BAB 2
LANDASAN TEORI

2.1 Pengertian Wajah

Wajah merupakan bagian dari tubuh manusia yang menjadi fokus di dalam interaksi sosial.Wajah juga merupakan salah satu bagian unik dari tubuh manusia yang mempunyai karakteristik yang berbeda.Wajah memiliki peranan vital dengan menunjukkan emosi/ekspresi.Wajah manusia menyediakan banyak informasi, banyak hal menarik yang bisa diperhatikan, dan dipelajari secara intensif.Beberapa penelitian yang menggali informasi dari wajah manusia adalah pengenalan wajah, pengenalan ekspresi wajah, dan pengenalan isyarat mimik wajah. Ketika manusia berinteraksi satu sama lain, mereka menggunakan berbagai macam isyarat dari wajah untuk menyampaikan informasi. Beberapa contohnya meliputi: mengangkat alis, arah tatapan mata, membuka mulut untuk berbicara, mengedip, dan ekspresi wajah lainnya.

Ekspresi wajah atau mimik wajah adalah salah satu bentuk komunikasi non-verbal yang merupakan hasil dari satu atau lebih gerakan atau posisi otot pada wajah serta dapat menyampaikan keadaan emosi dari seseorang kepada yang lainnya [5].Adapun gerakan atau posisi otot pada ekspresi wajah, bisa dilihat seperti pada tabel berikut [6]:

Tabel 2.1 Ekspresi yang tampak dari bagian-bagian wajah manusia

Emosi	Mata dan Alis	Dahi	Hidung	Mulut
Bahagia	Bagian bawah kelopak mata agak terangkat, terlihat ada kerutan, dan mata menyipit.	-	-	Bibir dan mulut melebar, kadang-kadang gigi terlihat.
Sedih	Ujung dalam alis terangkat	Berkerut	-	Sudut mulut tertarik kebawah dan bibir gemetar.

Terkejut	Seluruh alis terangkat dan mata membesar.	-	-	Rahang menurun dan mulut terbuka perlahan
Takut	Kelopak mata bagian atas terangkat, bagian putih mata terlihat jelas, kelopak mata bagian bawah menegang dan terangkat.	Berkerut	-	Bibir ditarik.
Marah	Alis ditarik ke dalam, mata menyipit.	-	Beberapa orang mengembangkan hidung	Bibir tertutup rapat.
Muak	Kelopak mata bagian bawah terangkat dan berkerut.	-	Berkerut	Merapat. Kedua bibir terangkat atau cemberut

2.2 Computer Vision

Computer vision adalah suatu ilmu di bidang komputer yang dapat membuat mesin atau robot untuk 'melihat'. Terdapat beberapa klasifikasi dari *vision* itu sendiri, yaitu *Low Level Vision*, *Medium Level Vision*, dan *High Level Vision*.

Low Level Vision meliputi *Sensing*, yaitu pengambilan input berupa gambar, dan *Preprocessing*, yaitu memperoleh suatu gambar sebelum diproses. *Medium Level Vision* meliputi proses *Segmentation, Description, Recognition*. *Segmentation* adalah proses pemisahan gambar *digital* kedalam beberapa *region*. *Description* merupakan proses mendeskripsikan suatu gambar, sedangkan *Recognition* merupakan pengenalan terhadap suatu gambar. Pada level yang lebih tinggi (*High Level Vision*) terdapat proses *Interpretation*, dimana *Interpretation*

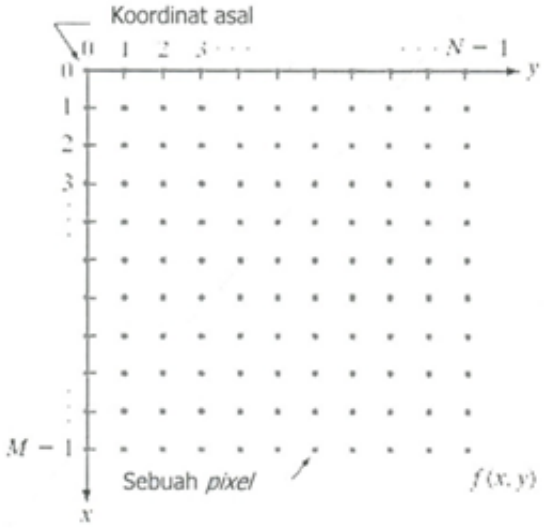
merupakan suatu kemampuan untuk memperkirakan bentuk asli dari gambar yang didapat, hal ini dapat dilakukan dengan cara mendapatkan berbagai informasi yang diperlukan pada gambar tersebut. Maka proses *Interpretation* memerlukan deteksi, indentifikasi, dan pengukuran dari fitur-fitur pada suatu gambar. Sistem *Stereo Eyes* (*Stereo Vision*) dalam skripsi ini hanya sampai dalam tahap Medium *Level Vision*, karena belum bisa melakukan interpretasi.

Sistem *Stereo Eyes* mempunyai konsep yang sama seperti 2 buah mata manusia yang terletak bersebelahan dengan jarak tertentu antara satu mata dengan yang lainnya. Ketika kedua mata tersebut melihat pada sebuah objek yang sama, masing-masing mata menangkap gambar dengan sudut pandang yang berbeda dan kemudian gambar tersebut dikirim ke otak untuk diproses. Ketika kedua gambar yang ditangkap oleh kedua mata tiba secara bersamaan di belakang otak, mereka bersatu menjadi satu gambar. Penglihatan tersebut dinamakan dengan penglihatan stereo (*stereo vision*). Pikiran manusialah yang menggabungkan kedua gambar dengan mencocokkan bagian yang sama dan menambahkan sedikit perbedaan, sehingga dapat memberikan informasi dari suatu gambar, salah satu informasi yang dapat diberikan adalah posisi objek dalam gambar tersebut. Salah satu metode yang digunakan untuk mendapatkan informasi posisi ini adalah perhitungan *disparity* (perbandingan 2 gambar) dari gambar stereo (*stereo image*) yang didapatkan dari penglihatan stereo (*stereo vision*). Meskipun begitu, metode perhitungan *disparity* hanya akan memberitahukan posisi objek tersebut terhadap objek lainnya tanpa informasi jarak. Informasi jarak merupakan perkembangan selanjutnya dari *computer vision* menggunakan *stereo vision*.

2.3 Pengolahan Citra Digital

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar dua dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (array) yang berisi nilai *real* maupun kompleks yang direpresentasikan dengan deretan bit tertentu.

Suatu citra dapat didefinisikan sebagai fungsi tertentu f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah kordinat spasial, dan amplituda f dititik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. [6] Gambar 2.1 menunjukan posisi koordinat citra digital.



Gambar 2.1. Koordinat Citra Digital

Citra digital dapat dituliskan dalam bentuk matrik sebagai berikut:

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

Gambar 2. 2 Matriks Citra Digital

2.3.1 Jenis Citra

Nilai suatu pixel memiliki dalam suatu rentang tertentu, dari nilai minimum sampai nilai maksimum. Jangkauan yang digunakan berbeda-beda tergantung dari jenis warnanya. Namun secara umum jangkauanya adalah 0-255. Citra dengan

penggambaran seperti ini digolongkan ke dalam citra integer. Berikut adalah jenis-jenis citra berdasarkan nilai pixelnya. [6]

a. Citra Biner

Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai pixel yaitu hitam dan putih. Citra biner juga sering disebut dengan citra black and white atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap pixel dari citra biner.

b. Citra Grayscale

Citra grayscale merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian RED = GREEN = BLUE. Nilai tersebut digunakan untuk menunjukan tingkat intensitas. Warna yang dimiliki adlah warna dari hitam, keabuan, dan putih. Tingkat keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. Citra grayscale memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan).

