BAB 2

TINJAUAN PUSTAKA

2.1 Citra

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra terbagi 2 yaitu ada citra yang bersifat analog dan ada citra yang bersifat digital. Citra analog adalah citra yang bersifat kontinu seperti gambar pada monitor televisi, foto sinar X, hasil CT Scan dll. Sedangkan pada citra digital adalah citra yang dapat diolah oleh komputer (T,Sutoyo et al. 2009: 9).

2.1.1 Citra Analog

Citra analog adalah citra yang bersifat kontinu, seperti gambar pada monitor televisi, foto sinar X, foto yang tercetak dikertas foto, lukisan, pemandangan alam, hasil CT *scan* dan lain sebagainya. Citra analog tidak dapat dipresentasikan dalam komputer sehingga tidak bisa diproses di komputer secara langsung. Oleh sebab itu, agar citra ini dapat diproses di komputer, proses konversi analog ke digital harus dilakukan terlebih dahulu. Citra analog dihasilkan dari alat-alat analog diantaranya adalah video kamera analog, kamera foto analog dan CT scan.

2.1.2 Citra Digital

Citra digital adalah sebuah citra f(x,y) yang telah di-diskretasi ke dalam koordinat spasial dan tingkat keabuan. Citra digital dinyatakan sebagai sebuah matrik NxM yang terdiri atas baris dan kolom untuk menyatakan sebuah titik pada citra dan elemen nilai matrik yang berupa nilai diskret menyatakan tingkat

keabuan pada titik tersebut. Citra digital tiap elemen dikenal sebagai elemen gambar (*picture element*) atau pixel.

(baris/tinggi = N, kolom/lebar = M)

N = jumlah baris $0 = y = N - 1$

M = jumlah kolom $0 = x = M - 1$

L = maksimal warna intensitas (derajat keabuan /*gray level*) $0 = f(x,y) = L - 1$

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix} \quad (2.1)$$

Untuk mendapatkan suatu citra digital diperlukan suatu konversi sehingga dapat diproses oleh komputer. Proses konversi tersebut dengan membuat kisi-kisi arah horizontal dan vertical sehingga diperoleh gambar dalam bentuk array 2 dimensi.

Proses tersebut disebut proses digitasi atau sampling. Semakin tinggi resolusi berarti semakin kecil ukuran pixelnya, berarti semakin halus gambar yang diperoleh karena informasi yang hilang akibat pengelompokan tingkat keabuan pada proses pembuatan kisi-kisi semakin kecil. Tetapi membutuhkan tempat penyimpanan bit yang makin besar pula.

2.1.2.1 Elemen – elemen Citra Digital

Citra digital mengandung sejumlah elemen-elemen dasar. Elemen-elemen dasar inilah yang dimanipulasi dalam pengolahan citra. Elemen-elemen yang penting diantaranya adalah sebagai berikut :

1. **Kecerahan (*brightness*)**

Kecerahan disebut juga sebagai intensitas cahaya yang dipancarkan pixel dari citra yang dapat ditangkap pleh sistem penglihatan. Kecerahan pada suatu titik (piksel) di dalam suatu citra sebenarnya adalah intensitas rata-rata dari suatu area yang melingkupinya.

2. **Kontras *(contrast)***

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

3. **Kontur (*countour*)**

Kontur adalah keadaan yang ditimbulkan oleh perubahan intensitas pada *pixel-pixel* yang bertetangga. Karena adanya perubahan intensitas inilah, maka tepi-tepi (*edge*) objek pada citra dapat dideteksi.

4. **Warna (*colour*)**

Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna mempunyai panjang gelombang (λ) yang berbeda-beda. Warna yang diterima oleh sistem visual (mata) merupakan hasil kombinasi cahaya dengan panjang gelombang yang berbeda-beda. Kombinasi warna yang memberikan rentang warna yang paling lebar adalah *red (R), green (g), blue (B).*

5. **Bentuk (*shape*)**

Bentuk adalah properti intrinstik dari objek tiga dimensi. Bentuk merupakan properti intrinsik utama untuk sistem visual manusia karena lebih sering menginterpretasikan suatu objek berdasarkan bentuknya daripada elemen lainya.

6. **Tekstur (*texture*)**

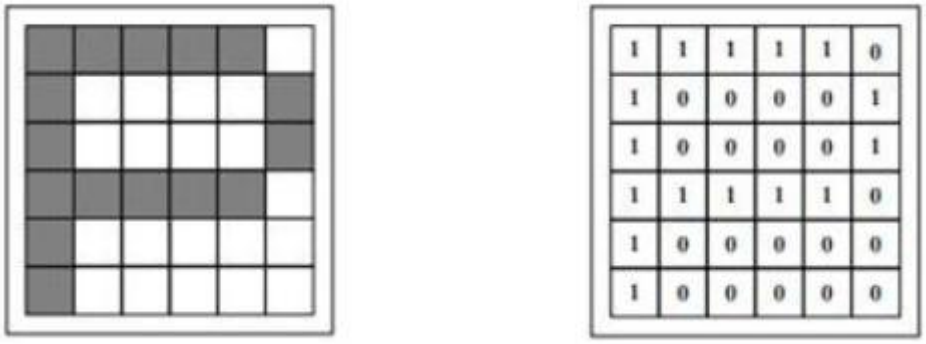
Tekstur dicirikan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan *pixel-pixel*, Tekstur merupakan karakteristik untuk menganalisis permukaan berbagai jernis citra objek.

