BAB II

DASAR TEORI

2.1. Pengenalan Pola

Mengelompokan data numerik dan simbolik (*citra*) secara otomatis oleh mesin (*komputer*). Tujuan pengelompokan adalah untuk mengidentifikasi suatu objek berdasarkan citra yang dihasilkan.



Gambar 2.1 Alur Pengenalan Pola

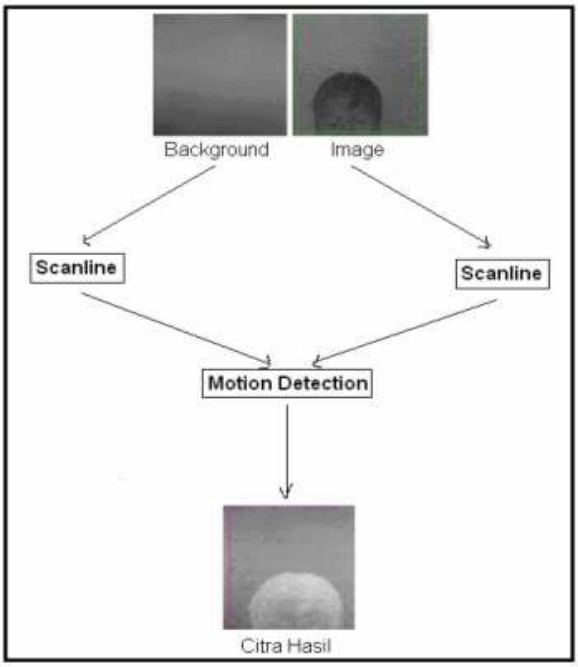
Pengenalan pola biasanya menggunakan proses atau cara tertentu, seperti dengan menggunakan segmentasi gambar tujuannya untuk memecah citra menjadi beberapa bagian, sehingga objek yang ada pada citra tersebut dapat dikenali dengan mudah, cara lainnya adalah dengan menggunakan *optical flow* tujuannya untuk membedakan antara daerah objek dengan daerah yang kosong, *optical flow* sangat terpengaruh oleh cahaya, biasanya *optical flow* ditandai oleh kotak-kotak kecil pada objek yang terdeteksi.

2.2. Motion Detection

Motion detection merupakan penelitian yang penting dalam keilmuan *computer vision*. Banyak metode *motion detection* yang telah ditemukan, satu diantaranya adalah dengan menghitung perbedaan nilai-nilai intensitas pada suatu piksel dari dua *frame* gambar yang diambil secara berturut-turut yang kemudian dilakukan proses *thresholding* untuk menentukan adanya perubahan objek atau

tidak. Sekalipun metode ini sangat sederhana dalam proses implementasinya, tapi metode ini merupakan metode dasar dari proses *motion detection*. Hanya saja metode ini kurang efektif untuk menentukan pergerakan objek secara keseluruhan, terutama bagian dalam dari objek yang bergerak, akan tetapi secara umun metode ini sudah mampu mengidentifikasi adanya perubahan objek. Pengurangan nilai-nilai intensitas setiap piksel yang ada pada *background* terhadap suatu *image* baru merupakan metode yang paling popular untuk proses *motion detection.*

Langkah – langkah dari *motion detection* secara umum adalah :



Gambar 2.2 *Motion detection*

Pada gambar 2.5 sebelah kiri atas merupakan gambar asli atau *current image* (*background*), sebelah kanan atas merupakan gambar dengan adanya objek atau *previous image*, sedangkan gambar dibawahnya merupakan hasil dari proses *motion detection*.

Langkah-langkah *motion detection* adalah :

1. Memasukkan gambar pertama sebagai *beckground*.
2. Memasukkan gambar kedua sebagai gambar yang ada objeknya.
3. Gambar pada langkah kesatu dan kedua lakukan *scanline*, fungsinya adalah untuk menemukan apakah ada perubahan objek atau tidak ada perubahan. *Scanline* dilakukan dari kiri atas gambar sampai dengan kanan bawah gambar.
4. Lakukan perbandingan antara gambar *background* dengan gambar satunya lagi, hal ini dilakukan supaya bisa mengetahui adanya perubahan atau adanya perbedaan antara gambar kesatu dan gambar kedua. Pada langkah ini biasanya ada proses *tresholding* dan deteksi pinggiran jika diperlukan. Ini juga disebut proses *motion detection*.
5. Setelah melakukan proses diatas maka akan dihasilkan citra hasil *motion detection*.

2.3. Metode Motion Detection

2.3.1. Metode Spatial Domain

Deteksi gerakan yang digunakan di aplikasi kamera pengawas pendeteksi gerakan ini adalah pemrosesan *citra spatial domain* dengan *point processing*. Metode ini bekerja dengan cara sebagai berikut :

1. Penetapan sebuah citra referensi

   Sebuah citra yang diambil oleh kamera pada saat situasi lingkungan yang stabil ditetapkan sebagai citra referensi. Citra ini kemudian disimpan sebagai acuan dalam membandingkan citra.

2. Pengambilan citra secara periodik

   Setiap jangka waktu tertentu, sebuah citra diambil oleh kamera. Citra ini kemudian disimpan untuk perhitungan selanjutnya.

3. Pendeteksian gerak

   Citra yang diambil secara periodik tadi dibandingkan dengan referensi dengan cara membandingkan setiap *pixel* yang berada di lokasi yang sama. Perhitungan perbandingan dilakukan dengan cara mengambil nilai absolute dari hasil pengurangan nilai RGB dari 2 *pixel* yang berbeda dilokasi yang sama. Jika nilai dari perhitungan tersebut melewati suatu batas nilai tertentu maka *pixel* tersebut dinyatakan sebagai *pixel* yang terdeteksi gerakan.

$$f(x,y) = \begin{cases} 1 \text{ if } T1 < ABS\,[p(x,y) - q(x,y)] \\ 0 \text{ if } T1 < ABS\,[p(x,y) - q(x,y)] \end{cases}$$

[II.2.1] Persamaan Pendeteksian Gerak Pada Metode Spatial Domain

Dimana :

f = *Array 2* dimensi yang digunakan untuk menyimpan hasil perhitungan.

x = Posisi *pixel* terhadap sumbu x.

y = Posisi *pixel* terhadap sumbu y.

T1 = Batas nilai untuk membatasi banyaknya perubahan nilai *pixel* sebelum sebuah *pixel* tersebut dinyatakan terhadap gerakan (*Threshold*).

ABS = Fungsi absolute.

p = Citra referensi.

q = Citra yang diambil secara periodic (citra yang dibandingkan).

Variabel *T1* merupakan sebuah nilai batas ambang perubahan 2 buah *pixel* di letak yang sama. Penggunaan variabel *T1* ini memiliki dua tujuan. Tujuan yang pertama adalah untuk mengeliminasi noise yang terjadi, sedangkan tujuan yang kedua adalah untuk memberikan batas ambang perubahan intensitas warna antara dua buah *pixel*. *Noise* terjadi karena suatu lingkungan yang pencahayaannya labil. Hal ini dapat terjadi karena intensitas cahaya yang datang tidak selalu sama dari waktu ke waktu.

Sebuah *array boolean f(x,y)* digunakan untuk menyimpan nilai dari perhitungan. Jika nilai *f(x,y)* bernilai *true* berarti dua buah *pixel* dari 2 buah citra yang berbeda dengan letak yang sama terjadi perubahan warna yang melewati batas ambang *T1*. Sebaliknya, jika *f(x,y)* bernilai *false* berarti kedua buah *pixel* tersebut tidak terjadi perubahan warna atau perubahan yang terjadi tidak melewati batas ambang *T1*. *Array* ini digunakan untuk menghitung jumlah *pixel* yang berubah. Jika jumlah *pixel* yang berubah melebihi nilai batas ambang *T2* maka citra tersebut dikatakan terdapat gerakan.

$$\text{res} = \begin{cases} 1 \text{ if } T2 < \sum_x \sum_y f(x,y) \\ 0 \text{ if } T2 \geq \sum_x \sum_y f(x,y) \end{cases}$$

[II.2.2] Persamaan Perhitungan Jumlah Pixel Pada Metode Spatial Domain

Dimana :

res = Hasil pendeteksi gerakan, bernilai 1 jika terdapat gerakan, dan bernilai 0 jika tidak terdapat gerakan.

T2 = Batas nilai untuk membatasi banyak jumlah *pixel* yang berubah sebelum sebuah citra dinyatakan terdapat gerakan (*Threshold*).

x = Posisi *pixel* terhadap sumbu x.

y = Posisi *pixel* terhadap sumbu y.

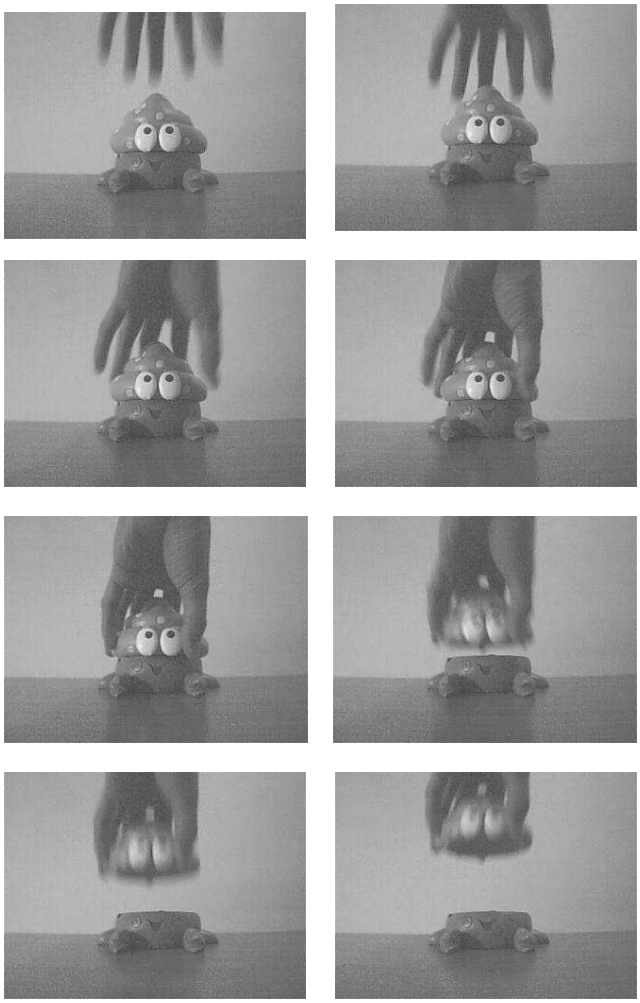
f = *Array* hasil perhitungan pendeteksian citra.