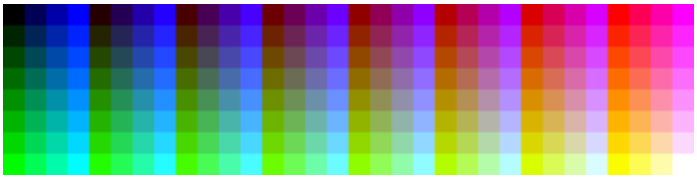
c. Citra Warna (8 Bit)

Setiap pixel dari citra wana (8 bit) hanya diwakili oleh 8 bit dengan jumlah maksimum yang dapat digunakan adalah 256 warna. Ada dua jenis citra warna 8 bit. Pertama, citra warna 8 bit dengan menggunakan palet warna 256 dengan setiap paletnya memiliki pemetaan nilai (colormap) RGB tertentu. Model ini lebih sering digunakan. Kedua, setiap pixel memiliki format 8 bit sebagai berikut Tabel 2.2 citra warna 8 bit truecolor:

Tabel 2.2 Citra Warna 8 bit Truecolor

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

Bentuk kedua ini dinamakan 8 bit truecolor. Berikut ini adalah warna-warnanya pada gambar 2.3



Gambar 2.3 (delapan) bit true color

d. Citra Warna (16 Bit)

Citra warna 16 bit (biasanya disebut citra highcolor) dengan setiap pixelnya diwakili dengan 2 byte memory (16 bit).

Warna 166 bit meiliki 65.536 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 ekstra . pemilihan komponen hijau degnan deret bit dikarenakan penglihatan manusia lebih sensitif terhadap warna hijau.

Tabel 2.3 Citra warna 16 bit

Bit															
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

e. Citra Warna (24 Bit)

Setiap pixel dari citra warna 24 bit diwakili degnan 24 bit 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 pixel dari citra warna 24 bit diwakili degnan 24 bit 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 Secara umum. Pengolahan citra digital menunjukan pada pemrosesan gambar dua dimensi menggunakan *Computer*. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan

sebuah larik (*array*) yang berisi nilai-nilai *real* maupun kompleks yang direpresentasikan dengan deretan bit tertentu.

Citra secara harafiah, adalah gambar pada bidang dua dimensi (dwimatra). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sedangkan citra digital adalah citra yang dapat diolah oleh *Computer*.

Suatu citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan *amplitude* f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x,y, dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatan bahwa citra tersebut citra digital. Sebuah citra dapat dibentuk dalam bentuk persamaan berikut.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,0) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.4 Matriks Pembentuk Citra

Nilai pada suatu irisan antara baris dan kolom (pada posisi x,y) disebut dengan *picture elements*, *Image elements, pels, atau pixel*. Istilah terakhir (*pixel*) paling sering digunakan pada citra digital.

2.3.2 Citra Digital

Citra digital dapat didefinisikan sebagai fungsi dua variabel, f(x,y), dimana x dan y adalah koordinat spasial dan nilai f(x,y), yang merupakan intensitas citra pada koordinat tersebut. Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (*Red, Green, Blue – RGB*).[6]

2.3.2.1 Karakteristik Citra Digital

Setiap citra digital memiliki beberapa karakteristik, antara lain yaitu :

1. Ukuran citra digital

Ukuran citra (*Image size*) menyatakan ukuran banyaknya *pixel* penyusun citra raster yang dinyatakan dalam matrik 2 dimensi, yaitu (X × Y) *Pixel*, dimana X menyatakan ukuran banyaknya *pixel* perbaris pada arah horizontal sedangkan Y menyatakan ukuran banyaknya *pixel* perkolom pada arah *vertikal*. Sebagai contoh, Citra digital berukuran 800 × 600 *pixel*, terdiri dari 800 × 600 *pixel* = 480.000 *pixel*, dengan susunan 800 *pixel* setiap baris pada arah *horizontal* dan 600 *pixel* setiap kolom pada arah *vertikal*.

2. Resolusi

Atribut citra digital yang tidak kalah pentingnya adalah resolusi (*resolution*), yang didefinisikan sebagai banyaknya *pixel* dalam setiap satuan panjang. Umumnya, resolusi dinyatakan dalam satuan dpi (dot per inch). Sebagai contoh, citra digital yang memiliki resolusi 75 dpi, berarti terdiri dari 75 dot (titik) pada setiap inchi. Semakin tinggi resolusi suatu citra digital, maka kualitasnya akan semakin baik.

2.3.2.2 Tipe Citra Digital

Berdasarkan format penyimpanan nilai warnanya citra digital terbagi dalamtiga tipe yaitu :

1. Citra *Biner*

   Citra yang hanya memiliki dua nilai yaitu 1 dan 0. Dinyatakan dalam rumus 2.1:

$$f_{(x,y)} = \Sigma \{0,1\} \qquad (2.1)$$

2. Citra Grey-scale

Citra yang terdiri dari satu layer warna dengan derajat keabuan tertentu. Untuk kebanyakan citra digital 8 bit, maka sistem Grey-scale diukur berdasarkan skala intensitas kecerahan, yang bernilai 0 – 255, dimana yang hitam pekat adalah 0 dan yang terputih adalah 255. Dinyatakan dalam rumus2.2 :

$f_{(x,y)} = \Sigma\ [0...255]$ (2.2)

3. Citra Berwarna

Citra standar (Bitmap) dengan ukuran 512 × 512 dan mempunyai kedalaman warna 24 bit atau dikenal dengan istilah true color, dalam hal ini kedalaman warna yang dimaksud yaitu jumlah bit yang dipakai untuk merepresentasikan tiap titik pada gambar tersebut dinyatakan dalam bit per *pixel* . komputer mempunyai tiga komponen warna yaitu, merah, hijau, dan biru. Ketiga komponen dasar ini dikenal dengan istilah warna RGB (*RedGreenBlue*). Untuk citra digital dengan 8 bit per *pixel* mempunyai 256 warna sedangkan pada citradengan 24 bit per *pixel* mempunyai 16 jutawarna, tiap *pixel* dinyatakan dengan:

Bit ke-0 sampai ke-7 untuk warna merah

Bit ke-8 sampai ke-15 untuk warna hijau

Bit ke-16 sampai ke-23 untuk warna biru

Sehingga kombinasi warnanya adalah $256^3 + 256^2 + 256^1 = 16.843.008$. Warna-warna ini merupakan warna dasar penyusun setiap *pixel* pada citra digital. Representasi dalam citra digital dinyatakan dalam persamaan 2.3, 2.4,dan 2.5:

$fR(x,y)\ \Sigma\ [0...255]$ (2.3)

$fG(x,y)\ \Sigma\ [0...255]$ (2.4)

$fB(x,y)\ \Sigma\ [0...255]$ (2.5)

1.2.3 Format File Citra Digital

Beberapa contoh format umum, pada citra digital yaitu Bitmap (BMP), *Joint Photographic Group Experts* (JPEG), *Graphics Interchange Format* (GIF), dan *Purchase orderrtable Network Graphics* (PNG).

1. *Bitmap* (BMP)

Bitmap merupakan format baku citra pada sistem operasi *windows* dan IBM OS/2. Citra berformat BMP merupakan citra yang tidak terkompresi, sehingga pada umumnya citra berformat BMP mempunyai ukuran yang relatif lebih besar dibandingkan dengan format citra lainnya. Intensitas *pixel* dari citra berformat

BMP dipetakan ke sejumlah bit tertentu. Panjang setiap *pixel* pada bitmap yaitu 4 bit, 8 bit, sampai 24 bit yang merepresentasikan nilai intensitas *pixel*. Dengan demikian ada sebanyak 28 = 256 derajat keabuan, mulai dari 0 sampai 255.

2. Joint Photographic Experts Group (JPEG)

Joint Photographic Group Experts (JPEG) merupakan standar kompresi file yang dikembangkan oleh Group Joint Photographic Experts menggunakan kombinasi DCT dan pengkodean Huffman untuk mengkompresikan citra. Citra JPEG merupakan citra terkompresi yang bersifat lossy, artinya citra tidak bisa dikembalikan ke bentuk aslinya.Citra ini memiliki ukuran yang relatif lebih kecil dibandingkan dengan citraberformat BMP karena telah terkompresi.

3. Graphics Interchange Format (GIF)

Graphics Interchange Format (GIF) merupakan format citra terkompresi. Ukuran dari citra berformat GIF merupakan yang terkecil dari semua format citra digital. Kombinasi warna yg tersedia sebanyak 256 warna. Citra jenis ini banyak dugunakan untuk keperluan website, untuk membuat sebuah icon, gambar dan keperluan lainnya yang mengharuskan menggunakan citra dengan ukuran yang kecil. Citra jenis ini juga dapat dibuat animasi.

4. Purchase orderrtable Network Graphics (PNG)

Purchase orderrtable Network Graphics (PNG) adalah salah satu format penyimpanan citra yang menggunakan metode kompresi yang tidak menghilangkan bagian dari citra tersebut (*lossless compression*). Citra berformat PNG merupakan salah satu format yang baik untuk digunakan pengolahan citra, karena format ini selain tidak menghilangkan bagian dari citra yang sedang diolah.

