BAB II
LANDASAN TEORI

2.1. Citra

Indrawati (2013), Citra atau bisa disebut dengan gambar merupakan salah satu komponen dari multimedia yang memegang peranan penting karena mengandung informasi dalam bentuk visual. Citra memiliki lebih banyak informasi yang dapat disampaikan dibandingkan dalam bentuk teks. Sedangkan menurut Fauzi (2007), Citra merupakan kumpulan elemen-elemen gambar (*pixel*) yang secara keseluruhan merekam suatu adegan (*scene*) melalui pengindera visual (kamera). Untuk kebutuhan pengolahan dengan bantuan komputer, citra disajikan dalam bentuk diskrit yang disebut citra digital.

Sutoyo et. al. (2009), menyatakan "Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek". Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optic berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan. Perekaman data citra dapat dibagi menjadi dua yaitu:

a. Citra Analog

   Citra analog yaitu terdiri dari sinyal-sinyal elektromagnetik yang tidak dapat dibedakan sehingga pada umumnya tidak dapat ditentukan ukurannya. Citra analog mempunyai fungsi yang kontinu. Hasil perekaman citra analog dapat bersifat optik yakni berupa foto (film foto konvensional) dan bersifat sinyal video seperti gambar pada monitor televisi.

b. Citra Digital

   Citra digital terdiri dari sinyal-sinyal yang dapat dibedakan dan mempunyai fungsi yang tidak kontinu yakni berupa titik-titik warna pembentuk citra. Hasil perekaman citra digital dapat disimpan pada suatu media penyimpanan.

Citra merupakan istilah lain dari gambar yang merupakan komponen multimedia yang memegang peranan sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu kaya akan informasi. Citra digital adalah citra hasil digitalisasi citra kontinu (analog). Tujuan dibuatnya citra digital adalah agar citra tersebut dapat diolah menggunakan komputer atau piranti digital.

Citra merupakan salah satu komponen multimedia yang memegang peranan penting sebagai bentuk informasi visual, karena karakteristiknya yang kaya dengan informasi. Secara harfiah, citra adalah gambar pada bidang dwimatra (dua dimensi). Secara matematis citra adalah gambar pada bidang dwimatra (dua dimensi) yang dihasilkan dari gambar analog dua dimensi yang kontinus menjadi gambar diskret melalui proses sampling. Gambar analog dibagi menjadi N baris dan M kolom sehingga menjadi gambar diskret. Persilangan antara baris dan kolom tertentu disebut dengan piksel. Contohnya adalah gambar/titik diskret pada baris n dan kolom m disebut dengan piksel [n,m].

2.1.1 Citra Digital

Munir (2004), Citra ada dua macam yaitu citra kontinu dan citra diskrit. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog, contohnya mata manusia, kamera analog. Citra diskrit dihasilkan dariproses digitalisasi terhadap citra kontinu contohnya kamera digital, scanner.

Putra (2009), secara umum pengolahan citra digital menunjuk pada pemrosesan gambar 2 dimensi mengunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada setiap data 2 dimensi. Citra digital merupakan sebuah larik (array) yang berisi nilai-nilai real maupun kompleks yang dipresentasikan dengan deretan bit tertentu.

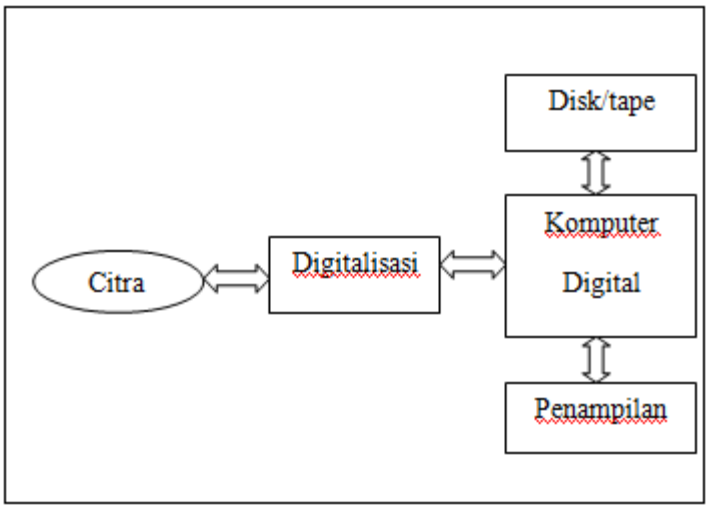
Yudistira (2010), pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Tujuan utama pengolahan citra adalah agar citra yang mengalami gangguan mudah diinterpretasi