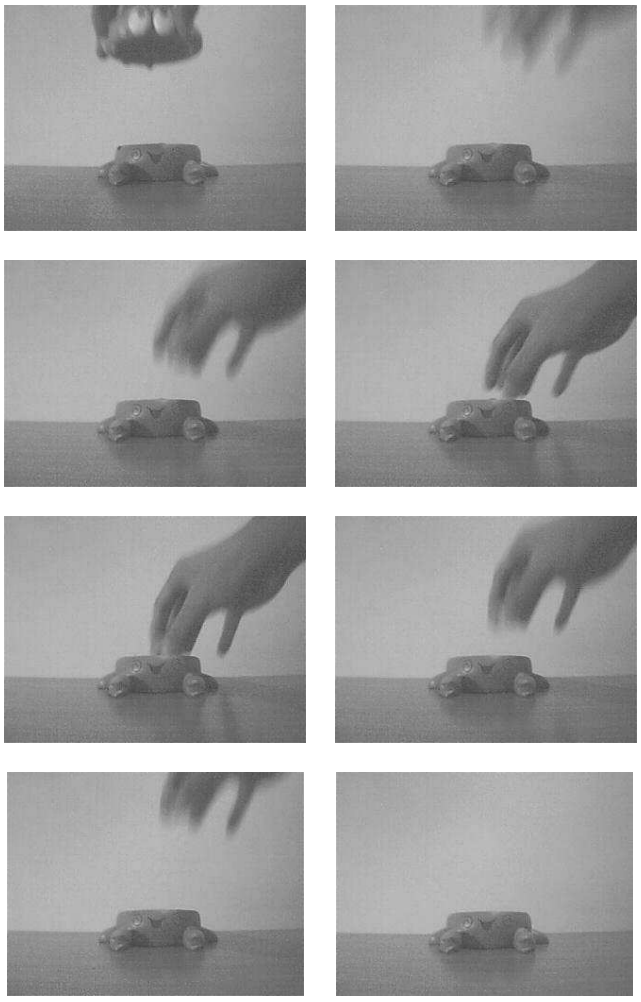
Terdapat dua metode yang dapat digunakan untuk menetapkan sebuah citra sebagai citra referensi. Metode yang pertama, citra referensi hanya diambil satu kali untuk pendeteksian seterusnya, sedangkan metode kedua, citra referensi diambil dari citra pembanding setiap kali citra pembanding telah lolos dari deteksi gerakan. Metode yang pertama digunakan apabila lingkungan yang dideteksi tidak ada perubahan cahaya selama proses deteksi gerakan berlangsung, misalnya di dalam ruangan.

Metode yang kedua digunakan apabila pencahayaan lingkungan yang dideteksi dapat berubah-ubah. Metode ini dapat digunakan pada lingkungan yang terpengaruh oleh cahaya matahari. Perubahan sinar matahari tidak dianggap sebagai gerakan karena perubahannya yang sangat lambat dan karena adanya variabel *T1*. Penggunaan metode kedua ini dapat berakibat tak terdeteksinya sebuah gerakan yang sangat lambat, metode ini hanya dapat mengenali perubahan yang ekstrim. Apabila proses pendeteksian termasuk untuk mendeteksi gerakan yang sangat lambat maka metode kedua ini tidak dapat digunakan, atau dapat

digunakan tetapi dengan mengurangi nilai batas ambang *T1*, sehingga perubahan kecil dapat terdeteksi oleh sistem.

Berikut adalah citra yang disimpan selama pengujian berlangsung. Nilai T1 yang digunakan sebesar 10 *pixel*, nilai T2 sebesar 12%, dan retang waktu pengambilan citra setiap 350ms. Terdapat 16 citra yang tersimpan akibat adanya gerakan. Setiap citra diberi keterangan mengenai jam, menit, detik dan milidetik saat citra tersebut disimpan.





Gambar 2.3 *Spatial Domain*

2.3.1.1. Algoritma Metode Spatial Domain

Langkah pertama yang dilakukan adalah melakukan perbandingan frame awal dengan frame ketika terjadi pergerakkan.

Kemudian setiap frame akan dianalisis dengan metode *Discrete Cosine Transformation* (DCT). Apabila terdeteksi suatu pergerakkan maka perubahan *pixel* akan ditandai dengan warna merah.

2.3.2. Metode Frequency Domain

Pengolahan citra digital dengan metode frequency domain, informasi citra digital ditransformasikan lebih dulu dengan transformasi Fourier, kemudian dilakukan manipulasi pada hasil transformasi Fourier tersebut. Setelah manipulasi selesai, dilakukan inverse transformasi Fourier untuk mendapatkan informasi citra kembali.

Metode frequency domain ini dapat digunakan untuk menyelesaikan masalah-masalah tertentu yang sulit jika dilakukan dengan metode spatial domain.

Transformasi Fourier dari f(x), didefinisikan sebagai berikut :

$$F(u) = \int_{-\infty}^{\infty} f(x) \exp[-j2\pi ux]dx$$

di mana

$$j = \sqrt{-1}$$

[II.2.3] Persamaan Transformasi Forier Pada Metode Frequency Domain

Sebaliknyan ,jika diketahui F(u), maka f(x) dapat diperoleh dengan Inverse Transformasi berikut :

$$f(x) = \int_{-\infty}^{\infty} F(u) \exp[j2\pi ux]du$$

[II.2.4] Persamaan Inverse Transformasi Pada Metode Frequency Domain

Kedua persamaan di atas diesbut dengan pasangan transformasi Fourier.

Jika f(x) adalah bilangan real, biasanya F(u) merupakan bilangan kompleks yang biasa diuraikan menjadi : $F(u) = R(u) + jI(u)$

[II.2.5] Persamaan Bilangan Kompleks Pada Metode Frequency Domain

dimana R(u) dan I(u) adalah komponen ral dan imajiner dari F(u).

Persamaan di atas juga sering dituliskan sebagai :

$F(u) = | F(u) | e^{j\Phi u}$

dimana

$| F(u) | = [R^2(u) + I^2(u)]^{1/2}$

$\Phi(u) = \tan^{-1}[I(u) / R(u)]$

[II.2.6] Persamaan Bilangan Kompleks Pada Metode Frequency Domain

Fungsi magnitude | F(u) | disebut juga spectrum Fourier dari f(x), dan Φ(u) disebut juga dengan sudut fase dari f(u).

Jika f(x) dijadikan diskrit maka persamaan transformasi Fourier diskrit adalah:

$f(x) = 1/N \sum f(x) \exp [-j2 \pi ux / N]$

dan

$f(x) = \sum F(u) \exp [j2 \pi ux / N]$

[II.2.7] Persamaan Transformasi Fourier Diskrit Pada Metode Frequency Domain

Karena pada pengolahan citra digital, data yang digunakan berbentuk digital/diskrit maka dapat digunakan kedua persamaan diatas untuk melakukan transformasi dan inverse transformasi Fourier.

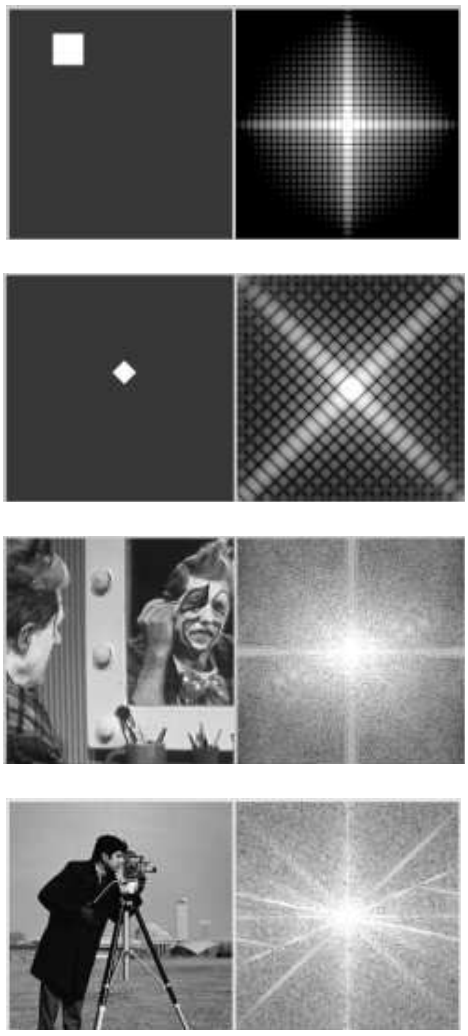
Untuk menganalisa citra pada frequency domain, hasil transformasi Fourier dapat ditampilkan sebagai citra, dimana intensitasnya proposional dengan besarnya | F(u) | atau spectrum Fourier. Namun karena *dynamic range* dari spectrum Fourier biasanya sangat besar, maka sebelum ditampilkan sebagai citra harus diubah menjadi :

$$D(u,v) = c \log (1 + | F(u,v) |)$$

[II.2.8] Persamaan Dynamic Range Pada Metode Frequency Domain

Dimana c adalah konstanta. Selanjutnya yang ditampilkan sebagai citra adalah nilai dari D(u,v). D(u,v) ini akan memiliki *dynamic range* yang lebih kecil dari pada | F(u,v) |.

Berikut adalah contoh gambarbeberapa citra dengan spectrum Fouriernya.



Gambar 2.4 *Spektrum Fourier Citra*

2.3.2.1. Algoritma Metode Frequency Domain

Langkah pertama yang dilakukan adalah membaca file citra yang akan diproses dan menyimpan informasi graylevel dari semua *pixel*nya ke dalam sebuah matriks.

Selanjutnya dilakukan transformasi Fourier dan dilanjutkan dengan penggeseran (*shifting*) hasil transformasi Fourier tersebut supaya hasil visualisasinya lebih mudah diamati dan dianalisa.

Setelah itu baru digunakan perintah *imshow* untuk menampilkan citra maupun spektrum Fouriernya. Namun pada saat spektrum Fourier akan ditampilkan, karena *dynamic range* dari spektrum Fourier sangat besar.

Proses ini akan memperkecil *dynamic range* sehingga dapat ditampilkan di layar dengan lebih jelas dan lebih mudah untuk dianalisa. Untuk menghilangkan noise, lebih dulu dibuat sebuah matriks h yang dimensinya sama dengan dimensi dari matriks untuk menyimpan hasil transformasi Fourier, yaitu 256 x 256. Matriks h ini dinisialisasi dengan diisi nilai satu pada setiap elemennya.

Selanjutnya bagian tertentu dari matriks yang menunjukkan area dimana noise berada pada spektrum Fourier citra diubah menjadi nol.

Jika noise pada spektrum Fourier citra lainnya berada di area yang berbeda, tentunya perintah di atas harus dimodifikasi angka-angkanya sehingga sesuai dengan area noise pada spektrum Fourier yang akan dihilangkan.

Setelah bagian tertentu dari matriks tersebut diberi nilai nol, sedangkan yang lainnya tetap bernilai satu, maka masing-masing elemen matriks tersebut

dikalikan dengan elemen-elemen yang bersesuaian dari matriks spektrum Fourier citra.

Perkalian ini akan menghasilkan spektrum Fourier baru dimana area noise telah hilang karena dikalikan dengan nol. Pada visualisasi spektrum Fourier hilangnya area noise ini ditunjukkan dengan warna hitam di area tersebut. Langkah berikutnya adalah melakukan inverse transformasi Fourier dari spectrum Fourier yang baru untuk mendapatkan informasi citra kembali.

