BAB 2
LANDASAN TEORI

2.1 Kecerdasan Buatan

Kecerdasan buatan berasal dari bahasa Inggris "Artificial Intelligence" atau disingkat AI, yaitu intelligence adalah kata sifat yang berarti cerdas, sedangkan artificial artinya buatan. Kecerdasan yang dimaksud disini merajuk pada mesin yang mampu berpikir, menimbang tindakan yang diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

Menurut Alan Turing, ahli matematika berkebangsaan Inggris yang dijuluki bapak komputer modern dan pembongkar sandi Nazi dalam era Perang Dunia II 1950, menetapkan definisi artificial intelligence: "Jika komputer tidak dapat dibedakan dengan manusia saat berbicara melalui terminal komputer, maka bisa dikatakan komputer itu cerdas, mempunyai kecerdasan".

Menurut John Carthy dari Stanford mendefinisikan kecerdasan sebagai "Kemampuan untuk mencapai sukses dalam menyelesaikan suatu permasalahan". Menurut Herbert Alexander Simon (June 15, 1916-February 9, 2001): "Kecerdasan buatan (artificial intelligence) merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas".

Menurut Rich and Knight (1991): "Kecerdasan buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang padasaat ini dapat dilakukan lebih baik oleh manusia".

Dari sini dapat dikatakan bahwa: cerdas adalah memiliki pengetahuan, pengalaman, dan penalaran untuk membuat keputusan dan mengambil tindakan. Jadi, agar mesin bisa cerdas (bertindak seperti manusia) maka harus diberi bekal pengetahuan dan diberi kemampuan untuk menalar [6].

2.1.1 Keuntungan Kecerdasan Buatan

Beberapa keuntungan kecerdasan buatan yaitu [7]:

1. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang

pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.

2. Kecerdasan buatan lebih mudah diduplikasi dan disebar. Men-*transfer* pengetahuan manusia dari satu orang ke orang lain butuh proses dan waktu lama. Disamping itu satu keahlian tidak akan pernah bisa diduplikasi secara lengkap. Sedangkan jika pengetahuan terletak pada satu sistem komputer, pengetahuan tersebut dapat di-*transfer* atau disalin dengan mudah dan cepat dari satu komputer ke komputer lain.
3. Kecerdasan buatan lebih murah dibandin dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebih murah dibanding harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam waktu yang sangat lama.
4. Kecerdasan buatan bersifat konsisten. Hal ini disebabkan karena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami senantiasa berubah – ubah.
5. Kecerdasan buatan dapat didokumentasikan. Keputusan yang dibuat komputer dapat didkomentasikan dengan mudah dengan melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibandingkan dengan kecerdasan alami.
7. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

2.1.2 Lingkup Utama Kecerdasan Buatan

Lingkup utama kecerdasan buatan antara lain [6]:

1. Sistem pakar. Komputer digunakan sebagai saran untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki para pakar.

2. Pengolahan bahasa alami. Dengan pengolahan bahasa alami ini diharapkan user mampu berkomunikasi dengan komputer dengan menggunakan bahasa sehari – hari.
3. Pengenalan ucapan. Melalui pengenalan ucapan diharapkan manusia mampu berkomunikasi dengan komputer dengan menggunakan suara.
4. Robotika dan sensor.
5. *Computer Vision.* Mencoba untuk mengintrepetasikan gambar atau objek-objek tampak melalui komputer.
6. *Intelligence Computer Aid Instruction.* Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar.

2.1.3 Penerapan Kecerdasan Buatan

Kecerdasan buatan dapat diterapkan pada beberapa bidang diantaranya [6]:

1. Visualisasi Komputer

   Kecerdasan buatan pada bidang visualisasi komputer ini memungkinkan sebuah sistem komputer mengenali gambar input. Contohnya mengenali sebuah pola pada suatu gambar.
2. Pengenalan Suara

   Kecerdasan buatan pada pengenalan suara ini dapat mengenali suara manusia. Cara mengenali suara ini dengan mencocokanya pada acuan yang telah diprogramkan terlebih dahulu. Contohnya perintah komputer dengan menggunakan suara user.
3. Sistem Pakar

   Kecerdasan buatan pada sistem pakar ini memungkinkan sebuah sistem komputer memiliki cara berpikir dan penalaran seorang ahli dalam mengambil keputusan, untuk memecahkan masalah yang ada pada saat itu. Contohnya program komputer yang dapat mendiagnosa penyakit dengan memasukan gejala – gejala yang dialami pasien.
4. Permainan Kecerdasan

   Kecerdasan buatan pada permainan ini memungkinkan sebuah sistem komputer untuk memiliki cara berpikir manusia dalam bermain. Contohnya

permainan yang memiliki fasilitas orang melawan komputer. Komputer sudah diprogram sedemikian rupa agar memiliki cara bermain seperti seorang manusia bukan bisa melebihi manusia.

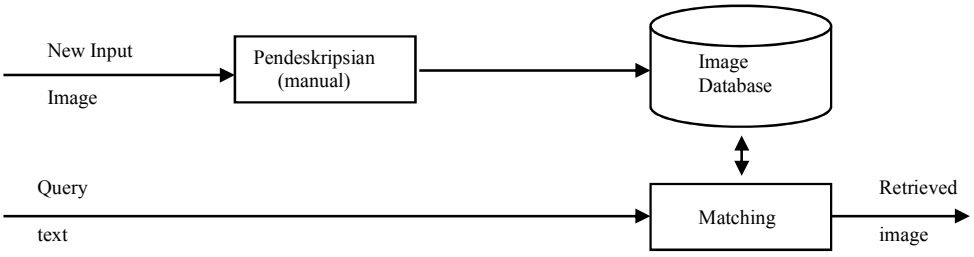
2.2 Temu Kembali Informasi (*Information Retrieval*)

Information Retrieval (Temu kembali Informasi) atau sering dikenal dengan Information Retrieval System (IRS) media layanan yang dapat digunakan oleh pengguna untuk mendapatkan informasi atau sumber informasi sesuai dengan pencarian yang dilakukan. Hasil informasi yang diperoleh dengan menggunakan kata kunci (keyword) bisa sesuai atau tidak sesuai dengan keinginan pengguna [8].