oleh manusia maupun mesin (komputer). Teknik pengolahan citra digital adalah mentransformasikan citra dua dimensi menjadi citra lain dengan mengunakan kompter. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Jadi masukannya berupa citra dan keluarannya juga berbentuk citra, dengan kualitas yang lebih baik dari citra masukan. Beberapa contoh operasi pengolahan citra adalah pengubahan kontras citra, penghilangan derau (noise) dengan operasi penapisan (filtering), penajaman (sharpening), pemberian warna semu (pseudocoloring), dan sebagainya. Operasi-operasi tersebut akan diterapkan pada pengolahan citra apabila:

1. Digunakan untuk meningkatkan kualitas penampakan atau menonjolkan beberapa aspek informasi yang terkandung dalam citra dengan perbaikan atau modifikasi citra;
2. Perlu pengelompokkan, pencocokan atau pengukuran elemen pada citra;
3. Perlu penggabungan sebagian citra dengan bagian citra lainnya.

2.1.2 Pengolahan Citra Digital

Berikut ini adalah blok sistem pengolahan citra digital:



Gambar 2.1 Diagram blok sistem pengolahan citra digital

Wijaya dan Prijono (2007), Digitalisasi citra yaitu mengubah citra masukan menjadi sinyal listrik dan kemudian mencuplikan sinyal tersebut dengan menggunakan *A/D Converter (Analog to Digital Converter).* Digitalisasi ini dapat berupa scanneratau kamera digital yang mengubah citra kontinue kedalam suatu

representasi numerik, sehingga citra ini dapat diproses oleh komputer digital. Jadi digitalisassi citra adalah proses mengubah citra analog menjadi citra digital.

Proses pengolahan data dapat dilakukan oleh komputer, baik berupa mikrokomputer sederhana (*microprocessor based computer*) atau komputer besar (*mainframe computer*), tergantung jumlah data dan jenis pengolahan. Proses penampil data merupakan salah satu segi yang penting karena bagaimanapun juga citra digital hasil olahan harus dapat dinilai oleh mata manusia melalui suatu penampil (*display)*. Penampil yang digunakan biasanya berupa suatu *graphic* monitor atau suatu *graphic* printer/plotter.

Hendriyani (2012), *Image processing* adalah suatu teknologi yang berfungsi untuk menyelesaikan masalah mengenai pengolahan citra. Tujuannya yaitu untuk mengolah citra sedemikian rupa, sehingga citra tersebut lebih mudah untuk diproses lebih lanjut atau diedit.

2.1.3 Jenis – jenis Citra Digital

Sutoyo et. al. (2009), menjelaskan jenis-jenis citra digital adalah sebagai berikut:

1. Citra Biner (Monokrom)

   Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai piksel yaitu hitam (0) dan putih (1). Citra biner juga disebut sebagai citra bw (Black and White) atau citra monokrom.

2. Citra *Grayscale* (skala keabuan)

   Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pikselnya. Dengan kata lain nilai bagian red=green=blue. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki citra *grayscale* adalah warna keabuan dengan berbagai tingkatan dari hitam hingga putih. Citra *grayscale* dapat diperoleh dari citra RGB. Nilai intensitas citra *grayscale* (keabuan) dihitung dengan nilai intensitas citra RGB dengan menggunakan persamaan:

   $$\text{Nilai keabuan} = 0.2989 * \text{R} + 0.5870 * \text{G} + 0.1140 * \text{B}$$

   Dimana,

R : Nilai intensitas warna merah

G : Nilai intensitas warna hijau

B : Nilai intensitas warna biru

3. Citra Warna (True Color)

   Citra warna sering disebut juga citra RGB atau citra *true color* karena dapat mempresentasikan warna objek menyerupai warna aslinya dengan mengkombinasikan ketiga warna dasar yaitu merah/*red* (R), hijau/*green* (G), dan biru/*blue* (B). Tiap piksel memiliki tiga nilai kanal yang mewakili tiap komponen dasar citra.

