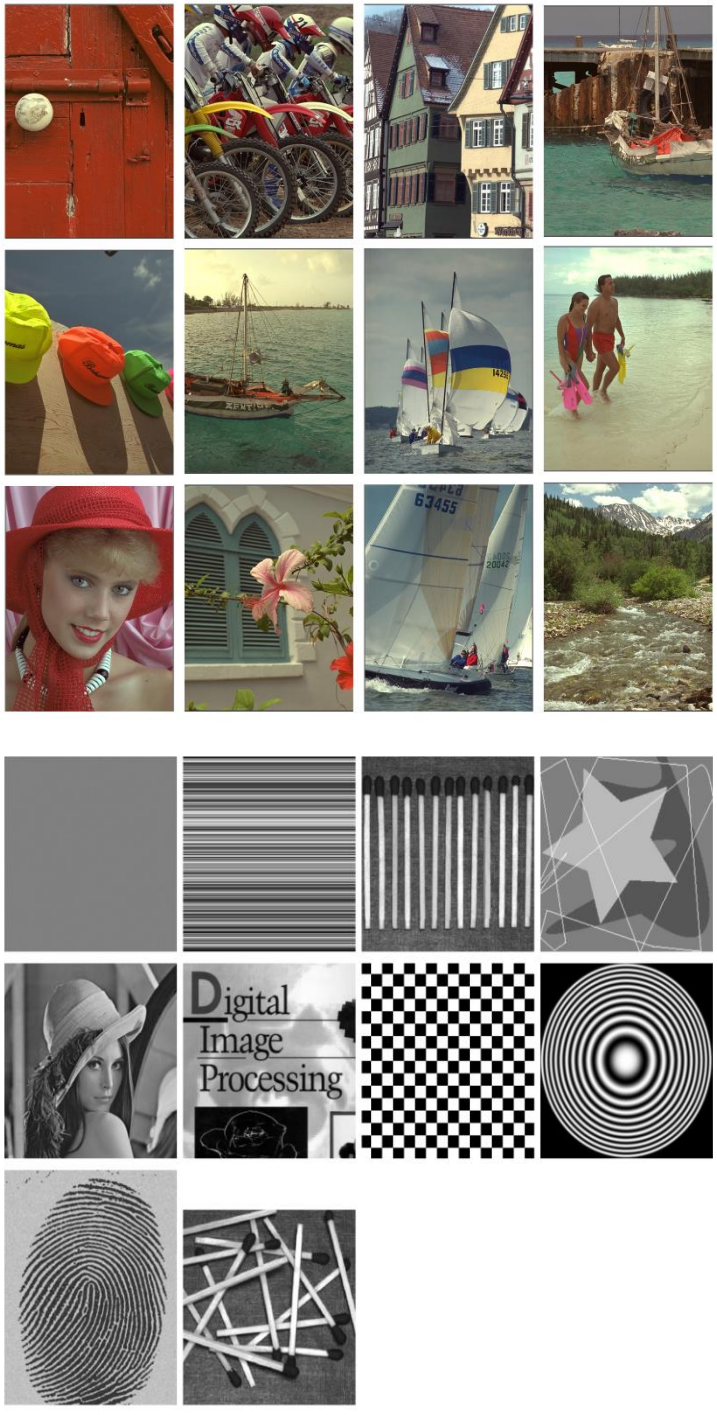
Video Coding Seminar
Image-Compression using Lapped Transform

Umair Khalid - 66512

Muhammad Waleed Khan -64684

Original Images with Colored and Grayscale



ORIGINAL - GRAY

Discrete Cosine Transform (DCT)

Explanation:

1. DCT is a mathematical technique used to convert spatial domain data into frequency domain data. It is widely used in image compression, most particularly in JPEG compression.

2. DCT works by breaking down an image into frequency components. DCT helps in isolating these low frequency components, which can be retained, while the less important high frequency component can be compressed or discarded, leading to efficient compression.

3. DCT is particularly effective for image compression because natural images typically have lower frequency components. By focusing on these low frequency components, DCT allows for high compression ratios with minimal loss in perceived image quality.

