Video Coding Seminar
Image-Compression using Lapped Transform

Suprabha Ghosh - 64365

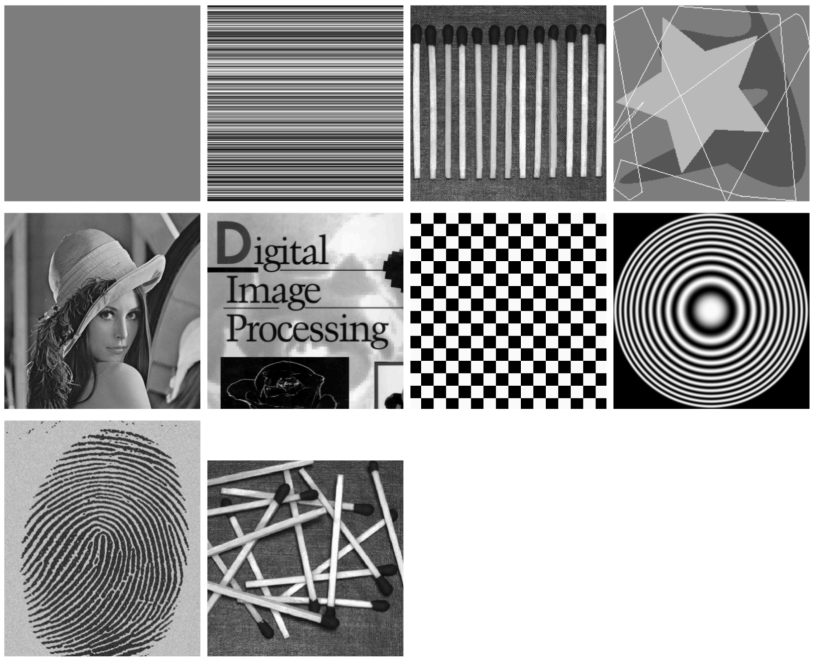
Ismail Artun Altinel - 64722

Tarun Devidas Ramani - 66160

31 July 2024

Original Images

ORIGINAL - COLOR



ORIGINAL - GRAY

Discrete Cosine Transform (DCT)

DCT Logic

1. **Frequency Domain Transformation**: DCT converts an image from the spatial domain (pixel values) to the frequency domain (cosine basis functions).
2. **Energy Compaction**: Most visual information in an image is concentrated in the lower-frequency components after DCT, allowing for the compression of data by keeping significant components and discarding the less important high-frequency ones.
3. **Quantization and Compression**: The DCT coefficients are quantized, reducing the precision of less significant components, thus achieving compression. The inverse DCT reconstructs the image from the compressed data.

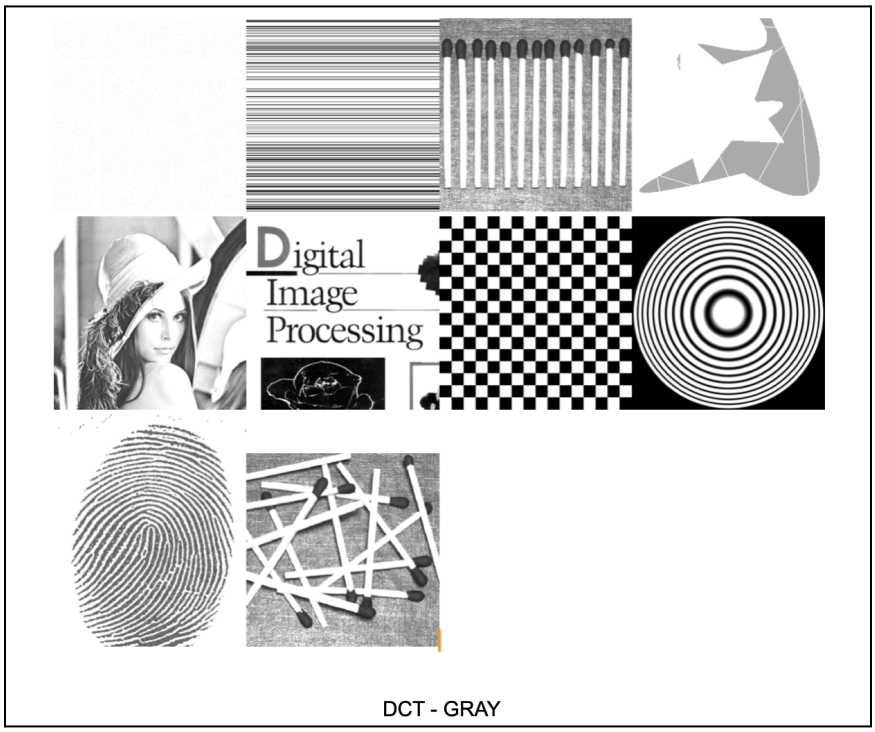
Code Logic

1. **Image Loading**: The image is loaded in grayscale and converted to float32 for precision in calculations.
2. **DCT Application**: `cv2.dct` computes the DCT of the image, producing a matrix of DCT coefficients.
3. **High-Frequency Amplification (Optional)**: The code amplifies high-frequency components, potentially enhancing details but risking noise introduction.
4. **Inverse DCT**: `cv2.idct` converts the DCT coefficients back into the spatial domain, reconstructing the image.
5. **Clipping and Saving**: The resulting image is clipped to ensure pixel values are valid (0-255), converted to uint8 format, and saved.

