JPEG compression

Lungoci Luca

30434

Robert Varga

25.05.2022

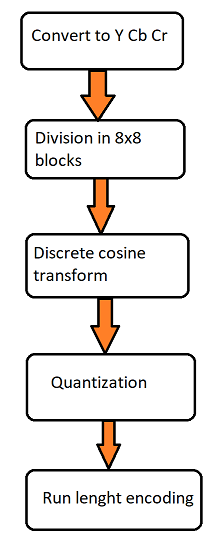
**1. Introduction**

This project implements the JPEG compression and decompression of a BMP image. This kind of compression is lossy but it very efficient in terms of size reduction.

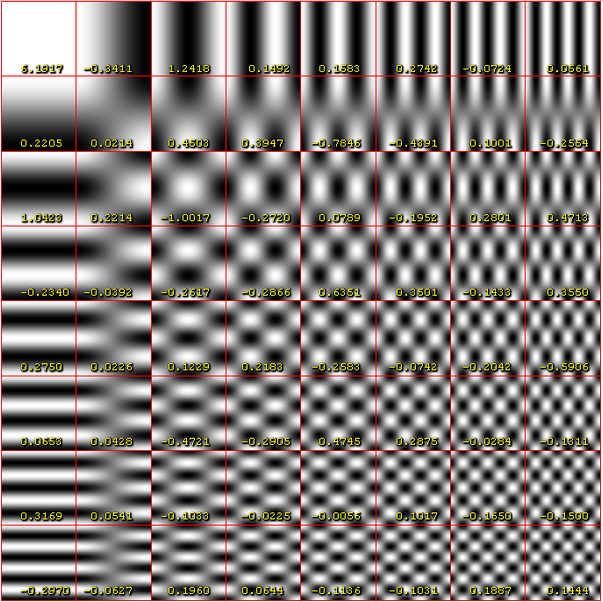
**2. State of the art**

Before writing code, I watched some clips explaining the process. I leaned how biological facts made this compression efficient. The human eye has more cells that are responsible for detecting light than cells responsible for colour. There was also explained how the high frequency part of images can be left out when doing the compression because the human eye will not notice them.

**3. Proposed method**



1. The first step is to convert the RGB image in YCC format (luminance, blue & red crominance). I also included in this step the reduction of the crominance matrices by 4 since color is less important for the human eye than light intensity. One pixel of the reduced matrix is the arithmetic average of 4 elements of the original one.
2. The second step is dividing the matrixes in 8x8 blocks.
3. Next we aplly discrete cosine transformation. What this step does is giving us a representation of the block in the form of how much we use from each frequency.

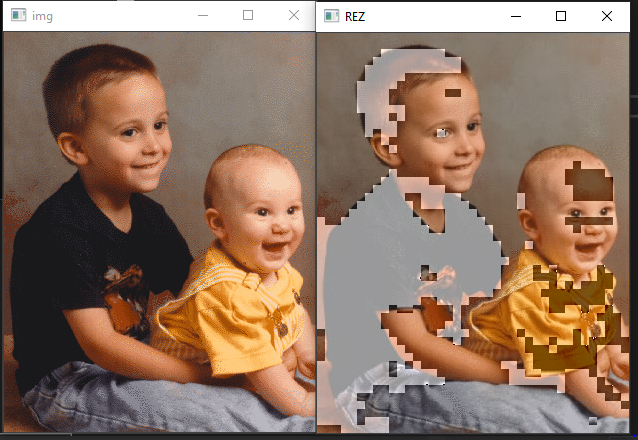


1. In the resulted matrix, each value is divided. The lower frequencies are divided by lower values than the high frequency. This is because the human eye can’t really observe those high frequencies.
2. After the previous quantization step, the resulted matrix has many 0 values in the left lower part. This is great for performing a run length encoding. This is the step where the significant size reduction happens.

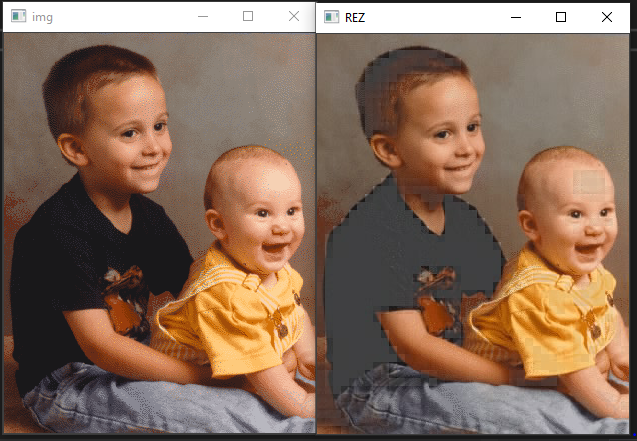
**4. Experimental results**

The main testing happened when picking values for the quantization part and picking sizes for the values saved in the file.

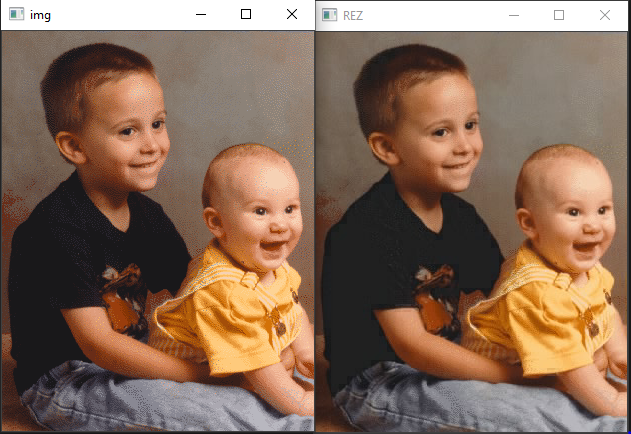
The compressed files contains an array with the run length encoding, pairs representing the value and the times it appears. The times a value appears is a small value witch can be represented on a byte. The value itself needed to be represented on two bytes, but if we stored the value on to bytes the compression would be inefficient.



In the picture above can be seen the result after storing the value and the times it appears on one byte. If we take a closer look, the result image has issues where the original image has low values or high values (for example the black shirt or the illuminated face of the baby). This issue appears as a result of the overflow, because of storing the values on one byte.



On the picture above there can be seen the first solution to the overflow issue: saturation. The low values are brought to -128 and the high ones to 127 so they can be written on a single byte. Even though the picture looks better, it is still pretty far away from looking like the original picture.



The final fix was done in the quantization part of the process, higher values were choosen for dividing the input 8x8 matrix. This way the values obtained in the end are lower and can closer to the values that can be written on a byte.

**5. Conclusions**

JPEG compression is a very useful process that enables us to reduce the significantly the size of the image we want to save. Even though this huge reduction in space may suggest a noticeable reduction in quality, the resulted image looks almost the same as the original image. This happens because the process removes attributes of the image that the human eye doesn’t really notice.

**Bibliography**

https://www.youtube.com/watch?v=Kv1Hiv3ox8I