2.1.2.2 Jenis Citra Digital

Berdasarkan format penyimpanan nilai warnanya Citra Digital terbagi menjadi lima macam [7] yaitu :

1. **Citra Biner (Monokrom)**

   Citra yang hanya memiliki dua nilai yaitu 1 dan 0. Citra biner juga disebut sebagai citra B&W *(black and white)* atau citra monokrom, setiap *pixel* pada citra biner cukup direpresentasikan dengan 1 bit.



Gambar 2.1 Citra biner dan array nilai matrik dari citra biner

2. **Citra Grayscale**

Pada citra dengan tipe *grayscale*, setiap piksel mewakili derajat keabuan dengan nilai antara 0 (hitam) samapai 255 (putih). Nilai intensitas paling rendah merepresentasikan wrna hitam dan nilai intensitas paling tinggi merepresentasikan warna putih. Pada jangkauan nilai 0 sampai 255, ini berarti bahwa piksel memiliki ukuran 8 bit atau 1 byte.



Gambar 2.2 Gambar *grayscale*

3. Citra Warna 8 bit

Setiap *pixel* dari citra warna (8 bit) hanya di wakili oleh 8 bit jumlah warna maksimum yang dapat digunakan adalah 256 warna. Ada dua jenis citra warna 8 bit. Pertama, cita warna 8 bit dengan menggunakan palet warna 256 dengan setiap paletnya memiliki pemetaan nilai (*colormap*) RGB tertentu. Model ini lebih sering digunakan. Kedua, setiap piksel memiliki format 8 bit sebagai berikut,

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
R	R	R	G	G	G	B	B

Gambar 2.3 Format citra 8 bit

4. Citra Warna 16 bit

Citra warna 16 bit (biasanya disebut sebagai citra *highcolor*) dengan setiap pikselnya diwakili dengan 2 byte memory (16 bit). Warna 16 bit memiliki 65.635 warna. Dalam formasi bitnya, nilai merah dan biru mengambil tempat di 5 bit di kanan dan kiri. Komponen hijau memiliki 5 bit ditambah 1 bit eksra. Pemilihan komponen hijau dengan deret 6 bit dikarenakan penglihatan manusia lebih sensitif terhadap warna hijau.

Bit -15	Bit -14	Bit -13	Bit -12	Bit -11	Bit -10	Bit -9	Bit -8	Bit -7	Bit -6	Bit -5	Bit -4	Bit -3	Bit -2	Bit -1	Bit -0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

Gambar 2.4 Format citra 16 bit

5. Citra Warna 24 bit

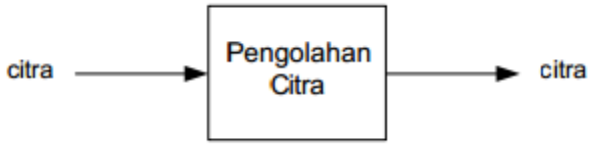
Setiap *pixel* dari citra warna 24 bit diwakili sehingga total 16.777.216 variasi warna. Variasi ini sudah lebih dari cukup untuk memvisualisasikan seluruh warna yang dapat dilihat penglihatan manusia. Penglihatan manusia dipercaya hanya dapat membedakan hingga 10 juta warna saja. Setiap poin informasi *pixel* (RGB) disimpan ke dalam 1 byte data. 8 bit pertama menyimpan nilai baru, kemudian diikuti dengan hijau pada 8 bit kedua dan pada 8 bit terakhir merupakan warna merah.

2.2 Pengolahan Citra

Pengolahan citra (*image processing*) pengolahan suatu citra (gambar) dengan menggunakan komputer secara khusus, untuk menghasilkan suatu citra yang lain Sesuai dengan perkembangan komputer itu sendiri, pengolahan citra mempunyai tiga tujuan utama, yaitu sebagai berikut:

1. Memperbaiki kualitas gambar dilihat dari aspek radiometrik (peningkatan kontras, transformasi warna, restorasi citra) dan dari aspek geometrik (rotasi, translasi, skala, transformasi geometrik).
2. Melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkadang pada citra.
3. Melakukan kompresi datau reduksi data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data.

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 2.5 Proses pengolahan citra

Pengubahan kontras citra seperti pada Gambar 2.5 adalah contoh operasi pengolahan citra. Contoh operasi pengolahan citra lainnya adalah penghilangan derau (*noise*) pada citra Lena (Gambar 2.6). Citra Lena yang di sebelah kiri mengandung derau berupa bintik-bintik putih (derau). Dengan operasi penapisan (*filtering*), derau pada citra masukan ini dapat dikurangi sehingga dihasilkan citra Lena yang kualitasnya lebih baik.



