penting karena memperkecil kebutuhan penyimpanan data, mempercepat pengiriman data, memperkecil kebutuhan *bandwidth*.

Dalam teknik kompresi data, redundansi dari data menjadi masalah utama. Kompresi data ditujukan untuk mereduksi penyimpanan data yang redundan Atau dalam istilah lain kompresi citra digital dilakukan untuk dengan cara meminimalkan jumlah bit yang diperlukan untuk merepresentasikan suatu data citra, namun seringkali kualitas gambar yang dihasilkan jauh lebih buruk dari aslinya karena keinginan kita untuk memperoleh rasio kompresi yang tinggi.

2.1.1 Faktor penting kompresi data

Dalam kompresi data terdapat 4 (empat) faktor penting yang perlu diperhatikan, yaitu: Time proses (waktu yang dibutuhkan untuk menjalankan prose), Completeness (kelengkapan data setelah file-file tersebut dikompres), Optimaly (perbandingan apakah ukuran file sebelum dikompres sama atau tidak dengan file yang telah dikompres). Tidak metode kompresi yang paling efektif untuk semua jenis file

2.1.2 Metode kompresi data

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi file input) menjadi sekumpulan code word, metode kompresi menjadi dua kelompok, yaitu:

a. Metode static : mennggunakan peta kode yang selalu sama. Metode ini membutuhkan dua fase (two pass) : fase pertama untuk menghitung

probabilitas kemunculan tiap symbol / karakter dan menentukan peta kodenya. Fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditranmisikan. Contoh: Algoritma Huffman static.

b. Metode dinamik (Adaftif) : menggunakan peta kode yang dapat berubah dari waktu ke waktu. Metode ini disebut adaftif karena peta kode mampu beradaptasi terkadang karakteristik isi file selama proses kompresi berlangsung. Metode ini bersifat one pass, karena isi file selam dikompres hanya diperlukan satu kali pembacaan terhadap isi file. Contoh: Algoritma LZW dan DMC.

Berdasarkan teknik pengkodean atau pengubahan simbol yang digunakan. Kompresi dapat dibagi kedalam 3 kategori, yaitu:

a. Metode *simbolwise*

Menghitung peluang kemunculan dari tiap symbol dalam file input, lalu mengkodekan satu symbol dalam satu waktu, dimana symbol yang lebih sering muncul diberikan lebih muncul. Contoh: Algoritma Huffman, Aritmetic Coding.

b. Metode *Dictionary*

Menggantikan karakter / fragmen dalam file input dengan indeks lokasi dari karakter / fragmen tersebut dalam sebuah kamus (Dictionary). Contoh: Algoritma LZW.

c. Metode *Predictive*

Menggunakan metode finite context atau finite state untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya. Contoh: Algoritma DMC.

2.1.3 Teknik kompresi data

Teknik kompresi data dapat digolongkan menjadi dua kelompok utama, yaitu: Lossy dan Lossless. Teknik kompresi secara lossy dimaksudkan dengan teknik kompresi data dengan menghilangkan ketelitian data utama guna mendapatkan data sekecil mungkin (kompresi data sebesar mungkin). Teknik kompresi secara lossless yaitu teknik kompresi data dengan mengurangkan jumlah data yang terjadi duplikasi (memiliki simbol yang sama) sebelum terjadi kompresi.

a. Lossy Compression

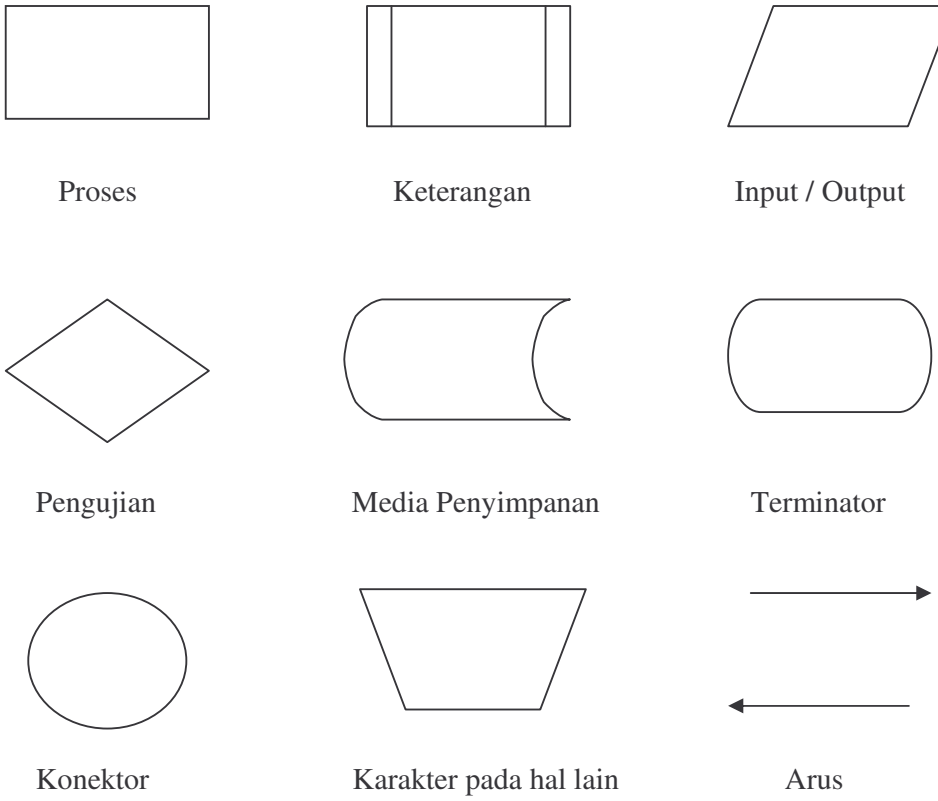
Tekinik kompresi dengan menghilangkan ketelitian dari data utama. Lossy Compression membuktikan lebih efektif diaplikasikan untuk kompresi gambar dan suara relatif besar ukurannya bila disimpan dalam computer, dan yang telah dikompresi akan tetap memiliki kemiripan walau sebagian isi dari data yang telah dihilangkan.

b. Lossless Compression.

