BAB II
DASAR TEORI

2.1 Teknologi Pengolahan Citra

Teknologi pengolahan citra sekarang ini sudah semakin berkembang, hal tersebut ditandai dengan mulai banyaknya aplikasi-aplikasi yang menerapkan teknologi ini. Pengolahan citra merupakan proses yang memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau komputer. Sedangkan definisi citra sendiri adalah gambar pada bidang 2 dimensi atau secara matematis pengertiannya adalah fungsi yang kontinu dari intensitas cahaya pada bidang 2 dimensi.

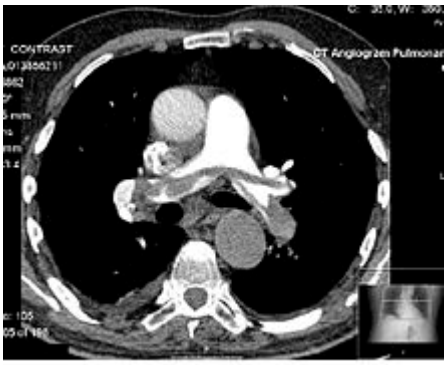
Pengolahan citra digital mulai dikembangkan pertama kali pada tahun 1964, dimana pada saat itu pengolahan citra digital digunakan untuk memperbaiki kualitas citra bulan dari distorsi di laboratorium *Jet Propulsion*. Sampai perkembangannya hingga sekarang, teknologi ini telah memiliki spektrum aplikasi yang sangat luas, diantaranya:

1. Dibidang biometrika, yang memanfaatkan sidik jari, retina, wajah, dan lainnya untuk teknologi sistem keamanan.



Gambar 2.1. Kunci Pintu Dengan Sidik Jari Biometrik

2. Dibidang biomedis terdapat tomografi terkomputerisasi (*computerized tomography*/CT) dan scanner *Magnetic Resonance Imaging* (MRI) yang digunakan untuk mengidentifikasi organ dalam tubuh.



Gambar 2.2. Hasil Pencitraan computerized tomography*(CT)*

3. Dibidang penginderaan jarak jauh *(remote sensing)* untuk mengetahui informasi penting tentang sumber-sumber alam, pencemaran, kerusakan alam di suatu wilayah menggunakan citra satelit.



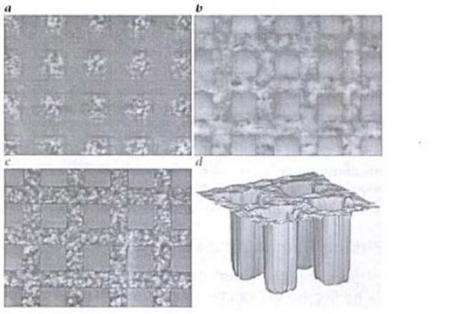
Gambar 2.3. Citra Satelit Daerah Tambang di Puncak Jaya Papua

4. Bidang Visual terdapat Ilmu arsitektur dan desain visual untuk membuat bangunan sebelum malakukan pembangunan yang sesungguhnya. Desain visual akan sangat mempermudah para arsitek dalam memberikan penjelasan rinci terhadap suatu rancangannya.



Gambar 2.4 sebuah model rumah

5. Bidang Volumetrik, Salah satu kemajuan penting dalam dunia Komputer grafis adalah bidang volumetrik yaitu untuk merekonstruksi suatu citra 3 dimensi dari citra 2 dimensi.



Gambar 2.5 pencitraan yang di hasilkan dari 2 dimensi ke 3 dimensi

6. Dibidang-bidang lainnya, seperti bidang fotografi, bidang desain visual, identifikasi objek, dll.

2.2 Pengenalan Wajah Sebagai Kontrol Akses Ruangan

Sistem pengenalan wajah adalah aplikasi pengolahan citra yang dapat mengidentifikasi atau memverifikasi seseorang melalui citra *digital* atau *frame video*. Belakangan ini, sistem pengenalan wajah telah menjadi salah satu aplikasi pengolahan citra yang cukup populer, khususnya di bidang sistem keamanan, verifikasi kartu kredit dan indentifikasi penjahat.