2.3 Temu Kembali Gambar (*Image Retrieval*)

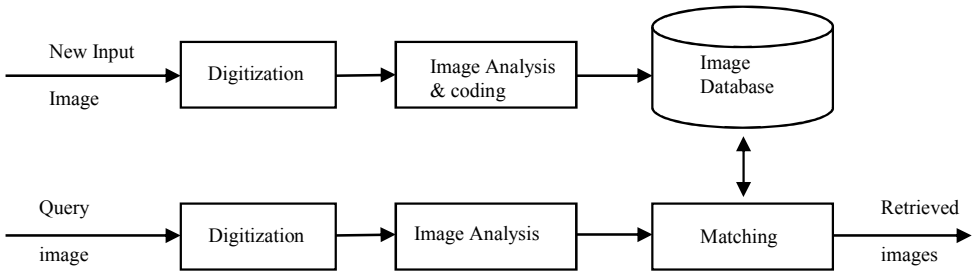
Image retrieval merupakan teknik yang digunakan untuk mencari gambar – gambar yang dinilai memiliki kemiripan dengan gambar acuan. Setiap gambar memiliki informasi fitur yang unik. Kriteria kemiripan antar gambar dapat didasarkan pada fitur – fitur seperti warna, tekstur, bentuk dan sebagainya. Terdapat dua teknik image retrieaval yang berkembang saat ini, yaitu tekstual dan berdasarkan isi.

1. Teknik tekstual yaitu berdasarkan pada deskripsi teks yang diberikan pada tiap gambar. Untuk pengambilan gambar digunakan *textual query*. Permasalahan pada teknik ini yaitu ketergantungan yang sangat tinggi pada manusia sebagai pendeskripsi gambar. Hal ini menyebabkan biaya tinggi. Disamping itu, jika data dalam ukuran besar (jumlah) dalam pendeskripsian data dapat terjadi pendeskripsian yang tidak konsisten, meluas dan sangat tergantung pada pemahaman dan penafsiran manusia terhadap gambar yang dilihat. Berikut skema system penyimpanan dan pemanggilan kembali gambar berdasarkan deskripsi teks [9].



Gambar 2.1 Skema Sistem Penyimpanan dan Pemanggilan Kembali Gambar Berdasarkan Deskripsi Teks

2. Teknik berdasarkan isi atau sering disebut content based image retrival (CBIR) adalah teknik pemanggilan gambar berdasarkan pada isi dari suatu gambar isi yang dimaksud dalam gambar merupakan warna, tekstur, bentuk relasi dan sebagainya. Permasalahan yang timbul pada teknik ini yaitu penafsiran gambar yang sesuai dan benar berdasarkan formulasi tertentu. Hal ini terjadi karena proses dilakukan oleh computer yang memiliki keterbatasan dalam menilai rasa, suasana dan keindahan. Berikut skema system penyimpanan dan pemanggilan kembali gambar berdasarkan konten gambar [9].



Gambar 2.2 Skema System Penyimpanan dan Pemanggilan Kembali Gambar Berdasarkan Konten Gambar [9]

2.4 Content Based Image Retrieval

Content Based Image Retrieval (CBIR) atau dikenal sebagai query dengan konten image dan pengambilan informasi visual berbasis konten (CBVIR) merupakan penerapan teknik computer vision untuk pengambilan gambar atau pencarian gambar digital dalam suatu database. Pencarian gambar berbasis konten (CBIR) berbeda dengan pendekatan berbasis konsep pada Concept Based Image Indexing. Ada beberapa teknik pada CBIR yang digunakan dalam pengambilan informasi yaitu query technique, semantic retrieval dan content comparison.

CBIR (Content Based Image Retrieval), atau disebut juga Query by Image Content (QBIC) merupakan suatu aplikasi dari Computer Vision untuk masalah image retrieval (pengambilan gambar), yaitu sebuah teknik yang menggunakan content visual dari gambar tetap dalam mencari kesamaan gambar dalam gambar database yang sangat besar menurut input atau sesuai dengan keinginan pengguna (user) [1].

2.5 Pengolahan Gambar (*Image processing*)

Pengolahan gambar adalah sebuah ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar baik itu meningkatkan kontras, transformasi warna, dan restorasi gambar, transformasi gambar seperti rotasi, translasi, skala, dan transformasi geometri, proses pemilihan sebuah gambar (feature images) yang optimal untuk tujuan analisis, melakukan penarikan informasi atau dekripsi objek atau pengenalan objek yang terkandung dalam gambar, melakukan kompresi atau reduksi sebuah data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data. Input dari pengolahan gambar adalah gambar, sedangkan output-nya adalah gambar hasul pengolahan.

2.5.1 Gambar Digital

Gambar digital adalah gambar yang dapat diolah oleh komputer. Gambar yang disimpan dalam memori komputer hanyalah sebuah angka-angka yang menunjukkan besar intensitas pada masing-masing *pixel* tersebut. Sebuah gambar dapat didefinisikan sebagai fungsi dua dimensi f(x,y), yang dimana x dan y adalah

korrdinat spasial, dan amplitude dari f pada sembarang pasangan koordinat (x,y) disebut *intensity* (intensitas) atau *gray level* (level keabuan) dari sebuah gambar pada titik tertentu. Ketika (x,y) dan nilai intensitas dari f adalah semua terbatas maka itu disebut digital image (gambar digital). Gambar digital terdiri dari sejumlah elemen tertentu, setiap elemen memiliki lokasi dan nilai tertentu. Elemen-elemen ini disebut *picture elements*, *image elements*, dan *pixels.* Pixel adalah istilah yang sering digunakan secara umum untuk menyatakan elemen pada gambar digital.

2.5.2 Grayscale

Grayscale adalah jenis gambar yang menangani gradiasi warna hitam dan putih, yang menghasilkan efek warna abu-abu. Pada jenis gambar ini, warna dinyatakan dengan intensitas. Dalam hal ini intensitas berkisar antara 0 sampai 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih. Konversi gambar normal (RGB) ke gambar grayscale dilakukan dengan menggunakan rumus (2.1) [10].

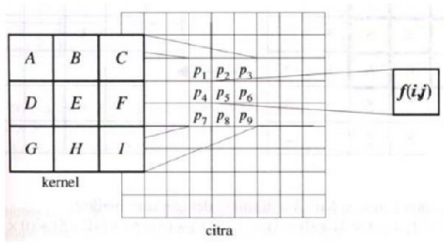
$$Grayscale = \frac{(R+G+B)}{3} \qquad (2.1)$$

2.5.3 Konvolusi Gambar

Konvolusi gambar adaolah teknik untuk menghaluskan suatu gambar atau memperjelas gambar dengan menggantikan nilai pixel dengan sejumlah nilai pixel yang sesuai atau berdekatan dengan pixel aslinya. Tetapi dengan adanya konvolusi, ukuran dari gambar tetap sama, tidak berubah. Untuk pengolahan gambar, operasi yang dilakukan adalah diskrit karena nilai koordinat pixel merupakan nilai yang diskrit. Selanjutnya *filter* atau *mask* yang digunakan pada pengolahan biasanya berukuran terbatas dalam artian bobot atau pengaruh dari titik-titik yang cukup jauh sudah tidak signifikan, sehingga dapat diabaikan (dianggap nol) [11]. Operasi konvolusi didefinisikan sebagai berikut.