(a) (b)

Gambar 2.6 (a) Citra Lena yang mengandung derau, (b) hasil dari operasi penapisan derau

2.3 Restorasi Citra

Restorasi citra adalah suatu langkah untuk mendapatkan citra yang lebih jelas dari citra yang terdegradasi dengan hanya mengetahui beberapa faktor degradasi dari citra tersebut. Restorasi citra berbeda dengan peningkatan kualitas citra (*image enhancement*) meskipun keduanya sama-sama bertujuan untuk memperbaiki kualitas citra. *Image enhancement* lebih banyak berhubungan dengan penajaman dari fitur tertentu dalam citra, sedangkan restorasi citra memanfaatkan pengetahuan tentang proses terjadinya degradasi untuk memperoleh kembali citra asal.

Restorasi citra berkaitan dengan penghilangan atau pengurangan degradasi pada citra yang terjadi karena proses akuisisi citra. Degradasi yang dimaksud termasuk derau (yang merupakan eror dalam nilai piksel) atau efek optis misalnya

blur (kabur) akibat kamera yang tidak fokus atau karena gerakan kamera. Teknik restorasi citra meliputi operasi neighbourhood dan juga penggunaan proses-proses pada domain frekuensi.

Model Degradasi sebuah citra dapat dimodelkan sebagai berikut:

1. $g(x,y)=f(x,y)*h(x,y)$; misalnya proses blurring menggunakan filter average dengan f(x,y) adalah citra asli, * menyatakan operasi spatial, h (x,y) menyatakan filter yang digunakan, dan g(x,y) adalah citra yang terdegradasi.
2. $g(x,y)=f(x,y)*h(x,y)+n(x,y)$; dengan n(x,y) adalah derau yang dimodelkan sebagai fungsi aditif (*random errors*).

2.4 *Inpainting*

Inpainting adalah suatu teknik untuk memodifikasi citra dalam bentuk yang tidak terdeteksi, dalam hal seni murni (lukis). Tujuan dan aplikasi dari kebanyakan inpainting, mulai dari restorasi lukisan dan foto yang rusak untuk menghilangkan atau mengganti obyek yang dipilih.[4]

Teknik *inpainting* berusaha menghasilkan citra keluaran dengan kualitas bagus atau rasional di mana daerah yang dihapus tidak terlihat jelas. Penelitian terkait berbagai teknik inpainting telah dilakukan.Setiap penelitian dibagi ke dalam dua pendekatan yaitu *pixel synthesized* (pixel disintesis untuk menyesuaikan spektrum citra) dan *exemplar-based* (dibuat potongan citra sumber yang sesuai dengan sekitarnya). Setiap pendekatan dibagi dalam dua skema yaitu *structure oriented* (mengembalikan struktur linear target yang dihapus) dan *texture oriented* (mengembalikan tekstur daerah yang dihapus).[1]

Pada mulanya *inpainting* merupakan teknik yang digunakan para ahli seni untuk merestorasi lukisan yang rusak karena usia, goresan, jamur, atau hal lainnya sehingga lukisan tersebut menjadi tidak seperti aslinya. Bertalmio dan kawan-kawan (Bertalmio,et.al,2001), merupakan pencetus pertama dalam usaha menuangkan ide *inpainting* dalam dunia digital, sehingga dikenal sebagai *digital inpainting*. Teknik klasik merupakan teknik pertama kali yang dikembangkan oleh Bertalmio dan kawankawan, yang menggunakan transformasi Laplace (Sapiro, 2005). Grossauer dan Scherzer (Grossauer dan Scherzer, 2005) menggunakan teknik persamaan Ginzburg-Landau pada ruang 2 dimensi dan 3 dimensi untuk melakukan *inpainting*. Pada kesempatan lain Grossauer mengembangkan teknik dengan mengkombinasikan persamaan diferensial partial (PDP) dan sintesis tekstur (*Texture Synthesis*/TS) (Grossauer,2005). Disini Grossauer mencoba menyempurnakan kombinasi algoritma dasar PDP dengan algoritma TS untuk mengisi daerah yang akan di *inpaint*. Telea (Telea,2005) mengembangkan algoritma yang didasarkan pada apa yang disebut *Level Sets and Fast Marching Method* yang di cetuskan oleh Sethian (Sethian,1994,1996).

Algoritma yang dikenal sederhana dan cepat, juga dapat digunakan untuk menyelesaikan permasalahan *inpainting* di kenalkan oleh Oliveira dan kawan-kawan (Oliveira, et.al,2001), yang dikenal dengan *Fast Digital Image Inpainting*. Cant mencoba memecahkan permasalah *inpainting* dengan metode multi skala untuk otomatisasi *inpainting* (Cant,2003). Criminisi dan Toyama, menampilkan algoritma yang digunakan untuk menghilangkan obyek besar yang didasarkan

pada perhitungan warna menggunakan *exemplar sintesis* (Criminisi dan Toyama, 2003,2004).

