The University of Melbourne
School of Computing and Information Systems
COMP90086 Computer Vision, 2023 Semester 2

Assignment 1: Image Filtering and Compression

Due:	7pm, 18 Aug 2023
Submission:	Source code (in Jupyter Notebook) and written responses (as .pdf)
Marks:	The assignment will be marked out of 6 points, and will contribute 6% of your total mark.

In this assignment, you will implement a classic image compression method, Laplacian pyramid compression.[1] This compression method builds a multiscale representation of an image called a Laplacian pyramid, show in Figure 1. It then compresses the higher spatial frequency parts of this representation (left parts of Figure 1) by quantizing them into fewer pixel values, which can be stored in fewer than 8 bits.

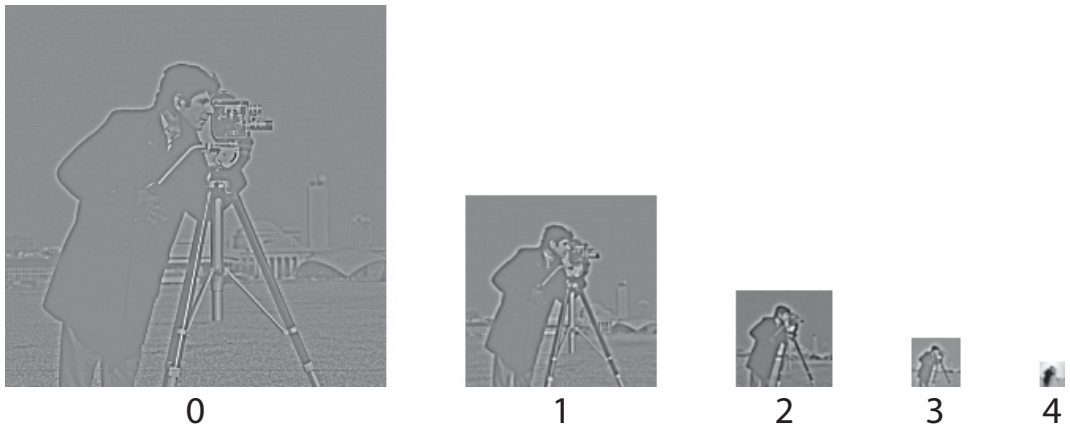

Figure 1: Laplacian pyramid representation of the "cameraman" test image.

How does this compress the image? Suppose we have a 256 x 256 pixel image. Represented as a unit8 (8-bit) image its size is 524,288 bits. Alternatively, we could represent this image as 7-level Laplacian pyramid and encode each level i with $i + 2$ bits. Each level of the pyramid has half the width/height of the previous level, so the total size is:

$$\sum_{i=0}^{6} \frac{256}{2^i} \times \frac{256}{2^i} \times (i+2) \tag{1}$$

This sums to 203,840 bits[2], which means the image has been compressed to about 40% of its

[1] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," in *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532-540, April 1983.

[2] If using equal-frequency bins for quantization, we would also need to save the pixel values corresponding to each bin, but this adds only a small amount of extra data, e.g. 32*(4+8+16+32+64+128) = 8064 bits if representing these as floats.

original size. (Unfortunately, we can't actually implement this memory saving in Python, because Python doesn't have uint datatypes smaller than uint8; instead, we will just simulate the effects.)

1. Laplacian image pyramid [2 pt]

Write a pair of functions to (a) create a Laplacian pyramid from an image and (b) reconstruct an image from its Laplacian pyramid. The pyramid construction function should take as input a grayscale image I and a number of levels n and output the set of n Laplacian pyramid images $\{L_0, L_1, ...L_{n-1}\}$. The reconstruction function should take the pyramid as input and output an image I.

The Laplacian pyramid of an image is created by progressively downsampling the image. The first $n-1$ layers encode the errors in the downsampling process. These layers are created by first downsampling the image to half its width/height, then upsampling it back to its original width/height. The error is computed by subtracting this recreated image from the original. The final layer of the Laplacian pyramid is just the final downsampled image. The original image can be recreated from the pyramid by, starting from the last layer, upsampling the layer and adding it to the previous layer.