2.1.4 Elemen – elemen Dasar Citra

Sutoyo et. al. (2009), menjelaskan elemen-elemen dasar citra, yaitu:

a. Kecerahan (*Brightness*)

   Yang dimaksud dengan kecerahan (*brightness*) adalah intensitas cahaya yang dipancarkan oleh piksel dari citra yang dapat ditangkap oleh system penglihatan. Kecerahan pada sebuah titik (piksel) didalam citra merupakan intensitas rata – rata dari suatu area yang melingkupinya.

b. Kontras (*contrast*)

   Kontras (*contrast*) menyatakan sebaran terang dan gelap dalam sebuah citra. Pada citra yang baik, komposisi gelap dan terang tersebar secara merata.

c. Kontur (*contour*)

   Yang dimaksud dengan kontur (*contour*) adalah keadaan yang ditimbulkan oleh perubahan intensitas pada piksel – piksel yang bertetangga. Karena adanya perubahan intensitas inilah mata mampu mendeteksi tepi – tepi objek dialam citra.

d. Warna (*color*)

   Warna (*color*) adalah persepsi yang ditangkap sistem visual terhadap perubahan panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna memiliki panjang gelombang yang berbeda-beda. Warna merah memiliki panjang gelombang (λ) yang paling tinggi, sedangkan warna violet mempunyai panjang gelombang (λ) yang paling rendah.

e. Bentuk (*shape*)

Bentuk adalah properti intrinsik dari objek 3 dimensi, dengan pengertian bahwa bentuk merupakan properti intrinsik utama untuk sistem visual manusia.

f. Tekstur (*texture*)

Tekstur (*texture*) dicirikan sebagai distribusi spasial dari derajat keabuan didalam sekumpulan piksel–piksel bertetangga. Tekstur adalah sifat – sifat atau karakteristik yang dimiliki oleh suatu daerah yang cukup besar sehingga secara alami sifat – sifat tadi dapat berulang dalam daerah tersebut. Tekstur adalah keteraturan pola – pola tertentu yang terbentuk dari susunan piksel – piksel dalam citra digital. Informasi tekstur dapat digunakan untuk membedakan sifat– sifat permukaan suatu benda dalam citra yang berhubungan dengan kasar dan halus, juga sifat – sifat spesifik dari kekasaran dan kehalusan permukaan tadi, yang sama sekali terlepas dari warna permukaan tersebut.

2.1.5 Operasi Pengolahan Citra

Wijaya dan Prijono (2007), menjelaskan operasi-operasi yang dilakukan dalam pengolahan citra banyak ragamnya. Namun, secara umum pada pengolahan citra terdapat enam jenis operasi pengolahan, yaitu:

1. Peningkatan kualitas citra (*image enchancement*)

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

   a. Perbaikan kontras gelap/terang
   b. Perbaikan tepian objek (*Edge enchancement*)
   c. Penajaman (*Sharpening*)
   d. Pemberian warna semu (*Pseudocoloring*)
   e. Penapisan derau (*Noise filtering*)

2. Restorasi citra (*image restoration*)

   Operasi ini bertujuan menghilangkan / meminimumkan cacat pada citra. Tujuan restorasi citra hampir sama dengan operasi peningkatan kualitas citra. Bedanya, pada restorasi citra penyebab degradasi gambar diketahui. Contoh – contoh operasi restorasi citra:

   a. Penghilangan derau (*noise*)
   b. Penghilangan kesamaran (*deblurring*)

3. Kompresi citra (*image compression*)

   Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam kompresi citra adalah citra yang telah dikompresikan harus tetap mempunyai kualitas gambar yang bagus. Contoh metode kompresi citra adalah metode JPEG. Misalkan ada citra kapal yang berukuran 258 KB. Hasil kompresi citra dengan metode JPEG dapat mereduksi ukuran citra semula sehingga menjadi 49KB saja.

4. Segmentasi citra (*image segmentation*)

   Operasi ini adalah suatu tahap pada proses analisis citra yang bertujuan untuk memperoleh informasi yang ada dalam citra tersebut dengan membagi citra kedalam daerah-daerah terpisah dimana setiap daerah adalah homogen dan mengacu pada sebuah kriteria keseragaman yang jelas. Segmentasi yang dilakukan pada citra harus tepat agar informasi yang terkandung didalamnya dapat diterjemahkan dengan baik.

