Image Compression

Sesi 10

Dosen Pembina :

Sriyani Violina

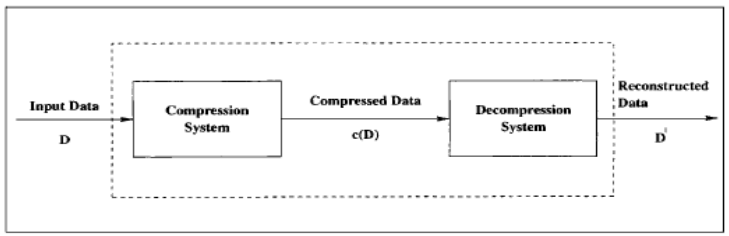
Danang Junaedi

Tujuan Kompresi *Image*

- Kompresi ➔ untuk apa?
  - Volume data yang besar
  - Bit rate tinggi ➔ *bandwidth* yang tinggi
  - Bayangkan sebuah video dengan resolusi 640x480 dengan 30 fps, dimana menggunakan penyimpanan 24-bit. Bila video berdurasi 1 jam berapa ukuran file video tersebut?

Image Compression

- Alternatif Solusi
  - Penambahan *storage* dan *bandwidth*
  - Kompresi data (*smart choice*…!!!)



Teknik kompresi yang diharapkan

- Proses kompresi/dekompresi yang cepat
- Membutuhkan *memory* yang kecil
- Kualitas citra kompresi yang bagus
- Proses transfer dan penyimpanannya mudah

Teknik Kompresi

- Berdasarkan hasilnya:
  - ***Lossless Compression****: based on the idea of breaking a file into a "smaller" form for transmission or storage and then putting it back together on the other end so it can be used again*. Ex : ZIP, PNG, GIF
  - ***Lossy Compression****: eliminate "unnecessary" bits of information, tailoring the file so that it is smaller*. Ex: JPEG, MP3, MPEG

Klasifikasi Teknik Kompresi

- *Entropy Encoding* (*Lossless*)
  - *Run Length Encoding* (RLE)
  - *Pattern Substitution*
  - Huffman
  - DPCM
- *Source Encoding* (*Lossy*)
  - *Quantizing Compression*
  - *Transfrom Encoding*
- *Hybrid Encoding* (*Lossy*)
  - JPEG

Run Length Encoding (RLE)

1	2	1	1	1	1
1	3	4	4	4	4
1	1	3	3	3	5
1	1	1	1	3	3

- Diubah dalam bentuk sekuensial
  ➔ 1 2 1 1 1 1 1 3 4 4 4 4 1 1 3 3 3 5 1 1 1 1 3 3 = 24 byte
- Dihitung jumlah kemunculan data
  ➔ (1,1) (2,1) (1,5) (3,1) (4,4) (1,2) (3,3) (5,1) (1,4) (3,2)
- Data Kompresi
  ➔ 1 1 2 1 1 5 3 1 4 4 1 2 3 3 5 1 1 4 3 2 = 20 byte

Shannon's *Source Coding Theorem*

- Pada umumnya teknik *lossless* akan melakukan proses penggantian simbol dengan suatu simbol biner
- Masalahnya:
  - Menentukan simbol biner sehingga data yang dikompresi menjadi lebih kecil dan dapat "dibalikkan" kembali ➔ harus unik

Shannon's *Source Coding Theorem*

Simbol	MODEL _A	MODEL_B	MODEL_C
a	00	0	0
b	01	10	1
c	10	110	00
d	11	111	01

- Misalkan
  - S = *aacabad*
- Model manakah yang dapat digunakan untuk *encode* S ??