2.3.3 Konvolusi

Konvolusi adalah salah satu proses filtering image yang sering dilakukan pada proses pengolahan citra. Proses konvolusi dilakukan dengan menggunakan matriks yang berjalan sepanjang proses dan digunakan untuk menghitung nilai representasi lokal dari beberapa piksel pada citra. [7][8][9]

2.3.4 Citra Grayscale

Grayscale adalah citra yang terdiri dari warna hitam, abu-abu, dan putih. Grayscale menunjukkan jumlah warna yang ada dalam satu citra. Makin besar angka grayscale, maka citra yang terbentuk makin mendekati kenyataan, biasanya grayscale disebut sebagai gambar hitam putih, dan memiliki 8 bit warna. Untuk menghasilkan warna *grayscale* digunakan rumus *Luminosity* pada persamaan 2.6 [9].

$$Grayscale = 0.2989 \times R + 0.5870 \times G + 0.1141 \times B \quad (2.6)$$

dimana:

R adalah nilai matriks berwarna merah,

G adalah nilai matriks berwarna hijau,

B adalah nilai matriks berwarna biru.

2.3.5 Deteksi Tepi Canny (*Edge Detection*)

Edge detection canny merupakan algoritma yang digunakan untuk menganalisis suatu citra tepi. Pengembang algoritma *Edge detection canny* adalah John F. Canny pada tahun 1986 dan menggunakan algoritma multi-tahap untuk mendeteksi berbagai tepi pada gambar.[8][9]

Algoritma *Canny* berjalan dalam 5 langkah yang terpisah yaitu:

Tabel 2.4 *Step Canny*

No.	Nama	Keterangan
1	*Smoothing*	Menggabungkan gambar untuk menghilangkan noise.
2	*Finding gradient*	Tepian harus ditandai pada gambar memiliki gradien yang besar.
3	*Non-maksimum-suppresion*	Hanya maxima lokal yang harus ditandai egde
4	*Double thresholding*	Tepian yang berpotensi ditentukan oleh thresholding.

5	*Edge tracking by hysteresis*	Tepian final ditentukan dengan menekan semua sisis 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. Adapun persamaan dan tahapan perhitungan *Gaussian Filter* adalah sebagai berikut:

Persamaan *Gaussian filter* : $G(x,y)/c = e^{-\frac{(x^2+y^2)}{2\sigma^2}}$ (2.7) [8][9]

dimana:

G=nilai perhitungan menggunakan rumus kernel gaussian.

c =nilai setiap mask kernel (caranya dengan membagi setiap nilai (x,y) yang didapat dari hasil perhitungan $e^{-\frac{(x^2+y^2)}{2\sigma^2}}$ dengan nilai terkecil yang ada pada matriks)

e = 2.71828182846

x = koordinat pixel horizontal

y = koordinat pixel vertikal

σ= 1.4

Tabel 2.5 Nilai x dan y pada matriks *mask Kernel*

(x,y)	**-2**	**-1**	**0**	**1**	**2**
-2	(-2,-2)	(-2,-1)	(-2,0)	(-2,1)	(-2,2)
-1	(-1,-2)	(-1,-1)	(-1,0)	(-1,1)	(-1,2)
0	(0,-2)	(0,-1)	(0,0)	(0,1)	(0,2)
1	(1,-2)	(1,1)	(1,0)	(1,-1)	(1,2)
2	(2,-2)	(2,-1)	(2,0)	(2,1)	(2,2)

Dengan menghitung persamaan 2.7 menggunakan σ= 1,4 dan menempatkan pada matriks, maka didapatkan nilai pada tabel 2.6, yaitu:

Tabel 2.6 Hasil perhitungan *mask kernel* pada persamaan *Gaussian filter*

(x,y)	**-2**	**-1**	**0**	**1**	**2**
-2	0.1299	0.2793	0.3605	0.2793	0.1299
-1	0.2793	0.6004	0.7745	0.6004	0.2793
0	0.3605	0.7745	1	0.7745	0.3605
1	0.2793	0.6004	0.7745	0.6004	0.2793
2	0.1299	0.2793	0.3605	0.2793	0.1299

Untuk menghitung nilai c pada tiap koordinat, ditentukan nilai terkecil dari hasil perhitungan *mask kernel*, kemudian setiap nilai pada koordinat dibagi dengan jumlah nilai keseluruhan matriks.

Cara menghitungnya:

1. Jumlah keseluruhan matriks = 0.1299 + 0.2793 + 0.3605 + 0.2793 + 0.1299 + 0.2793 + 0.6004 + 0.7745 + 0.6004 + 0.2793 + 0.3605 + 0.7745 + 1 + 0.7745 + 0.3605 + 0.2793 + 0.6004 + 0.7745 + 0.6004 + 0.2793 + 0.1299 + 0.2793 + 0.3605 + 0.2793 + 0.1299 = 10.6956.
2. Nilai c pada koordinat (0,0) = 1/10.6956 = 0.0935
3. Nilai c pada koordinat (-1,0) = (0,-1) = (0,1) = (1,0) = 0.7745/10.6956= 0.0721
4. Nilai c pada koordinat (-1,-1) = (-1,1) = (1,-1) = (1,1) = 0.6004/10.6956= 0.0561
5. Nilai c pada koordinat (-1,-2) = (-2,-1) = (-2,1) = (-1,2) = (1,-2) = (2,-1) = (1,2) = (2,2) = 0.2793 /10.6956= 0.0261
6. Nilai c pada koordinat (0,-2) = (-2,0) = (-2,1) = (0,2) = (2,0) = 0.3605/10.6945 = 0.0337
7. Nilai c pada koordinat (-2,-2) = (-2,2) = (2,-2) = (2,2) = 0.1299/10.6945= 0.0121

Sehingga diperoleh nilai matriks mask kernel pada tabel 2.6:

Tabel 2.7 Matriks *gaussian filter*

(x,y)	-2	-1	0	1	2
-2	0.0121	0.0261	0.0337	0.0261	0.0121
-1	0.0261	0.0561	0.0721	0.0561	0.0261
0	0.0337	0.0721	0.0935	0.0721	0.0337
1	0.0261	0.0561	0.0721	0.0561	0.0261
2	0.0121	0.0261	0.0337	0.02611	0.0121

Berikut merupakan persamaan matriks *gaussian filter* yang digunakan untuk menghilangkan *noise*.

$$\begin{bmatrix} 0.0121 & 0.0261 & 0.0337 & 0.0261 & 0.0121 \\ 0.0261 & 0.0561 & 0.0721 & 0.0561 & 0.0261 \\ 0.0337 & 0.0721 & 0.0935 & 0.0721 & 0.0337 \\ 0.0261 & 0.0561 & 0.0721 & 0.0561 & 0.0261 \\ 0.0121 & 0.0261 & 0.0337 & 0.0261 & 0.0121 \end{bmatrix} .\textit{Grayscale} \quad (2.8)$$

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 (2.9) dan persamaan (2.10).

Operator Sobel $Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Operator Sobel $Gy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

$G_x =$ OperatorSobel Gx . Hasil $Gaussian$ (2.9)[8][10]

$G_y =$ Operator Sobel Gy . Hasil $Gaussian$ (2.10)[8][10]

Magnitudo gradien (juga dikenal sebagai kekuatan tepi) dapat ditentukan sebagai jarak Euclidean yang diukur mengukur dengan menerapkan hukum Pythagoras seperti yang ditunjukkan dalamPersamaan (2.11). Yang terkadang disederhanakan dengan menerapkan ukuran jarak Manhattan seperti yang ditunjukkan dalam Persamaan (2.12) untuk mengurangi kompleksitas komputasi.

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{2.11}$$

$$|G| = |G_x| + |G_y| \tag{2.12}$$

dimana: Gx dan Gy adalah gradien pada masing-masing arah x dan y. Hal ini tampak jelas 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 (2.13).