The steps for creating the Laplacian pyramid are:

for i from 0 to $n-1$ **do**
  if $i < n-1$ **then**
    Filter I with the Gaussian kernel g
    Downsample the filtered image to half its original width/height by taking every other row/column.
Call the downsampled image D
    Upsample D by injecting zeros into every other row/column. Call the upsampled image U
    Filter U with the 5x5 kernel $4 \times g$
    $L_i \leftarrow I - U$
    $I \leftarrow D$
  else
    $L_i \leftarrow I$
  end if
end for

The steps for recreating the image from the pyramid are:

$I \leftarrow L_{n-1}$
for i from $n-2$ to 0 **do**
  Upsample I by injecting zeros into every other row/column. Call the upsampled image U
  Filter U with the kernel $4 \times g$
  $I \leftarrow U + L_i$
end for

The kernel g for filtering opertions is shown below. All filtering operations should return an output the same size as the original.

$$g = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{2}$$

The reconstructed image should match the original. In your write-up, show the Laplacian pyramid for an example image, and the reconstructed image obtained from that pyramid.

2. Quantization and compression [3 pt]

Write a function to quantize an image I into b bits. You can assume b is an integer in the range $[1, 7]$. To quantize an image, divide its pixel values into b^2 bins such that each bin contains approximately the same number of pixels, and then replace all pixels within a given bin with the mean pixel value in that bin.

Combine your quantization function with your code from question 1 to compress an image: generate the Laplacian pyramid of the image, compress some levels by quantizing them, and reconstruct a new image from the compressed pyramid. The number of pyramid levels and bits per level are free parameters – try to find values that will give a good balance between compression and image quality. As a rule, you should leave the last level of the pyramid (level $n - 1$) uncompressed and apply more compression to lower levels (with level 0 being the most compressed).

In your write up, discuss what happens to the reconstructed image as you increase the compression, using figures to support your answer. State the parameters that you feel give a good balance between compression and image quality.

3. Evaluation in the frequency domain [1 pt]

Evaluate the effect of this compression method in the frequency domain by computing the Fourier transform of an image (a) before and (b) after compression. In your write-up, explain any differences you observe in the magnitude and/or phase of the image before and after compression. Include a figure to illustrate your answer.

Tips

- Be aware of data types when creating (and visualising) your pyramid. The levels of the Laplacian pyramid represent reconstruction error, so the pixel values are not expected to be integers and they may be negative.
- OpenCV has built-in functions to compute image pyramids – feel free to use these if you get stuck, but in order to get full marks on Question 1 you should implement your own Laplacian pyramid code following the pseudocode given above.
- Because the Fourier magnitude of an image typically has a very large range, it can be hard to visualise. You may find it easier to interpret changes if you visualise the log of the magnitude instead.

Submission

You should make two submissions on the LMS: your code and a short written report explaining your method and results. The response to each question should be no more than 400 words.

Submission will be made via the Canvas LMS. Please submit your code and written report separately under the **Assignment 1** link on Canvas.

- Your **code** submission should include the Jupyter Notebook (please use the provided template) with your code and any image files we will need to run your code. Please include the cell output in your notebook submission if possible.

- Your written **report** should be a .pdf with your answers to each of the questions. The report should address the questions posed in this assignment and include any images, diagrams, or tables required by the question.

Evaluation

Your submission will be marked on the correctness of your code/method, including the quality and efficiency of your code. For effciency, your image processing steps should use convolution and/or matrix operations (your code should NOT include loops that iterate over image pixels). You should use built-in Python functions where appropriate and use descriptive variable names. Your written report should clearly address the questions, and include all of the specific outputs required by the question (e.g., images, diagrams, tables, or responses to sub-questions).

Late submission

The submission mechanism will stay open for one week after the submission deadline. Late submissions will be penalised at 10% of the total possible mark per 24-hour period after the original deadline. Submissions will be closed 7 days (168 hours) after the published assignment deadline, and no further submissions will be accepted after this point.

Updates to the assignment specifications

If any changes or clarifications are made to the project specification, these will be posted on the LMS.

Academic misconduct

You are welcome — indeed encouraged — to collaborate with your peers in terms of the conceptualisation and framing of the problem. For example, we encourage you to discuss what the assignment specification is asking you to do, or what you would need to implement to be able to respond to a question.

However, sharing materials — for example, showing other students your code or colluding in writing responses to questions — or plagiarising existing code or material will be considered cheating. Your submission must be your own original, individual work. We will invoke University's Academic Misconduct policy (`http://academichonesty.unimelb.edu.au/policy.html`) where inappropriate levels of plagiarism or collusion are deemed to have taken place.