Merupakan teknik kompresi untuk menghasilkan duplikasi dan inputan data dengan panjang bit yang lebih pendek. Tipe pemampatan ini digunakan untuk menyimpan record pada file pengolahan data. Contoh: Algoritma DMC

2.2 Flowchart

Flowchart adalah suatu alat yang cukup efisien untuk menggambarkan suatu algoritma agar dapat dengan mudah diimplementasikan pada suatu aplikasi. Tujuan utama penggunaan flowchart adalah untuk menggambarkan suatu hubungan penyelesaian masalah secara sederhan. Metode ini menggunakan simbol-simbol tertentu yang mudah dimengerti dan ditelaah distandarisasi dalam menyelesaikan masalah yang dihadapi. Adapun simbol-simbol program flowchart dapat dilihat pada gambar dibawah ini.



Gambar 2.2 Simbol-simbol Flowchart

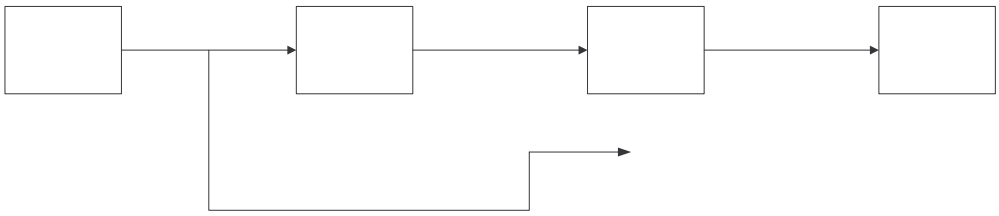
2.3 Encoding dan Decoding

Coding merupakan teknik untuk mendapatkan kode-kode tertentu (encoder), dari kode-kode tersebut dapat diaplikasikan untuk pemampatan data dan keamanan data. Dari data-data yang telah dikodekan tersebut, format-format isi dari dari data tersebut berbentuk kode-kode yang tidak bisa kit abaca. Agar kode tersebut bisa dibaca maka kita kodekan ulang data tersebut, hal ini dikenal decoding (decoder).

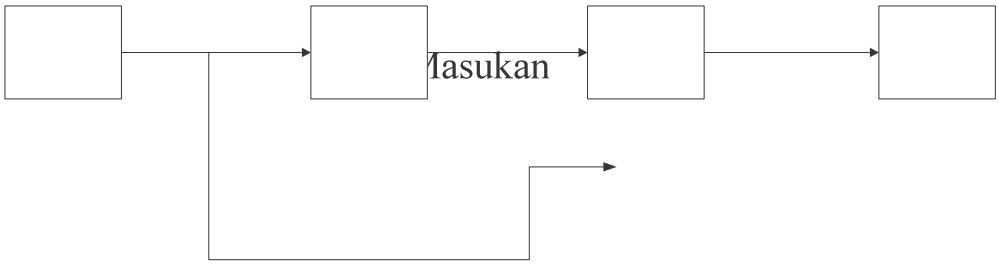
Secara umum pemampatan data merupakan teknik merubah suatu symbol-simbol menjadi suatu kode-kode. Pemampatan dikatakan efektif jika ukuran perolehan kode-kode tersebut sangat kecil dibandingkan dengan ukuran simbol aslinya. Dari suatu kode-kode atau simbol-simbol dasar suatu model akan dinyatakan dalam kode khusus.

Secara model sederhana suatu kumpulan data atau aturan-aturan untuk memproses masalah suatu simbol-simbol untuk menentukan suatu kode-kode sebagai hasil keluaran. Metode yang dimaksudkan ini adalah metode yang digunakan dengan menggunakan DMC.

Sedangkan proses dekompresinya sangat sederhana, yaitu mengeluarkan (output) run yang berulang dengan panjangnya ditambah satu. Proses encoding ditunjukkan pada gambar dibawah ini, sedangkan untuk proses decoding ditunjukkan pada gambar selanjutunya.



Gambar 2.1 Proses Encoding

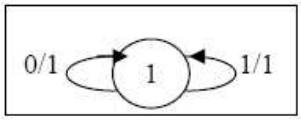


Gambar 2.3 Proses Decoding

2.4 Algoritma Dynamic Marcov Compression

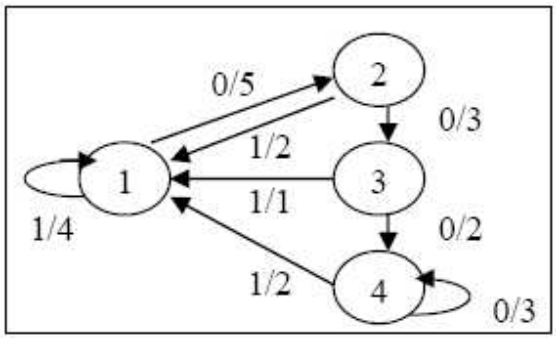
Algoritma DMC merupakan teknik pemodelan yang didasarkan pada model *finite-state* (model Markov). DMC membuat probabilitas dari karakter biner berikutnya dengan menggunakan pemodelan *finite-state*, dengan model awal berupa mesin FSA dengan transisi 0/1 dan 1/1, seperti ditunjukkan pada Gambar 2.1. DMC merupakan teknik kompresi yang adaptif, karena struktur mesin *finite-state* berubah seiring dengan pemrosesan file. Kemampuan kompresinya

tergolong amat baik, meskipun waktu komputasi yang dibutuhkan lebih besar dibandingkan metode lain



Gambar 2.4 Model awal DMC

Pada DMC, simbol alfabet input diproses per bit, bukan per byte. Setiap output transisi menandakan berapa banyak simbol tersebut muncul. Penghitungan tersebut dipakai untuk memperkirakan probabilitas dari transisi. Contoh: pada Gambar 2.4, transisi yang keluar dari *state* 1 diberi label 0/5, artinya bit 0 di state 1 terjadi sebanyak 5 kali.