Manusia dapat mengenali wajah seseorang dengan mudah dan cepat, tidak sama halnya dengan proses pengenalan wajah yang dilakukan oleh komputer. Proses pengenalan wajah menggunakan komputer berjalan lambat dan cukup rumit, selain itu terdapat faktor lain yang menambah kesulitan pengenalan wajah seperti orientasi wajah yang berbeda, pencahayaan, potongan rambut, kumis dan objek lain di sekitar wajah.

2.2.1 Pengenalan Wajah

Penelitian tentang pengenalan wajah telah berlangsung dari tahun 1960 hingga sekarang dengan beragan metode yang digunakan. Namun dari beragam

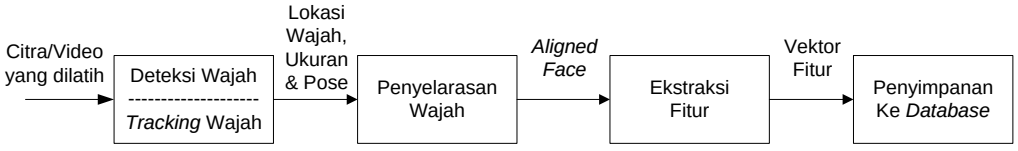
metode tersebut, hanya terdapat dua metode utama yang digunakan untuk pengenalan wajah, yaitu:[3]

- **Pengenalan berbasis fitur**, berdasarkan pada ekstraksi dari sifat-sifat organ individu yang terletak pada wajah seperti mata, hidung dan mulut, serta hubungan mereka satu sama lain.
- **Pengenalan berbasis pendekatan *eigenfaces***, berdasarkan pada ekstraksi informasi paling relevan yang terkandung dalam wajah. Informasi ini berupa suatu set karakteristik citra yang digunakan untuk menggambarkan variasi citra wajah. Tujuan dari proses ekstraksi tersebut adalah untuk mendapatkan *eigenvector* (*eigenface*) dari kovarian matriks distribusi, yang disusun dari kumpulan pelatihan citra wajah. Kemudian, setiap citra wajah diwakili oleh kombinasi linier dari *eigenvector.* Pengenalan dilakukan dengan mengklasifikasikan citra baru dengan citra yang telah disimpan dalam *database*.

Secara umum, meskipun didasari dengan metode-metode yang berbeda proses pengenalan wajah terdiri dari dua proses utama yaitu, proses pelatihan citra dan proses pengenalan citra. Kedua proses ini memiliki tahapan-tahapan yang hampir sama, perbedaannya terletak pada hasil ekstraksi fiturnya. Proses pelatihan citra wajah memiliki tahapan-tahapan sebagai berikut:

- Pendeteksian wajah atau *tracking* wajah.
- Penyelarasan wajah.
- Ekstraksi fitur.
- Penyimpanan fitur ke *database*.

Untuk lebih jelasnya, pada Gambar 2.6 ditunjukan diagram blok sistem pelatihan citra wajah secara umum:

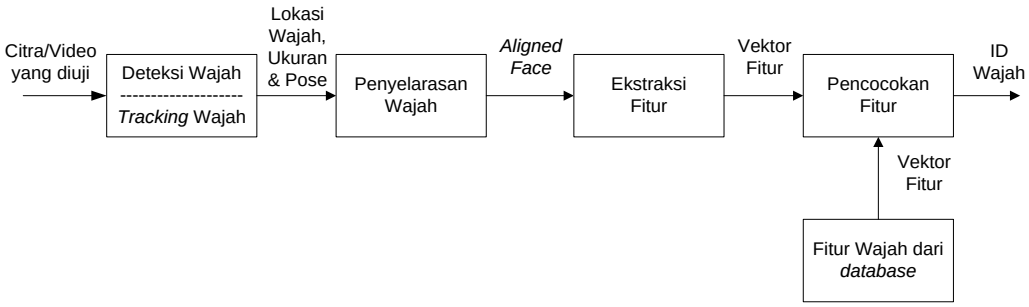


Gambar 2.6. Diagram Blok Proses Pelatihan Citra Wajah

Sedangkan pada proses pengenalan citra wajah, terdiri dari tahapan-tahapan sebagai berikut:

- Pendeteksian wajah atau *tracking* wajah.
- Penyelarasan wajah.
- Ekstraksi fitur.
- Pencocokan fitur yang diuji dengan fitur yang dilatih.

Untuk lebih jelasnya, pada Gambar 2.7 ditunjukan diagram blok sistem pengenalan citra wajah secara umum:

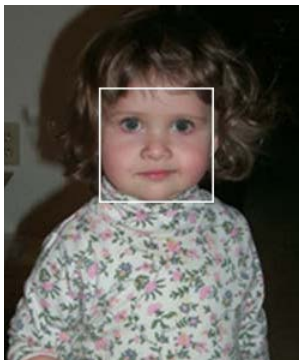


Gambar 2.7. Diagram Blok Proses Pengenalan Citra Wajah

2.2.1.1 Pendeteksian Wajah

Pendeteksian wajah adalah proses yang mensegmentasi area wajah dengan latar belakang dari suatu masukan citra. Proses ini bekerja dengan cara memeriksa citra yang dimasukan, apakah memiliki citra wajah atau tidak, jika memiliki, maka akan dilakukan pemisahan dengan cara memotong citra wajah dari latar belakang citra yang dimasukan. Jika masukan berbentuk *video*, proses yang dilakukan adalah proses *face tracking*. Secara umum, proses *face tracking* dan proses pendeteksian wajah mempunyai fungsi yang sama. Perbedaannya terletak pada proses pendeteksiannya saja, jika pada masukan berbentuk citra, sistem berjalan *offline* sehingga dapat menggunakan proses pendeteksian wajah, sedangkan pada masukan video, sistem berjalan secara *online* atau *real-time* yang membutuhkan pendeteksian secara langsung maka proses yang digunakan adalah proses *face tracking*. Terdapat banyak metode yang dapat digunakan untuk

melakukan proses pendeteksian wajah ini, salah satunya dengan menggunakan metode Viola-Jones.



Gambar 2.8. Proses Deteksi Wajah

2.2.1.2 Penyelarasan Wajah

Pada proses pendeteksian wajah, citra wajah yang didapatkan masih berupa perkiraan kasar atau masih memiliki kualitas yang cukup buruk seperti ukuran yang berbeda dengan ukuran normal, faktor pencahayaan yang kurang atau lebih, kejelasan citra yang buruk dan sebagainya. Maka perlu diadakannya proses penyelarasan.

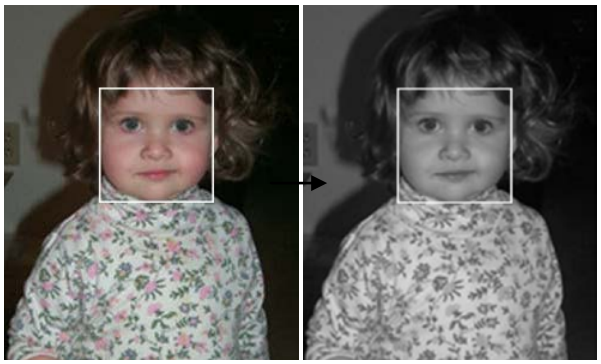
Proses penyelarasan wajah merupakan proses yang bertujuan untuk menormalisasi wajah dari citra wajah yang didapatkan dari proses pendeteksian wajah. Proses ini terdiri dari tahapan-tahapan sebagai berikut:

