BAB 2

LANDASAN TEORI

2.1 Pengertian Citra

Citra adalah representasi dari sebuah objek. Citra sebagai keluaran suatu sistem perekaman data dapat bersifat analog, berupa sinyal-sinyal video, seperti gambar pada monitor televisi atau yang bersifat digital yang dapat langsung disimpan pada suatu pita *magnetic*.

Menurut presisi yang digunakan untuk menyatakan titik-titik kordinat pada *domain spatial* (bidang), dan untuk menyatakan nilai keabuan atau warna dari suatu citra, maka secara teoritis citra dapat dikelompokkan menjadi empat kelas, yaitu : citra kontinu-kontinu, kontinu-diskrit, diskrit-kontinu, dan diskrit-diskrit. Dimana label pertama menyatakan presisi dari titik-titik koordinat pada bidang citra, sedangkan label kedua menyatakan presisi nilai keabuan atau warna. Kontinu dinyatakan dengan presisi angka tak terhingga, sedangkan diskrit dinyatakan dengan presisi angka terhingga. Komputer digital bekerja dengan angka-angka presisi terhingga. Dengan demikian, hanya citra dari kelas diskrit yang dapat diolah oleh komputer. Citra dari kelas tersebut lebih dikenal dengan citra digital.

Citra dalam format BMP lebih bagus dari pada citra dalam format JPEG, yang pada umumnya tidak dimampatkan sehingga tidak ada informasi yang hilang. Terjemahan bebas dari *bitmap* adalah pemetaan bit. Artinya, nilai intensitas *pixel* di dalam citra dipetakan ke sejumlah bit tertentu. Peta bit yang umum adalah 8, artinya setiap *pixel* panjangnya 8 bit. Delapan bit ini merepresentasikan nilai intensitas *pixel*. Dengan demikian ada sebanyak $2^8 = 256$ derajat keabuan, mulai dari 0 sampai 255.

Citra dalam format BMP ada tiga macam : citra biner, citra berwarna, dan citra hitam-putih (*grayscale*). Citra biner hanya mempunyai dua nilai keabuan, yaitu 0 dan 1. Oleh karena itu, 1 bit sudah cukup untuk merepresentasikan nilai *pixel*. Citra berwarna adalah citra yang lebih umum. Warna yang terlihat pada citra *bitmap* merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru. Setiap *pixel* disusun oleh tiga komponen warna : R (*red*), G (*green*), dan B (*blue*). Kombinasi dari tiga warna RGB tersebut menghasilkan warna yang khas untuk *pixel* yang bersangkutan.

Nilai setiap *pixel* tidak menyatakan derajat keabuan secara langsung, tetapi nilai *pixel* menyatakan indeks tabel RGB yang membuat nilai keabuan merah (R), nilai keabuan hijau (G), dan nilai keabuan biru (B) untuk *pixel* yang bersangkutan. Pada citra hitam-putih, nilai R = G = B untuk menyatakan bahwa citra hitam-putih hanya mempunyai satu kanal warna. Citra hitam-putih umumnya adalah citra 8-bit.

Citra yang lebih kaya warna adalah citra 24-bit. Setiap *pixel* panjangnya 24 bit, karena setiap *pixel* langsung menyatakan komponen warna merah, komponen warna hijau, dan komponen warna biru. Masing-masing komponen panjangnya 8 bit. Citra 24-bit disebut juga citra 16 juta warna, karena ia mampu menghasilkan 2^{24} = 16.777.216 kombinasi warna. (BN Budi Priyanto, 1992)

2.1.1 Citra Analog

Analog berhubungan dengan hal yang kontinu dalam satu dimensi, contohnya adalah bunyi diwakili dalam bentuk analog, yaitu suatu getaran gelombang udara yang kontinu dimana kekuatannya diwakili sebagai jarak gelombang. Hampir semua kejadian alam boleh diwakili sebagai perwakilan analog seperti bunyi, cahaya, air, elektrik, angin dan sebagainya. Jadi citra analog adalah citra yang terdiri dari sinyal - sinyal frekuensi elektromagnetis yang belum dibedakan sehingga pada umumnya tidak dapat ditentukan ukurannya.

2.1.2 Citra Digital

Citra digital adalah citra yang terdiri dari sinyal-sinyal frekuensi elektromagnetis yang sudah di-sampling sehingga dapat ditentukan ukuran titik gambar tersebut yang pada umumnya disebut *pixel*.

Untuk menyatakan citra (*image*) secara matematis, dapat didefinisikan fungsi f(x,y) di mana x dan y menyatakan suatu posisi dalam koordinat dua dimensi dan harga f pada titik (x,y) adalah harga yang menunjukkan warna citra pada titik tersebut. Citra digital adalah citra yang dinyatakan secara diskrit (tidak kontinu), baik untuk posisi koordinatnya maupun warnanya. Dengan demikian, citra digital dapat digambarkan sebagai suatu matriks, di mana indeks baris dan indeks kolom dari matriks menyatakan posisi suatu titik di dalam citra, dan harga dari elemen matriks menyatakan warna citra pada titik tersebut. Dalam citra digital yang dinyatakan sebagai susunan matriks seperti ini, elemen-elemen matriks tadi disebut juga dengan istilah *pixel* yang berasal dari kata *picture element*.

