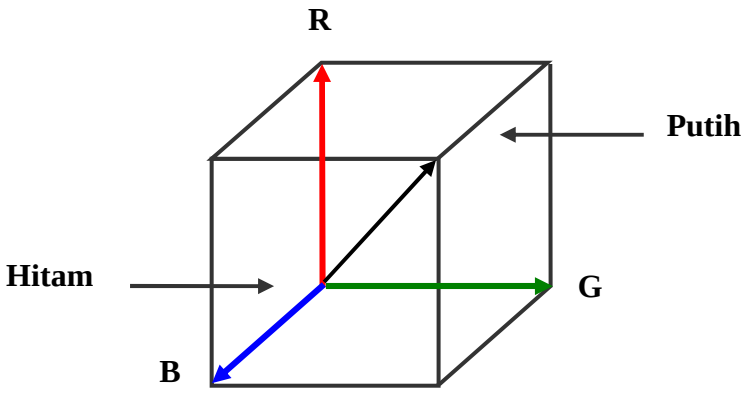
BAB II

LANDASAN TEORI

2.1 Citra

Citra merupakan dimensi spatial yang berisi informasi warna dan tidak bergantung pada waktu. Citra merupakan sekumpulan titik-titik dari gambar, yang disebut *pixel* (*picture elemen*t). Titik-titk tersebut menggambarakan posisi koordinat dan mempunyai intensitas yang dapat dinyatakan dengan bilangan. Intensitas ini menunjukan warna citra, melalui penjumlahan (*Red, Green dan Blue / RGB*).



Gambar 2.1. Koordinat RGB

Kordinat memberikan informasi warna pixel berdasarkan; *Brightness* (ketajaman) warna cahaya (hitam, abu-abu, putih) dari sumber, *Hue* (corak warna) yang ditimbulkan oleh warna (merah, kuning, hijau dll) dan merupakan panjang gelombang dominan dari sumber.

Misalnya citra dengan 8 bit per pixel mempunyai 256 warna dan citra dengan 24 bit mempunyai 32768 warna, jadi tiap pixel dinyatakan dengan ;

- bit 0 sampai dengan 7 untuk warna merah.
- bit 8 sampai dengan 15 untuk warna hijau
- bit 16 sampai dengan 24 untuk warna biru

Kemungkinan kombinasi warna yang ada adalah = 256.3 + 256.2 + 256.1 = *16.* 777.216 dimana nilai 0 menyatkan warna ***hitam*** sedangakan nilai 16 777.216 menyatakan warna ***putih***.Dari penjelasan singkat diatas, citra dapat diubah dari domain spatial menjadi domain yang lain, dengan tujuan untuk mempermudah pengkodean. Proses perubahan ini dinamakan *transformasi*.

2.2 JPEG

Joint Photograpic Experts Group (JPEG) dirancang untuk kompresi beberapa *full-color* atau *gray-scale* dari suatu gambar yang asli, seperti pemandangan asli di dunia ini. JPEGs bekerja dengan baik pada *continous tone images* seperi *photographs* atau semua pekerjaan seni yang menginginkan yang nyata; tetapi tidak terlalu bagus pada ketajaman gambar dan seni pewarnaan seperti penulisan, kartun yang sederhana atau gambar yang mengunakan banyak garis. JPEG sudah mendukung untuk 24-bit *color depth* atau sama dengan 16,7 juta warna (224 = 16.777.216 warna). *Progressive JPEGs (p-JPEGs)* adalah tipe dari beberapa persen lebih kecil dibandingkan *baseline JPEGs*: tetapi keuntungan dari JPEG dan tipe-tipenya telihat pada langkah-langkahnya sama seperti *iinterlaced GIFs*.

JPEG adalah algoritma kompresi secara *lossy*. JPEG bekerja dengan merubah gambar spasial dan merepresentasikan kedalam pemetaan frekuensi.

*Discrete Cosine Transform (*DCT*)* dengan memisahkan antara informasi frekuensi yang rendah dan tinggi dari sebuah gambar. Informasi frekuensi yang tinggi akan diseleksi untuk dihilangkan yang terikat pada penganturan kualitas yang digunakan. Kompresi dengan tingkatan yang lebih baik, tingkatan yang lebih baik dari informasi yang dihilangkan. Waktu Kompresi dan dekompresi dilaksanakan dengan simetris. JPEG Group's (IJG) decoder lebih ditingkatkan kemampuannya dibadingkan dengan encodernya. Manakala, ketika dperlihatkan 8 bits, mengurangi kuantisasi warna yang lambat. Banyak para penjual JPEG menawarkan untuk mempercepat hasil dari JPEG, kuantisasi warna dan kualitas dengan mengimplementasikan IJG.

JPEG dirancang untuk mengeksploitasi tingkatan dari mata kita; yakni bahwa mata kita tidak akan dapat mebedakan perubahan yang lembar terang dan warna dibandingkan dengan perbedaan suatu jarak apakah jauh atau dekat. Untuk itu JPEG sangat baik digunakan pada fotografi dan monitor 80-bit. JPEG sebenarnya hanyalah algoritma kompresi, bukan merupakan nama format file. File yang biasa disebut JPEG pada jaringan sebenarnya adalah JFIF (*JPEG File Interchange Format*).