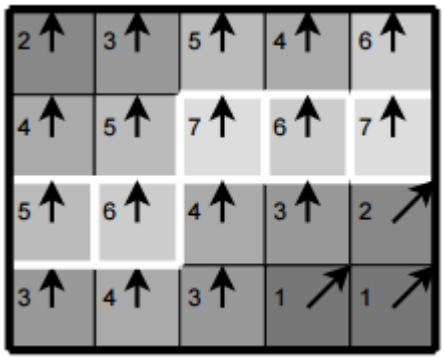
$$\theta = arctan\left(\frac{|G_x|}{|G_y|}\right) \tag{2.13}$$

Pada langkah ketiga bertujuan untuk mengkorversikan tepian yang masih blurred pada gambar hasil magnitude gradien hingga menghasilkan 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 2.5. 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 2.5 Ilustrasi Penghapusan Non Maksimum

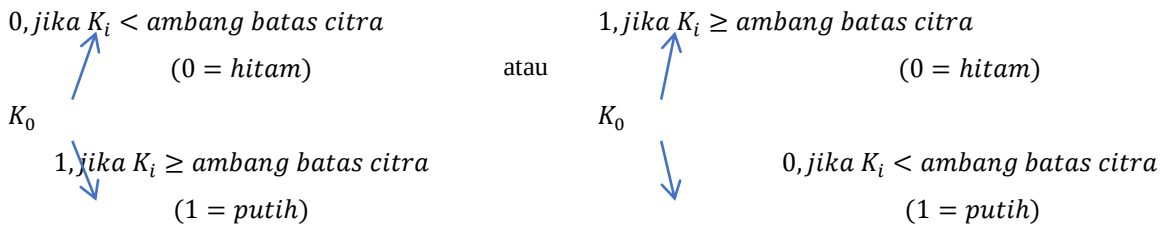
Piksel tepian yang tersisa setelah dilakukan penghapusan non maksimum ditandai dengan nilai piksel per piksel yang kuat. Kebanyakan dari titik ini adalah tapian yang nyata pada gambar, akan tetapi beberapa kemungkinan disebabkan oleh noise atau variasi wana karena permukaan yang kasar. Cara paling sederhana untuk membedakannya adalah 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.

2.3.6 Normalisasi

Normalisasi merupakan proses merubah dimensi region tiap karakter. Pada proses ini menggunakan proses otsu *Thresholding*, merupakan salah satu operasi yang termasuk ke dalam operasi titik dalam pengolahan citra digital. Operasi ini digunakan untuk mengubah citra dengan format skala keabuan yang dimiliki kemungkinan nilai lebih dari 2 ke citra biner yang memiliki 2 buah nilai (0 dan 1). Thresholding terdiri dari dua jenis, yaitu:

a. *Thresholding* Tunggal

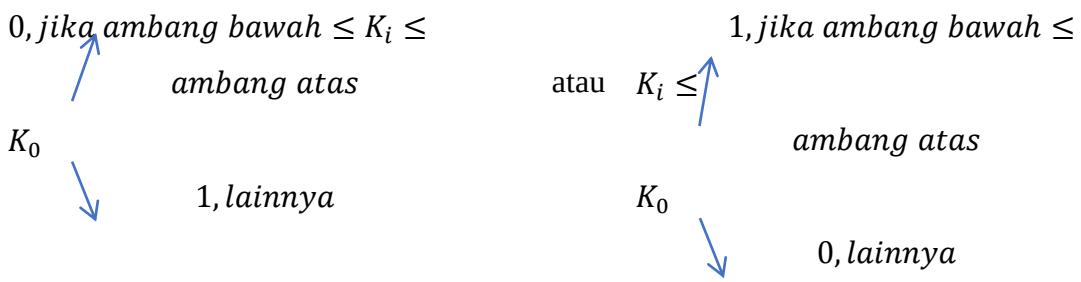
Thresholding Tunggal merupakan proses *thresholding* yang hanya memiliki sebuah nilai ambang batas atau batasan suatu citra.



Gambar 2.6 Fungsi Thresholding Tunggal

b. *Thresholding* Ganda

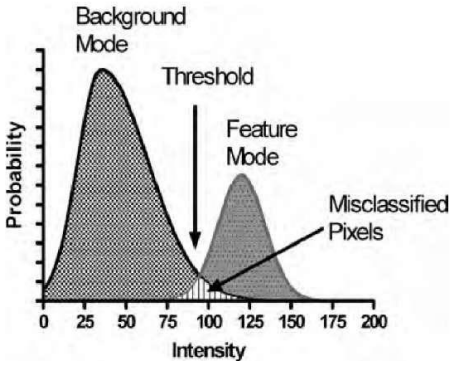
Memiliki ambang batas atas dan batas bawah.Dilakukan untuk menampilkan titik-titik yang memiliki rentang nilai skala keabuan tertentu.



Gambar 2.7 Fungsi Thresholding Ganda

2.3.6.1 *Otsu Threshold*

Metode *Otsu* dipublikasikan oleh *NobuyukiOtsu* pada tahun 1979. Metode ini menentukan nilai ambang dengan cara membedakan dua kelompok, yaitu objek dan latar belakang, yang memiliki bagian saling bertumpukan, berdasarkan histogram (lihat gambar 2.8).



Gambar 2.8 Penilaian nilai ambang untuk memperoleh hasil optimal

Prinsip metode *Otsu* dijelaskan sebagai berikut ini. Pertama-tama, probabilitas nilai intensitas *i* dalam histogram dihitung melalui persamaan 2.14 berikut:

$$\mathrm{p(i)} = \frac{\mathrm{n_i}}{\mathrm{N}}, \mathrm{p(i)} \geq 0\,, \sum_{\mathrm{i=1}}^{\mathrm{t}} \mathrm{p(i)} = 1 \qquad (2.14)$$

Dengan n_i menyatakan jumlah piksel berintensitas i dan N menyatakan, jumlah semua piksel dalam citra. Jika histogram dibagi menjadi 2 kelas (objek dan latar belakang), pembobotan pada kedua kelas dinyatakan pada masing-masing persamaan 2.15 dan 2.16 berikut:

$$\mathrm{w_1(t)} = \sum_{\mathrm{i=1}}^{\mathrm{t}} \mathrm{p(i)} \qquad (2.15)$$

$$\mathrm{w_2(t)} = \sum_{\mathrm{i=t+1}}^{\mathrm{L}} \mathrm{p(i)} = 1 - \mathrm{w_1(t)} \qquad (2.16)$$

Dalam hal ini, L menyatakan jumlah aras keabuan. Rerata kedua kelas dihitung melalui persamaan 2.17 dan persamaan 2.18 berikut:

$$m_1(t) = \sum_{i=1}^{t} i.\, p(i)/w_1(t) \quad (2.17)$$

$$m_2(t) = \sum_{i=1}^{t} i.\, p(i)/w_2(t) \quad (2.18)$$

Varian kedua kelas dinyatakan dengan persamaan 2.19 dan persamaan 2.20 berikut:

$$\sigma_1{}^2(t) = \sum_{i=1}^{t} (1 - m_1)^2 . \frac{p(i)}{w_1(t)} \quad (2.19)$$

$$\sigma_2{}^2(t) = \sum_{i=t+1}^{L} (1 - m_2)^2 . \frac{p(i)}{w_2(t)} \quad (2.20)$$

Varians total dapat dinyatakan dengan persamaan 2.21 berikut:

$$\sigma^2(t) = \sigma_W{}^2(t) + \sigma_B{}^2(t) \quad (2.21)$$

Dalam hal ini, $\sigma_w{}^2$ dinamakan sebagai *Within-Class Variance* (WCV) dan $\sigma_B{}^2$ disebut *Between-Class Variance* (BCV). WCV dapat dinyatakan dengan persamaan 2.22 berikut:

$$\sigma_W{}^2(t) = W_1(t) . \sigma_1(t)^2 + W_2(t) . \sigma_2(t)^2 \quad (2.22)$$

Rumus diatas menunjukkan bahwa wcv adalah jumlah varians kelas secara individual yang telah diboboti dengan probabilitas kelas masing-masing. Adapun BCV dinyatakan dengan persamaan 2.23 berikut:

$$\sigma_B{}^2(t) = W_1 . [m_1(t) - m_T]^2 + W_2 . [m_2(t) - m_T]^2 \quad (2.23)$$

Dalam hal ini, m_T adalah rerata total ($m_T = \sum_{i=1}^{N} i.P(i)$).

Nilai ambang optimum dapat diperoleh dengan 2 cara. Cara pertama dilakukan dengan meminimumkan WCV.Cara kedua dilakukan dengan memaksimumkan BCV. Namun, berdasarkan kedua cara tersebut, cara yang kedua lebih menghemat komputasi. [9]

2.4 Ekstraksi Fitur

Ekstraksi fitur merupakan nilai fitur yang terkandung di dalam sebuah citra yang mewakili ciri khusus dari sebuah citra. Nilai yang didapatkan dari hasil pengekstraksian fitur dari sebuah citra kemudian akan di proses untuk tujuan keperluan selanjutnya.