Untuk mendapatkan suatu citra digital, dapat digunakan alat yang memiliki kemampuan untuk mengubah sinyal yang diterima oleh sensor citra menjadi bentuk digital, misalnya dengan menggunakan kamera digital atau *scanner*. Cara yang lain adalah dengan menggunakan ADC (*Analog-to-Digital Converter*) untuk mengubah sinyal analog yang dihasilkan oleh keluaran sensor citra menjadi sinyal digital, misalnya dengan menggunakan perangkat keras *video capture* yang dapat mengubah sinyal video analog menjadi citra digital. (Aniati Murni, 1992).

2.2 Format *File* Citra

Sebuah format citra harus dapat menyatukan kualitas citra, ukuran *file* dan kompatibilitas dengan berbagai aplikasi. Saat ini tersedia banyak format grafik, dan format baru tersebut dikembangkan, diantaranya yang terkenal adalah BMP, JPEG, dan GIF. Setiap program pengolahan citra biasanya memiliki format citra tersendiri. Format dan metode dari suatu citra yang baik juga sangat bergantung pada jenis

citranya. Setiap format *file* citra memiliki kelebihan dan kekurangan masing-masing dalam hal citra yang disimpan.

Citra tertentu dapat disimpan dengan baik (dalam arti ukuran *file* lebih kecil dan kualitas gambar tidak berubah) pada format *file* citra tertentu, apabila disimpan pada format lain kadang kala dapat menyebabkan ukuran *file* menjadi lebih besar dari aslinya dan kualitas citra dapat menurun. Oleh karena itu, untuk menyimpan suatu citra harus diperhatikan citra dan format *file* citra apa yang sesuai. Misalnya format citra GIF sangat tidak cocok untuk citra fotografi, karena biasanya citra fotografi kaya akan warna, sedangkan format GIF hanya mendukung sejumlah warna sebanyak 256 (8 bit) saja. Format JPEG merupakan pilihan yang tepat untuk citra-citra fotografi, karena JPEG sangat cocok untuk citra dengan perubahan warna yang halus. (Aniati Murni, 1992).

2.2.1 Format *Bitmap*

Pada format *bitmap*, citra disimpan sebagai suatu matriks di mana masing-masing elemennya digunakan untuk menyimpan informasi warna untuk setiap *pixel*. Jumlah warna yang dapat disimpan ditentukan dengan satuan *bit-per-pixel*. Semakin besar ukuran *bit-per-pixel* dari suatu *bitmap*, semakin banyak pula jumlah warna yang dapat disimpan.

Format *bitmap* ini cocok digunakan untuk menyimpan citra digital yang memiliki banyak variasi dalam bentuknya maupun warnanya, seperti foto, lukisan, dan *frame* video. Format *file* yang menggunakan format *bitmap* ini antara lain adalah BMP, DIB, PCX, GIF, dan JPG. Format yang menjadi standar dalam sistem operasi Microsoft Windows adalah format *bitmap* BMP atau DIB.

Karakteristik lain dari *bitmap* yang juga penting adalah jumlah warna yang dapat disimpan dalam *bitmap* tersebut. Ini ditentukan oleh banyaknya bit yang digunakan untuk menyimpan setiap titik dari *bitmap* yang menggunakan satuan bpp (*bit per pixel*). Dalam Windows dikenal *bitmap* dengan 1, 4, 8, 16, dan 24 *bit per*

pixel. Jumlah warna maksimum yang dapat disimpan dalam suatu *bitmap* adalah sebanyak 2^n, dimana n adalah banyaknya bit yang digunakan untuk menyimpan satu titik dari *bitmap*. (Aniati Murni, 1992).

Berikut ini tabel yang menunjukkan hubungan antara banyaknya *bit per pixel* dengan jumlah warna maksimum yang dapat disimpan dalam *bitmap*.

Tabel 2.1 Hubungan Antara *Bit Per Pixel* dengan Jumlah Warna Maksimum Pada *Bitmap*

No	Jumlah *bit per pixel*	Jumlah warna maksimum
1	1	2
2	4	16
3	8	256
4	16	65536
5	24	16777216

2.2.2 Format JPEG

JPEG merupakan singkatan dari *Joint Photographic Experts Group*, merupakan suatu komite yang menyusun standar citra pada akhir tahun 80-an dan awal tahun 90-an. Kata "*Joint*" pada JPEG melambangkan status data di dalam kerja sama panitia ISO dan ITU_T. Format yang dihasilkan merupakan standar ISO IS-10918.

Format *file* ini dikembangkan oleh *C-Cube Microsystems* untuk memberikan sebuah metode yang efisien untuk menyimpan citra dengan jumlah warna yang sangat banyak seperti foto kamera. Perbedaan utama antara format JPEG dengan format citra yang lainnya adalah bahwa *file* JPEG menggunakan metode *lossy* untuk proses pemampatannya. Pemampatan secara *lossy* akan membuang sebagian data citra untuk memberikan hasil kompresi yang baik. Hasil *file* JPEG yang didekompres tidak begitu sesuai dengan citra aslinya, tetapi perbedaan ini sangat sulit dideteksi oleh mata manusia. (Munir Rinaldi, 2004).

