BAB II

LANDASAN TEORI

II.1 Perancangan

Perancangan menurut mempunyai 2 maksud, yaitu untuk memenuhi kebutuhan kepada pemakai sistem dan untuk memberikan gambaran yang jelas kepada pemogram komputer dan ahli-ahli teknik lainnya yang terlibat (Hanik Mujiati, 2012).

II.2 Aplikasi

Aplikasi adalah suatu program yang dibuat oleh pemakai yang ditujukan untuk melakukan suatu tugas khusus. Berdasarkan definisi di atas maka dapat disimpulkan bahwa aplikasi adalah program yang dibuat untuk melakukan tugas khusus dalam perusahaan (Neti, 2012).

II.3 Pengenalan Algoritma

Istilah algoritma berasal dari nama seorang pengarang berkebangsaan Arab bernama Ja'fat Mohammed bin Musa al Khowarizmi (tahun 790 – 840), yang sangat terkenal dengan sebutan bapak Aljabar. Secara defenisi algoritma adalah alur pemikiran yang logis yang dapat dituangkan ke dalam bentuk tulisan (Antonius Rachmat C, 2010, 4). Sebuah algoritma dikatakan benar (correct) jika algoritma tersebut berhasil mengeluarkan output yang benar untuk semua kemungkinan input (Ihsan Dedy Boy Marpaung, 2013).

Pertimbangan dalam pemilihan algoritma adalah algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah

masukan yang diberikan. Tidak peduli serumit apapun algoritma, jika memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik. Pertimbangan lain yang harus diperhatikan adalah seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk menyelesaikan masalah yang memerlukan aproksimasi hasil. Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya.

Selanjutnya adalah efisiensi algoritma. Efisiensi algoritma dapat ditinjau dari 2 hal yaitu efisiensi waktu dan memori. Meskipun algoritma memberikan keluaran yang benar, tetapi jika harus menunggu lama untuk mendapatkan keluarannya, algoritma tersebut bukanlah algoritma yang baik, setiap orang menginginkan keluaran yang cepat. Dalam kenyataannya, setiap orang bisa membuat algoritma yang berbeda untuk menyelesaikan suatu permasalahan, walaupun terjadi perbedaan dalam menyusun algoritma, tentunya diharapkan keluaran yang sama (Diana Efendi, 2011).

II.4 Pengertian Aplikasi

Aplikasi adalah suatu program yang dibuat oleh pemakai yang ditujukan untuk melakukan suatu tugas khusus. Berdasarkan definisi di atas maka dapat disimpulkan bahwa aplikasi adalah program yang dibuat untuk melakukan tugas khusus dalam perusahaan (Neti, 2010).

Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau suite aplikasi *(application suite)*. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antar muka pengguna yang

memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna.

II.5 Pengertian Citra

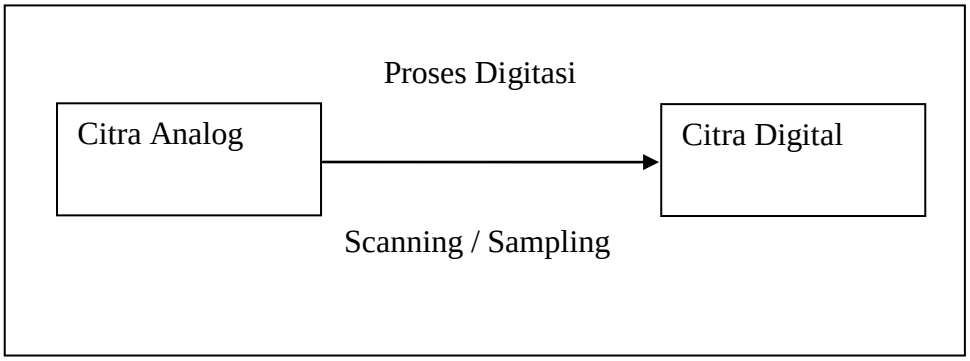
Dalam pengertian yang umum, citra adalah gambar, citra adalah gambaran visual mengenai suatu objek atau beberapa objek (Abdul kadir dan Adhi Susanto, 2013:2). Citra adalah suatu representasi (gambaran), kemiripin, atau imitasi dari suatu obek (T. Sutoyo, 2009:). Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinya-sinyal video seperti gambar pada monitor televisi atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan. Sebuah pixel adalah sampel dari pemandangan dalam bilangan bulat. Citra dikelompokkan menjadi dua bagian yaitu citra diam (*still image*) dan citra bergerak (*animated images*). Citra diam adalah citra tunggal yang tidak bergerak. Sebaliknya, citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun (sekuensial) sehingga memberi kesan pada mata sebagai gambar yang bergerak. Setiap citra di dalam rangkaian itu disebut *frame*. Gambar-gambar yang tampak pada film layar lebar atau televisi pada hakekatnya terdiri dari ratusan sampai ribuan *frame*. Citra dapat digolongkan menjadi dua jenis, yaitu citra analog dan citra digital.

