COL 783: Assignment 3

Pyramids and Wavelets

Due date: Friday, 11 October 2019

Overview

In this assignment you will explore the use of image pyramids and wavelet transforms for various applications including seamless blending, edge-preserving denoising, and image compression.

Requirements

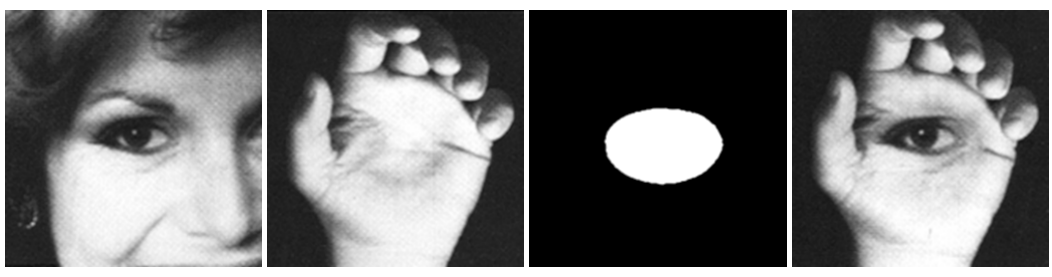
For simplicity, you may assume that the size of each image is a power of two. You are expected to make your implementations work for colour images as well.

Image blending using pyramids

Implement the Gaussian pyramid and the Laplacian pyramid as described by Burt and Adelson (1983). The output pyramid in both cases should be a list of images of smaller and smaller sizes. Further, implement a method to reconstruct the original image from the Laplacian pyramid.

As discussed in class, image pyramids can be used to seamlessly blend two images. To do so, you will need three images of the same size: the two images a and b to be blended, and a region image r defining which regions the two images should contribute to. Construct the Laplacian pyramids of a and b, blend their corresponding levels using the Gaussian pyramid of r, and reconstruct the blended image. See Section 3.2 of Burt and Adelson's paper for details.

You may use the following images from the Burt and Adelson paper as test inputs to try out your implementation. However, you should also include some creative results on a pair of images of your own choice. For example, you could try swapping your face with that of another student. You can use an image editing software like GIMP to manually paint the mask image.



Haar wavelets and denoising

Implement the Haar wavelet transform for images, which recursively decomposes an image into approximation and detail coefficients. The output should be in the form of a single 2D array containing the coefficients at all scales. Ensure that the transform is orthogonal, i.e. the sum of squared values is the same before and after the transform; this may require scaling the coefficients by $\sqrt{2}$ at each step. Also implement the inverse Haar transform, which reconstructs the original image from the wavelet coefficients.

Multiresolution representations such as pyramids and wavelets can be used for noise removal. For natural images, most of the detail coefficients are close to zero (due to smooth regions) while a few are quite large (due to edges). Therefore, suppressing low-amplitude detail coefficients while retaining high-amplitude ones is likely to reduce noise in smooth regions without significantly affecting edges and other important image features. Two common strategies to do so are hard thresholding and soft thresholding,

$$f_{\text{hard}}(w) = \begin{cases} w & \text{if } |w| > t, \\ 0 & \text{otherwise,} \end{cases} \qquad f_{\text{soft}}(w) = \begin{cases} w - t & \text{if } w > t, \\ 0 & \text{if } -t \le w \le t, \\ w + t & \text{if } w < -t, \end{cases}$$

where t is a threshold parameter. (Simoncelli and Adelson (1996) discuss a theoretical justification for such thresholding strategies.)

Choose a test image and add synthetic Gaussian noise, similar to the denoising component of the previous assignment. Compute its Haar wavelet transform, and perform hard and soft thresholding of the detail coefficients. Evaluate the quality of the reconstructed image via the SNR or PSNR. Then, do the same using the Laplacian pyramid instead. Here you may have to restrict thresholding to only the lowest few levels, or vary the threshold depending on the pyramid level to get reasonable results. Comment on the quality of the results obtained using the two methods.

Simple image compression

The fact that most detail coefficients are close to zero also suggests a simple method for lossy image compression: simply retain only the top $k\%$ wavelet coefficients, and replace the rest with zeroes. Then the wavelet representation will contain very long runs of zeroes, which can be compressed easily using run length encoding.

Implement a function that takes an image and a compression parameter $k \in [0, 100]$, and writes a compressed version of the image to a file in a binary format of your design. You are not required to perform any bit-level coding techniques like Huffman coding, but you should ensure that the output file size in bytes is approximately proportional to k. Of course, you also have to implement a decompression function that will read your compressed file and produce a lossy reconstruction of the original image.

Describe your compression scheme in your report, and show results on a few example images of your choice. Include at least one photographic image, and one image with large regions of constant colour like a diagram or a cartoon. (The latter are likely to compress better, since we are using Haar wavelets which are piecewise constant.) For various compression parameters, report the size of the compressed file in bytes, and the SNR or PSNR of the decompressed image.

Extra credit

Option 1: Study Swelden's article "Wavelets and the Lifting Scheme: A 5 Minute Tour" and implement the wavelet transform described therein. Show results of denoising and image compression using these, and discuss whether they perform better than Haar wavelets for photographic images.

Option 2: Implement some subset of Sunkavalli et al.'s "Multi-Scale Image Harmonization" (2010), which applies histogram matching to the Laplacian pyramids of two images in order to better match their visual appearance at different scales.

Submission and Evaluation

As usual, submit your code and a report describing your work and any relevant design and implementation choices. Include images demonstrating the results of each task of the assignment. You will be graded on both your working code and the comprehensiveness of the report.