Digunakannya uint8 adalah supaya informasi citra yang diperoleh memiliki tipe data *pixel* 8-bit atau 256 graylevel, sehingga sama dengan citra asalnya. Program selengkapnya yang disertai dengan penjelasan mengenai perintah-perintah yang digunakan dapat dilihat pada lampiran.

2.4. Aspek Ratio

Aspect ratio adalah perbandingan tinggi dan lebar suatu piksel. Piksel-piksel tersebut bisa berupa rangkaian titik-titik berbentuk garis, berbentuk kotak dan bentuk-bentuk lainnya. Untuk mendapatkan area x dan area y harus dikalikan dengan (yasp/xasp). Perbandingan tinggi dan lebar pada layar komputer tidak pernah 1 : 1 kecuali pada komputer IBM 8514. Karena itu jika menggambarkan bujur sangkar hanya dengan menyamakan panjang semua sisinya, yang akan tampak pada layar komputer adalah persegi panjang. Supaya sisi x tampak sama panjang dengan sisi y, maka harus mengalikannya dengan suatu vektor yang disebut *aspect ratio*. Misalkan x nilainya 5 dan y nilainya 10, maka nilai *aspect ratio* adalah 5:10 = 0.5.



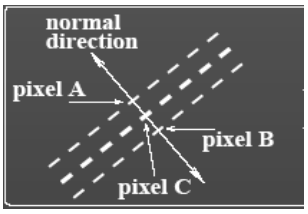
Gambar 2.5 *aspect ratio*

2.5. Pixel

Pixel atau piksel (*picture element /* unsur gambar*)* adalah tititk–titik kecil. Gambar apapun yang tampak di layar komputer sebenarnya tersusun dari titik-titik kecil yang disebut piksel.

Jika beberapa piksel diletakkan berderet, akan tampaklah suatu garis. Jadi semua garis, sehalus apapun tampaknya dilayar komputer, sebenarnya adalah deretan piksel.

Sebuah piksel memang bisa dianggap sebagai sebuah titik, namun dalam kenyataannya, piksel-piksel lebih mirip dengan persegi panjang kecil yang tingginya tidak sebanding dengan lebarnya.



Gambar 2.6 piksel A, piksel B dan piksel C

2.6. Video Digital

Video digital pada dasarnya tersusun atas serangkaian *frame*. Rangkaian *frame* tersebut ditampilkan pada layar dengan kecepatan tertentu , tergantung pada *frame rate* yang diberikan (dalam *frame/second*). Jika *frame rate* cukup tinggi,

mata manusia tidak dapat menangkap gambar atau *frame*, melainkan menangkapnya sebagai rangkaian yang kontinyu (*video*).

Masing-masing *frame* merupakan citra digital. Suatu citra digital direpresentasikan dengan sebuah matriks yang masing-masing elemennya merepresentasikan nilai intensitas. Jika I adalah matriks dua dimensi, I(x,y) adalah nilai intensitas yang sesuai pada posisi baris x dan kolom y pada matriks tersebut. Titik-titik ditempatkan *image* di *sampling* disebut *picture elements*, atau sering dikenal sebagai piksel.

Karakteristik suatu video digital ditentukan oleh resolusi *(resolution)* atau *frame dimention*, *pixel depth*, dan *frame rate*. Karakteristik-karakteristik ini akan menentukan tawar menawar antara kualitas video dan jumlah *bit* yang dibutuhkan untuk menyimpan atau mentransmisikannya.

2.6.1. Resolusi

Resolusi *(resolution)* atau *frame dimention* adalah ukuran sebuah *frame*. Resolusi dinyatakan dalam piksel x piksel. Semakin tinggi resolusi, semakin baik kualitas video yang dihasilkan, dalam artian bahwa ukuran fisiknya sama, video dengan resolusi tinggi akan lebih detail. Namun resolusi yang tinggi akan mengakibatkan jumlah *bit* yang diperlukan untuk menyimpan atau mentransmisikannya meningkat

2.6.2. Bit Depth

Bit Depth menentukan jumlah *bit* yang digunakan untuk merepresentasikan tiap piksel pada sebuah *frame*. Kedalaman *bit* dinyatakan dalam *bit/pixel*. Semakin banyak jumlah *bit* yang digunakan untuk merepresentasikan sebuah

piksel, yang berarti semakin tinggi kedalaman pikselnya, maka semakin baik pula kualitasnya, dengan konsekuensi jumlah *bit* yang diperlukan menjadi lebih tinggi. Dengan 1 *byte* (8*bit*) untuk tiap piksel, diperoleh 2^8 atau 256 level intensitas. Dengan level intensitas sebanyak itu, umumnya mata manusia sudah dapat dipuaskan. Kedalaman piksel paling rendah terdapat pada *binary value image* yang hanya menggunakan 1 *bit/ pixel*, sehingga hanya ada dua kemungkinan bagi tiap piksel, yaitu 0 untuk hitam dan 1 untuk putih. Pada kenyataannya semakin sedikit jumlah *bit* yang digunakan untuk tiap piksel, maka kualitas gambar akan semakin turun.

2.6.3. Frame Rate

Frame Rate menujukkan jumlah *frame* yang digambarkan tiap detik, dan dinyatakan dengan *frame/second*. Sehubungan dengan *frame rate* ini, ada dua hal yang diperlukan diperhatikan, yaitu kehalusan gerakan *(smooth mation)* dan kilatan *(flash)*. Kehalusan gerakan ditentukan oleh jumlah *frame* yang berbeda per detik. Untuk mendapatkan gerakan yang halus, video setidaknya harus menampilkan sedikitnya 25 *frame/second*. Kilatan ditentukan oleh jumlah berapa kali layar digambar perdetik (*frame rate*), dengan 29 *frame/second* kilatan sudah dapat dilenyapkan.

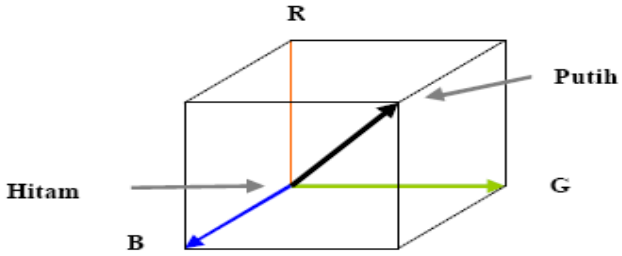
Video yang berkualitas baik akan memiliki *frame rate* yang tinggi, setidaknya sesuai dengan mata manusia, yang berarti membutuhkan jumlah *bit* yang lebih tinggi.

2.6.4. Representasi Warna

Pada video digital, umumnya data video dipisahkan menjadikan komponen-komponen, baik komponen warna ataupun komponen kecerahan, penyajian semacam ini disebut komponen *video*. Pada komponen *video* tiap komponen dipisahkan dengan cara tertentu. Beberapa cara pemisahan komponen tersebut adalah RGB, YUV dan YIQ.

2.6.4.1. Red Green Blue (RGB)

Red Green Blue (RGB) merupakan warna dasar yang ada pada system computer. Data video dapat dipisahkan menjadi komponen-komponen untuk masing-masing warna, yaitu merah *(red),* hijau *(green)* dan biru *(blue).* Warna tiap piksel ditentukan oleh kombinasi intensitas dari masing-masing komponen warna. Sebagai contoh, pada *RGB* 24 bit, masing-masing komponen warna dinyatakan dalam 8 *bit* atau 256 level. Contoh, warna biru langit direpsentasikan dengan R=181, G-189, dan B=249.



Gambar 2.7 Koordinat RGB

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

1. bit 0 sampai dengan 7 untuk warna merah.

2. bit 7 sampai dengan 15 untuk warna hijau
3. bit 16 sampai dengan 24 untuk warna biru

Kemungkinan kombinasi warna yang ada adalah = 2563 + 2562 + 2561 = *16.843.008*, dimana nilai 0 menyatkan warna hitam sedangakan nilai 16 843 008 menyatakan warna putih.

2.6.4.2. YUV

YUV adalah pemisahan komponen kecerahan (*luminance) dan komponen warna (crominanc).* Pemisahan komponen tidak hanya dilakukan dengan pemisahan warna, namun dapat juga dilakukan dengan memisahkan komponen kecerahan *(luminance)* dan komponen warna *(crominance).* Pada format PAL, sinyal kecerahan dinyatakan dengan Y, sedangkan dua signal warna dinyatakan dengan U dan V

Masing-masing komponen tersebut diperoleh dengan mentransformasikan RGB dengan rumus :

$$Y=0{,}299\ R + 0{,}587\ G + 0{,}114\ B$$

$$U=(B-Y) \times 0{,}493$$

$$V=(R-Y) \times 0{,}877 \qquad \text{[II.2.12]}$$

2.6.4.3. Luminance In-phase Quadrature (YIQ)

Luminance in-phase quadrature (YIQ) adalah pemisahan sinyal video menjadi komponen kecerahan dan komponen warna, dapat dilakukan juga sesuai dengan format *NTSC*, komponen kecerahan dinyatakan dengan Y, dan dua komponen warna dinyatakan dengan I dan Q.

Masing-masing komponen tersebut diperoleh dengan mentransformasikan *RGB* dengan rumus :

$$Y=0,299\ R + 0,587\ G + 0,114\ B$$

$$I=0,587R-0,275G-0,321B$$

$$Q=0,212R-0,523G-0,321B$$

[II.2.10] Rumus Mentransformasikan RGB

2.7. Web-Camera

Web-Camera atau yang biasa disebut dengan *webcam* adalah kamera atau video digital yang di hubungkan dengan komputer dan dapat dikoneksikan dengan sebuah jaringan. Kegunaan dari *webcam* adalah:

1) *Chatting.*
2) *Video conference.*
3) *Live audio.*
4) Dapat digunakan dengan *software* pendukung lainnya.

Webcam terhubung dengan komputer dengan media kabel *USB* atau *parallel port*. Agar *webcam* dapat digunakan oleh komputer, kita membutuhkan suatu perangkat lunak. Perangkat lunak itu akan menangkap tiap *frame* pada waktu tertentu, biasanya disebut dengan *frame rate*. *File* yang disimpan biasanya berformat avi (video), tapi dapat diubah dalam bentuk jpeg (*image*) atau yang lainnya agar tidak terlalu banyak memakan banyak tempat pada media penyimpanan.

2.8. Citra

Citra adalah fungsi dua dimensi yang terbentuk dari penglihatan dalam suatu tempat (*scane)* yang merupakan basis dari pembentukan citra.

Secara umum citra dibagi menjadi dua macam, yaitu :

1. Analog, yaitu citra hasil tangkapan lensa kamera terhadap tempat (*scane)* yang di-*scane* secara vertikal dan horizontal oleh *imager* kamera.
2. Digital, yaitu citra yang direpresentasikan sebagai sebuah matriks yang masing-masing elemennya merepresentasikan nilai intensitas.