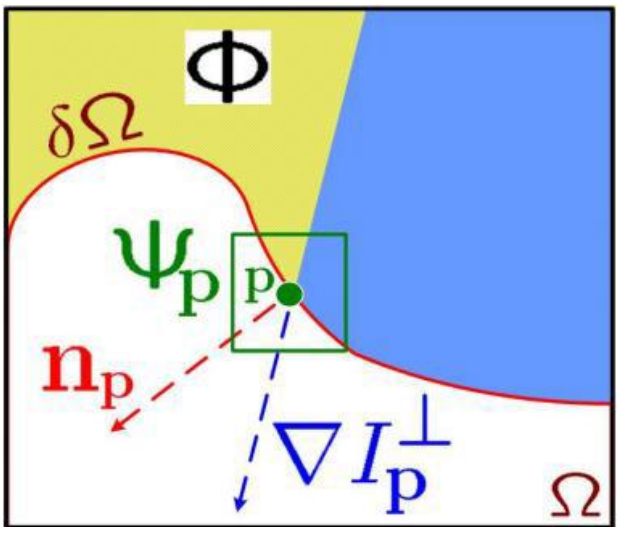
Jia dan Tang (Jia dan Tang, 2005) memandang *inpainting* dari sisi sintesis tekstur, yaitu dengan menggunakan pendekatan ND Tensor Voting mencoba mentranslasikan informasi warna citra dan tekstur yang disesuaikan dalam tensor ND. ND Tensor Voting merupakan suatu metode sintesis yang secara otomatis menyimpulkan warna yang hilang dan informasi tekstur dari gambar 2D. Demanet, Song dan Chan (Demanet et.al,2005), memberikan solusi *inpainting* dengan pendekatan yang disebut Peta Tekstur yang saling berhubungan, yang didefinisikan sebagai kaitan antar piksel yang kosong atau hilang dengan piksel lain dimana nilainya diambil dari dalam benih/bibit citra aslinya. Igehy dan Pereira (Igehy dan Pereira, 1997) menyajikan teknik yang didasarkan pada Hegeer dan Bergen (Hegeer dan Bergen, 1995), yaitu dengan algoritma sintesis tekstur dan mengkombinasikan dengan komposisi citra untuk dapat menghasilkan restorasi yang lebih baik.

2.5 *Exemplar Based*

Exemplar-based adalah salah satu metode inpainting yang dikenalkan oleh Criminisi [3][4]. Pada awalnya Bertalmio [1] memperkenalkan dua pendekatan yaitu *pixel synthesized* dengan skema *structured oriented* untuk mengisi daerah yang hilang berdasarkan informasi ishopote (propagasi struktur linier). Namun ketika diaplikasikan pada daerah target yang besar, akan dihasilkan citra kabur. Selanjutnya Bertalmio menggabungkan dua pendekatan yaitu *pixel synthesized* dengan skema *structure oriented* dan *exemplar-based* dengan skema *texture*

oriented. Citra sumber dibagi menjadi dua. Citra pertama yang menangkap informasi struktur diproses dengan *pixel synthesized* skema *structure oriented*. Sedangkan citra kedua yang menangkap informasi tekstur diproses dengan pendekatan *exemplar-based* skema *texture oriented*. Selanjutnya, hasil keduanya digabung. Terdapat empat proses penting yang dilakukan yaitu dekomposisi citra, *inpainting* dengan *pixel synthesized*, *inpainting* dengan *exemplar-based* dan penggabungan citra.

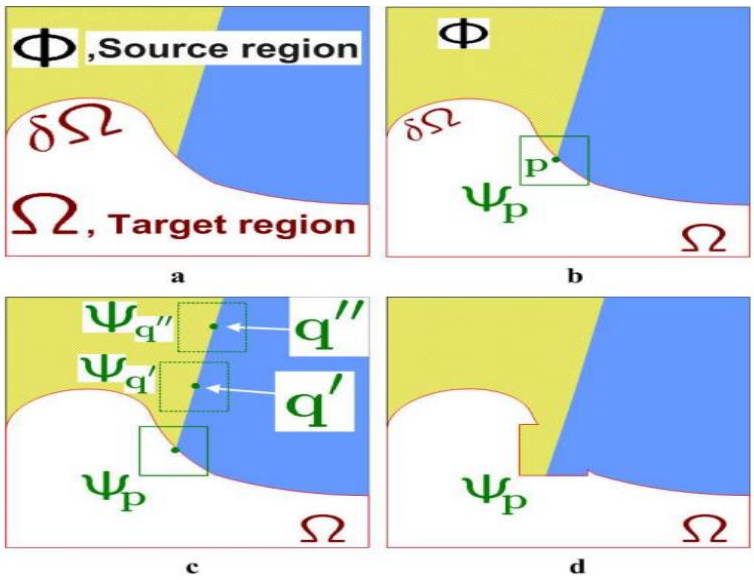
Kemudian Criminisi memperkenalkan pendekatan *exemplar-based* yang menggabungkan skema *structure oriented* dengan *texture oriented*. Metode ini membutuhkan satu proses untuk melakukan sintesis tekstur sekaligus perbaikan struktur linear. *Inpainting* dilakukan dengan memanfaatkan informasi tekstur (direpresentasikan dalam *confidence term*) dan informasi *isophote* atau struktur (*data term*). Kedua informasi tersebut dimanfaatkan untuk menentukan prioritas urutan pengisian daerah target.