2.4.1 *Discrete Cosine Transform* (DCT)

Discrete Cosine Transform (DCT) biasa digunakan untuk mengubah sebuah sinyal menjadi komponen frekuensi dasarnya [14]. DCT adalah sebuah transformasi yag mengubah sebuah kawasan spasial menjadi kawasan fekuensi dan sebaliknya kawasan frekuensi dapat dikembalikan ke kawasan spasial dengan menggunakan invers DCT. DCT pertama kali diperkenalkan oleh Ahmed, Natarajan dan Rao pada tahun 1974 dalam makalahnya yang berjudul "On image processing and a discrete cosine transform" (Watson, 1994).

Transformasi citra dilakukan dengan menggunakan DCT (Discrete Cosine Transform), sehingga dapat dikatakan bahwa penyisipan dilakukan pada ranah DCT. Penyisipan dilakukan terhadap citra bitmap dengan kedalaman warna 24 bit. DCT digunakan untuk metransformasikan nilai intensitas blok 8x8 pikselnya yang berurutan dari image menjadi 64 koefisien DCT ke dalam frekuensi dasarnya, diubah koefisien-koefisiennya dan kemudian ditransformasikan kembali dengan IDCT (*Inverse Discrete Cosine Transform*).

Proses DCT adalah proses transformasi data dari dominan spasial ke domain frekuensi, yang mampu memisahkan informasi pada frekuensi rendah hingga frekuensi tinggi. Dalam sebuah citra, informasi utama yang dikandungnya lebih banyak terdapat pada frekuensi rendah dan menengah, sedang informasi frekuensi tinggi merupakan informasi *detail*. Ini sebagaimana sifat visual mata manusia yang tidak begitu sensitive pada perubahan informasi detail tetapi sangat sensitive pada perubahan informasi global. Ini menunjukkan bahwa menghilangkan informasi detail citra tidak akan menghilangkan makna informasi yang terkandung dalam citra itu. Untuk menghilangkan informasi detail pada frekuensi tinggi ini dapat dilakukan melalui proses DCT.[15]

Proses DCT 2-D ini dilakukan secara terpisah, dimulai dari proses DCT arah-x dan kemudian hasilnya dilakukan proses DCT arah-y. Secara matematis, proses DCT sebuah citra adalah jumlah perkalian antara fungsi cosinus diskret dan fungsi citra numeric 2-D f(x,y) yang diberikan pada persamaan 2.24 dan persamaan 2.25.

$$F_{(u,y)} = \sqrt{\frac{2}{N}}, C_u \sum_{x=0}^{N-1} cos\left[\frac{(2x+1)u\pi}{2N}\right] . f_{(x,y)} \quad (2.24)$$

$$F_{(u,v)} = \sqrt{\frac{2}{N}}, C_v \sum_{y=0}^{N-1} F_{(u,y)} . cos\left[\frac{(2x+1)v\pi}{2N}\right] \quad (2.25)$$

Dimana C_u, C_v= $\sqrt{\frac{1}{2}}$ bila u, v = 0 dan C_u, C_v = 1, bila $u, v > 0$.

Persamaan 2.24 merupakan hasil dari proses DCT 2-D yang mempresentasikan domain frekuensi citra.

Dimana x dan y adalah koordinat diskret dalam domain spasial citra, u dan v adalah koordinat diskret dalam domain frekuensi. N menunjukkan ukuran citra, dimana dalam proses ini harus merupakan persegi-empat.

Selanjutnya persamaan 2.25, ini dapat direpresentasikan dalam bentuk perkalian matriks seperti yang ditunjukkan pada persamaan 2.26.

$$[F]_{u,v} = ([DC]_{u,x} . [F]_{x,y} . [DC]_{y,v}) \quad (2.26)$$

Yang berlaku umum untuk matiks persegi-empat dan merupakan perpangkatan dua. $[D]_{u,x}$ dan $[D]_{y,v}$ secara berurutan adalah matriks cosinus arah sumbu=x dan arah sumbu-y yang dapat dihitung dengan persamaan 2.27 dan persamaan 2.28.

$$DC_{(u,x)} = \sqrt{\frac{2}{N}} C_u \, cos\left[\frac{(2x+1)u\pi}{2N}\right] \quad (2.27)$$

$$DC_{(y,v)} = \sqrt{\frac{2}{N}} C_v \, cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (2.28)$$

Untuk mendapatkan kembali citra asli dari koefisien DCT berdomain frekuensi dilakukan proses balik dengan DCT *invers* (iDCT). Secara matematis, proses ini diberikan pada persamaan 2.29 dan 2.30, serta untuk proses dalam bentuk perkalian matriks pada persamaan 2.31.

$$F_{(u,y)} = \sqrt{\frac{2}{N}} . C_v \sum_{v=0}^{N-1} F_{(u,v)} . cos\left[\frac{(2x+1)v\pi}{2N}\right] \quad (2.29)$$

$$f_{(x,y)} = \sqrt{\frac{2}{N}}, C_u \sum_{x=0}^{N-1} cos\left[\frac{(2x+1)u\pi}{2N}\right].F_{(u,v)} \quad (2.30)$$

Proses DCT invers diawali dengan perkalian matriks antara koefisien DCT $F_{(u,v)}$ dan matriks cosinus invers arah-y $[iDC]_{v,y}$, kemudian hasilnya dikalikan dengan matriks cosinus invers arah-x $[iDC]_{x,u}$. $[iDC]_{x,u}$ dan $[iDC]_{v,y}$ ini secara berurutan diberikan pada persamaan 2.31 dan persamaan 2.32.

$$iDC_{(x,u)} = \sqrt{\frac{2}{N}} C_u \, cos\left[\frac{(2x+1)u\pi}{2N}\right] \quad (2.31)$$

$$iDC_{(v,y)} = \sqrt{\frac{2}{N}} C_v \, cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (2.32)$$

2.4.2 *Principal Component Analysis* (PCA)

Principal Component Analysis (PCA) adalah teknik statistik untuk menyederhanakan kumpulan data banyak-dimensi menjadi dimensi yang lebih rendah (*extration feature*)[13]. PCA merupakan.transformasi linier ortogonal yang mentransformasi data kesistem koordinat baru, sehingga keragaman terbesar dengan suatu proyeksi berada pada koordinat pertama (disebut principal komponen pertama), keragaman terbesar kedua berada pada koordinat kedua dan seterusnya. Konsep penggunaan PCA meliputi perhitungan nilai-nilai simpangan baku, matriks kovarian, nilai karakteristik (eigen value) dan vector karakteristik (eigen vector). PCA dapat menggunakan metode kovaransi atau korelasi. Jika diperlukan, data distandardisasi terlebih dahulu sehingga menghampiri sebaran normal baku. Dalam hal ini digunakan metode kovariansi dengan algoritma berikut:

1. Mengumpulkan data dalam bentuk matriks X berukuran M x N. Misalkan adalah vector N x 1.
2. Menghitung rata-rata dengan persamaan 2.29.

   $$\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x_i \quad (2.33)$$
3. Menghitung selisih rata-rata dengan persamaan 2.30

   $$.\Phi_i = x_i - \bar{x} \quad (2.34)$$
4. Menentukan matriks kovarian dari matriks X= $[\Phi_1,\Phi_2,\ldots,\Phi_M]$ (matriks N x M)

5. Menghitung kovarian dengan persamaan 2.31.

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^{\mathrm{T}} = \mathrm{XX}^{\mathrm{T}} \quad (2.35)$$

6. Menentukan nilai karakteristik dan vector karakteristik dari matrik kovarian dengan persamaan 2.32.

$$\mathrm{C} = \lambda_1 > \lambda_2 > \ldots\ldots > \lambda_N \text{ dan } \mathrm{C} = u_1, u_2, \ldots\ldots, u_n \quad (2.36)$$

7. Mengurutkan vector karakteristik *u* dan nilai karakteristik dalam matriks diagonal dalam urutan menurun sesuai dengan nilai peluang kumulatif terbesar untuk tiap vector karakteristik sehingga diperoleh nilai-nilai karakteristik yang dominan.