2.3 Kompresi Data

Kompresi data adalah metode yang dilakukan untuk mereduksi ukuran data atau *file*. Dengan melakukan kompresi atau pemadatan data, maka ukuran *file* atau data akan lebih kecil sehingga dapat mengurangi waktu transmisi saat data dikirim, dan tidak banyak menghabiskan ruang medis penyimpanan.

Seiring dengan bertambahnya waktu, ukuran *file* di dalam media penyimpanan akan semakin bertambah. Hal ini disebabkan karena semakin bertambahnya informasi yang disimpan dalam *file* ataupun penambahan program aplikasi baru di dalam komputer. Sementara kapasitas media penyimpanan pada komputer adalah bersifat tetap, sehingga *user* akan mengalami masalah dengan ruang kosong (*free space*) yang semakin kecil.

Untuk mengatasi masalah kapasitas penyimpanan, sangat penting bagi *user* untuk menjaga agar ruang kosong *hard disk* tetap optimal. Salah satu cara yang dapat dilakukan adalah dengan mengkompresi *file* data. Seiring itu, memasuki era internet, *transfer file* merupakan hal yang umum dilakukan oleh *user*. *Transfer file* dalam jaringan yang dapat dilakukan oleh *user* yaitu proses *download* (men-*transfer* suatu *file* dari komputer remote ke komputer yang meminta proses pengiriman melalui jaringan) dan *upload* (men-*transfer file* dari komputer lokal ke komputer *remote*).

Bila *file* yang hendak ditransfer berukuran besar, maka dapat menyebabkan data kongesti, yaitu dalam keadaan dimana informasi yang hendak ditransfer lebih besar daripada yang dapat dibawa oleh jalur komunikasi. Untuk meminimalisasi ukuran *file* yang hendak di-*transfer* dapat dilakukan dengan cara kompresi yaitu *file* yang hendak di-*transfer* diperkecil ukurannya terlebih dahulu.

Dengan teknik kompresi, *file* yang besar dapat diperkecil sedemikian rupa sehingga menghemat tempat namun masih dapat dikembalikan ke bentuk original ketika diperlukan. Kompresi berguna ketika hendak memadatkan sejumlah besar *file* ke dalam tempat penyimpanan atau hendak memperkecil waktu ketika hendak menyalin atau mentransmisi *file* melalui jaringan.

Terdapat berbagai perangkat seperti *Compress*, *Zip* atau *Pack* seperti *Stuffit* atau *Zipit* yang digunakan untuk mengkompresi berbagai jenis *file*. Algoritma ini hanya tergantung pada pemahaman isi *file* dan mencari redudansi dari pola sehingga dimungkinkan untuk kompresi.

Algoritma kompresi lainnya (seperti JPEG dan MPEG) bersifat *lossy* yaitu pada saat dekompresi tidak dapat mengembalikan bentuk *file* ke sifat originalnya. Algoritma demikian mengkompresi dengan meringkas data. Hasil ringkasan mempertahankan struktur umum dari data tetapi membuang beberapa detailnya. Untuk format suara, video dan gambar, ketidakakuratan ini masih bisa diterima, karena inti dari data masih tetap dipertahankan dan hanya menghilangkan beberapa *pixel* atau beberapa milidetik penundaan tampilan video. Namun untuk data teks, algoritma '*lossy'* tidak sesuai untuk diterapkan. Salah satu contoh algoritma untuk mengkompresi *file* teks dengan data 'Kompresi Metode Algoritma *Run Length Encoding*' dengan menghilangkan semua vocal akan menghasilkan teks menjadi 'Kmprs Mtd lgrtm Rn Lngth ncdng'.

Data asli dengan 45 karakter telah dikompresi menjadi 30 karakter dan hanya 66% dari ukuran aslinya. Untuk melakukan dekompresi, dapat dicoba untuk mencocokkan huruf konsonan dengan memasukkan huruf *vocal*, namun sangat sulit untuk menghasilkan kembali bentuk originalnya, karena luasnya kemungkinan kata *artificial intelligence* yang rumit, sehingga untuk *file* teks, tidak dapat digunakan algoritma kompresi *lossy*. (Munir Rinaldi, 2004).

2.3.1 Dekompresi

Sebuah data yang sudah dikompres tentunya harus dapat dikembalikan lagi kebentuk aslinya, prinsip ini dinamakan dekompresi. Untuk dapat merubah data yang terkompres diperlukan cara yang berbeda seperti pada waktu proses kompres dilaksanakan. Jadi pada saat dekompres terdapat catatan *header* yang berupa *byte-byte* yang berisi catatan mengenai isi dari *file* tersebut.

Catatan *header* akan menuliskan kembali mengenai isi dari *file* tersebut, jadi isi dari *file* sudah tertulis oleh catatan *header* sehingga hanya tinggal menuliskan kembali pada saat proses dekompres. Proses dekompres dikatakan sempurna apabila file kembali kebentuk aslinya. (Munir Rinaldi, 2004).

2.3.2 *Lossless* dan *Lossy Compression*

Dalam kompresi data *lossless*, data yang dikompresi dan didekompresi mempunyai replikasi yang sama dengan data asli. Sedangkan pada kompresi data *lossy*, data yang didekompresi dapat berbeda dari data asli.