Gambar 2.5 Sebuah model yang diciptakan oleh metode DMC

Secara umum, transisi ditandai dengan 0/p atau 1/q dimana p dan q menunjukkan jumlah transisi dari *state* dengan input 0 atau 1. Nilai probabilitas bahwa input selanjutnya bernilai 0 adalah $p/(p+q)$ dan input selanjutnya bernilai 1 adalah $q/(p+q)$. Lalu bila bit sesudahnya ternyata bernilai 0, jumlah bit 0 di transisi sekarang ditambah satu menjadi $p+1$. Begitu pula bila bit sesudahnya

ternyata bernilai 1, jumlah bit 1 di transisi sekarang ditambah satu menjadi q+1. Algoritma kompresi DMC diberikan pada Gambar 2.6. Masalah tidak terdapatnya kemunculan suatu bit pada *state* dapat diatasi dengan menginisialisasi model awal *state* dengan satu. Probabilitas dihitung menggunakan frekuensi relatif dari dua transisi yang keluar dari *state* yang baru. Berikut ini adalah

1. $s \leftarrow 1$ /* jumlah state sekarang */
2. $t \leftarrow 1$ /* state sekarang */
3. $T[1][0] = T[1][1] \leftarrow 1$ /* model inisialisasi */
4. $C[1][0] = C[1][1] \leftarrow 1$ /* inisialisasi untuk menghindari masalah frekuensi nol */
5. Untuk setiap input bit e :
   - i. $u \leftarrow t$
   - ii. $t \leftarrow T[u][e]$ /*ikuti transisi*/
   - iii. Kodekan e dengan probabilitas : $C[u][e] / (C[u][0] + C[u][1])$
   - iv. $C[u][e] \leftarrow C[u][e]+1$
   - v. Jika ambang batas *cloning* tercapai, maka :
     - a. $s \leftarrow s + 1$ /* state baru t' */
     - b. $T[u][e] \leftarrow s$; $T[s][0] \leftarrow T[t][0]$; $T[s][1] \leftarrow T[t][1]$
     - c. Pindahkan beberapa dari $C[t]$ ke $C[s]$

Gambar 2.6 Algoritma kompresi DMC

1. $s \leftarrow 1$ /* periksa jumlah state yang ada*/
2. $t \leftarrow 1$ /* state sekarang */
3. $T[1][0] = T[1][1] \leftarrow 1$ /* model inisialisasi */

/* Menentukan nilai kode transisi dengan nilai 0 dan 1 */

4. - Jika kode lebih besar sama dengan ambang batas cloning , maka:

   $T[u][e] \leftarrow s$; $T[s][1] \leftarrow T[t][1]$

   - Jika kode lebih kecil dari ambang batas cloning , maka:

   $T[u][e] \leftarrow s$; $T[s][0] \leftarrow T[t][0]$

   $s \leftarrow s + 1$ /* state baru *t*' */

   Pindahkan beberapa dari $C[s]$ ke $C[t]$

5. lakukan proses pembaharuan nilai bit:

   a. $u \leftarrow t$

   b. $t \leftarrow T[u][e]$ /*ikuti transisi*/

   /* perhitungan nilai probabilitas*/ jika nilainya dibawah nilai transisinya:

   c. $C[u][e] / (C[u][0] + C[u][1])$

   Jika tidak maka:

   d. $C[u][e] \leftarrow C[u][e]+1$

Gambar 2.7 Algoritma dekompresi DMC

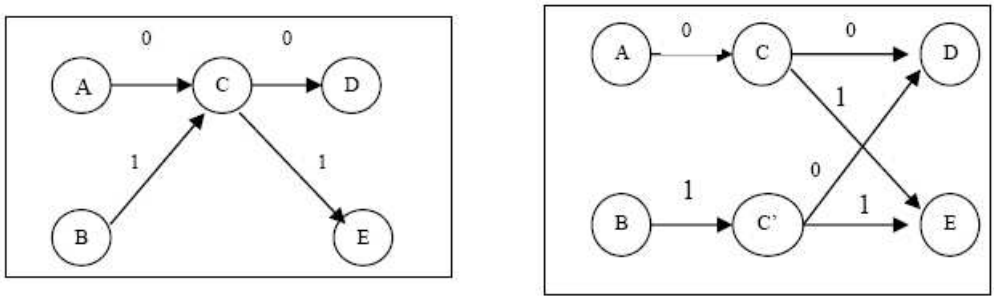
Jika frekuensi transisi dari suatu *state t* ke *state* sebelumnya, yaitu *state u*, sangat tinggi, maka *state t* dapat di-*cloning*. Ambang batas nilai *cloning* harus disetujui oleh *encoder* dan *decoder*. State yang di-*cloning* diberi simbol *t*' (lihat Gambar 2.4). Aturan *cloning* adalah sebagai berikut :

a. Semua transisi dari *state u* dikirim ke *state t*'. Semua transisi dari *state* lain ke *state* t tidak berubah.

b. Jumlah transisi yang keluar dari *t*' harus mempunyai rasio yang sama (antara 0 dan 1) dengan jumlah transisi yang keluar dari *t*.

c. Jumlah transisi yang keluar dari *t* dan *t*' diatur supaya mempunyai nilai yang sama

d. dengan jumlah transisi yang masuk.



a. Sebelum Cloning b. Setelah Cloning

Gambar 2.8 State yang di-*cloning*

Pada pembuatan program aplikasi kompresi data ini nilai ambang batas atau nilai maksimum frekuensi transisi untuk melakukan proses cloningnya adalah 8. Nilai inilah yang nantinya akan digunakan pada kedua proses DMC yaitu kompres dan dekompres. Nilai ini bebas ditentukan berapa saja asal nilai tersebut telah menjadi kesepakatan kedua algoritma baik algoritma kompres maupun algoritma dekompresnya. Jika frekuensi transisi melebihi nilai ambang batas tersebut maka secara otomatis sistem akan melakukan cloning state dengan membentuk state yang baru.