Shannon's *Source Coding Theorem*

- Misalkan sebuah data:
  - A={a1,a2,…,an}
- Probabilitas data :
  - P={p1,p2,…,pn}
- Dengan kedua informasi tersebut maka kita dapat memperkirakan *information content* dari suatu simbol
- Semakin besar probabilitas maka akan dikodekan dengan biner yang kecil

Shannon's *Source Coding Theorem*

- *Information Content I(a) entropy* dari suatu simbol a dimana kita telah mengetahui probabilitasnya dapat dirumuskan sebagai:

$$I(a_i) = \log_2 \frac{1}{p(a_i)} = -\log_2 p(a_i).$$

Basis log 2 menyatakan bahwa informasi dinyatakan dalam bentuk biner

Shannon's *Source Coding Theorem*

- Setelah mengetahui nilai *Information Content* suatu simbol, kita dapat menyatakan:
  - Suatu simbol a dengan probabilitas P(a) sebaiknya direpresentasikan dalam biner dengan $-\log_2 P(a)$ simbol
- Sehingga *entropy* seluruh data adalah:

$$E = \sum_{i=1}^{N} p(a_i) I(a_i) = -\sum_{i=1}^{N} p(a_i) \log_2 p(a_i).$$

Huffman Code's

- Dari Teori Shannon :
  - Sebuah *source* data dapat dikodekan dengan rata-rata panjang kode mendekati *entropy* dari *source* data
- Tahun 1952 D.A.Huffman mengajukan teknik *encoding* untuk menghasilkan panjang kode terpendek dari suatu *source data* dengan memanfaatkan probabilitasnya.

Sejarah Huffman Code's

- Diciptakan oleh David A. Huffman pada tahun 1951 sebagai **tugas kuliah** ketika di Massachusetts Institute of Technology (MIT).
- Dosen pengajar Huffman (bernama Robert M. Fano) menawarkan kepada para mahasiswanya bahwa siapa saja yang dapat menulis sebuah artikel tentang membangun pohon biner yang efisien akan mendapatkan nilai bagus tanpa harus menempuh ujian.
- Setelah lama mencoba dan hampir menyerah, Huffman akhirnya menemukan sebuah metode untuk membangun pohon biner berdasarkan frekuensi.
  - Tekniknya kemudian diakui sebagai teknik yang paling efisien, melebihi teknik buatan sang dosen sendiri.
- Binary Tree (pohon biner) yang dibuat oleh Huffman (disebut sebagai Huffman Tree) adalah dasar dari kompresi data dengan format ZIP yang kita kenal sekarang.
- Teknik ini juga dipakai sebagai salah satu algoritma penyusun format file gambar JPEG dan format file musik populer MP3.
  - Jadi, bila kita sekarang bisa mendengarkan MP3 player kita sambil berkegiatan, salah satu faktor yang membuat teknologi ini ada adalah algoritma Huffman.

Huffman Code's

- Teknik ini dikenal dengan Huffman Code's dengan pertimbangan:
  - *The more frequently occurring symbols can be allocated with shorter codewords than the less frequently occurring symbols*
  - *The two least frequently occurring symbols will have codewords of the same length, and they differ only in the least significant bit.*

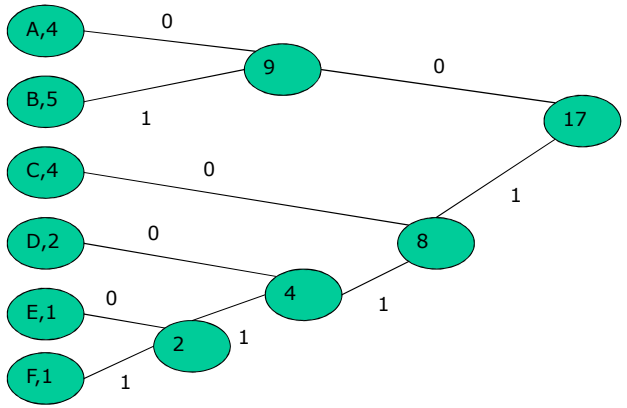
Huffman Code's

- Algoritma dalam *pseudo code*
  1. Hitunglah probabilitas dari setiap simbol yang ada
  2. Pasangkan setiap <simbol,probabilitas> dengan node
  3. Temukan 2 buah *node* dengan probabilitas terendah kemudian buatkah *node parent* dengan probabilitas gabungan dari 2 anaknya
  4. Berikan label untuk cabang dari anak ke *parent* dengan 0 dan 1 (sebaiknya konsisten)
  5. *Update node* (*node* anak diabaikan), lalu periksa jumlah *node* yang ada, bila jumlah *node* > 1 maka ulangi langkah 3
  6. Untuk menemukan kode setiap simbol, lakukan *traverse* dari *root* ke *leaf*, (label *branch* yang dilalui akan menjadi kode untuk *leaf*)