5. Analisis citra (*image analysis*)

   Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik analisis citra mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Proses segmentasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya. Contoh – contoh operasi analisis citra:

   a. Pendeteksian tepi objek (*edge detection*)
   b. Ekstraksi batas (*boundary*)

c. Representasi daerah (*region*)

6. Rekonstruksi citra (*image reconstruction*)

   Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunakan dalam bidang medis. Misalnya beberapa foto rontgen dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh.

2.2 Deteksi Tepi

Deteksi Tepi menggunakan Software MATLAB. MATLAB (Matrix Laboratory) adalah sebuah program untuk analisis dan komputasi numerik dan merupakan suatu bahasa pemrograman matematika lanjutan yang dibentuk dengan dasar pemikiran menggunkan sifat dan bentuk matriks.

GUIDE atau GUI builder merupakan sebuah *graphical user interface* (GUI) yang dibangun dengan obyek grafik seperti tombol (*button*), kotak teks, slider, menu dan lain-lain. Aplikasi yang menggunakan GUI umumnya lebih mudah dipelajari dan digunakan karena orang yang menjalankannya tidak perlu mengetahui perintah yang ada dan bagaimana kerjanya. Untuk Memulai GUIDE Matlab dapat dilakukan dengan dua cara, yaitu:
Melalui command matlab dengan mengetikkan: >> guide
Klik tombol Start Matlab dan pilihlah MATLAB, lalu pilih GUIDE (GUI Bulder)
Setelah kita masuk dalam fitur GUI, maka kita buat tampilan untuk program *Deteksi Tepi* ini dengan membuat 2 Axes dan 7 Pushbutton. Axes berguna untuk menampilkan sebuah grafik atau gambar (image). Axes sebenarnya tidak masuk dalam *UIControl*, tetapi axes dapat diprogram agar pemakai dapat berinteraksi dengan axes dan obyek grafik yang ditampilkan melalui axes. Sedangkan Pushbutton merupakan jenis kontrol berupa tombol tekan yang akan menghasilkan tindakan jika diklik, misanya tombol OK, Cancel, Hitung, Hapus, dan sebagainya.

2.3 Macam-macam Metode Untuk Proses Deteksi Tepi

Menuru Winarno, (2011) yang menjelaskan ada beberapa macam deteksi tepi yaitu diantaranya sebagai berikut :

1. Metode Robert

   Metode Robert adalah nama lain dari teknik differensial yang dikembangkan di atas, yaitu differensial pada arah horisontal dan differensial pada arah vertikal, dengan ditambahkan proses konversi biner setelah dilakukan differensial. Teknik konversi biner yang disarankan adalah konversi biner dengan meratakan distribusi warna hitam dan putih. Metode Robert ini juga disamakan dengan teknik DPCM (*Differential Pulse Code Modulation*).

2. Metode Prewitt

   Metode Prewitt merupakan pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF.

3. Metode Sobel

   Metode Sobel merupakan pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian dan gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan deteksi tepi.

   Ada beberapa metode pendeteksian tepi yang terkenal dan banyak digunakan, diantaranya adalah metode Robert, Prewitt, Sobel, dan Gonzales. Tapi di sini kita hanya akan membahas metode sobel saja karena operator sobel adalah operator yang banyak digunakan sebagai pendeteksian tepi karena kesederhanaan dan keampuhannya serta juga digunakan dalam penulisan tugas akhir ini. Metode sobel merupakan pengembangan dari metode robert dengan menggunakan filter HPF (high pass filter) yang diberi satu angka nol

penyangga. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan pendeteksian tepi.

Perhatikanlah bahwa operator sobel menempatkan penekanan atau pembobotan pada piksel-piksel yang lebih dekat dengan titik pusat jendela. Dengan demikian pengaruh piksel-piksel tetangga akan berbeda sesuai dengan letaknya terhadap titik di mana gradien dihitung. Gradien adalah hasil pengukuran perubahan dalam sebuah fungsi intensitas, dan sebuah citra dapat dipandang sebagai kumpulan beberapa fungsi intensitas kontinyu dari citra. Dari susunan nilai-nilai pembobotan pada jendela juga terlihat bahwa perhitungan terhadap gradien juga merupakan gabungan dari posisi horisontal dan vertikal.