II.5.1 Citra Analog

Citra analog adalah citra yang bersifat berterusan (*continue*), seperti gambar pada monitor televisi, foto sinar X, foto yang tercetak di kertas foto, lukisan, pemandangan alam, hasiil scan, gambar-gambar yang terekam dengan pita kaset, dan lain sebagainya (T. Sut, 2009:). Citra analog tidak dapat direpsentasikan dalam komputer sehingga tidak bisa diproses secara langsung. Oleh sebab itu, agar citra ini dapat diproses di komputer, proses konversi analog ke digital harus dilakukan terlebih dahulu. Citra dihasilkan dari alat-alat analog, seperti video kamera analog, kamera foto analog, WebCam, scan, sensor ultrasound pada sistem usg dan lain-lain.

II.5.2 Citra Digital

Citra digital adalah citra yang dapat diolah oleh komputer. Citra digital merupakan sebuah larik (array) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit tertentu (Darma Putra, 2010). Citra digital terdiri dari sinyal-sinyal frekuensi elektromagnetis yang sudah di*sampling*, dan ukuran *pixel* dari citra tersebut sudah dapat ditentukan. *Sampling* merupakan proses pembentukan citra digital dari citra analog. Suatu citra yang dicetak diatas kertas disebut dengan citra analog, jika citra analog tersebut di *scan* dengan alat *scanner* maka akan terjadi citra digital. Dengan demikian, *scanner* merupakan alat *sampling*. Proses pembentukan citra digital dari citra analog diperlihatkan pada gambar 2.1.



Gambar II.1 Pembentukan Citra Digital dari Citra Analog
Sumber: T.Sutoyo, 2009:12

Proses perubahan citra analog menjadi citra digital disebut digitalisasi citra. Citra yang dihasilkan dari peralatan digital (citra digital) langsung bisa diproses oleh komputer karena peralatan digital sudah terdapat sistem sampling dan kuantisasi. Sedangkan peralatan analog tidak dilengkapi ke dua sistem tersebut. Ke dua sistem inilah yang bertugas memotong-motong citra menjadi x kolom dan y baris (proses sampling), sekaligus menentukan besar intensitas yang terdapat pada titik tersebut (proses kuantisasi) sehingga menghasilkan resolusi citra yang diinginkan.

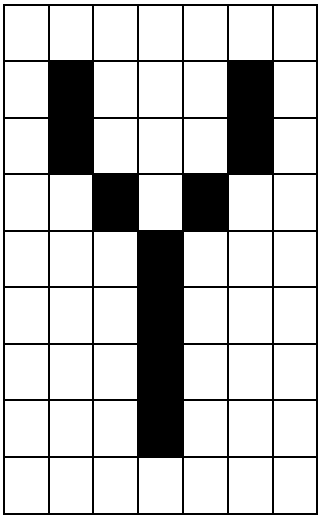
II.6 Jenis Citra Digital

Citra digital berhubungan erat dengan warna. Nilai data digital merepresentasikan warna dari citra. Citra digital berdasarkan warna penyusunnya dapat dibedakan menjadi 4 (empat) jenis, yaitu citra biner (*monochrome*), citra skala keabuan (*gray scale*), citra warna (*true color*), dan citra warna berindeks (Rinaldi Munir, 2004). Berikut ini penjelasan untuk masing-masing format citra digital.

II.6.1 Citra Biner (*Monochrome*)

Citra biner (*monochrome*) atau disebut juga *binary image*, merupakan citra digital yang setiap *pixel*-nya hanya memiliki 2 kemungkinan derajat keabuan, yaitu 0 dan 1. Nilai 0 mewakili warna hitam dan nilai 1 mewakili warna putih. Setiap piksel pada citra biner membutuhkan media penyimpanan sebesar 1 bit (Rinaldi Munir, 2004).

1	1	1	1	1	1	1
1	0	1	1	1	0	1
1	0	1	1	1	0	1
1	1	0	1	0	1	1
1	1	1	0	1	1	1
1	1	1	0	1	1	1
1	1	1	0	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1



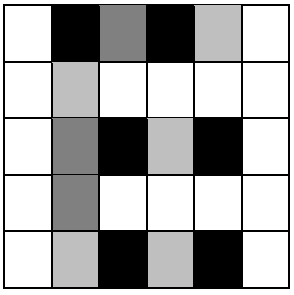
Gambar II.2 Bentuk Citra Biner
Sumber : Rinaldi Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, 2004

Gambar II.3 Citra Biner
Sumber : Rinaldi Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, 2004

II.6.2 Citra Skala Keabuan (*Grayscale*)

Citra skala keabuan atau disebut juga dengan citra aras keabuan memberikan kemungkinan yang lebih banyak. Format citra ini disebut dengan aras keabuan karena terdapat warna abu-abu diantara warna minimum (hitam) dan warna maksimum (putih).

Jumlah maksimum warna sesuai dengan bit penyimpanan yang digunakan, yaitu 4 bit atau 8 bit. Citra dengan skala keabuan 4 bit memiliki 2^4 = 16 kemungkinan warna, yaitu 0 (minimal) hingga 15 (maksimal). Sementara citra digital dengan skala keabuan 8 bit memiliki 2^8 = 256 kemungkinan, yaitu 0 (minimal) hingga 255 (maksimal) (Rinaldi Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, 2004, Bandung).