This process compresses the image by retaining critical visual information, reducing the amount of data while maintaining acceptable image quality.

For task-1, the given git repository (https://github.com/Karanraj06/image-compression) only compresses original images one by one. It needs inputs to convert. For Jpeg compression, image filename, block size, # of Coefficient parameters sent, and Color/grayscale binary input. Inspired by that repository we implement our own solution in our code and get a DCT for original images like down below.

Transformed Images with DCT



DCT - GRAY

DCT - COLOR

Lapped Transform

The Lapped Transform is an extension of block-based transforms like DCT but with overlapping blocks.
It aims to reduce block artifacts by overlapping adjacent blocks of data.

Theoretical Summary

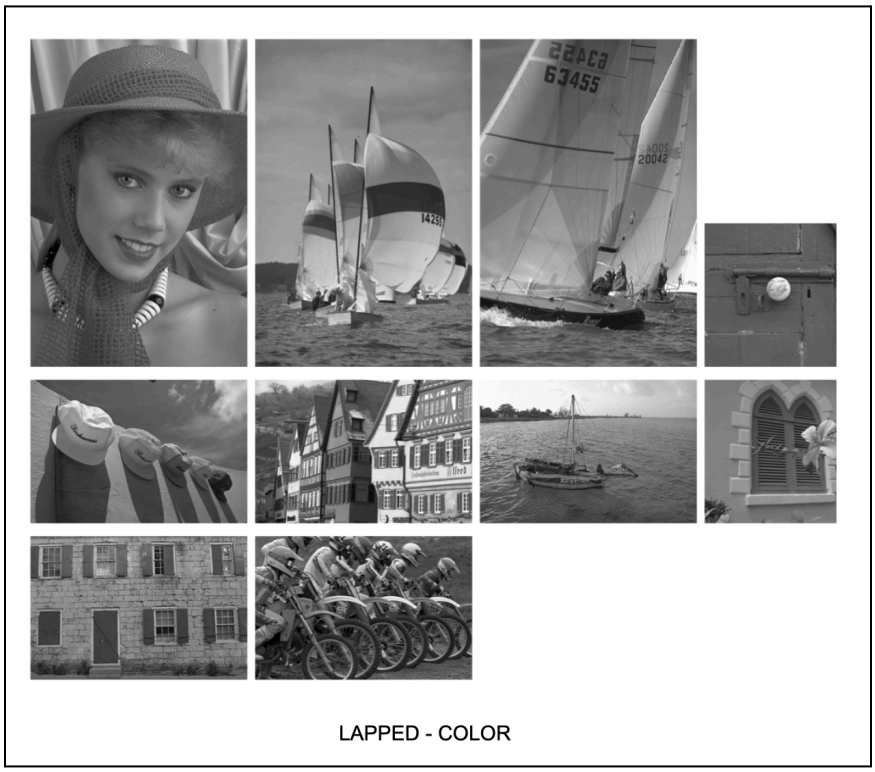
- **Purpose**: Reduces block boundary artifacts in image compression by using overlapping blocks.
- **Key Features**:
    - **Overlapping Blocks**: Adjacent blocks share data, smoothing transitions and reducing blockiness.
    - **Frequency Domain Transformation**: Like DCT, LT transforms image data to the frequency domain, where most visual information is concentrated in low frequencies.
    - **Redundancy Reduction**: Efficiently compresses data while maintaining high visual quality.