4. Metode Canny

   Operator *Canny* merupakan deteksi tepi yang optimal. Operator *Canny* menggu nakan *Gaussian Derivative Kernel* untuk menyaring kegaduhan dari citra awal untuk mendapatkan hasil deteksi tepi yang halus.

5. Metode Laplacian of Gaussian (log)

   Metode ini akan mendeteksi *zero crossing* ,untuk menentukan garis batas antara hitam dan putih, yang terdapat pada turunan kedua dari citra yang bersangkutan. Kekurangann dari penerapan perator laplacian adalah sangat sensitif terhadap noise, namun demikian edge detection dengan operator ini dapat di tingkatkan hasilnya dengan menerapkan *thresholding*.

2.4 Metode Roberts

Metode Roberts merupakan metode yang menggunakan operator Roberts. Operator Roberts adalah operator yang berbasis gradien yang menggunakan dua buah kernel yang berukuran 2x2 piksel. Operator ini mengambil arah diagonal untuk penentuan arah dalam penghitungan nilai gradien, sehingga sering disebut dengan operator silang. (Sutoyo dkk., 2009:228).

Perhitungan gradien dalam operator Roberts adalah sebagai berikut.

dengan, G = besar gradien operator Roberts
= gradien Roberts arah horisontal
= gradien Roberts arah vertikal

di mana dan dihitung menggunakan kernel konvolusi sebagai berikut. Sebenarnya, metode Roberts dalam mendeteksi tepi menghasilkan citra yang kurang memuaskan. Mungkin dikarenakan kernel yang digunakan berukuran 2x2 piksel dan penghitungan gradien hanya mengambil kedua arah diagonal.

Citra masukan berupa citra grayscale.

1. Konvolusikan citra grayscale dengan kernel Roberts horisontal (Rx) dan kernel Roberts vertikal (Ry).
2. Hitung besar gradien dengan rumus .
3. Citra keluaran merupakan hasil dari besar gradien (G).

2.5 Metode Sobel

Metode Sobel merupakan metode yang menggunakan operator Sobel. Operator ini menggunakan dua buah kernel yang berukuran 3x3 piksel untuk penghitungan gradien sehingga perkiraan gradien berada tepat di tengah jendela. (Sutoyo dkk., 2009:229).

Citra masukan berupa citra grayscale.

1. Konvolusikan citra grayscale dengan kernel Sobel horisontal (Sx) dan kernel Sobel vertikal (Sy).
2. Hitung besar gradien dengan rumus .
3. Citra keluaran merupakan hasil dari besar gradien (G).

2.6 MATLAB

Matlab adalah singkatan dari MATrix LABoratory, merupakan bahasa pemrograman yang dikembangkan oleh The Mathwork Inc. yang hadir dengan fungsi dan karakteristik yang berbeda dengan bahasa pemrograman lain yang sudah ada lebih dahulu seperti Delphi, Basic maupun C++. Matlab merupakan

bahasa pemrograman level tinggi yang dikhususkan untuk kebutuhan komputasi teknis, visualisasi dan pemrograman seperti komputasi matematik, analisis data, pengembangan algoritma, simulasi dan pemodelan dan grafik-grafik perhitungan Pada awalnya Matlab dibuat untuk memberikan kemudahan mengakses data matrik pada proyek LINPACK dan EISPACK. Saat ini matlab memiliki ratusan fungsi yang dapat digunakan sebagai *problem solver* baik permasalahan yang mudah maupun masalah-masalah yang kompleks dari berbagai disiplin ilmu.