Program kompresi data populer seperti WinZip, WinRar dan Win Ace merupakan salah satu contoh data kompresi data *lossless*. JPEG merupakan salah satu contoh dari kompresi data *lossy*.

2.3.3 Perbedaan antara *Compression Rate* dan *Compression Ratio*

Terdapat dua jenis aktivitas utama dalam aplikasi kompresi data, yaitu transmisi dan penyimpanan. Contoh aplikasi transmisi adalah *speech compression*, untuk transmisi secara *real time* melalui jaringan digital selular, sedangkan contoh aplikasi penyimpanan adalah kompresi *file* (contoh seperti program *Drivespace* dan *Doublespace*).

Istilah *"Compression Rate"* dipakai dalam transmisi, sementara istilah "Compression Ratio" berasal dari istilah teknik penyimpanan data. *Compression Rate* merupakan transmisi data dimana laju kompresi dari data yang dikompresi. Secara tipikal satuannya adalah *bits/sample, bits/character, bit/pixel*, atau *bits/second. Compression Ratio* merupakan teknik penyimpanan data dimana rasio atau perbandingan antara ukuran atau laju data yang dikompresi dengan ukuran atau laju dari data asli.

$$Compression\ Ratio = \frac{Size\ or\ rate\ of\ Compression\ Data}{Size\ or\ rateof\ Original\ Data} x100\% \quad \ldots\ldots\ldots\ldots (1)$$

Sebagai contoh, jika suatu image *grayscale* aslinya direpresentasikan oleh 8 *bits*/*pixel* (bpp) dan jika kompresi hingga 2 bpp, maka dapat dikatakan rasio kompresinya adalah 1 banding 4 (1:4), atau istilah lainnya adalah 75%. (Munir Rinaldi, 2004).

2.4 Metode Kompresi Data

Organisasi–organisasi yang mengoperasikan jaringan komputer sering kali mengharapkan dapat menekan biaya pengiriman data. Biaya pengiriman itu sangat tergantung dengan banyaknya *byte* data yang dikirimkan. Oleh karena itu, dengan melakukan kompresi terhadap data, akan dapat menghemat biaya pengiriman.

Istilah kompresi tersebut diterjemahkan dari kata bahasa Inggris "*compression*" yang berarti pemampatan. Dalam bidang teknik, kompresi berarti proses pemampatan sesuatu yang berukuran besar sehingga menjadi kecil. Dengan demikian, kompresi data berarti proses untuk pemampatan data agar ukurannya menjadi lebih kecil.

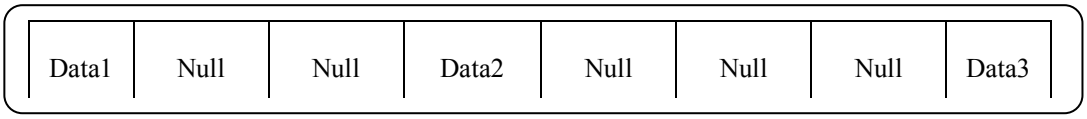
Pemampatan ukuran berkas melalui proses kompresi hanya diperlukan sewaktu berkas tersebut akan disimpan dan atau dikirim melalui media transmisi atau telekomunikasi. Apabila berkas tersebut akan ditampilkan lagi pada layar monitor, maka data yang terkompresi tersebut harus dikembalikan pada format semula agar dapat dibaca kembali. Proses mengembalikan berkas yang dimampatkan ke bentuk awal inilah yang disebut dekompresi.

Satuan yang cukup penting dalam kompresi data adalah *compression ratio* yang menggambarkan seberapa besar ukuran data setelah melewati proses kompresi dibandingkan dengan ukuran berkas yang asli.

Beberapa teknik kompresi data yang telah banyak digunakan adalah *Bit Mapping*, *Half byte packing*, *Diatomic Encoding*, *Huffman*, *Arithmatic Encoding*, *Run Length Encoding*. (Munir Rinaldi, 2004).

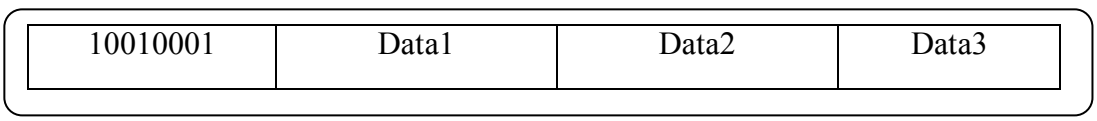
2.4.1 *Bit Mapping*

Pada teknik ini digunakan *bitmap* (peta bit) yang berisi data posisi dari karakter-karakter yang bukan *null*. Gambar 2.1 berikut ini adalah data asli data sebelum dikompres. Dari gambar tersebut dapat dilihat bahwa dari seluruh data (8 karakter) hanya ada tiga buah karakter yang bukan null.



Gambar 2.1 Data sebelum dikompresi

Pada gambar 2.1, dapat dilihat bahwa karakter yang disisipkan sebagai *bitmap* ditulis dalam biner, yaitu 10010001. Urutan bit semacam ini memberi informasi bahwa data yang bukan null terletak pada posisi pertama, keempat dan kedelapan. Apabila banyaknya karakter data yang dikompres cukup panjang, maka diperlukan *bitmap* yang lebih panjang pula. Dalam prakteknya, agar manipulasi *bitmap* lebih mudah ditentukan, maka panjang *bitmap* merupakan kelipatan 8, yang merupakan satuan standar dari sebagian besar komputer.