Code Logic

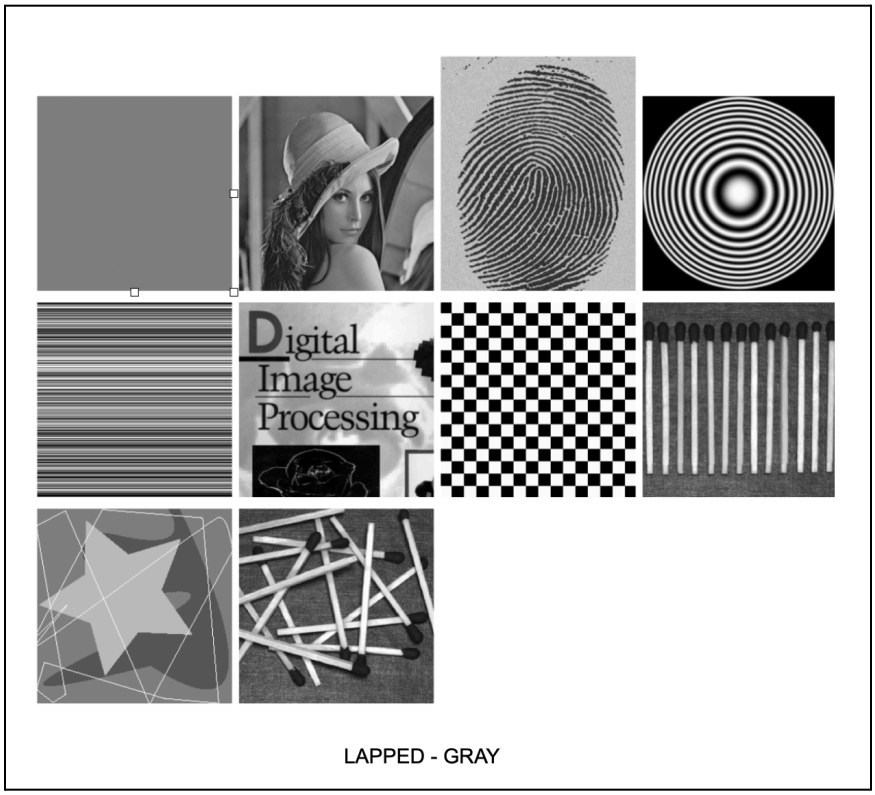
1. **Image Loading**
    - Load the image in grayscale and convert it to float32.
2. **Padding**
    - Pad the image so its dimensions are multiples of the step size (block size minus overlap).
3. **Initialization**
    - Initialize `compressed` to store the transformed blocks.
    - Initialize `count` to track the number of updates per pixel.
4. **Processing Overlapping Blocks**
    - Divide the image into overlapping blocks.
    - Apply 2D DCT to each block.
    - (Optional) Quantize the DCT coefficients.
    - Apply inverse DCT (IDCT) to revert blocks to spatial domain.
    - Overlap-add and average the blocks in `compressed`, updating `count`.
5. **Finalizing Image**
    - Divide `compressed` by `count` to average overlapping regions.
    - Clip values to the valid range [0, 255] and convert to uint8.
    - Save the processed image.

This process ensures high-quality compression by reducing artifacts and maintaining smooth transitions between blocks.

Transformed Images with Lapped Transform



LAPPED - COLOR



LAPPED - GRAY

Compression Ratio

- **Definition**: The compression ratio is a metric that measures the effectiveness of a compression algorithm. It is defined as the ratio of the original file size to the compressed file size. A higher compression ratio indicates more effective compression.
- **Importance**: It helps in evaluating how much data has been reduced and is crucial for applications requiring efficient storage and transmission.

Code Logic

1. **Directory Iteration**
    - The code iterates over each file in the `original_path` directory, which contains the original uncompressed images.
2. **File Size Retrieval**
    - For each image, the corresponding compressed versions (DCT and Lapped Transform) are located in the `compressed_dct_path` and `compressed_lbt_path` directories, respectively.
    - The `get_file_size` function is used to determine the file sizes in bytes for the original, DCT-compressed, and Lapped Transform-compressed images.
3. **Compression Ratio Calculation**
    - **DCT Compression Ratio**: Calculated as the original file size divided by the DCT-compressed file size.
    - **Lapped Compression Ratio**: Calculated as the original file size divided by the Lapped Transform-compressed file size.
    - These ratios quantify the extent to which the file size has been reduced by each compression technique.
4. **Data Aggregation**
    - For each image, the code stores the filename, original size, compressed sizes, and the calculated compression ratios in a dictionary.
    - These dictionaries are collected in a list called `ratios`.
5. **Output**
    - The function returns the list of dictionaries, providing a detailed report of the compression ratios for each image, facilitating a comparative analysis of the efficiency of the DCT and Lapped Transform compression methods.

This systematic approach allows for a clear comparison of the two compression techniques, highlighting their efficiency and effectiveness in reducing file sizes while retaining essential image information.

Compression Ratios: Color Images

The data presented in the table compares the sizes and compression ratios of the color images transformed using DCT and Lapped transforms.