Skala keabuan 4 bit (hitam = 0, putih = 15)

= 15	12	15	15	15	15	
	= 15	5	0	12	0	15
	= 15	8	15	15	15	15
	= 15	10	0	13	0	15

Gambar II.4 Bentuk Citra Skala Keabuan
Sumber : Rinaldi Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, 2004

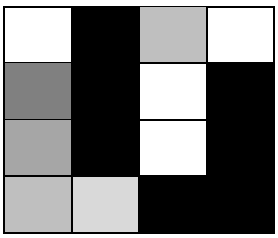
Gambar II.5 Citra Skala Keabuan
Sumber : Rinaldi Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, 2004

II.6.3 Citra Warna (*True Color*)

Pada citra warna (*true color*) setiap *pixel*-nya merupakan kombinasi dari tiga warna dasar merah, hijau dan biru, sehingga citra warna ini disebut juga citra RGB (*red*, *green*, *blue*). Setiap komponen warna memiliki intensitas sendiri dengan nilai minimum 0 dan nilai maksimum 255 (8 bit). Hal ini menyebabkan setiap piksel pada citra RGB membutuhkan media penyimpanan 3 byte. Jumlah kemungkinan kombinasi warna citra RGB 2^{24} = lebih dari 16 juta warna(Rinaldi Munir, 2004).

1 byte	1 byte	1 byte
blue	*green*	*red*

Citra warna



=	255	255	255	0	0	0	128	128	128	128	128	0
=	0	255	255	0	0	0	204	255	255	0	0	255
=	150	150	150	51	51	51	255	255	255	95	95	95
=	255	204	153	255	204	153	128	0	0	255	95	255

Gambar II.6 Bentuk Citra Warna
Sumber : Rinaldi Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, 2004

Gambar II.7 Citra Warna
Sumber : Rinaldi Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, 2004

II.7 Format Citra Digital

Untuk menyimpan foto dan citra digunakan formasi citra layar kuadrat (berbentuk kotak) yang terdiri atas titik-titik citra kecil yang disebut dengan piksel (*pixel*). Piksel disebut juga dengan dot. Piksel berbentuk bujur sangkar dengan ukuran relatif kecil. Banyaknya piksel tiap satuan luas tergantung pada resolusi yang digunakan. Keanekaragaman warna piksel tergantung pada bit *depth* yang dipakai. Semakin banyak jumlah piksel tiap satuan luas, semakin baik kualitas citra yang dihasilkan dan tentu akan semakin besar ukuran *file-nya* (Darma Putra, 2010).

Resolusi adalah jumlah piksel per satuan yang terdapat pada suatu citra. Satuan piksel yang sering dipakai adalah dpi (*dot per inchi*) atau ppi (*pixel per inchi*). Satuan dpi menentukan jumlah piksel yang ada setiap satu satuan luas. Dalam hal ini adalah satu inchi kuadrat. Resolusi sangat berpengaruh pada detail dan perhitungan citranya. Jika suatu citra dengan luas 1 inchi kuadrat dan jumlah dot adalah 60 x 60 (yang berarti mempunyai resolusi 3600 dpi) diperbesar

menjadi 10 inchi maka jumlah piksel tetap 3600 dpi, tetapi resolusinya berubah menjadi 3600 : 10 = 360. Hal ini menyebabkan citra menjadi kabur atau kasar.

Colordepth (kedalaman warna) sering disebut dengan *pixel depth* atau *color depth*. *Color depth* menentukan berapa banyak informasi warna yang tersedia untuk ditampilkan atau dicetak dalam setiap piksel. Semakin besar nilainya semakin bagus kualitas citra yang dihasilkan. Tentu ukurannya juga semakin besar. Misalkan suatu citra mempunyai *color depth* = 1, ini berarti hanya ada 2 kemungkinan warna ($2^1 = 2$) yang ada pada citra tersebut yaitu hitam dan putih. *Color depth* = 24 berarti mempunyai kemungkinan warna 2^{24} = 16,7 juta warna.

II.7.1 Citra Bitmap (BMP)

Citra bitmap sering disebut juga dengan citra raster. Citra bitmap menyimpan kode citra secara digital dan lengkap (cara penyimpanannya adalah per *pixel*). Citra bitmap dipresentasikan dalam bentuk matriks atau dipetakan dengan menggunakan bilangan biner atau sistem bilangan lain. Tampilan bitmap mampu menunjukkan kehalusan gradiasi bayangan dan warna dari sebuah gambar. Oleh karena itu, bitmap merupakan media elektronik yang paling tepat untuk gambar-gambar dengan perpaduan gradiasai warna yang rumit, seperti foto, kamera digital, video *capture* dan lain-lain (T. Sutoyo, 2009).

