BAB 2
LANDASAN TEORI

2.1 Pengertian Jantung

Jantung adalah sebuah rongga organ berotot yang memompa darah ke pembuluh darah dengan irama yang berulang. Istilah kardiak berarti berhubungan dengan jantung, dari kata Yunani *cardia* untuk jantung. Jantung adalah salah satu organ manusia yang berperan penting dalam sistem peredaran darah.

Letak Jantung berada agak sebelah kiri bagian dada, di antara paru-paru kanan dan paru-paru kiri. Beratnya kurang lebih 300 gram, besarnya kira-kira sebesar kepalan tangan. Fungsi jantung untuk memompa darah. Maka dengan adanya jantung, darah dapat dialirkan ke seluruh tubuh melalui pembuluh darah, dan jika peredaran ini terganggu maka inilah yang disebut sakit jantung.

Jantung hampir sepenuhnya diselubungi oleh paru-paru, tertutup oleh selaput ganda yang bernama *perikardium,* yang tertempel pada *diafragma*. Lapisan pertama menempel sangat erat pada jantung, sedangkan lapisan luarnya lebih longgar dan berair, untuk menghindari gesekan antar organ dalam tubuh yang terjadi karena gerakan memompa jantung.

Secara internal, jantung dipisahkan oleh sebuah lapisan otot menjadi dua belah bagian, dari atas ke bawah, menjadi dua pompa. Kedua pompa ini sejak lahir tidak pernah tersambung. Belahan ini terdiri dari dua rongga yang dipisahkan oleh dinding jantung. Maka dapat disimpulkan bahwa jantung terdiri dari empat rongga, serambi kanan dan kiri dan bilik kanan dan kiri.

Dinding serambi jauh lebih tipis dibandingkan dinding bilik karena bilik harus melawan gaya gravitasi bumi untuk memompa dari bawah ke atas dan memerlukan gaya yang lebih besar untuk menyuplai peredaran darah besar, khususnya pembuluh aorta, untuk memompa ke seluruh bagian tubuh yang memiliki pembuluh darah.

Tiap serambi dan bilik pada masing-masing belahan jantung disambungkan oleh sebuah katup. Katup di antara serambi kanan dan bilik kanan disebut katup *trikuspidalis* atau katup berdaun tiga. Sedangkan katup yang ada di

antara serambi kiri dan bilik kiri disebut katup *mitralis* atau katup *bikuspidalis* (katup berdaun dua).

Berikut bagian-bagian jantung :

1. Dinding dan Otot Jantung

Dinding jantung terdiri dari sel-sel unik yang disebut *myocardium* yang berisi otot jantung. Otot ini merupakan jenis satu-satunya yang terdapat pada tubuh karena dia harus memiliki oksigen sepanjang waktu untuk tetap hidup dan agar jantung yang melewati pembuluh darah arteri koroner harus berlangsung secara kontiniu.

Inilah sebabnya penyempitan pada arteri koroner amat berbahaya terhadap jantung. Bila pembuluh arteri menyempit, dapat menyebabkan hal-hal yang tidak diinginkan seperti sakit pada jantung mulai dari sakit dada, dan serangan jantung yang berpotensi meninggal mendadak. Adapun rongga atau ruangan dalam jantung dibagi menjadi dua ruang atas disebut serambi jantung atau *atrium* sebelah kanan dan kiri dan ruang bawah disebut bilik jantung atau *ventrical* sebelah kanan dan kiri. Ruangan-ruangan tersebut dipisahkan oleh dinding otot tebal disebut *septum*.

2. Klep Jantung

Jantung memiliki 4 klep yang berfungsi menentukan kearah mana darah mengalir, berapa besar alirannya, dan kapan menghentikan aliran tersebut, adapun masing-masing klep adalah

a. Klep *tricuspid*. Bila membuka membiarkan darah mengalir dari bilik kanan pada waktu jantung relaksasi dan menutup untuk mencegah darah mengalir kembali ke tubuh bila jantung kontraksi.
b. Klep *mitral*. Yang mengkontrol darah mengalir antara serambi kiri dan bilik kiri.
c. Klep *Pulmic*. Mengontrol aliran darah dari bilik kanan ke *arteri* yang menyuplai darah ke paru-paru.
d. Klep *aorta*, yang memisahkan bilik kiri dengan *aorta*. Bilamana terbuka membiarkan darah mengalir ke tubuh yaitu sewaktu jantung berkontraksi, dan tertutup bila jantung relaksasi.

Jadi klep-klep jantung membuka dan menutup aliran darah dalam rongga jantung, agar mengalir ke satu arah mencegah terjadinya arus balik.

3. Sistem Listrik

Sistem elektrik terdiri dari dua *node* elektris yang terletak di dalam jantung. Yang pertama, disebut *sinoatrial node* (SA), yang terdapat di bagian atas serambi kanan. Yang kedua, disebut *atrioventrical node* (AV). *Node-node* ini memancarkan aliran listrik pada waktu yang tepat, sehingga denyut jantung (irama ekspansi dan kontraksi) terkoordinasi. Sistem saraf mengendalikan bekerjanya sistem listrik tersebut yang dapat mempercepat atau memperlambat laju denyut jantung, tergantung pada sejauh mana pusat saraf tersebut tegang atau rileks [6].

2.1.1 Detak Jantung

Detak jantung merupakan hal yang sangat penting, karena apabila ada rangsangan pada suatu bagian tubuh, secepat kilat ia sampaikan ke segenap organ tubuh. Dengan demikian, arus listrik sebagai pembuka jalan, akan segera menimbulkan konstraksi yang disebut dengan denyut jantung. Hal ini jalan terus dengan irama yang teratur tanpa berhenti, menurut kecepatan yang disebut di atas tadi pada umumnya 70 kali dalam satu menit.