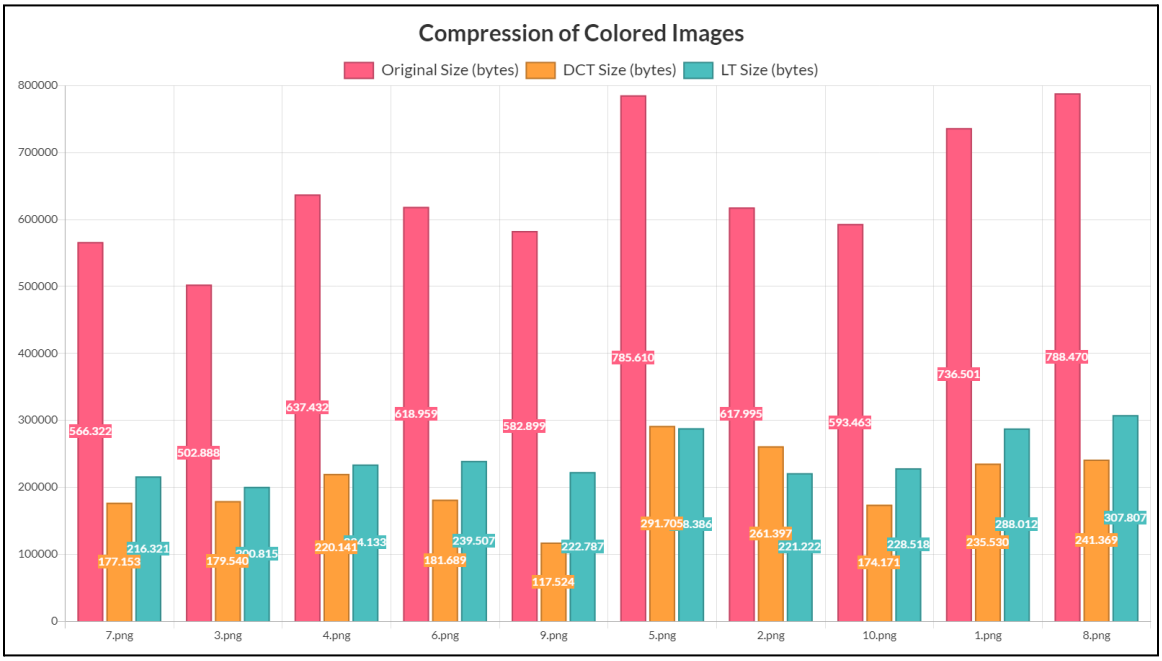
	Filename	Original Size (bytes)	DCT Size (bytes)	LT Size (bytes)	DCT Comp Ratio %	Lapped Comp Ratio %
1	7.png	566322	177153	216321	68.71867947916556	61.802472798160764
2	3.png	502888	179540	200815	64.2982135187159	60.06764925788645
3	4.png	637432	220141	234133	65.46439463346678	63.269336964570336
4	6.png	618959	181689	239507	70.64603632873906	61.30486833538247
5	9.png	582899	117524	222787	79.8380165345969	61.77948495365406
6	5.png	785610	291705	288386	62.86898079199603	63.29145504766996
7	2.png	617995	261397	221222	57.7024085955388	64.20327025299557
8	10.png	593463	174171	228518	70.65175082524101	61.49414538058817
9	1.png	736501	235530	288012	68.02041001980989	60.89455411465836
10	8.png	788470	241369	307807	69.38767486397707	60.9614823645795

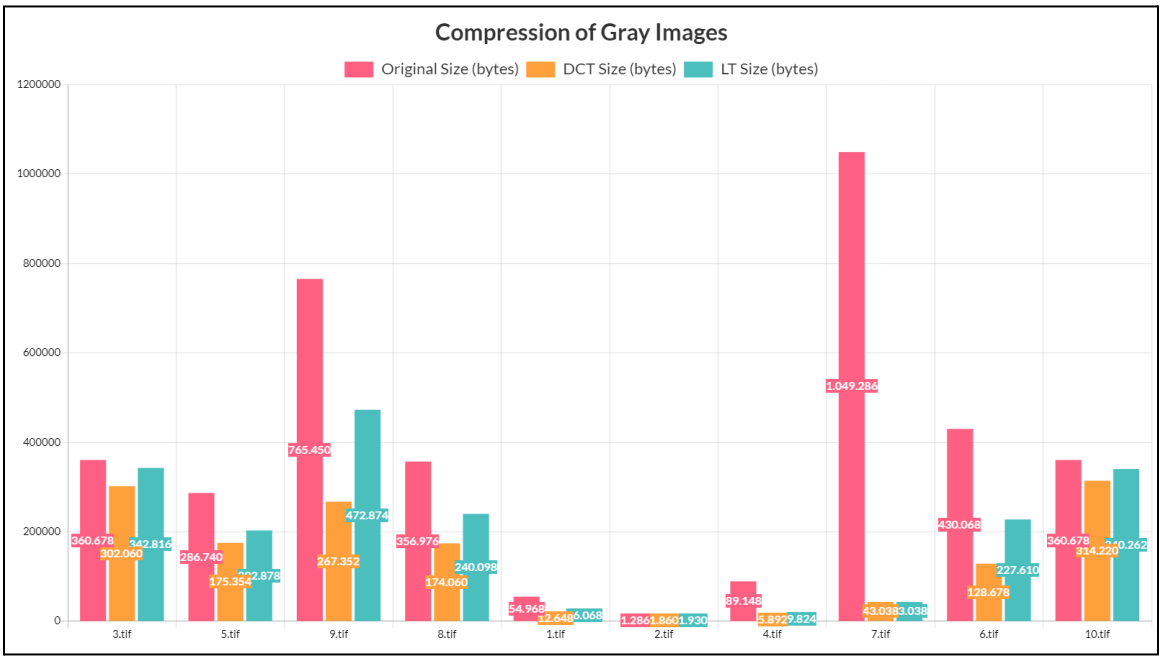
Compression Ratios: Color Images

The data presented in the table compares the sizes and compression ratios of the grayscale images transformed using DCT and Lapped transforms.