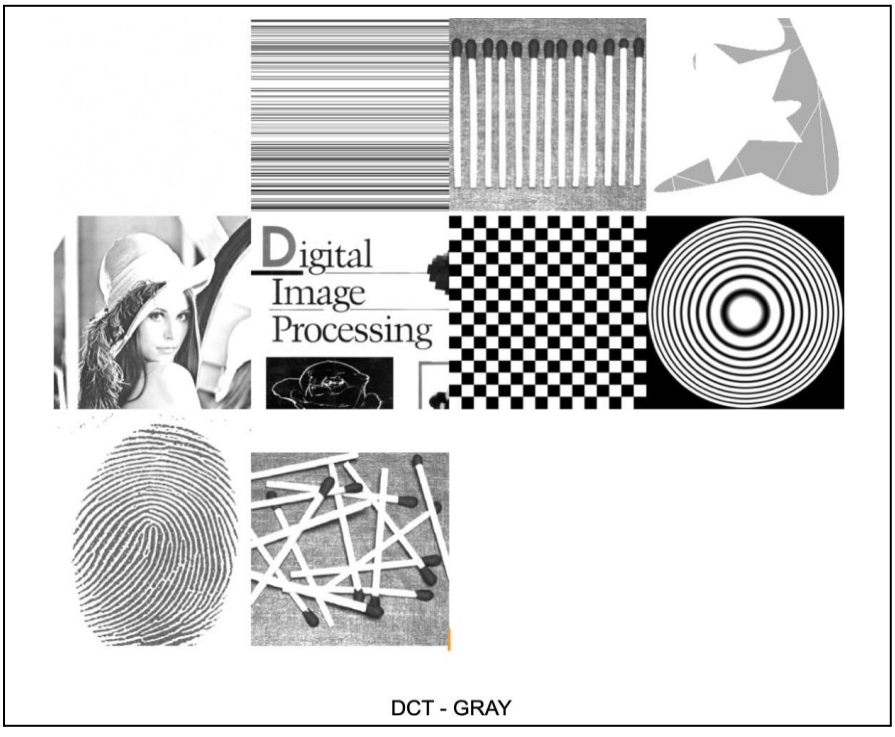
Color DCT Image processing Code explanation:

1. **Image Loading:** The function loads images from specific directory as grayscale images using ‘cv2.IMREAD_GRAYSCALE’. This means that images are processed as grayscale images despite being from color image directory.
2. **DCT APPLICATION:** The grayscale images are converted to floating point format using ‘np. float32’ and the DCT is applied using ‘cv2.dct’. This transforms the image data from spatial domain to frequency domain.
3. **High-Frequency Amplification:** The DCT coefficients are multiplied by 2, which implifies the high frequency components.
4. **Inverse DCT and Clipping:** It is applied to convert the image back to spatial domain. The resulting image is clipped to ensure pixel values are within 0-255 range.
5. **Saving the image:** The processed image is saved to directory content/drive/MyDrive/image-compression/dct_color

Color DCT Image processing Code explanation:

1. **Image Loading:** Images are loaded from specific directory as a grayscale using cv2.IMREAD_GRAYSCALE.
2. **DCT Application:** The image is converted to ‘np. float32’ and DCT is applied using ‘cv2.dct’ transforming into frequency domain.
3. **High Frequency amplifications:** The DCT coefficients are multiplied by 2 to enhance high frequency components.
4. **Inverse DCT and clipping**: The inverse DCT reconstruct the image from frequency domain and pixel values are clipped to range 0-255 and converted to ‘unit8’
5. **Saving the Image:** The processed image is saved to ‘/content/drive/MyDrive/image-compression/dct_grayscale’ with the directory being created if it does not exist.

Colored and Grayscale Images with DCT



DCT - GRAY

Lapped Transform

Overview: The lapped transform is an advanced technique designed to improve upon the DCT. It aims to address and reduce block artifacts, which can be noticeable in DCT based compression particularly at high compression rate.

Reduction of block artifacts:

Block artifacts: DCT can lead to visible discontinuities or artifacts at the edge of image blocks where blocks do not align perfectly with each other.

The lapped transform mitigates these artifacts by overlapping adjacent blocks of pixels. The overlapping allows for smoother transitions between blocks resulting in more visually pleasing image with fewer noticeable boundaries.

Implementation Details:

Pytorch 'Conv2d': In this project, the lapped transform was implemented using pytorch 'Conv2D' layer with 16x16 kernel size.