Huffman Code's

- Misalkan data:
  - S= AABAACCCCDDBBBBEF
- Hitung frekuensi data ≈ probalitas
  - a → 4, b → 5, c → 4, d → 2, e → 1, f → 1
- Proses pembangunan code sebagai berikut

Huffman Code's



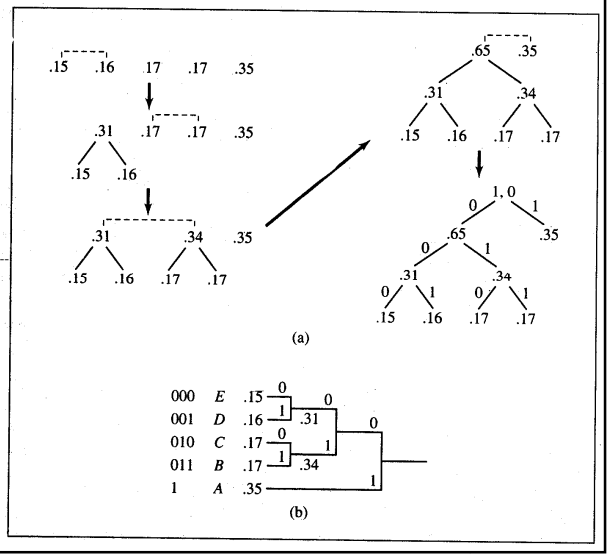
Huffman Code's

- Kode untuk setiap simbol:
  - A → 10 B → 11 C → 00
  - D → 010 E → 0111 F → 0110
  - ☐ Jadi data S= aabaaccccddbbbbef (17 byte) akan dikodekan menjadi :

    10 10 11 10 10 00 00 00 00 010 010 11 11 11 11 0111 0110 = 40 bit ≈ 5 byte
- Dalam implementasinya teknik Huffman Code's dapat dipercepat dengan menggunakan mekanisme *sorting data* (*insertion sort*)

Contoh:

x_i	p_i
A	0,35
B	0,17
C	0,17
D	0,16
E	0,15



Huffman Code's

Dari Huffman tree dapat dibuat tabel *codeword*:

Simbol	Biner	Jumlah Digit
A	1	1
B	011	3
C	010	3
D	001	3
E	000	3

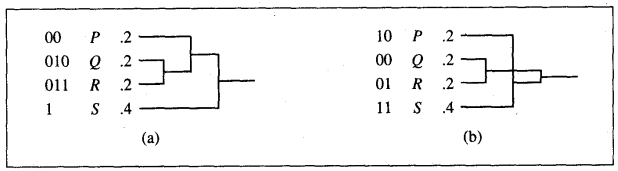
L_{Huff} = Σ Probabilitas Simbol * Jumlah Digit
= 0,35*1 + 0,17*3 + 0,17*3 + 0,16*3 + 0,15*3 = 2,3

H(S) = Entropy(S) = Σ- Probabilitas Simbol * Log_2 Probabilitas Simbol
= - 0.35 $\log_2$ 0.35 - 0.17 $\log_2$ 0.17 - 0.17 $\log_2$ 0.17 - 0.16 $\log_2$ 0.16
- 0.15 $\log_2$ 0.15
= 2,23284

Efisiensi = (H(S)/LHuff) * 100%
= (2,23284/2,3) * 100 %
= 97,08%

Huffman *Coding*

- Tergantung pada bagaimana memilih probabilitas terendah saat membangun Huffman *tree* → Huffman *tree* tidak unik
- Namun, panjang rata-rata *codeword* selalu sama utk *tree* yang berbeda

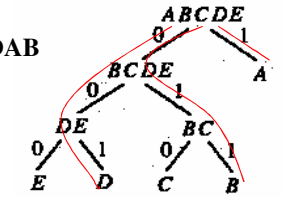


Huffman *Coding*

- Proses *coding*: mentransmisikan *codeword* sesuai dg simbol-simbol yg akan dikirim, mis ABAAD → 101111001
- Untuk *decode message*, konversi tabel harus diketahui penerima → dpt dibangun Huffman *tree*

