_2d)
cv2.imwrite('image/lena_reconstructed_2D.png', reconstructed_image_2d)
```

```python
# Implement 2D-IDCT
Tabnine | Edit | Test | Explain | Document | Ask
def idct_2d(dct):
    M, N = dct.shape
    idct = np.zeros((M, N))

    for x in tqdm(range(M), desc='2D-IDCT'):
        for y in range(N):
            sum_val = 0
            for u in range(M):
                for v in range(N):
                    c_u = 1 / np.sqrt(2) if u == 0 else 1
                    c_v = 1 / np.sqrt(2) if v == 0 else 1
                    sum_val += c_u * c_v * dct[u, v] * np.cos((2 * x + 1) * u * np.pi / (2 * M)) * np.cos((2 * y + 1) * v * np.pi /

            idct[x, y] = (2 / N) * sum_val

    return np.clip(idct, 0, 255)
```

Calculate PSNR (2D)

```python
# 5. Calculate PSNR (2D)
print("5. Calculate PSNR (2D)...")
psnr_value = psnr(image, reconstructed_image_2d)
print(f'PSNR between original and reconstructed image (2D): {psnr_value:.2f} dB')
```

Apply two 1D-DCT

```python
# 2. Apply two 1D-DCT
print("2. Apply two 1D-DCT...")
start_time = time.time()
dct_coefficients_1d = two_dct_1d(image)
end_time = time.time()
print(f'Two 1D-DCT runtime: {end_time - start_time:.4f} seconds')
```

```python
# Implement fast 2D-DCT using two 1D-DCT
Tabnine | Edit | Test | Explain | Document | Ask
def dct_1d(vector):
    N = len(vector)
    result = np.zeros(N)

    for u in range(N):
        sum_val = 0
        for x in range(N):
            sum_val += vector[x] * np.cos((2 * x + 1) * u * np.pi / (2 * N))

        c_u = 1 / np.sqrt(2) if u == 0 else 1
        result[u] = np.sqrt(2 / N) * c_u * sum_val

    return result


Tabnine | Edit | Test | Explain | Document | Ask
def two_dct_1d(image):
    M, N = image.shape
    dct_rows = np.zeros((M, N))

    # Apply 1D-DCT on rows
    for i in tqdm(range(M), desc='1D-DCT on rows'):
        dct_rows[i, :] = dct_1d(image[i, :])

    # Apply 1D-DCT on columns
    dct = np.zeros((M, N))
    for j in tqdm(range(N), desc='1D-DCT on columns'):
        dct[:, j] = dct_1d(dct_rows[:, j])

    return dct
```

Visualize the 1D-DCT coefficients in the log domain

```python
# 3. Visualize the 1D-DCT coefficients in the log domain
log_dct_1d = np.log(np.abs(dct_coefficients_1d) + 1)
plt.imshow(log_dct_1d, cmap='gray')
plt.title('1D-DCT Coefficients')
plt.savefig('image/DCT_coefficients_1D.png')
# plt.show()
```

Apply two 1D-IDCT

```python
# 4. Apply two 1D-IDCT
print("4. Apply two 1D-IDCT...")
reconstructed_image_1d = two_idct_1d(dct_coefficients_1d)
cv2.imwrite('image/lena_reconstructed_1D.png', reconstructed_image_1d)
```

```python
def idct_1d(vector):
    N = len(vector)
    result = np.zeros(N)

    for x in range(N):
        sum_val = 0
        for u in range(N):
            c_u = 1 / np.sqrt(2) if u == 0 else 1
            sum_val += c_u * vector[u] * np.cos((2 * x + 1) * u * np.pi / (2 * N))

        result[x] = np.sqrt(2 / N) * sum_val

    return result


Tabnine | Edit | Test | Explain | Document | Ask
def two_idct_1d(dct):
    M, N = dct.shape
    idct_temp = np.zeros((M, N))

    # Apply 1D-IDCT on columns
    for j in tqdm(range(N), desc='1D-IDCT on columns'):
        idct_temp[:, j] = idct_1d(dct[:, j])

    # Apply 1D-IDCT on rows
    idct = np.zeros((M, N))
    for i in tqdm(range(M), desc='1D-IDCT on rows'):
        idct[i, :] = idct_1d(idct_temp[i, :])

    return np.clip(idct, 0, 255)
```

Calculate PSNR (1D)

```python
# 5. Calculate PSNR (1D)
print("5. Calculate PSNR (1D)...")
psnr_value = psnr(image, reconstructed_image_1d)
print(f'PSNR between original and reconstructed image (1D): {psnr_value:.2f} dB')
```

Apply OpenCV's DCT

```python
# 2. Apply OpenCV's DCT
print("2. Apply OpenCV's DCT...")
start_time = time.time()
dct_coefficients_opencv = cv2.dct(np.float32(image))
end_time = time.time()
print(f"OpenCV's DCT runtime: {end_time - start_time:.4f} seconds")

log_dct_opencv = np.log(np.abs(dct_coefficients_opencv) + 1)
plt.imshow(log_dct_opencv, cmap='gray')
plt.title('DCT Coefficients (OpenCV)')
plt.savefig('image/DCT_coefficients_OpenCV.png')
# plt.show()
```

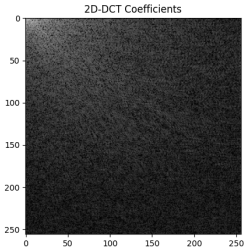
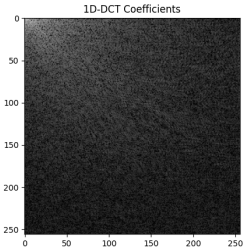
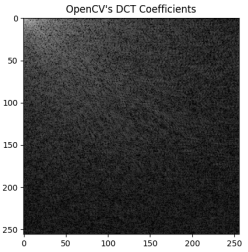
Apply OpenCV's IDCT

```python
# 3. Apply OpenCV's IDCT
print("3. Apply OpenCV's IDCT...")
reconstructed_image_opencv = cv2.idct(np.float32(dct_coefficients_opencv))
cv2.imwrite('lena_reconstructed_OpenCV.png', reconstructed_image_opencv)

psnr_value = psnr(image, reconstructed_image_opencv)
print(f'PSNR between original and reconstructed image (OpenCV): {psnr_value:.2f} dB')
```

Result

Visualize the DCT coefficients in the log domain

2D-DCT	1D-DCT	OpenCV
		

Reconstruct the image

2D-DCT	1D-DCT	OpenCV
		

Evaluate the PSNR

2D-DCT	1D-DCT	OpenCV
279.18 dB	281.46 dB	148.53 dB

Compare the runtime

2D-DCT	1D-DCT	OpenCV
4010.4473 seconds	17.4069 seconds	0.0003 seconds

2D-DCT 的複雜度是 N^4，而 1D-DCT 的複雜度是 $2*N^3$，因此 2D-DCT 會比 1D-DCT 多大概 ½*N 的時間。OpenCV 的 DCT 應該有做優化，所以快非常多。