Training filters: The filters used in 'Conv2D' layer were trained using images from the image coder repository. This training process optimizes the filters to effectively perform the lapped transform, adapting them to characteristics of images used.

Color Lapped Transform Code:

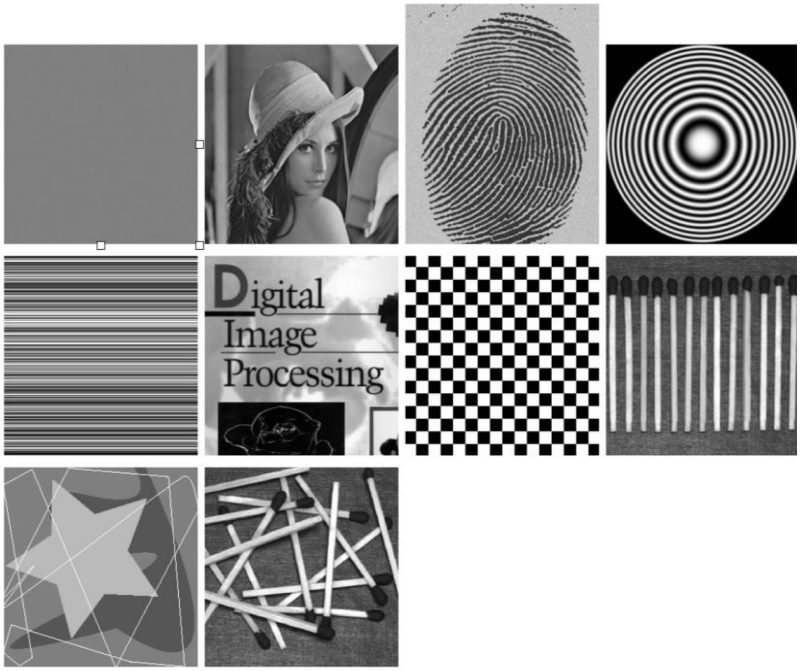
- **Lapped Transform Functions:**
  Padding: The image is padded to ensure it dimensions are multiple of block size minus overlap. This avoids boundary issues during processing.
  Block Processing: The image is divided into overlapping blocks of size 'block of size x block_size' . Each block undergoes DCT to convert into frequency domain.
  DCT and IDCT: The DCT is applied to each block followed by the inverse DCT to reconstruct the image from its frequency components.
  Overlap-add: Overlapping blocks are combined using overlap add technique. This ensures smooth transitions between blocks and reduces artifacts.
  Clipping: The pixel value is clipped to range 0-255 and converted to 'unit8' to produce final image.
- **Processing and Saving Images:**
  Image Loading: Images are loaded from specified directory in grayscale format.
  Transformation: The lapped transform function is applied to each image.
  Saving: The processed image is saved in the '/content/drive/MyDrive/image-compression/lapped_color' directory. If a directory does not exit , it is created.

Grayscale Lapped Transform code:

- **Image Loading:**
  Read image in grayscale from specified directory
- **Lapped Transform:**
  Padding: Add padding to align image dimensions for block processing.
  Block Processing: Applies DCT and IDCT to overlapping blocks of image.
  Clipping: Clip values to 0-255 and convert to 'unit8'.
- **Saving:**
  Output Directories:
  Saves processed images to /content/drive/MyDrive/image-compression/lapped_scale/ creating the directory if it does not exist.

- Colored and Grayscale Images with Lapped Transform

LAPPED - COLOR



LAPPED - GRAY

Compression Ratio

1. Compression Ratio:

Definition: This metric represents the ratio between size of original, uncompressed image and size of compressed image.

Purpose: The compression ratio helps assess how effectively a compression algorithm reduces the file size.

2. Significance:

Higher compression ratio: Indicates a greater reduction in file size which is desirable for saving storage space and reducing transmission time.

Quality trade off: A higher compression ratio often results in loss of image quality. The Goal is to achieve a good compression ratio while maintaining acceptable image quality.

Compression ratio Code Explanation

Function Definitions

1. get_file_size(file_path) :

Purpose: Retrieve the file size of specified file in bytes.

Implementation: Uses os.path.getsize() to get the file size if file exist otherwise return 0 .

2. Calculate_compression_ratios(original_path):

Purpose: Calculates and compares the compression ratios of images for both DCT and lapped transform methods

Process for Color Images:

1. Paths:

Original Path: The directory containing the original color images

DCT Path: The directory where DCT compressed color images are saved.