2.3 Kompresi

Informasi yang dirasakan redundan dapat dikompresi untuk meminimalisasi redundansi tersebut. Hal ini dapat pula dilakukan pada citra dijital yang beragam tipenya, dengan menerapkan berbagai metode kompresi yang ada. Tujuan utama dari kompresi pada citra dijital adalah untuk mengurangi

penggunaan memori, sehingga akan memudahkan penyimpanan, pengolahan serta pengiriman citra dijital tersebut. Dapat disimpulkan bahwa kompresi merupakan proses untuk menghilangkan berbagai kerumitan yang tidak penting (redundansi) dari suatu informasi, dengan memaksimalkan kesederhanaannya dan tetap menjaga kualitas penggambaran dari informasi tersebut.

2.3.1 Hal-hal penting dalam kompresi citra

a. Scalability/Progressive Coding/Embedded Bitstream

1. Adalah kualitas dari hasil proses pengkompresian citra karena manipulasi bitstream tanpa adanya dekompresi atau rekompresi.
2. Biasanya dikenal pada loseless codec.
3. Contohnya pada saat preview image sementara image tersebut didownload. Semakin baik scalability, makin bagus preview image.
4. Tipe scalability:
    a. **Quality progressive:** dimana image dikompres secara perlahan-lahan dengan penurunan kualitasnya
    b. **Resolution progressive:** dimana image dikompresi dengan mengenkode resolusi image yang lebih rendah terlebih dahulu baru kemudian ke resolusi yang lebih tinggi.
    c. **Component progressive:** dimana image dikompresi berdasarkan komponennya, pertama mengenkode komponen gray baru kemudian komponen warnanya.

b. **Region of Interest Coding**: daerah-daerah tertentu dienkode dengan kualitas yang lebih tinggi daripada yang lain.
c. **Meta Information**: image yang dikompres juga dapat memiliki meta information seperti statistik warna, tekstur, small preview image, dan author atau copyright information

2.3.2 Konsep Dasar Teknik Kompresi Citra

Teknik kompresi citra mengacu pada dua konsep dasar, yaitu :

1. Mengeksploitasi redundansi informasi yang terdapat pada pola sinyal citra digital. Metode ini digunakan pada teknik kompresi citra *lossless coding*.

   Redundansi tersebut dapat berupa:

   a. Redundansi Spasial akibat korelasi antara piksel-piksel yang bertetangga yang memiliki intensitas yang sama
   b. Redundansi Spektral akibat korelasi antara bidang-bidang warna yang berbeda
   c. Redundansi Temporal akibat korelasi frame-frame yang berbeda pada citra dinamis
2. Menggunakan deviasi dalam batas yang dapat ditoleransi dengan cara mengurangi detail citra yang tidak dapat ditangkap oleh penglihatan manusia. Resolusi spasial, waktu dan amplitudo disesuaikan dengan aplikasi yang digunakan. Metode ini digunakan pada teknik kompresi citra *lossy coding* dengan mengeksploitasi redundansi statistik dan visual.

   Teknik kompresi data :

a. Kompresi berbasis Statistik (*Lossless*)

b. Kompresi berbasis Kuantisasi (*Lossy*)

c. Kompresi berbasis Transformasi (*Lossless/Lossy*)

d. Kompresi berbasis Fraktal (*Lossy*)

2.3.3 Klasifikasi Teknik Kompresi

a. Entropy Encoding

1. Bersifat loseless
2. Tekniknya tidak berdasarkan media dengan spesifikasi dan karakteristik tertentu namun berdasarkan urutan data.
3. Statistical encoding, tidak memperhatikan semantik data.
4. Misal: Run-length coding, Huffman coding, Arithmetic coding

b. Source Coding

1. Bersifat lossy
2. Berkaitan dengan data semantik (arti data) dan media.
3. Misal: Prediction (DPCM, DM), Transformation (FFT, DCT), Layered Coding (Bit position, subsampling, sub-band coding), Vector quantization

c. Hybrid Coding

1. Gabungan antara lossy + loseless
2. misal: JPEG, MPEG, H.261, DVI

2.3.4 Sifat Kompresi berdasarkan hasil

a. Lossy Compression

1. Teknik kompresi dimana data hasil dekompresi tidak sama dengan data sebelum kompresi namun sudah “cukup” untuk digunakan. Contoh: Mp3, streaming media, JPEG, MPEG, dan WMA.
2. Kelebihan: ukuran file lebih kecil dibanding loseless namun masih tetap memenuhi syarat untuk digunakan.
3. Biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan, tidak begitu dilihat oleh manusia sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.
4. Misal terdapat image asli berukuran 12,249 bytes, kemudian dilakukan kompresi dengan JPEG kualitas 30 dan berukuran 1,869 bytes berarti image tersebut 85% lebih kecil dan ratio kompresi 15%.