2.5 Metode Haar Cascade Classifier

Proses deteksi adanya citra mata dalam sebuah gambar pada OpenCV, menggunakan sebuah metode yang dipublikasikan oleh Paul Viola dan Michael Jones tahun 2001. Umumnya disebut metode Haar Classifier. Metode ini merupakan metode yang menggunakan *statistical model* (*classifier*).[17][18][19]: Pendekatan untuk mendeteksi objek dalam gambar menggabungkan empat konsep utama:

1. Training data
2. Fitur segi empat sederhana yang disebut fitur Haar.
3. Integral image untuk pendeteksian fitur secara cepat.
4. Pengklasifikasi bertingkat (Cascade classifier) untuk menghubungkan banyak fitur secara efisien.

2.5.1 *Training* data pada Haar

Metode ini memerlukan 2 tipe gambar objek dalam proses *training* yang dilakukan, yaitu :

1. *Positive samples*

Berisi gambar obyek yang ingin di deteksi, apabila ingin mendeteksi mata maka *positive samples* ini berisi gambar mata, begitu juga obyek lain yang ingin dikenali.

2. *Negative samples*

Berisi gambar obyek selain obyek yang ingin dikenali, umumnya berupa gambar *background* (tembok, pemandangan, lantai, dan gambar lainnya). Resolusi untuk sampel negatif disarankan untuk memiliki resolusi yang sama dengan resolusi kamera.

Training dari Haar menggunakan dua tipe sampel diatas. Informasi dari hasil training ini lalu dikonversi menjadi sebuah parameter model statistik.

2.5.2 Sistem Kerja Algoritma *Haar Cascade Classifier*

Algoritma Haar menggunakan metode statistikal dalam melakukan pengenalan mata. Metode ini menggunakan *simple haar-like features* dan juga *a cascade of boosted tree classifier.* Classifier ini menggunakan gambar berukuran tetap. Cara kerja dari haar dalam mendeteksi mata adalah menggunakan teknik *sliding window* pada keseluruhan gambar dan mencari apakah terdapat bagian dari gambar yang berbentuk seperti mata atau tidak. Haar juga memiliki kemampuan untuk melakukan *scalling* sehingga dapat mendeteksi adanya mata yang berukuran lebih besar ataupun lebih kecil dari gambar pada *classifier*.

Tiap *feature* dari *haar-like feature* didefinisikan pada bentuk dari *feature*, diantaranya koordinat dari *feature* dan juga ukuran dari *feature* tersebut.

2.5.3 Pendeteksian Wajah (Haar Cascade Classifier)

Metode yang digunakan dalam penelitian ini untuk mendeteksi keberadaan wajah yang berasal dari webcam adalah *Haar Cascade Classifier*. Dimana fungsi detektor wajah akan menerima masing masing lokasi *image* guna mengklasifikasikan area wajah atau bukan area wajah.

Pengklasifikasi *Haar Cascade* tersimpan pada file dengan ekstensi XML yang berfungsi untuk memutuskan bagaimana pengklasifikasian masing-masing lokasi keberadaan wajah. Pengklasifikasi yang bias di *download* ada beberapa macam, diantaranya: pengklasifikasian wajah tambak depan (*frontal face*), klasifikasi profil wajah (*profile face*), pengklasifikasi seluruh tubuh (*full*

body/pedestrian), pengklasifikasian untuk tubuh bagian atas (*upper body*) dan pengklasifikasian tubuh bagian bawah (*lower body*).

Pada *OpenCV*, pengklasifikasi *Haar Cascade Classifier* untuk mendeteksi wajah tampak depan ada beberapa pilihan yang bias dimanfaatkan, yaitu:

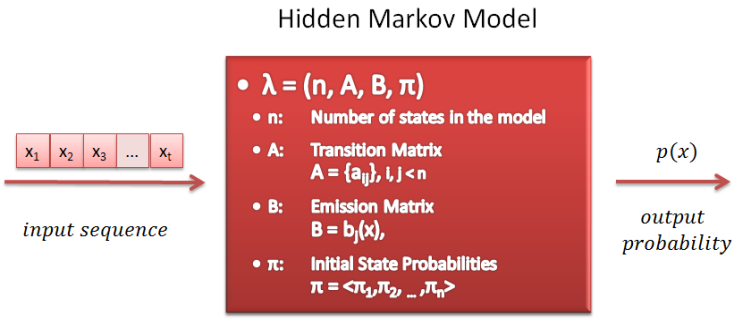
1. *"haarcascade_frontalface_default.xml"*
2. *"haarcascade_frontalface_alt.xml"*
3. *"haarcascade_frontalface_alt2.xml"*
4. *"haarcascade_frontalface_alt_tree.xml"*

Keempat klasifikasi tersebut akan memberikan hasil deteksi yang berbeda-beda, tergantung program dan kondisi lingkungan sekitarnya. Pada penelitian ini digunakan "*Haarcascade_frontalface_alt_tree.xml*".

2.6 *Hidden Markov Model* (HMM)

Model Markov Tersembunyi atau lebih dikenal sebagai *Hidden Markov Model* (HMM) adalah sebuah model statistik dari sebuah sistem yang diasumsikan sebuah Proses Markov dengan parameter yang tak diketahui, dan tantangannya adalah menentukan parameter-parameter tersembunyi (*state*) dari parameter-parameter yang dapat diamati (*observer*). Parameter-parameter yang ditentukan kemudian dapat digunakan untuk analisis yang lebih jauh, misalnya untuk aplikasi pengenalan pola (*Pattern Recognition*). Sebuah HMM dapat dianggap sebagai sebuah *Bayesian Network* dinamis yang paling sederhana.

Pada model Markov umum (*Vanilla/Visible Markov Model*), s*tate*-nya langsung dapat diamati, oleh karena itu probabilitas transisi keadaan (*state)*menjadi satu-satunya parameter. Di dalam Model Markov yang tersembunyi, *state*-nya tidak dapat diamati secara langsung, akan tetapi yang dapat diamati adalah variabel-variabel yang terpengaruh oleh *state*. Setiap *state*memiliki distribusi probabilitas atas token-token output yang mungkin muncul. Oleh karena itu rangkaian token yang dihasilkan oleh HMM memberikan sebagian informasi tentang sekuens state-state. [5][17][18][19][20]



Gambar 2.9 HMM Model

2.6.1 Penggunaan HMM

Ada 3 (tiga) permasalahan utama yang dapat diselesaikan HMM yaitu :

1. Diberikan parameter dari model, hitunglah probabilitas output berupa suatu barisan tertentu. Masalah ini diselesaikan oleh forward algorithm.
2. Diberikan parameter dari model, carilah barisan state tersembunyi yang palingmungkin menghasilkan output barisan tertentu (yang diberikan). Masalah ini diselesaikan oleh algoritma Viterbi.
3. Diberikan sebuah barisan output atau himpunan barisan sepertinya, maka temukan himpunan transisi *state* yang paling mungkin beserta probabilitas outputnya. Dengan kata lain, latihlah parameter HMM jika diberikan dataset barisan-barisan tertentu. Masalah ini diselesaikan oleh algoritma Baum-Welch.

2.6.2 Probabilitas Barisan (Sequence) yang Teramati

Probabilitas mengamati barisan Y = y(0), y(1), ... , y(L-1), dengan panjang barisan L diberikan oleh persamaan:

Di mana penjumlahannya meliputi seluruh node tersembunyi (hidden) pada barisan X = x(0), x(1), ... , x(L-1). Perhitungan kasar (brute force) P(Y) termasuk *intractable* untuk masalah-masalah di dunia nyata, karena jumlah hidden node yang sangat besar (pada umumnya). Akan tetapi, perhitungan nya dapat sangat dipercepat menggunakan algoritma *dynamic programming* yang dinamai *forward algorithm.*

2.6.3 Step by Step HMM

Berikut merupakan langkah-langkah analisis metode pada HMM yaitu menentukan *markov chain*, parameter HMM dan kemudian menyelesaikan permasalah pada HMM. [18][19][20][21]