Kriteria yang paling penting dari citra ini adalah kedalaman warna yaitu banyaknya warna bit per *pixel* yang didefinisikan dari sebuah warna (Rinaldi Munir, Pengolahan Citra Digital Dengan Pendekatan Algoritmik, 2004, Bandung). Bitmap dengan mengikuti kriteria warna maka dapat dibagi menjadi :

1. 8 bit = 256 warna (*gray scales*)
2. 24 bit = 16.777.216 warna

Secara umum dapat dikatakan dikatakan semakin banyaknya warna, maka akan diperlukan keamanan yang ketat atau tinggi dikarenakan bitmap memiliki area yang sangat luas dalam sebuah warna yang seharusnya dihindarkan. Dilihat dari kedalaman atau kejelasan dari sebuah warna, bitmap dapat mengambil sejumlah data tersembunyi dengan perbandingan sebagai berikut (ukuran ratio dari bitmap dalam byte : ukuran dari data yang disembunyikan) :

1. 8 bit = 256 warna : 8 : 1
2. 24 bit = 16.777.216 warna : 8 : 1

Perbandingan tersebut diperoleh dari penentuan LSB dalam satu byte, untuk citra 8 bit letak LSB adalah pada bit terakhir sedangkan untuk citra 24 bit letak LSB adalah pada bit ke 8, bit ke 16 dan bit ke 24 dimana masing-masing byte mewakili warna merah (*red*), hijau (*green*) dan warna biru (*blue*). Manipulasi pada bitmap tidak dapat dikonversikan atau diubah ke dalam bentuk format grafik yang lain karena data tersembunyi dalam file tersebut akan hilang. Format citra menggunakan metode kompresi yang lain (seperti JPEG) tidak digunakan dalam penelitian ini. Mengurangi ukuran dari *carrier* file sangatlah penting untuk melakukan transmisi *online*, yaitu dengan menggunakan utilitas kompresi (sepert : ARZ, LZH, PKZIP, WinZip), dikarenakan besar ukuran file sangatlah berpengaruh.

II.7.2 *Graphic Interchange Format* (GIF)

Graphic Interchange Format (GIF) dibuat oleh *Compuserve* pada tahun 1987 untuk menyimpan berbagai gambar dengan format bitmap menjadi sebuah *file* yang mudah untuk diubah pada jaringan komputer. GIF adalah *file* format grafik yang paling tua pada web, dan begitu dekatnya file format ini pada web saat itu sehingga kebanyakan *browser* menggunakan format ini. *File* GIF dapat disimpan dalam dua jenis *sort* yaitu secara berurutan (dari atas ke bawah) dan pembagian dengan baris (8 baris, 4 baris dan 2 baris). Pembagian baris pada gambar dengan resolusi gambar yang rendah dengan cepat dimana secara gradual datangnya untuk menjadikan lebih fokus, dengan *expense* dari penambahan kapasitas *file*.

Format *file* ini hanya mampu menyimpan dalam 8 bit (hanya mendukung *mode* warna *grayscale*, bitmap dan *indexed color*). Format *file* ini merupakan format standar untuk publikasi elektronik dan internet. Format file ini mampu menyimpan animasi dua dimensi yang akan diplubikasikan pada internet, desain halaman web dan publikasi elektronik. Format *file* GIF yang umum digunakan antara lain :

1. GIF87a : mendukung *interlacing* dan mampu beberapa image dalam 1 file, ditemukan tahun1987 dan menjadi standar.
2. GIF89a : kelanjutan dari 87a dan ditambahkan dengan dukungan *transparency*, mendukung teks, dan animasi.

Format GIF merupakan format *file* yang paling banyak disarankan dan digunakan. Beberapa alasan format *file* ini banyak digunakakan antara lain karena:

1. Ukuran *file* yang dihasilakn relatif kecil.
2. Mampu menggabungkan beberapa gambar menjadi satu kesatuan dan menampilkannya secara bergantian (animasi).
3. Warna latar belakang dapat dibuat transparan dan adanya teknologi *interlacing* yang akan membuat sebuah *file* di *load* secara utuh dengan kualitas yang ditampilkan secara bertahap.

II.7.3 *Joint Photographic Experts* (JPEG)

Joint Photographic Experts (JPEG) dirancang untuk kompresi beberapa *full color* atau *gray scale* dari suatu gambar yang asli, seperti pemandangan asli di dunia ini. JPEG bekerja dengan baik pada *continous tone images* seperti *photographs* tetapi tidak terlalu bagus pada ketajaman gambar dan seni pewarnaan seperti penulisan, kartun yang sederhana atau gambar yang menggunakan banyak garis. JPEG sudah mendukung 24 bit *color depth* atau sama dengan 16,7 juta warna (2^{24} = 16.777.216 warna), *progressive* JPEGs (p-JPEGs) adalah tipe dari beberapa persen lebih kecil dibandingkan *baseline* JPEGs. Tetapi keuntungan dari JPEG dan tipe-tipenya terlihat pada langkah-langkahnya sama seperti *interlaced* GIFs. JPEG adalah algoritma kompresi secara *lossy* (T. Sutoyo, 2009).

JPEG bekerja dengan merubah gambar spasial dan merepresentasikan ke dalam pemetaan frekuensi. *Discrete cosine transform* (DCT) dengan memisahkan antara informasi frekuensi yang rendah dan tinggi dari sebuah gambar.Informasi frekuensi yang tinggi akan diseleksi untuk dihilangkan yang terikat pada

pengaturan kualitas yang digunakan. Kompresi dengan tingkatanyang lebih baikdari yang dihilangkan. Waktu kompresi dan dekompresi dilaksanakan dengan simetris.

II.8 Pendeteksian Tepi

Pendeteksian tepi merupakan langkah pertama untuk melingkupi imformasi didalam citra. Tepi mencirikan batas-batas objek, oleh karena itu tepi berguna untuk proses segmentasi dan identifikasi objek didalam citra. Tujuan operasi pendeteksian tepi adalah untuk meningkatkan penampakan garis batas suatu daerah atau objek didalam citra (T.Sutoyo, 2009).