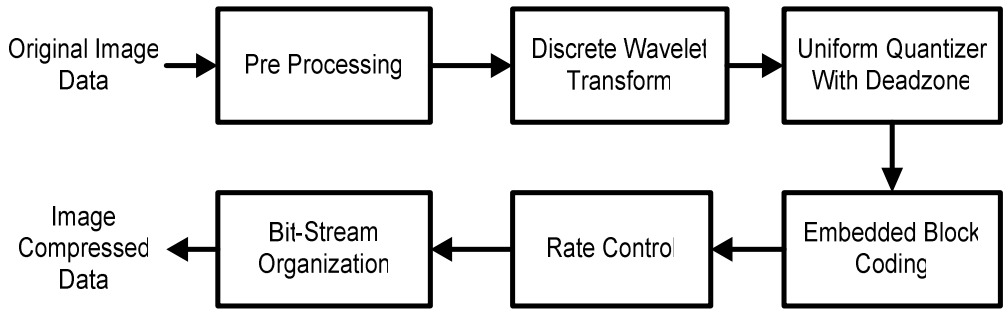
b. Loseless

1. Teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi. Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip
2. Teknik ini digunakan jika dibutuhkan data setelah dikompresi harus dapat diekstrak/dekompres lagi tepat sama. Contoh pada data teks, data program/biner, beberapa image seperti GIF dan PNG.
3. Kadangkala ada data-data yang setelah dikompresi dengan teknik ini ukurannya menjadi lebih besar atau sama.

2.4 Kompresi pada JPEG2000

JPEG2000 merupakan salah satu metode standard dan terbaru dalam melakukan kompresi terhadap citra dijital, yang dihasilkan oleh ISO (*International Standard Organization*) bekerja sama dengan ITU (*International Telecommunication Union*). JPEG2000 tersebut dikembangkan, karena standar metode kompresi yang sudah ada, yaitu JPEG, dirasakan masih belum dapat memenuhi kebutuhan kompresi citra dijital.

Teknik kompresi pada JPEG2000 termasuk metode kompresi yang simetris, yaitu proses kompresi dan dekompresinya menggunakan dasar algoritma yang sama, tetapi mempunyai arah yang berlawanan. Berikut ini skema proses kompresi pada JPEG2000



Gambar 2.2. Skema Proses kompresi JPEG 2000

2.4.1 Pre-Processing

Pada *Pre-Processing* dilakukan transformasi atas data-data piksel penyusun komponen citra dijital menjadi bentukan lain untuk memudahkan perhitungan. Pada Gambar 2.3 di bawah ini dapat dilihat tahapan-tahapan dalam *Pre-Processing.*



Gambar 2.3. Skema Pre Processing

a. Tiling

Tiling (pengubinan) dibutuhkan untuk mengatasi permasalahan keterbatasan sumber daya memori yang tersedia selama proses kompresi. Dalam t*iling* ini dilakukan pembagian area citra dijital menjadi persegi panjang dengan ukuran yang sama dan tidak saling tumpang tindih. Perbedaan ukuran dapat terjadi pada bagian tepi dari citra dijital. Ukuran dari *tile* atau ubin yang dibentuk adalah bebas. Sebuah citra dijital dapat saja hanya terbagi menjadi sebuah *tile*, yaitu dengan ukuran yang sama dengan citra dijital itu sendiri.

b. DC Level Shifting

Data yang diolah hendaknya merupakan bilangan yang berkisar pada nol. Jika penyusun dari masing-masing komponen pada citra dijital merupakan tipe non-negatif (*unsigned*), maka perlu dilakukan *DC Level Shifting* untuk masing-masing penyusun komponen tersebut. Pada *DC Level Shifting*, akan dilakukan penambahan sebuah bilangan penyeimbang yaitu *-2B-1*, sehingga penyusun komponen citra dijital akan berada pada kisaran $-2B-1 = x[n] < 2B-1$. *B* menyatakan kedalaman dari bit penyusun masing-masing komponen pada citra dijital, dan *x[n]* menyatakan bilangan penyusun yang berkisar pada nol.

Sedangkan apabila penyusun komponen citra dijital merupakan bilangan bertipe *signed*, maka proses ini tidak perlu dilakukan.

c. Irreversible Component Transformation (ICT)

Irreversible Component Transformation (ICT) merupakan transformasi komponen untuk tipe kompresi *lossy*. Sedangkan pada kompresi *lossless*, transformasi yang dilakukan disebut dengan *Reversible Component Transformation* (RCT). Dalam ICT ini, akan dilakukan transformasi dari format RCB menjadi *Luminance* (Y), dan *Chrominnce* (Cr dan Cb). Rumus yang digunakan untuk melakukan transformasi tersebut adalah sebagai berikut:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}16875 & -0{,}33126 & 0{,}5 \\ 0{,}5 & -0{,}41869 & -0{,}08131 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \ldots(1)$$