2.6.3.1 Markov Chains

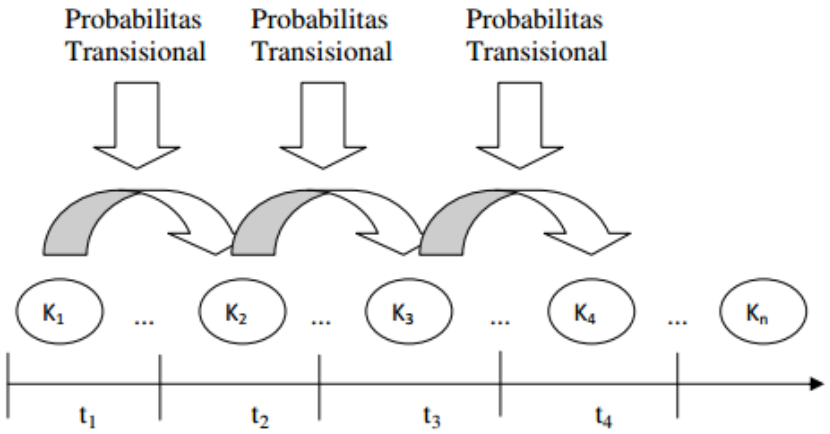
Rantai Markov *(Markov Chains)* adalah merupakan suatu teknik matematika yang biasa digunakan untuk melakukan pemodelan (modeling) berbagai macam sistem dan proses bisnis. Teknik ini dapat digunakan untuk memperkirakan perubahan-perubahan di waktu yang akan datang dalam variabel-variabel dinamis atas dasar perubahan-perubahan dari variabel-variabel dinamis tersebut di waktu yang lalau. Teknik ini dapat digunakan juga untuk menganalisis kejadian-kejadian di waktu-waktu mendatang secara matematis.

Model Rantai Markov (Markov Chains) ditemukan oleh seorang ahli Rusia yang bernama Andrey Markov pada tahun 1906, yaitu: "Untuk setiap waktu t, ketika kejadian adalah K_t dan seluruh kejadian sebelumnya adalah $K_{t(j)}$... , $K_{t(j-n)}$yang terjadi dari proses yang diketahui, probabilitas seluruh kejadian yang akan datang Kt(j) hanya bergantung pada kejadian $K_{t(j-1)}$dan tidak bergantung pada kejadian-kejadian sebelumnya yaitu $K_{t(j-2)}$,$K_{t(j-3)}$,..., $K_{t(j-n)}$.".

Rantai Markov *(Markov Chains)* apabia diberikan inputan keadaan dari keadaan saat ini, keadaan akan datang dapat diprediksi dan dapat di lepas dari keadaan di masa lampau. Artinya, deskripsi kondisi saat ini menangkap semua informasi yang mempengaruhi evolusi dari suatu sistem dimasa depan. Dengan kata lain, kondisi masa depan dituju dengan menggunakan prbabilitas bukan dengan determinitas.

Gambaran mengenai rantai Markov ini kemudian gerakan-gerakan dari beberapa variabel di masa yang akan datang bisa diprediksi berdasarkan gerakan-gerakan variabel tersebut pada masa lalu. K_{t4} dipengaruhi oleh kejadian K_{t3}, K_{t3} dipengaruhi oleh kejadian K_{t2} dan demikian seterusnya dimana perubahan ini

terjadi karena peranan probabilitas transisi (transition probability). Kejadian K_{t2} misalnya, tidak akan mempengaruhi kejadian K_{t4}.



Gambar 2.10 Markov Chains

Rantai Markov *(Markov Chains)* bermanfaat untuk menghitung probabilitas suatu kejadian teramati yang secara umum dapat dirumuskan sebagai berikut:

$$P(\sigma_i) = P(\sigma_t|\sigma_{t-1}, \sigma_{t-2}, \sigma_{t-3}, \dots, \sigma_{t-n}) \quad (2.37)$$

$\sigma_t$adalah kondisi saat ini, dan σ_t adalah kondisi pada waktu tertentu yang berhubungan dengan σ_t. Sedangkan σ_{t-i} adalah kondisi sebelum σ_t. Kemudian dapat diasumsikan bahwa sebelah kanan persamaan bersifat invariant, yaitu, dihipotesiskan dalam keseluruahan sistem, transisi diantara keadaan tertentu tetap sama dalam hubungan probabilistiknya. Berdasarkan asumsi tersebut, dapat terbentuknya suatu stet keadaan probabilitas transisi diantara dua keadaan $K_i$dan t_j :

$$Probabilitas\ Transisi = P(\sigma_t = K_t|\sigma_{t-1} = t_j), 1 \leq i, j \leq N \quad (2.38)$$

2.6.3.2 Parameter HMM

HMM mempunyai parameter-parameter distribusi sebagai berikut :

a. Probabilitas Transisi (A)

Parameter A merupakan parameter dengan ukuran MxM dengan M adalah jumlah state yang ada, parameter transisi dapat dituliskan dalam bentuk matriks seperti berikut:

$$A_{ij} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2n} \\ A_{31} & A_{32} & A_{33} & \cdots & A_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \cdots & A_{nn} \end{bmatrix}$$

$$A = \{a_{ij}\}, a_{ij} = P_r(x_{t+1} = q_j | x_t = q_i), 1 \leq j, i \leq N \quad (2.39)$$

dimana:

a_{ij} adalah problabilitas transisi i ke j

x_t adalah *state* pada saat waktu t

x_{t+1} adalah perpindahan *state*

t adalah waktu

N adalah jumlah state

Pr adalah problabilitas

b. Probabilitas observasi (B)

Parameter B merupakan probabilitas observasi atau probabilitas state merupakan kemunculan suatu state deretan seluruh state yang ada. Parameter B didalam HMM dituliskan dalam bentuk matriks kolom dengan MxN, dimana M merupakan jumlah seluruh state yang ada. Parameter B dapat dituliskan dalam bentuk matriks seperti berikut :

$$B_{mn} = \begin{bmatrix} B_{11} & B_{12} & B_{13} & \cdots & B_{1n} \\ B_{21} & B_{22} & B_{23} & \cdots & B_{2n} \\ B_{31} & B_{32} & B_{33} & \cdots & B_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & B_{m3} & \cdots & B_{mn} \end{bmatrix}$$

$$B = \{b_i\}, b_i(k) = P_r(O_t = V_k | x_t = q_i\ t) \quad (2.40)$$

dimana:

V_k menunjukkan simbol pengamatan ke-k dan O_t merupakan parameter vektor pada saat itu.

c. Distribusi keadaan awal(π)

Parameter phi (π), disebut sebagai parameter awal, merupakan probabilitas kemunculan suatu state di awal. Sama halnya dengan parameter B, parameter π juga dituliskan dalam bentuk matriks kolom dengan ukuran Mx1, dimana M adalah jumlah state-nya, parameter π dapat dituliskan dalam bentuk seperti berikut :

$\pi = [\pi_1, \pi_2, \pi_3, \dots, \pi_n]$

$$\pi = \{\pi_i\}, \pi_i = P_r(x_0 = q_i) \quad (2.41)$$

Sedangkan parameter tertentu HMM ada dua yaitu N dan M:

a. N, jumlah keadaan model. Dinotasikan himpunan terbatas untuk keadaan yang mungkin adalah $Q = \{q_1, \dots, q_N\}$

b. M, jumlah dari simbol observasi/keadaan, ukuran huruf diskret. Simbol observasi berhubungan dengan keluaran fisik dari sistem yang dimodelkan. Dinotasikan himpunan terbatas untuk observasi yang mungkin adalah

$V = \{V_1, \dots, V_M\}$

Parameter-parameter HMM ditaksir berdasarkan kriteria *maximum likelihood* (ML) dan algoritma *Baum-Welch* (EM = *Expectation Modification*).

2.6.3.3 Permasalahan Pada HMM

Agar HMM dapat diaplikasikan ke berbagai masalah nyata, ada tiga masalah mendasar dalam HMM yang harus diselesaikan [24][25], yakni:

2.6.3.3.1 Permasalahan Evaluasi

Yakni akan mencari *P(O|λ)* atau peluang dari barisan observasi:

O=$\{O_1 = V_{m_1}, O_2 = V_{m_2}, \dots, O_T = V_{m_T}\}$ jika diberikan HMM ; λ= (A,B,$\boldsymbol{\pi}$).

Peluang ini dapat ditentukan secara induksi dengan menggunakan algoritma *forward*.

Definisikan variabel *forward* $\alpha_t(i)$:

$$\alpha_t(i) = P(O_1 = V_{m_1}, O_2 = V_{m_2}, \dots, O_t = V_{m_t}, Q_t = S_i|\lambda) \quad (2.42)$$

Yaitu peluang barisan observasi $O_1, O_2, O_3, \dots, O_t$ dan *state* S_i pada waktu *t* jika diberikan λ.