Gambar 2.2 *Bitmap* data setelah dikompresi

Apabila data pada gambar 2.2. dikompres dengan teknik *bitmap*, maka ukuran berkas menjadi 4 karakter. Itu berarti setengah dari ukuran data semula. Untuk data yang berukuran besar, angka ini cukup menguntungkan.

Disamping kelebihan yang ada, teknik ini juga memiliki sedikitnya dua kelemahan. Pertama, keharusan bahwa ukuran semua data harus sama, misalnya 8 bit atau 16 bit. Apabila ukuran data yang akan dikompres berbeda, penentuan *bitmap*-nya akan mengalami kesulitan. Kedua, teknik ini tidak mempertimbangkan karakter lain yang frekuensinya diawali karakter yang akan dihilangkan (misalnya *null*), meskipun frekuensinya tidak jauh berbeda. Sebagai contoh, dalam suatu berkas *null* muncul sebanyak 100 kali, dan karakter "a" muncul 900 kali. Berdasarkan metode *bitmap* maupun *null suppression*, maka yang harus dihilangkan adalah karakter *null*, meskipun frekuensi kemunculan "a" juga cukup tinggi. Dengan demikian, seolah-olah teknik ini mengabaikan kesempatan untuk menghasilkan kompresi yang lebih baik. Untuk mengantisipasi kelemahan ini, ada teknik lain yang bisa memperbaikinya, yaitu teknik *run length*. (Munir Rinaldi, 2004).

2.4.2 *Half Byte Packing*

Teknik *Half Byte Packing* dapat dipandang sebagai turunan dari teknik *bit mapping*. Teknik ini memanfaatkan sifat-sifat yang unik dari suatu jenis data. Sebagai contoh, data yang berupa bilangan-bilangan desimal, tentu *interval* simbolnya hanya dari 0 sampai dengan 9. Kalau bilangan-bilangan tersebut disajikan dengan standar ASCII maka penyajian binernya cukup unik seperti terlihat pada tabel 2.2.

Tabel 2.2 Penyajian Bilangan Biner

No	Desimal	Biner
1	0	1100 0000
2	1	1100 0001
3	2	1100 0010
4	3	1100 0011
5	4	1100 0100
6	5	1100 0101
7	6	1100 0110
8	7	1100 0111
9	8	1100 1000
10	9	1100 1001

Dari tabel diatas terlihat bahwa 4 bit pertama dari setiap bilangan adalah sama, yaitu 1100. Dengan demikian, apabila 4 bit pertama ini dihilangkan, tidak akan mengubah informasi yang terkandung, asalkan ada kode khusus yang mencatat hal ini. Oleh karena banyaknya bit yang bisa dihilangkan sebanyak 4, maka satu *byte* akan bisa menampung 2 buah karakter.

Seperti halnya pada teknik-teknik lain, dalam *half byte packing* juga memerlukan karakter khusus sebagai tanda dari sederetan data yang sudah dikompres. Format dari teknik *half byte packing* dapat dilihat pada tabel 2.3.

Tabel 2.3 Format Teknik *Half Byte Packing*

Sc	Hbc	N2	N4	N6	…	N14
	N1	N3	N5	N7		N15

Keterangan dari tabel 2.3. adalah sebagai berikut : Sc adalah karakter khusus sebagai tanda kompresi, Hbc adalah penghitung banyaknya bilangan yang dikompres, Ni adalah bilangan yang sudah dikompres.

Pada teknik ini, penghitung banyak karakter yang dikompres biasanya terdiri dari 4 bit. Dengan demikian, hanya 16 deret data yang bisa dicatat yang masing-masing berukuran 4 bit. Contoh berikut menjelaskan kompresi untuk data berupa tanggal " 20 Januari 1995".

Data asli dalam bentuk desimal

2011995

Data asli dalam bentuk biner

1100 0010 1100 0000 1100 0001 1100 0001

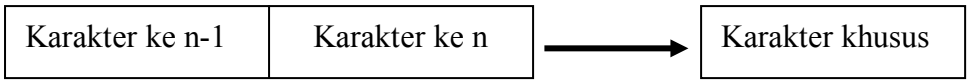
1100 1001 1100 1001 1100 0101

Data yang dikompres

@1100 0010 0000 0001 0001 1001 1001 0101

2.4.3 *Diatomic Encoding*

Diatomic encoding adalah teknik kompresi yang bekerja dengan cara mengganti dua buah kombinasi karakter dengan sebuah karakter lain. Teknik *diatomic encoding* memanfaatkan saat bahwa ada kombinasi tertentu dari dua buah karakter yang sering muncul pada suatu teks tertentu. Proses *diatomic encoding* dapat dilihat pada gambar 2.3.