2.4.2 Discrete Wavelet Transform (DWT)

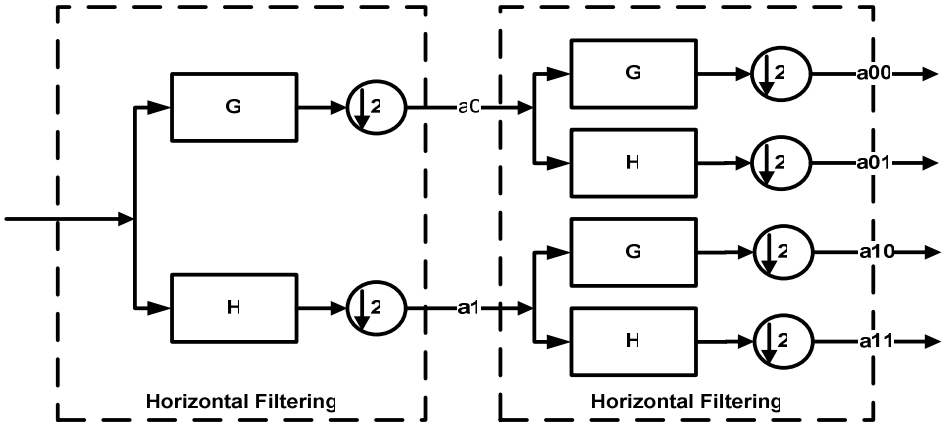
Pada transformasi *wavelet*, dilakukan penyaringan data menjadi *low pass* dan *high pass*. *Low pass* merepresentasikan bagian penting dari data dalam resolusi yang rendah, sedangkan *high pass* menyatakan detail dari data yang ditransformasikan. Pada JPEG2000, dilakukan DWT dua dimensi, yaitu DWT terhadap baris (horizontal), dan terhadap kolom (vertikal). Suatu tahapan DWT dua dimensi akan menghasilkan empat buah kuadran, yaitu:

LL: sub-kelompok *low* dari hasil transformasi pada baris dan kolom.

HL: sub-kelompok *high* dari hasil transformasi pada baris, dan sub-kelompok *low* dari transformasi kolomnya.

LH: sub-kelompok *low* dari hasil transformasi pada baris, dan sub-kelompok *high* dari transformasi kolomnya.

HH: sub-kelompok *high* dari hasil transformasi baris dan kolom.



Gambar 2.4. Transformasi Wavelet Dua Dimensi

Salah satu metode transformasi *wavelet* yang cukup sederhana adalah dengan menggunakan pendekatan *Lifting*. Dengan metode ini, konstanta *wavelet filter* seperti konstanta *filter* Daubechies (9,7) yang banyak digunakan, difaktorkan menjadi konstanta lainnya yang menyatakan *lifting steps.* Berikut ini beberapa konstanta yang digunakan untuk perhitungan dengan pendekatan *Lifting* ini:

a = -1.586 b = -0.052 c = 0.883 d = 0.444

2.4.3 Quantization

Quantization merupakan proses untuk membulatkan koefisien yang dihasilkan dari proses transformasi. Operasi ini bersifat *lossy*, kecuali jika nilai pembulatannya 1 dan koefisien hasil transformasi merupakan bilangan bulat, sebagai hasil yang diperoleh dari *Reversible Component Transformation* (RCT).

Masing-masing koefisien dari hasil transformasi yaitu *ab(u,v)*, dari sub kelompok *b*, dibulatkan dan menghasilkan *qb(u,v)*, berdasarkan rumus berikut:

$$q_b(u,v) = sign(a_b(u,v)) \left\lfloor \frac{|a_b(u,v)|}{\Delta_b} \right\rfloor \ldots(2)$$

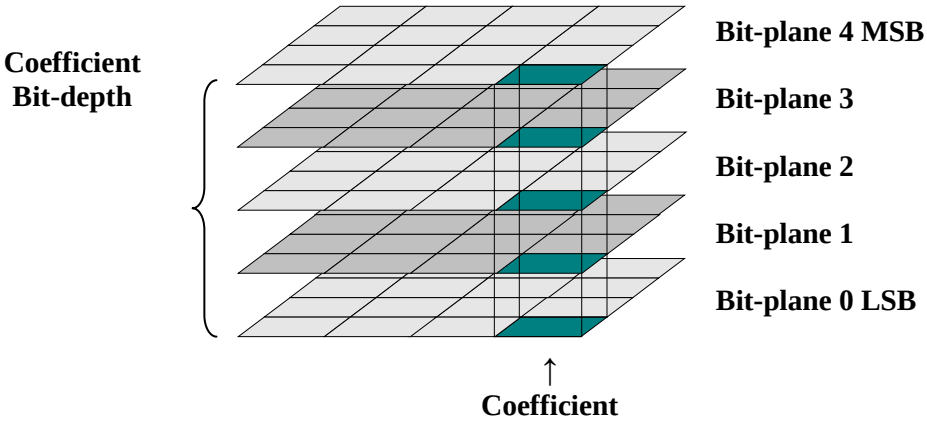
Dimana *b* menyatakan ukuran dari langkah pembulatan yang dilakukan terhadap koefisien, *sign(ab(u,v))* menyatakan tanda dari koefisien *(ab(u,v))*, dan operasi + *X* +digunakan untuk mendapatkan bilangan bulat terbesar yang kurang dari atau sama dengan *x*.

2.4.4 Entropy Coding

Metode yang digunakan dalam melakukan *Entropy Coding* adalah *Embedded Block Coding with Optimised Truncation*, atau yang disingkat sebagai EBCOT. EBCOT terdiri atas dua tahapan (*Tier*). *Tier* 1 bertanggung jawab dalam pemodelan pada data yang dikompresi, serta *entropy coding* yang dilakukan untuk membentuk *bit stream*, sedangkan *Tier* 2 mengatur data keluarannya membentuk paket-paket *bit stream* hasil kompresi.