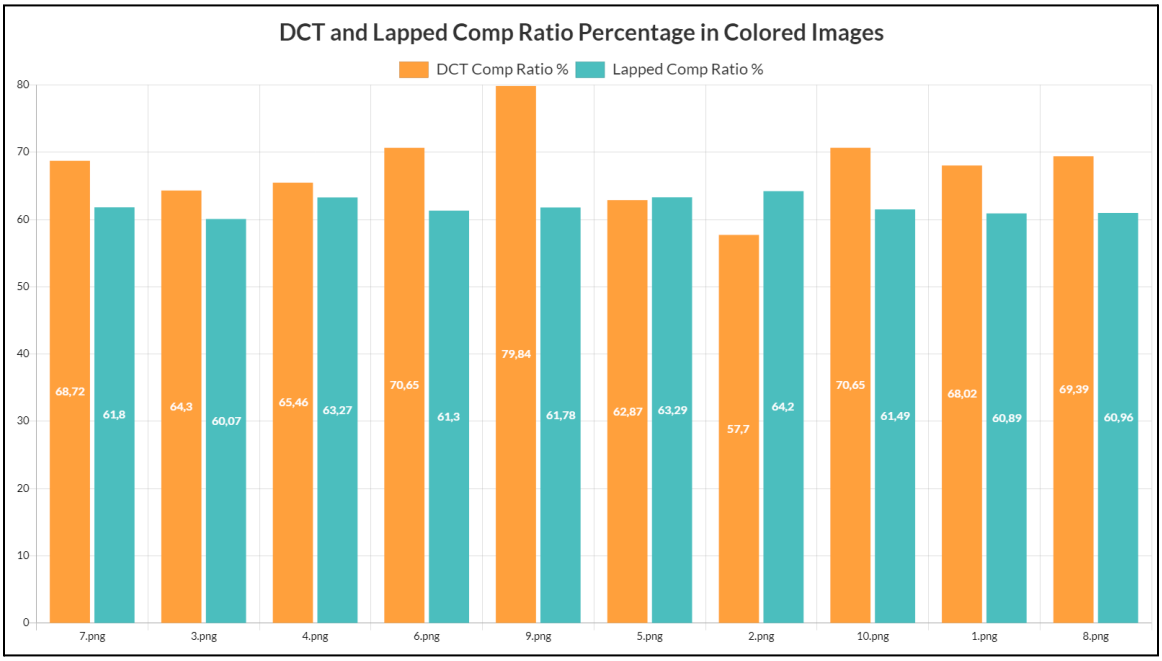
	Filename	Original Size (bytes)	DCT Size (bytes)	LT Size (bytes)	DCT Comp Ratio %	Lapped Comp Ratio %
1	3.tif	360678	302060	342816	16.252169525172036	4.9523397601184485
2	5.tif	286740	175354	202878	38.84564413754621	29.24670433145009
3	9.tif	765450	267352	472874	65.07257168985564	38.222744790646026
4	8.tif	356976	174060	240098	51.24041952400162	32.7411366590471
5	1.tif	54968	12648	26068	76.99024887207102	52.57604424392374
6	2.tif	1286	1860	1930	-44.63452566096424	-50.077760497667185
7	4.tif	89148	5892	9824	93.39076591735092	88.98012294162517
8	7.tif	1049286	43038	43038	95.89835373768449	95.89835373768449
9	6.tif	430068	128678	227610	70.0796153166476	47.07581126705544
10	10.tif	360678	314220	340262	12.880741270607018	5.6604505958223115