$$h(x,y) = f(x,y) * g(x,y) = \sum_{a=\infty}^{\infty} \sum_{a=\infty}^{\infty} f(a,b).g(x-a,y-b) \quad (2.2)$$

Fungsi penapis g(x,y) disebut juga konvolusi *filter*, konvolusi *mask*, konvolusi *kernel* atau *template.* Dalam bentuk diskrit konvoulsi kernel dinyantakan dalam bentuk matriks (umumnya matrik 3x3). Ukuran matriks ini biasanya lebih kecil dari ukuran gambar. Setiap elemen matriks disebut koefisien konvolusi



Gambar 2.3 Ilustrasi Konvolusi

$$f(i,j) = Ap_1+Bp_2+Cp_3+Dp_4+Ep_5+Fp_6+Gp_7Hp_8+Ip_9$$

contoh

misalkan gambar *f(x,y)* yang berukuran 5x5 dan sebuah *kernel / filter* yang berukuran 3x3 masing – masing adalah sebagai berikut

$$f(x,y) = \begin{bmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 2 & 5 & 2 & 4 & 4 \end{bmatrix} \text{ dan } g(x,y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Operasi konvolusi antara *f(x,y)* dengan *g(x,y)* dapat diilustrasikan sebagai berikut.

1. Tempatkan *kernel* pada sudut kiri atas, kemudian hitung nilai pixel pada posisi(0,0) dari kernel.

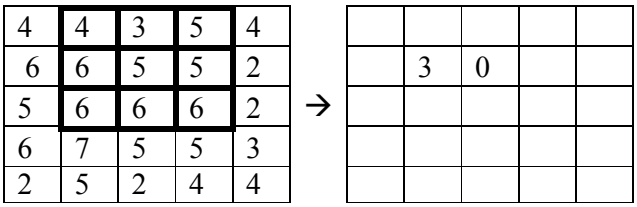
4	4	3	5	4
6	6	5	5	2
5	6	6	6	2
6	7	5	5	3
2	5	2	4	4

→

	3			

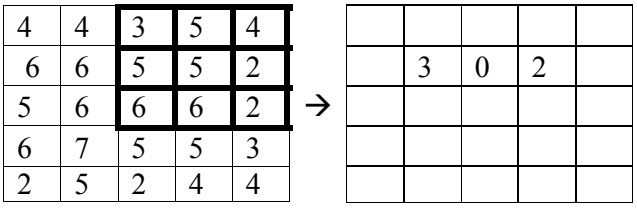
Hasil konvolusi = (0x4)+(-1x4)+(0x3)+(-1x6)+(4x6)+(-1x5)+(0x5)+(-1x6)+(0x6) = 3

2. Geser *kernel* satu pixel ke kanan, kemudian hitung pixel



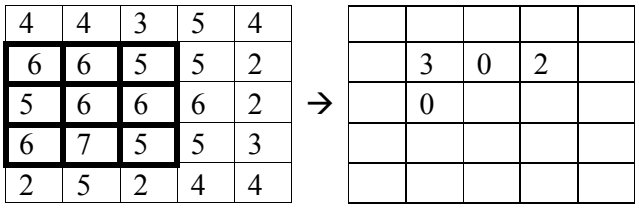
Hasil konvolusi = (0x4)+(-1x3)+(0x5)+(-1x6)+(4x5)+(-1x5)+(0x6)+(-1x6)+(0x6) = 0

3. Geser *kernel* satu pixel ke kanan, kemudian hitung pixel



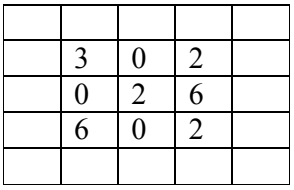
Hasil konvolusi = (0x3)+(-1x5)+(0x4)+(-1x5)+(4x5)+(-1x2)+(0x6)+(-1x6)+(0x2) = 2

4. Geser *kernel* satu pixel ke bawah, kemudian hitung pixel dari sis kiri



Hasil konvolusi = (0x6)+(-1x6)+(0x5)+(-1x5)+(4x6)+(-1x6)+(0x6)+(-1x7)+(0x5) = 0, kemudian lakukan pergeseran pixel seperti pada tahap 2 dan 3 setelah itu geser satu pixel ke bawah dan lakukan pergeseran pixel seperti pada tahap 2 dan 3.

5. Setelah semua tahap dilakukan maka konvolusi menghasilkan



Jika hasil konvolusi menghasilkan nilai *negative* maka nilai tersebut dijadikan 0. Sebaliknya jika hasil konvolusi menghasilkan nilai pixel lebih besar dari nilai keabuan maksimum (255), maka nilai tersebut dijadikan ke nilai keabuan maksimum. Adapaun masalah yang timbul, yaitu apabila pixel yang dikonvolusi adalah pixel pinggir, karena beberapa koefisien konvolusi tidak dapat diposisikan pada pixel – pixel gambar, maka solusinya pixel – pixel pinggir diabaikan, tidak dilakukan konvolusi.

2.5.4 Deteksi Tepi Canny

Deteksi tepi Canny merupakan algoritma yang digunakan untuk menganalisis suatu gambar tepi. Pengembang algoritma deteksi tepi canny adalah John F. Canny pada tahun 1986 dan menggunakan algoritma multi-tahap untuk mendeteksi berbagai tepi pada gambar.

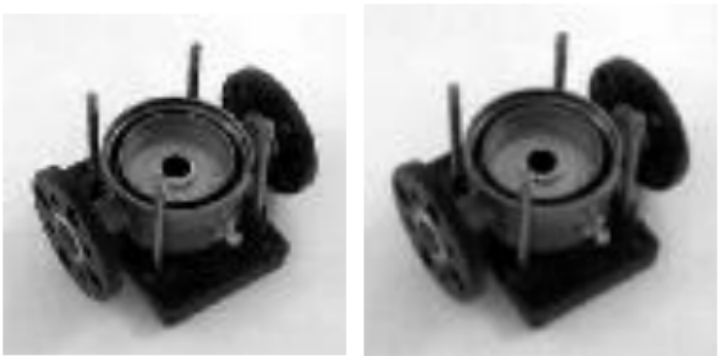
Algoritma Canny berjalan dalam 5 langkah yang terpisah yaitu :

a)	Penghalusan (*Smoothing*)	:	Menggabungkan gambar untuk menghilangkan noise.
b)	Mencari Gradien (Finding gradient)	:	Tepian harus ditandai pada gambar memiliki gradien yang besar.
c)	Non-maksimum-suppresion	:	Hanya maxima lokal yang harus ditandai Ende
d)	Double thresholding	:	Tepian yang berpotensi ditentukan oleh thresholding.
e)	Edge tracking by hysteresis	:	Tepian final ditentukan dengan menekan semua sisi yang tidak terhubung dengan tepian yang sangat kuat

Tidak dapat dipungkiri bahwa semua gambar yang diambil akan berisi sejumlah *noise*. Untuk mencegah *noise* salah dideteksi sebagai tepian, maka *noise* harus dikurangi. Oleh karena itu pada langkah pertama gambar harus diperhalus dengan menggunakan Gaussian filter. Inti dari *Gaussian filter* adalah standar deviasi dengan $\sigma = 1{,}4$ ditentukan pada persamaan (2.3) di bawah ini.