A 1
B 011
C 010
D 001
E 000

0011011 → DAB



- Masalah: pengirim (*encoder*) dan penerima (*decoder*) harus menggunakan *coding* (Huffman *tree*) yang sama

Huffman *Coding*

Bagaimana *encoder* memberi tahu *decoder* utk menggunakan *code* yang mana:

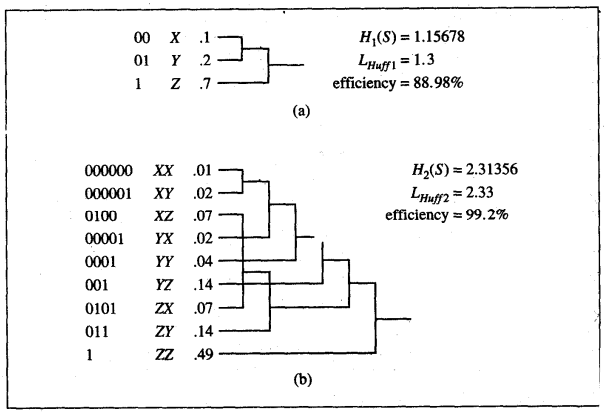
- Baik *encoder* dan *decoder* sudah sepakat sebelumnya utk menggunakan Huffman *tree* tertentu sebelum terjadi pengiriman *message*
- *Encoder* membangun Huffman *tree* yang baru (*fresh*) setiap *message* baru akan dikirimkan, dan mengirimkan tabel konversi bersama-sama dg *message*
  → Keuntungannya terasa jika digunakan utk *message* yang besar

Huffman *Coding*

Apakah masih ada ruang perbaikan utk Huffman *Coding*?

- Semua kemungkinan Huffman *tree* akan memberikan panjang rata-rata yang sama
- Namun ingat Shannon's *Fundamental Theorem*:
  - Semua kemungkinan pasangan simbol dpt digunakan untuk membangun Huffman *tree*
    → kompresi data dpt ditingkatkan
  - Namun perbaikan harus 'dibayar' dg 'Tabel Konversi' yang lebih besar

Contoh



Sumber: repository.binus.ac.id/content/T0034/T003446363.ppt

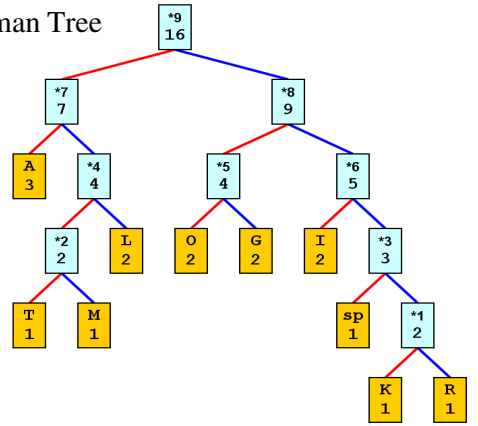
Studi Kasus

- Misalkan kita hendak melakukan kompresi untuk kalimat :
  – LOGIKA ALGORITMA
- Solusi
  – Buat tabel frekuensi :

K	R	T	M	sp	L	O	G	I	A
1	1	1	1	1	2	2	2	2	3

Solusi (contd)

- Buat Huffman Tree



TABEL HUFFMAN CODE

K	R	T	M	sp	L	O	G	I	A
11110	11111	0100	0101	1110	011	100	101	110	00

- L 011 3 bit
- O 100 3 bit
- G 101 3 bit
- I 110 3 bit
- K 11110 5 bit
- A 00 2 bit
- sp 1110 4 bit
- A 00 2 bit
- L 011 3 bit
- G 101 3 bit
- O 100 3 bit
- R 11111 5 bit
- I 110 3 bit
- T 0100 4 bit
- M 0101 4 bit
- A 00 2 bit

Quantizing Compression

- Teknik *Quantizing Compression* bersifat *lossy*
- Digunakan untuk mereduksi data dengan asumsi bahwa perubahan data tidak akan mempengaruhi *content*/informasi
- Melakukan pengkodean, rata-rata pada histogram, menggunakan matrik kuantisasi

Quantizing Compression (kode)