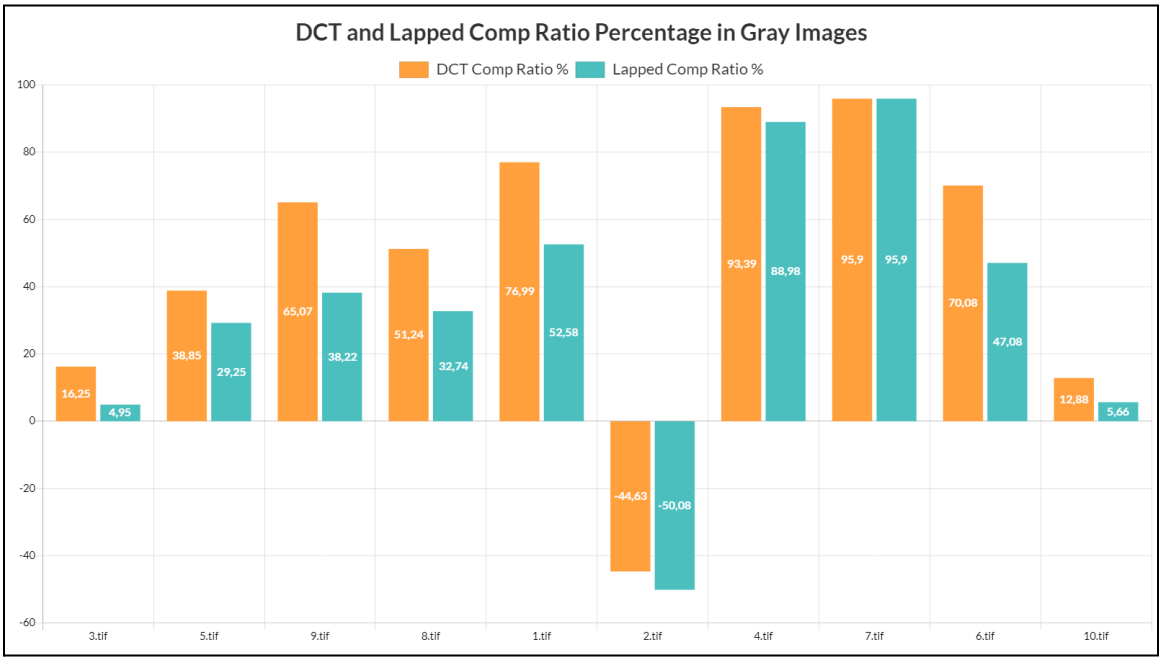
Compression in File Sizes





Compression in Percentage





Learned Perceptual Image Patch Similarity (LPIPS)

- **Definition**: LPIPS is a metric used to measure the perceptual similarity between two images. Unlike traditional metrics like MSE or PSNR, which operate on pixel-level differences, LPIPS evaluates the perceptual differences by comparing deep feature representations extracted from a neural network trained on image recognition tasks.

Calculation

- LPIPS uses a pre-trained neural network to extract feature maps from different layers for both the reference (original) and distorted (compressed) images. The difference is weighted and averaged to produce a single LPIPS score, which represents the perceptual similarity. A lower LPIPS value indicates higher perceptual similarity.

Code Logic

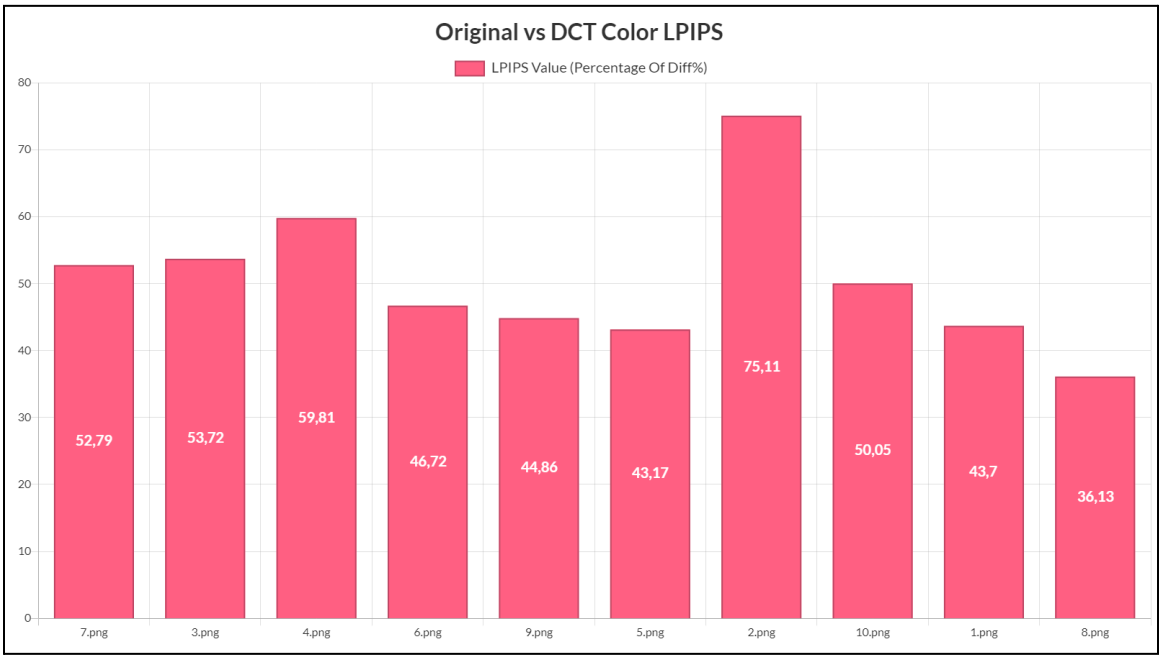
1. **Initialization**
    - The function initializes `total_lpips` to accumulate the LPIPS scores and `image_count` to count the number of processed images. An empty list `data` is created to store LPIPS values for each image.
2. **Iterate Over Original Images**
    - The function iterates through the files in the `original_path` directory. For each file, it checks if there is a corresponding compressed image in the `compressed_path` directory.
3. **Load and Transform Images**
    - Both the original and compressed images are loaded and converted to RGB format. The images are then transformed into tensors suitable for the neural network using a pre-defined `transform` function.
4. **Calculate LPIPS**
    - The LPIPS score between the original and compressed images is computed using the `lpips_model`, which outputs a tensor with the perceptual similarity score.
    - The score is extracted and added to `total_lpips`, and image count is incremented.
5. **Store Results**
    - For each image, the filename and its corresponding LPIPS score are stored in the `data` list as a dictionary.
6. **Create DataFrame**
    - A pandas DataFrame is created from the `data` list, summarizing the LPIPS scores for all processed images.
7. **Return**
    - The function returns the DataFrame, which can then be used for further analysis or comparison.