a. Grayscaling

Grayscaling citra merupakan tahapan pertama dari proses penyelarasan, pada tahap ini terjadi pengkonversian citra warna RGB menjadi citra berwarna abu. Citra warna RGB terdiri dari 3 parameter warna yaitu merah (*red*), hijau (*green*) dan biru (*blue*), jika citra warna RGB ini dimasukan ke dalam proses ekstraksi, maka proses tersebut akan sulit untuk dilakukan karena citra RGB terdiri dari 3 parameter, oleh karena itu diperlukan penyamaan parameter yaitu dengan melakukan tahap *grayscaling* ini. Berikut ini adalah persamaan tahap *grayscaling* citra[13]:

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

Dimana, nilai citra RGB adalah (r, g, b) dengan nilai *integer* diantara 0 sampai 255, dan x adalah nilai *grayscale*.



Gambar 2.9. Grayscaling Citra

b. Pemotongan

Pada tahapan ini terjadi pemotongan citra yang memisahkan citra wajah dengan citra masukannya, tujuannya untuk mengambil citra yang hanya diperlukan untuk proses ekstraksi, dalam hal ini adalah citra wajah dan membuang citra lain yang tidak diperlukan. Dimensi citra yang dipotong disesuaikan dengan dimensi dari proses segmentasi atau pengkotakan objek wajah yang dilakukan pada proses pendeteksian wajah.



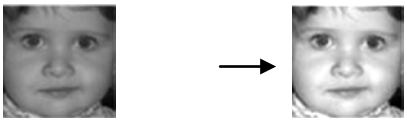
Gambar 2.10. Pemotongan Citra

c. Resizing (tahap normalisasi dimensi citra)

Pada tahap resizing citra, terjadi proses normalisasi dimensi citra wajah, yaitu proses pembesaran atau pengecilan dimensi citra wajah menjadi dimensi yang telah ditentukan. Tujuannya, untuk menyamakan dimensi wajah dari tiap citra yang dimasukan, sehingga pada proses ekstraksi citra nanti tidak ada perbedaaan dimensi dari matriks data citra wajah.

d. Equalizing (tahap koreksi tingkat kecerahan citra)

Tahap ini adalah tahapan terakhir dari proses penyelarasan, yang tujuannya untuk memperjelas nilai histogram dari citra wajah hasil tahapan-tahapan sebelumnya.



Gambar 2.11. Equalizing Citra

2.2.1.3 Ekstrasi Wajah

Ekstraksi fitur merupakan proses yang berfungsi untuk mendapatkan informasi yang efektif dan berguna untuk membedakan wajah-wajah orang yang berbeda dari citra wajah yang telah diselaraskan. Proses ini dilakukan menggunakan algoritma-algoritma ekstraksi seperti *principal component analysis* (PCA), *linear discriminant analysis* (LDA), *independent component analysis* (ICA) dan sebagainya. Informasi yang didapatkan dari ekstraksi fitur disebut vektor fitur, yaitu bentuk dasar pencarian citra berbasis konten, yang menangkap properti citra seperti warna dan tekstur.

2.2.1.4 Penyimpanan Fitur Wajah

Proses penyimpanan fitur, merupakan tahapan terakhir dari proses pelatihan citra wajah. Proses ini bertugas untuk menyimpan vektor fitur hasil ekstraksi citra wajah beserta identitas vektor fitur tersebut kedalam *database*, yang nantinya akan digunakan sebagai pembanding citra wajah yang diuji pada proses pengenalan wajah.

2.2.1.5 Pencocokan Fitur Wajah

Pencocokan fitur adalah tahapan proses yang hanya terdapat pada proses pengenalan citra wajah. Proses ini merupakan proses perbandingan fitur yang

telah diekstrak dari citra uji dengan fitur citra wajah dari *database*, yang sebelumnya telah melalui proses pelatihan citra. Proses perbandingan ini dilakukan menggunakan metode pengenalan pola, salah satunya adalah metode *nearest neighbours*.