2.8.1. Citra Digital

Citra digital dapat didefinisikan sebagai fungsi dua variabel, f(x,y), dimana x dan y adalah koordinat spasial dan nilai f(x,y) adalah intensitas citra pada koordinat tersebut. Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (*Red, Green, Blue - RGB*). Sebuah citra diubah ke bentuk digital agar dapat disimpan dalam memori komputer atau media lain. Proses mengubah citra ke bentuk digital bisa dilakukan dengan beberapa perangkat, misalnya scanner, kamera digital, dan handycam. Ketika sebuah citra sudah diubah ke dalam bentuk digital (selanjutnya disebut citra digital).

Agar dapat diolah dengan dengan komputer digital, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai *diskrit.* Representasi citra dari fungsi malar (kontinyu) 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 lebar x panjang). Citra digital tingginya *N,* lebarnya *M,* dan memiliki *L* derajat keabuan dapat dianggap sebagai fungsi:

$$f(\chi, \gamma) \quad \begin{matrix} 0 \le \chi \le N \\ 0 \le \gamma \le M \\ 0 \le f \le L \end{matrix}$$

[II.2.14] Fungsi Derajat Keabuan

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

$$(\chi, \gamma) = \begin{bmatrix} f(0,0) & f(0,1) & \ldots. & f(0,M) \\ f(1,0) & f(1,1) & \ldots. & f(1,M) \\ f(N-1,0) & f(N-1,1) & ..... & f(N-1,M-1) \end{bmatrix}$$

Gambar 2.8 Matrik NxM

Indeks baris (i) dan index 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) disebut *image element, picture element* atau *pixel* atau *pel.* Jadi, citra yang berukuran *N x M* mempunyai *NM* buah *pixel.* Sebagai contoh, misalkan sebuah citra berukuran 256 x 256 *pixel* dan direpresentasikan secara numerik dengan matriks yang terdiri dari 256 buah baris (di-indeks dari 0 sampai 255) dan 256 buah kolom (di-indeks dari 0 sampai 255) seperti contoh berikut:

$$\begin{pmatrix} 0 & 134 & 145 & \dots & \dots & 231 \\ 0 & 167 & 201 & \dots & \dots & 197 \\ 220 & 187 & 189 & \dots & \dots & 120 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 221 & 219 & 210 & \dots & \dots & 156 \end{pmatrix}$$

Gambar 2.9 Representasi citra dengan Matrik 256x256

Pixel pertama pada koordinat (0, 0) mempunyai nilai intensitas 0 yang berarti warna *pixel* tersebut hitam, *pixel* kedua pada koordinat (0,1) mempunyai intensitas 134 yang berarti warnanya antara hitam dan putih, dan seterusnya.

Proses digitalisasi citra ada dua macam:

1. Digitalisasi spasial *(x,*y), sering disebut sebagai penerokan *(sampling).*
2. Digitalisasi intensitas *f*(x, *y),* sering disebut sebagai *kuantisasi.*

2.9. Elemen-elemen Citra Digital

Citra digital mengandung sejumlah elemen-elemen dasar. Elemen-elemen dasar tersebut dimanipulasi dalam pengolahan citra dan dieksploitasi lebih lanjut dalam computer vision. Elemen-elemen dasar yang penting diantaranya adalah:

2.9.1. Kecerahan *(brightness)*

Kecerahan adalah kata lain untuk intensitas cahaya. Sebagaimana telah dijelaskan pada bagian penerokan, kecerahan pada sebuah titik *(pixel)* di dalam citra bukanlah intensitas yang sebenarnya, tetapi sebenarnya adalah intensitas rata-rata dari suatu area yang melingkupinya. Sistem visual manusia mampu

menyesuaikan dirinya dengan tingkat kecerahan *(brightness level)* mulai dari yang paling rendah sampai yang paling tinggi dengan jangkauan sebesar 10^{10}.

2.9.2. Kontras *(contrast)*

Kontras menyatakan sebaran terang *(lightness)* dan gelap *(darkness)* di dalam sebuah gambar. Citra dengan kontras rendah dicirikan oleh sebagian besar komposisi citranya adalah terang atau sebagian besar gelap. Pada citra dengan kontras yang baik, komposisi gelap dan terang tersebar secara merata.

2.9.3. Kontur *(contour)*

Kontur adalah keadaan yang ditimbulkan oleh perubahan intensitas pada *pixel-pixel* yang bertetangga. Karena adanya perubahan intensitas inilah mata kita mampu mendeteksi tepi-tepi *(edge)* objek di dalam citra.

2.9.4. Warna *(color)*

Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna mempunyai panjang gelombang (λ) yang berbeda. Warna merah mempunyai panjang gelombang paling tinggi, sedangkan warna ungu (*violet*) mempunyai panjang gelombang paling rendah.

Warna-warna yang diterima oleh mata (sistem visual manusia) merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda. Penelitian memperlihatkan bahwa kombinasi warna yang memberikan rentang warna yang paling lebar adalah *red (R), green* (G), dan *blue (B).*

Persepsi sistem visual manusia terhadap warna sangat relatif sebab dipengaruhi oleh banyak kriteria, salah satunya disebabkan oleh adaptasi yang

menimbulkan distorsi. Misalnya bercak abu-abu di sekitar warna hijau akan tampak keungu-unguan (distorsi terhadap ruang), atau jika mata melihat warna hijau lalu langsung dengan cepat melihat warna abu-abu, maka mata menangkap kesan warna abu-abu tersebut sebagai warna ungu (distorsi terhadap waktu).

2.9.5. Bentuk *(shape)*

Shape adalah properti intrinsik dari objek tiga dimensi dengan pengertian bahwa *shape* merupakan properti intrinsik utama untuk sistem visual manusia. Manusia lebih sering mengasosiasikan objek dengan bentuknya ketimbang elemen lainnya (warna misalnya). Pada umumnya, citra yang dibentuk oleh mata merupakan citra dwimatra (2 dimensi), sedangkan objek yang dilihat umumnya berbentuk trimatra (3 dimensi).

Informasi bentuk objek dapat diekstraksi dari citra pada permulaaan pra-pengolahan dan segmentasi citra. Salah satu tantangan utama pada *computer vision* adalah merepresentasikan bentuk, atau aspek-aspek penting dari bentuk.

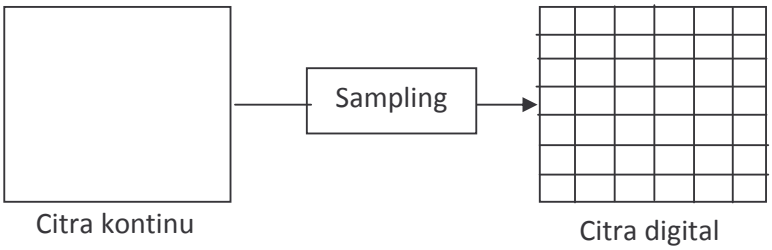
2.9.6. Tekstur *(texture)*

Tekstur dicirikan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan *pixel-pixel* yang bertetangga. Jadi, tekstur tidak dapat didefinisikan untuk sebuah *pixel*. Sistem visual manusia pada hakikatnya tidak menerima informasi citra secara independen pada setiap *pixel,* melainkan suatu citra dianggap sebagai suatu kesatuan. Resolusi citra yang diamati ditentukan oleh skala pada mana tekstur tersebut dipersepsi.

Sebagai contoh, jika kita mengamati citra lantai berubin dari jarak jauh, maka kita mengamati bahwa tekstur terbentuk oleh penempatan ubin-ubin secara keseluruhan, bukan dari persepsi pola di dalam ubin itu sendiri. Tetapi, jika kita mengamati citra yang sama dari jarak yang dekat, maka hanya beberapa ubin yang tampak dalam bidang pengamatan, sehingga kita mempersepsi bahwa tekstur terbentuk oleh penempatan pola-pola rinci yang menyusun tiap ubin.

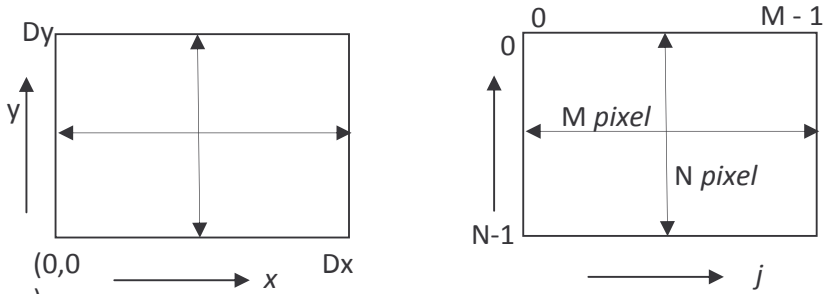
2.10. *Sampling*

Sampling, merupakan proses pengambilan informasi dari suatu titik *pixel* pada grid-grid yang berbentuk bujursangkar (kisi-kisi dalam horizontal dan vertikal). diperlihatkan dalam gambar dibawah ini.



Gambar 2.10 Sampling terhadap suatu citra

Terdapat perbedaan antara koordinat gambar (yang di sampling) dengan koordinat matriks (hasil digitalisasi). Titik asal (0, 0) pada gambar dan elemen (0, 0) pada matriks tidak sama. Koordinat *x* dan *y* pada gambar dimulai dari sudut kiri bawah (gambar 2.9 kiri), sedangkan penomoran *pixel* pada matriks dimulai dari sudut kiri atas. (gambar 2.9 kanan)



Gambar 2.11 koordinat *pixel*

Dalam hal ini :

$i = x , 0 \leq j \leq N-1$

$j = (M-y) \quad , 0 \leq j \leq M-1$

[II.2.12] Persamaan Penomoran Nilai Pixel Pada Matrix

$x = D_x/N$ increment

$y = D_y/N$ increment

N = jumlah maksimum *pixel* dalam satu baris

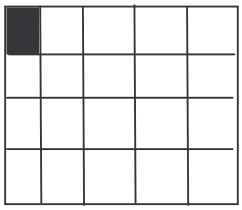
M = jumlah maksimum *pixel* dalam satu kolom

D_x = lebar gambar (dalam inchi)

D_y = tinggi gambar (dalam inchi)

Elemen (i,j) didalam matriks menyatakan rata - rata intensitas cahaya pada area citra yang direpresentasikan oleh *pixel*. Sebagai contoh, tinjau citra biner yang hanya mempunyai 2 derajat keabuan, 0 (hitam) dan 1 (putih). Sebuah gambar yang berukuran 10 x 10 inchi dinyatakan dalam matriks yang berukuran 5x4, yaitu lima baris dan 4 kolom. Tiap elemen gambar lebarnya 2.5 inchi dan tingginya 2 inci akan diisi dengan sebuah nilai bergantung pada rata-rata intensitas cahaya pada area tersebut.