Comparison of DCT and Lapped Transform

LPIPS for Color Images (DCT and Lapped Transform)

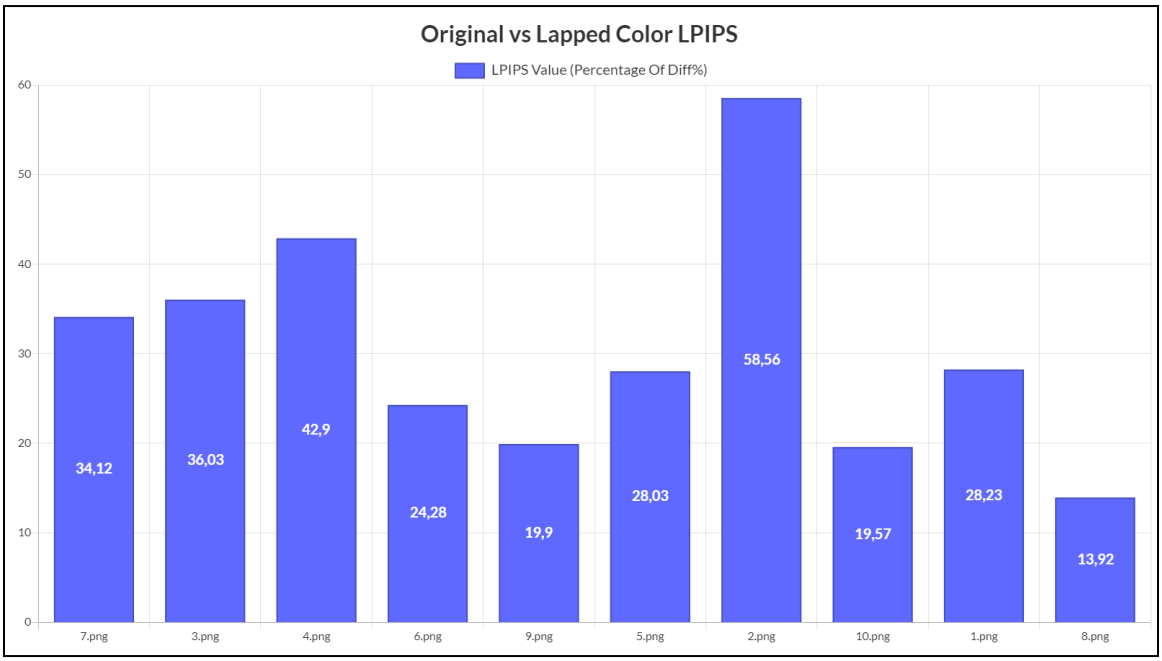
LPIPS Scores for Color Images DCT:

	File Name	LPIPS Value (Percentage Of Diff%)
0	7.png	52.786756
1	3.png	53.720617
2	4.png	59.807819
3	6.png	46.721593
4	9.png	44.856182
5	5.png	43.165866
6	2.png	75.108820
7	10.png	50.046206
8	1.png	43.696877
9	8.png	36.130676



LPIPS Scores for Color Images Lapped :

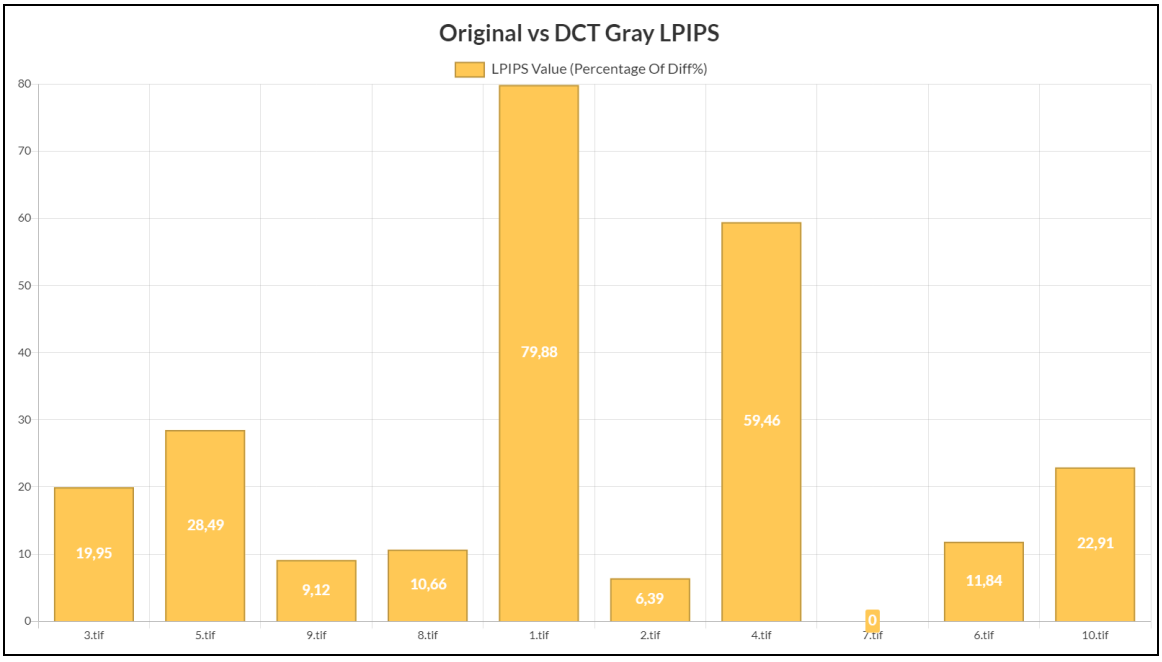
	File Name	LPIPS Value (Percentage Of Diff%)
0	7.png	34.117484
1	3.png	36.033899
2	4.png	42.898655
3	6.png	24.276394
4	9.png	19.903517
5	5.png	28.031820
6	2.png	58.564758
7	10.png	19.571954
8	1.png	28.234750
9	8.png	13.924675