Dari proses perbandingan fitur tersebut akan menghasilkan nilai jarak terdekat yang menandakan nilai fitur citra uji hampir menyamai dengan fitur citra latih. Nilai jarak ini akan menjadi nilai masukan untuk nilai kemiripan citra.

Nilai kemiripan citra merupakan nilai tingkat kemiripan citra uji dengan citra latih, semakin besar nilainya menandakan bahwa orang yang sedang diamati adalah orang yang sama dengan orang yang citra wajahnya telah disimpan dalam *database*. Nilai kemiripan ini sebelum mengeluarkan *output* hasil pengenalan wajah akan melalui proses *threshold* terlebih dahulu.

Proses *threshold* pengenalan wajah adalah proses penyaringan nilai kemiripan citra, dimana jika nilai kemiripan berada dibawah *threshold* yang telah ditentukan maka *output* hasil pengenalan tidak akan ditampilkan. Jika, berada diatas *output* hasil pengenalan akan ditampilkan. Tujuannya agar *output* hasil pengenalan yang ditampilkan bernilai benar, karena memiliki nilai kemiripan yang tinggi.

Setelah, proses-proses tersebut dilakukan proses terakhir adalah proses pencarian tingkat keakuratan. Keakuratan adalah kondisi yang menunjukan kebenaran atau ketepatan suatu objek yang diamati, dalam sistem pengenalan ini adalah kebenaran bahwa citra wajah yang diujikan benar-benar citra wajah orang yang dimaksud.

2.2.1.6 Pendeteksian Mata

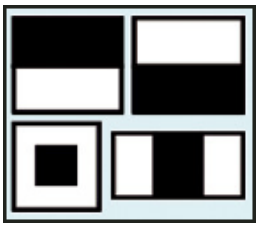
Proses pendeteksian mata secara umum sama dengan proses pendektesian wajah. Perbedaanya pada proses ini objek yang diamati hanya mencakup pada area mata saja di luar objek tersebut akan diabaikan. Karena, metoda yang digunakan sama dengan metoda yang ada dalam proses pendeteksian wajah. Citra mata yang dihasilkan pada proses ini akan mengalami proses penyelarasan citra.

2.2.1.7 Metode Viola-Jones

Metode Viola-Jones merupakan salah satu metode pendeteksian objek yang cukup populer, yang dapat memberikan hasil dengan tingkat keakuratan yang cukup tinggi dan dengan kecepatan yang sangat tinggi. Metode ini, diusulkan oleh Paul Viola dan Michael Jones pada tahun 2001, dengan menggabungkan empat kunci utama untuk mendeteksi suatu objek, yaitu[8]:

- Fitur persegi sederhana, disebut fitur *Haar*
- *Integral image* untuk pendeteksian fitur dengan cepat
- Metoda AdaBoost *machine-learning*
- *Cascade classifier* untuk mengkombinasikan banyak fitur

Fitur yang digunakan dalam metode Viola-Jones adalah fitur *Haar* yang didasari oleh *Haar wavelets. Haar wavelets* adalah satu gelombang panjang berupa gelombang persegi (satu interval tinggi satu dan satu interval rendah). Dalam dua dimensi, gelombang persegi ini digambarkan dengan sepasang persegi yang berdekatan – satu terang dan satu gelap.

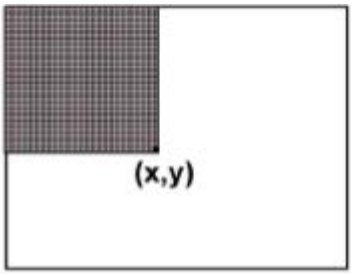


Gambar 2.12. Contoh Fitur Haar

Ada atau tidaknya fitur *Haar* dalam suatu citra ditentukan dengan cara mengurangi nilai rata-rata piksel daerah gelap dengan nilai rata-rata piksel daerah terang. Jika hasilnya diatas ambang maka fitur tersebut dikatakan ada.