2	9	6	4	8	2	6	3	8	5	9	3	7
3	8	5	4	7	6	3	8	2	8	4	7	3
3	8	4	7	4	9	2	3	8	2	7	4	9
3	9	4	7	2	7	6	2	1	6	5	3	0
2	0	4	3	8	9	5	4	7	1	2	8	3

Histogram :

- Warna 0 = 2
- Warna 1 = 2
- Warna 2 = 9

Dikodekan menjadi 0 (Jumlahnya 13 pixel)

- Warna 3 = 11
- Warna 4 = 9

Dikodekan menjadi 1 (Jumlahnya 20 pixel)

- Warna 5 = 4
- Warna 6 = 5
- Warna 7 = 8

Dikodekan menjadi 2 (Jumlahnya 17 pixel)

- Warna 8 = 9
- Warna 9 = 6

Dikodekan menjadi 3 (Jumlahnya 15 pixel)

Asumsi : jumlah pixel per kategori max 20

Quantizing Compression (kode)

0	3	2	1	3	0	2	1	3	2	3	1	2
1	3	2	1	2	2	1	3	0	3	1	2	1
1	3	1	2	1	3	0	1	3	0	2	1	3
1	3	1	2	0	2	2	0	0	2	2	1	0
0	0	1	1	3	3	2	1	2	0	0	3	0

- Bisakah data ini dibalikkan?

Quantizing Compression (matrik)

- Menggunakan matrik dan pembulatan

3588	-86	-27	-105	-2	1	44	19
-100	-79	24	34	45	-22	-14	-43
18	-64	-35	-20	-15	-17	21	15
-184	3	72	-17	0	-53	8	-13
14	9	44	11	6	-15	-3	-23
39	99	122	-25	-13	-18	-6	26
7	-116	-46	-93	30	35	-4	28
-53	-110	137	-56	-19	-3	-11	3

÷

11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31
19	21	23	25	27	29	31	33
21	23	25	27	29	31	33	35
23	25	27	29	31	33	35	37
25	27	29	31	33	35	36	39

=

326	-6	-1	-6	0	0	1	0
-7	-5	1	1	2	0	0	-1
1	-3	-1	0	0	0	0	0
-10	0	3	0	0	-1	0	0
0	0	1	0	0	0	0	0
1	4	4	0	0	0	12	0
0	-4	-1	-3	0	1	0	0
-2	-4	4	-1	0	0	0	0

Arithmetic Coding

- Pada teknik ini, simbol yang ada akan direpresentasikan dengan bilangan real mulai dari 0-1. Konsekuensi?
- *Aritmethic Coding* menawarkan efisiensi yang lebih baik dibandingkan dengan Huffman Code's
- Sesuai digunakan untuk jumlah simbol sedikit (*binary simbol*) dan simbol dengan *highly skewed probabilities*

Cara 1: http://marknelson.us/1991/02/01/arithmetic-coding-statistical-modeling-data-compression/

Arithmetic Coding [*Encoding*]

- Contoh Penerapan *arithmetic coding*
  - Encoding " BILL GATES"
- Informasi yang dimiliki

Character	Probability	Range
SPACE	1/10	0.00 - 0.10
A	1/10	0.10 - 0.20
B	1/10	0.20 - 0.30
E	1/10	0.30 - 0.40
G	1/10	0.40 - 0.50
I	1/10	0.50 - 0.60
L	2/10	0.60 - 0.80
S	1/10	0.80 - 0.90
T	1/10	0.90 - 1.00

Cara 1: http://marknelson.us/1991/02/01/arithmetic-coding-statistical-modeling-data-compression/

Arithmetic Coding [*Encoding*]

- Algoritma:
  - Set low to 0.0
  - Set high to 1.0
  - While there are still input symbols do
    - get an input symbol
    - code_range = high - low.
    - high = low + range*high_range(symbol)
    - low = low + range*low_range(symbol)
    - End of While
  - output low

Cara 1: http://marknelson.us/1991/02/01/arithmetic-coding-statistical-modeling-data-compression/

Arithmetic Coding [*Encoding*]

- *Encoding* Bill Gates

New Character	Low value	High Value
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
SPACE	0.25720	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.2572168
E	0.257216772	0.257216776
S	0.2572167752	0.2572167756