Lapped Path: The directory where lapped transform compressed color images are saved.

2.**Processing:**

File Size:

Original Size: Size of original Image file.

DCT Size: Size of DCT compressed image file

Lapped Size: Size of lapped transform compressed image file.

Compression Ratios:

DCT Compressed ratio: Ratio of original size to DCT compressed size.

Lapped Compressed Ration: Ratio of original size to lapped transform compressed size.

3. Results:

Data Collection: Appends results for each image to a list.

Output: Save the results to a CSV file compression_ratios_color.csv for further analysis.

Process for Grayscale Images:

As I explained paths, processing and result for color images. The same thing will implement for grayscale images with a very little modifications for example:

Results:

Output: Save the result to CSV file compression_ratio_grayscale.csv for further analysis.

Color Images Compression Ratio:

This data just compares the original, DCT and Lapped size, plus also compares the compression ratio of DCT and lapped.

	A	B	C	D	E	F	G
1	Filename	Original Size (bytes)	DCT Size (bytes)	Lapped Size (bytes)	DCT Compression Ratio	Lapped Compression Ratio	
2	9.png	582899	116593	260903	4.999433928280428	2.2341598218494996	
3	6.png	618959	181129	260903	3.4172275008419413	2.3723721076415374	
4	20.png	492462	120209	260903	4.096714888236321	1.8875290816893635	
5	22.png	701970	221542	260903	3.168563974	2.690540162435848	
6	10.png	593463	173128	260903	3.427885726167922	2.2746499656960633	
7	14.png	692201	237476	260903	2.914825077060419	2.6530971280514213	
8	21.png	637051	178082	260903	3.577290237081794	2.4417158867471818	
9	23.png	557596	181810	260903	3.066916011220505	2.1371774184275383	
10	24.png	706397	246989	260903	2.8600342525375626	2.7075081543715482	
11	3.png	502888	177543	260903	2.832485651363332	1.9274902933274052	
12	8.png	788470	241111	260903	3.270153580715936	3.022081003284746	
13	5.png	785610	290383	260903	2.705426970587121	3.011119074905233	
14	11.png	621023	252478	260903	2.459711341186163	2.380283093716822	
15	7.png	566322	176180	260903	3.2144511295266205	2.170622798511324	
16	12.png	531024	54399	260903	9.76165003	2.0353311383924293	
17	2.png	617995	260903	260903	2.3686772478660654	2.3686772478660654	
18	1.png	736501	235394	260903	3.1288010739441106	2.822892032671146	
19	19.png	671476	204230	260903	3.2878421387651176	2.573661475720862	
20	13.png	822712	284900	260903	2.8877220077220076	3.1533251821558204	
21	16.png	534247	199708	260903	2.6751407054299277	2.047684	
22	18.png	780947	288388	260903	2.7079732859897083	2.9932465322361184	
23	4.png	637432	211718	260903	3.0107595953107436	2.4431761995837533	
24	15.png	612582	192527	260903	3.1817978777002707	2.3479300736288966	
25	17.png	602078	205098	260903	2.935562512	2.3076699003077774	

Grayscale Images Compression Ratio:

This data just compares the original, DCT and Lapped size, plus also compares the compression ratio of DCT and lapped.

	A	B	C	D	E	F
1	Filename	Original Size (bytes)	DCT Size (bytes)	Lapped Size (bytes)	DCT Compression Ratio	Lapped Compression Ratio
2	4.tif	89148	5928	328318	15.03846154	0.271529432
3	6.tif	430068	128294	328318	3.35220665	1.30991295
4	3.tif	360678	310126	328318	1.163004714	1.098562979
5	10.tif	360678	315474	328318	1.143289146	1.098562979
6	2.tif	1286	1860	328318	0.691397849	0.003916934
7	7.tif	1049286	43038	328318	24.38045448	3.195944176
8	5.tif	286740	176250	328318	1.626893617	0.873360583
9	1.tif	54968	12648	328318	4.345983555	0.167423047
10	8.tif	356976	173328	328318	2.059540294	1.087287325
11	9.tif	765450	265058	328318	2.887858506	2.331428676
12						

LPIPS(Learned perceptual image patch similarity)

Introduction: LPIPS is a metric designed to evaluate the perceptual similarity between two images.