Tetapi jantung sebagai pompa, mempunyai empat ruangan sendiri, yang masing-masingnya mempunyai peranan penting. Karena darah itu dipompakan bukan hanya kepada satu peredaran, melainkan kepada dua peredaran yang sama sekali berbeda. Yang besar adalah peredaran umum, mengalir ke seluruh bagian tubuh. Tetapi setelah tiba di ujung perjalanannya, darah itu kembali lagi ke sumber semula. Perjalanan kembali ini lebih pendek, dan melintasi paru-paru yang mengubah komponen darah itu.

Demikian cara kerjanya seluruh sistem tersebut. Darah dikirimkan ke *antrium* di sebelah kanan melalui pembuluh-pembuluh utama yang disebut *vena* (*vena cava*). Ini adalah darah yang telah dikumpulakan dari seluruh bagian tubuh pada saat itu, lalu dilimpahkan ke *atrium* (serambi).

Pada denyutan jantung yang berikutnya, jantung itu berkontaksi, lalu darah yang ada di *atrium* dilimpahkan ke *ventricle* di sebelah kanan ketika katup penghubung itu terbuka. Kemudian pada saat jantung itu istrahat diantara denyutannya itu, tertutuplah katupnya pada saat dimana *atrium* diisi dengan darah yang datang dengan cara yang sama.

Pada kontraksi berikutnya ketika *ventricle* sebelah kanan mengempis, katupnya terbuka, lalu memancarkan darah ke pembuluh darah besar yang menyalurkannya ke jantung.

Disini darah itu disebarkan melalui jaringan besar kemudian melalui jaringan yang halus sampai ke jaringan semakin halus, tibalah pada pembuluh darah yang sangat halus apabila tidak disini masing-masing sel darah bersentuhan dengan gelembung udara yang dihirup ke dalam paru-paru [7].

2.1.2 Perhitungan Detak Jantung

Denyut atau detak merupakan pemeriksaan pada pembuluh nadi atau *arteri*. Ukuran kecepatannya diukur pada beberapa titik denyut misalnya denyut *arteri radialis* pada pergelangan tangan, *arteri brachialis* pada lengan atas, *arteri karotis* pada leher, *arteri poplitea* pada belakang lutut, *arteri dorsalis pedis* atau *arteri tibialis posterior* pada kaki. Pemeriksaan denyut dapat dilakukan dengan bantuan *stetoskop*.

Secara umum denyut nadi maksimum orang sehat saat berolah raga adalah 80% x (220-usia) untuk kebutuhan *fitness*. Lebih akurat, rumusan perhitungan denyut nadi maksimum 210-(0,5 x umur)-(0,05 x berat badan (dalam *pound*))+4 untuk pria, sedangkan untuk wanita adalah 210-(0,5 x umur)-(0,05 x berat badan (dalam *pound*)). Catatan: 1 kg = 2,2 *pound*.

Dalam olahraga, diberikan 3 (tiga) tingkatan kebutuhan yaitu :

Untuk sehat : 50-70% denyut nadi maksimum

Untuk kebugaran (*fitness*) : 70-80% denyut nadi maksimum

Untuk atlit (*performance*) : 80-100% denyut nadi maksimum.

Denyut jantung yang normal yakni 60-100 kali setiap menit, sedang denyut jantung lambat kurang dari 60 kali per menit dan yang cepat lebih dari 100

kali per menit. Nadi adalah denyut nadi yang teraba pada dinding pembuluh darah arteri yang berdasarkan *systol* dan *gystole* dari jantung.

Jumlah denyut nadi yang normal berdasarkan usia seseorang adalah:

1. Bayi baru lahir :140 kali per menit
2. Umur di bawah umur 1 bulan :110 kali per menit
3. Umur 1 - 6 bulan :130 kali per menit
4. Umur 6 - 12 bulan :115 kali per menit
5. Umur 1 - 2 tahun :110 kali per menit
6. Umur 2 - 6 tahun :105 kali per menit
7. Umur 6 - 10 tahun : 95 kali per menit
8. Umur 10 - 14 tahun : 85 kali per menit
9. Umur 14 - 18 tahun : 82 kali per menit
10. Umur di atas 18 tahun : 60 - 100 kali per menit
11. Usia Lanjut : 60 -70 kali per menit

Mengetahui denyut nadi merupakan dasar untuk melakukan latihan fisik yang benar dan terukur. Dari denyut nadi, dapat diketahui intensitas atau seberapa keras seseorang melakukan latihan. Atau seberapa keras jantungnya bekerja. Secara umum, yang perlu diperhatikan dalam olahraga adalah frekuensi dan intensitas. Frekuensi adalah berapa kali seminggu seseorang melakukan olahraga. Sedangkan intensitas dilihat dari denyut nadi.

Sebenarnya ada banyak cara untuk mengukur denyut nadi. Salah satu metode yang dianggap efektif untuk menentukan denyut nadi adalah *Formula Karvonen*. Menurut metode ini, denyut nadi dapat diukur melalui pembuluh *arteri radialis* yang ada di pergelangan tangan atau pembuluh *arteri carotis* yang ada di leher. Tetapi, yang umum digunakan adalah melalui pergelangan tangan.

Pertama-tama yang perlu diketahui adalah denyut nadi normal. Hasilnya dapat diperoleh dengan menghitung denyut nadi saat bangun pagi, sebelum melakukan aktivitas apapun. Hasil ini juga sering disebut denyut nadi istirahat (*resting heart rate*). Agar diperoleh hasil yang akurat, Sebaiknya pengukuran dilakukan tiga hari berturut-turut. Kemudian, ambil rata-ratanya.

Jika denyutnya di atas 100, berarti ada sesuatu yang tidak beres. Mungkin sedang demam, sakit tenggorokan, akan haid, atau ada masalah lain. Sebelum berolahraga, Sebaiknya juga mengukur denyut nadi. Bila di atas 100, Sebaiknya tidak berolahraga dulu. Kemudian, di tengah latihan inti, diukur lagi. Jika melebihi zona latihan, sebaiknya mengurangi intensitas latihan. Misalnya, mengurangi kecepatan lari. Sedangkan bila denyutnya di bawah zona latihan, berarti latihan yang dilakukan sia-sia. Namun, pengukuran saat latihan ini hanya dapat dilakukan pada saat berolahraga.