Metode DMC yang diterapkan dalam penelitian ini bertipe dinamik, dimana hanya dilakukan satu kali pembacaan terhadap file input. Kalkulasi dilakukan secara *on the fly* (proses perhitungan probabilitas dilakukan bersamaan dengan pengkodean data).

2.5 Algoritma Huffman

Algoritma Huffman, yang dibuat oleh seorang mahasiswa MIT bernama David Huffman, merupakan salah satu metode paling lama dan paling terkenal

dalam kompresi teks. Algoritma Huffman menggunakan prinsip pengkodean yang mirip dengan kode *Morse*, yaitu tiap karakter (simbol) dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang.

1. Pass pertama

   Baca (*scan*) file input dari awal hingga akhir untuk menghitung frekuensi kemunculan tiap karakter dalam file. *n* ← jumlah semua karakter dalam file input. *T* ← daftar semua karakter dan nilai peluang kemunculannya dalam file input. Tiap karakter menjadi *node* daun pada pohon Huffman.
2. Pass kedua

   Ulangi sebanyak (*n* -1) kali :

   a. Item *m1* dan *m2* ← dua *subset* dalam *T* dengan nilai peluang yang terkecil.
   b. Gantikan *m1* dan *m2* dengan sebuah item {*m1*,*m2*} dalam *T*, dimana nilai peluang dari item yang baru ini adalah penjumlahan dari nilai peluang *m1* dan *m2*.
   c. Buat *node* baru {*m*1, *m*2} sebagai *father node* dari node *m*1 dan *m*2 dalam pohon Huffman.
3. *T* sekarang tinggal berisi satu item, dan item ini sekaligus menjadi node akar pohon Huffman. Panjang kode untuk suatu simbol adalah jumlah berapa kali simbol tersebut bergabung dengan item lain dalam *T*.

Gambar 2.9 Algoritma kompresi Huffman

Sebagai contoh, dalam kode ASCII *string* 7 huruf “ABACCDA” membutuhkan representasi 7 × 8 bit = 56 bit (7 byte), dengan rincian sebagai berikut:



Gambar 2.10 kode ASCII

Untuk mengurangi jumlah bit yang dibutuhkan, panjang kode untuk tiap karakter dapat dipersingkat, terutama untuk karakter yang frekuensi kemunculannya besar. Pada *string* di atas, frekuensi kemunculan A = 3, B = 1, C = 2, dan D = 1, sehingga dengan menggunakan algoritma di atas diperoleh kode Huffman seperti pada Tabel 2.1.

Tabel 2.1 Kode Huffman untuk "ABACCDA"

Simbol	Frekuensi	Peluang	Kode Huffman
A	3	3/7	0
B	1	1/7	110
C	2	2/7	10
D	1	1/7	111

Dengan menggunakan kode Huffman ini, *string* "ABACCDA" direpresentasikan menjadi rangkaian bit : 0 110 0 10 10 111 0. Jadi, jumlah bit yang dibutuhkan hanya 13 bit. Dari Tabel 2.1 tampak bahwa kode untuk sebuah simbol/karakter tidak boleh menjadi awalan dari kode simbol yang lain guna menghindari keraguan (ambiguitas) dalam proses dekompresi atau *decoding*. Karena tiap kode Huffman yang dihasilkan unik, maka proses dekompresi dapat dilakukan dengan mudah. Contoh: saat membaca kode bit pertama dalam rangkaian bit "011001010110", yaitu bit "0", dapat langsung disimpulkan bahwa kode bit "0" merupakan pemetaan dari simbol "A". Kemudian baca kode bit selanjutnya, yaitu bit "1". Tidak ada kode Huffman "1", lalu baca kode bit selanjutnya, sehingga menjadi "11". Tidak ada juga kode Huffman "11", lalu baca

lagi kode bit berikutnya, sehingga menjadi “110”. Rangkaian kode bit “110” adalah pemetaan dari simbol “B”.

2.6 Citra digital

Citra alami (analog) merupakan suatu fungsi kontinu dari intensitas cahaya dalam bidang dua dimensi dimana x dan y menyatakan koordinat citra dan *f(x,y)* menyatakan intensitas atau informasi warna citra pada koordinat tersebut. Secara matematis dapat dinyatakan sebagai berikut:

$$f(x, y)$$

x,y menyatakan suatu koordinat bidang dua dimensi dan *f(x,y)* menyatakan intensitas atau tingkat kecerahan pada titik koordinat tersebut *(x,y).*

Seperti telah diketahui bahwa citra merupakan suatu fungsi intensitas dalam bidang dua dimensi, dimana intensitas berasal dari suatu sumber cahaya dan cahaya adalah suatu bentuk energi, maka berlaku keadaan dimana fungsi intensitas terletak diantara:

$$0 \leq f(x, y) <\sim$$

Nilai *f(x,y)* sebenarnya adalah hasil kali dari:

1. *i(x,y)* = jumlah cahaya yang bersal dari sumbernya (*illumination*), nilainya antara 0 samapai tidak berhigga, dan
2. *r(x,y)* = derajat kemampuan objek memantulkan cahaya (*reflection*), nilainya antara 0 dan 1.

Hakekatnya citra yang dilihat oleh mata kita terdiri atas berkas-berkas cahaya yang dipantulkan oleh benda-benda disekitar kita. Secara alamiah,

f(x,y) merupakan fungsi dari sumber cahaya yang menerangi objek, *i(x,y)* dan jumlah cahaya yang dipantulkan oleh objek, *r(x,y).* Dengan demikian *f(x,y) dapat* dinyatakan sebagai:

$$f(x, y) = i(x, y) \,.\, r(x, y)$$

dimana:

$0 < i(x,y) < \sim$ (iluminasi sumber cahaya)

$0 < r(x,y) < 1$ (koefisien pantul objek)

Intensitas f dari gamar hitam putih pada suatu titik (x,y) disebut skala keabuan (*grey level*), yang dalam hal ini derajat keabuannya bergerak dari hitam ke putih, sedangkan citranya disebut citra hitam-putih (*greyscale image*) atau citra monokrom.