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (2.3)\ [10]$$

Pada persamaan diatas merupakan matriks *Gaussian filter* yang digunakan untuk menghilangkan untuk menghilangkan *noise* pada gambar.



(a) Original (b) Diperhalus

Gambar 2.4 *Grayscale* yang diperhalus

Algoritma Canny pada dasarnya menemukan titik tepi pada gambar grayscale dengan perubahan nilai intensitas yang paling besar, daerah ini ditemukan dengan menentukan gradien gambar. Gradien pada setiap piksel gambar yang telah diperhalus ditentukan dengan menerapkan operator Sobel. Langkah Kedua adalah memperkirakan gradien pada arah x dan y. Hal tersebut ditunjukkan dalam Persamaan (2.4).

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \; M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad (2.4)$$

Magnitudo gradien (juga dikenal sebagai kekuatan tepi) dapat ditentukan sebagai jarak Euclidean yang diukur mengukur dengan menerapkan hukum Pythagoras seperti yang ditunjukkan dalam Persamaan (2.5). Yang terkadang disederhanakan dengan menerapkan ukuran jarak Manhattan seperti yang ditunjukkan dalam Persamaan (2.6) untuk mengurangi kompleksitas komputasi.

$$|G| = \sqrt{G_x^2} + \sqrt{G_y^2} \qquad (2.5)$$

$$|G| = |G_x| + |G_y| \qquad (2.6)$$

dimana: Gx dan Gy adalah gradien pada masing-masing arah x dan y. Hal ini tampak jelas dari Gambar 3, bahwa gambar dengan gradien yang besar sering menunjukkan tepian yang cukup jelas. Namun, tepian biasanya luas dan dengan demikian tidak dapat menunjukkan persis di mana tepian yang sebenarnya. Untuk menentukan tepian yang sebenarnya ini, arah tepian harus ditentukan dan disimpan seperti ditunjukkan dalam Persamaan (2.7).

$$\theta = arctan\left(\frac{|G_x|}{|G_y|}\right) \qquad (2.7)$$

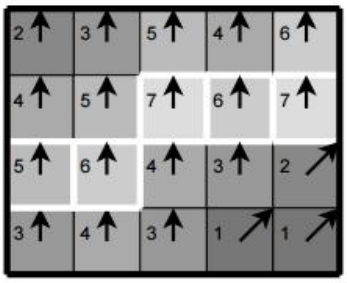


Gambar 2.5 Magnitudo gradien dari penerapan Operator Sobel terhadap gambar yang sudah dihaluskan

Pada langkah ketiga bertujuan untuk mengkorversikan tepian yang masih blurred pada gambar hasil magnitude gradien hingga menhasilkan tepian yang tajam. Pada dasarnya hal ini dilakukan dengan mempertahankan semua maxima lokal dalam gambar gradien dan menghapus segala sesuatu yang lain. Algoritma adalah untuk setiap piksel pada gambar gradien adalah sebagai berikut:

1. Putar arah gradien θ ke arah 45˚terdekat, kemudian hubungkan dengan 8 titik tetangga yang terhubung dengannya.
2. Bandingkan nilai piksel tepian saat ini dengan nilai piksel tepian dalam arah positif dan negatif gradien. Jika arah gradien adalah utara (θ =90 ◦), bandingkan dengan piksel ke utara dan selatan.
3. Jika nilai piksel tepian saat ini adalah yang terbesar, maka simpan nilai tepian tersebut, namun jika bukan, hapus nilai tersebut.

Sebuah contoh sederhana dari penghapusan non maksimum ditunjukkan pada Gambar 2.6. Hampir semua piksel yang memiliki arah gradien yang menunjukkan arah utara, oleh karena itu mereka dibandingkan dengan piksel atas dan bawah. Piksel yang berubah menjadi maksimal dalam perbandingan ini ditandai dengan warna putih pada perbatasan, sisanya dihapus. Gambar 5 menunjukkan efek pada gambar tes.



Gambar 2.6 Ilustrasi Penghapusan Non Maksimum



Gambar 2.7 Penghapusan Non Maksimum

Piksel tepian yang tersisa setelah dilakukan penghapusan non maksimum ditandai dengan nilai piksel per piksel yang kuat. Kebanyakan dari titik ini adalah tapian yang nyata pada gambar, akan tetapi beberapa kemungkinan disebabkan oleh noise atau variasi wana karena permukaan yang kasar. Cara paling sederhana untuk membedakannya adalaha menggunakan nilai threshold (ambang batas) sehingga hanya tepian dengan nilai yang kuar yang akan dipertahankan. Disini pada algoritma Canny menggunakan sistem *thresholding* ganda yang dimana *thresholding* tersebut akan dibandingkan dengan *thresolding* yang memiliki nilai yang lebih besar, apabila nilai lebih besar maka nilai *thresolding* tersebut akan

digunakan sebagai acuan menentukan tepian dan nilai *thresolding* yang kecil akan dihilangkan [11].



Gambar 2.8 Hasil akhir *Edge Detection Canny*

2.5.5 Ekstraksi Fitur

Ektraksi fitur atau featur extraction merupakan suatu pengambilan ciri / feature dari suatu bentuk yang nantinya nilai yang didapatkan akan dianalisis untuk proses selanjutnya. feature extraction dilakukan dengan cara menghitung jumlah titik atau pixel yang ditemukan dalam setiap pengecekan, dimana pengecekan dilakukan dalam berbagai arah tracing pengecekan pada kordinat kartesial dari gambar digital yang dianalisis.