Untuk menentukan ada atau tidaknya ratusan fitur *Haar* pada suatu citra secara efektif, Viola-Jones menggunakan teknik yang disebut *integral image*. *Integral image* merupakan teknik yang dapat mempercepat proses pendeteksian suatu objek dengan menggabungkan unit terkecil dari citra yaitu nilai-nilai piksel

menjadi suatu representasi citra baru, caranya dengan menjumlahkan piksel yang ada di sebelah kiri dan atas titik tersebut.



Gambar 2.13. Nilai Piksel (*x, y*) Adalah Nilai Semua Piksel Daerah Arsir

Sebagai contoh terdapat citra masukan dengan dimensi *5x5* dengan nilai-nilai piksel sebagai berikut:

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* dari citra input diatas adalah:

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

Nilai piksel pada *integral image* ini didapatkan dengan cara menjumlahkan nilai suatu piksel dengan nilai piksel disebelah 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* dapat didapatkan dengan waktu yang cepat, berikut ini adalah contoh perbandingan pencarian nilai fitur dengan menggunakan *integral image* dan tanpa *integral image*:

- Tanpa *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

Nilai fitur Haar = | (total piksel hitam) – (total piksel putih) |

= | (jumlah nilai piksel baris-2 kolom-2 sampai nilai piksel baris-5 kolom-3) – (jumlah nilai piksel baris-4 kolom-2 sampai nilai piksel baris-5 kolom-5) |

= | (5+9+6+8+5+6+4+5) – (7+7+5+6+6+7+3+6) |
= | 48 - 47 |
= 1

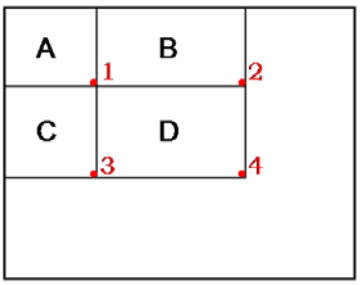
- Dengan *integral image*

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

Untuk menghitung fitur dengan menggunakan *integral image*, terdapat rumusan sebagai berikut:

D = D + A - (B + C) (2.2)

Dimana, D adalah nilai piksel kanan bawah, A adalah nilai piksel kiri atas, B adalah piksel atas dari piksel D dan C adalah nilai piksel kiri dari piksel D. Berikut adalah gambaran dari piksel-piksel tersebut:

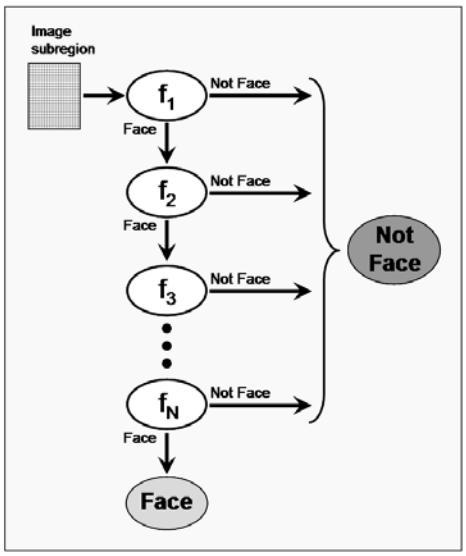


Gambar 2.14.Pencarian Nilai Piksel D Pada Integral Image

Nilai fitur Haar = | (total piksel hitam) – (total piksel putih) |
= | (73+2-(14 + 13)) – (133+13-(73+26)) |
= | 48 - 47 |
= 1