Purpose: Unlike traditionally metric that focus on pixel by pixel distributions, LPIPS assesses how similar two images are in terms of human visual perception.

How it works: LPIPS relies on deep learning model that have been trained on large dataset to understand visual similarity. These models extract features from images and compute similarity based on these high level features rather than raw pixel values.

Comparison with traditional metrics: Traditional metrics like MSE or PSNR measure pixel level differences and can be sensitive to perceptual quality.

Application in Image compression: In context of image compression, LPIPS can be used to compare compressed images with their original counterparts. This helps in evaluating how well different compression techniques maintain perceptual quality.

Selecting Methods: By using LPIPS, one can select compression algorithms that provide better visual fidelity even if pixel wise error might be higher.

Code Logic

1. Initializations:

- **Variables:**
  Total_lpips inititalized to accumulate the total lpips across all images.
  Image_count: Initialized to count the number of images processed.
  Data: An empty list is created to score the LPIPS values for each image

2. Iterate over original Images

- **File Processing:**
  The function iterates through filenames in the original_path directory.
  For each filename, it constructs paths for corresponding original and compressed images in their respective directories.
  It checks for existence of both images, skipping anyfiles where either image is missing.

3. Load and transform images:

- **Image Loading:**
  Original Image loaded from original_path and converted to RGB format.
  Compressed image loaded from compressed_path and also converted to RGB format
- **Image Transformation:**
  Both images are transformed into tensors using a predefined transformation pipeline, which includes resizing tensor conversion and normalization.

4. Calculate LPIPS:

- The lpips score is computed between original and compressed images using lpips_model
- The score is added to total_lpips and image_count is incremented

5. Store Results: For each image, a directory containing the filename and its lpips score is appended to data list

Comparison of DCT and Lapped Transform

LPIPS for Color Images

	A	B	C
1	File Name	LPIPS Value	
2	9.png	0.7115689516067505	
3	6.png	0.7279881238937378	
4	20.png	0.7428398132324219	
5	22.png	0.7082239985466003	
6	10.png	0.7222594618797302	
7	14.png	0.6887439489364624	
8	21.png	0.731932	
9	23.png	0.6937296390533447	
10	24.png	0.6912038326263428	
11	3.png	0.6866110563278198	
12	8.png	0.6874560117721558	
13	5.png	0.6947752237319946	
14	11.png	0.6403255462646484	
15	7.png	0.7258425951004028	
16	12.png	0.7357351779937744	
17	2.png	0	
18	1.png	0.7274039387702942	
19	19.png	0.7086590528488159	
20	13.png	0.7519646883010864	
21	16.png	0.7508564591407776	
22	18.png	0.714102	
23	4.png	0.6935297250747681	
24	15.png	0.7152734994888306	
25	17.png	0.7326942682266235	

LPIPS for Grayscale Images

	A	B
1	File Name	LPIPS Value
2	4.tif	0.70881903
3	6.tif	0.75058484
4	3.tif	0.69567096
5	10.tif	0.71283692
6	2.tif	0.95079648
7	7.tif	0.7755518
8	5.tif	0.74488962
9	1.tif	0.77284408
10	8.tif	0.83917487
11	9.tif	0.77363211

Conclusion

Summary of findings

- **Compression Ratios:**

Color Images: The analysis revealed that the compression ratios for images compressed using the lapped transform were generally higher than those for images compressed using DCT. This means that lapped transform typically achieved the higher degree of compression, resulting in smaller file size for same level of image quality

Grayscale Images: Similarly, for grayscale images, the lapped transform also demonstrated the better compression efficiency compared to DCT. The results indicated that lapped transform can reduce the file size more effectively while preserving image quality

- **LPIPS Score:**

Color Images: The perceptual similarity as measured by lpips showed that images compressed by using DCT and lapped transform had varying levels of perceptual quality. Image compressed with lapped transform had lower lpips score indicating better preservation of perceptual quality compared to DCT.

Grayscale Images: The results were consistent with those for color images. The lapped transform typically achieved the better perceptual similarity, suggesting that it better maintained the visual fidelity of images compared to DCT.

- **Overall Performance:**

The lapped transform outperformed DCT in both color and grayscale image regarding both compression ratios and perceptual quality. This suggests that lapped transform is more effective for maintaining image quality while achieving higher compression.