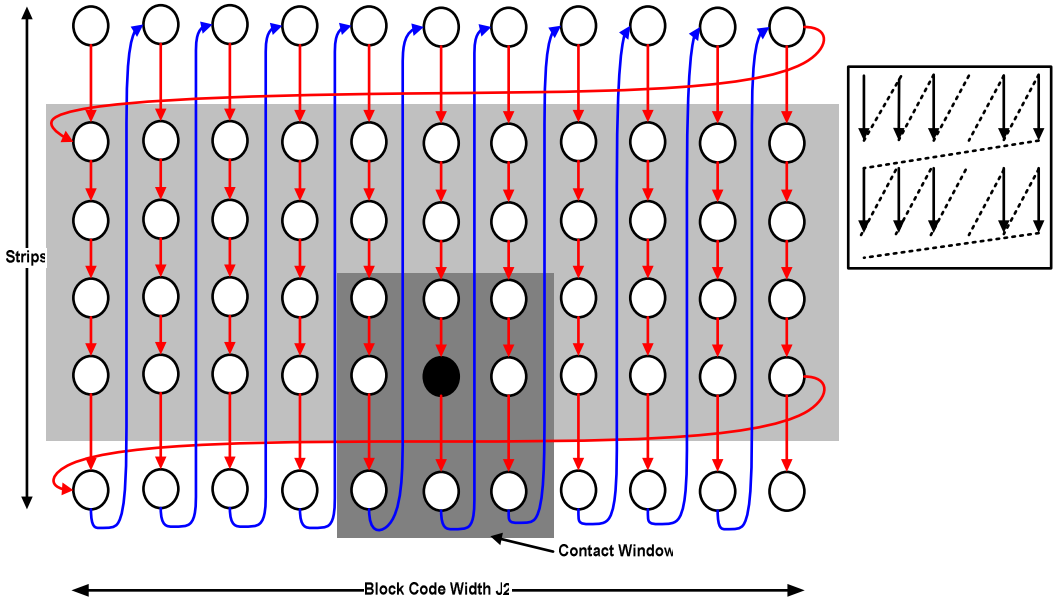
Dalam EBCOT, dilakukan pengelompokan koefisien yang diperoleh dari transformasi *Wavelet* menjadi persegi dengan ukuran yang lebih kecil, untuk dikodekan secara independen. Standar ukuran yang digunakan adalah *2n*, tetapi tidak boleh kurang dari *4x4*. EBCOT dilakukan pada masing-masing lapisan bit penyusun citra dijital yang disebut sebagai *bitplane*, dengan perhitungan yang dimulai dari lapisan bit yang paling signifikan (*Most Significant Bit*) atau MSB, ke bit yang kurang signifikan (*Least Significant Bit*) atau LSB. Lapisan MSB yang

dipilih sebagai *bitplane* pertama yang dikode, setidaknya harus mengandung sebuah bit 1.



Gambar 2.5. EBCOT pada Code Block yang diuraikan menjadi bit plane

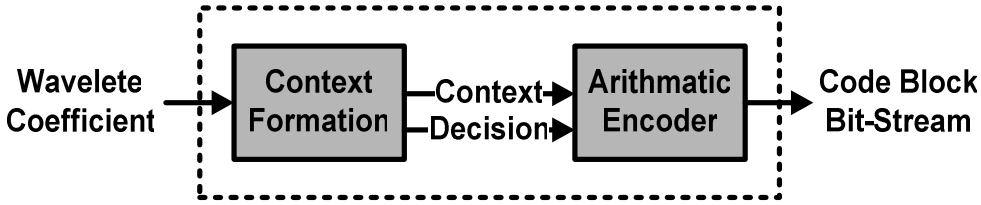
Proses perhitungan pada masing-masing *code block*, dilakukan dengan urutan perhitungan yang khusus. Dimulai dari pojok kiri *code block*, dengan melakukan perhitungan terhadap empat baris koefisien pada kolom pertama, kemudian diikuti dengan perhitungan pada empat baris data pertama di kolom kedua, hal tersebut dilakukan hingga mencapai kolom yang paling kanan. Proses tersebut kemudian diulangi untuk empat baris data yang kedua, dari kolom yang paling kiri sampai paling kanan. Proses tersebut dilakukan hingga mencapai bagian bawah *code block.* Pola perhitungan tersebut dapat dilihat pada Gambar 2.6 di bawah ini:



Gambar 2.6. Pola Perhitungan Koefisien Bit dalam Code Block

a. Tier 1

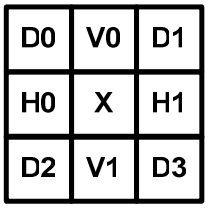
Skema *Tier* 1 pada EBCOT dapat dilihat pada Gambar 2.7 di bawah ini:



Gambar 2.7. Skema EBCOT Tier 1

Context Formation

Context vector menyatakan *significance state* dari kedelapan koefisien bit yang menjadi tetangga dari sebuah bit seperti tampak pada Gambar 2.8 di bawah ini. Jika koefisien dari masing-masing tetangga tersebut signifikan, maka tetangga-tetangga tersebut akan bernilai 1, tetapi jika sebaliknya akan bernilai 0.