Area 2.5 x 2.0 inchi pada sudut kiri atas gambar dinyatakan dengan lokasi (0, 0) pada matriks 5 x 4 yang mengandung nilai 0 (yang berarti tidak ada intensitas cahaya). Area 2.5 x 2.0 inchi pada sudut kanan bawah gambar dinyatakan dengan lokasi (4,3) pada matrik 5x 4 yang mengandung nilai 1 (yang berarti iluminasi maksimum).



Gambar 2.12 Matrik sampling

Untuk memudahkan implementasi, jumlah *sampling* biasanya diasumsikan perpangkatan dari dua,

$$N = 2^n$$

[II.2.13] Rumus Penentuan Jumlah Sampling

yang dalam hal ini,

N = jumlah *sampling* pada suatu baris/kolom

n = bilangan bulat positif

Contoh ukuran penerokan: 256 x 256 *pixel,* 128 x 256 *pixel.* Pembagian gambar menjadi ukuran tertentu menentukan resolusi (yaitu derajat rincian yang dapat dilihat) spasial yang diperoleh. Semakin tinggi resolusinya yang berarti semakin kecil ukuran *pixel* (atau semakin banyak jumlah *pixel-nya),* semakin halus gambar yang diperoleh karena informasi yang hilang akibat pengelompokan derajat keabuan pada penerokan semakin kecil.

2.11. Kuantisasi

Kuantisasi adalah proses membagi skala keabuan *(0,* L) menjadi *G* buah level yang dinyatakan dengan suatu harga bilangan bulat *(integer),* biasanya *G* diambil perpangkatan dari dua.

$$G = 2^m$$

[II.2.14] Rumus Membagi Skala Keabuan

G = derajat keabuan

m = bilangan bulat poitif

Tabel 2.1 Kuantisasi

Skala	Rentang nilai Keabuan	*Pixel* Depth
2^1 (2nilai)	0,1	Ibit
2^2 (4nilai)	0 sampai 7	2 bit
2^3 (16 nilai)	0 sampai 15	3 bit
2^8 (256 nilai)	0 sampai 255	8 bit

Hitam dinyatakan dengan nilai derajat keabuan terendah yaitu 0, sedangkan putih dinyatakan dengan nilai derajat keabuan tertinggi misalnya 15 untuk 16 level. Jumlah bit yang dibutuhkan untuk mereprentasikan nilai keabuan *pixel* disebut *kedalaman pixel (pixel depth).* Citra sering diasosiasikan dengan kedalaman *pixel-nya.* Jadi, citra dengan kedalaman 8 bit disebut juga citra 8-bit (atau citra 256 warna). Pada kebanyakan aplikasi, citra dikuantisasi pada 256 level dan membutuhkan 1 *byte* (8 bit) untuk representasi setiap *pixel-nya (G* - 256 = 2^8). Sedangkan citra biner *(binary image)* hanya dikuantisasi pada dua level: 0 dan 1. Tiap *pixel* pada citra biner cukup direpresentasikan dengan 1 bit, yang mana bit 0 berarti hitam dan bit 1 berarti putih.

Besarnya daerah derajat keabuan yang digunakan menentukan resolusi kecerahan dari gambar yang diperoleh. Sebagai contoh, jika digunakan 3 bit untuk

menyimpan harga bilangan bulat, maka jumlah derajat keabuan yang diperoleh hanya 8, jika digunakan 4 bit, maka derajat keabuan yang diperoleh adalah 16. Semakin banyak jumlah derajat keabuan (berarti jumlah bit kuantisasinya makin banyak), maka semakin baik pula gambar yang diperoleh, karena derajat keabuan yang semakin tinggi akan menghasilkan sebuah citra yang mendekati citra aslinya.

2.12. *Edge Detection*

Yang dimaksud dengan *edge detection* adalah pendeteksian terhadap perubahan nilai intensitas derajat keabuan yang mendadak besar dalam jarak yang pendek. Perbedaan intensitas inilah yang menampakan rincian pada gambar. Tepi biasanya terdapat pada batas antara dua daerah berbeda pada citra. Tepi dapat diorientasikan dengan suatu arah, dan arah ini berbeda-beda tergantung pada perubahan intensitas. Ada tiga macam tepi yang terdapat dalam *citra digital*, yaitu:

a. Tepi curam, yaitu tepi dengan perubahan intensitas yang tajam. Arah tepi berkisar 90 derajat.
b. Tepi landai, yaitu tepi dengan sudut arah yang kecil. Tepi landai dapat dianggap terdiri dari sejumlah tepi-tepi lokal yang lokasinya berdekatan.
c. Tepi yang mengandung derau, yaitu umumnya tepi yang terdapat pada aplikasi *computer vision* mengandung derau.

Operasi peningkatan kuallitas citra (*image enhancement*) dapat dilakukan terlebih dahulu sebelum pendeteksian tepi. Tujuan dari pendeteksian tepi adalah untuk meningkatkan penampakan garis batas suatu daerah atau objek dalam citra. Nilai intensitas L(x,y) dari gambar diambil dari persamaan :

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

[II.2.15] Persamaan Nilai Intensitas

Ini dapat dihitung dengan menggunakan *convolution filter*.

Ketika citra masukkan diperlihatkan sebagai kumpulan piksel tersendiri, selanjutnya mencari lilitan kernel tersendiri yang dapat diperkirakan asalnya. Tiga kernel kecil yang biasa digunakan diperlihatkan pada gambar dibawah ini :

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

−1	2	−1
2	−4	2
−1	2	−1

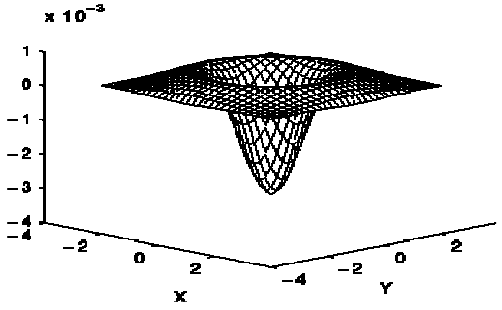
Gambar 2.13 Tiga kernel yang biasa digunakan dari *Laplacian filter*

Fungsi LoG 2-D dengan standar persamaan Gaussian :

$$LoG(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

[II.2.16] Persamaan Gaussian

Dan diperlihatkan pada gambar 2.11 :

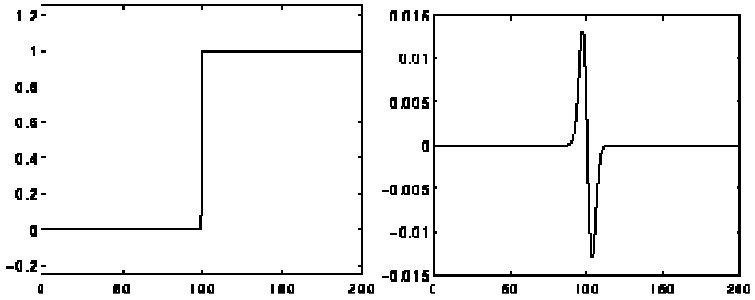


Gambar 2.14 Fungsi 2-D Laplacian of Gaussian. X dan y adalah tanda dari standar deviasi (σ).

Perkiraan kernel dari fungsi LoG (untuk Gaussian σ = 1.4) diperlihatkan pada gambar 2.15 :

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

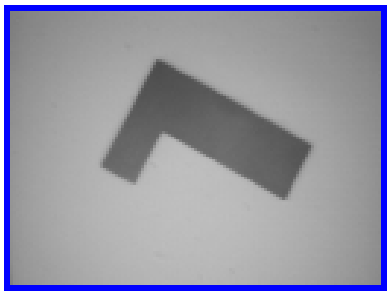
Gambar 2.15 Perkiraan fungsi LoG dengan Gaussian $\sigma = 1.4$



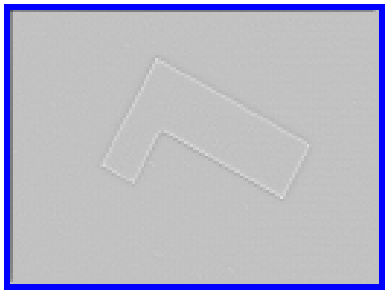
Gambar 2.16 Tanggapan dari 1-D LoG Filter dari tahap pinggiran.

Gambar sebelah kiri menunnjukan 1-D piksel, gambar sebelah kanan menunjukan grafik dari 1-D LoG filter dengan Gaussian $\sigma = 3$ piksel.

Hasil dari deteksi pinggiran (*edge detection*) pada proses diatas adalah :



Gambar 2.17 gambar sederhana dengan pinggiran yang jelas



Gambar 2.18 Hasil dari deteksi pinggiran (edge detection)

2.13. *Tresholding*

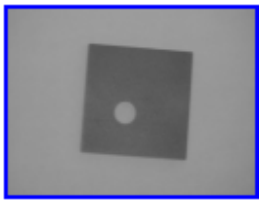
Thresohlding adalah proses perubahan *gray-level image* kedalam *bilevel image*. *Bilevel image* adalah suatu *image monochrome* yang terdiri dari *piksel* hitam dan putih. *Bilevel image* terdiri dari banyak informasi yang penting dari sebuah *image* (angka, posisi dan bentuk dari objek). Pengelompokan *image* oleh piksel *bilevel* dapat memperkecil dan memudahkan beberapa pengoperasian proses *image* seperti membentuk *recognition* dan *classification*.

Secara keseluruhan nilai *bilevel*s menunjukan hasil yang dapat di klasifikasikan menjadi hitam(0) dan putih(1). Adanya kekasaran (*noise)* dan efek cahaya (*illumination)* akan mempengaruhi nilai yang dihasilkan dari pemrosesan.

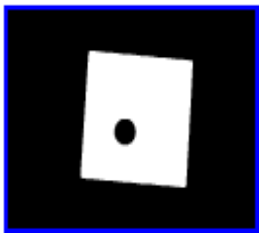


Gambar 2.19 Proses Tresholding

Masukkan dari operasi *tresholding* adalah gambar *grayscale* atau gambar berwarna. Pada implementasi sederhana, keluaran dari operasi *tresholding* adalah gambar binari yang diperlihatkan dengan pembagian (segmentasi). Piksel yang hitam dianggap sebagai *background* dan piksel putih dianggap sebagai *foreground.* Pada implementasi pembagian (segmentasi) sederhana ditentukan dengan satu parameter yang diketahui sebagai intensitas dari *threshold*. Pada satu nilai, piksel yang lain pada gambar dikompres dengan *threshold* ini. Jika intensitas piksel lebih tinggi dari pada *threshold*, maka piksel akan diset sebagai warna putih pada keluarannya. Jika piksel lebih kecil dari *threshold*, maka akan diset sebagai warna hitam. Banyak *Threshold* dapat dispesifikasikan, jadi sekumpulan dari nilai intensitas dapat diset sebagai warna putih sementara yang lainnya diset sebagai warna hitam.



