BAB II
LANDASAN TEORI

2.1 Kecerdasan Buatan

Artificial Intelligent (AI) atau kecerdasan buatan merupakan cabang dari ilmu komputer yang konsern dengan pengautomatisasi tingkah laku cerdas. Pernyataan tersebut juga bisa dijadikan definisi dari AI. Definisi ini menunjukan bahwa AI adalah bagian dari komputer sehingga didasarkan pada *sound theoretichal* (teori suara) dan prinsip-prinsip aplikasi dari bidangnya. Prinsip-prinsip ini meliputi struktur data yang digunakan dalam representasi pengetahuan, algoritma yang digunakan untuk mengaplikasikan pengetahuan tersebut, serta bahasa dan teknik pemrograman yang digunakan dalam mengimplementasikannya.

Namun, definisi di atas belum memadai karena pada kenyataan intilah kecerdasan itu sendiri tidak didefinisikan atau dipahami dengan sangat baik. Kerap kali kita mengetahui tingkah laku cerdas saat melihatnya sendiri. Akan tetapi, keraguan akan muncul ketika orang lain menyatakan kecerdasan dengan cara khusus menurut cara mereka yang cukup membantu dalam evaluasi kecerdasan program komputer. Dengan demikian, belum ada suatu definisi yang cocok untuk *Artificial Intelligent* (AI) itu sendiri.

Definisi AI ini menjadi masalah tersendiri yang harus dijawab: apakah kecerdasan tersebut merupakan suatu "*Single Faculty*", atau apakah hanya sebuah nama untuk kumpulan perbedaan dan tidak ada hubungannya dengan kemampuan? Apa perlunya mempelajari AI? Apakah hubungannya dengan proses belajar, kreativitas dan intuisi? Dapatkah kecerdasan disimpulkan dari tingkah yang dapat diobservasi ataukah membutuhkan pembuktian dari suatu mekanisme internal yang

khusus? Bagaimana pengetahuan direpresentasikan dalam jaringan saraf dari kehidupan? Semua pertanyaan tersebut merupakan pertanyaan yang mungkin belum terjawab dan tentunya akan membantu kita untuk terus berusaha menggeluti, mempelajari dan mencari metode yang dapat menjadi solusi bagi pertanyaan ini, dan nantinya dapat membentuk suatu intisari AI modern. AI menawarkan media dan uji teori dari kecerdasan. Teori-teori tersebut dapat dinyatakan dalam bahasa program komputer, diuji dan dieksekusi pada komputer.

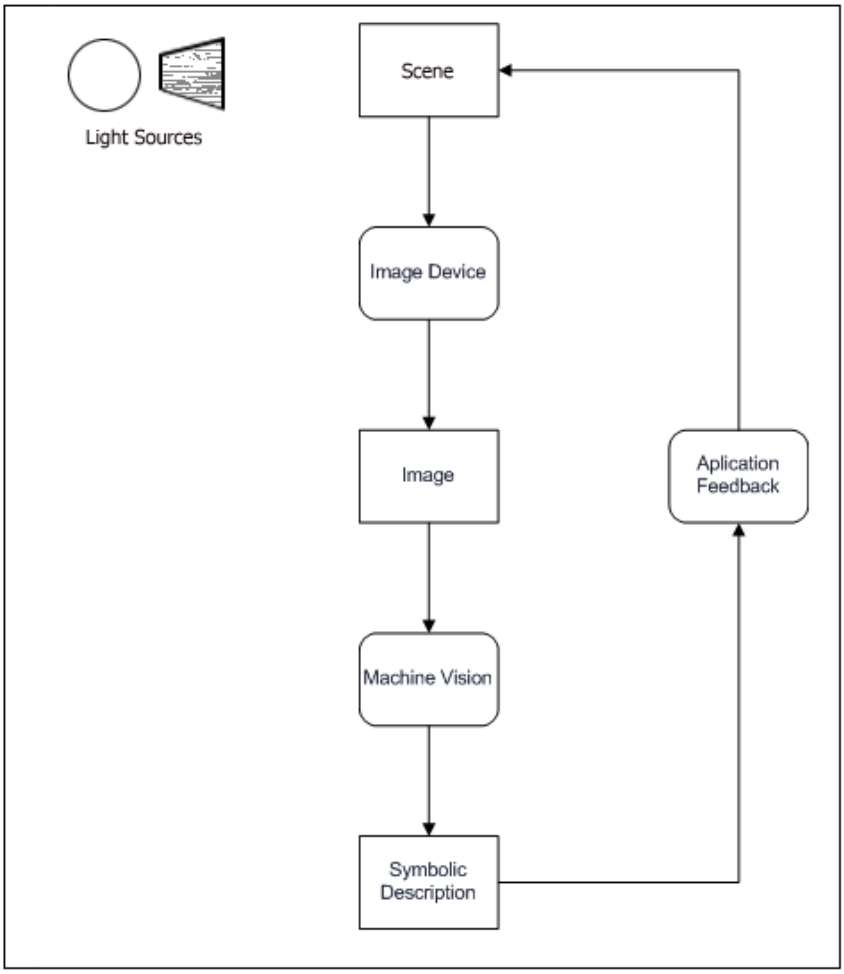
2.2 Komputer Vision

Titik, garis, kurva dan berbagai bentuk bidang serta semua kombinasinya merupakan isi suatu keadaan yang disebut alam nyata. Alam nyata adalah istilah yang diambil untuk mewakili ruang tempat hidup semua makhluk hidup dan benda mati beserta semua keindahan yang terdapat di dalamnya, yang memberikan berbagai macam kombinasi gerak, kombinasi warna atau kombinasi antara keduanya sehingga akan sangat rumit untuk menyatakan dan memvisualisasikan semua kombinasi tersebut dengan menggunakan komputer.

Disisi lain orang-orang juga mengembangkan bagaimana suatu mesin pintar dapat memahami dan mengerti semua keadaan seperti telah dipaparkan di atas, bahkan juga dapat berkomunikasi dengan semuanya. Inilah konsep yang mendasari komputer vision.

2.1.1 Elemen-Elemen Komputer Vision

Gambar 2.6 di bawah ini adalah struktur yang mendasari elemen-elemen suatu mesin vision.



Gambar 2. 1 Struktur Komputer Vision

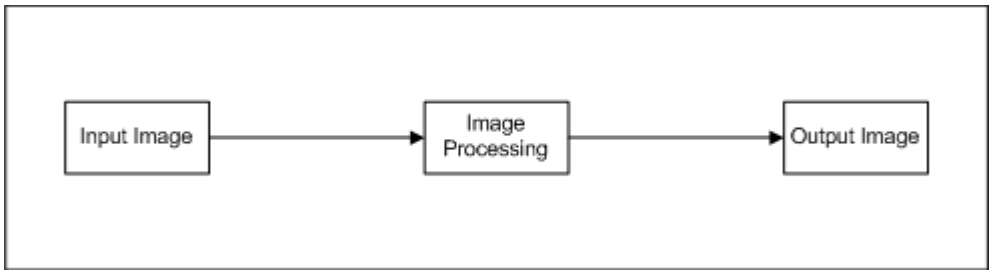
a. *Light sources*, merupakan sumber cahaya yang digunakan sebagai sumber untuk aplikasi seperti layar laser, sistem robotika dan sebagainya.
b. *Scence*, merupakan kumpulan objek.
c. *Image Device*, merupakan alat yang digunakan untuk mengubah gambar menjadi suatu yang dimengerti oleh mesin.
d. *Image*, merupakan gambar dari suatu objek yang merupakan representasi dari keadaan sesungguhnya.

e. *Machine vision*, merupakan mesin yang menginterpretasikan gambar yang berkenaan dengan cirri-ciri pola maupun objek yang dapat ditelusuri oleh sistem.
f. *Symbolic description*, merupakan sistem yang dapat digunakan untuk menganalogikan struktur kerja sistem ke simbol-simbol tertentu yang dapat mimengerti oleh sistem.
g. *Aplication feedback*, merupakan suatu keadaan yang dapat memberikan respon untuk menerima gambar dari suatu sistem penglihatan.

Dari paparan di atas ada tiga elemen yang mendasari suatu sistem vision, yaitu *Image Processing, Pattern Classification* dan *Scene Analysis*.

2.1.1.1 *Image Processing*

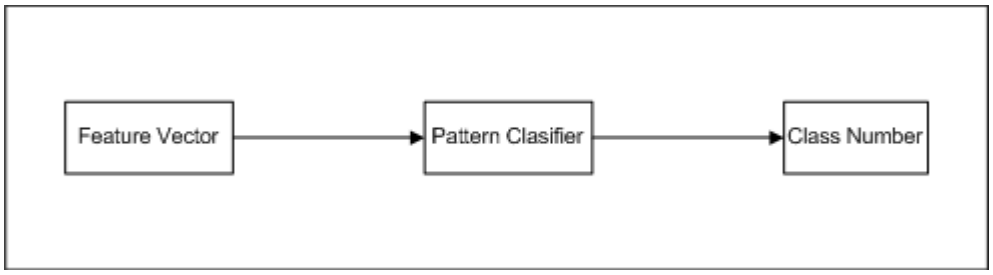
Bagian ini berfungsi mengubah atau mengkonversi gambar eksternal menjadi suatu representasi yang dibutuhkan. Berikut adalah bagian dari *image processing*:



Gambar 2. 2 Fase *Image Processing*

2.1.1.2 Klasifikasi Pola

Ide klasifikasi pola (*pattern classification*) ini adalah bagaimana suatu mesin pintar (dalam hal ini adalah komputer) dapat mengetahui berbagai macam dan bentuk pola, seperti garis, kurva, bayangan dan berbagai pola lainnya. Artinya, Jika mesin tersebut diberi suatu input berupa pola tertentu maka mesin dapat mengerti pola yang diberikan itu. Berikut adalah bagian dari suatu proses klasifikasi pola:



Gambar 2. 3 Fase Klasifikasi Pola

2.1.1.3 *Scene Analysis*

Seperti telah dijelaskan sebelumnya, yang merupakan suatu permasalahan yang tergolong rumit dalam komputer vision adalah bagaimana memperoleh informasi dari suatu paparan (baik itu berupa gambar atau pola-pola tertentu)

2.3 Pengenalan Pola

Pengenalan pola adalah salah satu cabang ilmu pengetahuan dari bidang kecerdasan buatan. Ada beberapa definisi yang berbeda tentang pengenalan pola, diantaranya :

1. Penentuan suatu objek fisik atau kejadian ke dalam salah satu atau beberapa kategori. (Duda dan Heart).
2. Ilmu pengetahuan yang menitik beratkan pada deskripsi dan klasifikasi (pengenalan) dari suatu pengukuran (Schalkoff).
3. Proses untuk memberikan nama ω untuk pengamatan x (Schürman).

Berdasarkan beberapa definisi diatas, pengenalan pola bisa didefinisikan sebagai cabang kecerdasan buatan yang menitik beratkan pada metode pengklasifikasian objek ke dalam kelas-kelas tertentu untuk menyelesaikan masalah tertentu.

2.3.1 Komponen Sistem Pengenalan Pola

Sistem pengenalan pola dasar terdiri dari :

1. Sensor

   Sensor digunakan untuk menangkap objek yang ciri serta fiturnya akan diekstrasi.
2. Mekanisme pre-*processing*

   Mekanisme pengolahan objek yang ditangkap oleh sensor biasanya digunakan untuk mengurangi kompleksitas ciri yang akan dipakai untuk proses klasifikasi.
3. Mekanisme pencari fitur (manual/otomatis)

   Bagian ini digunakan untuk mengekstrasi ciri yang telah melalui tahapan pre-*processing* untuk memisahkannya dari fitur-fitur pada objek yang tidak diperlukan dalam proses klasifikasi.
4. Algoritma pemilah

   Pada tahapan ini proses klasifikasi dilakukan dengan menggunakan algoritma klasifikasi tertentu. Hasil dari tahapan ini adalah klasifikasi dari objek yang ditangkap kedalam kriteria-kriteria yang telah ditentukan.

2.3.2 Fitur

Fitur adalah segala jenis aspek perbedaan, kualitas atau karakteristik. Fitur bisa berwujud simbolik (misalnya warna) atau numerik (misalnya tinggi). Kombinasi dari *d*-buah fitur dinyatakan sebagai fektor kolom dimensi-*d* dan disebut vector fitur. Ruang dimensi-*d* yang dibentuk oleh vector fitur disebut sebagai ruang fitur. Objek dinyatakan sebagai sebuah titik didalam ruang fitur. Penggambaran demikian disebut sebagai diagram hambur (*scatter plot*). Kualitas dari vector fitur dilihat dari kemampuannya membedakan objek yang berasal dari kelas yang berbeda-beda dimana berlaku ketentuan berikut :

1. Objek dalam kelas yang sama harus memiliki nilai *vector fitur* yang sama.

2. Objek dalam kelas yang berbeda harus memiliki nilai *vector fitur* yang berlainan.

2.3.3 Pola

Pola adalah komposit atau gabungan dari fitur yang merupakan sifat dari sebuah objek. Dalam klasifikasi, pola berupa sepasang variable (x, ω), dimana :

1. x adalah sekumpulan pengamatan atau fitur (vector fitur).
2. ω adalah konsep dibalik pengamatan (label).

2.3.4 Pemilah

Tugas dari pemilah adalah merekat ruang fitur ke dalam daerah-daerah yang dilabeli dengan kelas. Garis batas antar daerah keputusan disebut sebagai pembatasan keputusan**.** Pemilahan *vector fitur* x meliputi penentuan daerah keputusan yang sesuai dengan pengelompokan x kedalam kelas ini.

2.3.5 Pendekatan Pengenalan Pola

Aplikasi pengenalan pola dapat dibuat dengan beberapa pendekatan. Ada pendekatan yang menggunakan basis statistikal untuk menghasilkan pola. Pendekan lainnya menggunakan struktur dari pola yang menyediakan informasi fundamental untuk pengenalan pola. Pendekatan yang lain adalah dengan membangun dan melatih suatu arsitektur yang secara akurat mengasosiasikan *input* pola tertentu dengan respon yang diharapkan. Suatu masalah bisa diselesaikan dengan salah satu atau beberapa pendekatan tadi, tinggal dipertimbangkan pendekatan mana yang lebih tepat untuk masalah yang akan dipecahkan.

1. Pendekatan pengenalan pola statistikal (StatPR)

Pendekatan pola statistikal memiliki asumsi suatu basis statistic untuk algoritma klasifikasi. Sekelompok karakteristik pengukuran yang menunjukan fitur diekstrasi dari data *input* dan digunakan untuk menentukan setiap vektor vitur kedalam salah satu kelas c. Fitur diasumsikan dihasilkan secara natural sehingga model yang

bersangkutan adalah suatu *state of nature* atau kelas-kelas probabilitas atau fungsi kepadatan peobabilitas (*probability density function*) yang telah dikondisikan. Dengan demikian kesimpulannya adalah sebagai berikut :

a. Pola dipilah berdasarkan model statidtik dari fitur.
b. Model statistic didefenisikan sebagai sebuah keluarga dari fungsi kerapatan peluang bersyarat kelas Pr(x|ci) – peluang vektor vitur x jika diberikan kelas ci.

2. Pendekatan pengenalan pola statistik (SyntPR)

Untuk pendekatan statistik dapat dijelaskan sebagai berikut :

a. Pola dipilah berdasarkan keserupaan ukuran struktural.
b. ‘Pengetahuan’ direpresentasikan secara *formal grammar* atau deskripsi relasional (graf)
c. SyntPR dipakai tidak hanya untuk pemilahan, tetapi juga untuk deskripsi.
d. Biasanya SyntPR memformulasikan descripsi hierarkis dari pola kompleks yang tersusun dari pola bagian yang lebih sederhana.

3. Pendekatan pengenalan pola Neural (NeuroPR)

Pendekatan ini menggunakan metode jaringan saraf tiruan untuk mengidentifikasi pola. Pendekatan ini dapat dijelaskan sebagai berikut :

a. Pemilahan dilakukan berdasarkan tanggapan suatu neuron jaringan pengolah sinyal (neuron) terhadap stimulus masukan (pola).
b. ‘Pengetahuan’ disimpan dalam sambungan antar neuron dan kekuatan pembobot sinaptik.
c. NeuronPR dapat dilatih, non-algoritmik, strategi *black box.*
d. NeuronPR sangatlah menarik karena dengan jumlah lapisan dan neuron secukupnya, JST dapat membentuk semua jenis daerahkeputusan yang rumit sekalipun.

Jangkauan dari aplikasi-aplikasi sistem pengenalan pola cukup beragam, diantaranya :

1. Voice recognition

Beberapa sistem rahasia menggunakan pengenalan suara sebagai kunci bagi penggunanya.

2. Fingerprint identification

Sistem pengenalan sidik jari telah dipakai secara luas sebagai penggsnti password atau pin untuk mengakses system computer tertentu.

3. Face identification

Badan penegak hukum sedang mengembangkan system untuk mengidentifikasi para buronan dengan melakukan *scanning* pada sejumlah besar data wajah para pelaku yang sudah di-*database*-kan berdasarkan foto pelaku kejahatan tersebut.

4. Handwriting identification

Aplikasi perbankan menggunakan pengenalan tulisan untuk membuktikan pelaku transaksi adalah orang yang benar-benar berhak

5. Optical Character Recognition (OCR)

OCR sekarang digunakan secara luas pada *took retail* untuk menionggkatkan produktivitas pada konter pengecekan barang

6. Robot Vision

Banyak aplikasi robotik menggunakan pengenalan pola untuk mengenali objek tertentu pada lingkungan yang unik.

2.4 Pengenalan Wajah

Secara umum sistem pengenalan citra wajah dibagi menjadi dua jenis, yaitu sistem *feature based* dan sistem *image based.* Pada sistem pertama digunakan fitur yang diekstrasi dari komponen citra wajah (mata, hidung, mulut, dll.) yang kemudian hubungan antara fitur-fitur tersebut dimodelkan secara geometris. Sedangkan sistem kedua menggunakan informasi mentah dari piksel citra yang kemudian direpresentasikan dalam metode tertentu.

Proses pengenalan wajah yang dilakukan oleh komputer tidak semudah dan secepat proses pengenalan yang dilakukan oleh manusia. Manusia dapat dengan mudah mengenali wajah seseorang dengan cepat tanpa rasa harus berpikir. Manusia juga tidak terpengaruk oleh orientasi wajah orang tersebut, misalnya wajah orang tersebut dengan keadaan agak menoleh, merunduk dan menengadah asalkan ada batas-batas yang masih bisa dilihat. Sedangkan komputer selain lamban dalam pengenalan juga kesulitan pada orientasi wajah yang berlainan, pencahayaan, latar belakang yang berbeda, potongan rambut, kumis atau jenggot, berkacamata atau tidak dan sebagainya.

Memang otak manusia lebih memiliki keuntungan dalam mengatasi masalah dimana aturan eksplisit tidak dapat dengan mudah diformulasikan, sedangkan komputer mempunyai keuntungan pada bidang seperti matematika dimana aturan-aturan mudah diformulasikan. Oleh karena itu banyak dilakukan penelitian untuk mencari algoritma-algoritma yang tepat bagi komputer agar dapat mengenali suatu wajah dengan memperhatikan faktor kecepatan dan akurasi.

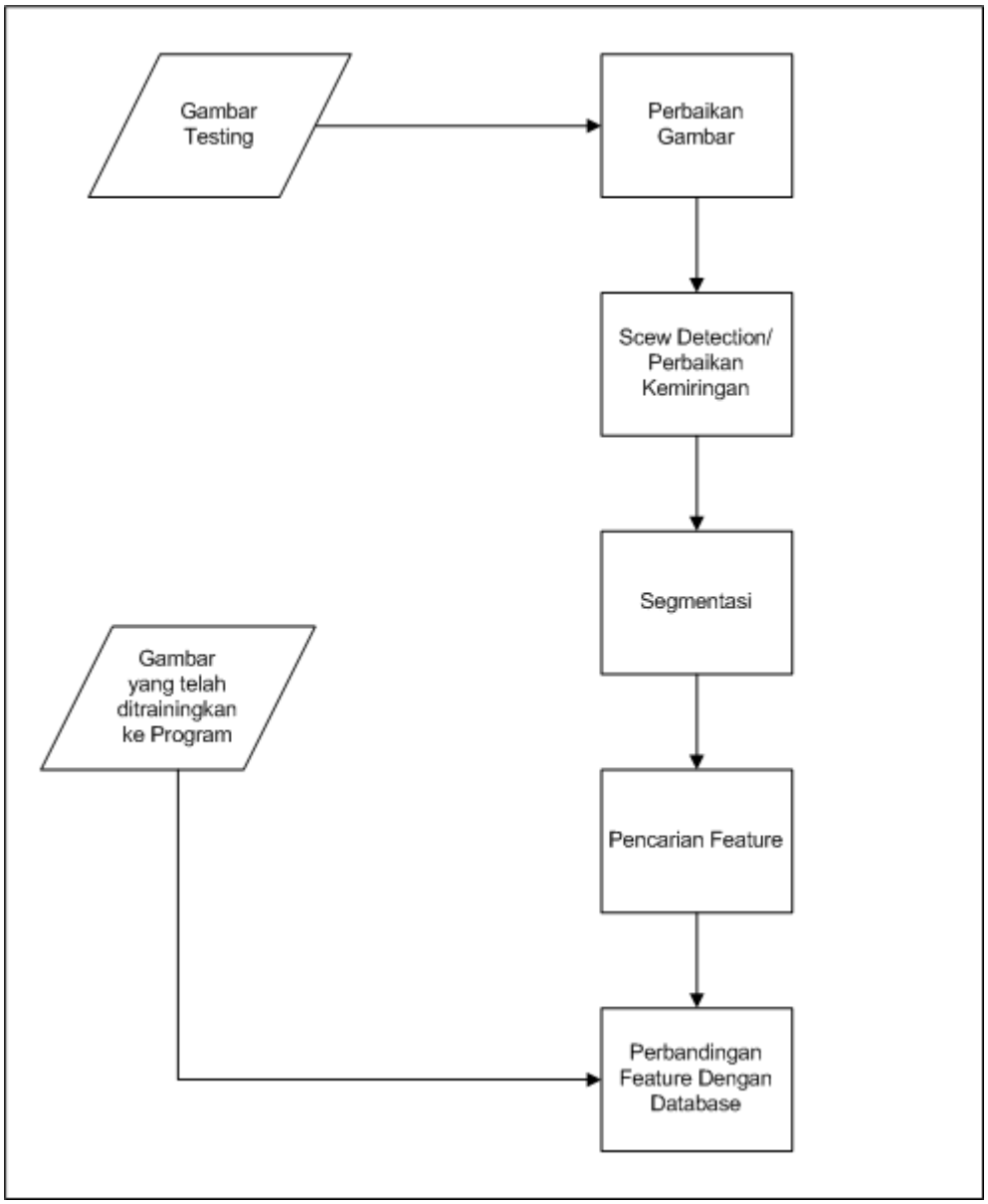
2.4.1 Tahapan Umum Pengenalan Wajah

Algoritma untuk mengenali wajah ada bermacam-macam. Akan tetapi pada umumnya ada tahapan-tahapan umum yang dipakai dalam pengenalan wajah. Tahapan-tahapan umum dari pengenalan wajah.

1. Perbaikan Gambar
2. Segmentasi
3. Pencarian *Feature*
4. *Scew Detection*/Perbaikan Kemiringan
5. Identifikasi

Untuk lebih jelasnya tahapan ini bisa dilihat dalam gambar 2.4. Meskipun pada umumnya pengenalan wajah harus melalui tahapan-tahapan tersebut akan tetapi

dalam kenyataan banyak peneliti yang melompati beberapa tahap, misalkan tahap segmentasi dilompati sehingga wajah yang dimasukkan harus sudah tersegmentasi manual. Hal ini mungkin disebabkan keterbatasan waktu atau biaya bagi peneliti.



Gambar 2.4 Tahapan Utama Pengenalan Wajah

2.4.1.1 Perbaikan Gambar

Input gambar dari hasil *scanning* belum tentu bagus atau kabur, sehingga bukan merupakan *input* yang bagus bagi program. *Image* hasil scanning biasanya mengandung banyak *noise* yang harus dihilangkan. Kemudian harus diterapkan algoritma-algoritma untuk mempertajam gambar bila gambar kabur. Seperti *filtering* yang teorinya ada pada *image processing*. Setelah itu dilakukan pengaturan pencahayaan pada *image*, sehingga diharapkan semua *image* mempunyai pencahayaan yang sama untuk memudahkan proses pengenalan nanti.

2.4.1.2 Segmentasi

Tahap ini dilakukan segmentasi dengan suatu algoritma untuk mengenali gambar mana yang merupaka wajah. Apabila gambar setelah dicek oleh algoritma bukan merupakan wajah, maka proses berhenti dan keluar. Akan tetapi bila wajah tersebut ada bentuk oval dari wajah, maka *image* tersebut akan disegmentasi untuk membuang latar belakangnya sehingga hasilnya adalah wajah oval saja tanpa latar belakang. Yang dimaksud wajah oval yaitu bentuk oval dari wajah, yaitu wajah tanpa rambut, telinga atau leher.

2.1.4.3 Pencarian *Feature*

Ada dua macam *feature* pada wajah, yaitu *holistic feature* dan *partial feature*. Pada *partial feature* pada pengenalan wajah biasanya disebut sebagai *facial feature*. Contoh *facial feature*-nya adalah warna dan bentuk rambut, besar dan letak mata hidung, mulut, telinga dan lain-lain. Sedangkan pada *holistic feature*, setiap *feature*-nya adalah suatu karakteristik dari seluruh wajah. Maksudnya wajah dianggap sebagai suatu kesatuan yang utuh. *Participal Conponent Analysis* adalah teknik yang menggunakan *holistic feature*. Jadi *Participal Conponent Analysis* menganggap suatu wajah adalah suatu kesatuan dan dibandingkan dengan wajah yang lain.

2.1.4.4 Perbaikan Kemiringan

Gambar oval dari wajah seseorang didapatkan dari hasil segmentasi dapat tegak, miring ke kiri atau ke kanan atau bahkan terbalik. Oleh karena itu proses perbaikan kemiringan ini diperlukan, sehingga gambar yang miring atau terbalik bisa diperbaiki agar menjadi gambar yang benar-benar tegak.

Sebenarnya pada gambar wajah, kemiringan bukan hanya sesederhana bila kita mengamati tulisan. Tulisan atau huruf hanya bisa miring ke kiri atau ke kanan. Pada gambar selain miring ke kiri atau ke kanan, ada juga kemiringan pada orientasi wajah. Maksudnya gambar orang tidak selalu menghadap ke depan, namun bisa menoleh beberapa derajat ke kanan atau ke kiri. Maka seharusnya ada algoritma untuk mengatur orientasi wajah tersebut.

Facial Feature biasanya dimanfaatkan untuk perbaikan kemiringan. Misalkan dengan mengamati keadaan lurus atau miringnya dari letak dua mata, maka dapat ditentukan pada sudut rotasi bagi wajah. Letak hidung juga harus diperhatikan agar rotasi wajah yang dihasilkan tidak terbalik, Sehinga mata harus diatas hidung dan bukan sebaliknya. Berdasarkan *feature* ini maka gambar hanya bisa dirotasi kemiringannya sehingga menjadi tegak, tetapi tetap saja tidak ada algoritma untuk memperbaiki orientasi wajah.

2.1.4.5 Identifikasi

Pada tahap akhir akan diambil kesimpulan. Kesimpulan yang diambil bisa berbeda tergantung pada aplikasi yang diinginkan, misalnya kesimpulan yang diambil yaitu untuk menggolongkan individu pada *image* berdasarkan jenis kelaminnya, atau melakukan pengenalan terhadap *image* yang di-*input*-kan atau langsung memberi identitas *image* tersebut.

Setelah komputer tahu tentang *feature* dari *testing image* yang di-*input*-kan maka dapat dilakukan perbandingan dengan *feature-feature image* pelatihan. Tentu saja sebelumnya *image-image* pelatihan juga melalui tahap yang sama dari tahap satu

sampai lima. Setelah dilakukan perbandingan maka dipilih satu wajah *training* yang *feature*-nya paling mirip dengan *feature image testing.* Wajah *training* inilah yang merupakan *output* dari pengenalan wajah, yaitu sebagai wajah pelatihan yang mirip dengan wajah testing.

2.5 Image Processing

Image *grayscale* lebih mudah untuk dianalisa apabila dibandingkan dengan *image* berwarna. *Image* berwarna adalah kombinasi dari tiga warna utama merah, hijau dan biru yang biasa disebut dengan sistem warna RGB. Terdiri dari 24 bit, masing-masing 8 bit untuk merah, hijau dan biru. Akan lebih mudah untuk dianalisa maka *image* terlebih dahulu diubah menjadi *image grayscale* dengan cara memberi nilai yang sama untuk masing-masing nilai 8 bit tersebut.

2.6 *Participal Komponent Analysis* (PCA)

Metode *Paticipal Component Anayisis* dibuat pertama kali oleh para ahli statistik. Metode PCA pertama kali ditemukan pada Karl Person pada tahun 1901, yang memakainya pada konteks biologi. Meskipun pertama kali ditemukan oleh Karl Person, sebenarnya prosedur umum seperti yang telah diketahui sekarang ini ditemukan oleh Harold Hotelling dimana paper yang menjadi pioneer ini muncul pada tahun 1933, yang membahas pada bidang *psychometri*. Kemudian tidak ada perkembangan baru pada teknik ini, dan perkembangannya baru mulai pesat pada tahun 1930 dan awal 1940. Setelah itu perkembangannya berkurang sampai komputer berhasil didesain sehingga diaplikasikan teknik ini pada masalah-masalah yang masuk akal. Pada tahun 1947 teori ini muncul lagi dan cukup independen sebagai teori probabilitas yang ditemukan oleh Karhunen dan kemudian dikembangkan oleh Loeve pada tahun 1963. Sehingga teori ini juga dinakan teori Karhunen-Loeve *transform* pada bidang ilmu komunikasi.

PCA adalah teori yang termasuk dalam bidang *multivariate analysis* secara sederhana dapat dijelaskan sebagai metode yang berhubungan dengan variable dalam

jumlah besar pada satu atau banyak percobaan. Beberapa bidang lain dalam *multivariate analysis* adalah *common factor analysis, multiple regression, multipal diskriminanat analysis of variance* dan *civariance, conjoint analysis, connonical correlation, multi dimensional scalling, correspondence analysis, linear probability model* dan *simultaneous structural equalition modeling.*

Prinsip dasar dari PCA adalah memproyeksikan *image* ke dalam ruang *eigen*-nya. Caranya adalah dengan mencari *eigenvector* yang dimiliki setiap *image* dan memproyeksikannya ke dalam ruang *eigen* yang didapat tersebut. Besarnya dimensi ruang *eigen* bergantung pada jumlah *image* yang dimiliki oleh program *training*.

2.6.1 Memasukkan data tiap *pixel* ke dalam matriks

Misalkan ada *m image* masing masing yang berdimensi 100 x 100 *pixel* = 10.000 *pixel*. Matriks yang merepresentasikan *image-image* tersebut berdimensi 10.000 x m adalah :

$$u=\begin{bmatrix} u_{1,1} & u_{1,2} & \cdot & \cdot & u_{1,m} \\ u_{2,1} & u_{2,2} & \cdot & \cdot & u_{2,m} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ u_{10.000,1} & u_{10.000,2} & \cdot & \cdot & u_{10.000,m} \end{bmatrix}$$

2.6.2 Mencari *Image* Rata – Rata

Image rata-rata adalah rata-rata dari semua *pixel image-image training*.

$$\bar{u} = \frac{1}{m}\sum_{k=1}^{m} u_{1,k}$$

Dari matriks di atas didapat rata-rata total PCA dengan dimensi 10.000 x 1, yaitu:

$$\bar{u} = \begin{bmatrix} \bar{u}_{1,1} \\ \cdot \\ \cdot \\ \bar{u}_{10.000,1} \end{bmatrix}$$

Matriks $\bar{u}$ ini kemudian disimpan agar tidak perlu melakukan proses *training* berulang-ulang

2.6.3 Mencari *Covariance* Matriks

Covariance matriks dicari dengan mengalikan matriks *u* dengan transposenya. Hasilnya adalah matriks berdimensi 10.000 x 10.000 sebagai berikut:

$$C = u \times u^t$$

Selanjutnya dilakukan dekomposisi *eigen* sehingga berlaku rumusan sebagai berikut:

$$C \times V = \lambda \times V$$

λ = Matriks *eigenvalue*

V = Matriks *eigenvector*

Dimana V dan λ adalah matriks berdimensi *n x n* (*n* adalah jumlah *pixel image*) sebagai berikut:

$$V = \begin{bmatrix} V_{1,1} & \cdot & \cdot & \cdot & \cdot & V_{1,n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ V_{n,1} & \cdot & \cdot & \cdot & \cdot & V_{n,n} \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdot & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_n \end{bmatrix}$$

Pencarian *eigenvector* dan *eigenvalue* ini dapat dibantu dengan menggunakan metode *Jacobi*.

2.6.3.1 Transformasi Jacobi

Inti dari metode *Jacobi* adalah mentransformasikan suatu matriks, misalkan matriks **A** menjadi matriks **B** yang mempunyai nilai-nilai *eigen* yang sama dengan matriks **A,** tetapi mempunyai bentuk yang lebih sederhana sehingga semua nilai *eigen* dapat dihitung dengan lebih mudah. Hal ini dapat dilakukan asalkan setelah transformasi didapat suatu matriks orthogonal **R** yang jika dikalikan dengan **A** sesuai dan akan menghasilkan suatu matriks **B** yang mempunyai nilai-nilai eigen yang sama dengan **A**.

Bentuk matriks **B** yang diinginkan adalah matriks diagonal. Jika mengubah bentuk matriks **A** dari:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Menjadi

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ 0 & b_{22} & b_{23} \\ 0 & 0 & b_{33} \end{bmatrix}$$

Atau

$$\mathbf{B} = \begin{bmatrix} ab_{11} & 0 & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

Atau

$$\mathbf{B} = \begin{bmatrix} b_{11} & 0 & 0 \\ 0 & b_{22} & 0 \\ 0 & 0 & b_{33} \end{bmatrix}$$

Maka nilai eigen matriks B akan mudah didapat karena merupakan nilai-nilai elemen diagonalnya. Dengan metode *Jacobi* suatu matriks B yang bentuk matriks

diagonalnya diperoleh dengan menjadikan elemen-elemen bukan nol di atas diagonal menjadi nol satu per satu. Hal ini dapat diperoleh dengan menggunakan matriks rotasi plan **R**(p,q). Matriks rotasi **R**(p,q) mempunyai ukuran yang sama dengan **A** yang pada dasarnya adalah sebuah matriks unitas, kecuali elemen R_{pp}, R_{pq}, R_{qp}, R_{qq} yang ditentukan oleh:

$R_{pp} = \cos\theta$

$R_{pq} = -\sin\theta$

$R_{qp} = \sin\theta$

$R_{qq=} \cos\theta$

Dimana θ adalah suatu nilai yang harus ditentukan. p dan q adalah lajur dan kolom dari elemen pada matriks **A** yang hendak dinolkan. Matriks R adalah sebuah matriks orthogonal. Sebagai Bukti hal ini Jika R_{3x3} maka:

$$\mathbf{R}(1,2) = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Dimana $r_{11} = \cos\theta$, $r_{12} = -\sin\theta$, $r_{21} = \sin\theta$ dan $r_{22} = \cos\theta$. Dapat dibuktikan bahwa $R(1,2)R(1,2)^T = 1$

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos^2\theta + \sin^2\theta & 0 & 0 \\ 0 & \cos^2\theta + \sin^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = 1$$

Karena $\mathbf{R^T} = \mathbf{R^{-1}}$ maka matriks **R** adalah matriks orthogonal dan dengan menggunakan **R** maka matriks **A** dapat ditransformasikan menjadi **B** yang mempunyai nilai yang sama dengan **A.**

$$\mathbf{R^T A R = B}$$

Dengan merancang matriks **R** sedemikian, maka dapat menjadikan elemen (p,q) tertentu pada **B** menjadi nol. Sebagai penjelasan digunakan matriks $\mathbf{A}_{3x3:}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Jika ingin mengubah a_{13} menjadi nol, maka bentuk matriks R(1,3).

$$\mathbf{R}(1,3) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

Kemudian:

$$\mathbf{R}^{\mathbf{T}}(1,3)\ \mathbf{A} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\begin{bmatrix} a_{11}\cos\theta + a_{31}\sin\theta & a_{12}\cos\theta + a_{32}\cos\theta & a_{13}\cos\theta + a_{33}\sin\theta \\ a_{21} & a_{22} & a_{23} \\ -a_{11}\cos\theta + a_{31}\sin\theta & -a_{11}\cos\theta + a_{31}\sin\theta & -a_{11}\cos\theta + a_{31}\sin\theta \end{bmatrix}$$

Selanjutnya:

$$\mathbf{R}^{T}(1,3)\ \mathbf{A}\ \mathbf{R}(1,3) = \mathbf{B} =$$

$$\begin{bmatrix} a_{11}\cos\theta + a_{31}\sin\theta & a_{12}\cos\theta + a_{32}\cos\theta & a_{13}\cos\theta + a_{33}\sin\theta \\ a_{21} & a_{22} & a_{23} \\ -a_{11}\cos\theta + a_{31}\sin\theta & -a_{11}\cos\theta + a_{31}\sin\theta & -a_{11}\cos\theta + a_{31}\sin\theta \end{bmatrix}$$

Lakukan hal yang sama untuk nilai yang hendak dinolkan. Iterasi dilakukan hingga semua elemen non-diagonal menjadi lebih kecil dari suatu kriteria konvergensi. Pada setiap tahap ditentukan elemen non-diagonal yang akan dijadikan nol. Jelas bahwa akan ada banyak pilihan elemen yang akan dinolkan. Pada metode *Jacobi* elemen dengan nilai terbesarlah yang akan dihilangkan.

2.6.4 Mencari *Feature* PCA

Feature adalah komponen–komponen penting dari *image-image training* yang didapatkan dari proses *training*. *Feature* inilah yang nanti akan digunakan untuk mengidentifikasikan *image* yang akan dikenali.

Feature dapat dicari dengan mentransformasikan *image* asal ke dalam ruang *eigen* dengan menggunakan persamaan sebagai berikut

$$f = \sum_{i=1}^{m} (I_i - \bar{u})^T \times V$$

m = Jumlah *image training*
Ii = *Image input*
u = *Image* rata-rata total
V= Matriks *eigenvector*

Dimana *I* = data tiap *pixel* dari *image training* ke-*I*, m adalah jumlah *image training* dan *V* adalah *eigenvector*. f merupakan matriks berukuran 10.000, yaitu:

$$f = \begin{bmatrix} f_{1,1} & f_{1,10.000} \\ \cdot & \cdot \\ \cdot & \cdot \\ f_{m,1} & f_{m,10.000} \end{bmatrix}$$

Matriks f ini juga disimpan agar tidak perlu melakukan proses *training* berulang – ulang.

2.6.5 *Recognition* dengan PCA

Pengenalan suatu *image I* menggunakan PCA merupakan operasi dari 3 buah matriks utama yaitu $\bar{u}$ *V* dan *f* yang sudah tersimpan seperti pada keterangan di atas. *Image* yang akan dikenali disebut *Ix*. Selanjutnya diproyeksikan juga ke dalam bidang multidimensi seperti di atas dengan metode PCA. Selanjutnya dicari *distance* yang paling minimum. *Distance* adalah jarak antara *Ix* dengan masing-masing *feature image training*. Untuk menghitung *distance* ini tidak perlu menghitung sebanyak 10.000 dimensi seperti yang ada pada matriks *f* diatas, tetapi cukup diambil beberapa *feature* penting misalnya 20, 30 dan seterusnya. Berdasarkan jumlah *feature* yang diambil *recognition* presentase dari PCA dapat berubah-ubah. Jumlah *feature* yang optimal akan didapatkan dari hasil eksperimen beberapa kali

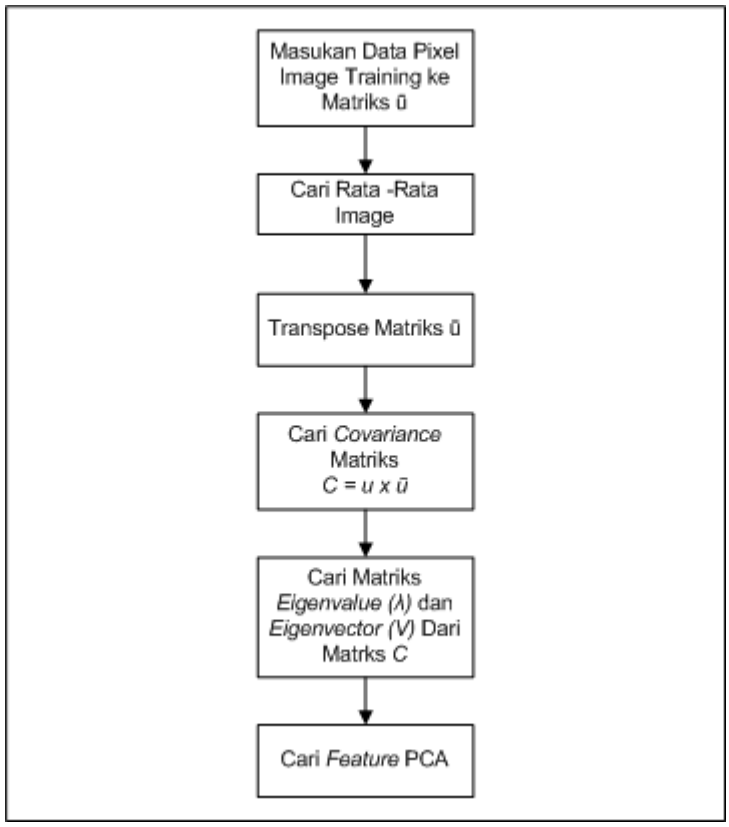
Distance = $|f_x - f_{database}|$

$$= \sqrt{(f_{x1} - f_{database\ 1})^2 + (f_{x1} - f_{database\ 2})^2 + \cdots + (f_{x1} - f_{database\ z})^2}$$

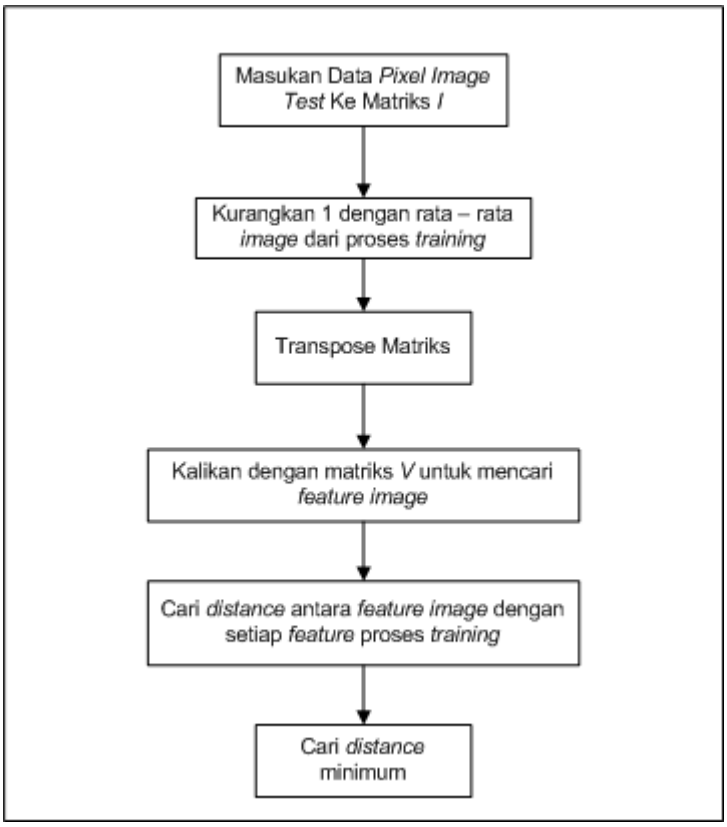
Dimana *z* adalah jumlah *feature* yang diambil

Setelah didapat semua *distance* maka dicari yang paling minimum, maka dilihat *distance* minimum tersebut bersesuaian dengan image ke-berapa dan diambil nama orang dari *image* tersebut.

Secara garis besar, langkah-langkah dalam training dan recognition PCA dapat dilihat pada gambar



Gambar 2 5. *Flowchart* Proses Training PCA



Gambar 2 6. *Flowchart Proses Recognition* PCA

Secara garis besar langkah-langkah dari metode PCA adalah sebagai berikut:

- Cari Matriks *u*
- Cari matriks rata-rata $\bar{u}$
- Cari Matriks Covariance C = u x u^t
- Cari *eigenvalue* (λ) dan *eigenvector* (V) dari matriks C
- Cari matriks *feature* = $f = (I \times \bar{u}) \times V$
- Matriks f akan digunakan untuk mencari *distance* dari *image* yang akan dikenali

2.7 Algoritma Pengenalan Wajah

Sekarang pengenalan wajah telah dikembangkan untuk banyak aplikasi, terutama untuk aplikasi keamanan. Penggunaan wajah sebagai *identifier* mempunyai banyak manfaat, terutama kepraktisannya karena tidak memerlukan kartu atau foto untuk identifikasi. Masalah utamanya adalah sebuah *image* yang mewakili sebuah gambar yang terdiri dari *vector* yang berukuran relatif besar. Ada banyak teknin untuk mereduksi dimensi dari *image* yang diproses. Salah satunya yang akan dibahas disini adalah dengan menggunakan *eigenface algoritm*.

2.7.1 *Eigenvalue, Eigenvektor* dan *Eigenface*

Transformasi ruang seperti translasi, rotasi, refleksi, *strechting* dan kompresi atau kombinasi dari transformasi ini, dapat divisualisasikan dengan efek yang dihasilkan pada vektor. Vektor divisualisasikan sebagai panah yang menunjuk 1 titik ke titik yang lain.

1. *Eigenvector* dari suatu transformasi adalah vektor-vektor yang tidak mengalami perubahan atau hanya dikalikan dengan *scalevector* setelah transformasi.
2. *Eigenvalue* dari suatu *eigenvector* adalah *scalevector* dimana *eigenvector* dikalikan.

Misalnya akan menghitung *eigenvalue* dari suatu matriks yang diberikan. Jika matriks yang diberikan kecil, maka dapat menghitungnya secara simbolis dengan karakteristik polynomial. Akan tetapi ini mustahil untuk matriks dengan ukuran yang lebih besar. Pada penelitian ini akan digunakan metode numerik.

2.7.1.1 Mencari *Eigenvalue*

Salah satu tool penting dalam mendeskripsikan *eigenvalue* dari suatu matriks bujursangkar adalah polynomial karakteristik : jika λ adalah *eigenvalue* dari A maka akan ekuivalen dengan persamaan linear $(A - \lambda i)\ v = 0$ (dimana i adalah matriks

identitas) yang memiliki pecahan non zero v (suatu *eigenvector*), sehingga akan ekuivalen dengan determinan :

$$\det (A - \lambda I) = 0$$

Fungsi $p(\lambda) = \det (A - \lambda I)$ adalah sebuah polynomial dalam λ karena determinan dengan perhitungan *sum of product.* Semua *eigenvalue* dari suatu matriks *A* dapat dihitung dengan menyelesaikan persamaan $pA(\lambda) = 0$. Jika *A* adalah matriks ukuran *n* x *n*, maka *pA* memiliki derajat *n* dan *A* paling banyak *n* buah *eigenvalue*.

2.7.1.2 Mencari *Eigenvector*

Ketika *eigenvalue* λ diketahui, *eigenvector* dapat dicari dengan memecahkan : $(A - \lambda I)v = 0$. Dalam beberapa kasus bisa dijumpai beberapa matriks tanpa *eigenvalue* misalnya A:

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

Dimana karekteristik polinomialnya adalah $\lambda 2 + 1$ sehingga *eigenvalue*-nya adalah bilangan kompleks *i, -i. Eigenvector* yang berasosiasi juga tidak riil.

Perhatikan lagi contoh berikut ini. Jika diberikan matriks *A* :

$$\begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$$

Maka polynomial karakteristiknya dapat dicari sebagai berikut :

$$det \begin{bmatrix} (2 - \lambda) & -1 \\ 0 & (3 - \lambda) \end{bmatrix} = \lambda^2 - 5\lambda + 6 = 0$$

Ini adalah persamaan kuadrat. Akarnya adalah $\lambda = 2$ dan $\lambda = 3$ yaitu:

$= (2 \,.\, -\lambda) \,.\, (3 \,.\, \lambda) - (0 \,.\, -1)$
$= (2 \,.\, 3) + (2 \,.\, -\lambda) + (-\lambda \,.\, 3) + (-\lambda \,.\, -\lambda) - (0)$

$= 6 - 2\lambda - 3\lambda + \lambda^2 - 0$
$= 6 - 5\lambda + 6 - 0$
$= \lambda^2 - 5\lambda + 6 = 0$

Adapun *eigenvector* yang bisa didapat ada 2 buah. *Eigenvector* pertama dicari dengan mensubsitusikan $\lambda = 3$ ke dalam persamaan. Misalnya Y_o adalah *eigenvector* uang berasosiasi dengan *eigenvalue* $\lambda = 3$. Set Y_o dengan nilai

$$Yo = \begin{bmatrix} Xo \\ Yo \end{bmatrix}$$

Sunbsitusikan Y_o dengan v pada persamaan : $(A - \lambda I)\ v = 0$ maka akan mendapatkan :

$$\begin{cases} (2 - 2)\ X_o + -Y = 0 \\ 0 + (3 - 3)\ Y_o = 0 \end{cases}$$

Bisa disederhanakan menjadi : $-X_o - Y_o = 0$ atau $Y_o = -X_o$.

$$\begin{bmatrix} (2-3) & -1 \\ 0 & (3-3) \end{bmatrix} \begin{bmatrix} Xo \\ Yo \end{bmatrix} = \begin{bmatrix} (2-3)Xo & +(-1)Yo \\ 0 & (3-3)Yo \end{bmatrix}$$

$= -1 \cdot Xo + -1 \cdot Yo$
$= 0 + 0 \cdot Yo$
$= - Xo - Yo$

Sehingga *eigenvector* untuk *eigenvalue* = 3 adalah

$$Yo = \begin{bmatrix} Xo \\ Yo \end{bmatrix} = \begin{bmatrix} Xo \\ -Xo \end{bmatrix} = Xo \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

2.7.1.3 *Eigenface*

Eigenface adalah kumpulan dari *eigenvector* yang digunakan untuk masalah *computer vision* pada pengenalan wajah manusia. Banyak penulis lebih menyukai istilah *eigenimage*. Teknik ini telah digunakan pada pengenalan tulisan tangan, pembacaan bibir, pengenalan suara dan pencitraan medis.

Dalam istilah layman, *eigenface* adalah sekumpulan *standardized face ingredient* yang diambil dari analisis statistik dari banyak gambar wajah. Suatu wajah manusia dapat dipandang sebagai kombinasi dari wajah-wajah standar ini. Wajah seseorang bisa saja terdiri dari 10% dari wajah 1, 20% wajah 2, dan seterusnya sehinga jika ingin merekam wajah seseorang untuk pengenalan wajah maka bias digunakan jauh lebih sedikit fitur dari pada yang ditangkap oleh foto digital.

Untuk menghasilkan *eigenface*, sekumpulan besar citra digital dari wajah manusia diambil dari kondisi pencahayaan yang sama kemudian dinormalisasi setelah itu diolah pada resolusi yang sama (misalnya *m* x *n*), dan kemudian diperlakukan sebagai vektor dimensi *mn* dimana komponennya diambil dari nilai pikselnya. Untuk menentukan *eigenface* dari sekumpulan citra wajah, banyak alternatif cara yang digunakan. Salah satunya seperti uraian berikut ini

2.7.1.4 Algoritma *Eigenface*

Eigenface adalah salah satu algoritma pengenalan wajah yang didasarkan pada *Participal Component Analysis* (PCA) yang dikembangkan di MIT. Algoritma *eigenface* secara keseluruhan cukup sederhana. *Training image* direpresentasikan ke dalam sebuah *vectorflat* (gabungan vektor) dan digabung secara bersama-sama menjadi sebuh matriks tunggal. *Eigenfaces* dari masing-masing citra kemudian di ekstrasi dan disimpan dalam file *temporary* atau *database*. Test *image* yang masuk diidentifikasikan juga nilai *eigenface*-nya dan dibandingkan dengan *eigenface* dalam *database* atau file temporary. Adapun algoritma selengkapnya adalah:

1. Buat *MakeFlatVectors* (*ImageList, N, M*): Image List adalah kumpulan dari *N training image*, dimana setiap image adalah *W* x *H* piksel. *M* adalah ukuran dari *vector flat* yang harus dibuat.
2. Gabungkan setiap *image* dalam *WH* elemen *vector* dengan menggabungkan semua baris. Buat *image* matriks sebagai matriks *N* x *WH* berisi semua gambar yang digabung.

3. Jumlahkan semua baris pada *image* matriks dan bagi dengan N untuk mendapatkan rataan vektor gabungan. Namakan vektor elemen *WH* dengan *R*.
4. Kurangi *image* matriks dengan *average image R*. Namakan matriks baru ukuran *N* x *WH* sebagai *R*'.
5. Jika pada elemen-elemen dari matriks *R'* ditemukan nilai negative, ganti nilainya dengan nilai 0.

Kemudian identifikasi dilakukan dengan proyeksi dengan algoritma sebagai berikut:

1. Buat ProjectToFaceSpace(test_image): Image berukuran *W* x *H* Piksel.
2. Gabungkan elemen vektor *WH* dan namakan *img*.
3. Load nilai *R* dari database.
4. Kurangi img dengan *R* hingga didapat *img'*.
5. Jika pada *img'* ditemukan elemen dengan nilai negatif, ganti dengan nilai 0 untuk mendapatkan vektor ukuran *img"*.

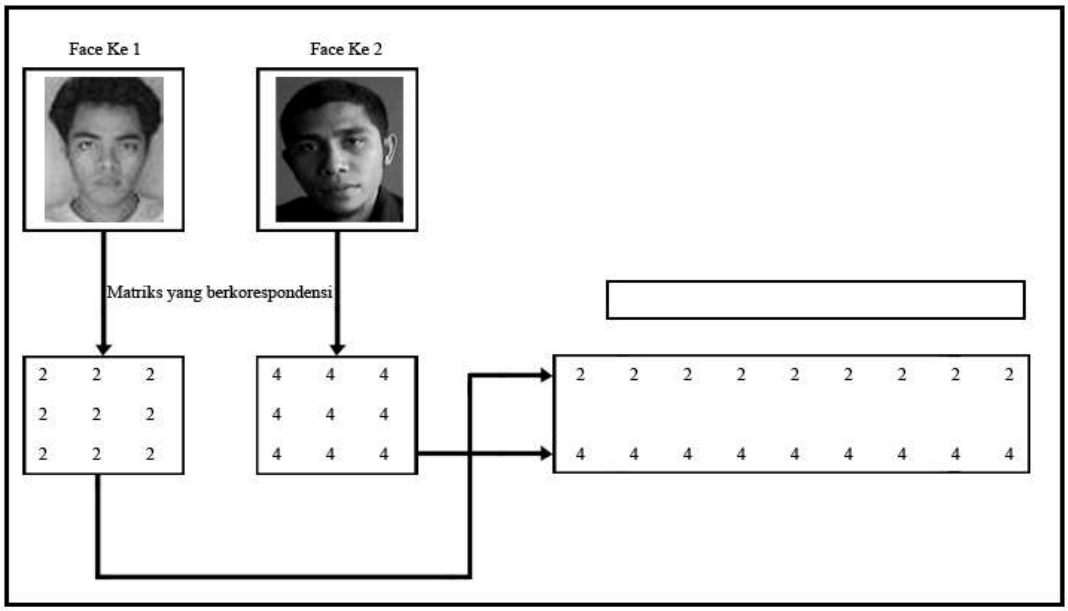
Proses terakir adalah identifikasi yaitu dengan memproyeksikan *test image* ke *face space* dan menghitung *score*.

1. Load semua wajah yang sudah diproyeksikan ke database
2. Proj=projectToFaceSpace(test_image).
3. Lakukan operasi pengurangan, proj dengan semua wajah yang telah diproyeksikan. Ambil nilai absolutnya dan jumlahkan, hasil adalah *score*.
4. Ambil *score* terkecil sebagai hasil dari wajah yang telah diproyeksikan. Wajah ini menjadi hasil identifikasi

Untuk lebih jelasnya algoritma *eigenface* dapat diuraikan sebagai berikut :

1. Penyusunan *Flatvector*

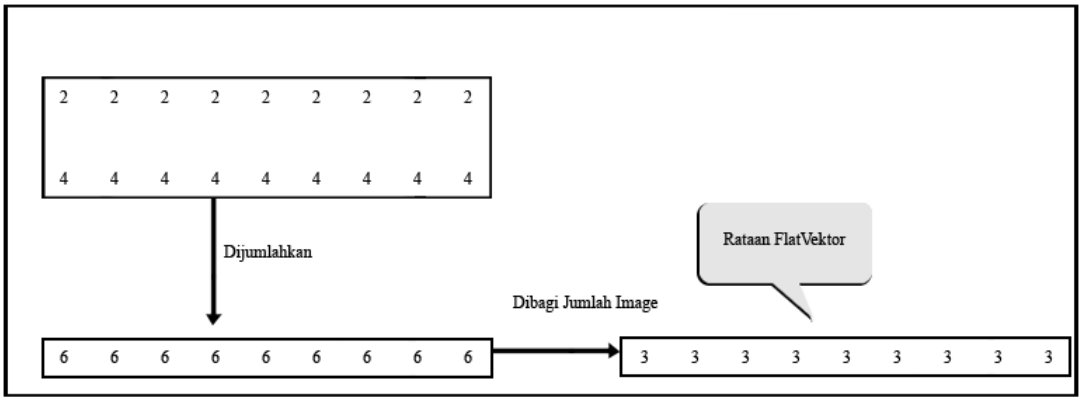
Langkah pertama adalah menyusun seluruh *training image* menjadi suatu matriks tunggal. Misalnya *image* yang disimpan berukuran *H* x *W* piksel dan jumlahnya *N* buah, maka akan dimiliki *vectorflat* dengan dimensi *N* x (*W* x *H*). Misalnya dalam *training image* terdapat 2 *image* dengan ukuran 3 x 4 piksel maka kita akan mempunyai *eigenvector* ukuran 2 x 9. Ilustrasinya sebagai berikut :



Gambar 2. 5 Penyusunan *Flatvector*

2. Perhitugan Rataan *Flatvektor*

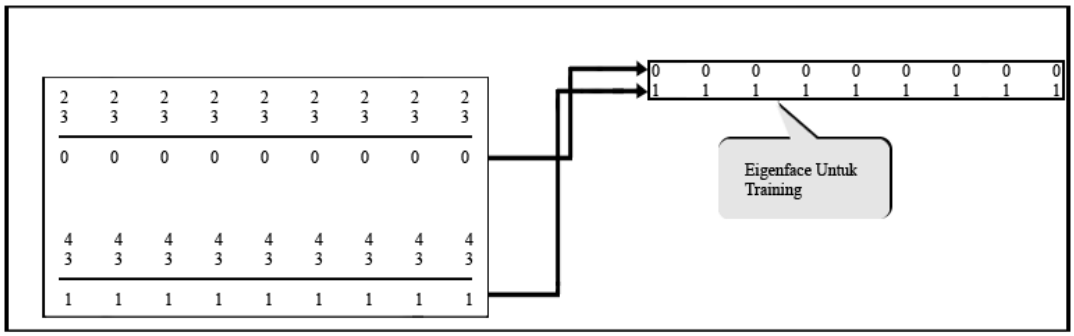
Dari *vectorflat* yang telah diperoleh, jumlahkan seluruh barisnya sehingga diperoleh matriks berukuran 1 x (*W* x *H*). Setelah itu bagi matriks tersebut dengan jumlah *image N* untuk mendapatkan rataan *vectorflat*.



Gambar 2. 6 Rataan *Flatvector*

3. Tentukan Nilai *Eigenface*

Dengan menggunakan rataan *vectorflat*, akan dihitung *eigenface* untuk matriks *vectorflat* yang telah disusun. Caranya dengan mengurangi baris-baris pada matriks *vectorflat* dengan rataan *vectorflat*. Jika didapatkan nilai di bawah nol, ganti nilainya dengan nol.

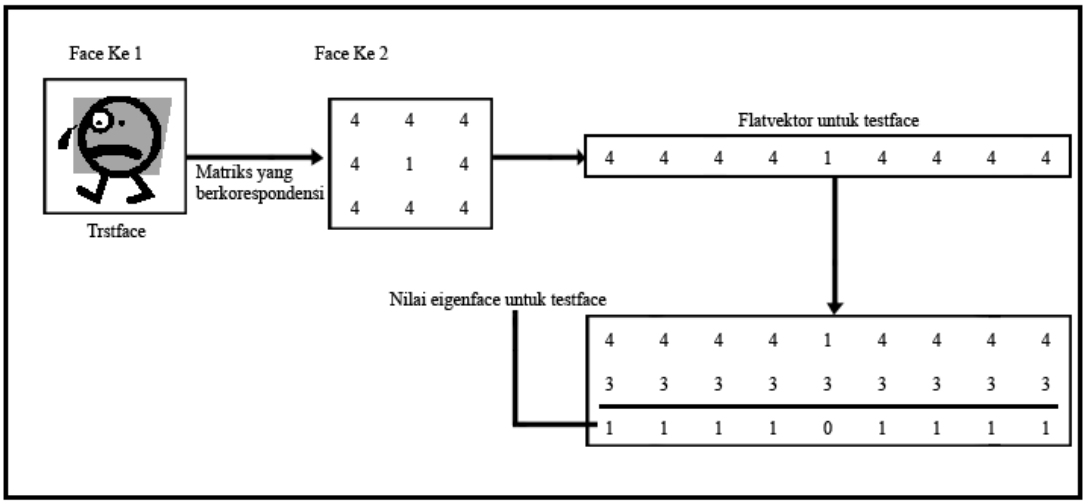


Gambar 2. 7 Perhitungan *Eigenface*

4. Proses Identifikasi

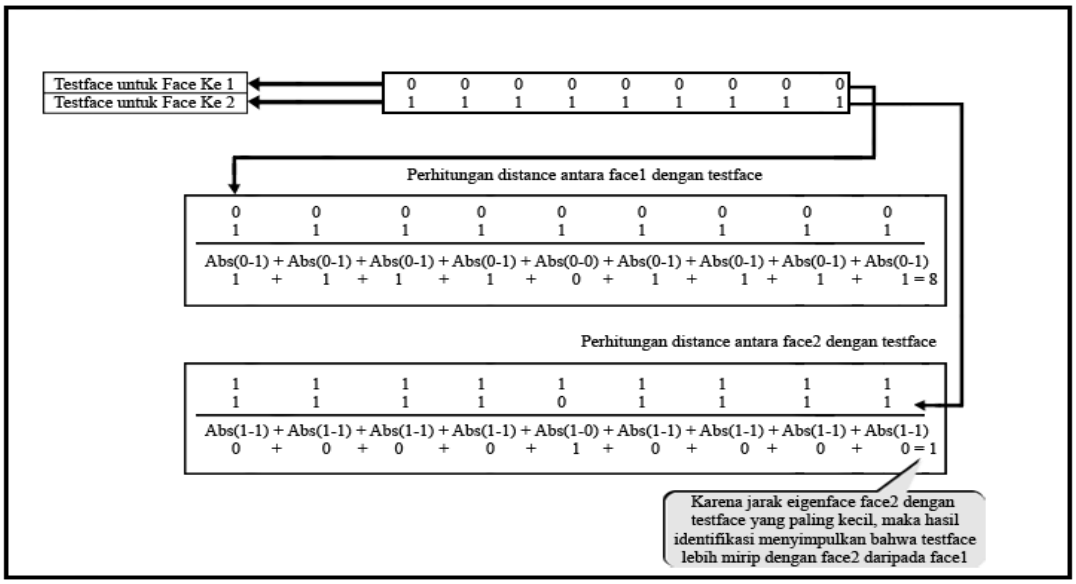
Jika diberikan citra yang akan diidentifikasi (*testface*), maka langkah identifikasinya adalah sebagai berikut :

Kalkulasi nilai *eigenface* untuk matriks *testface,* dengan cara yang sama dengan penentuan *eigenface* untuk *vectorflat*.



Gambar 2. 8 *Eigenface* untuk *testface*

Setelah nilai *eigenface* untuk *terstface* diperoleh maka kita bisa melakukan identifikasi dengan menentukan jarak (*distance*) terpendek dengan *eigenface* dari *eigenvector training image*. Caranya dengan menentukan nilai *absolute* dari pengurangan baris *i* pada matriks *eigenface training image* dengan *eigenface* dari *testface,* kemudia jumlahkan elemen-elemen penyusun vektor yang dihasilkan dari pengurangan tadi dan ditemukan jarak *d* indeks *i*. Lakukan untuk semua baris. Cari nilai *d* yang paling kecil.



Gambar 2. 9 Proses Indentifikasi Dengan *Input Image Testface*