Beberapa kelebihan Matlab jika dibandingkan dengan program lain seperti Fortran, dan Basic adalah :

a. Mudah dalam memanipulasi struktur matriks dan perhitungan berbagai operasi matriks yang meliputi penjumlahan, pengurangan, perkalian, invers dan fungsi matriks lainnya.
b. Menyediakan fasilitas untuk memplot struktur gambar (kekuatan fasilitas grafik tiga dimensi yang sangat memadai).
c. *Script* program yang dapat diubah sesuai dengan keinginan *user*.
d. Jumlah *routine-routine powerful* yang berlimpah dan terus berkembang.
e. Kemampuan *interface* (misal dengan bahasa C, word dan mathematica).
f. Dilengkapi dengan *toolbox*, *simulink*, *stateflow* dan sebagainya, serta mulai melimpahnya*source code* di internet yang dibuat dalam matlab (contoh *toolbox* misalnya : *signal processing*, *control system*, *neural networks* dan sebagainya).

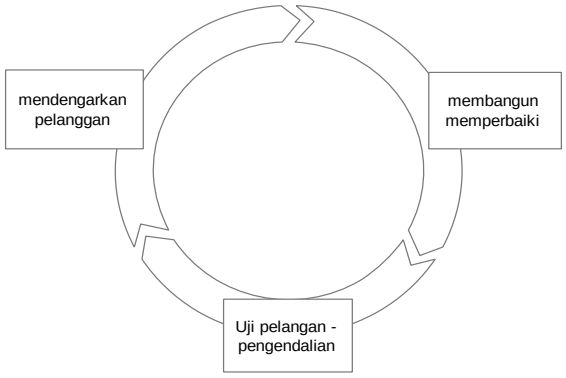
Penggunaan MATLAB meliputi bidang-bidang :

- Matematika dan Komputasi
- Pembentukan Algorithm
- Akuisisi Data
- Pemodelan, simulasi dan Pembuatan *Prototype*
- Analisis Data, Explorasi, dan Visualisasi
- Grafik Keilmuan dan Bidang Rekayasa Lingkungan kerja

2.7 Metode Pengembangan Perangkat Lunak Protoype

Pressman (2002), menyatakan bahwa *prototype model* merupakan metode yang efektif dalam merancang perangkat lunak.*Prototypemodel*dimulai dengan

mengumpulkan kebutuhan.Pengembang dan pelanggan bertemu dan mendefinisikan object keseluruhan dari perangkat lunak, mengidentifikasi segala kebutuhan yang diketahui dan kemudian melakukan "perancangan kilat". Perancangan kilat berfokus pada penyajian dari aspek-aspek perangkat lunak tersebut yang akan nampak bagi pelanggan atau pemakai (contohnya pendekatan input dan format output). Perancangan kilat membawa kepada kontruksi sebuah *prototype*.*Prototype* tersebut dievaluasi oleh pelanggan dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak.



Gambar 2.2*Prototipe* paradigma

Prototypemodel juga dapat didefinisikan sebagai proses pengembangan suatu prototipe secara cepatuntuk digunakan terlebih dahulu dan ditingkatkan terus menerus sampai didapatkan sistem yangutuh. *Prototypemodel*merupakan proses yang digunakan untuk membantu pengembang perangkatlunak dalam membentuk *prototype* dari perangkat lunak yang harus dibuat.Proses pada model *prototyping* dapat dijelaskan sebagai berikut :

1) Pengumpulan kebutuhan : D*eveloper* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akandibutuhkan berikutnya.
2) Perancangan : Perancangan dilakukan cepat dan rancangan mewakili semua aspek perangkat lunak yang diketahui, dan rancangan ini menjadi dasar pembuatan*prototype*.
3) Evaluasi *prototype* : Klien mengevaluasi *prototype* yang dibuat dan digunakan untuk memperjelas kebutuhan perangat lunak.

Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. *Prototype-prototype* dibuat untuk memuaskan kebutuhan klien dan untuk membangun perangkat lunak lebih cepat, namun tidak semua *prototype* bisa dimanfaatkan.Demi kebutuhan klien lebih baik p*rototype* yang dibuat diusahakan dapat dimanfaatkan.

2.8 UML (*Unified Markup Languange*)

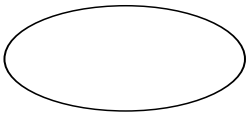
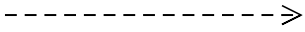
Rosa dan Shalahuddin (2014) menyatakan UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodelogi dalam mengembangkan sistem berorientasi object dan juga merupakan alat untuk mendukung pengembangan sistem.UML saat ini banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum pada dunia industri perangkat lunak dan pengembangan software.

Alat bantu yang dipergunakan dalam perancangan berorientasi objek berbasis UML adalah sebagai berikut :

2.8.1*Use CaseDiagram*

Use casediagram merupakan pemodelan untuk kelakuan (*behavior*) sistem yang dibuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem yang akan dibuat. Dapat dikatakan use case digunakan untuk mengetahui fungsi apa saja yang dibuat dan siapa saja yang berhak mengunakan fungsi-fungsi tersebut. Berikut simbol yang digunakan pada *use case diagram* :

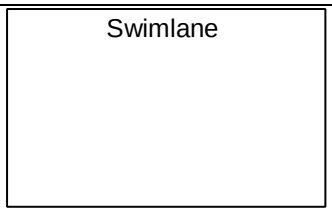
Tabel 2.13 Simbol *Use Case Diagram*

Gambar	Keterangan
	Use case menggambarkan fungsionalitas yang disediakan system sebagai unit-unit yang bertukar pesan antar unit dengan actor, biasanya dinyatakan dengan mengunakan kata kerja diawal nama use case.
 Actor	Aktor adalah orang lain atau system lain yang mengaktifkan fungsi dari target system. Untuk megidentifikasikan actor, harus ditentukan pembagian kerja dan tugas-tugas yang berkaitan dengan peran pada target system. Orang atau system bias muncul dalam beberapa peran.
	Ososiasi antara actor dan use case yang mengunakan panah terbuka untuk mengindikasikan bila actor berinteraksi secara pasif dengan system.
	Ososiasi antara actor dan use case yang mengunakan garis untuk mengindikasikan siapa atau apa yang meminta interaksi langsung bukan mengindikasikan aliran data.
 <<include>>	Include merupakan pemangilan use case oleh use case lainya , contohnya pemanggilan sebuah fungsi program.
 <<extend>>	Perluasan dari use case lain jika kondisi atau syarat terpenuhi.

2.8.2 *Activity Diagram*

Secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun *use case*. *Activity diagram* dapat juga digunakan untuk memodelkan action yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari action tersebut**.**Simbol- simbol yang digunakan dalam *activity diagram*yaitu :

Tabel 2.14Simbol *Activity Diagram*

Gambar	Keterangan
	Mengambarkan suatu proses/kegiatan sistem
	Menunjukan eksekusi dari suatu aksi
	Start Point merupakan awak dari suatu aktivitas
	End point, akhir aktivitas
	Fork/Rake Node Satu aliran atau beberapa aliran yang pada tahap tertentu berubah menjadi satu atau beberapa aliran.
	Swinlane, pembagian activity diagram untuk menunjukan siapa melakukan apa

2.8.3 *Class Diagram*

Class diagram merupapakan hubungan antara kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sitem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.*Class diagram* juga menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan constraint yang berhubungan dengan objek yang dikoneksikan.

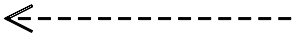
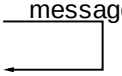
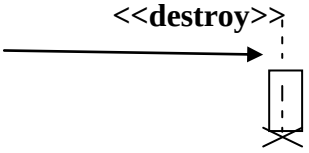
Class diagram secara khas meliputi : kelas (*class*), relasi, *Associations*, *Generalizion* dan *Agregation*, Atribut, operasi, dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau object.

2.8.4 *Sequence Diagram*

Sequencediagram menggambarkan kelakuan objek pada *use case* dengan mendeksripsikan waktu hidup objek dan pesan yang dikirimkan dan

diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel 2.15Simbol *Sequence Diagram*

Gambar	Keterangan
	Orang, proses atau system lain yang berinteraksi dengan sistem
1: Masukan	Menyatakan bahwa suatu objek mengirimkan data atau masukan atau informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
	Activation, activation mewakili sebuah eksekusi operasi dari objek.
1: Keluaran 	Message return, simbol membalas pesan
1: [condition] message name 	Recursive, mengambarkan pengiriman pesan untuk dirinya sendiri.
<<**create**>>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah objek yang di buat.
<<**destroy**>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy.
Garis Hidup / Lifeline	Menyatakan kehidupan suatu objek

Nama objek: nama kelas	Menyatakan objek yang berinteraksi pesan