Gambar 2.20 Masukkan gambar *Thresholding*



Gambar 2.21 Hasil *Thresholding*

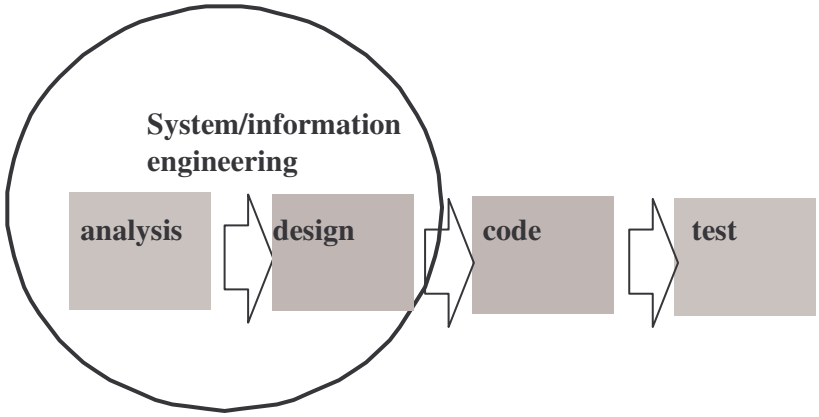
2.14. Metode Pengembangan Model Perangkat Lunak

2.14.1. Metode Waterfall

2.14.1.1. Pengertian Metode Waterfall

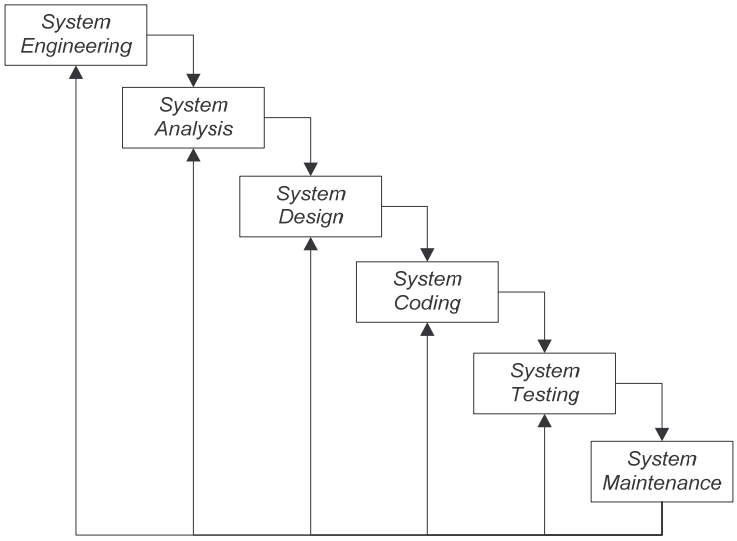
Model ini adalah model klasik yang bersifat sistematis, berurutan dalam membangun software. Berikut ini ada dua gambaran dari Waterfall Model. Sekalipun keduanya menggunakan nama-nama fase yang berbeda, namun sama dalam intinya.

Fase-fase dalam Waterfall Model menurut referensi Pressman :



Gambar 2.22 Wterfall Model menurut Pressman

Fase-fase dalam Waterfall Model menurut referensi Sommerville :



Gambar 2.23 Wterfall Model menurut Sommerville

2.14.1.2. Fase-fase Pada Waterfall Model

1. ***Requirements analysis and definition :*** Mengumpulkan kebutuhan secaralengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh program yang akan dibangun.fase ini harus dikerjakan secara lengkap untuk bias menghasilkan desain yang lengkap.
2. ***System and software design :*** Desain dikerjakan setelah kebutuhan selesai dikumpulkan secara lengkap.
3. ***Implementation and unit testing :*** Desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji secara unit.
4. ***Integration and system testing :*** Penyatuan unit-unit program kemudian diuji secara keseluruhan (system testing).

5. ***Operation and maintenance :*** Mengoperasikan program dilingkungannya dan melakukan pemeliharaan seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya.

2.14.2. Metode Spiral

2.14.2.1. Pengertian Metode Spiral

Pada Spiral Model prosesnya digambarkan sebagai spiral. Setiap loop mewakili satu fase dari software process. Loop paling dalam berfokus pada kelayakan dari sistem, loop selanjutnya tentang definisi dari kebutuhan, loop berikutnya dengan desain sistem dan seterusny. Pada Spiral Model setiap loop dibagi menjadi beberapa sector.

2.14.2.2. Sektor-sektor Pada Spiral Model

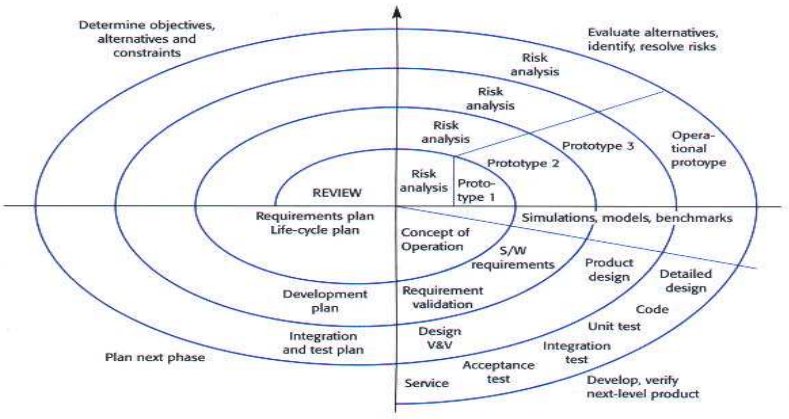
1. ***Objective setting*** **(menentukan tujuan) :** Menentukan tujuan dari fase yang ditentukan. Batasan-batasan pada proses dan produk sudah diketahui. Perencanaan sudah disiapkan. Resiko dari proyek sudah diketahui. Alternative strategi sudah disiapkan berdasarkan resiko-resiko yang diketahui, dan sudah direncanakan.
2. ***Risk assessment and reduction*** **(penanganan dan pengurangan resiko) :** Setiap resiko dianalisis secara detail pada sector ini. Langkah-langkah penanganan dilakukan, misalnya membuat prototype untuk mengetahui ketidak cocokan kebutuhan.
3. ***Development and validation*** **(pembangunan dan pengujian) :** Setelah evaluasi resiko, maka model pengembangan sistem dipilih. Misalnya jika resiko user interface dominan, maka membuat prototype

User Interface. Jika bagian keamanan yang bermasalah, makamenggunakan model formal dengan perhitungan matematis, dan jika masalahnya adalah integrasi system, maka model Waterfall lebih cocok.

4. ***Planning*** **(perencanaan) :** Proyek dievaluasi atau ditinjau ulang dan diputuskan untuk terus ke fase loop selanjutnya atau tidak. Jika melanjutkan ke fase berikutnya rencana untuk loop selanjutnya.

Pada Model Spiral, resiko sangat dipertimbangkan. Resiko adalah sesuatu yang dapat memungkinkan terjadinya sesuatu yang mengakibtakan kesalahan.

Model Spiral merupakan pendekatan yang realistic berskala besar, karena setiap kemajuan yang dicapai selama proses dapat diamati dengan baik. Namun demikian, waktu yang cukup panjang mungkin bukan pilihan bagi pengguna, karena waktu yang lama sama dengan biaya yang lebih besar.



Gambar 2.24 Spiral Model

2.14.3. Metode Prototyping

2.14.3.1. Pengertian Metode Prototyping

Kadang-kadang user hanya memberikan beberapa kebutuhan umum software tenpa detail input, proses atau detail output. Dilan waktu mungkin dimana tim pembangun (*developer*) tidak yakin terhadap efisiensi dari algoritma yang digunakan, tingkat adaptasi terhadap system operasi atau rancangan form user interface. Ketika situasi seperti ini terjadi, Model Prototyping sangat membantu proses pengembangan software.

2.14.3.2. Proses-prose Pada Prototyping Model

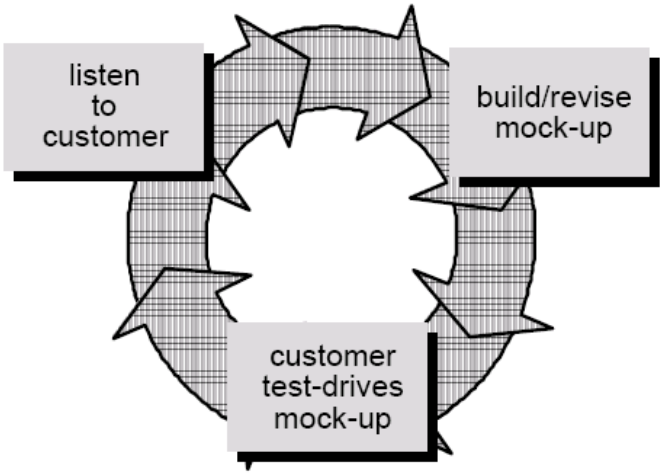
Prose pada Model Prototyping yang digambarka pada gambar 2.4 , bisa dijelaskan sebagai berikut :

1. **Pengumpulan Kebutuhan :** Developer dan user bertamu untuk menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detail kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan.
2. **Perancangan :** Perancangan dilakukan cepat dan rancangan mewakili semua aspek software yang diketahui, dan rancangan ini menjadi dasar pembuatan prototype.
3. **Evaluasi Prototype :** User mengevaluasi prototype yang dibuat dan digunakan untuk memperjelas kebutuhan software.

Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. Prototype-prototype dibuat untuk memuaskan kebutuhan user dan untuk memahami kebutuhan user lebih baik.

Prototype yang dibuat dapat dimanfaatkan kembali untuk membangun software lebih cepat, namun tidak semua prototype bias dimanfaatkan.

Agar model ini bisa berjalan dengan baik, perlu disepakati bersama oleh user dan developer bahwa prototype yang dibangun merupakan alat untuk mendefinisikan kebutuhan software.



Gambar 2.25 Prototyping Model

2.15. Basis Data *(Database)*

2.15.1. Pengertian Basis Data

Basis data adalah kumpulan data yang saling berelasi satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di dalam *hardware* komputer dan dijalankan dengan menggunakan *software* untuk melakukan manipulasi untuk kegunaan tertentu. Adapun kegunaan basis data adalah :

1. Basis data merupakan komponen penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.

2. Menentukan kualitas informasi yang akurat, tepat pada waktunya, dan relevan, dimana informasi dikatakan bernilai jika manfaatnya lebih efektif dibandingkan dengan biaya untuk mendapatkannya.
3. Mengurangi duplikasi data *(data redudancy).*
4. Meningkatkan relasi antar data *(data relatability).*
5. Mengurangi pemborosan *external storage.*

Di dalam basis data terdapat beberapa istilah berikut yang juga dikenal sebagai urutan jenjang data :

1. *Characters*

   Merupakan bagian data yang terkecil yaitu berupa karakter numerik, huruf, maupun karakter spesial (simbol) yang membentuk item data/ *field.*

2. *Field*

   Merepresentasikan suatu atribut dari *record* yang menunjukan suatu item dari data, misalnya nama.

