Haar Transform

Dr. François Pitié
Electronic and Electrical Engineering Dept.
pitief@tcd.ie www.sigmedia.tv

Instructions: You are required to compile a report that answers specific instructions included in the report. Include relevant code and output figures in your report.

Requirements: Labs Reports should be typed up and submitted electronically using the **PDF file format** via the module's blackboard page by the specified deadline. Remember to put your name and student number on the top of your reports.

You must have read and understood the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at: https://www.tcd.ie/calendar.

Introduction

In this lab, we are going to look at the Haar transform and how it performs as a image transform for compression. The lab will cover the concepts of entropy, quantisation and Rate-Distortion curve.

We will be working with the image (`kodim19_512.png`). Exceptionally, you are advised to work with doubles but NOT to rescale the values between 0-1:

```
I = double(imread('kodim19_512.png'));
```

1 Image Entropy

The entropy is a fundamental measure of compression that can be simply derived from the image histogram.

Q 1.1 *Write a MATLAB function* `calcEntropy` *that calculates the Entropy of an image. The function definition is below:*

```
function entropy = calcEntropy(Y)
% This function takes as input a 2D array Y containing
% the image intensities and returns the entropy.
```

Save this function in a m-script called `calcEntropy.m` *to allow you to call the function by the name* `calcEntropy`. *With this function in your m-file you can now calculate entropies simply by doing* `e = calcEntropy(Y)` *for instance.*

Your first step will be to calculate the histogram of the greyscale image `Y`. *The*

histogram should contain 2048 bins ranging from the minimum value of the input image to the maximum value of the input image. Then use this histogram to compute the entropy in bits/pixel.

Remember that in MATLAB, `log2(x)` *returns the logarithm to the base 2 of* x*. If* x *is a vector it returns a vector containing the* $\log_2$ *result for each element. Also, by definition* $0 \times \log(0) = 0$*, so make sure your code does the same.*

Q 1.2 *Quantise the image using a quantisation step of* $Q_{step} = 15$*:*

```
pic = round(pic/Q_step);
```

Estimate the entropy (H_{qi}*) of the quantised image. (Again, make sure that* `I` *takes values in 0-255, and not 0-1, as otherwise the quantisation step would be incorrect)*

Q 1.3 *Explain any difference between* H_{qi} *and* H_o*.*

Q 1.4 *Using the MATLAB* `psnr` *function, compute the PSNR between the quantised and unquantised images. Read carefully the function help and make sure that the PSNR is in a reasonable range (e.g. between 20dB and 50dB).*

Q 1.5 *Comment on the visual quality of the quantised and unquantised images.*

2 The 2D Haar Transform

The basic idea in Transform image coding is to provide a compact representation for the image data. You will verify this with the 2D Haar Transform in this section.

Q 2.1 *Write a MATLAB function that implements the 1-level Haar Transform and outputs an image of its subbands. The function definition should be as follows:*

```
function H = calcHaarLevel1(Y)
% This function takes as input a 2D array Y containing
% the image intensities of a picture and returns the 1-level
% Haar Transform
```

You can test that your function works by checking the results on the provided test image (`load test.mat`*). You can also look at the provided* `calcInvHaar` *function to get some inspiration.*

Q 2.2 *Apply a quantisation step size of* $Q_{step} = 15$ *to the Haar transform of the image and calculate the resulting Entropy* H_{qhaar}*. The entropy averaged over the subbands can be computed with the provided* `calcEntropyHaar(H,n)` *function, where* `H` *is the input Haar image and* `n` *is the number of levels of the Haar transform (in this case* `n=1`*). This function requires your* `calcEntropy` *function.*

Q 2.3 *Is* $H_{qhaar} < H_{qi}$*? Why?*

Q 2.4 *Using the* `calcInvHaar(H,n)` *for a level* `n=1`*, show the decoded Haar image at* $Q_{step} = 15$ *and visually compare it against the original image.*

3 Rate-Distortion Curve

To properly measure the compression ability of the Haar transform, we need to measure the entropy of the Haar transform at different quantisation levels. The entropy alone is however not enough when dealing with lossy compression. We also need to record a quality metric, such as the MSE or the PSNR. Indeed, a solid black image would have minimal entropy but would be probably very dissimilar to the original image.

The proper approach is thus to plot what we call the Rate-Distortion curve. Changing the value of the quantisation step allow us to obtain different pairs of entropy and PSNR values. The Rate (in our case the entropy) - Distortion (in our case the PSNR) curve is the curve that connects these operating points.

In the rest of the lab, the operating points of the R-D curves will be obtain for values of Q_{step} in $\{2, 4, 8, 16, 32, 64, 128\}$.

Q 3.1 *Graph the R-D curve for the original image, without Haar transform. To do this, simply compute the entropy and psnr for the proposed levels of quantisation on the original image.*

Assuming that the entropy and psnr values are stored in the vectors `e` *and* `p`*, you can obtain the PSNR graph with this code:*

```
figure; plot(e, p, '-+');
xlabel('Entropy bits/pixel');
ylabel('PSNR (dB)');
title('R-D curve for no transformation');
```

Q 3.2 *Add to the graph the R-D curve for the level 1 Haar 2D transform (and add a legend to the plot).*

To do this, quantise the Haar transform using the same Q_{step} for all the coefficients. The entropy averaged over the subbands can be computed with the provided `calcEntropyHaar` *function. To compute the psnr, you must first decode the quantised images using the provided* `calcInvHaar` *function. Make sure that the decoded images resemble the original image.*

Q 3.3 *Looking at the RD-curves graph, is the level 1 Haar 2D transform more efficient at compressing the image than no transformation?*

4 The Multi-Level 2D Haar Transform

Q 4.1 *Write a MATLAB function that implements the n-level Haar Transform and outputs a image of its subbands. The function definition should be as follows:*

```
function H = calcHaar(Y, n)
% This function takes as input a 2D array Y containing
% the image intensities of a picture and returns the 1-level
% Haar Transform
% n is the number of levels used.
```

To implement this function you can adapt your level 1 implementation and recursively call the Haar Transform on the LoLo subband. You can test that your function works by checking the results on the provided test image.

Q 4.2 *Plot in a single graph (with title and legend) the RD-curves for the levels 1 to 5 of the Haar transform. Is using a higher levels of Haar transform beneficial?*