Gambar 2.3 Proses *Diatomic Encoding*

Masing–masing bahasa maupun jenis berkas mempunyai ciri yang khas dalam hal kombinasi karakter–karakternya. Sebagai contoh dalam bahasa Indonesia banyak ditemui kombinasi karakter "a" dengan karakter "n", karena kombinasi ini disubsitusikan dengan sebuah karakter lain yang khas, maka akan diperoleh pengurangan ukuran berkas maksimum setengahnya.

2.4.4 Algoritma Kompresi Huffman

Algoritma Huffman dikembangkan oleh David A Huffman saat di bangku kuliah di *Massachusetts Institute Technology* . Algoritma ini merupakan karya ilmiah dari Huffman yang ditulis pada tahun 1952 dengan judul '*A Method for the Construction of Minimum Redundancy Codes*'. Kode Huffman merupakan hasil pengembangan dari algoritma yang sebelumnya dikembangkan oleh Claude Shannon dan R.M Fano pada awal tahun 1950 dan dikenal sebagai Shannon – Fano Coding.

Algoritma Huffman termasuk jenis kompresi '*lossless'*. *Lossless* artinya bahwa kompresi dilakukan tanpa kehilangan informasi (yaitu pada proses dekompresi menghasilkan *file* sebenarnya). Kelebihan algoritma ini terletak pada kebanyakan *file* data mengandung redundansi. Huffman menemukan cara untuk memanfaatkan redundansi pada suatu *file* sehingga mengecilkan ukuran *file* tanpa kehilangan informasi yang terkandung di dalamnya. Pengertian redundansi dalam algoritma

Huffman mengacu pada perulangan karakter tertentu lebih dari satu kali dalam suatu dokumen.

Algoritma ini menggunakan suatu variabel sebagai representasi biner untuk setiap karakter dalam suatu dokumen, dengan memberikan biner yang terpendek dan karakter dengan frekuensi yang lebih jarang muncul sebagai representasi biner yang terpanjang. Cara ini dapat mengurangi ukuran *file* secara signifikan karena representasi konvensional dari karakter menggunakan panjang yang sama untuk setiap karakter, sehingga menghabiskan banyak tempat. (Munir Rinaldi, 2004).

2.4.5 *Arithmatic Encoding*

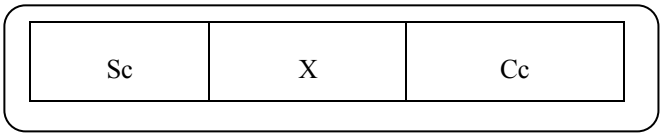
Metode *arithmetic encoding* diusulkan oleh Elias, dan yang diperkenalkan oleh Abramson dalam *Information Theory* (Abramson 1963). Implementasi tentang teknik Elias dikembangkan oleh Rissanen (1976), Pasco (1976), Rubin (1979), dan Witten et al (1987). Algoritma *arithmetic encoding* termasuk jenis kompresi '*lossless'*.

Di dalam metode *arithmetic encoding* suatu pesan yang telah dikompresi diwakili oleh suatu nilai statistik yang berkisar antara 0 dan 1. Masing-masing karakter dari pesan sumber diwakili oleh nilai interval yang berkisar antar 0 dan 1. (Gonzales Rafael C, 1992)

2.4.6 *Run Length Encoding* (RLE)

Berbeda dengan teknik-teknik sebelumnya yang bekerja berdasarkan karakter per karakter, teknik *run length* ini bekerja berdasarkan sederetan karakter yang berurutan. *Run Length Encoding* adalah suatu algoritma kompresi data yang bersifat *lossless*. Algoritma ini mungkin merupakan algoritma yang paling mudah untuk dipahami dan diterapkan.

Run Length Encoding (RLE) adalah algoritma kompresi yang sangat mendasar. RLE mempunyai hak paten bebas, yang berarti seseorang dapat menggunakan algoritma kompresi RLE dengan bebas. Metode kompresi ini sangat sederhana, yaitu hanya memindahkan pengulangan *byte* yang sama berturut-turut (secara terus menerus). Data masukan akan dibaca dan sederetan karakter yang sesuai dengan deretan karakter yang sudah ditentukan sebelumnya disubstitusi dengan kode tertentu. Kode khusus ini biasanya terdiri dari tiga buah karakter, seperti yang terlihat pada gambar 2.4.



Gambar 2.4 Tiga buah karakter

Dari gambar 2.4. diatas kita dapat melihat bahwa : Sc adalah karakter khusus yang dipakai sebagai tanda kompresi. X adalah karakter berurutan yang akan dikompres, Cc adalah banyaknya karakter yang akan dihapus. Sebagai contoh, apabila karakter khusus (Sc) yang digunakan adalah # dan Cc dalam bilangan desimal, maka jika digunakan untuk mengkompres potongan string : “jarrrrrringan”, maka akan diperoleh hasil sebagai berikut “Jar#6ingan”. Misalnya sebuah sumber pesan yang berisi “aaaabccccbbbbabbbb”, maka dengan menggunakan algoritma kompresi data RLE akan menjadi “a4b1c4b4a1b4”. Kompresi metode RLE adalah menjumlahkan pengulangan *byte* atau karakter yang sama berturut-turut dan menampilkan hanya sebuah karakter yang mengalami pengulangan disertai dengan nilai jumlah pengulangan *byte* atau karakter, sedang untuk *byte* atau karakter yang tidak terjadi pengulangan maka karakter tersebut tidak akan dikompresi.