Cara menghitung denyut nadi seseorang adalah dengan cara meletakkan jari pada pergelangan tangan (jangan menggunakan ibu jari), atau dapat juga meraba daerah leher disamping tenggorokan, atau dapat juga dengan secara langsung menempelkan telinga pada dada orang yang akan diperiksa untuk mendengar detak jantungnya.

Denyut nadi pada orang yang sedang berisitirahat adalah sekita 60 – 80 permenit untuk orang dewasa, 80 – 100 permenit untuk anak-anak, dan 100 – 140 permenit pada bayi. Namun denyut nadi bisa lebih cepat jika seseorang dalam keadaan ketakutan, habis berolah raga, atau sakit panas. Umumnya denyut nadi akan meningkat sekitar 20 kali permenit untuk setiap satu derajat *celcius* penderita sakit panas. Sebagai catatan, denyut nadi yang terlalu cepat, terlalu lambat, atau tidak beraturan dapat berarti gangguan pada jantung. Adapun cara perhitungannya adalah sebagai berikut:

Tempel dan tekankan (Jangan terlalu keras) tiga jari (telunjuk, tengah, manis) salah satu tangan pada pergelangan tangan yang lain. Temukan denyut nadi. Setelah itu, barulah mulai menghitung. Hitunglah denyut nadi selama 15 detik. Kemudian, hasilnya dikalikan 4 atau bisa juga dengan perhitungan selama 30 detik, kemudian hasilnya dikalikan 2 untuk menghasilkan *beat* per menit.

Denyut nadi normal : 60 - 100/menit

Denyut nadi maksimal : 220/menit

atau untuk mengetahui *beat* per menit (bpm) dapat diketahui dengan menggunkan rumus BPM yaitu : $\text{BPM} = \frac{60}{waktu\ perhitungan} * jumlah\ detak$ ..... (2.1)

Umur zona latihan (*training zone*; yaitu tingkat intensitas dimana anda bisa berolahraga): 70% - 85% dari denyut nadi maksimal [7].

2.1.3 Pengertian *Plethysmografi* dan *Photoplethysmograph*

Plethysmografi merupakan suatu teknik untuk mendeteksi/mengukur perubahan *volume* di dalam suatu organ. *Plethysmograf* merupakan suatu instrumen yang digunakan untuk mengukur perubahan *volume* di dalam suatu organ atau seluruh tubuh. Biasanya merupakan hasil dari fluktuasi darah atau udara yang terkandung didalamnya.

Photoplethysmograph merupakan sebuah metode pengukuran suatu besaran berdasarkan perubahan yang menggunakan sifat-sifat optik. Pemantauan detak jantung dengan menggunkan metode ini akan memanfaatkan sifat-sifat optik dari jaringan tubuh seperti kulit dan darah. Aliran darah adalah sumber informasi yang utama. Ketika melakukan pengukuran dipilih bagian tubuh yang mengandung banyak arteri atau arteriola, salah satunya adalah jari tangan. Metode pengukuran ini dibutuhkan adanya sumber cahaya untuk dapat menembus jari tangan agar dapat melihat perubahan yang ada pada jaringan tubuh dan darah (perubahan warna merah). Caranya adalah sumber cahaya diletakan pada jari tangan, cahaya yang melewati jari tangan akan mengalami perubahan, perubahan cahaya tersebut sangat dipengaruhi oleh perubahan *volume* darah [1].

2.2 Pengolahan Citra

Data atau informasi tidak hanya disajikan dalam bentuk teks, tetapi juga dapat berupa gambar, audio (bunyi, suara, musik), dan video. Keempat macam data atau informasi ini sering disebut multimedia. Era teknologi informasi saat ini tidak dapat dipisahkan dari multimedia. Situs web (*website*) di internet dibuat semenarik mungkin dengan menyertakan visualisasi berupa gambar atau video yang dapat diputar. Beberapa waktu lalu istilah SMS (*Short Message Service*) begitu populer bagi pengguna telepon genggam (*handphone* atau HP). Tetapi, saat ini orang tidak hanya dapat mengirim pesan dalam bentuk teks, tetapi juga dapat

mengirim pesan berupa gambar maupun video, yang dikenal dengan layanan MMS (*Multimedia Message Service*).

Citra (*image*) sebagai salah satu komponen multimedia memegang peranan sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Ada sebuah peribahasa yang berbunyi "sebuah gambar bermakna lebih dari seribu kata" (*a picture is more than a thousand words*). Maksudnya tentu sebuah gambar dapat memberikan informasi yang lebih banyak daripada informasi tersebut disajikan dalam bentuk kata-kata (tekstual) [8].

2.2.1 Pengertian Pengolahan Citra

Meskipun sebuah citra kaya informasi, namun seringkali citra yang kita miliki mengalami penurunan mutu (degradasi), misalnya mengandung cacat atau derau (*noise*), warnanya terlalu kontras, kurang tajam, kabur (*blurring*), dan sebagainya. Tentu saja citra semacam ini menjadi lebih sulit diinterpretasi karena informasi yang disampaikan oleh citra tersebut menjadi berkurang.

Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin), maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Bidang studi yang menyangkut hal ini adalah pengolahan citra (*image processing*) [8].