Gambar 2.7 Konvensi simbol metode

Gambar 2.7 menjelaskan tentang simbol-simbol yang digunakan dalam metode ini. Simbol Ω merupakan simbol dari daerah target pada citra yang akan diproses. Simbol δΩ digunakan untuk melambangkan kontur. Kontur dihasilkan dari hasil penyambungan *polygon-polygon* yang telah ditetapkan oleh *user* sebelumnya. Simbol ϕ melambangkan daerah sumber dari citra yang akan dilakukan inpainting. Simbol ψp melambangkan *patch* pada titik *p*. Simbol *np* merupakan simbol untuk titik normal dari kontur.

Alur proses metode *exemplar based inpainting* dijelaskan seperti gambar 2.8 berikut :



Gambar 2.8 Alur proses *inpainting*

Gambar 2.8 menjelaskan bagaimana alur *inpainting* dari metode ini bekerja. Yang pertama dilakukan adalah menentukan daerah target pada citra yang akan direkontruksi. Setelah itu didapat daerah target, daerah sumber dan kontur. Kemudian menentukan *patch* dan dicari prioritas *patch* yang terbaik ke arah luar

dari kontur. Pencarian pada suatu daerah sumber dibutuhkan infomasi-informasi pixel daerah sekitar titik kontrol dikontur dengan ukuran 3x3. Setelah mendapatkan informasi dari potongan 3x3 tersebut akan dicari potongan daerah sumber yang memiliki informasi mirip dengan potongan didaerah kontur tadi. Jarak pencarian disini dilakukan dari patch dikontur ke arah tegak lurus sampai batas daerah sumber terluar.

Langkah-langkah pada metode *exemplar based* dijelaskan seperti di bawah ini :

1. Penghitungan prioritas *patch*
   a. Hitung *Confidence term C(p)* dengan persamaan sebagai berikut :

      $$C(p) = \frac{\sum_{q \in \psi_p \cap \bar{\Omega}} C(q)}{|\psi p|} \quad (2.2)$$

      dimana:

      $|\psi_p|$: luas ψ_p

      Pada saat inisialisasi:

      $$C(p) = 1 \leftrightarrow p \in \phi \text{ dan } C(p) = 0 \leftrightarrow p \in \Omega \quad (2.3)$$

   b. Hitung *data term* dengan persamaan sebagai berikut

      $$D(p) = \frac{|\nabla I_p^{\perp} . np|}{\alpha} \quad (2.4)$$

      dimana

      α : factor normalisasi (255 untuk gambar *grey-level*).

      $\nabla I \frac{1}{p}$: *isophote* (arah dan intensitas) pada titik *p*

   c. Hitung prioritas *patch* tertinggi dengan persamaan :

      $$P(\mathrm{p}) = C(\mathrm{p}) * D(\mathrm{p}) \quad (2.5)$$

2. Pencarian *exemplar*

Dicari potongan yang sesuai untuk mengisi daerah target dengan persamaan sebagai berikut :

$$\psi\bar{q} = arg\ \min_{\psi q \in \phi}\ d(\psi\bar{p}, \psi q) \qquad (2.6)$$

dimana $\boldsymbol{d(\psi\bar{p}, \psi q)}$ adalah jarak Eucledian antara 2 patch $\boldsymbol{\psi\bar{p}}$ dan $\boldsymbol{\psi q}$.

$\boldsymbol{\phi}$: seluruh region yang berada di daerah sumber dengan titik pusat q dan radius r

(ψp) : potongan yang diambil dari citra biner hasil kontruksi dengan posisi sama dengan ψp

3. Penyisipan *exemplar*

   Potongan replika disisipkan di daerah target.

4. Update *Image*

   Nilai *confidence term* diperbaharui untuk potongan-potongan baru yang muncul akibat penyisipan *exemplar*. *Update* dilakukan dengan persamaan sebagai berikut :

   $$C(p)\ \forall p \mid p \in\ \psi\bar{p} \cap \Omega \qquad (2.7)$$

   Keterangan :

   Nilai p lama akan digantikan dengan nilai p baru dimana nilai p baru merupakan elemen nilai prioritas *patch* tertinggi.

2.6 Pemrograman Berorientasi Objek

Pengembangan sistem berorientasi objek berbeda dari pengembangan sistem konvensional (aliran data) yang memandang perangkat lunak sebagai fungsi dan data yang tersimpan. Perbedaan pengembangan sistem berorientasi objek dengan pengembangan sistem konvensional yaitu pada pengembangan sistem secara

konvensional, kebanyakan berfokus pada fungsi, tetapi terdapat pendekatan yang berfokus pada data terutama pada basis data dan pemodelan informasi, sedangkan pengembangan berorientasi objek berpusat pada objek yang mengkombinasikan data dan fungsi keduanya sekaligus [9].

