BAB 2
TINJAUAN PUSTAKA

2.1 Profil Mirzatama Raya

Profil Mirzatama Raya berisi penjelasan yang memaparkan sejarah, visi, misi, tujuan dan struktur organisasi Mirzatama Raya.

2.1.1 Sejarah MIRZATAMA RAYA

Mirzatama Raya (MTR) yaitu sebuah perusahaan yang bergerak dalam bidang jasa service dan maintenance rotating equipment seperti gas turbine, steam turbine, centrifugal pump, centrifugal compressor dan turbomachinery. Perusahaan ini berdiri sejak tahun 17 Agustus 2004 yang didirikan oleh Rudi Fuad sebagai pemilik utama perusahaan yang berfokus pada jasa services akan tetapi seiring berkembangnya perusahaan maka merambah ke manufactur. Perusahaan ini pada awalnya hanya memiliki 4 orang karyawan yang memegang peranan penting seperti staff engineer, staff bisnis dan staff drafting, dan staff produksi seiring berkembangnya perusahaan bertambah hingga 30 karyawan tetap hingga saat ini. Perusahaan ini berkembang pesat sehingga mendapatkan penghargaan sebagai vendor terbaik pada tahun 2017 pada acara *gathering of vendor*. Perusahaan ini sudah merambah ke berbagai bidang kebutuhan penghasil listrik tenaga air bahkan tenaga uap.

2.1.2 Visi, Misi Dan Tujuan

Visi merupakan suatu pandangan jauh tentang perusahaan, tujuan-tujuan perusahaan dan apa yang harus dilakukan untuk mencapai tujuan tersebut pada masa yang akan datang. Sedangkna misi adalah pernyataan tentang apa yang harus dikerjakan oleh perusahaan dalam upaya mewujudkan visi. Adapun visi misi dari mirzatama raya adalah sebagai berikut :

1. **Visi**

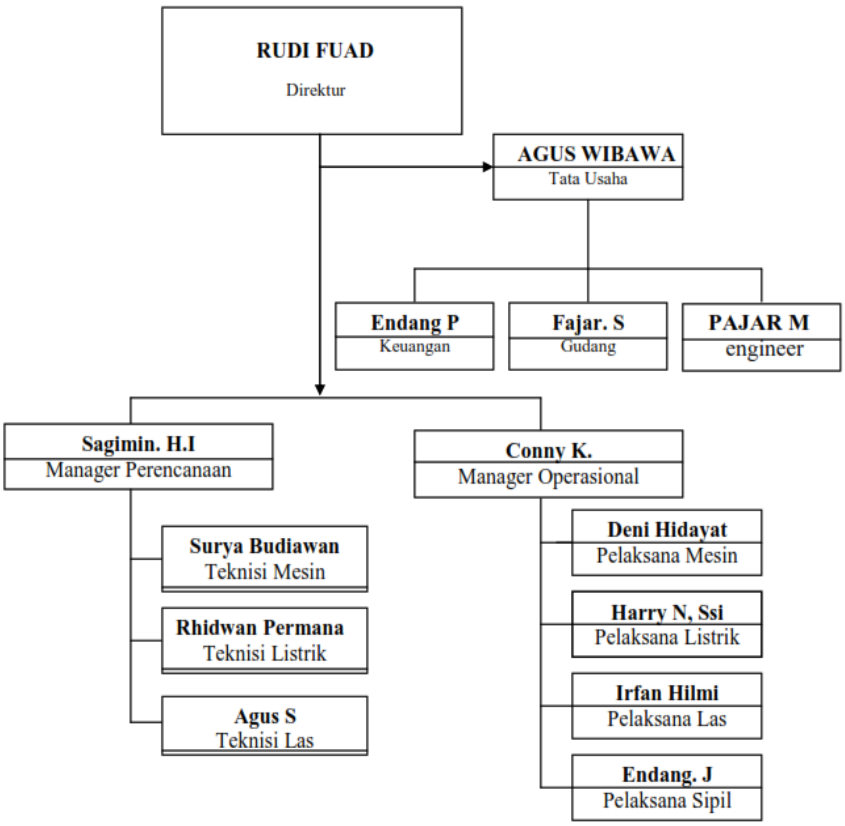
Menyebarkan keuntungan dengan mengembangkan industri pada penguasaan ilmu pengetahuan dan teknologi muktahir.

1. Misi

1. Membangun pembangunan yang berkelanjutan pada SDM.
2. Mengelola kerjasama dengan penguasaan ilmu pengetahuan dan teknologi pemilik.
3. Menambahkan nilai bagi *stakeholder* dengan menerapkan etika bisnis dan aktif dalam *corporate social responsibility*.

2.1.3 Struktur Organisasi

Mirzatama raya (MTR) terdiri dari beberapa bagian yang saling memiliki peranan penting. Berikut struktur organisasi pada perusahaan ini dapat dilihat pada Gambar 2.1:



Gambar 2.1 Struktur Organisasi Mirzatama Raya

2.2 Landasan Teori

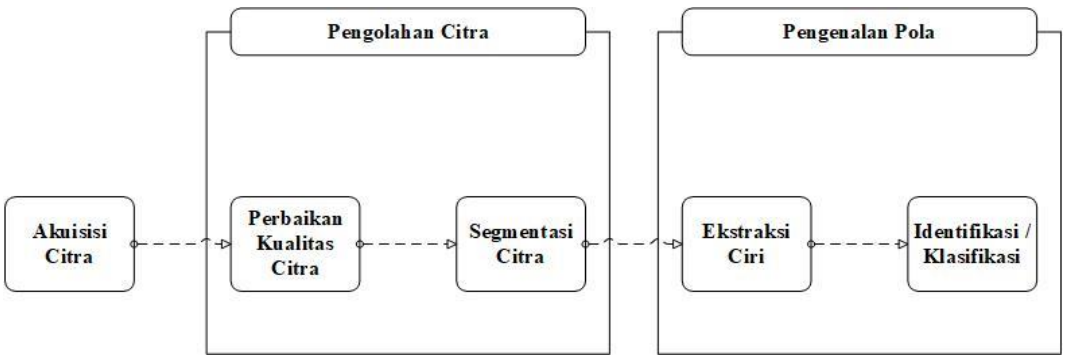
Landasan teori merupakan kumpulan teori – teori yang menjadi dasar dalam pembangunan aplikasi yang dikutip dari berbagai referensi. Landasan teori dimanfaatkan untuk penelitian sebagai pengetahuan dasar dari penelitian yang akan dilakukan.

2.2.1 Presensi

Presensi menurut KKBI berarti kehadiran sedangkan absensi berarti ketidakhadiran, presensi biasa diterapkan untuk menjaga kedisiplinan dari suatu institusi atau perusahaan. Presensi sendiri yaitu suatu pencatatan data kehadiran bagian dari informasi aktivitas suatu institusi yang berisi data – data kehadiran yang diatur atau disusun sedemikian rupa sesuai kebutuhan informasi dari institusi tersebut, Sehingga ketika ada pihak yang membutuhkan informasi tersebut dapat dicari dengan mudah.

2.2.2 Visi Komputer (*Computer Vision*)

Visi komputer adalah bidang interdisipliner yang berhubungan dengan bagaimana komputer dapat dibuat untuk mendapatkan pemahaman tingkat tinggi dari gambar digital atau video. Dari perspektif teknik, ia berusaha untuk mengotomatisasi tugas-tugas yang dapat dilakukan oleh sistem visual manusia. Dalam *computer vision* pengolahan citra (*image processing*) merupakan tahapan awal yang selanjutnya akan dilakukan pengenalan pola (*pattern recognition*). Berikut ini merupakan langkah – langkah yang umumnya dilakukan dalam merancang sebuah sistem pada *computer vision* (pengolahan citra dan pengenalan pola) dapat dilihat pada Gambar 2.2*:*



Gambar 2.2 Langkah Umum Computer Vision

Adapun langkah – langkah pada proses ini akan dijelaskan perpoin yaitu :

2.2.2.1 Akuisisi Citra

Akuisisi citra yaitu proses untuk menangkap dengan cara scan ataupun mengcapture citra dengan suatu alat tertentu sesuai domain kasus yang sedang diteliti.

2.2.2.2 Pengolahan Citra (*Image Processing*)

Pengolahan citra yaitu satu cabang dari ilmu informatika (komputer). Pengolahan citra berkutat pada usaha untuk melakukan transformasi suatu citra atau gambar menjadi citra yang lain dengan menggunakan teknik tertentu, agar sebuah citra mudah untuk dianalisa maka di ubah ke *grayscale* dengan memberi nilai yang sama yaitu 8 bit dari nilai 24 bit dengan warna RGB *(red,green,blue)* pada masing – masing warna . Berikut teknik yang umum digunakan :

1. **Perbaikan Kualitas Citra (*Image Enchancemet*)**

Perbaikan kualitas citra yaitu merupakan tahapan *pre – processing* dalam pengolahan citra yang bertujuan untuk meningkatkan kualitas citra tersebut. Indikator citra dengan kualitas yang baik yaitu hasil segmentasi, jika tanpa melalui proses perbaikan kualitas citra hasil dari akuisisi sudah dapat tersegmentasi dengan baik maka tahapan perbaikan citra boleh tidak dilakukan, akan tetapi hasil dari segmentasi belum baik maka perlu dilakukan tahapan perbaikan kualitas pada citra. Oleh sebab itu perbaikan citra bisa dikatakan bersifat opsional. Perbaikan kualitas citra dapat dilakukan pada operasi titik, operasi spasial dan operasi transformasi. Metode pada perbaikan kualitas citra di antaranya yaitu *itensity adjustment, contrast stretching, filtering* (*median filter, low pass filter,high pass filter,* dan sebagainya).

2. **Segmentasi Citra**

Dalam proses pengolahan citra terkadang dibutuhkan pengolahan hanya pada objek tertentu saja. Oleh sebab itu diperlukan proses untuk memisahkan objek yang dikehendaki dengan objek yang lainnya yang tidak dikehendaki. Proses pemisahan objek yang dikehendaki (*foreground*) dengan objek lain yang tidak dikehendaki (*background*) disebut

dengan proses segmentasi citra. Umumnya hasil dari proses segmentasi berupa citra biner yang terdiri dari 1 dan 0 (nol), dimana *foreground* berlogika 1 dan *background* berlogika 0. Metode pada segmentasi citra di antaranya yaitu *thresholding*, *multithresholding*, *active contour*, deteksi tepi, dan sebagainya.

2.2.2.3 Pengenalan Pola (*Pattern Recognition*)

Pengenalan pola yaitu suatu proses untuk mengklasifikasi objek sesuai kaidah ilmu dari cabang kecerdasan pada metode kelas – kelas tertentu untuk menyelesaikan masalah.

1. Ekstraksi Ciri (*Feature Extraction*)

Pada proses mengenali objek tertentu dibutuhkan parameter – parameter yang mencirikan objek tersebut. Ciri yang digunakan untuk membedakan objek yang satu dengan yang lainnya diantaranya yaitu ciri bentuk, ciri ukuran, ciri geometri, ciri tekstur dan ciri warna pada citra. Misalnya untuk mencirikan ukuran suatu objek yang termasuk dalam kelas ukuran besar maka digunakan parameter luas dan keliling. Nilai dari parameter-parameter tersebut kemudian dijadikan sebagai data masukan dalam proses identifikasi/ klasifikasi. Pada proses pengenalan pola yang kompleks dibutuhkan ciri yang kompleks pula, oleh sebab itu perlu dilakukan kajian mengenai ciri apa yang benar-benar dapat membedakan antara objek satu dengan objek yang lain.

2. Identifikasi Atau Klasifikasi

Dalam proses ini, nilai parameter-parameter yang merepresentasikan ciri objek pada masing-masing kelas dijadikan sebagai data masukan. Data tersebut kemudian diolah sehingga diperoleh suatu rumusan untuk dapat mengenali objek. Dalam tahapan identifikasi, umumnya dilakukan dua proses utama yaitu proses pelatihan dan proses pengujian. Proses pelatihan dilakukan menggunakan sekumpulan data latih yang memuat parameter ciri atau *feature* yang digunakan untuk membedakan antara objek satu dengan objek lainnya. Proses pelatihan memetakan data latih menuju target latih melalui suatu rumusan (algoritma identifikasi/klasifikasi). Algoritma yang digunakan dipilih berdasarkan pada karakteristik ciri atau *feature* dari objek. Algoritma yang biasa digunakan antara lain jaringan syaraf tiruan, *support vector machine, k-means clustering, k-nearest neighbor*, logika *fuzzy*, *fuzzy c-means clustering, naive bayes*, dan lain – lain.

2.2.2.4 Pengenalan Wajah (Face Recognition)

Pada pengenalan wajah terdapat dua jenis yang mendasar yaitu pada sistem *feature based* dan *image based*. Pada sistem pengenalan wajah dengan *feature based* yang akan diekstraksi meliputi komponen citra pada wajah itu sendiri seperti mata, hidung, mulut, dan lainnya. Dari hasil ekstraksi tersebut kemudian dimodelkan secara geometris. Sedangkan pada sistem pengenalan wajah *image based* menggunakan informasi dari piksel citra wajah yang kemudian direpresentasikan dalam metode tertentu yang sudah disesuaikan dengan pengambilan metode ini. Pada tahapan pengenalan wajah secara umum terbagi ke dalam dua bagian yaitu proses pelatihan dan proses pengenalan, diantaranya :

1. **Proses Umum Pelatihan Citra Wajah**

Pada tahap pendeteksian pada wajah secara garis besar akan dijelaskan sebagai berikut :

a. Pendeteksian Wajah *(Tracking Face)*

Pada tahap ini dilakukan pendeteksian secara langsung *(online)* jika sebuah masukan citra dari *video* maka dilakukakan proses *tracking face*, prinsif kerjanya hampir sama dengan yang dilakukan secara *offline*, dimana tahapan secara *offline* melakukan proses segmentasi dulu untuk memisahkan latarbelakang dengan area wajah tadi, lalu melakukan pemotongan dari area wajah jika terdeteksi dari masukan citra. Pada proses secara *online* menggunakan metode Viola – Jones. Berikut gambaran proses ini dapat dilihat pada Gambar 2.3:

Gambar 2.3 Proses *Face Tracking*

b. Penyelarasan Wajah (*Normalization Face*)

Pada proses ini dilakukan untuk menormalisasikan pada proses deteksi wajah karena banyak factor yang mempengaruhi pendeteksian seperti pencahayaan, ukuran, dan tingkat kejelasan citra dan sebagainya. Pada proses ini perlu preprocessing citra yang bertujuan untuk memudahkan saat pendeteksian diantaranya :

1. *Grayscaling*

Pada tahap proses penyelarasan wajah dilakukan konversi citra wajah dari citra wajah berwarna menjadi abu – abu. Citra wajah yang berwarna memiliki 3 parameter diantaranya merah *(Red)*, hijau *(Green)*, dan biru *(blue)*. Jika citra warna RGB langsung diekstraksi maka proses yang berlangsung kan sulit untuk dilakukan maka perlu dikonversi terlebih dahulu. Berikut persamaan untuk konversi ke *grayscale* [1] :

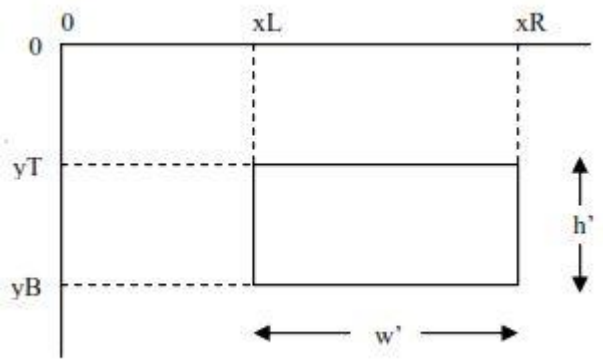
$$X = 0.299r + 0.587g + 0.114b \quad (2.1)$$

Dimana nilai dari RGB yaitu (r, g, b) dengan rentang nilai pada integer 0 – 255 dan nilai x sebagai grayscale dan angka – angka 0.299,0.587 dan 0.114 adalah bobot dari warnanya. Berikut ilustrasi hasil dari proses ini dapat dilihat pada Gambar 2.4 :

Gambar 2.4 *Grayscale* Citra

2. *Croping*

Pada proses ini dilakukan pemotongan pada bagian yang diperlukan saja dari citra wajah untuk proses ekstraksi. Pada proses pemotongan sesuai dengan dimensi dari proses segmentasi pada proses deteksi wajah dengan memberikan ciri kotak pada area wajah. Persamaan ini dapat dilihat pada 2.2, 2.3 dan ilustrasi hasilnya dapat dilihat pada Gambar 2.5, Gambar 2.6:



Gambar 2.5 Ilustrasi Proses *Croping* Citra Sumber[3]

Dimana pada persamaan ini :

$$x' = x - xL$$
$$y' = y - yT \quad (2.2)$$

Untuk x = xL sampai xR, untuk y = yT sampai yB dimana (xL,yT) dan (xR,yB) adalah koordinat titik pojok kiri atas dan pojok kanan bawah citra yang akan di-crop sehingga ukuran dari citra ini citra menjadi :

$$w' = xR - xL$$
$$h' = yB - YT \quad (2.3)$$

Berikut ilustrasi hasil dari proses croping ini dapat dilihat pada gambar dibawah ini :

Gambar 2.6 Ilustrasi Hasil *Croping* Citra

3. *Resizing*

Proses ini hanya untuk menormalisasikan citra dimensi pada wajah seperti pembesaran atau pengecilan dimensi citra sesuai kebutuhan . Tujuan dari proses ini untuk menyamakan setiap dimensi dari citra wajah yang dimasukan seperti 80 x 80 piksel. Berikut persamaannya dapat dilihat pada 2.4 dan ilustrasi hasilnya dapat dilihat pada Gambar 2.7 dibawah ini :

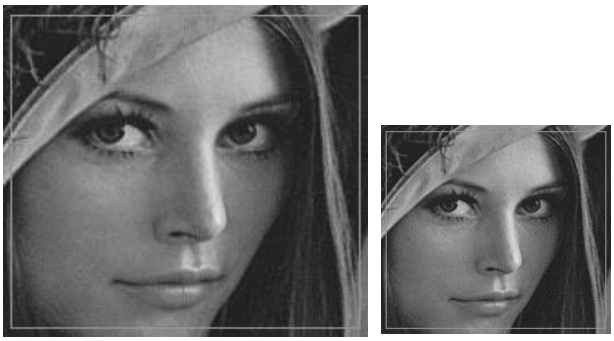
$$x' = Sh\ x$$
$$y' = Sv\ y \qquad (2.4)$$

dimana pada persamaan ini :

Sh = faktor skala horisontal

Sv = faktor skala vertikal

Sehingga ukuran citra menjadi w' = Sh w untuk lebar citra dan h' = Sv h untuk tinggi dari citranya.berikut hasil ilustrasi dari proses ini dapat dilihat pada gambar dibawah ini :



Gambar 2.7 Ilustrasi Hasil Proses *Resizing* Citra

4. *Equalizing*

Pada proses ini adalah tahapan akhir pada proses penyelarasan, yang bertujuan untuk memperjelas nilai histogram citra wajah. Dimana proses ini untuk memperjelas citra yang gelap, berikut persamaan dari histogram *equalizing* ini [1] :

$$K_0 = round\left(\frac{C_{i\,.}\left(2^K-1\right)}{w.h}\right) \quad (2.5)$$

Dimana :

C_i = Cacah/distribusi kumulatif nilai skala keabuan ke – i dari citra asli

Round = fungsi pembulatan ke bilangan terdekat, misal : 22,4 menjadi 22

K_0 = nilai keabuan hasil *histogram equalization*

W = lebar citra

H = tinggi citra

Gambar 2.8 *Equalizing Histogram* Citra

c. Ektraksi Fitur Wajah (*Extraction Fitures*)

Pada tahap ini yaitu proses untuk mendapatkan sebuah informasi dari citra yang bertujuan untuk membedakan citra wajah yang sudah diselaraskan. Pada proses ini menggunakan metode *principal component analysis* (PCA). Informasi dari ekstraksi fitur ini disebut dengan vektor fitur sebagai bentuk dasar pencarian citra berbasis konten seperti warna dan tekstur.

d. Simpan Fitur Wajah Ke Database (*Save Fiture Face To Database*)

Pada proses ini hanya menyimpan hasil ekstraksi fitur yaitu vektor fitur yang akan digunakan sebagai identitas ke dalam *database* yang nantinya digunakan sebagai pembanding citra wajah pada proses pengenalan wajah.

2. Proses Umum Pengenalan Citra Wajah

Pada tahap ini sama dengan tahapan umum pendeteksian citra wajah pada proses pelatihan juga akan tetapi yang membedakanya pada tahap akhir diantaranya :

a. Pendeteksian Wajah (*Tracking Face*).

b. Penyelarasan Wajah (*Normalization Face*).

c. Ektraksi Fitur Wajah (*Extraction Fitures*).

d. Pencocokan Fitur .

Pada point 1 sampe 3 sudah dijelaskan, tahapan ini berbeda dengan sebelumnya dimana proses ini hanya membandingkan fitur yang sudah diekstraksi ke database dengan fitur citra uji wajah yang sebelumnya melalui tahap pelatihan. Pada proses ini menggunakan metode pengenalan pola yaitu *nearest neighbours*.

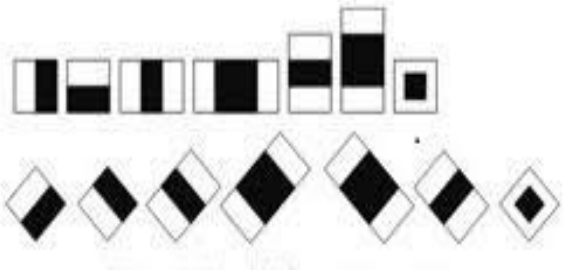
Proses ini membandingkan nilai fitur yang akan menghasilkan nilai jarak terdekat yang menandakan nilai fitur uji mendekati nilai fitur citra latih. Pada nilai jarak ini menjadi nilai masukan untuk nilai kemiripan citra pada wajah. Nilai kemiripan pada citra wajah merupakan nilai tingkat kemiripan dari citra uji dengan citra latih. Semakin besar nilainya menandakan bahwa citra wajah orang yang sedang diamati adalah citra wajah dari orang yang sama dengan citra wajahnya yang sudah tersimpan di *database*. Sebelum nilai kemiripan ini mengeluarkan hasil pengenalan wajah akan memalui proses *threshold* terlebih dahulu. Pada proses *threshold* ini merupakan proses penyaringan nilai kemiripan pada citra, dimana pada proses ini jika nilai kemiripan lebih besar dari *threshold* maka yang sudah ditentukan hasilnya akan ditampilkan jika lebih kecil maka tidak akan ditampilkan, tujuan dari proses ini untuk memberikan hasil yang bernilai benar atau memiliki kemiripan yang tinggi. Setelah proses *threshold* dilakukan proses pencarian tingkat akurasi, dimana pada proses ini menunjukan keakuratan dan ketepatan pada pengenalan wajah yang dimaksud agar sesuai dengan yang diharapkan.

2.2.3 Viola – Jones

Metode viola – jones merupakan salah satu metode yang paling populer digunakan karena memberikan hasil dengan keakuratan dan kecepatan yang cukup tinggi. Metode ini dikemukakan oleh Paul Viola dan Michael Jones, metode ini terbagi ke dalam 4 komponen yaitu *haar like feature, integral image, adaptive boosting* dan *cascade of classifier*[2].

1. *haar like feature*

Fitur *Haar* ini didasari oleh *Haar Wavelets* yaitu satu gelombang panjang berupa persegi dimana terdiri dari satu interval dengan nilai tinggi dan rendah yang digambarkan dengan persegi panjang berwarna terang dan hitam.



Gambar 2.9 Contoh *Haar Like Fiture*

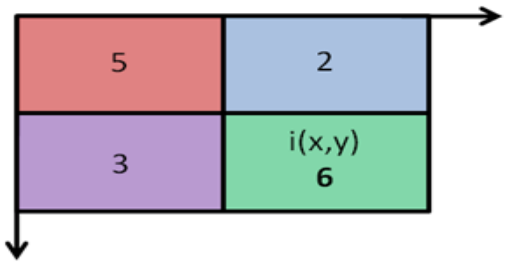
Berikut persamaan nilai *Haar Like feature* :

$$F(Haar) = \sum \text{putih} - \sum \text{hitam} \quad (2.6)$$

Dimana F adalah nilai fitur keseluruhan, ∑ putih nilai fitur pada area terang dan ∑ hitam pada area gelap. Fitur *Haar* ini ditentukan dengan mengitung selisih dari nilai rata – rata piksel terang dan gelap jika nilainya diatas ambang maka fitur tersebut dikatakan ada. Untuk menentukan fitur *Haar* ini dengan menggunakan teknik *integral image.*

2. *Integral Image*

dimana teknik ini mampu mempercepat proses pendeteksian suatu objek yang dikehendaki dengan menggabungkan bagian terkecil dari citra yaitu nilai – nilai peksel menjadi citra baru dengan menjumlahkan piksel yang ada disebelah kiri dan atas titik tersebut.



Gambar 2.10 Contoh *Integral Image*

Berdasarkan Gambar 2.10 diatas pada titik (x,y) maka persamaanya :

$$ii(x,y) = \sum x' \leq x, y' \leq y\, i(X',Y') \quad (2.7)$$

dimana ii(x,y) citra integral pada lokasi x,y dan i (x',y') nilai dari piksel citra asli.

Berikut contoh perhitungan dengan citra masukan berdimensi 5x5 dengan nilai piksel sebagai berikut dapat dilihat pada Tabel 2.1 :

Tabel 2.1 Contoh Perhitungan Integral Image

2	**4**	**7**	**5**	**8**
1	**5**	**9**	**7**	**7**
4	**6**	**8**	**5**	**6**
3	**5**	**6**	**6**	**7**
4	**4**	**5**	**3**	**6**

Maka *Integral Image* nya dapat dilihat pada Tabel 2.2:

Tabel 2.2 Hasil Perhitungan Integral Image

2	**6**	**13**	**18**	**26**
3	**12**	**28**	**40**	**55**
7	**22**	**46**	**63**	**84**
10	**30**	**60**	**83**	**111**
14	**38**	**73**	**99**	**133**

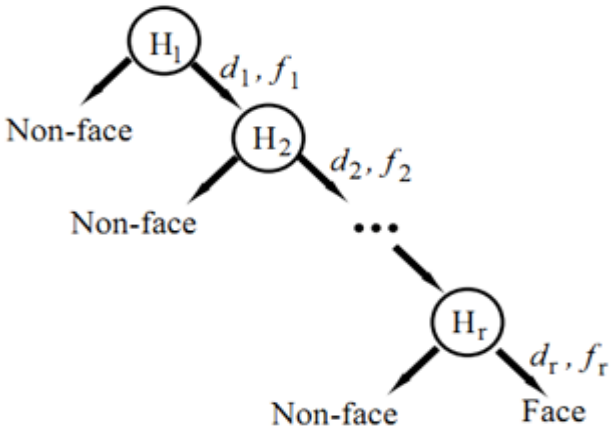
Caranya dengan menjumlahkan nilai piksel dengan nilai piksel sebelah kiri, kiri atas dan atas dari piksel tersebut, Sebagai contoh, nilai dari piksel pada baris ke-2 dan kolom ke-2 pada ilustrasi di atas diperoleh dengan cara menjumlahkan nilai piksel baris ke-1 kolom ke-1, piksel baris ke-1 kolom ke-2, piksel baris ke-2 kolom ke-1 dengan piksel baris ke-2 kolom ke-2, maka didapatkan piksel dengan nilai 12 (2+4+1+5).

Dengan integral image ini maka perhitungan untuk mendapatkan nilai fitur Haar didapatkan dengan waktu yang sangat cepat. Selanjutnya setelah fitur yang dikehendaki

didapatkan maka menggunakan metode *AdaBoost machine – learning* untuk mengetahui bahwa pada citra tersebut ada wajah apa tidak dalam suatu citra masukan.

3. *AdaBoost machine – learning*

Cara kerja metode *Adaboost* ini dengan menggabungkan banyak *classifier* yang lemah menjadi satu *classifier* kuat, *classifier* sendiri yaitu suatu ciri yang menandakan adanya objek wajah atau tidak pada suatu citra, tujuan dari penggabungan *classifier* lemah ini untuk menjadikan *classifier* yang kuat karena tiap *classifier* yang lemah memiliki tingkat kebenaran yang kurang akurat jadi mesti digabungkan. Berikut gambaran secara umum untuk mendeteksi wajah dengan metode *Adaboost* :



Gambar 2.11 *Metode Adaboost*

Pada gambar diatas menjelaskan bahwa pada metode ini akan menyeleksi nilai fitur dari citra sebagai inputan, jika pada inputan tersebut tidak ada indikasi fitur wajah maka akan diteruskan ke proses selanjutnya sampai pada proses terakhir. Pada proses terakhir ini akan disimpulkan ada fitur wajah yang terdeteksi dari inputan citra tadi, proses ini membandingkan nilai dari fitur dengan nilai ambang yang telah ditentukan sebelumnya pada setiap tingkat, jika nilai fitur sama dengan nilai ambang atau diatasnya maka terdeteksi adanya wajah dalam fitur tersebut. Fitur ini diurutkan berdasarkan bobot terberat ke bobot terendah sehingga dapat secepat mungkin untuk mengklasifikasi ada atau tidaknya wajah dalam citra masukan pengurutan ini disebut dengan *Cascade of classifier*. Berikut persamaan rumus dari metode ini :

$$F(x)=\sum_{t=1}^{T} \propto_t h_t(x) \quad (2.8)$$

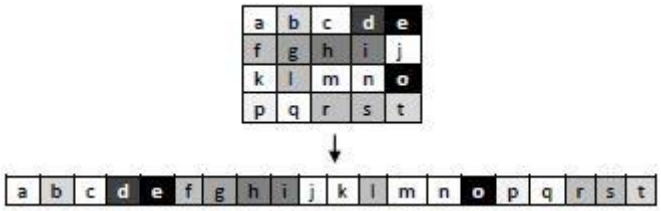
dimana $h_t(x)$ = *classifier* lemah , α = tingkat pembelajaran (*learning rate*) dan H(x) dilambangkan dengan F(x) *Strong Final Classifier.*

4. *Cascade Of Classifier*

Pada tahap ini mengkombinasikan *classifier* ke dalam suatu *Cascade Of Classifier* sehingga pada proses ini kecepatan pun meningkat karena pada proses hanya berfokus pada area citra yang berpeluang saja. Hal ini dilakukan untuk menentukan dimana letak objek yang sedang dicari pada suatu citra wajah.

2.2.4 *Principal Component Analysis* (PCA)

Principal Component Analysis (PCA) yaitu sebuah cara untuk mengidentifikasi sebuah pola pada data kemudian mengekspresikan data ke bentuk yang lain, untuk menunjukan perbedaan dan persamaan antar pola. Tujuan dari proses ini yaitu untuk mereduksi dimensi yang besar dari ruang data *(observed variables)* menjadi dimensi yang lebih kecil dari ruang fitur *(independent variables),* yang dibutuhkan untuk menggambarkan data lebih sederhana. PCA ditemukan oleh Karl Pearson pada tahun 1901 dan telah mengalami perkembangan hingga 1963 oleh Karhunen – Louve. Metode ini digunakan untuk keperluan ekstraksi fitur pada citra, dimana pada dimensi citra jauh lebih besar dari jumlah data sampel yang digunakan. Sebagai contoh sebuah citra mempunyai tinggi (h) dan lebar (w), dengan dimensi citra (n), n = h x w dari kombinasi linier h dan w maka dimensi citra ini merupakan dimensi yang tinggi. Dengan metode PCA ini dimensi yang tinggi ini akan direduksi menjadi dimensi yang rendah, jumlah dari dimensi ini tergantung dari data pelatihan dan jumlah sampel pada masing – masing data pelatihan. Jika jumlah data pelatihan (k) dan masing – masing data model (s) maka jumlah sampel total (m) dengan persamaan m = k x s. Berikut ilustrasi dapat dilihat pada gambar di bawah ini :



Gambar 2.11 Perubahan Dimensi PCA Dari 4x5 Menjadi 1x20

1. Algoritma *Eigenfaces*

Eigenfaces sendiri berasal dari bahasa jerman yaitu *eigenwert* dimana kata *eigen* sendiri berarti karakteristik dan *wert* sendiri yaitu nilai. *Eigenfaces* sendiri adalah suatu algoritma yang berdasarkan *Principle Component Analysis* (PCA). cara kerja dari algoritma ini yaitu dengan menghitung *eigenvector* dimana nilai *eigenvector* ini adalah sebuah karakteristik sehingga dinyatakan *eigenfaces* dari suatu citra yang sudah direpresentasikan kedalam sebuah matrik yang berukuran besar.

Algoritma dari *eigenfaces* ini secara garis besarnya yaitu dengan citra matrik (T) direpresentasikan ke dalam sebuah himpunan matrik(T1,T2.....Tm) kemudian cari rata – rata (Ψ) dari himpunan matrik tadi setelah didapat nilai (Ψ) lalu gunakan untuk mencari nilai *eigenvector* (ν) dan *eigenvalue* (λ) dari himpunan matrik. Kemudian gunakan nilai *eigenvector* untuk mendapatkan nilai *eigenfaces* dari suatu citra. Jika ada sebuah citra yang akan dikenali atau diuji maka prosesnya sama pada citra yang akan dikenali (T_{uji}) untuk mencari *eigenvector* (ν) dan *eigenvalue* (λ) kemudian cari nilai *eigenfacesnya.* Setelah itu barulah citra yang akan diuji tersebut memasuki tahapan pengenalan dengan metode berdasarkan jarak terkecil yaitu *euclidean distance*[16]. Untuk tahapan lengkapnya sebagai berikut :

1. Menyiapkan data citra pada sebuah himpunan S yang berisi data pelatihan citra (T_1,T_2.....T_m). berikut persamaanya :

   S = (T_1,T_2.....T_m)
2. Selanjutnya yaitu cari rata – rata (Ψ) dari citra pelatihan .berikut persamaanya :

   $$(\Psi) = \frac{1}{M}\sum_{n=1}^{M} \quad (2.9)$$
3. Kemudian cari selisih (Φ) dari masing Tm dengan citra rata – rata (Ψ)

$$\Phi_i = T_i - \Psi \quad (2.10)$$

4. Langkah selanjutnya mencari matrik kovarian (C)

$$C = \frac{1}{M} \sum_{n=1}^{M} \quad \Phi_n \Phi^T_n = AA^T \text{ dimana } L = A^TA \text{ dan } L = \Phi^T_m \Phi_n \quad (2.11)$$

5. Kemudian hitung nilai *eigenvector* dan *eigenvalue* dari matrik kovarian

$$C \times v_i = \lambda_i \times v_i \quad (2.12)$$

6. Setelah *eigenvector* didapat maka cari nilai *eigenfaces*

$$\mu_i = \sum_{k=1}^{M} \quad v_{ik} \Phi_{k \text{ dimana I} = 1....M} \quad (2.13)$$

7. berikutnya yaitu tahapan mengenali sebuah citra yang dinyatakan mirip atau tidak. Dimana citra uji akan dikenali dengan cara mencari nilai eigenfaces.

$$\mu_{uji} = v \times (T_{new} - \Psi) \text{ dimana } \Omega = \mu_1,... \mu_m \quad (2.14)$$

8. tahap terakhir dengan mencocokan nilai terkecil atau terpendek dari nilai eigenfaces citra latih dan citra uji dengan metode *euclidean distance*.

$$\varepsilon_k = \| \Omega - \Omega_k \| \quad (2.15)$$

2.2.5 *Object Oriented Analysis and Design*

Analisis dan Desain Berorientasi Objek (*Object Oriented Analysis and Design*) adalah 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. Pengertian "berorientasi objek" berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya.

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memerika requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem. Berikut pemaparannya yaitu :

2.2.5.1 Pemograman Berorientasi Objek

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 berkorespondensi dengan objek-objek dunia nyata.

OOP (*Object Oriented Programming*) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dikemas ke dalam kelas-kelas atau objek-objek. Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan. Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek di dunia nyata. Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dimiliki oleh objek akan dienkapsulasi (dibungkus) menjadi satu kesatuan. Sistem berorientasi objek merupakan sebuah sistem yang kompunennya dibungkus (dienkapsulasi) menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat, dan kompunen lainnya yang dapat berinteraksi satu sama lain. Terdapat beberapa konsep utama pada metodologi berorientasi objek, diantaranya :

1. Kelas (*class*), kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dari himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (*atribut*), kelakuan (operasi/metode), hubungan (*relationship*), dan arti. Suatu kelas dapat diturunkan dari kelas yang lain, dimana atribut dari kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*object*), abstraksi dari sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Abstraksi (*abstraction*), prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi suatu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

4. Enkapsulasi (*encapsulation*), pembungkusan atribut data dan layanan (operasi-operasi) yang dimiliki objek untuk menyembunyikan implementasi dari objek sehingga objek lain tidak mengetahui cara kerjanya.

5. Pewarisan (*inheritance*), mekanisme yang memungkinkan suatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

6. Polimorfisme (*polymorphism*), kemampuan seuatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

2.2.5.2 OOA (*Object Oriented Analysis*)

OOA mempelajari permasalahan dengan menspesifikasikannya atau mengobservasi permasalahn tersebut dengan menggunakan metode berorientasi objek. Biasanya analisa sistem dimulai dengan adanya dokumen permintaan *(requirement)* yang diperoleh dari semua pihak yang berkepentingan. Misal: *clien,developer*, pakar, dan lain-lain. Dokumen permintaan memiliki 2 fungsi yaitu : memformulasikan kebutuhan klien dan membuat suatu daftar tugas. Analisis berorientasi obyek (OOA) melihat pada domain masalah, dengan tujuan untuk memproduksi sebuah model konseptual informasi yang ada di daerah yang sedang dianalisis. Model analisis tidak mempertimbangkan kendala-kendala pelaksanaan apapun yang mungkin ada, seperti konkurensi, distribusi, ketekunan, atau bagaimana sistemharus dibangun. Kendala pelaksanaan ditangani selama desain berorientasi objek (OOD). Sumber-sumber untuk analisis dapat persyaratan tertulis pernyataan, dokumen visi yang formal, wawancara dengan *stakeholder* atau pihak yang berkepentingan lainnya. Sebuah sistem dapat dibagi menjadi beberapa domain, yang mewakili bisnis yang berbeda, teknologi, atau bidang yang diminati, masing-masing dianalisis secara terpisah. Hasil analisis berorientasi objek adalah deskripsi dari apa sistem secara fungsional diperlukan untuk melakukan, dalam bentuk sebuah model konseptual. Itu biasanya akan disajikan sebagai seperangkat menggunakan kasus, satu atau lebih UML diagram kelas, dan sejumlah diagram interaksi. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat

persyaratan yang ditentukan pelanggan. UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software. *Unified Model Language* (UML) adalah bahasa universal untuk :

1. Memvisualisasikan grafis model yang tepat.
2. Menetapkan model yang tepat, lengkap, dan tidak ambigu untuk mengampil semua keputusan penting dalam analisis, desain dan implementasi.
3. Membangun model yang dapat dihubungkan langsung dengan bahasa pemrograman.
4. Mendokumentasikan semua informasi yang dikumpulkan oleh tim sehingga memungkinkan untuk berbagi informasi.

2.2.5.3 OOD (*Object Oriented Design*)

OOD mengubah model konseptual yang dihasilkan dalam analisis berorientasi objek memperhitungkan kendala yang dipaksakan oleh arsitektur yang dipilih dan setiap non-fungsional – teknologi atau lingkungan – kendala, seperti transaksi throughput, response time, run – waktu platform, lingkungan pengembangan, atau bahasa pemrograman. Jenis-jenis diagram UML diantaranya :

1. ***Use Case Diagram***

Use case diagram menggambarkan kebutuhan sistem dari sudut pandang *user*. Digunakan untuk menggambarkan hubungan antara internal sistem dan eksternal sistem atau hubungan antara *use case* dan *actor*.

a. *Actor*

Actor adalah sesuatu *(entitas)* yang berhubungan dengan sistem dan berpartisipasi dalam *use case*. *Actor* menggambarkan orang, sistem atau entitas eksternal yang secara

khusus membangkitkan sistem dengan input atau masukan kejadian-kejadian, atau menerima sesuatu dari sistem. *Actor* dilukiskan dengan peran yang mereka mainkan dalam use case, seperti Staff, Kurir dan lain-lain. Dalam *use case* diagram terdapat satu aktor pemulai atau *initiator actor* yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau *participating actor*. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut.

b. *Use case*

Use case yang dibuat berdasar keperluan aktor merupakan gambaran dari "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari interaksinya dengan aktor.

c. *Relationship*

Relasi *(relationship)* digambarkan sebagai bentuk garis antara dua simbol dalam *use case diagram*. Relasi antara *actor* dan *use case* disebut juga dengan asosiasi *(association*). Asosiasi ini digunakan untuk menggambarkan bagaimana hubungan antara keduanya. Relasi-relasi yang terjadi pada *use case* diagram bisa antara *actor* dengan *use case atau use case dengan use case*. Relasi antara *use case* dengan *use case* :

1. *Include* yaitu pemanggilan *use case* oleh *use case* lain atau untuk menggambarkan suatu *use case* termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<*include*>>.
2. *Extend* yaitu digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak kontrol form dan mendeklarasikan ekstension pada *use case* utama. Atau dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis berpanah dengan tulisan <<*extend*>>.
3. *Generalization/Inheritance* yaitu dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan *base-*

parent use case. Digambarkan dengan garis berpanah tertutup dari *base use case* ke *parent use case*.

2. *Activity Diagram*

Diagram aktivitas menggambarkan proses bisnis dan urutan aktivitas-aktivitas yang mendukung penggambaran tindakan sistem baik yang bersifat kondisional maupun paralel. Tindakan kondisional dilukiskan dengan cabang *(branch)* dan penyatuan *(merge).* Sebuah *branch* memiliki sebuah *transition* masuk atau yang disebut dengan *incoming transition* dan beberapa *transition* keluar atau yang disebut dengan *outgoing transition* dari *branch* yang berupa keputusan-keputusan. Hanya satu dari *outgoing transition* yang dapat diambil, maka keputusan-keputusan tersebut harus bersifat *mutually exclusive*. *Else* digunakan sebagai keterangan singkat yang menunjukkan bahwa *transition* "*else*" tersebut harus digunakan jika semua keputusan yang ada pada *branch* salah.

a. *Initial*, merupakan titik awal untuk memulai aktivitas.
b. *Final*, merupakan titik akhir untuk mengakhiri aktivitas.
c. *Activity*, menandakan sebuah aktivitas.
d. *Decision*, merupakan pilihan untuk mengambil keputusan/kondisi if-else.
e. *Fork*/*Join*, digunakan untuk menunjukan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan menjadi satu.

3. *Class Diagram*

Class diagram merupakan bagian yang paling penting dalam analisa dan perancangan berorientasi objek. Dalam UML diagram kelas digunakan untuk memodelkan *static structure* dari sistem informasi. Kelas merupakan himpunan dari objek yang sejenis yang mempunyai atribut dan perilaku *(behaviors/method)* yang sama. Atribut adalah sebuah nilai data karakteristik yang dimiliki oleh obyek sebuah kelas sedangkan *method* adalah perilaku atau operasi yang dikenakan oleh suatu kelas. Pada gambar kelas terdapat tiga bagiannya diantaranya :

a. *Class*

Class adalah blok - blok pembangun pada pemrograman berorientasi objek. Sebuah class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan *property/atribut class*. Bagian akhir mendefinisikan *method-method* dari sebuah *class*.

b. *Assosiation*

Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 *class*, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum *Composition multiplisitas* pada sebuah relationship seperti *One-to-one, one-to-many, many-to-many*.

1. *Dependency*

Kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.

2. *Aggregation*

Aggregation mengindikasikan keseluruhan bagian relationship dan biasanya disebut sebagai relasi "mempunyai sebuah" atau "bagian dari". Sebuah aggregation digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi/tidak solid.

3. *Generalization*

Sebuah relasi *generalization* sepadan dengan sebuah relasi *inheritance* pada konsep berorientasi objek. Sebuah *generalization* dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "parent"- nya/induknya.

4. ***Sequence Diagram***

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar termasuk pengguna, *display*, dan sebagainya berupa *message* yang disusun dalam suatu urutan waktu. Secara khusus, diagram ini berasosiasi dengan *use case*. *Sequence diagram*

menggambarkan *behavior internal* sebuah sistem. Dan lebih menekankan pada penyampaian *message* dengan parameter waktu.

a. *Object*

Object atau biasa juga disebut partisipan merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama objek didalamnya yang diawali dengan sebuah titik koma.

b. *Actor*

Actor juga dapat berkomunikasi dengan object, maka actor juga dapat diurutkan sebagai kolom.

c. *Lifeline*

Lifeline mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk Lifeline adalah garis putus-putus vertikal yang ditarik dari sebuah objek.

d. *Activation*

Activation dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. *Activation* mengindikasikan sebuah objek yang akan melakukan sebuah aksi.

e. *Message*

Message digambarkan dengan anak panah *horizontal* antara *Activation*. *Message* mengindikasikan komunikasi antara *object-object*.

2.2.6 Perangkat Keras

Perangkat keras yaitu alat – alat yang digunakan dalam penelitian untuk menunjang proses pembangunan sebuah sistem yang akan dibangun maupun dikembangkan.

1. *Webcam* (*Web* Kamera)

Webcam adalah sebuah kamera video digital kecil yang dihubungkan ke komputer biasanya melalui *port USB* atau pun *port com*. pada umumnya *Webcam* tidak membutuhkan kaset atau tempat penyimpanan data, data hasil perekaman yang didapat langsung ditransfer ke komputer. Istilah "*webcam*" mengarah pada jenis kamera yang digunakan untuk kebutuhan layanan berbasis web. Defenisi lain tentang *Webcam* adalah sebuah periferal berupa kamera sebagai pengambil citra/gambar dan mikropon (optional) sebagai pengambil

suara/audio yang dikendalikan oleh sebuah komputer atau oleh jaringan computer seperti gambar Gambar 2.12 dibawah ini :

Gambar 2.12 *Webcame*

Webcam bekerja seperti halnya sebuah kamera digital hanya saja *webcam* ini di desain untuk computer jadi tidak bisa dibawa ke mana-mana dan jauh lebih simple di banding kamera-kamera pada umumnya. Sebuah *web camera* biasanya dilengkapi dengan *software*, *software* ini mengambil gambar-gambar dari kamera digital secara terus menerus ataupun dalam interval waktu tertentu dan menyiarkannya melalui koneksi *internet*. Ada beberapa metode penyiaran, metode yang paling umum adalah *software* mengubah gambar ke dalam bentuk file PNG.

2.2.7 Perangkat Lunak

Perangkat lunak yaitu bagian yang bersangkutan dengan penelitian dalam membangun sistem maupun pengembangan sistem dalam penelitian yang sedang dilakukan.

1. *Eclipse*

Eclipse adalah sebuah IDE(*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari *Eclipse*:

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

b. *Mulit-language: Eclipse* dikembangkan dengan bahasa pemrograman *Java*, akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol,Python, Perl,PHP, dan lain sebagainya.

c. *Multi-role:* Selain sebagai IDE untuk pengembangan aplikasi, *Eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan *web*, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *Eclipse* yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2. *MySQL*

MySQL adalah sebuah perangkat lunak sistem manajemen basis data *SQL* (bahasa Inggris: *database management system*) atau *DBMS* yang multialur, multipengguna, dengan sekitar 6 juta instalasi di seluruh dunia. *MySQL AB* membuat *MySQL* tersedia sebagai perangkat lunak gratis di bawah lisensi *GNU General Public License* (*GPL*), tetapi mereka juga menjual di bawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan penggunaan *GPL*. Tidak sama dengan proyek-proyek seperti *Apache*, di mana perangkat lunak dikembangkan oleh komunitas umum,

dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, *MySQL* dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia *MySQL AB*, di mana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan *MySQL* AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL memiliki beberapa keistimewaan, antara lain :

a. Portabilitas *MySQL* dapat berjalan stabil pada berbagai sistem operasi seperti *Windows*, *Linux*, *FreeBSD*, *Mac Os X Server*, *Solaris*, *Amiga*, dan masih banyak lagi.

b. Perangkat lunak sumber terbuka *MySQL* didistribusikan sebagai perangkat lunak sumber terbuka, di bawah lisensi *GPL* sehingga dapat digunakan secara gratis.

c. *Multi-user MySQL* dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

d. *Performance tuning MySQL* memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak *SQL* per satuan waktu.

1. Ragam tipe data *MySQL* memiliki ragam tipe data yang sangat kaya, seperti *signed* / *unsigned integer, float, double, char, text, date, timestamp,* dan lain-lain.

2. Perintah dan Fungs**i** *MySQL* memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).

3. Keamanan *MySQL* memiliki beberapa lapisan keamanan seperti *level subnetmask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.

4. Skalabilitas dan Pembatasan *MySQL* mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*records*) lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

5. Konektivitas *MySQL* dapat melakukan koneksi dengan klien menggunakan protokol *TCP/IP*, *Unix soket* (*UNIX*), atau *Named Pipes* (*NT*).

6. Lokalisasi *MySQL* dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.

7. Antar Muka *MySQL* memiliki antar muka (*interfaces*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi *API* (*Application Programming Interface*).

8. Klien dan Peralatan *MySQL* dilengkapi dengan berbagai peralatan (*tool*) yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk online.

9. *Struktur table MySQL* memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE*, dibandingkan basis data lainnya semacam *PostgreSQL* ataupun *Oracle*. Berikut Fitur serta kapabilitas yang dimiliki oleh *MySQL*:

1. Unjuk kerja yang tinggi dalam memproses *query* sederhana, dalam arti dapat memproses lebih banyak *SQL* per satuan waktu.
2. Memiliki lebih banyak tipe data seperti : *signed/unsigned integer* yang memiliki panjang data sebesar 1,2,3,4 dan 8 *byte*, *FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET* dan tipe *ENUM*.
3. Mendukung *field* yang dijadikan *Index*, dengan maksimal 32 *index* dalam satu tabel. *.
4. *MYSQL* memiliki beberapa lapisan keamanan, seperti *subnetmask*, nama *host*, dan izin akses *user* dengan sistem perijinan yang mendetail serta sandi/*password* terenkripsi.
5. Konektivitas *MySQL* dapat melakukan koneksi dengan klien menggunakan protokol *TCP/IP ,Unix soket (UNIX),*atau *Named Pipes(NT).*
6. *Multi-user MySQL* dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
7. *Command and function, MySQL* memiliki fungsi dan operator secara penuh yang mendukung perintah *select* dan *where* dalam *query*.
8. *Structure Table, MySQL* memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE* dibandingkan *DBMS* lainnya.
9. Mendukung penuh terhadap kalimat *SQL GROUP BY* dan *ORDER BY*. Mendukung terhadap fungsi penuh (*COUNT(), DISTINCT() AVG(), STD(), SUM(), MAX() dan MIN()).*

3. *Xampp*

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache HTTP* Server, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan *Perl*. Nama *XAMPP* merupakan singkatan dari X (empat sistem operasi apapun), *Apache*, *MySQL*, *PHPdan Perl*. Program ini tersedia dalam *GNU General Public License* dan bebas, merupakan *web* server yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis. Untuk mendapatkanya dapat mendownload langsung dari web resminya.

4. *Java*

Java yaitu bahasa pemograman yang dapat dijalankan diberbagai *flatform* dan *mobile*. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin *Virtual Java* (*JVM*). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (general purpose), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi.

5. *OpenCV*

OpenCV (*Open Computer Vision)* adalah sebuah API *(Application Programming Interface)* yang dirilis dibawah lisensi *BSD* yang dapat dipakai oleh siapa saja karena bersifat *open source* diperuntukan untuk komersial dan akademis. *OpenCV* sendiri dirancang untuk

efisiensi komputasi yang berfokus pada aplikasi secara *Real – time*. Dapat ditulis dengan berbagai bahasa pemograman seperti *C / C++* , *Phyton*, *Java* dan *Php* yang telah dioptimalkan. Pada *OpenCV* mampu dijalankan pada berbagai jenis *Flatform* seperti *Windows*, *Linux, Android* dan *Mac OS*. Pada perpustakaan dapat memanfaatkan *Multi – core* yang diaktifkan dengan *OpenCL* dan *CUDA* yang sedang dikembangkan pada saat ini. *OpenCV* sendiri ditulis secara asli pada bahasa pemograman *C++* dan mempunyai template antarmuka yang bekerja dengan mulus dengan kontainer *STL*. *OpenCV* dibangun untuk media pembelajaran pada bidang ilmu visi komputer yang menyediakan infrastruktur umum untuk mempercepat pada penggunaan persepsi mesin dalam produk komersial, memudahkan bisnis dalam memanfaatkan dan memodifikasi kode.

Pada *OpenCV* sudah ada 2500 algoritma yang sudah dioptimalkan yang mencangkup satu set lengkap visi komputer klasik dan algoritma pembelajaran pada mesin. Algoritma ini dapat digunakan untuk *Face Recognition, Face Detection, Face atau Object Tracking* dan lain – lain.

6. *JavaCV*

JavaCV adalah *liberary* dari *OpenCV* yang memungkinkan mengakses perpustakaan –perpustakaan langsung dari *Java Virtual Machine*(*JVM*) dan pada platform *Android*, dengan *javaCV* fungsi – fungsi dari *OpenCV* dapat dipanggil melalui bahasa pemograman yang compatible dengan .NET seperti bahasa C dan C++ sehingga lebih aman untuk digunakan layaknya *OpenCV*.