3. *Record*

   Merupakan kumpulan dari *field*, menggambarkan suatu unit data individu tertentu. Kumpulan *record* membentuk suatu file. Misalnya, file personalia, tiap-tiap *record* mewakili data tiap-tiap karyawan.

4. *File*

   File terdiri dari *record-record* yang menggambarkan satu kesatuan data yang sejenis. Misalnya, file mata kuliah berisi data tentang semua mata kuliah yang ada.

5. *Database*

   Merupakan kumpulan dari file/ tabel.

2.15.2. Normalisasi Basis Data

Normalisasi merupakan sebuah teknik dalam logika desain sebuah basis data, yaitu mengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redudansi).

a. Normalisasi ke-1

   Suatu data dikatakan normal pertama jika sudah memiliki karakter berikut ini :

   a. Mendefinisikan atribut kunci *(key)*.
   b. Tidak ada grup yang berulang.
   c. Semua atribut bukan *key* tergantung pada atribut kunci.

b. Normalisasi ke-2

   Suatu data dikatakan normal kedua jika sudah memiliki karakter berikut ini :

   a. Sudah memenuhi dalam bentuk normal pertama.
   b. Sudah tidak ada ketergantungan parsial, dimana seluruh *field* tergantung pada sebagian *key field.*

c. Normalisasi ke-3

   Suatu data dikatakan normal jika sudah memiliki karakter berikut ini :

   a. Sudah berada pada bentuk normal kedua.
   b. Tidak ada ketergantungan transitif (*field* bukan *key* bergantung pada *field* bukan *key* lainnya).

2.15.3. ERD (Entity Relationship Diagram)

ERD merupakan notasi grafis dalam pemodelan data konseptual yang mendeskripsikan hubungan antara penyimpanan (tabel). *ERD* digunakan untuk memodelkan struktur data dan hubungan antar data, kita dapat menguji data dengan mengabaikan proses yang harus dilakukan serta bisa mengetahui data apa yang kita perlukan, serta bagaimana data-data tersebut bisa saling berhubungan. Pada dasarnya terdapat 3 macam simbol yang digunakan untuk menggambarkan struktur dan hubungan antar data, yaitu :

a. Entiti

   Suatu objek, merupakan bagian dari sitem yang dapat diidentifikasi dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat. Entiti dapat berupa sebuah kelompok dari sesuatu, harus dapat dibedakan.



b. Atribut

   Merupakan elemen dari entiti. Setiap entiti memiliki atribut yang berfungsi untuk mendeskripsikan karakternya. Misalnya, entiti mahasiswa memiliki atribut nim, nama, dan sebagainya

   .



c. *Relationship* (hubungan)

   Merupakan penghubung antar satu entiti dengan entiti yang lain, ataupun satu entiti ke dalam entiti itu sendiri. *Relationship* dibagi menjadi 3 jenis, yaitu :

1. Satu ke satu (1-1)

   Seorang supir hanya dapat menyetir satu mobil. Satu mobil pun hanya bisa digunakan oleh seorang supir.



2. Satu ke banyak/ banyak ke satu(1-n / n-1)

   Satu kelas dapat terdiri dari banyak siswa, dan banyak siswa bisa belajar di dalam satu kelas. Tapi tidak sebaliknya, seorang siswa tidak bisa belajar di kelas yang berbeda.



3. Banyak ke banyak (n-n)

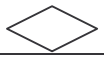
   Seorang karyawan bisa mengerjakan satu atau lebih proyek. Dan setiap proyek bisa dikerjakan oleh seorang atau lebih karyawan.



2.15.4. Flowchart

Flowchart adalah blok-blok diagram yang menggambarkan alur sebuah program dan merupakan alat bantu yang digunakan untuk menggambarkan sistem picikan yang tersusun secara baik. Saat ini telah diterbitkan standar untuk diagram alir oleh Institut Nasional Amerika (ANSI). Gambar simbol-simbol flowchart dalam standar ANSI dapat dilihat pada tabel 2.1

Tabel 2.2 Simbol-simbol Flowchart dalam standar ANSI

Simbol	Keterangan
	Input/ Output
	Proses/ Pengolahan
	Penghubung
	Akhir, Penyela
	Garis Alir
	Keputusan
	Proses yang dirumuskan
	Persiapan
	Operasi Manual
	Pita Magnetic
	Dokumen
	Tampilan

2.15.5. Sistem Manajemen Basis Data *(Database Management System - DBMS)*

Sistem Manajemen Basis Data adalah perangkat lunak yang didesain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar, serta menjadi alternatif penggunaan secara khusus untuk aplikasi, seperti penyimpanan data dalam file dan menulis kode aplikasi yang spesifik untuk pengaturannya.

2.16. DFD (Data Flow Diagram)

2.16.1. Pengertian *DFD (Data Flow Diagram)*

DFD (Data Flow Diagram) adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut.

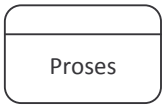
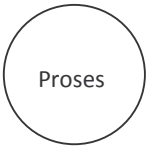
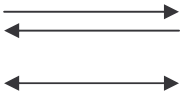
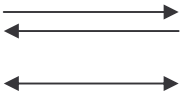
DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan.

DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. Kelebihan utama pendekatan aliran data, yaitu :

1. Kebebasan dari menjalankan implementasi teknis sistem.
2. Pemahaman lebih jauh mengenai keterkaitan satu sama lain dalam sistem dan subsistem.
2. Mengkomunikasikan pengetahuan sistem yang ada dengan pengguna melalui diagram aliran data.
3. Menganalisis sistem yang diajukan untuk menentukan apakah data-data dan proses yang diperlukan sudah ditetapkan.

2.16.2. Simbol-Simbol *DFD (Data Flow Diagram)*

Tabel 2.3 Simbol-simbol *DFD (Data Flow Diagram)*

Gane/ Sarson	Yourdon/ De Marco	Keterangan
		Entitas eksternal adalah entity (human/system) yang berinteraksi langsung dengan sistem yang akan dibuat atau dikembangkan
		Proses merupakan gambaran proses-proses atau fungsi-fungsi yang terdapat dalam suatu sistem.
		Aliran data merupakan arah aliran data yang digunakan dalam suatu proses.
		Data store merupakan media penyimpanan data yang digunakan dalam system yang dibangun/dibuat.

2.17. MySQL

2.17.1. Pengertian MySQL

MySQL adalah salah satu database yang sering digunakan dalam aplikasi berbasis web, namun sebenarnya aplikasi database ini sering pula digunakan dalam aplikasi desktop.

MySQL mempunyai beberapa kelebihan, selain RDBMS yang handal, database open source, mensupport trigger dan store procedure juga dapat berjalan pada lebih dari 20 platform termasuk Linux, Windows, OS/X, HP-UX, AIX, dan Netware. Versi terakhir MySQL adalah versi 6.2 yang masih dalam proses pengembangan.

2.18. Algoritma

2.18.1. Pengertian Algoritma

Algoritma adalah urutan langkah logis tertentu untuk memecahkan suatu masalah. Yang ditekankan adalah urutan langkah logis, yang berarti algoritma harus mengikuti suatu urutan tertentu, tidak boleh melompat-lompat. Alur pemikiran dalam menyelesaikan suatu pekerjaan yang dituangkan secara tertulis. Yang ditekankan pertama adalah alur pikiran, sehingga algoritma seseorang dapat juga berbeda dari algoritma orang lain. Sedangkan penekanan kedua adalah tertulis, yang artinya dapat berupa kalimat, gambar, atau table tertentu.

Dalam bidang computer, algoritma sangat diperlukan dalam menyelesaikan berbagai masalah pemrograman, terutama dalam komputasi numeris. Tanpa algoritma yang dirancang baik maka proses pemrograman akan menjadi salah, rusak, atau lambat dan tidak efisien. Pelaksanaan algoritma adalah komputer.

Manusia dan komputer berkomunikasi dengan cara, manusia memberikan perintah-perintah kepada komputer berupa instruksi-instruksi yang disebut program.

2.19. *Visual Studio 2008*

2.19.1. Sejarah *Visual Studio 2008*

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, Integrated Development Environment (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

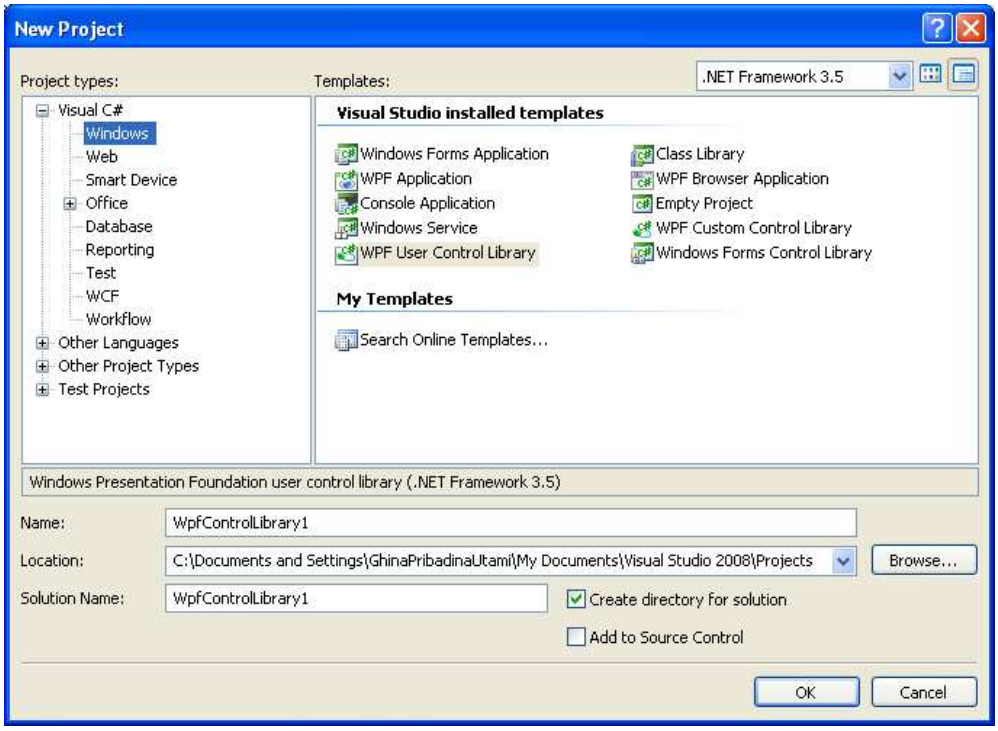
Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

Visual Studio kini telah menginjak versi Visual Studio 9.0.21022.08, atau dikenal dengan sebutan Microsoft Visual Studio 2008 yang diluncurkan pada 19

November 2007, yang ditujukan untuk platform Microsoft .NET Framework 3.5. Versi sebelumnya, Visual Studio 2005 ditujukan untuk platform .NET Framework 2.0 dan 3.0. Visual Studio 2003 ditujukan untuk .NET Framework 1.1, dan Visual Studio 2002 ditujukan untuk .NET Framework 1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan Visual Studio .NET, karena memang membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Studio .NET, terdapat Microsoft Visual Studio 6.0 (VS1998).