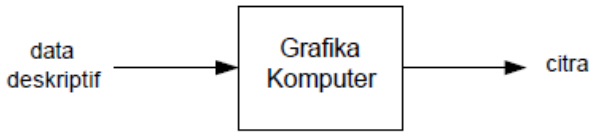
Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Umumnya, operasi-operasi pada pengolahan citra diterapkan pada citra bila:

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokkan, dicocokkan, atau diukur.
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

Di dalam bidang komputer, ada tiga bidang studi yang berkaitan dengan data citra, namun tujuan ketiganya berbeda, yaitu:

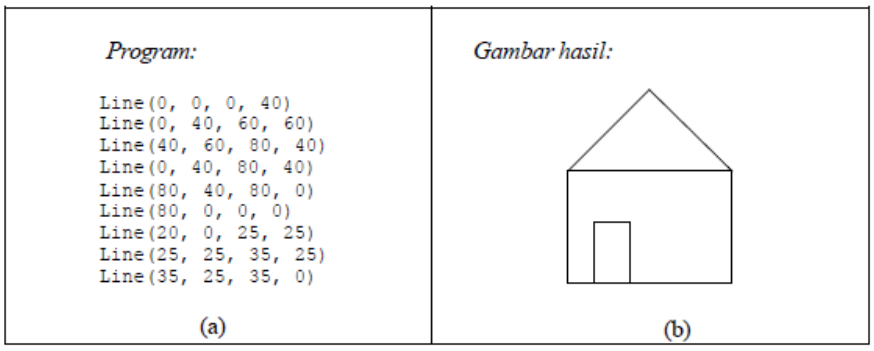
1. Grafika Komputer (*computer graphics*).

Grafika Komputer bertujuan menghasilkan citra (lebih tepat disebut grafik atau *picture*) dengan primitif-primitif geometri seperti garis, lingkaran, dan sebagainya. Primitif-primitif geometri tersebut memerlukan data deskriptif untuk melukis elemen-elemen gambar. Contoh data deskriptif adalah koordinat titik, panjang garis, jari-jari lingkaran, tebal garis, warna, dan sebagainya. Grafika komputer memainkan peranan penting dalam visualisasi dan *virtual reality*.



Gambar 2.1 Alur Grafika Komputer

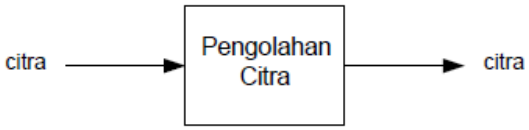
Contoh grafika komputer misalnya menggambar sebuah rumah yang dibentuk oleh garis-garis lurus, dengan data masukan berupa koordinat awal dan koordinat ujung garis, seperti pada gambar 2.2.



Gambar 2.2 Contoh Program Grafika Komputer

2. Pengolahan Citra (*image processing*).

   Pengolahan Citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik daripada citra masukan. Termasuk ke dalam bidang ini juga adalah pemampatan citra (*image compression*).

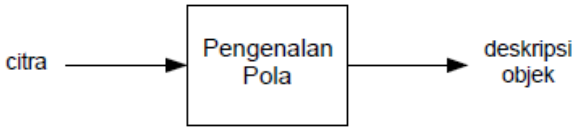


Gambar 2.3 Alur Pengolahan Citra

3. Pengenalan Pola (*pattern recognition/image interpretation*).

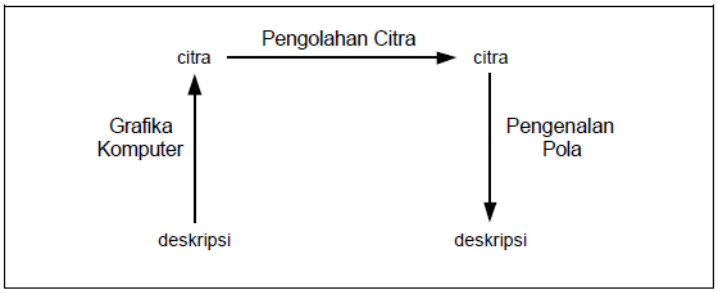
   Pengenalan Pola mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer). Tujuan pengelompokan adalah untuk mengenali suatu objek di dalam citra. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek di alam sehingga mampu membedakan suatu objek dengan objek lainnya.

   Kemampuan sistem visual manusia inilah yang dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut, dan memberikan keluaran berupa deskripsi objek di dalam citra.



Gambar 2.4 Alur Pengenalan Pola

Hubungan antara ketiga bidang (grafika komputer, pengolahan citra, pengenalan pola) ditunjukkan seperti gambar 2.5.



Gambar 2.5 Hubungan Grafika Komputer, Pengolahan Citra dan Pengenalan Pola

2.2.2 *Threshold*

Thresholding merupakan salah satu teknik segmentasi yang baik digunakan untuk citra dengan perbedaan nilai intensitas yang signifikan antara latar belakang dan objek utama. Dalam pelaksanaannya *thresholding* membutuhkan suatu nilai yang digunakan sebagai nilai pembatas antara objek utama dengan latar belakang, dan nilai tersebut dinamakan dengan *threshold*.

Thresholding digunakan untuk mempartisi citra dengan mengatur nilai intensitas semua *pixel* yang lebih besar dari nilai *threshold* T sebagai latar depan dan yang lebih kecil dari nilai *threshold* T sebagai latar belakang. Biasanya pengaturan nilai *threshold* dilakukan berdasarkan histogram *grayscale* [9].

Nilai T dapat ditentukan dengan salah satu dari 3 cara berikut:

1. Nilai Ambang Global (*Global Threshold*)

   T = T{f(x,y)}

   dengan T tergantung pada nilai *gray* level dari *pixel* pada posisi x,y.

2. Nilai Ambang Lokal (*Local Threshold*)

   T = T{A(x,y), f(x,y)}

   dengan T tergantung pada properti *pixel* tetangga. A(x,y) menyatakan nilai *pixel* tetangga.