Misalkan pada kata “aaab” yang terjadi pengulangan *byte* atau karakter “a” sebanyak 3 kali dan *byte* atau karakter “b” tidak mengalami perulangan sehingga kata tersebut akan dikompresi menjadi “a3b”. Contoh lain, pada pesan “aaaabccccbbbbabbbb”, karakter pertama “a”, karakter kedua “a”, karakter ketiga “a”, karakter keempat “a”, dan karakter kelima “b”, dikarenakan pada karakter kelima tidak sama dengan karakter sebelumnya, sehingga 4 karakter pertama yang mengalami perulangan akan dijumlahkan semuanya dan nilai yang dijumlah adalah banyaknya

karakter yang akan diulang, sehingga output keempat karakter yang pertama setelah dikompresi adalah hanya sebuah karakter dan diikuti dengan nilai perulangan yaitu "a4". Setelah dilakukan kompresi 4 karakter pertama, akan dilanjutkan ke karakter berikutnya yaitu karakter kelima "b", kemudian karakter keenam "c", dikarenakan karakter keenam tidak sama dengan karakter kelima, dan karakter kelima tidak mengalami perulangan sehingga karakter yang tidak mengalami perulangan akan ditambahkan kepada pesan "a4", sehingga menjadi "a4b", dan seterusnya sehingga pesan "aaaabccccbbbbabbbb" setelah dikompresi akan menjadi "a4bc4b4ab4", dapat diperhatikan bahwa pesan "aaaabccccbbbbabbbb" yang berukuran 18 *byte* atau karakter dapat dilihat dari tabel 2.4.

Tabel 2.4 Metode Kompresi RLE

No	Simbol Masukan	Simbol Keluaran	Total
1	aaaa	a	4
2	b	b	1
3	cccc	c	4
4	bbbb	b	4
5	a	a	1
6	bbbb	b	4

Dekompresi metode RLE adalah dengan menguraikan nilai angka menjadi karakter yang diikuti nilai angka sebanyak nilai angka tersebut, misalnya kata "a4b", nilai 4 menyatakan banyaknya huruf "a" yang diundang, sehingga setelah di dekompresi kata "a4b" akan menjadi "aaaab".

Tabel 2.5 Metode Dekompresi RLE

No	Simbol Masukan	Total	Simbol Keluaran
1	a	4	aaaa
2	b	1	b
3	c	4	cccc
4	b	1	a
5	a	1	a
6	b	4	bbbb

Untuk mencatat tiap-tiap *byte* nilai yang diikuti maka digunakan suatu *counter* (pencatat jumlah banyaknya data yang keluar), *counter* menandakan berapa banyak pengulangan *byte* yang tercatat. Ini adalah suatu gagasan dasar metode kompresi *Run Length Encoding*, metode kompresi *Run Length Encoding* tidaklah akan mempunyai suatu hasil baik dan sebagai contoh untuk *file* yang lebih besar, sehingga untuk mengkompresi data yang kecil tidak menguntungkan, tetapi contoh di bawah ini akan menunjukkan implementasi yang sesuai, menetapkan *repeatition minimum* 2 dan akan memampatkan hanya jika dua atau lebih *bytes* adalah sama. Sebagai contoh dari pesan adalah mencari dua *byte* nilai yang sama dan mencatat suatu nilai berapa banyak *byte* nilai yang sama dan berulang.

Teknik *run length* ini mempunyai kelemahan, yaitu terbatasnya bilangan Cc. Apabila Cc dalam bilangan desimal untuk jumlah karakter lebih dari 9 buah, maka diperlukan lebih dari satu digit Cc. Oleh karena itu, untuk mengkodekan karakter Cc, digunakan untuk mengkodekan 256 karakter. Walaupun jumlah ini tidak terlalu besar, tetapi untuk kebanyakan berkas teks, jumlah ini sudah lebih dari cukup.

Algoritma metode kompresi RLE hanya efisien dengan data *file* yang berisi kelompok data (*byte* / karakter) yang berulang dan dapat digunakan pada *file* teks. *File* teks berisi banyak kelompok data yang berupa spasi atau tabulator, tetapi juga dapat diterapkan untuk citra (gambar) yang berisi area hitam atau putih yang besar.

Secara fisik, citra adalah representasi dari objek-objek dalam keadaan diam atau bergerak pada suatu media seperti kertas, film atau monitor. Representasi objek

untuk menghasilkan suatu citra dapat dilakukan secara analog dan secara digital. Pengolahan citra secara analog relatif sangat terbatas, karena analisa citra hanya dapat dilakukan secara visual dan memerlukan adaptasi sebaik mungkin pada sifat-sifat optik dan fisik dari sistem visual manusia. Pengolahan citra digital dilakukan dengan cara merepresentasikan setiap warna pembentuk citra dengan suatu angka, sehingga memungkinkan untuk mengolahnya dengan computer. Dewasa ini pengolah citra digital cukup berkembang pesat dan diterapkan secara luas untuk memenuhi berbagai kebutuhan. (Gonzales Rafael C, 1992)

2.5 Jenis Kompresi Data

Algoritma untuk jenis data *lossless* (tanpa kehilangan data) yang banyak digunakan antaranya : Huffman, RLE (*Run Length Encoding*), LZ77 (Limpel-Zev), LZ78, dan LZW (Limpel-Zel-Welch), BZIP2, *Eliminator* dan *Variable Bit Count* (VBC). Sedangkan untuk jenis kompresi *lossy* (kehilangan beberapa bagian data), algoritma yang banyak digunakan antara lain : *Differential Modulation*, *Adaptive Coding* dan *Discrete Cosine Transform* (DTC).