II.8.1 Konvolusi

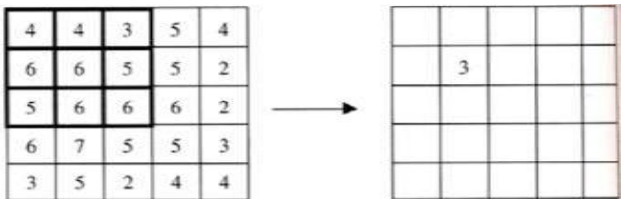
Deteksi tepi merupakan salah satu proses pengolahan citra yang menggunakan filter atau penapis. Untuk mengaplikasikan penapis pada citra, digunakan metode konvolusi. Konvolusi dinyatakan dalam matriks, dimana setiap elemen matriks penapis tersebut dinamakan koefisien konvolusi. Operasi bekerja dengan menggeser kernel piksel per piksel, yang hasilnya kemudian disimpan dalam matriks baru.Untuk lebih jelasnya, berikut contoh konvolusi yang terjadi antara citra f(x,y) berukuran 5x5 dengan sebuah kernel berukuran 3x3 yang diperlihatkan pada gambar 2.8.

$$F(x,y) = \begin{pmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 3 & 5 & 2 & 4 & 4 \end{pmatrix} \qquad g(x,y) = \begin{pmatrix} 0 & -1 & 0 \\ -1 & .4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Gambar II.8 ***MatriksCitra dan KernelsebelumKonvolusi***
Sumber : Sutoyo T, Teori Pengolahan Citra Digital, ANDI Yogyakarta, 2009.

Tanda . (titik) menunjukkan posisi (0,0) dari kernel Tahapan untuk mendapatkan hasil konvolusi yang terjadi antara citra dan kernel

di atas dapat dilihat pada Gambar2.9 (Sutoyo, 2009)..



Gambar 2.9 ***Tahapan Proses Pembentukan Konvolusi***
Sumber : Sutoyo T, Teori Pengolahan Citra Digital, ANDI Yogyakarta, 2009.

Sehingga diperoleh hasil akhir dari proses konvolusi tersebut, yang ditunjukkan pada gambar II.10.

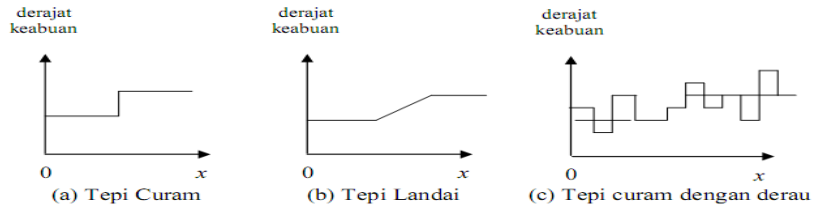
	3	0	8	
	0	2	6	
	6	0	2	

Gambar II.10 ***Hasil Kovolusi Citra dan Kernel***
Sumber : Sutoyo T, Teori Pengolahan Citra Digital, ANDI Yogyakarta, 2009.

II.8.2 Definisi Tepi

Tepi (*edge*) adalah perubahan nilai intensitas derajat keabuan yang mendadak besar dalam jarak yang singkat. Tepi biasanya terdapat pada batas antara dua daerah berbeda pada suatu citra. Tepi dapat diorientasikan dengan satu arah, dan arah ini berbeda-beda bergantung pada perubahan intensitas.Ada tiga macam tepi yang terdapat didalam citra digital ditunjukan pada gambar 2.11,yaitu:

1. Tepi curam yaitu tepi dengan perubahan intensitas yang tajam. Arah tepi berkisar 90^0.
2. Tepi landai yaitu tepi dengan sudut arah yang kecil. Tepi landai dapat dianggap terdiri dari sejumlah tepi-tepi lokal yang lokasinya berdekatan.
3. Tepiyang mengandung derau (noise).Umumnya tepi yang terdapat pada aplikasi komputer mengandung derau.Operasi peningkatan kualitas citra dapat dilakukan terlebih dahulu sebelumpendeteksiantepi.



Gambar II.11 *Jenis-jenis Tepi*
Sumber : Sutoyo T, Teori Pengolahan Citra Digital, ANDI Yogyakarta, 2009.

II.8.3 Tujuan Pendeteksian Tepi

Pendeteksian tepi merupakan langkah pertama untuk melingkupi informasi didalam citra. Tepi mencirikan batas-batas objek dan karena itu tepi berguna

untuk proses segmentasi dan identifikasi objek didalam citra.Tujuan operasi pendeteksian tepi adalah untuk meningkatkan penampakan garis batas suatu daerah atau objek didalam citra (Darma Putra, 2010).