3. Nilai Ambang dinamis (Dynamic Threshold)

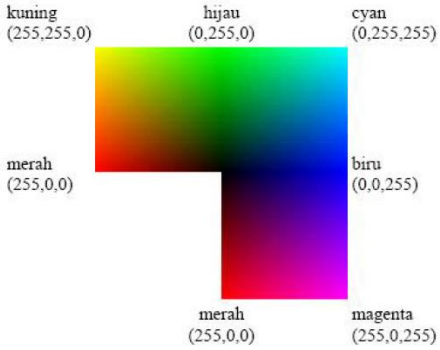
   T = T{x,y, A(x,y), f(x,y)}

   dengan T tergantung pada koordinat-koordinat *pixel*.

2.2.3 Warna RGB

RGB adalah suatu model warna yang terdiri dari merah, hijau, dan biru, digabungkan dalam membentuk suatu susunan warna yang luas. Setiap warna dasar, misalnya merah, dapat diberi rentang nilai. Untuk monitor komputer, nilai rentangnya paling kecil = 0 dan paling besar = 255. Pilihan skala 256 ini didasarkan pada cara mengungkap 8 digit bilangan *biner* yang digunakan oleh mesin komputer. Dengan cara ini, akan diperoleh warna campuran sebanyak 256 x 256 x 256 = 1677726 jenis warna.

Sebuah jenis warna, dapat dibayangkan sebagai sebuah vektor di ruang dimensi 3 yang biasanya dipakai dalam matematika, koordinatnya dinyatakan dalam bentuk tiga bilangan, yaitu komponen-x, komponen-y dan komponen-z. Misalkan sebuah vektor dituliskan sebagai r = (x,y,z). Untuk warna, komponen-komponen tersebut digantikan oleh komponen R(ed), G(reen), B(lue). Jadi, sebuah jenis warna dapat dituliskan sebagai berikut: warna = RGB(30, 75, 255). Putih = RGB (255,255,255), sedangkan untuk hitam= RGB(0,0,0). Bentuk representasi warna dari sebuah citra digitial dapat dilihat pada gambar 2.6.



Gambar 2.6 Representasi Warna RGB Pada Citra Digital

Misalnya terdapat gambar berukuran 100 pixel x 100 *pixel* dengan *color encoding* 24 bit dengan R = 8 bit, G = 8 bit, B = 8 bit, maka *color encoding* akan mampu mewakili 0 ... 16.777.215 (mewakili 16 juta warna), dan ruang *disk* yang dibutuhkan = 100 x 100 x 3 bit (karena RGB) = 30.000 bit = 30 KB atau 100 x 100 x 24 bit = 240.000 bit [10].

2.2.4 *Frame Rate* dan *Frame Size*

Ketika serangkaian gambar mati yang bersambung dimainkan dengan cepat dan dilihat oleh mata manusia, maka gambar-gambar tersebut akan terlihat seperti sebuah pergerakan yang halus. Jumlah gambar yang terlihat setiap detik disebut dengan *frame rate*. Diperlukan *frame rate* minimal sebesar 10 fps (*frame per second*) untuk menghasilkan pergerakan gambar yang halus. Film-film yang dilihat di gedung bioskop adalah film yang diproyeksikan dengan *frame rate* sebesar 24 fps, sedangkan video yang dilihat pada televisi memiliki *frame rate* sebesar 30 fps (tepatnya 29.97 fps). *Frame rate* digunakan sebagai format standar NTSC, PAL dan SECAM yang berlaku pada negara-negara didunia.

Lebar dan tinggi *frame* video disebut dengan f*rame size*, yang menggunakan satuan *pixel*, misalnya video dengan ukuran *frame* 640×480 *pixel*. Dalam dunia video digital, *frame size* disebut juga dengan resolusi. Semakin tinggi resolusi gambar maka semakin besar pula informasi yang dimuat, berarti akan semakin besar pula kebutuhan memori untuk membaca informasi tersebut. Misalnya untuk format PAL D1/DV berukuran 720×576 *pixel*, format NTSC DV 720×480 *pixel* dan format PAL VCD/VHS (MPEG-1) berukuran 352×288 *pixel* sedangkan format NTSC VCD berukuran 320×240 *pixel* [11].

2.2.5 Format Kamera *Surface Preview*

Surface preview merupakan sebuah proses untuk melakukan *viewfinder* atau pratinjau atas tampilan dari objek yang akan diambil gambarnya, juga dapat berfungsi sebagai monitor dari sebuah citra harus diambil, diubah dan dihapus.

Hasil dari proses *surface preview* adalah citra dari setiap *frame* yang menghasilkan format citra YUV, namun Android 2.2 adalah versi pertama yang memiliki *decoder* YUV format (*YuvImage*), dalam versi sebelumnya, *decoding* harus dilakukan dengan tangan [12].

2.2.6 Konversi YUV ke RGB

YUV dikenal juga sebagai Y'CbCr dan YpbPr adalah sebuah *color space* dengan Y merupakan komponen *luminance* (*brightness*) dan U dan V adalah komponen *chrominance* (warna). YUV biasanya digunakan pada aplikasi video. Sinyal YUV diciptakan dari sumber RGB (*red, green, blue*), dengan diberi bobot tertentu, nilai R, G dan B ditambahkan untuk menghasilkan sinyal Y, yang menyatakan terang-gelap secara keseluruhan, atau *luminance* dari titik tersebut. Sinyal U lalu dibentuk dengan mengurangkan sinyal *blue* dari RGB dengan Y, dan V dengan mengurangkan *red*.

Untuk mengkonversi dari YUV ke RGB adalah sebagai berikut :

R = 1,164(Y - 16) + 1,596(V - 128).....(2.2)

B = 1,164(Y - 16) + 2,018(U - 128).....(2.3)

G = 1,164(Y - 16) – 0,813(V - 128) – 0,391(U - 128).....(2.4)

Perhatikan bahwa 8-bit YCbCr dan data R'G'B ' harus berada di 0 dan 255 tingkat untuk menghindari *underflow* dan masalah kelebihan *wrap-around* [13].

