Image Compression

Examples
Example 1 – Huffman code

• Find Huffman code for

<table>
<thead>
<tr>
<th>Gray Level</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Algorithm:
(1) Find the gray level probabilities
(2) Sort the probabilities
(3) Combine the smallest two by addition
(4) Repeat steps 2-3 until only two are left
(5) Work backward, generate code

Lavg = ?
H = ?
Matlab

• Function to compute Huffman code
  – huffman.m (on course website)
  – from http://users.ecs.soton.ac.uk/rm/resources/matlabhuffman

\[
\text{codewords} = \text{huffman} (\text{probabilities});
\]

Output: a cell array (1..n) containing the codewords for the symbols
Input parameter: an array (1..n) of the symbol probabilities

• A cell array is a data type with indexed data containers called cells. Each cell can contain any type of data.
• You access an element of the array using curly brackets; e.g., “codewords{1}”
• In our case, each element of “codewords” is an array of 0’s and 1’s
clear all
close all

% Define probabilities
probabilities = [0.05, 0, 0.1, 0.25, 0.4, 0.2];  % Simple test example

% Call Huffman coding function
codewords = huffman(probabilities);

disp('The symbols, their probabilities and the allocated Huffman codewords are:');
% For each codeword...
for index = 1:length(codewords)
    % ...display its bits in reverse order
    disp([num2str(index), '    ', num2str(probabilities(index)), '    ', num2str(codewords{index}(length(codewords{index}):-1:1))]);
end

% Calculate the symbol entropy
entropy = sum(probabilities.*log2(1./probabilities));
entropy = 0;
for index = 1:length(codewords)
    if probabilities(index) > 0
        entropy = entropy + probabilities(index)*log2(1./probabilities(index));
    end
end

% Calculate the average Huffman codeword length
av_length = 0;
for index = 1:length(codewords)
    av_length = av_length + probabilities(index)*length(codewords{index});
end

disp(['The symbol entropy is: ', num2str(entropy)]);
disp(['The average Huffman codeword length is: ', num2str(av_length)]);
Try on a grayscale image

• Determine probabilities from the histogram

% Real image
I = imread('cameraman.tif');
imshow(I,[]);
figure, imhist(I);
[H,r] = imhist(I);
probabilities = H/(size(I,1)*size(I,2));
Try on a Laplacian image

- Take the image at lowest level of the Laplacian pyramid
  - It is just the original image minus a low passed version
  - Determine the probabilities from the histogram

```matlab
% Laplacian image
I = imread('cameraman.tif');
I = double(I);
G1 = imresize(I, 0.5); % apply lowpass, then scale by 0.5
G11 = imresize(G1, 2, 'nearest'); % scale up by 2
L = I - G11; % residual image
L = int8(L);
imshow(L,[]);
figure, imhist(L);
[H,r] = imhist(L);
probabilities = H/(size(L,1)*size(L,2));
```
Example 3 - Predictive Coding

- Takes advantage of interpixel redundancy
- Predict next pixel from previous pixel, encode only the difference from the actual and the predicted

**Figure 8.19** A lossless predictive coding model: (a) encoder; (b) decoder.
Matlab code – predictive coding

clear all
close all

I = double(imread('cameraman.tif'));

% Do a lossless predictor algorithm, that predicts each pixel I(x,y) as
% simply the value of the previous pixel I(x-1,y), and transmits only the
% error between the predicted value and the actual value.

R = zeros(size(I)); % Output prediction residuals image
for c=2:size(I,2)
    R(:,c) = I(:,c) - I(:,c-1);
end
R = int8(R);
figure, imshow(R, []), impixelinfo
figure, imhist(R);

[H,r] = imhist(R);
probabilities = H/(size(R,1)*size(R,2));

- Entropy of residual image?
- Bit length if you encode using Huffman?
Example 4 – Run Length Encoding

- Function rle(x) from Mathworks website

```matlab
function data = rle(x)
% data = rle(x) (de)compresses the data with the RLE-Algorithm
% Compression:
% if x is a numbervector data{1} contains the values
% and data{2} contains the run lenths
%
% Decompression:
% if x is a cell array, data contains the uncompressed values
%
%   Version 1.0 by Stefan Eireiner (<a href="mailto:stefan-e@web.de?subject=rle">stefan-e@web.de</a>)
%   based on Code by Peter J. Acklam
%   last change 14.05.2004

if iscell(x) % decoding
    i = cumsum([ 1 x{2} ]);  
    j = zeros(1, i(end)-1);  
    j(i(1:end-1)) = 1;       
    data = x{1}(cumsum(j));
else % encoding
    if size(x,1) > size(x,2), x = x'; end % if x is a column vector, transpose
    i = [ find(x(1:end-1) ~= x(2:end)) length(x) ];
    data{2} = diff([ 0 i ]);
    data{1} = x(i);
end
```
clear all
close all

I = imread('circlesBrightDark.png');
[height, width] = size(I);
fprintf('Image is %dx%d pixels, total bytes = %d
', height, width, height*width);

I = double(I);
imshow(I, []);

% Call run length encoding function. The result is a 1x2 cell array, where
% result{1} is the list of run values, and result{2} is the corresponding
% list of run lengths.
result = rle(I(:));       % Need to pass in a row or column vector

runValues = result{1};
runLengths = result{2};
nRuns = size(runLengths);
fprintf('Number of runs = %d
', nRuns);
figure, hist(runValues), title('Histogram of run values');
figure, hist(runLengths), title('Histogram of run lengths');

% Calculate the total number of bits.
maxRunValue = max(runValues(:));
nbitsRunValue = ceil(log2(maxRunValue));       % #bits required for run values
fprintf('Maximum image value = %d, so %d bits are required.
', ...
        maxRunValue, nbitsRunValue);

maxRunLength = max(runLengths(:));
nbitsRunLength = ceil(log2(maxRunLength));    % #bits required for run lengths
fprintf('Maximum run length = %d, so %d bits are required.
', ...
        maxRunLength, nbitsRunLength);

totalBits = nRuns * nbitsRunValue + nRuns * nbitsRunLength;
fprintf('Total number of bits = %d
', totalBits);