II.9 Algoritma Canny

Deteksi tepi yang menggunakan *canny* yang diimplementasikan oleh pemetaan vektorintensitas menggunakan transformasi linear dan kemudian mendeteksi tepi berdasarkan sudut antara intensitas vector dan diproyeksikan kedalam ruang bagian tepi. Representasi dalam bentuk kernel operator. *canny*ditunjukkan pada gambar di bawah ini :

	x-1	x	x+1
y-1	z_1	z_2	z_3
y	z_4	z_5	z_6
y+1	z_7	z_8	z_9

(a) Posisi pada citra f

-1	0	1
$-\sqrt{2}$	0	$\sqrt{2}$
-1	0	1

(b) G_x

1	$\sqrt{2}$	1
0	0	0
-1	$-\sqrt{2}$	-1

(c) G_y

Gambar II. 12 *Frei-Chen*

Sumber : Abdul Kadir dan Adhi Susanto, 2013

II.10 Unified Modelling Language

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual dan juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek (Haviluddin, 2011:1).

Sejarah UML sendiri terbagi dalam dua fase; sebelum dan sesudah munculnya UML. Dalam fase sebelum, UML sebenarnya sudah mulai diperkenalkan sejak tahun 1990an namun notasi yang dikembangkan oleh para ahli analisis dan desain berbeda-beda, sehingga dapat dikatakan belum memiliki standarisasi. Fase kedua; dilandasi dengan pemikiran untuk mempersatukan metode tersebut dan dimotori oleh *Object Management Group* (OMG) maka pengembangan UML dimulai pada akhir tahun 1994 ketika Grady Booch dengan metode OOD (*Object-Oriented Design*), Jim Rumbaugh dengan metode OMT (*Object Modelling Technique*) mereka ini bekerja pada Rasional Software Corporation dan Ivar Jacobson dengan metodeOOSE (*Object-Oriented Software Engineering*) yang bekerja pada perusahaan Objectory Rasional. Sebagai pencetus metode-metode tersebut mereka bertiga berinisiatif untuk menciptakan bahasa pemodelan terpadu sehingga pada tahun 1996 mereka berhasil merilis UML versi 0.9 dan 0.91 melalui Request for Proposal (RFP) yang dikeluarkan oleh OMG (Braun, et.al. 2001) (Haviluddin, 2011:1).

Kemudian pada Januari 1997 IBM, ObjecTime, Platinum Technology, Ptech, Taskon, Reich Technologies dan Softeam juga menanggapi Request for Proposal (RFP) yang dikeluarkan oleh OMG tersebut dan menyatakan kesediaan untuk bergabung. Perusahaan-perusahaan ini menyumbangkan ide-ide mereka, dan bersama para mitra menghasilkan UML revisi 1.1. Fokus dari UML versi rilis 1.1 ini adalah untuk meningkatkan kejelasan UML Semantik versi rilis 1.0. Hingga saat ini UML versi terbaru adalah versi 2.0 (http://www.uml.org/).

Saat ini sebagian besar para perancang sistem informasi dalam menggambarkan informasi dengan memanfaatkan UML diagram dengan tujuan utama untuk membantu tim proyek berkomunikasi, mengeksplorasi potensi desain, dan memvalidasi desain arsitektur perangkat lunak atau pembuat program (Haviluddin, 2011:1)..

Secara filosofi UML diilhami oleh konsep yang telah ada yaitu konsep permodelan Object Oriented karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik (Haviluddin, 2011:1-2).

II.10.1 Tujuan Penggunaan UML

Tujuan dari penggunaan diagram seperti diungkapan oleh Schmuller J. (2004), “The purpose of the diagrams is to present multiple views of a system; this set of multiple views is called a model”. Berikut tujuan utama dalam desain UML adalah (Sugrue J. 2009) :

1. Menyediakan bagi pengguna (analisis dan desain sistem) suatu bahasa pemodelan visual yang ekspresif sehingga mereka dapat mengembangkan dan melakukan pertukaran model data yang bermakna.
2. Menyediakan mekanisme yang spesialisasi untuk memperluas konsep inti.
3. Karena merupakan bahasa pemodelan visual dalam proses pembangunannya maka UML bersifat independen terhadap bahasa pemrograman tertentu.
4. Memberikan dasar formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar terhadap penggunaan alat desain sistem yang berorientasi objek (OO).
6. Mendukung konsep pembangunan tingkat yang lebih tinggi seperti kolaborasi, kerangka, pola dan komponen terhadap suatu sistem.
7. Memiliki integrasi praktik terbaik.

(Haviluddin, 2011:2).

II.10.2 Komponen UML

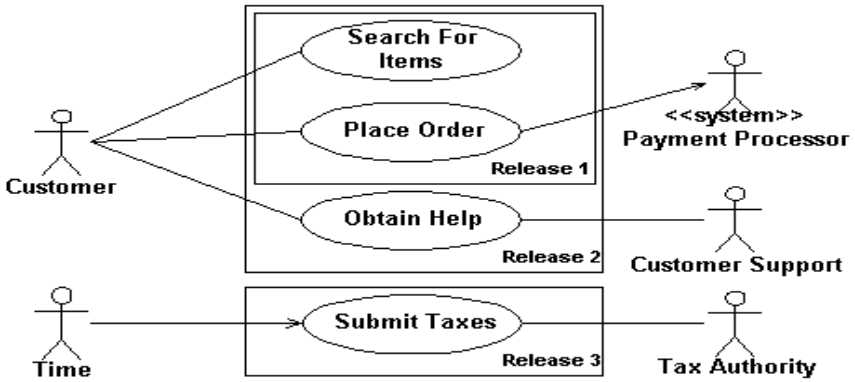
Sejauh ini para pakar merasa lebih mudah dalam menganalisa dan mendesain atau memodelkan suatu sistem karena UML memiliki seperangkat aturan dan notasi dalam bentuk grafis yang cukup spesifik (Haviluddin, 2011:3). Komponen atau notasi UML diturunkan dari 3 (tiga) notasi yang telah ada sebelumnya yaitu Grady Booch, OOD (Object-Oriented Design), Jim Rumbaugh, OMT (Object Modelling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering). Pada UML versi 2 terdiri atas tiga kategori dan memiliki 13 jenis diagram (Haviluddin, 2011:3),yaitu :