Secara induktif *P(O|λ)* dapat dihitung sebagai berikut:

1. Inisialisasi

Untuk state (t) =1, dimana termasuk observasi(1) = 1

$$\alpha_i(i) = \pi(\mathrm{i})\mathrm{b}_i(O_1) \quad (2.43)$$

2. Induksi

Pada tahap ini akan dihitung $\alpha_2(j) dan\ \alpha_3(j)$

Probabilitas pada t = 2 sebagai berikut :

$$\alpha_{i+1}(j) = \left[\sum_{i=1}^{n} \alpha_1(i)\alpha_{ij}\right]\mathrm{b}_j(\mathrm{O}_2) \quad (2.44)$$

3. Terminasi

Dari tahap terminasi maka algoritma menghaslkan penyelesaian sebagai berikut :

$$P(O \mid \lambda) = \sum_{i=1}^{n} a_T(i) \quad (2.45)$$

2.6.3.3.2 Permasalahan Decode

Akan dicari barisan state yang optimal Q^* =$\{Q_1^*, Q_2^*, \dots, Q_T^*\}$jika diberikan barisan observasi *O =*$\{O_1, O_2, \dots, O_T\}$dan model λ = (A B, ,π). Barisan state terbaik yang akan ditentukan yaitu berupa lintasan tunggal yang terhubung dari *t =1,2, ...,T*.

Untuk menyelesaikan decoding problem ini digunakan algoritma Viterbi. Dalam algoritma Viterbi digunakan dua variabel bantu, yaitu:

1. $\delta(i) = \max_{Q_1, Q_2, \dots, Q_{t-1}} P(Q_1 = S_i, \dots, Q_T = S_i, O_1 = V_{m_1}, \dots, O_t = V_{m_t} \mid \lambda)$ (2.46)

Dengan induksi diperoleh:

$$\delta_{t+1}(j) = \max_{1 \le i \le N}\left[\delta_t(i).a_{ij}\right].b_j(O_{t+1}) \quad (2.47)$$

2. $\psi_t(i) = \underset{Q_1,Q_2,\ldots,Q_{t-1}}{\operatorname{argmax}} P(Q_1 = S_{i_1}, \ldots, Q_t = S_i, O_1 = V_{m_1}, \ldots, O_t = V_{m_t} \mid \lambda)$

(2.48)

Langkah-langkah dalam algoritma Viterbi untuk menentukan barisan state terbaik yaitu:

1. Inisialisasi

$\delta(i) = \pi_i . b_i(O_1), \quad 1 \le i \le N$ (2.49)

$\psi_t(i) = 0$

2. Rekursi

$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) . a_{ij}] . b_j(O_t), \quad 2 \le t \le T \quad 1 \le j \le N$ (2.50)

$\psi_t(j) = \underset{1 \le i \le N}{\operatorname{argmax}} [\delta_{t-1}(i) . a_{ij}], \quad 2 \le t \le T \quad 1 \le j \le N$ (2.51)

3. *State* terbaik pada waktu T (Q_T)

$P^* = \max[\delta_T(i)] \quad 2 \le t \le T$

$q_T^* = \arg\max[\delta_T] \quad 1 \le i \le N$ (2.52) dan (2.53)

4. Barisan *state* terbaik pada *t*=*T*-1,*T*-2,….,1

$q_T^* = \psi_{t+1}(q_{t+1}^*) \quad t = T-1, T-2, \ldots 1$ (2.54)

2.6.3.3.3 Permasalahan Learning

Jika diberikan sebuah HMM, dan barisan observasi $O = O_1, O_2, \ldots, O_T$, bagaimana mengatur parameter model $\lambda = (A\ B, \pi)$ agar $P(O|\lambda)$ maksimum.

Untuk menyelesaikan masalah ini digunakan Algoritma *Baum-Welch*. Dalam algoritma *Baum-Welch*, didefinisikan tiga variabel, yaitu variabel *forward*, variabel *backward*, dan variabel $\xi(i,j)$. Variabel *forward* dan variabel *backward* akan digunakan dalam perhitungan variabel $\xi(i,j)$ dan variabel $\gamma_t(i)$.

Variabel pertama, variabel *forward* telah didefinisikan pada persamaan (2.43), serta tahapan induksi diberikan oleh persamaan (2.44).

Variabel kedua, yakni variabel *backward* didefinisikan sebagai,

$\beta_t(i)$= P($O_{t+1}, O_{t+2,\ldots,} O_T | Q_t = S_i, \lambda$)

yaitu, peluang dari barisan observasi parsial $O = O_1, O_2, \ldots, O_T$, diberikan *state*$S_i$ pada waktu t dan model λ. Selanjutnya, $\beta_t(i)$ dapat diselesaikan secara induksi sebagai berikut:

1. Inisialisasi

$\beta_T(i)$=1, $1 \leq j \leq N$ (2.55)

2. Induksi

$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j)$, $t = T-1, T-2, \ldots, 1, 1 \leq j \leq N$ (2.56)

Variabel ketiga, yakni variabel $\xi(i,j)$ didefinisikan sebagai,

$\xi(i,j)$= P($Q_t = S_i, Q_{t+1} = S_j, O | \lambda$) (2.57)

yaitu, peluang proses pada saat t berada pada *state*S_i dan pada saat t+1 berada pada state S_j, jika diberikan barisan O dan model λ.

Persamaan (2.55) dapat ditulis

$$\xi(i,j) = \frac{P(Q_t = S_i, Q_{t+1} = S_j, O|\lambda)}{P(O|\lambda)} \quad (2.58)$$

atau dapat diekspresikan dengan variabel *forward-backward* sebagai

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(i)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(i)} \quad (2.59)$$

2.7 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 berorientasiobjek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, polymorphism (polimorfisme) dan inheritance (pewarisan).

2.7.1 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.

2.7.2 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.

2.7.3 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.

2.7.4 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.

2.8 UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi syarat dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. UML menawarkan sebuah syarat untuk merancang model sebuah sistem. UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering). UML Menggabungkan konsep berarah objek seperti Booch, OMT, OOSE, dsb sehingga menjadi bahasa pemodelan tunggal. UML menekankan pada apa yang dapat dikerjakan dengan metode-metode tersebut.

2.8.1 Analisis dan desain Berorientasi Objek (OOAD)

OOAD adalah metode analisis yang memerikasa requirements dari sudut psayang kelas kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek system atau subsistem. OOAD merupakan cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek,yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas.

1. Objek

Abstraksi dan sesuatu yang mewakili dunia nyata. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan.

2. Kelas

Kumpulan objek-objek dengan karakter yang sama. Sebuah kelas mempunyai sifat(atribut),kelakuan (operasi/metode),hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dari kelas yang lain,dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2.8.2 Komponen UML

Berikut ini merupakan komponen-komponen dari UML yaitu :

2.8.2.1 View

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. View bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram.

Beberapa jenis view dalam UML antara lain: use case view, logical view, component view, concurrency view,dan deployment view.

a. *Use case view*

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan external actors. Actor yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. View ini digambarkan dalam use case diagrams dan kadang-kadang dengan activity diagrams. View ini digunakan terutama untuk pelanggan, perancang (designer), pengembang (developer), dan penguji sistem (tester).

b. *Logical view*

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (class, object,danrelationship) dan kolaborasi dinamis yang terjadi ketika object mengirim pesan ke object lain dalam suatu fungsi tertentu.

View ini digambarkan dalam class diagrams untuk struktur statis dan dalam state, sequence, collaboration, dan activity diagram untuk model dinamisnya. View ini digunakan untuk perancang (designer) dan pengembang (developer).

c. *Component*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari code module diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya.

View ini digambarkan dalam component view dan digunakan untuk pengembang (developer).

2.9 *Use Case Diagram*

Usecase diagram adalah sebuah gambaran dari fungsi sistem yang dipasang dari sudut pandang pemakai yang memperkenalkan suatu sistem yang akan dibangun.

2.9.1 Actor

Actor adalah segala sesuatu yang perlu berinteraksi dengan sistem untuk pertukaran informasi.

2.10 *Sequence Diagram*

Diagram sequence merupakan salah satu yang menjelaskan bagaimana suatu operasi itu dilakukan; message (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.

2.11 *Class Diagram*

Class diagram adalah sebuah grafik presentasi dari gambaran statis yang menunjukan sekumpulan model elemen yang terdeklarasi (statis), seperti kelas, tipe dan isinya serta hubungannya.