Pengembangan sistem berorientasi objek mempunyai kelebihan sebagai berikut:

1. Pendekatan objek menuntut penggunaan ulang komponen-komponen program sebelumnya.
2. Perangkat lunak yang dikembangkan dengan berorientasi objek mempermudah pemeliharaan.
3. Sistem berorentasi objek lebih mudah diadaptasi menjadi sistem yang lebih besar dibuat dengan cara membangun subsistem-subsistem yang dapat digunakan.

2.7 Pemodelan Sistem

Dalam pembuatan pemodelan sistem berbasis objek diperlukan tool yang digunakan untuk membangung perangkat lunak, yaitu *Unified Modeling Language* (UML).

2.7.1 Pemodelan Menggunakan UML

UML (Unified Modeling Language) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berorientasi objek. UML sendiri terdiri atas pengelompokkan diagram-diagram sistem menurut aspek atau sudut pandang tertentu. Diagram adalah yang menggambarkan permasalahan maupun solusi dari permasalahan suatu model. UML mempunyai 9 diagram, yaitu; *use-case*, *class*,

object, state, sequence, collaboration, activity, component, dan *deployment* diagram.

1. *Use Case Diagram*, menggambarkan sekelompok use cases dan aktor yang disertai dengan hubungan diantaranya. Diagram use cases ini menjelaskan dan menerangkan kebutuhan / requirement yang diinginkan / dikehendaki user/pengguna, serta sangat berguna dalam menentukan struktur organisasi dan model dari pada sebuah sistem.

   Tujuan utama pemodelan *use-case* adalah:

   a. Memutuskan dan mendeskripsikan kebutuhan-kebutuhan fungsional sistem.
   b. Memberikan deskripsi jelas dan konsisten dari apa yang seharusnya dilakukan,sehingga model *use case* digunakan di seluruh proses pengembangan untuk komunikasi dan menyediakan basis untuk pemodelan berikutnya yang mengacu sistem harus memberikan fungsionalitas yang dimodelkan pada *use-case*.
   c. Menyediakan basis untuk melakukan pengujian sistem yang memverifikasi sistem.Menguji apakah sistem telah memberikan fungsionalitas yang diminta.
   d. Menyediakan kemampuan melacak kebutuhan fingsionalitas menjadi kelas-kelas dan operasi-operasi aktual di sistem.Untuk menyederhanakan perubahan dan ekstensi ke sistem dengan mengubah model *use-case* dan kemudian melacak *use-case* yang dipengaruhi ke perancangan dan implementasi sistem.

   Elemen diagram *use-case* adalah:

a) Aktor

   Aktor adalah pemakai sistem, dapat berupa manusia atau sistem terotomatisasi lain.Aktor adalah sesuatu atau seseorang yang

berinteraksi dengan sistem, yaitu siapa atau apa yang menggunakan sistem.Yang dimaksud dengan berinteraksi adalah aktor mengirim atau menerima pesan ke atau dari sistem, atau mempertukarkan informasi dengan sistem. Dalam perspektif aktor, use-case melakukan sesuatu yang berharga bagi aktor.

Aktor adalah tipe (kelas), bukan instan. Aktor merepresentasikan peran bukan pemakai individu dari sistem.Aktor mempunyai nama. Nama yang dipilih seharusnya menyatakan peran aktor.

b) Use-Case

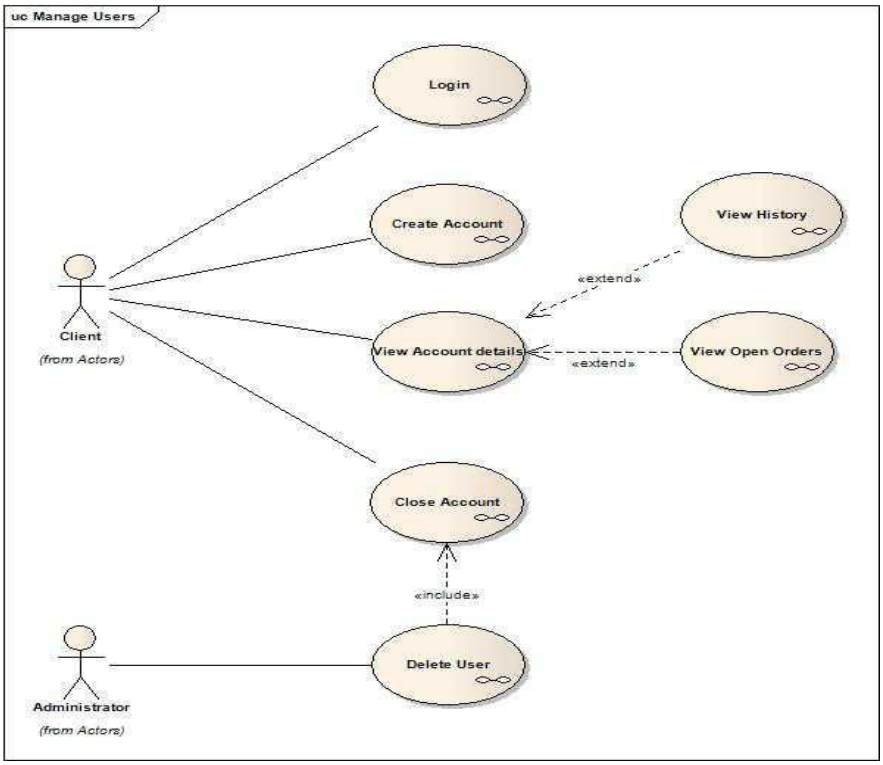
Use-case adalah cara spesifik penggunaan sistem oleh aktor.Ciri-ciri dari *use-case* adalah:

1. Pola perilaku yang harus dipenuhi oleh sistem.
2. Sekuen transaksi terhubung yang dilakukan aktor dan sistem.
3. Memberikan sesuatu yang berharga bagi aktor.

c) Hubungan antar *use-case*.

Keterhubungan antar *use-case* dengan *user-case* lain berupa generalisasi antar *use-case*, yaitu:

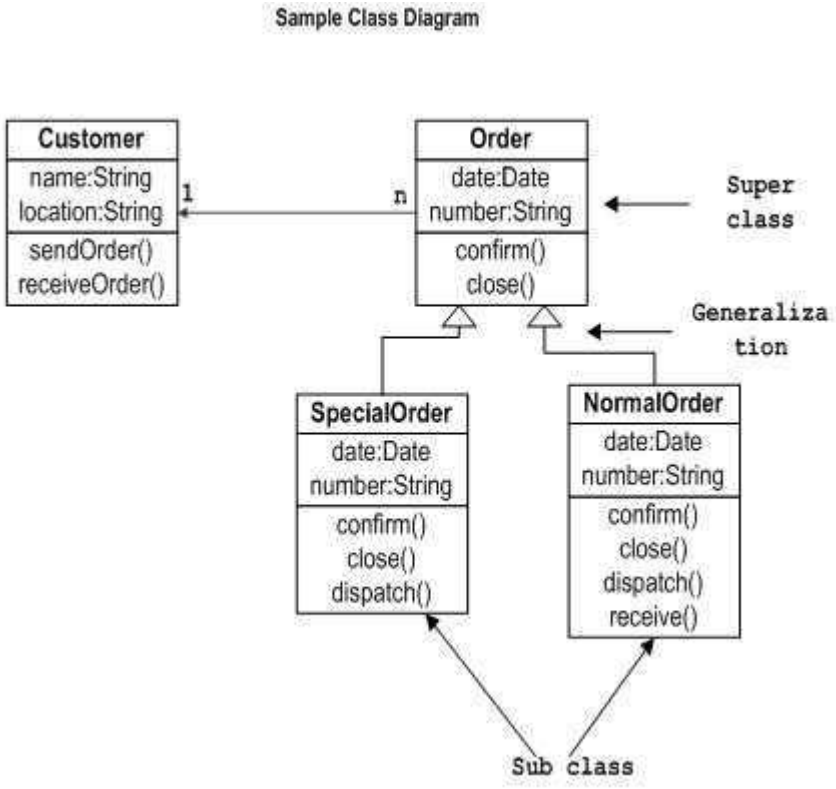
1. Include, perilaku *use-case* merupakan bagian dari *use-case* yang lain.
2. Extend, perilaku *use-case* memperluas perilaku *use-case* yang lain.



Gambar 2.9 Contoh *Use Case diagram*

2. *Class Diagram* adalah bagan yang memperlihatkan struktur statis dari kelas aktual di dalam sistem.

   Pemodelan kelas menunjukkan kelas-kelas yang ada di sistem dan hubungan antar kelas-kelas itu, atribut-atribut dan operasi-operasi di kelas-kelas.