1. Use Case Diagram

   Diagram yang menggambarkan actor, use case dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah use case digambarkan sebagai elips horizontal dalam suatu diagram UML use case. Use Case memiliki dua istilah (Haviluddin, 2011:3)

   a. System use case; interaksi dengan sistem.

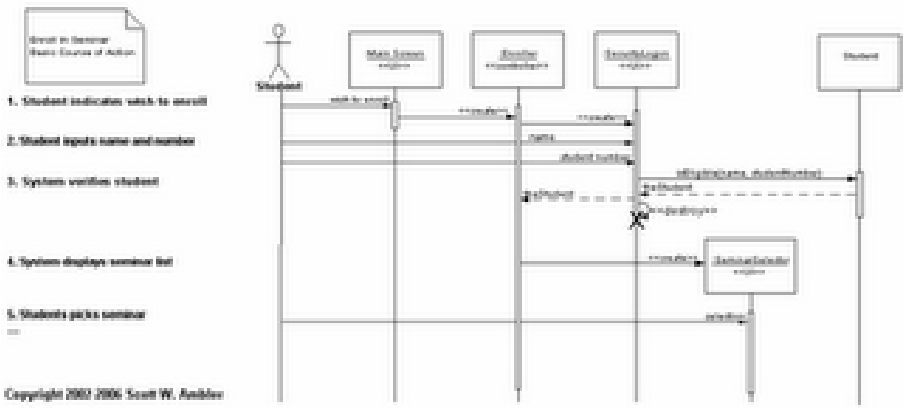
   b. Business use case; interaksi bisnis dengan konsumen atau kejadian nyata



Gambar II.13 Use Case Diagram
Sumber: Haviluddin, 2011:4

2. Sequence Diagram

   Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya sequence diagram adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan use case diagram (Haviluddin, 2011:4)



Gambar II.14 Sequence Case Diagram
Sumber: Haviluddin, 2011:4

II.11 Android

Android adalah sekumpulan perangkat lunak yang ditujukan bagi perangkat bergerak mencakup sistem operasi, middleware, dan aplikasi kunci. Android Standart Development Kid (SDK) menyediakan perlengkapan dan Application Programming Interface (API) yang diperlukan untuk mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java (Abdul Kadir, 2014).

Android dikembangkan oleh Google bersama Open Handset Alliance (OHA) yaitu aliansi perngkat seluler terbuka yang terdiri dari 47 perusahaan Hardware, *Software* dan perusaan telekomunikasi ditujukan untuk mengembangkan standar terbuka bagi perangkat selular (Abdul Kadir, 2014).

II.11.1 Sejarah dan Perkembangan Android

Pada mulanya terdapat berbagai macam sistem oprasi pada perangkat selular, diantaranya sistem operasi Symbian, Microsoft Windows Mobile, Mobile Linux, iPhone, dan sistem operasi lainnya. Namun diantara sistem operasi yang

ada belum mendukung standar dan penerbitan API yang dapat dimanfaatkan secara keseluruhan dan dengan biaya yang murah. Kemudian Google ikut berkecimpung di dalamnya dengan platform Android, yang menjanjikan keterbukaan, keterjangkauan, open source dan framework berkualitas (Abdul Kadir, 2014).

Pada tahun 2005, Google mengakuisisi perusahaan Android Inc. untuk mamulai perkembangan platform Android. Dimana terlibat pengembangan ini adalah Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Pada pertengahan 2007 sekelompok pemimpin industri bersama-sama membentuk analisis aliansi perangkat selular terbuka, Open Handset Alliance (OHA). Bagian dari tujuan aliansi ini adalah berinovasi dengan cepat dan menaggapi kebutuhan konsumen dengan lebih baik, dengan produk awalnya adalah platform Android. Dimana Android dirancang untuk melayani kebutuhan operator telekomunikasi, manufaktur handset, dan pengembangan aplikasi (Abdul Kadir, 2014).

Android pertama kali diluncurkan pada 5 November 2007, dan smartphone pertama yang menggunakan sistem operasi Android dikeluarkan oleh T-Mobile dengan sebutan G1 pada bulan September 2008. Hingga saat ini Android telah merilis beberapa versi Android untuk menyempurnakan versi sebelumnya. Selain berdsarkan penomoran, pada setiap versi Android terdapat kode nama berdasarkan nama-nama kue. Hingga saat ini sudah terdapat beberapa versi yang telah diluncurkan, diantaranya: versi 1.1 dirlis pada 9 maret 2009, versi 1.5 dirilis pada 30 April 2009 diberi nama *Cupcake,* versi 1.6 dirilis pada 15 September 2009 diberi nama *Donut,* versi 2.0 dirilis pada 26

Oktober 2009 diberi nama *Éclair*, versi 2.2 dirilis pada 20 Mei 2010 diberi nama *Froyo* (*Frozen Yoghurt*), versi 2.3 dirilis pada 6 Desember 2010 diberi nama *Gingerbread*, versi 3.0 dirilis pada Mei 2011 diberi nama *Honeycomb*, versi 4.0 dirilis pada 19 Oktober 2011 diberi nama ICS (*Ice Cream Sandwich*) (Abdul Kadir, 2014).