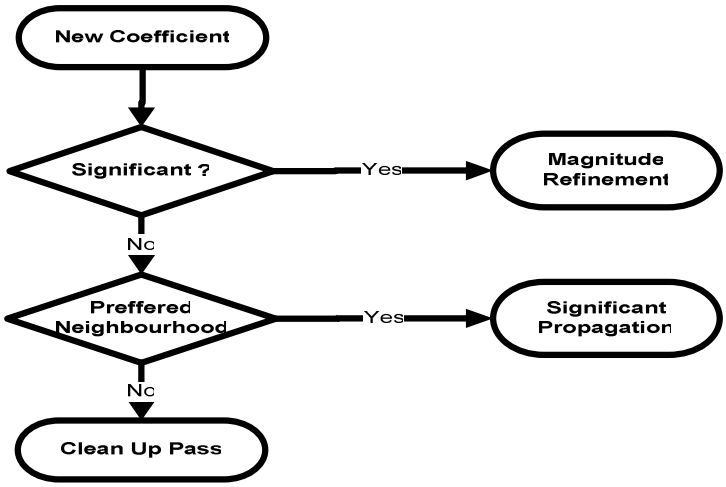


Gambar 2.8. Bit bit dalam Perhitungan Context Vector

Terdapat empat pengkodean yang dilakukan dalam *entropy coding* ini, yaitu *significant coding, sign coding, magnitude refinement,* dan *cleanup coding*. Keempat pengkodean tersebut dilakukan dalam tiga buah tahapan, yaitu:

1. *Significance* dan *sign coding* dalam *significance propagation pass*.
2. *Magnitude refinement coding* dalam *magnitude refinement pass*.
3. *Cleanup* dan *sign coding* dalam *cleanup pass*.

Masing-masing koefisien bit akan dikodekan dengan salah satu dari ketiga tahapan tersebut, berdasarkan kondisi dari bit yang sesuai dengan salah satu tahapan. Skema dari ketiga tahapan tersebut dapat dilihat pada Gambar 2.9 berikut ini:

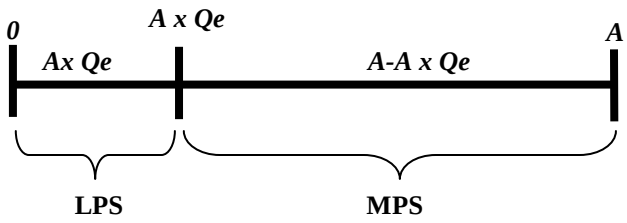


Gambar 2.9. Skema Entropy Coding

Arithmatic Coding

Pengkodean yang digunakan dalam *arithmatic coding* ini adalah MQ-Coder. MQ-Coder tidak hanya menggunakan simbol dari bit masukan, tetapi lebih mengindikasikan jenis bit yang diolah, apakah bit tersebut termasuk bit yang kemungkinan munculnya lebih sering (*Most Probable Symbol*), disingkat dengan MPS; atau sebaliknya, yaitu lebih jarang muncul (*Less Probable Symbol*), disebut sebagai LPS.

Masing-masing kemungkinan tersebut mempunyai nilai terbatas dalam sebuah interval, yang terbagi menjadi dua buah sub interval. Dalam interval tersebut, MPS berada pada bagian atas dari LPS, dengan prosentase lebih besar. Penggambaran, serta pembagian dari interval tersebut dapat dilihat pada Gambar 2.10 di bawah ini:



Gambar 2.10 Pembagian Interval pada MQ Coder

Pada Gambar 9 di atas, *A* menyatakan panjang interval, dan *Qe* menyatakan perkiraan probabilitas dari LPS, sedangkan basis, atau batas bawah interval dinyatakan dengan *C*. Panjang interval dibatasi dalam jangkauan $0.75 = A < 1.5$. Jika nilai *A* berada di bawah batasan minimum, maka *A* akan digandakan, hingga berada dalam batasan yang seharusnya. Proses ini dinamakan dengan *renormalization*. Jika nilai *A* berubah, maka *C* juga harus diubah dengan proses

yang sama. Karena nilai *A* yang selalu berada dalam batasan tersebut, maka nilai *A* akan selalu dianggap sebanding dengan 1 (*A »* 1). Sebuah kondisi dimana sub interval MPS kurang dari LPS dapat terjadi jika *Qe* mendekati 0.5 dengan nilai *A* yang kecil. Apabila kondisi tersebut terjadi, maka harus dilakukan pembalikan nilai antara MPS dengan LPS, yang disebut sebagai *Conditional exchange*

b. Tier 2

MQ Coder yang dilakukan pada masing-masing *code block*, seperti yang telah dijelaskan di atas, akan menghasilkan *bit stream*, berupa *byte-byte* data mentah yang telah terkompresi. Pada *Tier* 2 dalam tahapan EBCOT ini, akan dilakukan pengaturan terhadap *bit stream* tersebut, sehingga representasi akhir dari citra dijital yang telah dikompresi dapat lebih terstruktur, yaitu dengan membentuk *layer.*

Layer menyatakan kumpulan dari urutan tahapan pengkodean yang dilakukan oleh *code block* yang terdapat dalam *subband* dan juga komponen. Kontribusi dari masing-masing *code block* terhadap *layer* berubah-ubah, tergantung *bit plane* yang diperhitungkan dalam *coding pass.* Setiap pertambahan *layer*, akan menambah kualitas citra dijital hasil kompresi yang dihasilkan.