Ektraksi fitur terbagi kedalam tiga macam, yaitu :

a. Ektraksi fitur bentuk

   Bentuk dari suatu objek adalah karakter konfigurasi permukaan yang diwakili oleh garis dan kontur. Fitur bentuk dikategorikan bergantung pada teknik yang digunakan. Kategori tersebut adalah berdasarkan batas (*boundary-based*) dan berdasarkan daerah (*region-based*). Teknik berdasarkan batas (*boundary-based*) menggambarkan bentuk daerah dengan menggunakan karakteristik eksternal, contohnya adalah piksel sepanjang batas objek. Sedangkan teknik berdasarkan daerah (*region-based*) menggambarkan bentuk wilayah dengan menggunakan karakteristik internal, contohnya adalah piksel yang berada dalam suatu wilayah

b. Ektraksi fitur tekstur

   Pada ekstraksi fitur ini, fitur pembeda adalah tekstur yang merupakan karakteristik penentu pada gambar. Teknik statistik yang

terkenal untuk ekstraksi fitur adalah matriks *gray level co-occurrence*. Teknik tersebut dilakukan dengan melakukan pemindaian untuk mencari jejak derajat keabuan setiap dua buah piksel yang dipisahkan dengan jarak d dan sudut θ yang tetap. Biasanya sudut yang digunakan adalah 0^{o}, 45^{o}, 90^{o}, dan 135^{o}.

c. Ektraksi fitur warna

Pada ekstraksi fitur warna, ciri pembeda adalah warna. Biasanya ekstraksi fitur ini digunakan pada gambar berwarna yang memiliki komposisi warna RGB (*red, green, blue*)

2.6 Singular Value Decomposition (SVD)

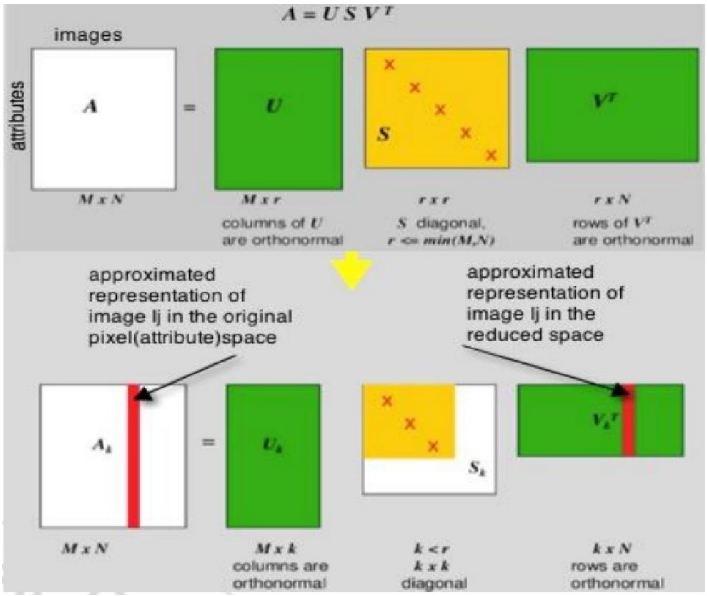
SVD adalah metode aljabar linier yang memecah matriks A berdimensi m x n menjadi tiga matriks USV. m merupakan nilai dari ekstraksi fitur dari masing - masing gambar dan n merupakan banyak gambar dalam database. Berikut persamaan yang dilakukan pada SVD ditunjukan pada persamaan (2.8):

$$\boldsymbol{A = USV^{T}} \qquad \textbf{(2.8)}$$

Keterangan :

A : Matrik ukuran m x n
U : Matrik ukuran m x r
S : Matrik ukuran r x r
V^{T} : Matrik ukuran r x n

Dimana S adalah matriks diagonal berisi nilai scalar (eigen values) berdimensi r x r dan r ditentukan sebelumnya,U dan V adalah matriks orthogonal yang berisi egen vektor. Berikut merupakan ilustrasi dari SVD [12].



Gambar 2.9 Proses LSI Pada Gambar

Untuk memudahkan pemahaman SVD, berikut adalah contoh matrik ordo kecil A yang akan digunakan untuk perhitungan.

Contoh

Diketahui :

Matrik A dengan ordo 2x3

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

Hitung SVD dari matrik A

Jawab

1. Untuk mendapatkan matrik U diawali dengan menghitung AA^T

$$AA^T = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

Setelah mendapat nilai AA^T dilanjutkan dengan menghitung nilai eigen dan eigen vektor dari AA^T. untuk mencari nilai eigen dan eigen vektor digunakan persamaan $\mathbf{det(AA^T - \lambda I) = 0}$ **(2.9)**

$$det\left(\begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

$$det\left(\begin{bmatrix}11-\lambda & 1\\ 1 & 11-\lambda\end{bmatrix}\right)=0$$

$$\left((11-\lambda)(11-\lambda)-1 \text{ x } 1\right)=0$$

$$((\lambda^2-22\lambda+121)-1=0$$

$$(\lambda^2-22\lambda+120)=0$$

$$(\lambda-10)(\lambda-12)=0$$

Sehingga diperoleh nilai $\lambda=10$ dan $\lambda=12$, dimana nilai λ merupakan nilai eigen. Untuk memperoleh vektor eigen niai λ disubstitusikan ke persamaan $(\mathbf{AA^T}-\boldsymbol{\lambda})\bar{\boldsymbol{x}}=\mathbf{0}$ **(2.10)**

Untuk nilai $\lambda=10$ diperoleh hasil sebagai berikut :

$$\left(\begin{bmatrix}11-10 & 1\\ 1 & 11-10\end{bmatrix}\right)\bar{x}=0$$

$$\left(\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}\right)\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix}$$

$x_1+x_2=0\,;\; x_1+x_2=0$

$x_1=-x_2$ sehingga diperoleh eigen vektor $\begin{bmatrix}1\\ -1\end{bmatrix}$

Untuk nilai $\lambda=12$ diperoleh hasil sebagai berikut :

$$\left(\begin{bmatrix}11-12 & 1\\ 1 & 11-12\end{bmatrix}\right)\bar{x}=0$$

$$\left(\begin{bmatrix}-1 & 1\\ 1 & -1\end{bmatrix}\right)\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix}$$

$-x_1+x_2=0\,;\; x_1-x_2=0$

$x_1=x_2$ sehingga diperoleh eigen vektor $\begin{bmatrix}1\\ 1\end{bmatrix}$

Vektor eigen tersebut menjadi kolom vektor pada matrik diurut berdasarkan nilai eigen. Vektor eigen dari nilai eigen terbesar menjadi kolom pertama sedangkan kolom berikutnya ditempati vektor eigen yang memiliki nilai eigen yang lebih kecil.

$$\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

Kemudian matrik tersebut dilakukan normalisasi terhadap kolom vektor dengan menggunakan *gramschmidt.* Perhitunganya adalah sebagai berikut :

$$\overline{u_1} = \frac{\overline{v_1}}{|\overline{v_1}|} \rightarrow \frac{[1,1]}{\sqrt{1^2+1^2}} = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$$

$$\overline{u_2} = \frac{\overline{v_1}}{|\overline{v_1}|} \rightarrow \frac{[1,-1]}{\sqrt{1^2+(-1)^2}} = \left[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right]$$