II.11.2 Kelebihan Android

Sudah banyak platform untuk perangkat selular saat ini, termasuk didalamnya Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiM0), dan banyak lagi. Namun ada beberapa hal yang menjadi kelebihan Android. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada platform lain, Android adalah yang pertama menggabungkan hal seperti berikut:

1. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap sistem karena berbasiskan Linux dan open source. Pembuat perangkat menyukai hal ini karena dapat membangun platform yang sesuai yang diinginkan tanpa harus membayar royality. Sementara pengembang software menyukai karena Android dapat digunakan diperangkat manapun dan tanpa terikat oleh vendor manapun.
2. Arsitektur komponen dasar Android terinspirasi dari teknologi internet Mashup. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.

3. Banyak dukungan service, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, database SQL, browser dan penggunaan peta. Semua itu sudah tertanam pada Android sehingga memudahkan dalam pengembangan aplikasi.
4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja sistem menjadi lebih stabil. Pengguna tak perlu kawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.
5. Dukungan grafis dan suarat terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh Flash menyatu dalam 3D menggunakan OpenGL memungkinkan membuat aplikasi maupun game yang berbeda.
6. Portabilitas aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan menggunakan bahas pemrograman Java dan dieksekusi oleh mesin virtual Dalvik, sehingga kode program portabel antara ARM, X86, dan arsitektur lainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard,* layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program.

(Abdul Kadir, 2014)

II.11.3 Java

Java adalah bahasa pemrograman yang disusun oleh James Gosling yang dibantu oleh rekan-rekannya seperti Patrick Naugton, Chris Warth, Ed Frank, dan Mike Sheridan di suatu perusahaan perangkat lunak yang bernama Sun Microsystem, pada tahun 1991. Bahasa pemrograman ini mula-mula diinisialisasi dengan nama "oak", namun pada tahun 1995 diganti namanya menjadi "Java" (Abdul Kadir, 2014).

Alasan utama pembentukan bahasa java adalah untuk membuat aplikasi-aplikasi yang dapat diletakkan diberbagai macam perangkat elektronik, seperti microwave oven dan remote control, sehingga Java harus bersifat portable atau yang sering disebut dengan platform independent (tidak tergantung pada platform). Itulah yang menyebabkan dalam dunia pemrograman Java, dikenal adanya istilah 'write once, run everywhere', yang berarti kode program hanya ditulis sekali, namun dapat dijalankan dibawah platform manapun, tanpa harus melakukan perubahan kode program (Abdul Kadir, 2014).

II.11.4 Arsitektur Java

Secara arsitektur, Java tidak berubah sedikitpun semenjak awal mula bahasa tersebut dirilis. Kompiler Java (yang disebut dengan **Javac** atau *Java Compiler*) akan mentransformasikan kode-kode dalam bahsa Java ke dalam suatu *bytecode*. Apa itu *bytecode*? *Bytecode* adalah sekumpulan perintah hasil kompilasi yang kemudian dapat dieksekusi melalui sebuah mesin komputer abstrak, yang disebut dengan JVM (*Java Virtual Machine*). JVM juga sering

dinamakan sebagai *interpreter,* karena sifatnya yang selalu menerjemahkan kode-kode yang tersimpan dalam *bytecode* dengan cara baris demi baris (Abdul Kadir, 2014).

II.12 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platformindependent*). 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. *Multi-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* (Adi Nugroho, 2008)

II.12.1 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 (Adi Nugroho, 2008).

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* UML2 tersedia untuk membuat

diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya (Adi Nugroho, 2008).

II.12.2 Android SDK

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di *release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada *platform* android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi-netral, android member anda kesempatan unutk membuat aplikasi yang kita butuhkan yang ubkan merupakan aplikasi bawaan Hadphone/*Smartphone* (Adi Nugroho, 2008). Beberapa fitur-fitur android yang paling penting adalah :

1. *Framework* : aplikasi yang mendukung pengganti komponen dan reusable. b. *Dalvik Virtual Machine* dioptimalkan untuk perangkat *mobile*
2. Integrated *Browser* verdasarkan engine *open source* WebKit.
3. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (Opsional Ekselerasi *hardware*)
4. SQLite untuk penyimpanan data.
5. Media *Support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PING, GIF), GSM Telephony.

6. *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *hardware*)
7. Kamera, GPS, Kompas, dan *Accelerometer* (tergantung *hardware*)
8. Lingkungan *Development* yang lengkap dan termasuk pernagkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE Eclipse.

Untuk source SDK Android ini dapat dilihat dan didownload langsung di situs resmi pengembang SDK Android di http://www.developer.android.com (Adi Nugroho, 2008).