Image Compressor

Code

```
im=imread('C:\Users\shekh\Documents\Projects\Image Compressor\input_img.jpg');
im = double(im)/255;
im = rgb2gray(im);
subplot(2, 1, 1)
imshow (im)
title("Original Image");
img_dct=dct2(im);
img_pow=(img_dct).^2;
img_pow=img_pow(:);
[B,index]=sort(img_pow);
B=flipud(B);
index=flipud (index);
compressed_dct=zeros(size(im));
coeff = 20000;
% maybe change the value
for k=1:coeff
compressed_dct(index(k))=img_dct(index(k));
end
im_dct=idct2(compressed_dct);
subplot(2, 1, 2)
imshow(im_dct)
title("Compressed Image");
% for saving this image use "imwrite" command
imwrite(im_dct,'compressed_imageDCT.jpg');
```

How To Run

1. Run the code using MATLAB.
2. Download or Prepare Your Image: Make sure you have an image named ‘input_image.jpg’ and adjust the im variable to point to your image file.

Explanation

MATLAB Code Explanation with Relevant Theory

```matlab
im = imread('C:\Users\shekh\Documents\Projects\Image
Compressor\input_img.jpg');
im = double(im) / 255;  % Convert image to double precision and normalize to [0, 1]
im = rgb2gray(im);      % Convert the image to grayscale
```

```
subplot(2, 1, 1)
imshow(im)
title("Original Image");
```

1. **Reading and Preprocessing**:
   - `im = imread('C:\Users\shekh\Documents\Projects\Image Compressor\input_img.jpg');`: Reads the input image file.
   - `im = double(im) / 255;`: Converts the image to double precision and normalizes pixel values to the range [0, 1].
   - `im = rgb2gray(im);`: Converts the RGB image to grayscale, as DCT is typically applied on grayscale images for compression.

   Theory:
   - **Digital Images**: Digital images are typically represented as matrices of pixel values, where each pixel stores information about color (RGB values in color images or intensity in grayscale).
   - **Normalization**: Normalizing pixel values to [0, 1] is a common preprocessing step to ensure consistent numerical range for computations.

```matlab
% Perform 2D DCT (Discrete Cosine Transform) on the grayscale image
img_dct = dct2(im);
```

2. **Applying 2D DCT**:
   - `img_dct = dct2(im);`: Computes the 2D Discrete Cosine Transform (DCT) of the grayscale image.

   Theory:
   - **Discrete Cosine Transform (DCT)**: DCT is a widely used transform technique in signal and image processing, including JPEG image compression. It transforms a signal from spatial domain to frequency domain, where it represents the image in terms of frequency components (DCT coefficients).
   - **Image Compression with DCT**: In image compression, DCT is applied in blocks (typically 8x8 pixels). It transforms each block of pixels into its frequency components. By discarding or quantizing high-frequency components (which represent fine details), we can achieve compression while preserving perceptual quality to some extent.

```matlab
% Compute the power spectrum of the DCT coefficients
img_pow = (img_dct).^2;
img_pow = img_pow(:);  % Flatten the power spectrum into a column vector
```

```

3. **Computing Power Spectrum**:
   - `img_pow = (img_dct).^2;`: Computes the power spectrum of the DCT coefficients by squaring each coefficient.
   - `img_pow = img_pow(:);`: Flattens the 2D matrix of power spectrum values into a column vector.

   **Theory**:
   - **Power Spectrum**: The power spectrum represents the amount of energy in each frequency component after applying DCT. Higher values indicate higher energy (more significant components in terms of image details).
   - **Flattening**: Flattening into a vector makes it easier to manipulate and sort the DCT coefficients based on their energy levels for compression.

```matlab
% Sort the DCT coefficients in descending order of magnitude
[B, index] = sort(img_pow);
B = flipud(B);          % Flip to get descending order
index = flipud(index);  % Flip the index accordingly
```

4. **Sorting Coefficients**:
   - `sort(img_pow)`: Sorts the power spectrum values in ascending order.
   - `flipud(B)` and `flipud(index)`: Flips the sorted power spectrum values and indices to descending order.

   Theory:
   - **Sorting for Compression**: By sorting the DCT coefficients based on their magnitude (energy), we prioritize the coefficients that contribute most to the image structure. This allows us to retain the most significant components and discard less important ones for compression.

```matlab
% Set the number of coefficients to keep (compression ratio)
coeff = 20000;  % Adjust this value to control the compression ratio
```

5. **Setting Compression Ratio**:
   - `coeff = 20000;`: Specifies the number of top DCT coefficients to retain for compression.

   Theory:
```

- **Compression Ratio**: The compression ratio determines how much data (in terms of DCT coefficients) we retain versus discard. Higher `coeff` values retain more coefficients, resulting in higher image quality but less compression. Lower values achieve higher compression but may introduce noticeable artifacts.

```matlab
% Compress the image by retaining only the top coefficients
compressed_dct = zeros(size(im));
for k = 1:coeff
    compressed_dct(index(k)) = img_dct(index(k));
end
```

6. **Compressing Image**:
   - `compressed_dct(index(k)) = img_dct(index(k));`: Copies the top `coeff` DCT coefficients from `img_dct` to `compressed_dct`.

   Theory:
   - **Compression Process**: By storing only the top DCT coefficients (`coeff` number of coefficients), we discard the least significant components. This significantly reduces the amount of data required to represent the image while attempting to preserve image quality to a perceptually acceptable level.

```matlab
% Perform inverse DCT to reconstruct the compressed image
im_dct = idct2(compressed_dct);
```

7. **Inverse DCT and Reconstruction**:
   - `im_dct = idct2(compressed_dct);`: Computes the inverse 2D DCT to reconstruct the compressed image.

   Theory:
   - **Inverse DCT**: Inverse DCT reverses the DCT process, transforming frequency components back to spatial domain (original image space). It reconstructs the image from the compressed DCT coefficients.

```matlab
subplot(2, 1, 2)
imshow(im_dct)
title("Compressed Image");

% Save the compressed image using imwrite command
imwrite(im_dct, 'compressed_imageDCT.jpg');
```

```

8. **Displaying and Saving Compressed Image**:
   - `subplot(2, 1, 2)`: Sets up the second subplot for displaying the compressed image.
   - `imshow(im_dct)`: Displays the reconstructed compressed image.
   - `title("Compressed Image")`: Sets the title for the subplot.
   - `imwrite(im_dct, 'compressed_imageDCT.jpg')`: Saves the reconstructed compressed image as `compressed_imageDCT.jpg`.

   **Theory**:
   - **Visualization**: Displaying both the original and compressed images side by side allows visual comparison of the compression quality.
   - **Saving**: Saving the compressed image allows further analysis or distribution.

### Summary:

This MATLAB script demonstrates a basic image compression technique using Discrete Cosine Transform (DCT). It applies DCT to the grayscale version of the input image, sorts DCT coefficients based on their energy levels, retains the top coefficients, and reconstructs the compressed image using inverse DCT. Adjusting the `coeff` parameter controls the compression ratio, affecting the trade-off between image quality and compression level. DCT-based compression is widely used in image formats like JPEG due to its efficiency in preserving perceptual quality while achieving significant data reduction.
```