Sehingga matrik U yang diperoleh adalah

$$\mathrm{U} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

2. Untuk mendapatkan matrik V diawali dengan menghitung A^TA. Kemudian dicari nilai eigen dengan persamaan $\mathbf{det(A^TA - \lambda I) = 0}$ **(2.11)**

$$A^TA = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

$$\det\left(\begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} 10-\lambda & 0 & 2 \\ 0 & 10-\lambda & 4 \\ 2 & 4 & 2-\lambda \end{bmatrix}\right) = 0$$

$$(10-\lambda)[(10-\lambda)(2-\lambda)-16] + 2[0-(20-2\lambda)] = 0$$

$$\lambda(\lambda-10)(\lambda-12) = 0$$

Sehingga diperoleh nilai eigen yaitu $\lambda = 0,\ \lambda = 10,\ \lambda = 12$. Kemudian untuk memperoleh vektor eigen nilai λ disubstitusikan ke persamaan $\mathbf{(A^TA - \lambda)\bar{x} = 0}$ **(2.12)**

Untuk $\lambda = 12$

$$\begin{bmatrix} 10-\lambda & 0 & 2 \\ 0 & 10-\lambda & 4 \\ 2 & 4 & 2-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -2 & 0 & 2 \\ 0 & -2 & 4 \\ 2 & 4 & -10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$-2x_1 + 2x_3 = 0 \ ; \quad x_1 = 1, \qquad x_3 = 1$$

$-2x_2 + 4x_3 = 0\,;\quad x_2 = 2x_3 \rightarrow x_2 = 2$

Sehingga diperoleh eigen vektor $\begin{bmatrix}1\\2\\1\end{bmatrix}$

Untuk $\lambda = 10$

$$\begin{bmatrix}10-\lambda & 0 & 2\\ 0 & 10-\lambda & 4\\ 2 & 4 & 2-\lambda\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\begin{bmatrix}0 & 0 & 2\\ 0 & 0 & 4\\ 2 & 4 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$2x_3 = 0\,;\quad x_3 = 0$

$2x_1 + 4x_2 = 0$

$2x_1 = -4x_2\,;\ x_1 = 2,\quad x_2 = -1$

Sehingga diperoleh eigen vektor $\begin{bmatrix}2\\-1\\0\end{bmatrix}$

Untuk $\lambda = 0$

$$\begin{bmatrix}10-\lambda & 0 & 2\\ 0 & 10-\lambda & 4\\ 2 & 4 & 2-\lambda\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$\begin{bmatrix}10 & 0 & 2\\ 0 & 10 & 4\\ 2 & 4 & 2\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$10x_1 + 2x_3 = 0\,;$

$2x_3 = 10x_1\,;\ x_3 = -5$

$10x_2 - 20 = 0;\ x_2 = 2$

$2x_1 + 8 - 10 = 0;\ x_1 = 1$

Sehingga diperoleh eigen vektor $\begin{bmatrix}1\\2\\-5\end{bmatrix}$

Setelah diurutkan berdasarkan nilai eigen terbersar maka terbentuk matrik sebagai berikut :

$$\begin{bmatrix}1 & 2 & 1\\ 2 & -1 & 2\\ 1 & 0 & -5\end{bmatrix}$$

Kemudian matrik tersebut dilakukan normalisasi terhadap kolom vektor dengan menggunakan *gramschmidt.* Perhitunganya adalah sebagai berikut :

$$\overline{u_1} = \frac{\overline{v_1}}{|\overline{v_1}|} \rightarrow \frac{[1,2,1]}{\sqrt{1^2+2^2+1^2}} = \left[\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}},\right]$$

$$\overline{u_2} = \frac{\overline{v_2}}{|\overline{v_2}|} \rightarrow \frac{[2,-1,0]}{\sqrt{2^2+(-1)^2+0^2}} = \left[\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}}, 0\right]$$

$$\overline{u_3} = \frac{\overline{v_3}}{|\overline{v_3}|} \rightarrow \frac{[1,2,-5]}{\sqrt{1^2+2^2+(-5)^2}} = \left[\frac{1}{\sqrt{30}}, \frac{2}{\sqrt{30}}, \frac{-5}{\sqrt{30}},\right]$$

Sehingga matrik V yang diperoleh adalah :

$$\mathrm{V} = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{6}} & 0 & -\frac{5}{\sqrt{30}} \end{bmatrix}$$

Kemudian dilakukan transpose pada matriks V menjadi :

$$\mathrm{V^T} = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & -\frac{5}{\sqrt{30}} \end{bmatrix}$$

3. Matrik S merupakan diagonal dari akar nilai eigen positif, nilai eigen diurutkan berdasarkan nilai yang terbesar, nilai eigen yang terbesar akan ditempatkan pada baris pertama dan diikuti dengan akar nilai eigen terbesar berikutnya.

$$\mathrm{S} = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

Sehingga matrik SVD dari A yaitu :

$\mathrm{A} = \mathrm{U}, \mathrm{S}, \mathrm{V^T}$

$$\begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & -\frac{5}{\sqrt{30}} \end{bmatrix}$$

2.7 Essential Dimension of Latent Semantic Indexing (EDLSI)

EDLSI merupakan pengembangan dari LSI yang dilakukan oleh Konstathis. LSI mempunyai kekurangan yaitu pada komputasi waktu yang relatif lama. EDLSI signifikan dalam meningkatkan kinerja pengambilan data yang tidak didapatkan dari LSI, dan memberikan peningkatan *runtime* kinerja yang lebih tinggi. EDLSI memperoleh nilai document akhir dengan menghitung bobot LSI tradisional menggunaan nilai kecil untuk *k-rank* dan nilai bobot dari *vector space* [3]. Hasil vector perhitungan adalah :

$$\boldsymbol{w = (x)(q^T.A_k) + (1 - x)(q^T.A)} \qquad \boldsymbol{(2.13)}$$

Keterangan :

w : Bobot vektor
x : Bobot faktor dimana x yaitu ($0 \leq x \leq 1$)
q^T : Query
A_k : Matrik pembobotan LSI
A : Matrik

Adapun untuk tahapan untuk EDLSI itu sendiri antara lain :

1. Membuat matriks A
2. Melakukan proses SVD terhadap matriks A sehingga didapatkan matriks U,S,V^T
3. Melakukan reduksi baris dan kolom dengan menentukan nilai k-rank pada matriks SVD
4. Membuat matriks Ak yang merupakan hasil perkalian dari matriks SVD, matriks U, S, V
5. Melakukan perhitungan EDLSI dengan menggunakan persamaan 2.13

2.8 OOP (Object Oriented Programming)

Secara spesifik, pengertian berorientasi objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, polymorphism (polimorfisme) dan inheritance (pewarisan) [13].

2.8.1 Konsep Dasar OOP

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi atau sistem lainnya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespodensi dengan objek-objek dunia nyata. Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek

1. Kelas (Class)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau tercipta dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diwariskan ke kelas yang baru.