Cara 1: http://marknelson.us/1991/02/01/arithmetic-coding-statistical-modeling-data-compression/

Arithmetic Coding [*Decoding*]

- Algoritma:
  - get encoded number
  - Do
    - find symbol whose range straddles the encoded number
    - output the symbol
    - range = symbol low value - symbol high value
    - subtract symbol low value from encoded number
    - divide encoded number by range
  - until no more symbols

Cara 1: http://marknelson.us/1991/02/01/arithmetic-coding-statistical-modeling-data-compression/

Arithmetic Coding [*Decoding*]

- *Decoding*

Encoded Number	Output Symbol	Low	High	Range
0.2572167752	B	0.2	0.3	0.1
0.572167752	I	0.5	0.6	0.1
0.72167752	L	0.6	0.8	0.2
0.6083876	L	0.6	0.8	0.2
0.041938	SPACE	0.0	0.1	0.1
0.41938	G	0.4	0.5	0.1
0.1938	A	0.2	0.3	0.1
0.938	T	0.9	1.0	0.1
0.38	E	0.3	0.4	0.1
0.8	S	0.8	0.9	0.1
0.0				

Cara 2: CS3204-Pengolahan Citra Chpater 9 Kompresi Citra, IT Telkom

Arithmetic Coding [*Encoding*]

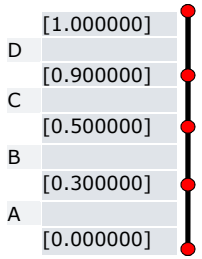
- Contoh penerapan *arithmetic coding*
- Informasi yang dimiliki

Sbl.	P(s)	Cumulative P(s)	Range Simbol
A	0.3	0.3	[0.000 , 0.300]
B	0.2	0.5	[0.300 , 0.500]
C	0.4	0.9	[0.500 , 0.900]
D	0.1	1	[0.900 , 1.000]

Cara 2: CS3204-Pengolahan Citra Chpater 9 Kompresi Citra, IT Telkom

Arithmetic Coding [*Encoding*]

- Range Simbol Awal
  - Range simbol ini akan digunakan untuk menentukan bilangan yang mewakili data yang di kompresi
- Note: Pada umumnya hasil akhir proses encoding akan dikonversi dalam bentuk biner



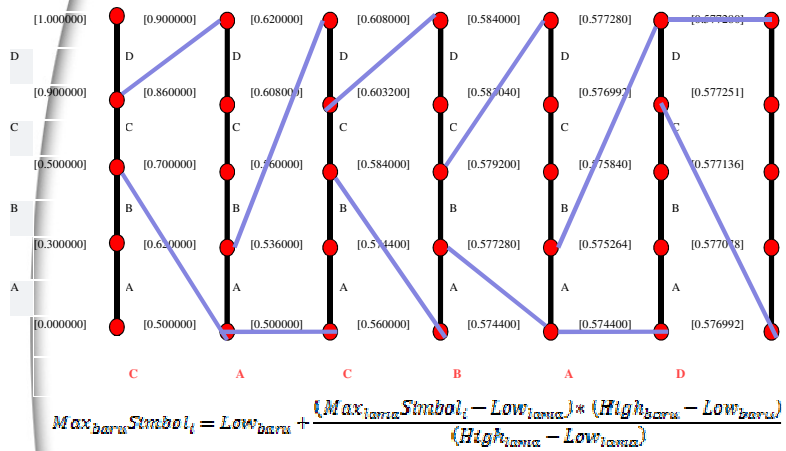
Cara 2: CS3204-Pengolahan Citra Chpater 9 Kompresi Citra, IT Telkom

Arithmetic Coding [*Encoding*]

- Pesan akan di-*encode* menjadi sebuah bilangan beserta informasi *range* simbol
- Proses penghasilan dengan memasukkan pesan pada *range* simbol dan melakukan update range simbol
- Misalkan simbol : " C A C B A D"

Cara 2: CS3204-Pengolahan Citra Chpater 9 Kompresi Citra, IT Telkom

Arithmetic Coding [*Encoding*]



$$Max_{baru}Simbol_i = Low_{baru} + \frac{(Max_{lama}Simbol_i - Low_{lama}) * (High_{baru} - Low_{baru})}{(High_{lama} - Low_{lama})}$$

