GUJARAT TECHNOLOGICAL UNIVERSITY
Chandkheda, Ahmedabad.

Vishwakarma Government EngineeringCollege, Chandkheda

A
Project Report on

Data Compression of images through Quantization

Prepared as a part of the requirements for the subject of

ANALOG AND DIGITAL COMMUNICATION(3151104)
B. E. SEMESTER – V
ELECTRONICS & COMMUNICATION DEPARTMENT

Name : Sandeep Kumar

Enrollment No : 210170111015

Prof. Urvisha Fatak
(Faculty Guide)

Dr. Arun B. Nandurbarkar
(Head of the Department)

Academic year (2023-2024)

Vishwakarma Government Engineering College Chandkheda – Ahmedabad

Certificate

This is to certify that Mr./Ms. ________________________

Enrollment No.______________________, Student of B. E. Semester V ____branch

has satisfactorily Project work in the subject Analog and Digital Communication (3151104) within four walls of Vishwakarma Government Engineering College, Chandkheda – Ahmedabad during the academic year 2023-2024

He / She is eligible to appear in the University Examination.

Faculty in– Charge: HOD

Date: (EC Department)

CONTENTS

Sandeep Kumar
210170111015
Analog and Digital Communication

Data Compression of images through Quantization using huffmandict( )

- **Quantization**: Since continuous intensity values are not possible with digital signal processing, another step is required after spatial and temporal sampling. ***The conversion of continuous intensity data to discrete values is known as quantization***. It can be compared to a domain transformation from continuous to discrete. A quantizer is a quantization mapping that is distinct.

- **Compression** is the process of lowering the amount of data required to represent a file, picture, or video without sacrificing the original material's quality .

- It also decreases the amount of data that must be stored and/or sent. Compressing anything

- is limiting the amount of a piece of data or information.

- Compression may be done in a variety of methods, each with its own set of advantages and disadvantages. Reduce the amount of unessential bits in the data is one simple way. The second methodology allows prioritising the data and determining which portions should be omitted. Arithmetic coding is a lossless data compression technique.

- For the computers to understand an image which is captured

- by a light sensor, it first needs to be digitized. Final step for digitalization is **Quantization**.

Applications of Compression:

1) Bitmaps
2) Silence in audio data, or pauses in conversation etc.
3) Suppression of zeros in a file (Zero Length Suppression)
4) Backgrounds in images
5) Blanks in text or program source files
6) Other regular image or data tokens

Sandeep Kumar
210170111015

Quantization in MATLAB:

```
clc
close all
clear all

figure(1)
I = imread('C:\Users\Sandeep\Downloads\onepiecergb.jpe');
imshow(I)
axis off
title('Original Image')

Quantization_level = max(I(:));

keyboard

%Qunatization Level Reduction to 8

thresh8 = multithresh(I,7);

valuesMax8 = [thresh8 max(I(:))]

[quant8_I_max, index] = imquantize(I,thresh8,valuesMax8);

valuesMin8 = [min(I(:)) thresh8];

quant8_I_min = valuesMin8(index);

%Qunatization Level Reduction to 4

thresh4 = multithresh(I,3);

valuesMax4 = [thresh4 max(I(:))]

[quant4_I_max, index] = imquantize(I,thresh4,valuesMax4);

valuesMin3 = [min(I(:)) thresh4]

quant4_I_min = valuesMin3(index);
keyboard
figure(2)
multi = cat(4,quant8_I_min,quant8_I_max,quant4_I_min,quant4_I_max);
montage(multi); % montage plot
title('Minimum Interval Value           Maximum Interval Value')
ylabel('L = 4                            L = 8')
```

Sandeep Kumar
210170111015

Original Image


```
valuesMax8 = 1×8 uint8 row vector
    33    61    92   127   162   199   233   255
```

```
valuesMax4 = 1×4 uint8 row vector
    67   133   204   255
```

```
valuesMin3 = 1×4 uint8 row vector
     0    67   133   204
```


```
valuesMax4 = 1×4 uint8 row vector
    67   133   204   255
```

```
valuesMin3 = 1×4 uint8 row vector
     0    67   133   204
```


Sandeep Kumar
210170111015
Analog and Digital Communication

huffmandict( ) function of MATLAB:

Generate Huffman code dictionary for source with known probability model

Syntax

[dict,avglen] = huffmandict(symbols,prob)

[dict,avglen] = huffmandict(symbols,prob,N)

[dict,avglen] = huffmandict(symbols,prob,N,variance)

Description

[dict,avglen] = huffmandict(symbols,prob)

generates a binary Huffman code dictionary, dict, for the source symbols, symbols, by using the maximum variance algorithm. The input prob specifies the probability of occurrence for each of the input symbols. The length of prob must equal the length of symbols. The function also returns average codeword length avglen of the dictionary, weighted according to the probabilities in the input prob.

[dict,avglen] = huffmandict(symbols,prob,N)

generates an N-ary Huffman code dictionary using maximum variance algorithm. N must not exceed the number of source symbols.

[dict,avglen] = huffmandict(symbols,prob,N,variance)

generates an N-ary Huffman code dictionary with the specified variance.

MATLAB PROGRAM:

adc_huffman_messi.mlx | messi_Data_compression_using_huffman_coding.mlx | Untitled.mlx | +

```
% reading the image
Image=imread('C:\Users\Sandeep\Downloads\messi.jpg')
figure,imshow(Image);
% calculating the frequency of each pixels
[frequency,pixelValue]=imhist(Image());
% suming all the frequencies
tf = sum(frequency) ;
% calculating the probability of each pixel
probability = frequency ./ tf ;
% creating the dictionary
dict=huffmandict(pixelValue,probability);   %generates binary huffmann code dictionary
% converting the image pixels to 1D array
imageOneD = Image(:)    %column wise hoga
% encoding
testVal = imageOneD ;
encodedVal = huffmanenco(testVal,dict)
% decoding
decodedVal = huffmandeco(encodedVal,dict)
% displaying the length
```

```
kb = 8 * 1024 ;
disp(["before compression size is",numel(de2bi(testVal))/kb]) ;
disp(["after compression size is",numel(encodedVal)/kb]) ;
disp(["after reconstruction size is",numel(de2bi(decodedVal))/kb]) ;
% recovering the original image from 1D Array
[rows, columns, numberOfColorChannels]= size(Image);
oi = reshape(testVal,[rows, columns, numberOfColorChannels]) ;
imwrite(oi,'C:\Users\Sandeep\Downloads\messi.jpg');
% recovering the decoded image from 1D Array
decodedVal = uint8(decodedVal);
figure,imshow(decodedVal);
```

Sandeep Kumar
210170111015
Analog and Digital Communication

Explaination :

imhist( ) :

The **`imhist`** function creates a histogram plot by defining *n* equally spaced bins, each representing a range of data values, and then calculating the number of pixels within each range. You can use the information in a histogram to choose an appropriate enhancement operation.

huffmanenco( ) :

Encode sequence of symbols by Huffman encoding

Syntax

code = huffmanenco(sig,dict)

Description

code = huffmanenco(sig,dict) encodes input signal sig using the Huffman codes described by input code dictionary dict. sig can have the form of a vector, cell array, or alphanumeric cell array. If sig is a cell array, it must be either a row or a column. dict is an N-by-2 cell array, where *N* is the number of distinct possible symbols to encode. The first column of dict represents the distinct symbols and the second column represents the corresponding codewords. Each codeword is represented as a row vector, and no codeword in dict can be the prefix of any other codeword in dict. You can generate dict using the huffmandict function.

huffmandeco( ) :

Decode binary code by Huffman decoding

Sandeep Kumar
210170111015

Syntax:

sig = huffmandeco(code,dict)

Description

sig = huffmandeco(code,dict) decodes the numeric Huffman code vector, code, by using the Huffman codes described by input code dictionary dict. Input dict is an *N*-by-2 cell array, where *N* is the number of distinct possible symbols in the original signal that encodes code. The first column of dict represents the distinct symbols, and the second column represents the corresponding codewords. Each codeword is represented as a numeric row vector, and no codeword in dict can be the prefix of any other codeword in dict. You can generate dict by using the huffmandict function and code by using the huffmanenco function. If all symbols in dict are numeric, output sig is a vector. If any symbol in dict is alphabetic, sig is a one-dimensional cell array.

MATLAB OUTPUT:

```
Image = 675×1200×3 uint8 array
Image(:,:,1) =

    98   130   163   174   178   182   188   189   194   191   187   183   178   167   156   144   116    86    47    19    10     6     5     1     2
   101   130   161   172   178   184   190   190   193   190   187   183   178   167   155   143   112    83    45    19    10     8     7     3     2
   105   130   157   168   178   187   193   191   196   193   189   184   177   164   150   136   108    80    42    18    10     9     8     4     2
   109   130   153   165   178   189   195   192   199   196   191   185   176   160   143   128   106    78    41    17    10     8     8     3     2
   112   131   153   163   177   189   195   192   197   194   191   185   176   159   141   126   104    77    40    17     9     7     6     1     2
   114   133   154   164   177   188   195   192   191   190   188   184   177   160   143   127   102    75    39    16     9     6     5     1     2
   114   135   157   166   176   186   193   191   191   190   189   185   176   159   140   123    98    71    37    15     9     7     6     1     2
   114   136   160   168   176   185   192   191   196   195   192   186   175   155   134   116    94    68    35    14     9     8     8     3     2
     :
```

Sandeep Kumar
210170111015


```
imageOneD = 2430000×1 uint8 column vector
      98
     101
     105
     109
     112
     114
     114
     114
     111
     116
       ⋮

encodedVal = 14506678×1
       1
       1
       1
       0
       0
       0
       0
       0
       1
       1
       ⋮

decodedVal = 2430000×1
      98
     101
     105
     109
     112
     114
     114
     114
     111
     116
       ⋮

    "before compression size is"    "2373.0469"
    "after compression size is"    "1770.8347"
    "after reconstruction size is"    "2373.0469"
```

References:

https://in.mathworks.com/help/comm/ref/huffmandict.html

YOUTUBE

https://youtu.be/uTdBFr8Fn-w?si=41LC_ijvzUMg7R08

++