2. Objek (Object)

   Objek adalah abstraksi dan sesuatu yang mewakilkan dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status atau halhal lain yang bersifat abstrak. Objek merupakan entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakukan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yang diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (Method)

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

4. Atribut (Attribute)

   Atribut dari sebuah kelas adalah variable global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh

objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dan sebagainya.

5. Abstraksi (Absraction)

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

6. Enkapsulasi (Encapsulation)

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

7. Pewarisan (Inheritance)

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

8. Antarmuka (Interface)

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

9. Reusability

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

10. Generalisasi dan Spesialisasi

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

11. Komunikasi Antar Objek

    Komunikasi Antar Objek dilakukan lewat pesan (message) yang dikirim dari suatu objek ke objek yang lain.

12. Polimorfisme (Polymorphism)

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

13. Package

Package adalah sebuah container atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda [14].

2.8.2 Analisis Berorientasi Objek

Analisis berorientasi objek atau Object Oriented Analysis (OOA) adalah tahap untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah sistem berorientasi objek. OOA biasanya menggunakan kartu CRC (Component, Responsibility, Collaborator) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (Unifed Modeling Language) pada bagian diagram use case, diagram kelas, dan diagram objek [14].

2.8.3 Desain Berorientasi Objek

Desain berorientasi objek atau Object Oriented Design (OOD) adalah tahapan perantara untuk spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasikan objek. Pemodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat lunak dengan menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML (Unified Modeling Language). Kendala dan permasalahan pembangunan sistem berorientasi objek biasanya dapat dikenali dalam tahap ini [14].

2.8.4 Pemodelan

Pemodelan adalah gambaran dari realita yang sederhana dan dituangkan dalam bentuk pemetaan dengan aturan tertentu. Pemodelan dapat menggunakan bentuk yang sama dengan realitas. Pemodelan juga banyak digunakan untuk merencanakan suatu hal agar kegagalan dan resiko yang mungkin terjadi dapat

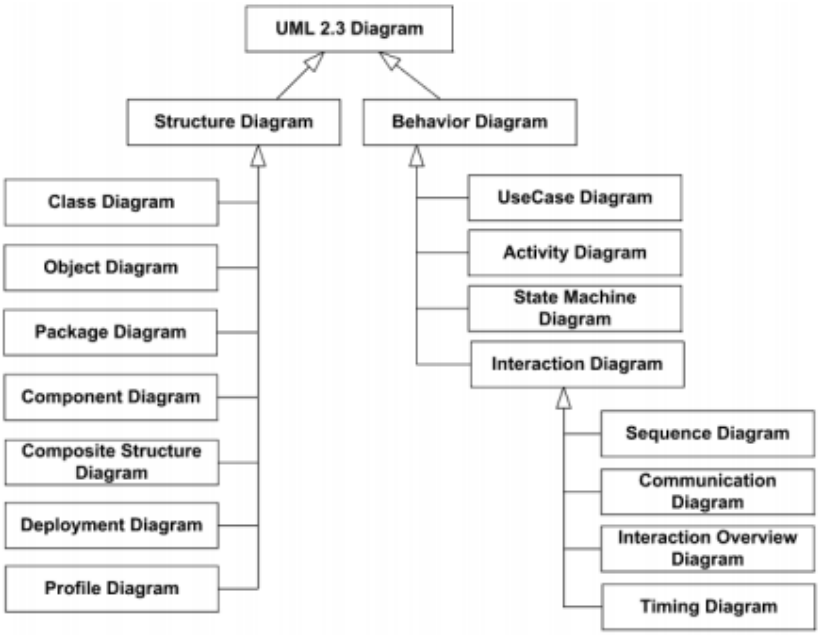
meminimalisir. Pada dunia pembangunan perangkat lunak juga diperlunakan pemodelan. Pemodelan perangkat lunak digunakan untuk mempermudah langkah berikutnya dari pengembangan sebuah sistem sehingga lebih terencana. Pemodelan pada pembangunan perangkat lunak untuk menvisualkan perangkat lunak yang akan dibuat [14].

2.9 UML (Unified Modeling Language)

UML *(Unified Modelling Language)* adalah sebuah "bahasa" yang telah menjadi standar dalam industri visualisasi, merancang, dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML dapat digunakan untuk membuat model untuk semua jenis perangkat lunak, dimana perangkat lunak tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentum, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu : Grady Booch OOD (Object Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique) dan Ivar Jacobson OOSE (Object Oriented Software Engineering) [14].

2.9.1 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada Gambar *2.10*



Gambar 2.10 Gambar *UML*

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

1. Structure diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. Behavior diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. Interaction diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem [14].

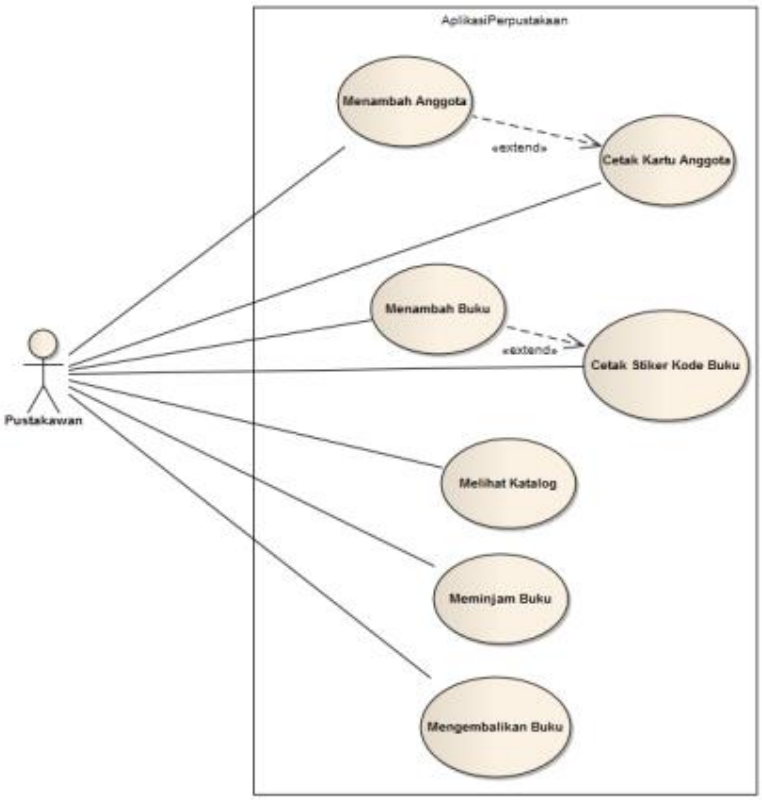
2.9.2 Diagram Use Case (*Use Case Diagram*)

Diagram use case merupakan pemodelan untuk kelakuan (behaviour) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada use case adalah nama didefinisikan sesimpel mungkin dan dapat

dipahami. Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case [14].