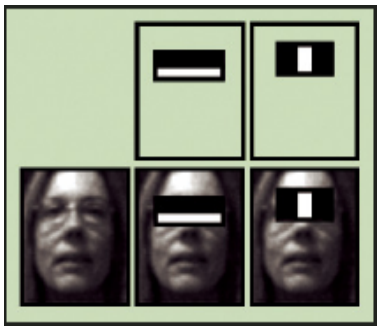
Selanjutnya, setelah nilai fitur didapatkan Viola-Jones menggunakan metode AdaBoost *machine-learning* untuk mengetahui suatu fitur apakah merepresentasikan ada tidaknya wajah dalam suatu citra masukan. Metode AdaBoost menggabungkan banyak *classifier* lemah menjadi satu *classifier* kuat. *Classifier* adalah suatu ciri yang menandakan adanya objek wajah dalam suatu citra. Sedangkan *classifier* lemah adalah suatu jawaban benar namun memiliki tingkat kebenaran yang kurang akurat, jika digabung maka *classifier* lemah tersebut akan menghasilkan suatu *classifier* kuat. Berikut ini adalah gambaran dari metode AdaBoost yang digunakan untuk mendeteksi wajah:

Gambar 2.15, menjelaskan bagaimana metode AdaBoost ini akan menyeleksi nilai fitur dari citra masukan, jika pada fitur tersebut tidak terdeteksi adanya wajah maka akan disimpulkan tidak ada wajah, jika ada akan diteruskan ke tingkat selanjutnya sampai ke tingkat terakhir, lalu akan disimpulkan pada citra masukan terdeteksi adanya wajah. Proses penyeleksian fitur tersebut dengan cara membandingkan nilai fitur dengan nilai ambang yang telah ditentukan pada tiap tingkat, jika nilai fitur sama atau diatas nilai ambang maka terdeteksi adanya wajah dalam fitur tersebut.



Gambar 2.15. Metode AdaBoost Machine-Learning

Tingkatan proses penyeleksian fitur diurutkan berdasarkan bobot terberat sampai teringan, sehingga dapat secepat mungkin untuk mengklasifikasikan bahwa tidak terdeteksi wajah dalam citra masukan, pengurutan proses penyeleksian fitur ini disebut *cascade classifier*.



Gambar 2.16. Fitur Tingkat Pertama Pada Metode Viola-Jones

2.2.1.8 Principal Component Analysis (PCA)

merupakan teknik linier untuk memproyeksikan data vektor yang berdimensi tinggi ke vektor yang mempunyai dimensi rendah[12]. PCA ditemukan

pada tahun 1901 oleh Karl Pearson, dan telah mengalami pengembangan hingga terakhir oleh Karhunen-Louve pada tahun 1963.

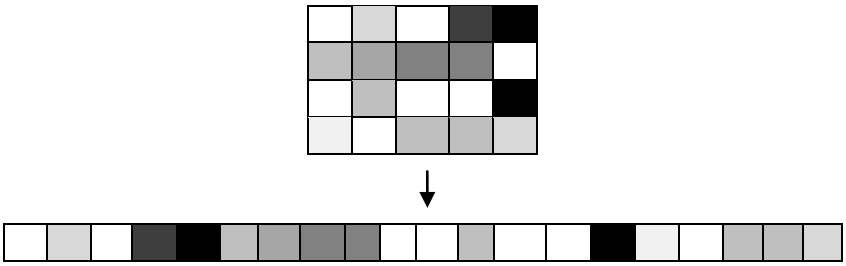
PCA banyak digunakan sebagai alat eksplorasi dalam analisis data dan untuk membuat model prediksi. Sedangkan, pada sistem pengenalan wajah, PCA digunakan untuk keperluan ekstraksi fitur citra, dimana jumlah dimensi dari citra jauh lebih besar dibandingkan dengan jumlah data sampel yang digunakan. Misalnya diketahui suatu citra dengan ukuran hxw, dengan h adalah tinggi citra dan w adalah lebar citra, maka dimensi citra adalah n dengan n=hxw. Jika suatu citra dengan ukuran dimensi n, maka jumlah kombinasi linier dari citra sebanyak n merupakan dimensi yang tinggi, dan ini merupakan masalah besar pada komputasi ketika proses pengukuran. Dengan menggunakan PCA, dimensi yang tinggi tersebut dapat direduksi menjadi dimensi yang rendah. Jumlah dimensi yang dihasilkan oleh PCA tergantung pada jumlah data yang digunakan oleh pelatihan dan jumlah sampel pada masing-masing data pelatihan. Misalkan jumlah data pelatihan adalah k dan masing-masing data mempunyai s model, maka jumlah sampel keseluruhan adalah m dengan m=kxs.