Cara 2: CS3204-Pengolahan Citra Chpater 9 Kompresi Citra, IT Telkom

Arithmetic Coding [*Encoding*]

- Pada akhir proses kita akan memperoleh sebuah range simbol akhir, dalam kasus ini adalah [0.576992,0.57728]
- Pesan " C A C B A D" dapat kita kodekan menjadi suatu bilangan pada *range* terakhir
- Misalkan dengan *midpoint interval* maka pesan dikodekan menjadi 0.577136

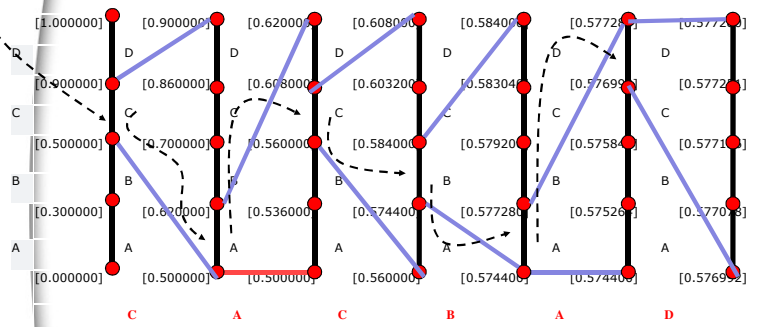
Cara 2: CS3204-Pengolahan Citra Chpater 9 Kompresi Citra, IT Telkom

Arithmetic Coding [*Decoding*]

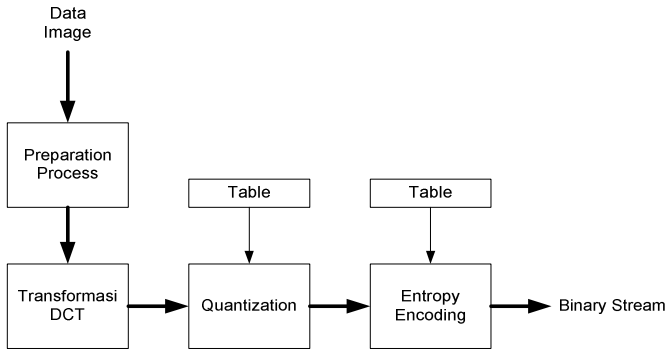
- Untuk melakukan *Decoding* kita membutuhkan *range* simbol awal
- Langkahnya adalah:
  - Cocokkan nilai data kode dengan range yang ada pada range simbol, lalu ekstrak kode yang sesuai dengan range simbol
  - Pecah range simbol sesuai dengan hasil pencocokan (Mengubah range simbol)

Arithmetic Coding [Decoding]

0.577136 (data yang akan di-decoding, di cocokkan dengan interval kemudian dilakukan pembagian interval seperti pada tahap encoding



JPEG - **Joint Photographic Experts Group**



JPEG

1. Tahap Persiapan (*Preparation Process*)

Pada tahap ini dilakukan proses membagi citra menjadi blok 8x8

10	12	11	10	11	12	13	20
11	11	12	16	17	18	12	11
10	10	15	16	14	13	19	20
20	18	16	16	15	14	12	11
12	13	11	11	10	13	14	17
10	11	17	16	15	12	12	13
17	18	18	10	11	12	14	15
11	12	20	19	18	17	17	15

JPEG

2. Tranformasi DCT

- Transformasi DCT bertujuan mengubah menghitung frekuensi-frekuensi pembentuk dari citra blok 8x8 dan memisahkan frekuensi rendah dan frekuensi tinggi dari hasil tranformasi DCT.
- Transformasi DCT terhadap blok 8x8 dapat dilakukan dengan rumus :

$$DCT(u,v) = \frac{1}{4}.C(u).C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}.f(x,y).\cos\left[\frac{(2.x+1).u.\pi}{16}\right].\cos\left[\frac{(2.y+1).v.\pi}{16}\right]$$

Dimana : $C(z) = \begin{cases} \frac{1}{\sqrt{2}}, & z = 0 \\ 1, & z > 0 \end{cases}$