2.3 Metode *Moving Average Filter*

Metode *Moving Average Filter* (filter rata-rata bergerak) adalah filter yang paling umum digunakan, terutama karena digital *filter* ini paling mudah untuk digunakan dan dipahami karena memiliki kesederhanaan dalam penggunaannya, teknik ini digunakan untuk mengambil nilai rata-rata beberapa sampel yang berdekatan dan hasil yang diperoleh adalah kurang lebih adalah nilai aslinya. Jadi hasil dari *Moving Average Filter* tersebut "Meredam" *noise* yang terjadi, selain itu *filter* ini juga meningkatkan sedikit kinerja komputasi dalam persoalan waktunya. Berikut adalah persamaan *Moving Average Filter*:

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j] \quad (2.5)$$

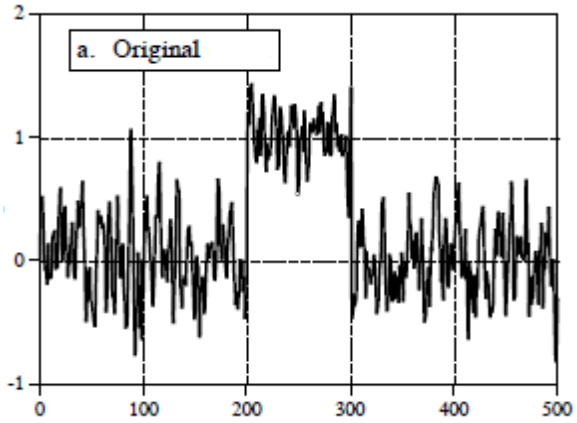
Persamaan *filter* rata-rata bergerak. Dalam Persamaan ini, x[ ] adalah nilai *input*, y[ ] nilai *output*, dan M adalah jumlah *point* yang digunakan dalam rata-rata

bergerak. Ini persamaan hanya menggunakan *point* pada satu sisi contoh *output* sedang dihitung.

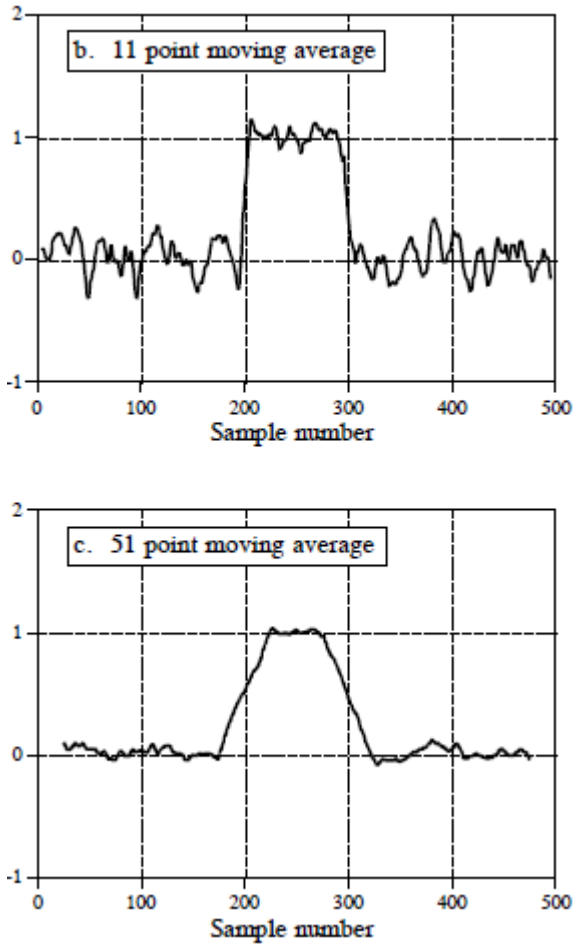
Dimana x[ ] adalah nalai *input*, y[ ] adalah nilai *output* dan M banyaknya angka dalam *average*. Contoh menggunakan 5 *point moving average filter*.

$$y[80] = \frac{x[78] + x[79] + x[80] + x[81] + x[82]}{5}$$

Berikut adalah contoh penggunaan dari *Moving Average Filter* pada gambar 2.7 dan 2.8.



Gambar 2.7 Pergerakan Original



Gambar 2.8 Setelah dilakukan *Moving Average Filter*

Contoh *moving average filter* diatas. Pada (b) dan (c), nilai disaring dengan 11 dan 51 *point moving average filter* masing-masing. Dimana jika jumlah *point filter* meningkat maka *noise* menjadi lebih rendah, dan tepi menjadi kurang tajam. Sehingga berdasarkan contoh diatas dapat disimpulkan bahwa *Moving average filter* dapat menghasilkan solusi yang optimal bila memiliki nilai *noise* yang rendah dan *filter* yang sesuai dapat menghasilkan tepi yang tajam.

Jumlah pengurangan *noise* sama dengan akar kuadrat dari jumlah *point* dalam rata-rata. Sebagai contoh 100 titik rata-rata bergerak *filter* mengurangi kebisingan dengan faktor 10 [14].

2.4 Pengenalan Android

2.4.1 Sejarah Android

Android adalah sebuah sistem operasi untuk peranggkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka.

Perilisan pertama android pada tanggal 5 November 2007, android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, *Google* merilis kode-kode android di bawah lisensi *apache*, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler.

Di dunia ini terdapat dua jenis distributor sistem operasi android. Pertama yang mendapat dukungan penuh dari *Google* atau *Google Mail Services* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung *google* atau dikenal sebagai *Open Handset Distribution* (OHD).

Sekitar September 2007 *Google* mengenalkan *Nexus one*, salah satu jenis *smartphone* yang menggunakan android sebagai sistem operasinya. Telepon seluler ini diproduksi oleh HTC *Corporation* dan tersedia di pasaran pada 5 Januari 2010. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android ARM Holdings, Atheros Communications, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan *Open Handset Alliance*, OHA mengumumkan produk perdana mereka, android perangkat *mobile* yang merupakan memodifikasi kernel linux 2.6. Sejak android dirilis telah dilakukan berbagai pembaruan berupa perbaikan *bug* dan penambahan fitur baru.

Tidak hanya menjadi sistem operasi di s*martphone*, saat ini android menjadi pesaing utama dari apple pada sistem operasi *Table* PC. Pesatnya pertumbuhan android selain faktor yang disebutkan di atas adalah karena android itu sendiri adalah *platform* yang sangat lengkap baik itu sistem operasinya, aplikasi dan *tool* pengembangan, *market* aplikasi android serta dukungan yang sangat tinggi dari komunitas *open source* di dunia, sehingga android terus

berkembangan pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia [3].

2.4.2 Arsitektur Android

Secara garis besar arsitektur android dapat dijelaskan dan digambarkan sebagai berikut [3] :

1. *Applications* dan *Widgets*

Applications dan *Widgets* ini adalah *layer* di mana kita berhubungan dengan aplikasi saja, di mana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *Applications Frameworks*

Android adalah "*Open Develoment Platform*" yaitu android menawarkan kepada pengembang untuk membangaun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, sistem informasi *resources*, menjalankan *service background*, mengatur *alarm*, dan menambahkan status *notifications*, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang kategori ini. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunkan (*reuse)*.

Sehingga bisa kita simpulkan *application frameworks* inti adalah *layer* di mana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat, seperti *content providers* yang berupa sms dan panggilan telepon.

Komponen-komponen yang termasuk di dalam *applications frameworks* adalah sebagai berikut:

a. *Views*
b. *Contenet Provider*
c. *Resource Manager*

d. *Notification Manager*
e. *Activity Manager*

3. *Libraries*

Libraries ini adalah *layer* di mana fitur-fitur android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di atas kernel, *layer* ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL, serta:

a. *Libraries* media untuk pemutaran media audio dan video
b. *Libraries* untuk manajemen tampilan
c. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
d. *Libraries SQLite* untuk dukungan *database*
e. *Libraries* SSL dan *WebKit* terintegrasi dengan *web browser* dan *security*
f. *Libraries LiveWebcore* mencakup modern *web browser* dengan *engine embeded web view*
g. *Libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API's

4. *Android Run Time*

Layer yang membuat aplikasi android dapat dijalankan di mana dalam prosesnya menggunakan implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi android. Di mana android *run time* dibagi menjadi dua bagian yaitu:

a. *Core Libraries*: aplikasi android dibangun dalam bahasa java sementara dalvik sebagai virtual mesinnya bukan *Virtual Machine Java*, sehingga diperluakan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa java/c yang ditangani oleh *core libraries*.
b. *Dalvik Virtual Machine*: Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, di mana merupakan pengembangan yang mampu membuat linux kernel untuk melakukan *treading* dan manajemen tingkat rendah.

5. *Linux Kernel*

Linux kernel adalah *layer* di mana inti dari *operating* sistem dari android itu berada. Berisi *file-file system* yang mengatur sistem *processing, memory*,

resource, drivers, dan sistem-sistem operasi android lainnya. Linux *kernel* yang digunakan android adalah linu kernel *release* 2.6.

2.4.3 *Heap* Dan *Stack*

Adapun pengertian dari *heap* dan *stack* adalah sebagai berikut [15]:

1. *Heap* (tempat *instance variable* dan *object*)

Heap merupakan tumpukan (di Java) tempat menyimpan variabel yang dialokasikan secara dinamis seperti objek. Saat tumpukan berjalan rendah pada memori, JVM akan menjalankan pengumpulan sampah. Pengumpulan sampah menggunakan siklus pengolahan yang akan memperlambat telepon. Untuk meningkatkan ukuran *heap* maksimum yaitu dengan menggunakan aplikasi yang sangat dekat dengan semua ruang tumpukan yang tersedia. Dimana aplikasi tersebut akan memaksa untuk menjalankan pengumpulan sampah secara sering, yang akan digunakan *CPU*. Ada kemungkinan bahwa menurunkan ukuran tumpukan maksimum dapat bermanfaat mencegah aplikasi untuk mendapatkan memori lebih dari yang dibutuhkan (dengan memaksa untuk mengumpulkan sampah lebih cepat), tapi itu semua tergantung pada bagaimana *VM Dalvik* dilaksanakannya.

2. *Stack* (tempat *local variable* dan tumpukan *method*)

Stack dicadangkan untuk penyimpanan data *CPU* (biasanya, isi dari *register*) sebelum beralih konteks ke yang lain (atau benang, yang berarti memuat satu set data yang berbeda dalam *register*). Zona memori dimana data dari aplikasi *user* (*Dalvik VM* adalah salah satu dari mereka, meskipun aplikasi sedang menjalankan aplikasi lain) disimpan disebut tumpukan.

2.5 *Garbage Collection*

Garbage collection adalah salah satu fitur canggih milik bahasa pemrograman java, yang berguna untuk mengumpulkan memori setelah digunakan. Banyak contoh kasus yang menyebabkan seorang *programmer* itu menjadi handal atau dia menjadi seorang *programmer* yang gagal hanya karena managemen memori pada program yang dibuat. Misalnya, pada bahasa C, kita

mengenal *pointer* yang bisa digunakan untuk pengalamatan variabel dan juga sebagai variabel dinamis. Apabila *programmer* handal dalam mengolah memori, maka program yang dibuat pun akan menjadi sangat handal. Namun sebaliknya, *pointer* bisa menjadi bumerang bagi *programmer* apabila dia salah dalam mengatur memori yang dimiliki.

Dalam bahasa pemrograman java, *pointer* dikenal sebagai objek yang merupakan *instance* dari suatu *class* dan memiliki *reference* terhadap *class* tersebut. Dengan adanya *garbage collection* ini, maka proses untuk pengumpulan memori dapat dilakukan secara otomatis oleh bahasa java. Sehingga, akan lebih memudahkan *programmer*. Disamping program akan menjadi lebih handal, *programmer* juga akan lebih mudah dalam mengatur penggunaan memorinya [15].