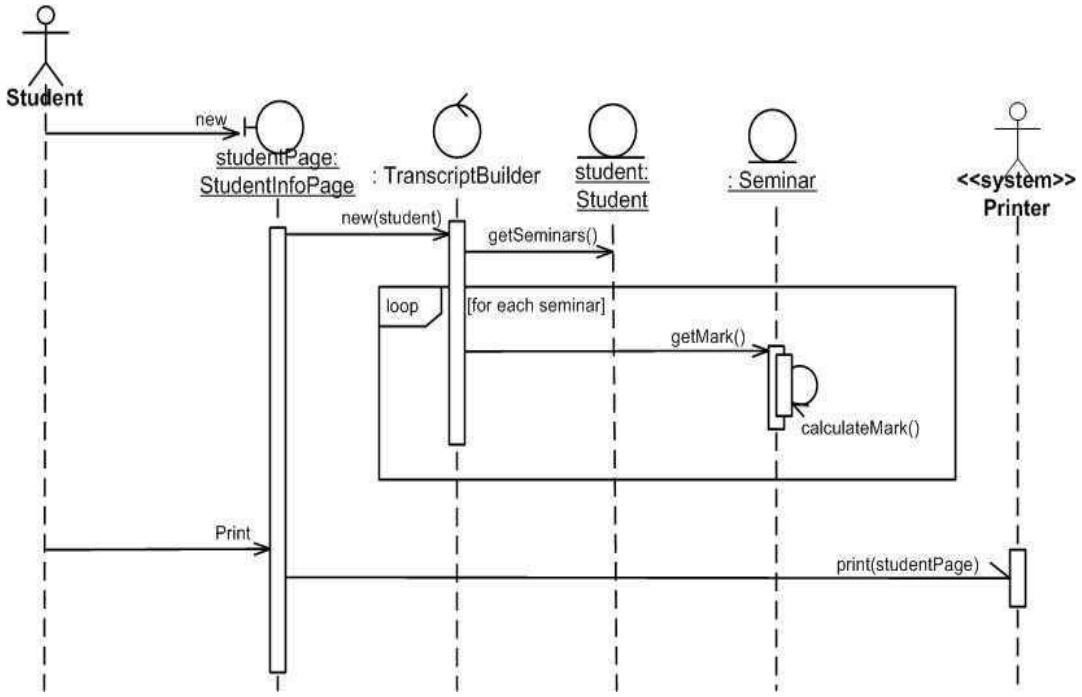


Gambar 2.10 Contoh *Class diagram*

3. *Sequence Diagram*, yang memperlihatkan kolaborasi dinamis antara objek-objek dengan suatu urutan pesan (*a sequence of message*) antar objek tersebut.

   Diagram sekuen digunakan untuk:

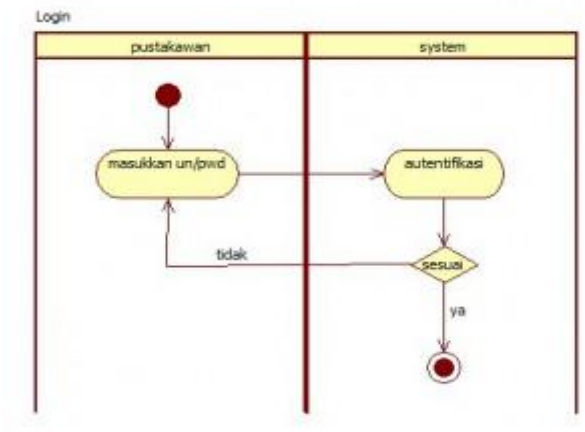
   1. Overview perilaku sistem.
   2. Menunjukkan objek-objek yang diperlukan.
   3. Mendokumentasikan skenario dari suatu diagram use case
   4. Memeriksa jalur-jalur pengaksesan



Gambar 2.11 Contoh *Sequence diagram*

6. *Activity Diagram* adalah bagan yang memperlihatkan aliran urutan aktifitas.

Berikut merupakan contoh dari *activity diagram* Login :



Gambar 2.12 Contoh *Activity Diagram*

2.8 Eclipse IDE

Eclipse adalah sebuah IDE (Integrated Development Environment) untukmengembangkan perangkat lunak dan dapat dijalankan di semua platform (platformindependent). Berikut ini adalah sifat dari Eclipse:

a. Multi-platform: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
b. 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.
c. 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.

2.9 Pengujian *Blackbox*

Pengujian *Blackbox* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.

Ciri-ciri Pengujian *Blackbox*:

1. Pengujian *Blackbox* berfokus pada kebutuhan fungsional pada software, berdasarkan pada spesifikasi kebutuhan dari software.

2. Pengujian *Blackbox* bukan teknik alternatif daripada white box testing. Lebih daripada itu, ia merupakan pendekatan pelengkap dalam mencakup error dengan kelas yang berbeda dari metode white box testing.
3. Pengujian *Blackbox* melakukan pengujian tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites. juga disebut sebagai behavioral testing, specification-based testing, input/output testing atau *functional testing*

2.10 Pengujian Citra

Kriteria penilaian citra secara obyektif berdasarkan pengukuran *error* pada citra yang diuji. Parameter-parameter yang digunakan sebagai kriteria penilaian obyektif adalah sebagai berikut :

1. *Mean Square Error (MSE)*

   MSE adalah nilai error kuadrat rata-rata antara citra asli dengan citra manipulasi (dalam kasus steganografi ; MSE adalah nilai error kuadrat rata-rata antara citra asli (cover-image) dengan citra hasil penyisipan (stego-image)

   Rumus MSE dapat dituliskan sebagai berikut :

$$\text{MSE} = \frac{1}{M.N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(f(x,y) - g(x,y) \right)^2 \qquad (2.8)$$

2. *Peak Signal to Noise Ratio (PSNR)*

   Peak Signal to Noise Ratio (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR merupakan parameter standar untuk menilai kualitas suatu citra secara obyektif dengan

membandingkan *noise* terhadap sinyal puncak. PSNR biasanya diukur dalam satuan desibel.

Rumus PSNR adalah sebagai berikut :

$$\text{PSNR} = 20.\log 10 \left(\frac{255}{\sqrt{MSE}} \right) \qquad (2.9)$$