JPEG

2. Tranformasi DCT (cont.)

10	12	11	10	11	12	13	20
11	11	12	16	17	18	12	11
10	10	15	16	14	13	19	20
20	18	16	16	15	14	12	11
12	13	11	11	10	13	14	17
10	11	17	16	15	12	12	13
17	18	18	10	11	12	14	15
11	12	20	19	18	17	17	15

DCT

Frekuensi >>, Penting <<

3588	-86	-27	-105	-2	1	44	19
-100	-79	24	34	45	-22	-14	-43
18	-64	-35	-20	-15	-17	21	15
-184	3	72	-17	0	-53	8	-13
14	9	44	11	6	-15	-3	-23
39	99	122	-25	-13	-18	-6	26
7	-116	-46	-93	30	35	-4	28
-53	-110	137	-56	-19	-3	-11	3

Frekuensi >>, Penting <<

JPEG

3. Quantisasi

- Proses Quantisasi bertujuan untuk menghilangkan nilai-nilai yang tidak penting (dalam hal ini nilai-nilai yang berada pada daerah frekuensi tinggi) pada matrix hasil dari Transformasi DCT.

$$Quantized\ \_Value(i,j) = \frac{DCT\ \_Matrix(i,j)}{Quantum\ \_Matrix(i,j)}$$

JPEG

3. Quantisasi (Cont.)

3588	-86	-27	-105	-2	1	44	19
-100	-79	24	34	45	-22	-14	-43
18	-64	-35	-20	-15	-17	21	15
-184	3	72	-17	0	-53	8	-13
14	9	44	11	6	-15	-3	-23
39	99	122	-25	-13	-18	-6	26
7	-116	-46	-93	30	35	-4	28
-53	-110	137	-56	-19	-3	-11	3

÷

11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31
19	21	23	25	27	29	31	33
21	23	25	27	29	31	33	35
23	25	27	29	31	33	35	37
25	27	29	31	33	35	36	39

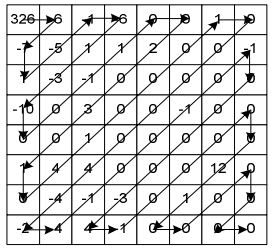
=

326	-6	-1	-6	0	0	1	0
-7	-5	1	1	2	0	0	-1
1	-3	-1	0	0	0	0	0
-10	0	3	0	0	-1	0	0
0	0	1	0	0	0	0	0
1	4	4	0	0	0	12	0
0	-4	-1	-3	0	1	0	0
-2	-4	4	-1	0	0	0	0

JPEG

4. Entropy Encoding

- Entropy Encoding adalah teknik kompresi yang bersifat lossless. Tahap ini bertujuan untuk mengkompresi matrix hasil quantisasi, bisa menggunakn metode huffman atau RLE
- Proses Entropy Encoding terhadap hasil quantisasi di atas dengan pembacaan zig-zag :



Hasil encoding jika menggunakan RLE :
326,-6,-7,1,-5,1,6,1,-3, [***0,3***] , -1,1,[***0,2***],2,[***0,1***],3, [***0,1***], 1, [***0,1***],4,1,[***0,3***],1,[***0,5***],4,-4,-2,4,-1,[***0,2***],-1,[***0,1***], -1, [***0,4***],-3,4,1,[***0,5***],12,1,[***0,7***] = 49 byte

Referensi

1. _____,2008,CS3204-Pengolahan Citra, Departement Teknik Informatika-IT Telkom
2. Hendrawan,-, HUFFMAN CODING[online],url: *telecom.ee.itb.ac.id/~hend/ET5014/**HuffmanCoding**_09.ppt* ,ITB
3. repository.binus.ac.id/content/T0034/T003446363.ppt
4. Mark,1991,**Arithmetic Coding + Statistical Modeling = Data Compression[online]**,url: http://marknelson.us/1991/02/01/arithmetic-coding-statistical-modeling-data-compression/, Tanggal Akses: 21 April 2011
5. http://computer.howstuffworks.com/file-compression3.htm
6. http://en.wikipedia.org/wiki/Lossy_data_compression
7. http://en.wikipedia.org/wiki/Lossless_data_compression