Derajat Keabuan memiliki rentang nilai dari l_{min} sampai l_{max} atau:

$$l_{min} < f < l_{max}$$

Dengan demikian:

$$l_{min} = i_{min} - r_{min}$$

$$l_{min} = i_{max} - r_{max}$$

Kualitas citra digital dapat diukur dari resolusi spasial maupun kecerahannya. Resolusi spasial berhubungan dengan banyaknya *pixel* yang dimiliki oleh citra digital. Resolusi kecerahan berhubungan dengan seberapa baik kecerahan suatu *pixel* dari citra digital dapat merepresentasikan intensitas cahaya dari citra analognya.

Secara umum, citra terbagi menjadi dua jenis, yaitu:

a. Citra *greyscale*

Citra *greyscale* merupakan fungsi dua dimensi dari intensitas cahaya f(x,y) dengan x,y yang merupakan koordinat spasial dan nilai f merupakan kecerahan (level keabuan atau *gray level*) dari citra. Pada citra *greyscale* memiliki 256 derajat keabuan dimana nilai intensitas terendah 0 yang merepresentasikan warna hitam sampai dengan 255 yang merepresentasikan warna putih. Citra *greyscale* digital memiliki jumlah bit sebanyak 8 bit tiap *pixelnya*.

b. Citra berwarna (*multispektral*)

Citra *multispektral* merupakan citra dengan banyak komponen warna atau multi kanal dan komponen warna yang terdapat didalamnya dinyatakan sesuai dengan model warnanya. Citra warna memiliki dua macam skema warna secara fundamental, yaitu skema warna *RGB* dan *CMYK*.

Skema warna RGB merupakan skema warna berdasarkan cahaya yang diterima oleh mata manusia dan digunakan pada perangkat yang terdiri bahan fosfor karena bahan tersebut akan mengeluarkan cahaya RGB (merah, hijau, biru). Pada citra berwarna, dibentuk oleh *pixel-pixel* yang memiliki tiga komponen skema warna RGB yaitu *Red* (merah), *Green* (hijau), dan *Blue* (biru). Skema warna menunjukkan aturan pencampuran warna pada citra. Secara umum citra berwarna digital pada tiap *pixelnya* direpresentasikan dalam 24 bit dimana tiap komponen warna (RGB) terdiri dari 8 bit. Menampilkan warna tertentu dalam model ini adalah dengan menentukan

komposisi masing-masing plane tersebut sehingga skema tersebut disebut *skema warna additive.*

Skema warna CMYK (Cyan Magenta Yellow Black) digunakan pada bahan yang dicetak dengan menggunakan tinta. Proses kerja skema ini akibat tinta dari warna primer akan menyerap warna tertentu dan warna putih yang ditujukan ke kertas cetakan, sehingga disebut dengan *skema subtractive.*

2.7 Digitalisasi citra

Agar dapat diolah dengan komputer digital, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai *diskrit*. Representasi citra dari fungsi malar (kontinu) menjadi nilai-nilai *diskrit* disebut *digitalisasi*. Citra yang dihasilkan inilah yang disebut citra digital (*digital image*). Pada umumnya citra digital berbentuk empat persegipanjang dan dimensi ukurannya dinyatakan sebagai tinggi x lebar (atau panjang x lebar).

Citra digital yang tingginya *N,* lebarnya *M,* dan memiliki *L* derajat keabuan dapat dianggap sebagai fungsi:

$$f(x,y)\begin{cases}0 \le x \le M\\ 0 \le y \le N\\ 0 \le f \le L\end{cases}$$

Citra digital yang berukuran N x M lazim dinyatakan dengan matriks yang berukuran *N* baris dan *M* kolom sebagai berikut:

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M) \\ f(1,0) & f(1,1) & \dots & f(1,M) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Indeks baris (*i*) dan indeks kolom (*j*) menyatakan suatu koordinat titik pada citra, sedangkan *f(i,j)* merupakan intensitas (derajat keabuan) pada titik (*i,j*). Masing-masing elemen pada citra digital (berarti elemen matriks) yang menunjukkan intensitas citra di suatu titik disebut *pixel* (*picture element*). Jadi, citra yang berukuran *N x M* mempunyai *NM* buah *pixel*.

Proses digitalisasi citra dibagi menjadi dua macam:

1. Digitalisasi spasial (*x,y*), sering disebut sebagai penerokan (*sampling*). Untuk memudahkan implementasi, jumlah *sampling* biasanya diasumsikan perpangkatan dua,

   $N = 2^n$

   Dimana:

   N = jumlah *sampling* pada suatu baris/kolom

   n = bilangan bulat positif

   Pembagian gambar menjadi ukuran tertentu menentukan resolusi spasial yang diperoleh. Semakin tinggi resolusinya, berarti semakin kecil ukuran *pixel* (atau semakin banyak jumlah *pixel*-nya) semakin halus gambar yang diperoleh karena informasi yang hilang akibat pengelompokkan derajat keabuan pada *sampling* semakin kecil.

2. Digitalisasi intensitas *f(x,y)*, sering disebut *kuantisasi.*

   Langkah setelah proses *sampling* adalah *kuantisasi*. Proses ini membagi skala keabuan (0, *L*) menjadi *G* buah level yang dinyatakan dengan suatu harga bilangan bulat (*integer*), biasanya *G* diambil perpangkatan dari 2,

$G = 2^m$

Dimana:

G = derajat keabuan

m = bilangan bulat positif

Penyimpanan citra digital yang *disampling* menjadi *N x M* buah *pixel* dan dikuantisasi menjadi $G = 2^m$ level derajat keabuan membutuhkan memori sebanyak:

$b = N \times N \times m$

Contoh, untuk menyimpan suatu citra berukuran 512 x 512 pixel dengan 256 derajat keabuan membutuhkan memori sebesar: 512 x 512 x 8 bit = 2048.000 bit.