2.6 Windows API (*Application Programming Interface*)

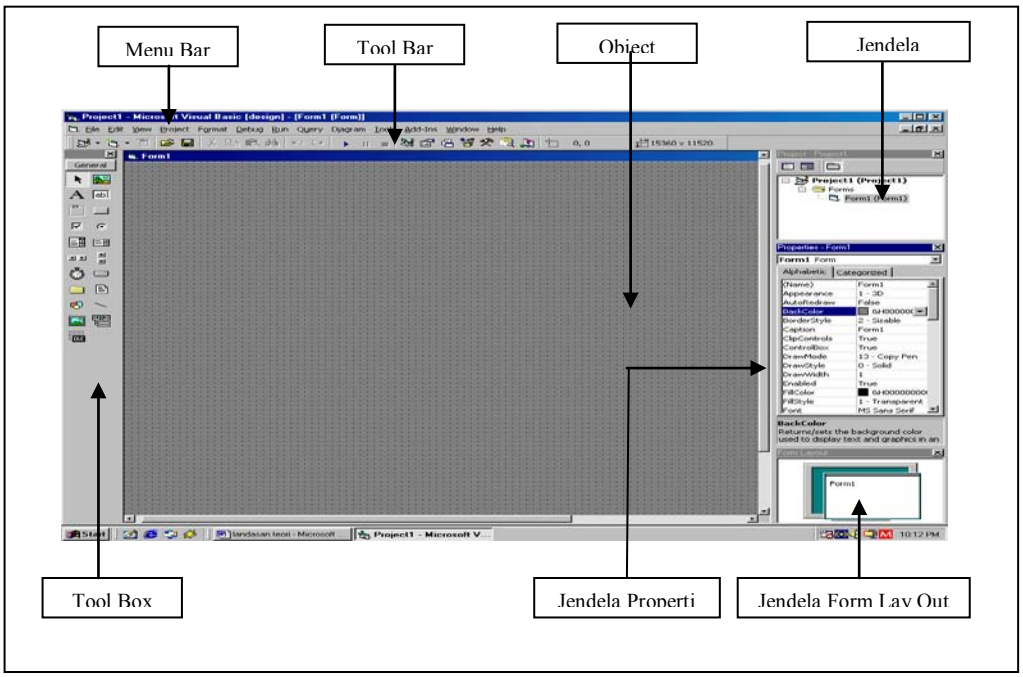
Windows API adalah sekumpulan fungsi-fungsi eksternal yang terdapat dalam *file-file* perpustakaan *Windows* (*library windows*) atau *file library* lainnya yang dapat digunakan dalam pembanguna program. Fungsi ini dapat menangani semua yang berhubungan dengan *Windows* seperti pengaksesan disk, *interface printer*, grafik Windows, kotak dialog, Windows *shell*, memainkan musik dan sebagainya. (Hadi Rahadian, 1997)

2.7 Visual Basic 6.0

Visual Basic merupakan bahasa pemrograman yang berorientasi objek (*Object Oriented Programming / OOP*). OOP adalah pemrograman yang terdiri dari beberapa objek yang berkomunikasi atau berhubungan dan melakukan suatu aksi dalam suatu kejadian (*event*), sehingga istilah objek banyak digunakan dalam pemrograman *Visual Basic* ini. Objek-objek digambarkan pada layar dan melakukan properti terhadap objek yang digambarkan lalu menuliskan metode-metode terhadap objek tersebut sesuai dengan tujuan program.

Pada pemrograman *Visual Basic*, perancangan program dimulai dengan perencanaan dan pendefenisian tujuan program, lalu merancang keluaran dan media hubungan dengan pemakai, dan langkah terakhir adalah penulisan kode program tersebut. *Visual Basic* menyediakan IDE (*Integrated Development Environment*) sebagai lingkungan tempat bekerja untuk menghasilkan program aplikasi pada *Visual Basic*. Komponen-komponen IDE terdiri dari *control* menu, baris menu, *toolbar*, *toolbox*, *form window*, *form layout window*, *properties window, project explorer, kode window, object window dan event window,* yang mana setiap komponen memiliki tujuan dan kegunaan masing-masing.

Visual Basic memiliki beberapa jenis form. Dalam perancangan ini form yang digunakan adalah form induk (*MDI Form*) dan form anak (*MDI Child*). MDI singkatan dari *Multiple Document Interface.* MDI dirancang untuk program aplikasi yang membutuhkan banyak form. MDI Form adalah jenis form yang berfungsi sebagai pusat pengaturan *form-form* lain atau disebut *form* induk karena dapat menampilkan *form* lain di dalamnya.



Gambar 2.5 Menu Utama Visual Basic 6.0

(M. Agus J Alam, 2000)