LPIPS for Grayscale Images (DCT and Lapped Transform)

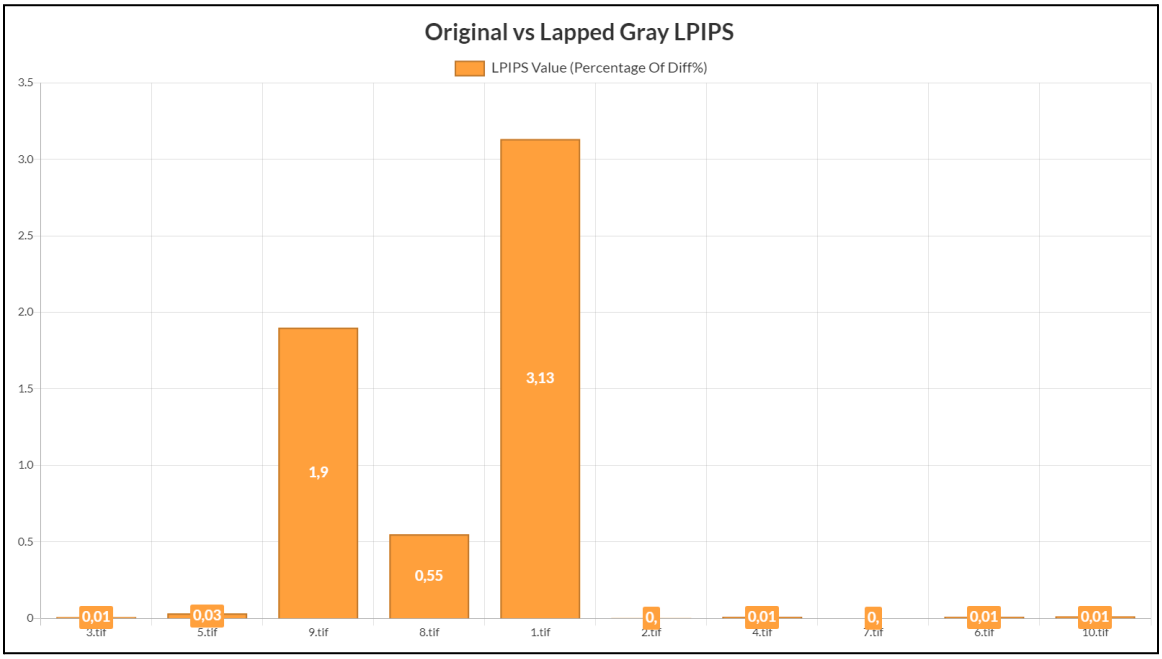
LPIPS Scores for Gray Images DCT:

	File Name	LPIPS Value (Percentage Of Diff%)
0	3.tif	19.953492
1	5.tif	28.491843
2	9.tif	9.121278
3	8.tif	10.662258
4	1.tif	79.879838
5	2.tif	6.394510
6	4.tif	59.455323
7	7.tif	0.000000
8	6.tif	11.840390
9	10.tif	22.912861



LPIPS Scores for Gray Images Lapped:

	File Name	LPIPS Value (Percentage Of Diff%)
0	3.tif	0.007921
1	5.tif	0.032098
2	9.tif	1.900384
3	8.tif	0.549822
4	1.tif	3.133356
5	2.tif	0.001509
6	4.tif	0.010007
7	7.tif	0.000426
8	6.tif	0.010335
9	10.tif	0.012815



Conclusion

This project aimed to evaluate and compare the performance of two image compression techniques: Discrete Cosine Transform (DCT) and Lapped Transform. The assessment focused on two primary metrics: compression ratio and Learned Perceptual Image Patch Similarity (LPIPS).

1. **Overall Findings:**
    - **Trade-offs**: The DCT method offers higher compression ratios but at the cost of potential visual artifacts. The Lapped Transform provides a good balance between compression and image quality, making it suitable for scenarios where preserving visual fidelity is crucial.

In conclusion, the choice between DCT and Lapped Transform compression methods depends on the specific requirements of the application, such as the acceptable level of quality degradation and the importance of file size reduction. For applications requiring high visual quality, Lapped Transform is the recommended approach, whereas DCT might be favored in scenarios prioritizing smaller file sizes.