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. Use case merupakan fungsionalitas yang disediakan sistem sebagai unitunit yang saling bertukar pesan antar unit atau aktor

Pada Gambar *2.11* berikut ini merupakan contoh use case diagram dalam pembuatan sistem informasi perpustakaan.



Gambar 2.11 ***Use Case Diagram***

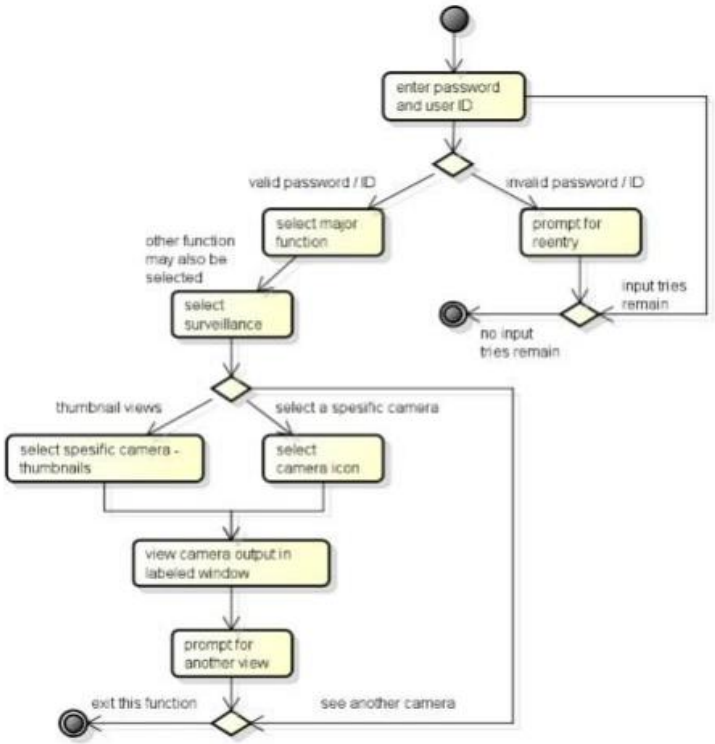
2.9.3 Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan

aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Rancangan menu yang ditampilkan pada perangkat lunak. Contoh Activity Diagram dapat dilihat pada Gambar *2.12*.

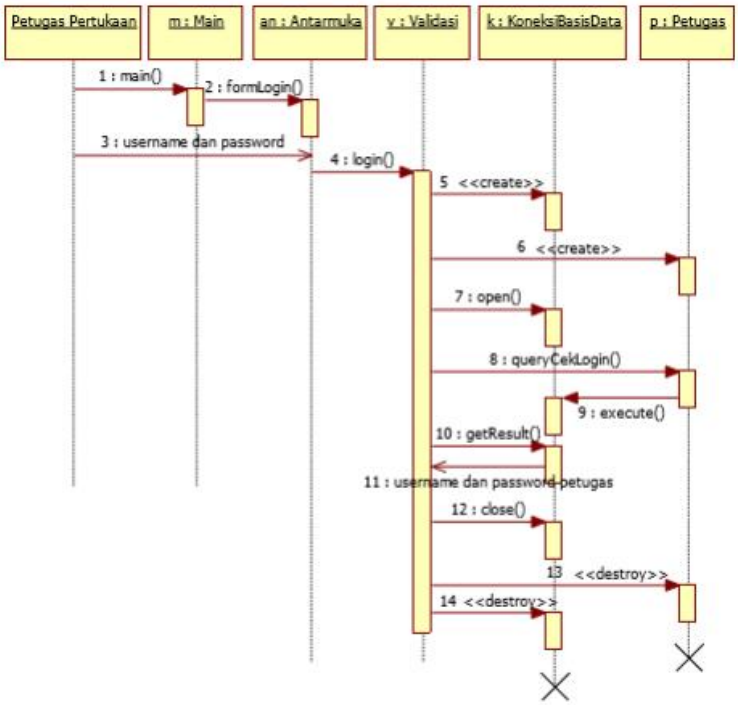


Gambar 2.12 Contoh *Activity Diagram*

2.9.4 Diagram Sekuen (*Sequence Diagram*)

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

antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada use case. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Contoh Sequence Diagram dapat dilihat pada Gambar *2.13*. [14]



Gambar 2.13 Contoh *Sequence Diagram*

2.9.5 Diagram Kelas (*Class Diagram*)

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas mimiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah

fungsifungsi yang dimiliki oleh suatu kelas. Contoh Class Diagram dapat dilihat pada Gambar *2.14*



Gambar 2.14 Contoh *Class Diagram*

Syarat penamaan pada use case adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.10 Bahasa Pemograman C#

Bahasa pemograman yang dipakai dalam membuat aplikasi ini adalah C#. C#(baca : See-Sharp) adalah bahasa pemograman baru yang diciptkan Microsoft yang digunakan oleh banyak developer .NET untuk mengembangkan aplikasi dengan platform .NET. C# ini merupakan bahasa pemograman yang sederhana dan simple. C# disebut sederhana dan simple, karna bahasa ini didasarkan kepada bahasa C dan C++. C# juga merupakan Object Oriented Language, dan bisa digunakan untuk membuat berbagai macam aplikasi, pengolahan kata, grafik, spreadsheets, atau bahkan membuat compiler untuk sebuah bahasa pemograman.

2.11 Confusion Matrix

Confusion matriks berisi informasi tentang klasifikasi aktual dan yang telah diprediksi yang dilakukan oleh sistem klasifikasi. Kinerja sistem tersebut umumnya dievaluasi dengan menggunakan data dalam matriks. Tabel berikut menunjukan confusion matriks untuk dua kelas.

Tabel 2-1 Confusion Matrix

Confusion Matrix		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positive	False Positive
	Negatif	False Negative	True Negative

True positive adalah jumlah record positif yang diklasifikasikan sebagai positif, false positive adalah jumlah record positif yang diklasifikasikan sebagai negatif, false negative adalah jumlah record negatif yang diklasifikasikan sebagai positif, true negative adalah jumlah record negatif yang diklasifikasikan sebagai negatif. Setiap kolom dari confusion matriks merupakan contoh kelas yang telah diprediksi, sedangkan setiap baris mewakili contoh di kelas yang sebenarnya. Setelah di dapat true positive, false positive, true negative dan false negatif, selanjutnya hitung nilai precision dan akurasinya. Precision dan recall adalah ukuran terhadap suatu kelas yang telah diprediksi. Berikut persamaan dari precision dan akurasi [15].

$$Akurasi = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.14)$$

$$Precision = \frac{TP}{TP + FP} \quad (2.15)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.16)$$

Keterangan :

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatif