BAB II

LANDASAN TEORI

Untuk dapat membangun atau membuat sebuah sistem keamanan yang bersifat dinamis dimana akan menggabungkan sebuah metode *face recognition* yang selanjutnya akan disebut pengenal wajah dengan *face tracker* maka diperlukan beberapa komponen pendukung.

Komponen utama dalam pembangunan sistem keamanan ini adalah sebuah metode pengenal wajah dan sebuah metode pendeteksi pergerakan wajah (*face tracker*). Dalam pembuatan modul sistem keamanan ini akan digunakan metode *Haar-Cascade* yang sudah terdapat pada OpenCV dan SVM. *Haar-Cascade* digunakan dalam pendeteksi wajah dan pergerakan wajah. Metode SVM digunakan untuk pengenalan wajah.

2.1 Citra

Menurut Kamus Besar Bahasa Indonesia mengandung arti rupa, gambar, gambaran. Sedangkan menurut kamus Webster, citra adalah representasi, kemiripan atau imitasi dari suatu objek atau benda. Jika melihat secara harafiah citra mengandung arti sebuah gambar yang terdapat dalam bidang dua dimensi panjang dan lebar.

Selain itu citra adalah representasi dua dimensi dari dunia visual, menyangkut berbagai macam disiplin ilmu yang mencakup seni, *human vision*, astronomi, teknik, dan sebagainya. Dua dimensi dari dunia visual tersebut memiliki fungsi f(x,y) yang terdapat pada koordinat spasial (x,y) di bidang X. Sedangkan bidang Y mendefinisikan suatu ukuran intensitas cahaya atau kecemerlangan titik tersebut (Fairhurst, 1988).

Sehingga secara garis besar citra sendiri dapat dikatakan sebagai sebuah gambar yang tampak menyerupai benda aslinya dikarenakan adanya perbedaan intensitas cahaya yaitu gelap dan terang dalam bidang dua dimensi.

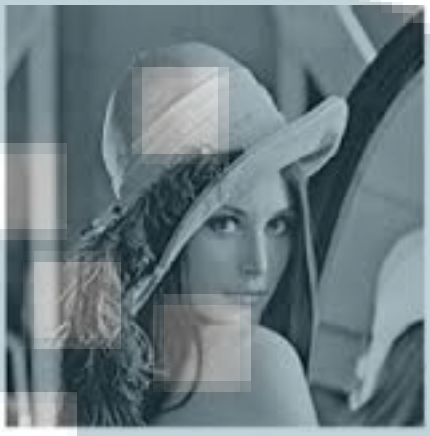
Menurut Lintz Jr. dan Simonett, terdapat tiga rangkaian kegiatan yang diperlukan dalam pengenalan obyek yang tergambar pada citra (Sutanto, 1994), yaitu:

1. Deteksi, adalah pengamatan adanya suatu objek, misalnya pada gambaran sungai terdapat obyek yang bukan air.
2. Identifikasi, adalah upaya mencirikan obyek yang telah dideteksi dengan menggunakan keterangan yang cukup. Misalnya berdasarkan bentuk, ukuran, dan letaknya, obyek yang tampak pada sungai tersebut disimpulkan sebagai perahu motor.
3. Analisis, yaitu pengumpulan keterangan lebih lanjut. Misalnya dengan mengamati jumlah penumpangnya, sehingga dapat disimpulkan bahwa perahu tersebut perahu motor yang berisi dua belas orang.

Citra terbagi menjadi dua bagian (Sitorus *et al.*, 2006), yaitu :

1. Citra Diam

   Citra diam dapat kita artikan sebagai gambar dua dimensi yang tidak bergerak. Contohnya adalah foto.



Gambar 2.1: Citra Diam

2. Citra Bergerak

   Citra bergerak dapat kita artikan sebagai gambar dua dimensi yang tampak oleh mata seolah olah gambar tersebut bergerak. Padahal yang sebenarnya terjadi adalah gabungan dari beberapa citra diam yang disusun dan dijalankan secara berurutan sehingga tampak bergerak. Contohnya adalah film animasi.

2.2 *Grayscale*

Secara pengolahan digital, gambar yang bersifat *grayscale* sebenarnya dapat diinterpretasikan dalam bentuk *array* dua dimensi (X, Y) di mana setiap elemen pada *array* tersebut memiliki intensitas warna yang berbeda pada gambar tersebut sesuai dengan koordinat yang telah ditentukan dan yang pada akhirnya akan menjadi sebuah gambaran utuh yang bersifat *grayscale* atau abu - abu.

Menurut kamus webster *grayscale* adalah serangkaian warna dari hitam hingga putih yang melalui nuansa abu - abu. Sedangkan menurut Kamus Besar Bahasa Indonesia *grayscale* atau abu - abu adalah warna antara hitam dan putih.

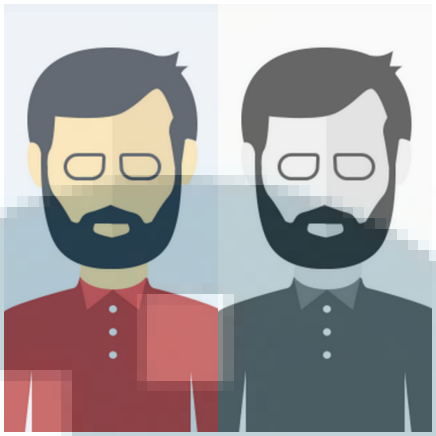
Gambar 2.2: Tingkat Keabuan Warna

Setiap elemen pada *array* gambar dapat juga dan sering kita sebut sebagai *pixel*. Nilai tiap elemen pada *array* atau *pixel* tersebut berbeda beda tergantung dari intensitas warna yang dihasilkan. Nilai elemen tersebut terdiri dari 256 nilai yang dimulai dari angka 0 hingga 255 dimana nilai 0 menunjukkan intensitas yang paling gelap atau hitam, sedangkan nilai 255 menunjukkan nilai yang paling terang atau putih.

Dalam menentukan nilai sebuah *grayscale* kita dapat menggunakan sebuah fungsi / persamaan sederhana, yaitu W = (R + G + B) / 3 dimana W adalah nilai warna abu - abu, R adalah nilai warna merah, G adalah nilai dari warna hijau dan B adalah nilai dari wana biru.

Gambar 2.3 telah mengalami perubahan warna dari RGB menjadi *grayscale* / abu-abu. Untuk menentukan warna tingkat keabuannya, kita tinggal memasukkan nilai setiap warna per *pixel* ke dalam persamaan tersebut.

Berikut adalah contoh konversi RGB menjadi *Grayscale*



Gambar 2.3: Perbedaan Warna Asli dengan *Grayscale*

R = 155

G = 133

B = 134

W = (155 + 133 + 134) / 3

W = 140,67

Dari hasil perhitungan tersebut, dapat kita simpulkan bahwa warna yang akan muncul tingkat keabuan / kegelapannya berada di tengah kebawah sekitar 40% hingga 50%. Nilai tersebut dapat kita peroleh jika kita membandingkan nilai W dengan nilai 0 sebagai hitam dan nilai 255 sebagai putih.

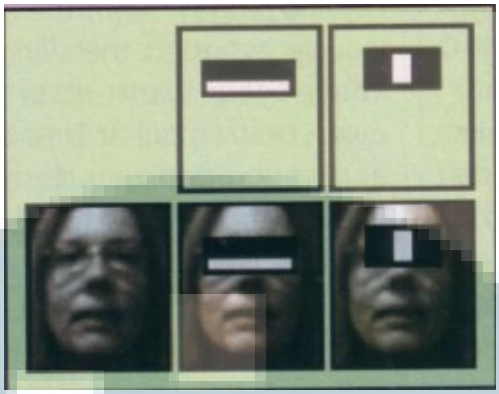
Grayscale tidak sama dengan *monochrome*. *Grayscale* adalah abu abu yang dapat terbagi berdasarkan intensitas kecerahan warnanya seperti abu muda, abu sedang maupun abu tua, sedangkan *monochrome* adalah hitam atau putih. Elemen warna pada gambar *monochrome* hanya terdapat dua jenis, yaitu hitam atau putih.

2.3 *Haar-like Feature / Haar-Cascade*

Haar-like feature merupakan metode yang sangat sering digunakan dalam pendeteksian sebuah obyek. Dalam pendeteksian wajah, *haar-like feature* memiliki beberapa tahapan yang harus dilalui untuk menentukan terdapat tidaknya sebuah wajah. Gambar 2.4 ini adalah alur proses dari metode *haar-like feature*.

Gambar 2.4: Alur Haar-Cascade

Fitur dari haar akan dijalankan ketika akan mencari obyek yang ingin dicari dan disini adalah wajah manusia. Gambar 2.5 ini adalah contoh gambar yang sedang diidentifikasi oleh fungsi haar. Nilai yang muncul dari fitur haar tersebut yang akan kemudian dimasukkan kedalam *integral image*.



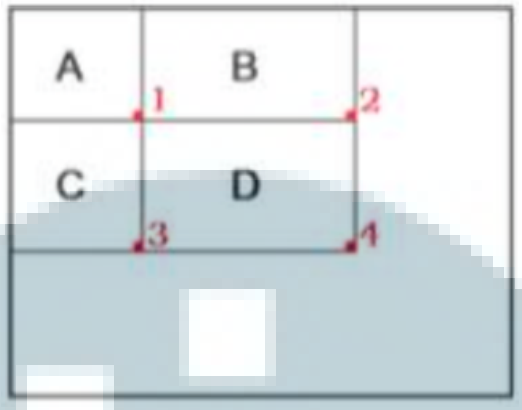
Gambar 2.5: Hasil Fitur Persegi Haar-Cascade

Haar-like feature memiliki tiga jenis fitur berdasarkan jumlah persegi panjang di dalamnya (Krishna dan Srinivasulu, 2012) seperti pada gambar 2.6 tersebut.

Gambar 2.6: Fitur Haar-Cascade

Pada gambar 2.6 memperlihatkan bahwa fitur A dan B memiliki dua buah persegi panjang. Sedangkan fitur C memiliki tiga buah persegi panjang dan untuk fitur D memiliki empat buah persegi panjang di dalamnya. Untuk menghitung nilai dari fitur ini, kita harus mengurangkan nilai piksel pada area putih dengan nilai *pixel* pada area hitam.

Untuk mempermudah dalam penghitungan nilai dari haar fitur itu sendiri, kita dapat menggunakan sebuah media berupa *Integral Image* (Gambar 2.7). *Integral*



Gambar 2.7: *Integral Image*

Image adalah sebuah citra yang nilai pada setiap pikselnya merupakan penjumlahan dari piksel lainnya. Nilai piksel yang dijumlahkan adalah nilai piksel dari kiri atas hingga kanan bawah. Untuk menghitung nilai haar terdapat rumusan sebagai berikut:

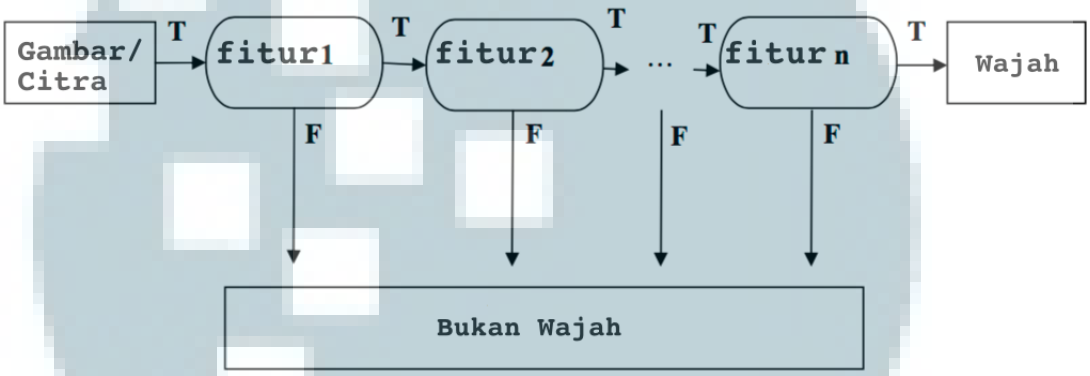
$$D = D + A - (B + C)$$

Keterangan:

- D adalah nilai piksel kanan bawah
- A adalah nilai piksel kiri atas
- B adalah piksel atas dari piksel D
- C adalah nilai piksel kiri dari piksel D

Dalam metode haar ini, Viola dan Jones menggunakan sebuah metode *AdaBoost*. *AdaBoost* ini sendiri adalah sebuah metode dimana ia akan menggabungkan banyak *clasifier - clasifier* yang lemah untuk menjadikannya sebuah *clasifier* yang kuat.

AdaBoost akan menggabungkan fitur yang dimiliki oleh *haar-like feature* tersebut dengan cara bertahap. Setiap tahap akan mengklasifikasikan dan membuang *sub picture* yang dinilai tidak sesuai dengan yang diinginkan. Terjadinya penyaringan bertahap ini yang akhirnya membuat namanya menjadi *haar cascade*. Gambar 2.8 adalah alur kerja klasifikasi bertingkat.



Gambar 2.8: Alur Klasifikasi Bertingkat

Pada klasifikasi tingkat pertama, tiap *sub picture* akan diklasifikasi menggunakan satu fitur. Klasifikasi ini kira-kira akan menyisakan 50% *sub picture* untuk diklasifikasi di tahap kedua. Seiring dengan bertambahnya tingkatan klasifikasi, maka diperlukan syarat yang lebih spesifik sehingga fitur yang digunakan menjadi lebih banyak. Jumlah *sub picture* yang lolos klasifikasi pun akan berkurang hingga mencapai jumlah sekitar 2% pada akhirnya (Dzulkamain *et al.*, 2011). Gambar 2.9 adalah hasil yang muncul jika pendeteksian menggunakan *haar-cascade* berhasil.

Kotak pada wajah seperti gambar 2.9 tersebut dapat muncul dan berukuran sesuai dengan wajah dikarenakan adanya nilai x, w, y, h pada obyek tersebut.

Keterangan:

Gambar 2.9: Hasil Deteksi Wajah dengan Haar-Cascade

- x = Koordinat titik x. Dimulai dari kiri
- w = Besaran nilai lebar gambar yang terdeteksi
- y = Koordinat titik y. Dimulai dari atas
- h = Besaran nilai panjang / tinggi gambar yang terdeteksi

Pada penelitian ini pendeteksian menggunakan metode *haar-cascade* karena metode ini memiliki sebuah kelebihan. Kelebihannya yaitu proses komputasinya yang sangat cepat, karena hanya bergantung pada jumlah piksel dalam persegi bukan setiap nilai piksel dari sebuah gambar (Viola dan Jones, 2001).

2.4 *Support Vector Machines* (SVM)

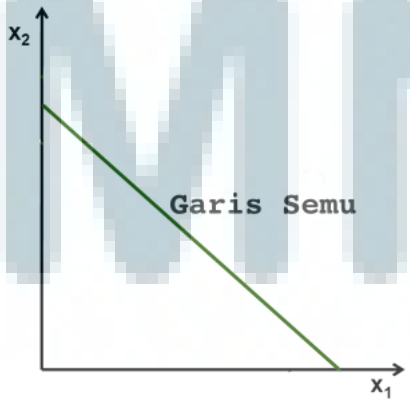
Support Vector Machine (SVM) adalah sebuah metode atau teknik pembelajaran mesin yang paling mutakhir yang merupakan pembaruan dari teknik pembelajaran mesin terdahulunya yaitu *neural network*. Metode SVM dipilih karena

memiliki hasil yang lebih baik jika dibandingkan dengan pendahulunya yaitu *neural network*. Selain itu SVM juga memiliki algoritma yang lebih baik dan dapat mencari solusi dengan tahapan yang singkat (Anguita dan Boni, 2002).

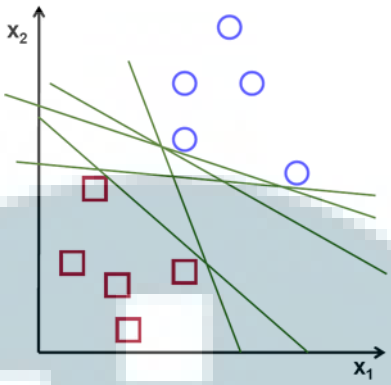
Kedua teknik pembelajaran tersebut sama sama mempelajari sebuah pola dari gambar *input*-an yang telah kita berikan kepada komputer berupa kumpulan data / dataset yang kita inginkan untuk dipelajari oleh komputer. Kumpulan data tersebut adalah pembandingan antara data yang kita inginkan dengan data data lainnya.

Dalam pembuatan *face recognition* dibutuhkan dua buah dataset. Dataset pertama berupa *database* wajah yang benar. Sedangkan dataset kedua berisikan *database* wajah yang salah. Kedua dataset tersebut digunakan oleh sistem untuk mengenali wajah pada citra *input*-an berikutnya.

Data yang telah diolah oleh SVM dengan metode *linear* akan terbagi menjadi dua bagian yaitu hasil yang diinginkan dengan hasil yang salah atau tidak diinginkan yang dipisahkan oleh sebuah garis semu (gambar 2.10).



Gambar 2.10: Garis Semu / Bias

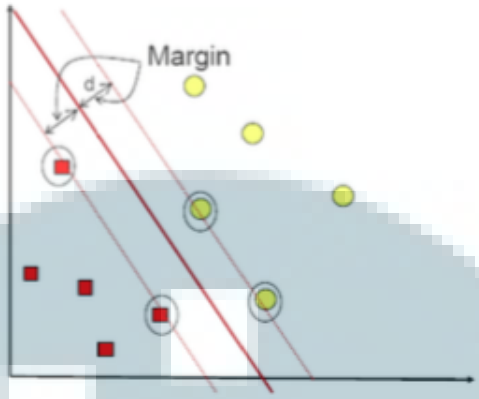


Gambar 2.11: Hasil SVM

Kita dapat mencari garis semu tersebut dengan menggunakan persamaan f(x) = ax1 + bx1 + c. Dengan adanya persamaan ini, maka SVM dapat membedakan kesamaan pola yang kita berikan sesuai apa tidak dengan membandingkan nilai dari persamaan tersebut. Jika f(x) > 0, maka pola tersebut kita nyatakan benar, dan sebaliknya, jika f(x) < 0, maka pola tersebut dinyatakan berbeda / salah.

Gambar 2.11 merupakan gambaran hasil dari proses SVM pada gambar - gambar kita. Lingkaran yang berwarna biru menunjukkan adanya kesesuaian pola pada gambar yang kita berikan, sedangkan persegi berwarna merah menunjukkan adanya perbedaan pola. Obyek yang hasilnya berada di bawah garis semu akan dinyatakan sebagai *False* atau salah.

Hasil yang berada pada garis bias membuat sistem salah mengambil keputusan. Guna untuk mencegah hal tersebut, SVM mencari dua buah *hyperlane* optimal. *Hyperlane* adalah garis semu tambahan yang telah ditambahkan dengan jarak tertentu (*margin*). Dengan adanya *hyperlane* tersebut, sistem dapat memperkecil kesalahan ketika melakukan *training* (Wang, 2005)



Gambar 2.12: Garis Hyperlane

Berdasarkan hasil dari proses SVM ini, maka kita dapat mengenali dan mengetahui pola dari setiap gambar. Dalam *face recognition* sendiri, kita mengambil pola yang benar dalam proses verifikasinya.

2.5 *K-Fold Cross-Validation*

K-fold cross-validation biasa disebut juga sebagai *rotation estimation*. Dengan metode ini sebuah dataset / *database* yang utuh dipecah secara acak menjadi k subset. Setiap subset memiliki jumlah data yang sama antar setiap subsetnya agar hasil yang didapat akurat.

Pada penelitian ini, dataset dipecah menjadi 10 *fold* (k=10). Dataset dipecah menjadi 10 bagian dikarenakan menurut penelitian terdahulu, *stratified 10-fold cross validation* direkomendasikan untuk pemilihan model (Kohavi, 1995). Setiap menjalankan uji tingkat akurasi, 9 *fold* akan dijadikan sebagai dataset. Sedangkan 1 *fold* sisanya akan dijadikan sebagai bahan pengujian. Berikut adalah simulasi dari pembagian dataset menjadi 10 subset:

- Tahap 1 = Dataset : k1-k9 - Bahan Uji : k10
- Tahap 2 = Dataset : k1-k8 + k10 - Bahan uji : k9
- Tahap 3 = Dataset : k1-k7 + k9-k10 - Bahan uji : k8
- Tahap 4 = Dataset : k1-k6 + k8-k10 - Bahan uji : k7
- Tahap 5 = Dataset : k1-k5 + k7-k10 - Bahan uji : k6
- Tahap 6 = Dataset : k1-k4 + k6-10 - Bahan uji : k5
- Tahap 7 = Dataset : k1-k3 + k5-k10 - Bahan uji : k4
- Tahap 8 = Dataset : k1-k2 + k4-k10 - Bahan uji : k3
- Tahap 9 = Dataset : k1 + k3-k10 - Bahan uji : k2
- Tahap 10 = Dataset : k9-k10 - Bahan uji : k1

Penilaian *k-fold cross-validation* terhadap akurasi model secara keseluruhan dihitung dengan mengambil nilai rata - rata dari semua hasil akurasi individu k, seperti yang ditunjukkan dengan persamaan berikut:

$$CVA = \frac{1}{k}\sum_{i=1}^{k} A_i$$

Gambar 2.13: Rumus K-Fold Cross-Validation

keterangan:

- CVA : Akurasi cross-validation

- k : Jumlah fold yang digunakan
- A : Ukuran akurasi dari masing-masing fold

2.6 *Receiver Operating Characteristic*

Metode analisa dengan *Receiver Operating Characteristic* (ROC) adalah sebuah alat yang berguna untuk mengevaluasi, mengklasifikasi dan sebagai alat grafis untuk menampilkan tingkat akurasi dalam sebuah penelitian (Zou et al., 2007). Selain itu metode ROC juga dapat untuk menganalisa tingkat performa suatu obyek penelitian (Hanley, 1989).

Dalam metode ROC ini terdapat dua klasifikasi biner. Jika hasilnya benar maka akan mendapat label klasifikasi sebagai *positive* (P). Dan jika hasilnya salah maka akan mendapatkan label klasifikasi sebagai *negative* (N).

Dari dua klasifikasi biner tersebut dapat muncul empat variasi kemungkinan yang akan timbul (gambar 2.14). Kemungkinan tersebut adalah :

- **TP** *(True Positive)* : Dapat dikatakan sebagai TP apabila hasil yang diprediksi dan hasil yang sesungguhnya adalah benar (P). Contoh :

  Hasil Prediksi : benar

  Hasil Sebenarnya : benar

- **TN** *(True Negative)* : Dapat dikatakan sebagai TN apabila hasil yang diprediksi dan hasil yang sesungguhnya adalah salah (N). Contoh :

  Hasil Prediksi : salah

Hasil Sebenarnya : salah

- **FP** *(False Positive)* : Dapat dikatakan sebagai FP apabila hasil yang diprediksi benar (P) tetapi hasil yang sesungguhnya adalah salah (N). Contoh :

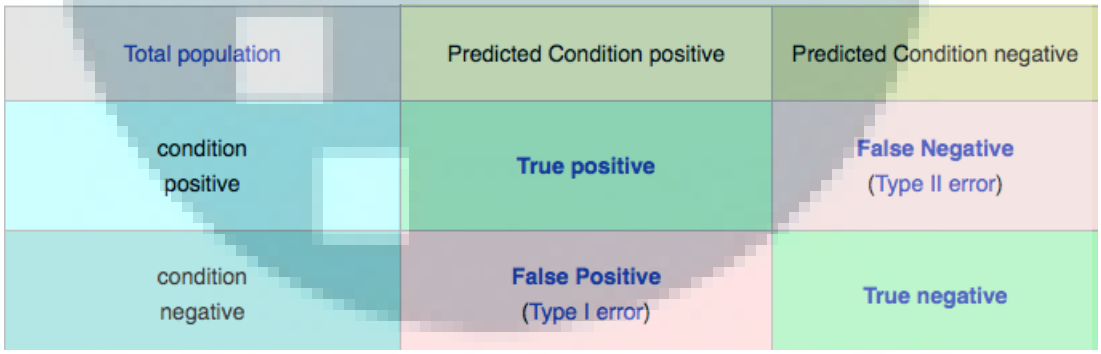
  Hasil Prediksi : benar

  Hasil Sebenarnya : salah

- **FN** *(False Negative)* : Dapat dikatakan sebagai FN apabila hasil yang diprediksi salah (N) tetapi hasil yang sesungguhnya adalah benar (P). Contoh :

  Hasil Prediksi : salah

  Hasil Sebenarnya : benar



Gambar 2.14: Variasi Klasifikasi Biner ROC

Dalam penelitian ini metode ROC digunakan dalam pengukuran tingkat akurasi dari hasil uji coba terhadap responden terpilih. Agar kita dapat mengukur tingkat akurasinya, *variable* TP dan TN adalah kunci utamanya. TP pada penelitian ini adalah jumlah sistem dapat melakukan pengenalan wajah benar dengan benar. Kemudian TN adalah sistem dapat mengenali wajah yang salah dengan benar. Pengukuran tingkat akurasi dengan metode ROC memiliki rumus sebagai berikut:

(Jumlah *True Positive* + Jumlah *True Negative*) / Jumlah *Sample*

2.7 Notasi O Besar

Dalam mengukur tingkat kompleksitasnya, pada penelitian ini menggunakan metode notasi O besar. Angka dalam notasi O besar menunjukkan kompleksitas sebuah program atau sebuah algoritma (Fanani, 2007). Notasi O besar biasa disebut juga sebagai notasi Landau *(Landau Notation)* atau Notasi Asimptotik *(Asymptotic Notation)* yang adalah notasi matematika yang digunakan untuk menggambarkan sifat suatu fungsi asimptotik.

Notasi O besar pertama kali diperkenalkan oleh Paul Bachmann dari Jerman. Kemudian notasi ini dipopulerkan oleh Edmund Landau sehingga disebut sebagai notasi Landau. O Besar sendiri sebenarnya adalah singkatan dari *order of* dalam bahasa Inggris, yang adalah lambang dari Omicron besar. Pada penggunaannya digunakan huruf O besar bukan angka 0 (bilangan nol).

Terdapat dua macam jenis dari notasi O besar ini. Jenis pertama adalah asimptotik tak hingga *(Infinite Asymptotik)* dan asimptotik sangat kecil *(Inifinitesimal Asymptotik)*. Keduanya memiliki konsep yang sama tetapi memiliki perbedaan pada penggunaannya.

(Infinite Asymptotik) dapat digunakan untuk mengukur dan menganalisa kompleksitas sebuah program. Sedangkan *(Inifinitesimal Asymptotik)* dapat digunakan untuk menggambarkan kesalahan dalam sebuah aproksimasi dalam fungsi matematika. Berikut ini adalah contoh dari penggunaan *(Infinite Asymptotik)* :

$T(n)=4n^2 2n+2$

Jika pada persamaan tersebut nilai N kita berikan angka 100, maka nilai dari $4n^2$ akan mendominasi nilai lainnya. Dengan demikian kita dapat mengabaikan nilai - nilai lainnya karena tidak berdampak secara signifikan. Dengan kata lain kita hanya mengambil nilai yang terbesar yang dijadikan sebuah acuan besaran kompleksitas dari sebuah program.

2.8 Floating point Operations Per Second

Floating point Operations Per Second atau yang biasa disebut dengan FLOPS adalah kemampuan komputer dalam melakukan perhitungan bilangan pecahan *(floating point)* dalam satu satuan waktu (Manzoor, 2012). FLOPS merupakan satuan pengukuran kecepatan kinerja suatu *mikroprosesor* biasanya dalam suatu aplikasi ilmiah.

Satuan dalam FLOPS ada beraneka ragam, seperti *kiloflops* (kFLOPS), *megaflops* (MFLOPS), *teraflops* (TFLOPS), *petaflops* (PFLOPS) hingga *yottaflops* (YFLOPS). Tetapi untuk sebuah komputer pada umumnya hanya mencapai pada tahap *megaflops* (MFLOPS).

Untuk mengetahui tingkat FLOPS pada sebuah komputer, terdapat dua cara. Pertama adalah dengan melakukan perhitungan manual. Kita dapat melakukan perhitungan manual dengan rumus

FLOPS = sockets x (cores/socket) x clock x (FLOPs / cycle)

Selain dengan cara perhitungan manual tersebut, kita dapat menggunakan aplikasi Linpack. Tujuan dari Linpack adalah mengukur seberapa cepat komputer dapat menyelesaikan sebuah masalah.

2.9 Python

Bahasa pemrograman python adalah bahasa pemrograman level tinggi yang dikembangkan oleh Guido van Rossum pada tahun 1990an. Python merupakan program *open source* yang memiliki skala besar dan fitur yang cukup lengkap. Python sendiri mengijinkan dan membuat para penggunanya dapat melakukan pemrograman yang lebih sederhana jika dibandingkan dengan bahasa pemrograman C/C++.

Gambar 2.15: Logo Pyhton

Selain *open source*, python juga mendukung berbagai paradigma dalam pemrograman seperti *object oriented*, struktural, imperative serta *functional*. Tak hanya itu, python juga dapat di jalankan pada berbagi *Operating System* seperti Windows, Linux, dan Mac serta dapat diunduh secara gratis.

Berikut adalah beberapa tipe data yang terdapat pada pyhton (Knowlton, 2008):

- *Number* : meyimpan data dalam bentuk angka (numeric) yang dapat bersifat *int (integer), long, float* dan *complex*
- *string* : menyimpan data dalam bentuk huruf yang diawali dan diakhiri oleh tanda petik dua (")
- *list* : menyimpan beberapa item dalam satu kesatuan yang dipisahkan oleh tanda koma (,) dan diawali dan diakhiri oleh tanda kurung kotak ([])
- *tuple* : menyimpan beberapa item dalam satu kesatuan yang dipisahkan oleh tanda koma (,) dan diawali dan diakhiri oleh tanda kurung (()). Isi data tidak dapat dirubah
- *dictionary* : menyimpan data lain seperti *array* yang diawali dan diakhiri dengan tanda kurung kurawal ()

Pyhton memiliki beberapa perluasan fungsi lainnya guna menunjang penggunanya dalam melakukan pemrograman. Fungsi tersebut antara lain seperti OpenCV, NumPy, Scikit Learn, dan lainnya.

2.9.1 OpenCV

OpenCV atau yang dapat juga disebut dengan *Open Source Computer Vision Library* adalah sebuah perangkat lunak yang telah dikembangkan oleh intel sejak tahun 1999 dalam penelitian mereka sebagai lanjutan dari peneletian mereka mengenai tampilan tiga dimensi yang pada akhirnya menghasilkan sebuah perangkat lunak yang dapat mengolah citra secara *real time*.

Computer vision berhubungan berhubungan erat dengan perolehan gambar, pemrosesan, klasifikasi, pengenalan, dan menjadi penggabungan, pengurutan pembuatan keputusan menuju pengenalan (Low, 1991).

Dengan adanya *vision* tersebut maka komputer dapat melihat seperti manusia dengan bantuan kamera dan dapat membuat aksi, mengambil keputusan bahkan dapat mengenali berbagai macam objek.

Perangkat lunak ini dapat digunakan secara lintas *Operating Sistem* dan bebas digunakan oleh siapa saja *(open source)*. OpenCV juga mendukung beberapa bahasa pemrograman yang biasa digunakan seperti C, C++, Python, maupun Java.

Karena kegunaan dari OpenCV ini dibuat untuk melakukan pengolahan citra, maka dengan menggunakan OpenCV ini, setiap proses yang berhubungan dengan pengolahan citra akan berjalan lebih cepat jika kita menggunakan OpenCV ini.

2.9.2 NumPy

Bahasa pemrograman Python awalnya tidak dirancang untuk melakukan komputasi numerik. Akan tetapi masyarakat ilmiah / teknik tertarik untuk membuat bahasa pemrograman Python untuk melakukan komputasi tersebut, sehingga sebuah kelompok kepentingan khusus yang disebut matriks-sig didirikan pada tahun 1995 dengan tujuan mendefinisikan sebuah *array* paket komputasi dengan Python.

Salah seorang anggotanya adalah Guido van Rossum. Ia adalah seorang python desainer dan pemelihara python itu sendiri. Guido menanamkan sebuah

perluasan dari python yang berupa sintaks pengindeksan untuk membuat berbagai komputasi numerik menjadi lebih mudah.

Paket matriks untuk python dibuat dan diselesaikan oleh Jim Fulton, kemudian digeneralisasi oleh Jim Hugunin menjadi Numeric. kemudian muncul sebuah paket baru yang disebut Numarray yang dibuat sebagai pengganti dari Numeric dikarenakan sifatnya yang lebih *flexible*.

Tetapi dalam penggunaannya Numarray memiliki kecepatan operasi yang lebih cepat hanya ketika mengolah sebuah *array* yang besar. Jika mengolah *array* yang kecil, Numarray terlalu kompleks sehingga kecepatan operasinya lebih lambat jika dibandingkan oleh Numeric. Melihat hal tersebut membuat Numeric dan Numarray digunakan pada waktu dan fungsi yang berbeda sesuai dengan kebutuhan.

NumPy adalah sebuah perluasan dari python yang dapat mendukung pengolahan multi dimensi *array* dan matriks. Dalam melakukan pengolahan *array* dan matriks, NumPy dibantu oleh sebuah kumpulan *(library)* fungsi matematika tingkat tinggi agar prosesnya dapat berjalan dengan baik. NumPy sendiri muncul sebagai pembaharuan dari Numeric dan Numarray yang memiliki tingkat kecepatan operasi yang lebih baik dari pendahulunya. Hasil dari NumPy disajikan sebagai *array* numpy, sehingga dapat terintegrasi secara baik dengan *library* Python lainnya (Walt *et al.*, 2011).

Contoh dasar dalam penggunaan NumPy pembuat array di Pyhton :

Python 2.7.10 (default, Oct 23 2015, 19:19:21)

[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin ”help”, ”copyright”, ”credits” or ”license” for more information.

```
import numpy as np
x = np.array([1,2,3])
x
array([1, 2, 3])
```

2.9.3 *Scikit Learn*

Scikit Learn atau biasa dikenal sebagai sklearn adalah perpanjangan dari python yang digunakan secara khusus sebagai media pembelajaran bagi mesin. Dengan *Scikit Learn* kita dapat mengimplementasikan berbagai algoritma pembelajaran mesin terdahulu dengan antarmuka yang mudah digunakan (Pedregosa *et al.*, 2011). *Scikit Learn* sendiri mempunyai banyak kegunaan dan yang paling banyak digunakan adalah untuk *data mining*, *data analyst* dan *face recognition*.

Fitur yang dimiliki oleh *Scikit Learn* yaitu :

- *Classification*

  Classification berguna untuk menentukan kepemilikan sebuah objek tertentu. Contoh: Mata adalah objek milik wajah

- *Regression*

  Regression berguna untuk memprediksi sebuah atribut secara berkelanjutan yang terkait dengan sebuah objek

- *Clustering*

  Clustering berguna untuk menyatukan dan menggabungkan dari beberapa objek yang memiliki kesamaan menjadi sebuah satu kesatuan

- *Dimensionality Reduction*

  Dimensionality Reduction berguna untuk mengurangi jumlah *variable* yang sangat banyak menjadi sesuai dengan keinginan kita

- *Model Selection*

  Model Selection berguna untuk membandingkan, memvalidasi dan memilih parameter dan model

- *Preprocessing*

  Preprocessing berguna untuk melakukan ekstraksi dan normalisasi

2.10 Penelitian Terdahulu

Pada penelitian yang berjudul *BioID: A Multimodal Biometric Identification System* yang telah dilakukan oleh Robert dan Dieckmann pada tahun 2000, mereka telah mencoba untuk menutup kelemahan yang terdapat dalam sistem keamanan satu fitur *face recognition*. Kelemahan yang mereka temukan adalah kesulitan dalam membedakan wajah seseorang jika mereka adalah kembar identik. Oleh karena itu mereka merancang sebuah sistem keamanan yang menggabungkan 3 buah fitur, yaitu *face recognition, lip movement recognition* dan *voice recognition*.

Dalam penelitian ini untuk pengambilan datanya, mereka menggunakan vi-

deo yang berdurasi 1 detik yang terdiri dari 25 *frame*. Dari video ini mereka melakukan ekstraksi untuk wajah, bibir dan suaranya.

Untuk melakukan *face recognition* pada penelitian ini menggunakan metode *Hausdorff Distance* yang memiliki tingkat kompleksitas n^2. Mereka mengambil gambar pertama yang muncul dalam video tersebut. Setelah gambar terambil, gambar tersebut melalui beberapa proses yaitu *rotating, scaling* dan merubah warnanya menjadi *grayscale*. Setelah gambar berubah menjadi *grayscale* barulah proses pengenalan wajah dilakukan berdasarkan data *training* yang telah ada.

Dalam *lip movement recognition*, 17 *frame* awal diambil sebagai data *training*. 17 *frame* dengan asumsi bahwa bibir mulai bergerak pada frame kedua sehingga didapatkannya 16 gambar bibir yang bergerak. Guna mengurangi besaran data yang digunakan, pada penelitian ini menggunakan *3D fast Fourier transformation* yang memiliki tingkat kompleksitas sebesar n^2. 16 gambar yang telah ada tersebut dirubah menjadi *one- dimensional lip movement feature vector*. Hasil inilah yang dijadikan sebagai data untuk *training* dan *classification*.

Saat pembuatan *voice recognition*, *vector quantification* dengan kompleksitas n^2 digunakan untuk melakukan klasifikasi. Suara yang diambil adalah suara tunggal respondennya. Suara - suara yang dihasilkan setiap individu memiliki *pattern* yang berbeda - beda. Dari setiap *pattern* tersebut memiliki matriks. Matriks yang telah ada digabungkan oleh *vector quantification* menjadi satu buah matriks besar. Hasil inilah yang digunakan mereka sebagai data *training* dalam melakukan proses klasifikasi.

Setelah ketiga fitur selesai dibuat, fitur - fitur tersebut digabungkan dan menjadi sebuah sistem keamanan yang baru. Hasil tingkat akurasi yang dicapai pada penelitian ini mencapai 99%.

Pada penelitian ini, mereka menggunakan komputer intel Pentium II dengan kecepetan 0.45 *gigaflops* (GFLOPS). Dengan komputer tersebut, proses *training* diselesaikan dalam waktu 15 menit untuk setiap *pattern* dan dalam proses klasifikasi hanya memakan waktu beberapa milidetik.