Secara keseluruhan, resolusi gambar ditentukan oleh *N* dan *m.* Makin tinggi nilai *N (*atau *M)* dan *m*, maka citra yang dihasilkan semakin bagus kualitasnya. Untuk citra dengan jumlah objek yang sedikit, kualitas citra ditentukan oleh nilai *m.* Sedangkan untuk citra dengan jumlah objek yang banyak, kualitasnya ditentukan oleh *N (*atau *M).*

2.8 Operasi citra

Operasi-operasi yang dilakukan dalam pengolahan citra banyak macamnya. Namun secara umum, operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut:

a. Perbaikan kualitas citra (*image enhancemen*)

   Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat didalam citra lebih ditonjolkan.

b. Pemugaran citra (*image restoration*)

   Operasi bertujuan untuk menghilangkan atau meminimumkan cacat pada citra.

c. Pemampatan citra (*image compression*)

   Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit.

d. Segmentasi citra (*image segmentation*)

   Jenis operasi ini bertujuan untuk memecah citra kedalam beberapa segmen dengan suatu kriteria tertentu.

e. Analisis citra (*image analysis*)

   Operasi ini bertujuan untuk menghitung besar kuantitatif dari citra untuk menghasilkan deskripsinya.

f. Rekontruksi citra (*image reconstruction*)

   Operasi ini bertujuan untuk membentuk ulang objek dari beberapa objek dan beberapa citra hasil proyeksi. Operasi rekontruksi citra banyak digunakan dalam bidang medis.

2.9 Elemen citra digital

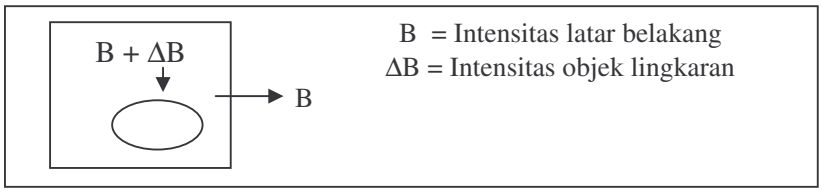
Dalam pemrosesan citra digital, diperlukan elemen-elemen sebagai berikut:

a) Kecerahan (*Brightness*)

Kecerahan yaitu intensistas yang terjadi pada satu titik citra. Lazimnya pada citra, kecerahan ini merupakan kecerahan rata-rata dari suatu daerah lokal. Sistem visual daerah manusia mempu menyesuaikan dirinya dengan tingkat kecerahan dengan jangkauan dari yang terendah sampai yang tertinggi. Batas penyesuaian gelap (terendah) disebut *Scotopic threshold*, sedangkan batas penyesuaian terang (tinggi) disebut dengan *Glare threshold.* Sebagai contoh *Scotopic threshold* yaitu terasanya pada mata kita setelah lampu dipadamkan (terang ke gelap). Sedangkan pada *Scotopic threshold* yaitu terasanya mata kita setelah keluar dari gedung bioskop (gelap ke terang).

b) Kontras (*Contras*)

Kontras menyatakan sebaran terang dan gelap didalam sebuah citra. Citra dengan kontras rendah dicirikan oleh sebagian komposisi citranya adalah terang dan sebagian besar gelap. Pada citra dengan kontras yang baik komposisi gelap dan terang tersebar secara merata. Untuk menentukan kepekaan kontras (*contrast sensitivity*) pada mata manusia, dilakukan cara pengukuran sebagai berikut:



Gambar 2.11 Pengukuran Kepekaan Kontras

Keterangan:

Suatu bidang citra dengan intensitas sebesar B, kita perbesar intensitas objek lingkaran sehingga intensitasnya menjadi B + ΔB. Pertambahan intensitas (ΔB) ini dilakukan sampai mata manusia dapat mendeteksi perbedaan ini.

c) Acuity

Kemampuan mata manusia untuk merinci secara detail bagian-bagian pada suatu citra (pada sumbu visual).

d) Kontur

Suatu keadaan pada citra dimana terjadi perubahan intensitas dari suatu titik ke titik tetangganya. Dengan perubahan intensitas ini mata manusia sanggup mendeteksi pinggiran atau kontur suatu benda.

e) Warna (*Color*)

Warna adalah reaksi yang dirasakan oleh sistem visual mata manusia terhadap perubahan panjang gelombang (λ) cahaya. Setiap warna memiliki panjang gelombang masing-masing. Wana *merah* memeliki panjang gelombang paling tinggi, sedangkan warna *ungu* (*violet*) mempunyai panjang gelombang paling rendah. Warna-warna yang diterima oleh mata merupakan hasil kombinasi cahaya dengan panjang gelombang yang

berbeda. Penelitian memperlihatkan bahwa kombinasi warna yang memberikan rentang warna yang paling lebar adalah *Red* (R), *Green* (G), *Blue* (B).

f) Bentuk (*Shape*)

Pada umumnya citra yang dibentuk oleh mata merupakan citra dua dimensi (2D), sedangkan objek yang diamati biasanya adalah tiga dimensi (3D). Kesulitannya, banyak benda tiga dimensi setelah diproyeksikan kebidang dua dimensi terlihat sama. Misalnya suatu ruangan terlihat berbentuk trapesium pada bidang citra dua dimensi. Dalam hal ini kita tidak tahu apakah hal ini memang disebabkan oleh bentuk ruangan yang panjang ataukah memang ruangan tersebut berbentuk trapesium.

g) Tekstur

Pada hakekatnya, sistem visual manusia tidak menerima informasi citra secara terpisah pada setiap titik, tetapi suatu citra dianggapnya sebagai satu kesatuan. Jadi definisi kesamaan suatu objek perlu dinyatakan dalam bentuk kesamaan dari suatu himpunan parameter citra. Atau dengan kata lain dua buah citra tidak dapat disamakan hanya dengan satu parameter saja.

h) Waktu dan Pergerakan

Respon suatu sistem visual tidak hanya berlaku pada faktor ruang tetapi juga pada faktor waktu. Sebagai contoh, bila citra statis ditampilakan bergantian secara cepat. Maka kita akan mendapatkan kesan melihat citra yang bergerak.

i) Deteksi dan Pengenalan

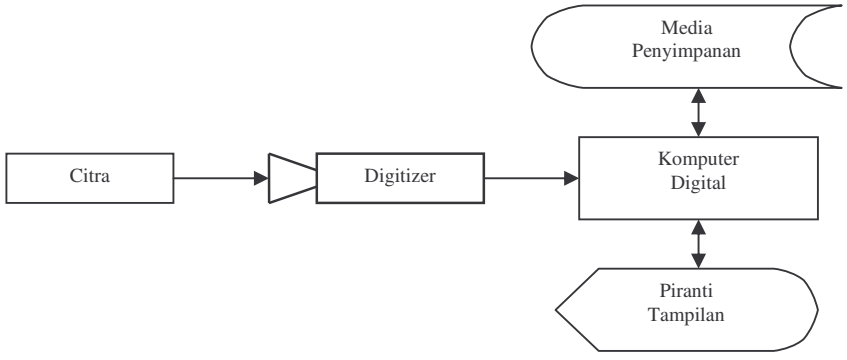
Dalam mendeteksi aerta mengenali suatu citra, sering tidak hanya sistem visual yang bekerja, tetapi juga seluruh ingatan yang kita miliki. Contohnya apabila kita melihat seorang wanita dari samping, kita mungkin menafsirkannya sebagai wanita muda. Tetapi setelah melihat wajahnya secara keseluruhan, sebenarnya kita terkecoh dan faktanya menunjukkan sebaliknya.

2.10 Elemen pemrosesan citra digital

Secara umum, elemen yang terlibat dalam pemrosesan citra dapat dibagi menjadi empat komponen, yaitu:

a) *Digitizer*,
b) Komputer digital,
c) Piranti tampilan,
d) Piranti penyimpanan.

Keempat komponen tersebut, dapat ditunjukkan pada gambar berikut:



Gambar 2.12 Elemen Sistem Pemrosesan Citra

Digitizer (*digital image acquisition system*) merupakan sistem penangkap citra digital yang melakukan penjelajahan citra dan mngkonversinya kerepresentasi numerik sebagai masukan bagi komputer digital. Hasil dari digitalizer adalah matrik yang elemen-elemennya menyatakan nilai intesitas cahaya pada satu titik. Contoh digitalizer adalah kamera digital dan scanner. Digitizer terdiri atas tiga komponen dasar, yaitu *sensor citra* yang bekerja sebagai pengukur intensitas cahaya, perangkat penjelajah yang berfungsi merekam hasil pengukuran intensitas pada seluruh bagian citra, dengan pengubah analog ke digital yang berfungsi melakukan pencuplikan dan kuantisasi.

Komputer digital digunakan pada sistem pemrosesan citra dapat bervariasi, dari komputer mikro sampai komputer besar yang mampu melakukan bermacam-macam fungsi pada citra digital resolusi tinggi.

Piranti tampilan peraga berfungsi mengkonversi matriks intensitas yang mempresentasikan citra ke tampilan, yang dapat di interpretasikan oleh mata manusia. Contoh piranti tampilan adalah monitor dan printer.

Media penyimpanan adalah piranti yang mempunyai kapasitas memori besar sehingga citra dapat disimpan secara permanen agar dapat diproses lagi pada waktu lain.

2.11 Kriteria penilaian kualitas citra digital

Pada bagian ini dibahas mengenai criteria-kriteria penilaian kualitas baik buruknya suatu citra dan kriteria penilaian dibagi menjadi dua yaitu kriteria penilaian objektif dan kriteria penilaian subjektif.

1. Kriteria penilaian objektif

Kriteria penilaian Objektif ini didasarkan pada batas error yang diperbolehkan untuk citra yang akan diolah. Untuk citra *f(x,y)* dan citra hasil proses *g(x,y),* maka beberapa parameter yang dijadikan sebagai kriteria penilaian objektif adalah sebagai berikut :

a) Peak Signal to Noise Ratio (PSNR), kita dapat membuat ukuran kualitas hasil pemampatan citra menjadi ukuran kuantitatif dengan menggunakan PSNR. PSNR dihitung untuk mengukur perbedaan antara citra semula dengan citra hasil pemampatan dengan citra semula, dengan rumus:

$$PSNR = 10\log(255/MSE)^2$$

b) Mean Square Error (MSE) :

$$MSE = \frac{1}{MN}\sum_{X=0}^{M=X}\sum_{Y=0}^{N=Y}(citra_hasil(x,y) - citra_asli(x,y))^2$$

c) Mean Absoluted Error (MAE) :

$$MAE = \frac{1}{MN}\sum_{X=0}^{M=X}\sum_{Y=0}^{N=Y}\left|citra_hasil(x,y) - citra_asli(x,y)\right|$$

d) Menghitung rasio kompresi (R) dn Persentasi P :

R=[(((KCA - KCH) / KCA)*100)]

Keterangan:

R = Nilai rasio.

KCA = Kapasitas Citra Asli

KCH = Kapasitas Citra Hasil

e) Number of Region (NR) :

Yaitu jumlah region atau segmen hasilsegmentasi.

f) Time (t) :

Yaitu lama proses yang dibutuhkan dalam proses segmentasi.

2. Kriteria penilaian subjektif

Kriteria ini ditentukan berdasarkan hasil pengamatan oleh mata manusia. Dengan menggunakan kriteria ini, baik buruknya citra hasil pengolahan ditentukan oleh pengamat sendiri. Sebagai contoh, dua buah citra yang memiliki salah satu yang sama pada kriteria penilaian kualitas citra secara objektif dapat mempunyai kualitas subjektif yang berbeda, tergantung dari persepsi visual pengamat. Beberapa kriteria hasil penilaian subjektif yang banyak digunakan adalah sebagai berikut:

a) Excellent (skor 9 atau 10)

Citra hasil segmentasi yang diamati mempunyai kualitas sangat baik, menggambarkan *region* citra dengan tepat atau mendekati tepat.

b) Fine (skor 7 atau 8)

Citra hasil segmentasi yang diamati masih mempunyai kualitas tinggi, menggambarkan *region* citra dengan sedikit gangguan-gangguan atau kesalahan.

c) Passable (skor 5 atau 6)

Citra hasil segmentasi yang diamati mempunyai kualitas agak baik, menggambarkan *region* citra dengan gangguan-gangguan atau kesalahan atau kesalahan yang sedikit berarti.

d) Marginal (skor 3 atau 4)

Citra hasil segmentasi yang diamati mempunyai kualitas buruk, menggambarkan *region* citra dengan gangguan yang cukup besar.

e) Inferior (skor 1 dan 2)

Citra hasil segmentasi yang diamati mempunyai kualitas yang sangat buruk, tetapi *region* citra masih dapat diamati secara kasar dengan gangguan-gangguan yang sama jelas atau sangat besar.

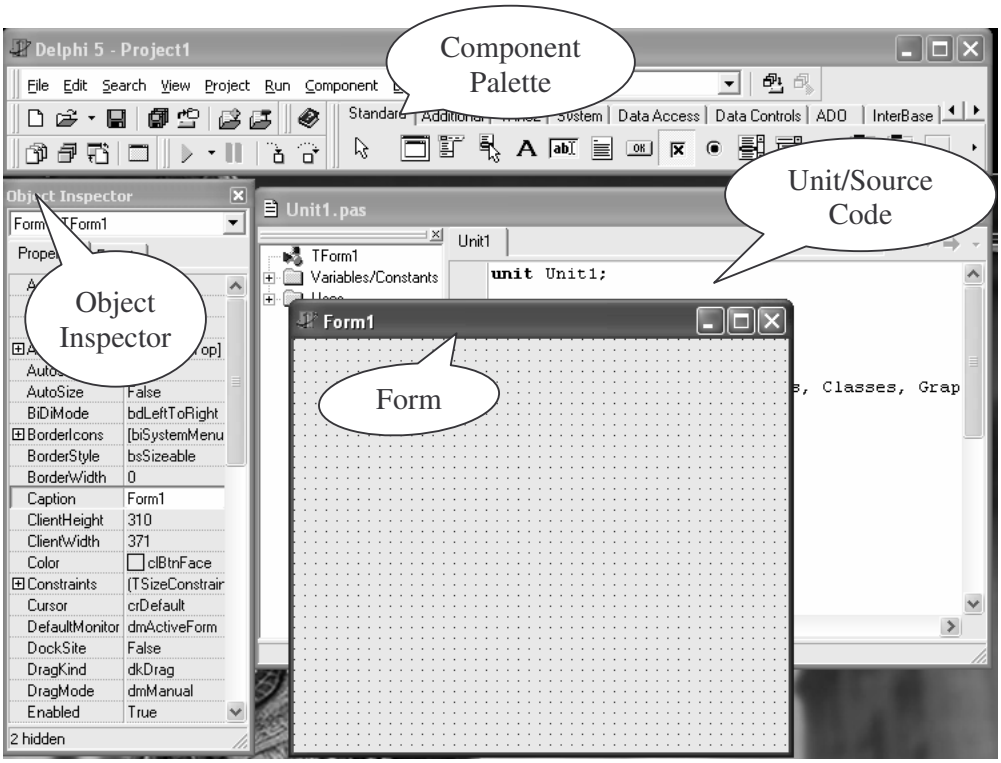
f) Unusable (skor 0)

Citra hasil segmentasi yang diamati sangat buruk sudah tidak dapat diamati lagi.

2.12 Program simulasi yang digunakan

Program yang digunakan untuk mensimulasikan masalah ini adalah Borland Delphi 6.0. Delphi adalah bahasa pemograman visual dalam lingkungan windows. (*under windows*) yang menggunakan bahasa pascal. Keberadaan bahasa pemrograman Delphi tidak dapat dipisahkan dari bahasa Turbo Pascal karena Delphi merupakan generasi penerus dari Turbo Pascal.

Delphi telah memanfaatkan suatu teknik pemrograman yang disebut RAD yang telah membuat pemrograman lebih mudah. Delphi adalah suatu bahasa pemrograman yang telah memanfaatkan metode pemrograman Object Orientied Programming (OOP). Lingkungan kerja Borland Delphi dapat dilihat pada gambar 2.13 dibawah ini.



Gambar 2.13 Lingkungan kerja Borland Delphi

Delphi menyediakan fasilitas yang lengkap untuk membangun suatu program aplikasi, diantaranya adalah IDE (*integrated Development Environment*) dengan IDE maka kita akan sangat terbantu karena semua kebutuhan pemograman telah disediakan dalam suatu tampilan. IDE Delphi terdiri atas menu, *speedbar*, *component pallete*, *object inspector*, *form* dan *editor code*. Semua itu akan kita dapatkan pada saat pertama membuka Delphi.

Fungsi dari elemen-elemen di atas adalah:

- Object Inspector : suatu windows yang berguna untuk mengatur suatu object baik properti, event dan method.

- Form : digunakan sebagai layar / windows yang digunakan sebagai lembar kerja kita. Di form-lah semua komponen seperi tombol dan komponen lainnya disimpan.
- Window Unit / Source Code : layar yang berisi perintah-perintah yang akan dieksekusi oleh komputer. Di layar inilah kita akan mengisikan program-program.
- Component Palette : layar yang berisikan komponen-komponen yang dipakai dalam program kita.