2.6 *Eclipse*

2.6.1 Pengenalan *Eclipse*

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform-independent*). Berikut ini adalah sifat dari *Eclipse*:

1. *Multi-platform*: target sistem operasi *eclipse* adalah *Microsoft Windows*, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. *Mulit-language*: *eclipse* dikembangkan dengan bahasa pemrograman java, akan tetapi *eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
3. *Multi-role*: selain sebagai IDE untuk pengembangan aplikasi, *eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *Eclipse* yang membuatnya populer

adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2.6.2 Sejarah *Eclipse*

Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak IBM *Visual Age for Java* 4.0. Produk ini diluncurkan oleh IBM pada tanggal 5 November 2001, yang menginvestasikan sebanyak US$ 40 juta untuk pengembangannya. Semenjak itu konsursium *Eclipse Foundation* mengambil alih untuk pengembangan *Eclipse* lebih lanjut dan pengaturan organisasinya.

2.6.3 Arsitektur *Eclipse*

Sejak versi 3.0, *eclipse* pada dasarnya merupakan sebuah kernel, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam *eclipse* sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari *eclipse* yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP:

1. *Core platform*
2. *OSGi*
3. *SWT (Standard Widget Toolkit)*
4. *JFace*
5. *Eclipse Workbench*

Secara standar *eclipse* selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat *eclipse* kompatibel untuk mengembangkan program java, dan PDE *(Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. *Eclipse* beserta *plug-in* nya diimplementasikan dalam bahasa pemrograman java.

Konsep *eclipse* adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, *Eclipse* tidak saja untuk mengembangkan program java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT

(C/C++ *Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh *eclipse*, *plug-in* UML 2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya. Salah satu situs yang menawarkan *plug-in* secara gratis [16].

2.7 Pengenalan Pemrograman Berorientasi Objek

2.7.1 Pengertian Pemrograman Berorientasi Objek

Metodologi bereorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut [17]:

1. Meningkatkan Produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan Pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengodean.
3. Kemudahan Pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya Konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan Kualitas Perangkat Lunak

   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.7.2 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap anaslisis, perancangan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodelogi berorientasi objek [17]:

1. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau terciptakan dan kelas tersebut. Sebuah kelas akan mempunya sifat (atribut), kelakuan (oprasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain

yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat dipengaruhi pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (*method*)

   Operasi atau metode atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4. Atribut (*attribut*)

   Artibut dari sebuah kelas adalah variabel global yang memiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan (*inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*interface*)

   Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas atau objek yang umum dengan kelas objek yang khusus. Misalnya kelas lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antarobjek

    Komunikasi antarobjek dilakuakan lewat pesan (*message*) yang dikirim dan satu objek ke objek lainnya.

12. Polimorfisme (*polymorphism*)

    Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. *Package*

    Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.8 Pengenalan UML

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standardisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebauh ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

Banyak orang yang membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan oleh banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan

pemrograman prosedural atau struktural. Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML) [18].

2.8.1 Sejarah UML

Bahasa pemrograman berorientasi objek pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada *developer* pengembang bahasa pemrograman berorientasi objek selanjutnya. Karena banyaknya metodologi-metodologi yang berkembang pesat pada saat itu, maka munculah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi yang berkembang pada saat itu [18].

2.8.2 Diagram UML

Pada UML 2.3 terdapat 13 macam diagramyang dikelompokan dalam 3 kategori. Berikut ini penjelasan singkat dari pembagian kategori tersebut [18].

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun antar subsistem pada suatu sistem.

2.8.2.1 *Class Diagram*

Diagram kelas atau *Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebagai jenis-jenis kelas berikut [18]:

1. Kelas main
   Kelas yang memiliki fungsi awal dieksekusi ketika sistem berjalan.
2. Kelas yang menangani tampilan sistem
   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai
3. Kelas yang diambil dari pendefinisian *use case*
   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case.*
4. Kelas yang diambil dari pendefinisian data
   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

2.8.2.2 *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakukan (*behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antar satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang akan ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case* [18].

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat

itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor.

2.8.2.3 *Use Case Scenario*

Setiap *use case* harus dijelaskan alur prosesnya melalui sebuah deskripsi *use case* (*use case description*) atau *scenario use case* [18].

Deskripsi *use case* berisi:

1. Nama *use case* yaitu penamaan *use case* yang menggunakan kata kerja
2. Deskripsi yaitu penjelasan mengenai tujuan *use case* dan nilai yang akan didapatkan oleh actor.
3. Kondisi sebelum (*pre-condition*) yaitu kondisi-kondisi yang perlu ada sebelum *use case* dilakukan.
4. Kondisi sesudah (*post-condition*) yaitu kondisi-kondisi yang sudah dipenuhi ketika *use case* sudah dilaksanakan.
5. Alur dasar (*basic flow*) yaitu alur yang menceritakan jika semua aksi yang dilakukan adalah benar atau proses yang harusnya terjadi
6. Alur alternatif (*alternatif flow*) yaitu alur yang menceritakan aksi alternatif, yang berbeda dari alur dasar.

2.8.2.4 *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut [18]:

1. Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* di mana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.

3. Rancangan pengujian di mana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

2.8.2.5 *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

Banyak diagram sekuen harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang akan didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak [18].

2.9 *Adobe Photoshop CS 5*

Adobe Photoshop adalah salah satu *software* untuk mengolah foto ataupun gambar yang sangat popular saat ini. Dengan *Adobe Photoshop* dapat memperbaiki dan mempercantik foto atau gambar yang diinginkan dengan menambah beberapa efek dalam foto tersebut, sehingga foto akan menjadi sebuah foto dengan tampilan yang beda dan lebih menarik [19].