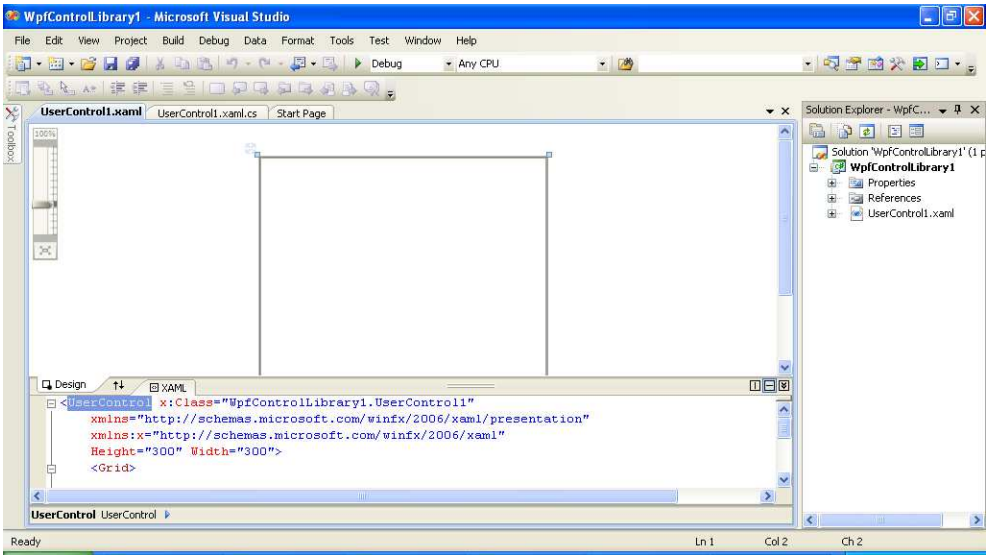
2.19.2. *Integrated Development Environment* (IDE) *Visual Studio 2008*

Merupakan lingkungan pengembangan dari *Visual Studio 2008* yang terintegrasi. Ketika pertama kali *Visual Studio 2008* diaktifkan maka akan muncul muncul tampilan seperti pada gambar 2.26.



gambar 2.26 Layar *New Project*

Setelah kita keluar dari layar *new project*, selanjutnya akan tampil layar kerja atau area kerja dari *Visual Studio 2008*.



gambar 2.27 Layar kerja *Visual Studio 2008*

Keterangan dari gambar diatas :

1. Menubar

   Menubar berfungsi untuk memilih tugas-tugas tertentu, seperti memulai, membuka dan menyimpan *project*, mengompilasi *project* menjadi file *executable* (*exe*) dan lain-lain.

2. Toolbar

   Toolbar memiliki fungsi yang sama seperti menubar, dan juga berfungsi seperti jalan pintas, Karena lebih praktis dalam penggunaannya.

3. Toolbox

   Toolbox berisi komponen-komponen yang merupakan sarana untuk membentuk *user interface*.

Bila *Toolbox* tidak muncul klik tombol *Toolbox* () pada bagian *Toolbar* atau klik menu *View* > *Toolbox*.

4. Jendela Form Designer

   Form designer merupakan tempat untuk merancang *user interface*, dimana kontrol-kontrol yang dibutuhkan bisa diletakan.

   Bila Jendela *Form designer* tidak muncul klik tombol *View Object* () pada bagian *Project Explorer* atau klik menu *View* > *Object*.

5. Jendela Kode

   Jendela kode merupakan tempat untuk menuliskan kode-kode bagi objek yang dibuat. Melalui jendela ini dapat mengatur karakteristik dari form yang dibuat.

   Bila Jendela *Kode* tidak muncul klik tombol *View Code* () di pada bagian *Project Explorer* atau klik menu View > *Code*.

6. Jendela Project

   Jendela *project* adalah jendela yang menampilkan semua file yang berhubungan dengan aplikasi atau *project* yang saat itu sedang dijalankan.

   Bila *Project Explorer* tidak muncul klik tombol *Project Explorer* () pada bagian *Toolbar* atau klik menu *View* > *Project Explorer*.

7. Jendela Properties

   Jendela properties ini berisi daftar property untuk objek (form atau control) yang dipilih dan berfungsi untuk mengatur karakteristik, seperti warna, ukuran dan lain-lain.

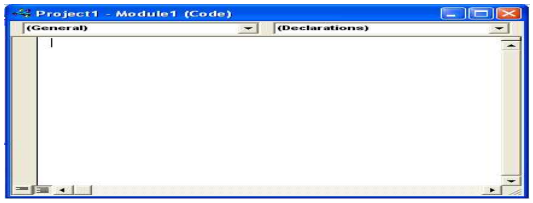
   Bila Jendela *Properties* tidak muncul klik tombol *Properties Window* () pada bagian *Toolbar.*

Banyak Sekali *tool–tool* dan jendela- jendela pendukung lainnya yang ada di lingkungan *Visual Studio 2008* sebagai alat untuk mengembangkan aplikasi yang akan dibuat. Seperti jendela *form layout* yang menunjukan tampilan form pada saat dijalankan.



gambar 2.28 Tampilan jendela form layout

Ada juga jendela modul yang merupakan tempat untuk menuliskan kode-kode yang lebih bersifat global. Pada jendela modul ini, bisa mendeklarasikan variabel dan fungsi dengan awalan *private* maupun *public*. Jika diberi awalan *private*, variabel atau fungsi tersebut hanya berlaku untuk pemakaian modul yang bersangkutan. Sedangkan awalan *public* akan mengakibatkan variabel atau fungsi tersebut bersifat global. Jendela modul ini bisa ditampilkan dengan mengklik menu *project* pada menubar, kemudian pilih *add module*.



gambar 2.29 Tampilan jendela modul

2.19.3. Struktur program *Visual Studio 2008*

Pada saat membangun sebuah aplikasi, digunakan *project* untuk mengatur semua file yang digunakan dalam suatu aplikasi.

Project itu sendiri terdiri atas :

a. Sebuah file project yang mempertahankan semua komponen (.vbp)

b. Sebuah file untuk formnya (.frm)

c. Sebuah file untuk setiap module standard **(.bas)**

d. Sebuah atau beberapa file yang berisi kontrol Active – X **(.ocx)**

e. Sebuah file resource **(.res)**, file ini tidak harus ada.

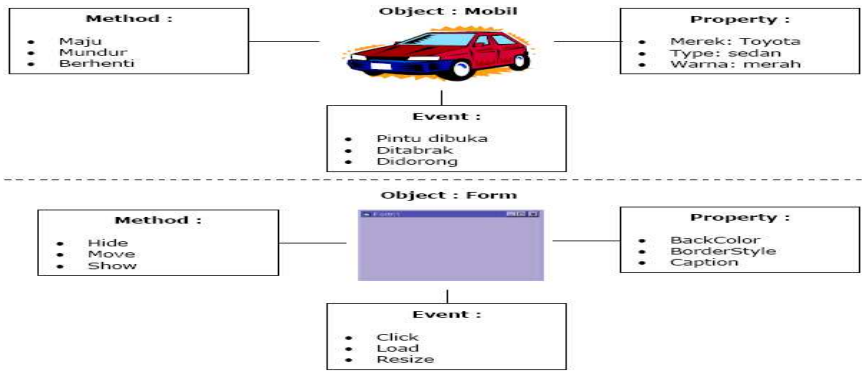
File Project adalah semua daftar file dan objek yang telah dihubungkan dengan project, termasuk informasi pengaturan yang ada.

2.19.4. Istilah-istilah dalam *Visual Studio 2008*

Dalam pemrograman berbasis objek , ada beberapa istilah yang perlu dipahami yaitu *object , property, method* dan *event* sebagai berikut :

1. *Object* adalah komponen di dalam sebuah program
2. *Property* adalah karakteristik yang dimiliki object
3. *Method* adalah aksi yang dapat dilakukan oleh object
4. *Event* adalah kejadian yang dapat dialami oleh object

Sebagai ilustrasi dapat dianggap sebuah mobil sebagai objek yang memiliki *property, method* dan *event*.



Gambar 2.30 Ilustrasi objek, property, method dan even

Implementasinya dalam sebuah aplikasi pembuatan form, maka form tersebut memiliki *property*, *method*, dan *event*. Sebagaimana pemrograman visual lain seperti Delphi dan Java, VB juga bersifat *event driven progamming*. Artinya dapat menyisipkan kode program pada event yang dimiliki suatu obyek.

2.19.5. Data dan Variabel dalam *Visual Studio 2008*

Visual Studio 2008 mengenal beberapa tipe data, antara lain :

1. String adalah tipe data untuk teks (huruf, angka dan tanda baca).
2. Integer adalah tipe data untuk angka bulat.
3. Single adalah tipe data untuk angka pecahan.
4. Currency adalah tipe data untuk angka mata uang.
5. Date adalah tipe data untuk tanggal dan jam.
6. Boolean adalah tipe data yang bernilai *true* atau *false*.

Data yang disimpan di dalam memory komputer membutuhkan sebuah wadah. Wadah inilah yang disebut dengan variabel. Setiap variabel untuk menyimpan data dengan tipe tertentu membutuhkan alokasi jumlah memori (*byte*) yang berbeda. Variabel dibuat melalui penulisan deklarasi variabel di dalam kode program :

Dim <nama_variabel> As <type_data>

Aturan di dalam penamaan variabel :

1. Harus diawali dengan huruf.
2. Tidak boleh menggunakan spasi. Spasi bisa diganti dengan karakter *underscore* (_).
3. Tidak boleh menggunakan karakter-karakter khusus (seperti : +, -, *, /, <)

4. Tidak boleh menggunakan kata-kata kunci yang sudah dikenal oleh Visual Studio 2008 (seperti : dim, as, string, integer, dll).

Sebuah variabel hanya dapat menyimpan satu nilai data sesuai dengan tipe datanya. Cara mengisi nilai data ke dalam sebuah variabel :

<nama_variabel> = <nilai_data>

Untuk tipe data tertentu nilai_data harus diapit tanda pembatas. Tipe data *string* dibatasi tanda petik ganda : "nilai_data". Tipe data *date* dibatasi tanda pagar : #nilai_data#. Type data lainnya tidak perlu tanda pembatas.

Sebuah variabel mempunyai ruang-lingkup (*scope*) dan waktu-hidup (*lifetime*) :

1. Variabel global adalah variabel yang dapat dikenali oleh seluruh bagian program. Nilai data yang tersimpan didalamnya akan hidup terus selama program berjalan.
2. Variabel lokal adalah variabel yang hanya dikenali oleh satu bagian program saja. Nilai data yang tersimpan didalamnya hanya hidup selama bagian program tersebut dijalankan.

Variabel yang nilai datanya bersifat tetap dan tidak bisa diubah disebut konstanta. Penulisan deklarasi konstanta di dalam kode program :

Const <nama_kontanta> As <type_data> = <nilai_data>