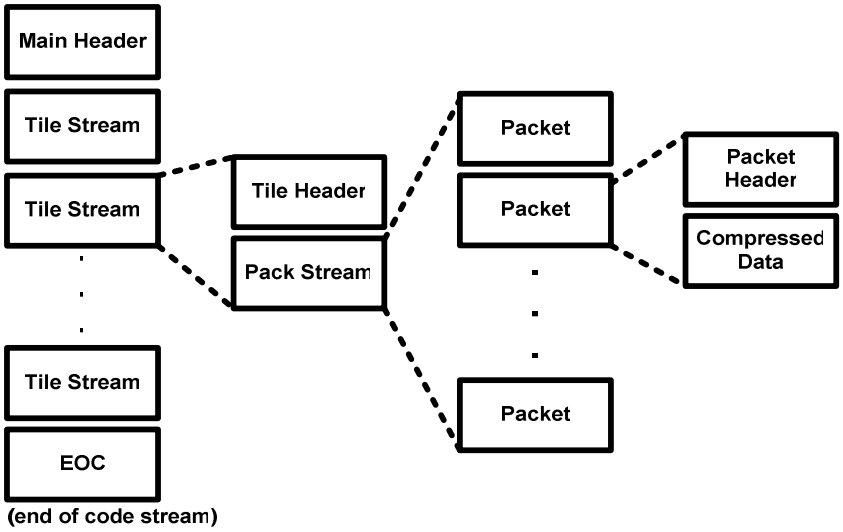
2.4.5 Rate Control

Rate Control merupakan proses pengaturan *code stream*, dalam rangka pencapaian *bit rate* yang diiginkan (*target bit rate*). Dalam penentuan *target bit*

rate ini, akan dilakukan perhitungan terhadap titik-titik potong optimal dari *bit stream* yang telah dihasilkan.

2.4.6 Bit Stream Organization

Dalam tahapan ini, akan dilakukan pengelompokan dari *bit stream* hasil kompresi, menjadi paket paket data. Paket-paket tersebut, dihasilkan dari *code block-code block* yang berada pada *subband-subband* dalam level resolusi yang sama. Masing-masing paket yang terbentuk akan memiliki paket *header*, yang antara lain berisi informasi mengenai *layer*, resolusi, komponen, serta posisi. Pengorganisasian dari *bit stream* dapat dilihat pada Gambar 2.11 berikut:



Gambar 2.11. Organisasi Code Stream

2.5 LZW *(Lempel Ziv Welth)*

Algoritma LZW dikembangkan dari metode kompresi yang dibuat oleh Ziv dan Lempel pada tahun 1977. Algoritma ini melakukan kompresi dengan menggunakan *dictionary*, di mana fragmen-fragmen teks digantikan dengan

indeks yang diperoleh dari sebuah “kamus”. Prinsip sejenis juga digunakan dalam kode Braille, di mana kode-kode khusus digunakan untuk merepresentasikan kata-kata yang ada.

Pendekatan ini bersifat *adaptif* dan efektif karena banyak karakter dapat dikodekan dengan mengacu pada *string* yang telah muncul sebelumnya dalam teks. Prinsip kompresi tercapai jika referensi dalam bentuk *pointer* dapat disimpan dalam jumlah bit yang lebih sedikit dibandingkan *string* aslinya. Algoritma kompresi LZW diberikan pada Gambar 2.12.

1. *Dictionary* diinisialisasi dengan semua karakter dasar yang ada : {‘A’..’Z’,’a’..’z’,’0’..’9’}.
2. *P* ← karakter pertama dalam *stream* karakter.
3. *C* ← karakter berikutnya dalam *stream* karakter.
4. Apakah *string* (*P* + *C*) terdapat dalam *dictionary* ?
   - Jika ya, maka *P* ← *P* + *C* (gabungkan *P* dan *C* menjadi *string* baru).
   - Jika tidak, maka :
     i. *Output* sebuah kode untuk menggantikan *string P*.
     ii. Tambahkan *string* (*P* + *C*) ke dalam *dictionary* dan berikan nomor/kode berikutnya yang belum digunakan dalam *dictionary* untuk *string* tersebut.
     iii. P ← C.
5. Apakah masih ada karakter berikutnya dalam *stream* karakter ?
   - Jika ya, maka kembali ke langkah 2.
   - Jika tidak, maka *output* kode yang menggantikan *string P*, lalu terminasi proses (*stop*).

Gambar 2.12. Algoritma Kompresi LZW

Sebagai contoh, string “ABBABABAC” akan dikompresi dengan LZW. Isi *dictionary* pada awal proses diset dengan tiga karakter dasar yang ada: “A”, “B”, dan “C”. Tahapan proses kompresi ditunjukkan pada Tabel 1. Kolom *posisi* menyatakan posisi sekarang dari *stream* karakter dan kolom *karakter* menyatakan karakter yang terdapat pada posisi tersebut.

Kolom *dictionary* menyatakan *string* baru yang sudah ditambahkan ke dalam *dictionary* dan nomor indeks untuk *string* tersebut ditulis dalam kurung siku. Kolom *output* menyatakan kode output yang dihasilkan oleh langkah kompresi. Hasil proses kompresi ditunjukkan pada Gambar 2.13.

Tabel 1. Tahapan proses kompresi LZW

Langkah	Posisi	Karakter	Dictionary	Output
1	1	A	[4] A B	[1]
2	2	B	[5] B B	[2]
3	3	B	[6] B A	[2]
4	4	A	[7] A B A	[4]
5	6	A	[8] A B A C	[7]
6	9	C	---	[3]

stream karakter	:	a	b	b	ab	aba	c
		↓	↓	↓	↓	↓	↓
kode *output*	:	[1]	[2]	[2]	[4]	[7]	[3]
frasa baru yang		4	5	6	7	8	
ditambahkan ke		=	=	=	=	=	
dictionary		ab	bb	ba	aba	abac	

Gambar 2.13. Hasil proses Kompresi

Proses dekompresi pada LZW dilakukan dengan prinsip yang sama seperti proses kompresi. Algoritma diberikan pada Gambar 4. Pada awalnya, *dictionary* diinisialisasi dengan semua karakter dasar yang ada. Lalu pada setiap langkah,

kode dibaca satu per satu dari *stream* kode, dikeluarkan *string* dari *dictionary* yang berkorespondensi dengan kode tersebut, dan ditambahkan string baru ke dalam *dictionary*. Tahapan proses dekompresi ini ditunjukkan pada Tabel 2.

Metode LZW yang diterapkan dalam penelitian ini bertipe dinamik, dimana hanya dilakukan satu kali pembacaan (*one-pass*) terhadap file yang akan dikompresi. Pengkodean data dilakukan secara *on the fly*, bersamaan dengan proses penambahan *string* baru ke dalam *dictionary*.

1. *Dictionary* diinisialisasi dengan semua karakter dasar yang ada : {‘A’..’Z’,’a’..’z’,’0’..’9’}.
2. *CW* ← kode pertama dari *stream* kode (menunjuk ke salah satu karakter dasar).
3. Lihat *dictionary* dan *output string* dari kode tersebut (*string.CW*) ke *stream* karakter.
4. *PW* ← *CW*; *CW* ← kode berikutnya dari *stream* kode.
5. Apakah *string.CW* terdapat dalam *dictionary* ?
   - Jika ada, maka :
     i. *output string.CW* ke *stream* karakter
     ii. *P* ← *string.PW*
     iii. *C* ← karakter pertama dari *string.CW*
     iv. tambahkan *string* (*P*+*C*) ke dalam *dictionary*
   - Jika tidak, maka :
     i. *P* ← *string.PW*
     ii. *C* ← karakter pertama dari *string.PW*
     iii. *output string* (*P*+*C*) ke *stream* karakter dan tambahkan string tersebut ke dalam *dictionary* (sekarang berkorespondensi dengan *CW*);
6. Apakah terdapat kode lagi di *stream* kode ?
   - Jika ya, maka kembali ke langkah 4.
   - Jika tidak, maka terminasi proses (*stop*).

Gambar 2.14. Algoritma Dekompresi LZW

Tabel 2. Tahapan Proses Dekompresi LZW

Langkah	Kode	Output	Dictionary
1	[1]	A	---
2	[2]	B	[4] A B
3	[2]	B	[5] B B
4	[4]	A B	[6] B A
5	[7]	A B A	[7] A B A
6	[3]	C	[8] A B A C

2.6 Perhitungan Kualitas Citra

Perhitungan kualitas citra digital yang merupakan hasil modifikasi, terhadap citra dijital yang asli, dapat dilakukan dengan menghitung nilai MSE (*Mean Square Error*) dan juga nilai PSNR (*Peak Signal-to-noise ratio*). *Peak Signal to Noise Ratio* (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR biasanya diukur dalam satuan desibel. Pada tugas akhir kali ini, PSNR digunakan untuk mengetahui kualitas (validasi) citra hasil kompresi.

Untuk menentukan PSNR, terlebih dahulu harus ditentukan nilai rata-rata kuadrat dari error (MSE - *Mean Square Error*). Perhitungan MSE adalah sebagai berikut :

$$MSE = \frac{1}{N.M} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left[\left\| f(i,j) - f'(i,j) \right\|^2 \right] \ldots (3)$$

Dimana :

MSE = Nilai *Mean Square Error* dari citra tersebut

M = panjang citra tersebut (dalam piksel)

N = lebar citra tersebut (dalam piksel)

(i,j) = koordinat masing-masing piksel

f(i,j) menyatakan citra dijital yang asli sebelum dikompresi, sedangkan *f'(i,j)*, merupakan citra dijital hasil kompresi. Nilai MSE yang besar, menyatakan bahwa penyimpangan atau selisih antara citra hasil modifikasi dengan citra aslinya cukup besar.

Sedangkan untuk perhitungan nilai PSNR, dapat dilakukan dengan rumus berikut:

$$PSNR = 10\log\left(\frac{(255)^2}{MSE}\right) = 20\log\left(\frac{255}{\sqrt{MSE}}\right) \ldots(4)$$

Semakin besar PSNR, maka kualitas citra hasil modifikasi akan semakin baik, sebab tidak banyak data yang mengalami perubahan, dibandingkan aslinya.

Menghitung rasio kompresi dan persentase :

Rasio = [*(((KCA - KCH) / KCA)*100)*] ... (5)

*Persentase = ((KCH*100) / KCA) %* ... (6)

Keterangan:

KCA = Kapasitas Citra Asli

KCH = Kapasitas Citra Hasil