Sebagai contoh, misalnya terdapat data sampel 50 orang dengan 4 pose citra yang digunakan untuk pelatihan, maka m = 50x4 = 200. Jika tinggi citra 150 *pixel* dan lebar 110 *pixel*, maka n = 150x110 = 165.000 *pixel*. Pada kasus ini, metode ekstraksi fitur menggunakan PCA sangat cocok untuk digunakan karena nilai m<<n (200<<165.000). Pengurangan dimensi yang sangat signifikan ini akan sangat membantu untuk mempercepat waktu komputasinya dalam melakukan klasifikasi.

Data pelatihan mxn dapat dituliskan dengan bentuk matriks yang persamaannya sebagai berikut:

$$x = \begin{bmatrix} x_{1,1}x_{1,2}\, x_{1,3} \ldots & x_{1,n} \\ x_{2,1}x_{2,2}x_{2,3} & x_{2,n} \\ x_{3,1}\, x_{3,2}x_{3,3} & x_{3,n} \\ \ldots & \ldots \\ x_{m,1}x_{m,2}x_{m,3} \ldots & x_{m,n} \end{bmatrix} \tag{2.3}$$

Jika n>>m dan n merupakan dimensi citra, m adalah jumlah citra dilatih. Perlu dicatat, n merupakan vektor baris hasil bentukan hxw dengan h sebagai tinggi citra dan w merupakan lebar citra. Setiap citra yang digunakan harus dibentuk menjadi matriks baris atau matriks kolom. Sebagai ilustrasi dapat dilihat pada gambar berikut:



Gambar 2.17. Representasi Perubahan Dimensi dari 4x5 Menjadi 1x20

2.2.1.9 Ekstraksi Fitur Menggunakan PCA

Proses ekstraksi fitur bukanlah proses yang singkat, proses ini terdiri dari banyak tahapan, berikut ini adalah bentuk algoritma ekstraksi fitur menggunakan PCA:

a. Pembentukan matriks data citra wajah

Hal pertama yang harus dilakukan adalah pembentukan matriks yang datanya diambil dari *pixel* setiap citra. Berdasarkan contoh yang telah dijelaskan diatas yaitu, mxn dimana m adalah jumlah citra yang dilatih dan n merupakan dimensi dari citra tersebut, lalu diekstraksi menjadi citra dengan dimensi yang lebih kecil, hasilnya diproyeksikan menjadi sebuah matriks seperti pada persamaan (2.3).

b. Pencarian rata-rata seluruh citra

Setelah matriks data citra wajah terbentuk, maka proses berikutnya adalah proses perhitungan untuk mencari rata-rata hasil seluruh citra. Pencarian nilai rata-rata ini tujuannya untuk mengetahui *noise* atau persamaan tiap vector yang dapat menganggu keakuratan perhitungan pada PCA, yang dapat dihitung dengan menggunakan rumus:

$$\tau_i = \frac{(x_{1,i} + x_{2,i} + x_{3,i} + \ldots + x_{m,i})}{m} \qquad (2.4)$$
$$= \frac{\sum_{j=1}^{m} x_{j,i}}{m} = [\tau_1, \tau_2, \ldots\ldots\ldots, \tau_n]$$

Terlihat bahwa hasil perhitungan persamaan diatas adalah merupakan sebuah *array* yang memuat nilai n, $(\tau_1, \tau_2, \ldots\ldots\ldots, \tau_n)$. *Array* tersebut nantinya akan digunakan untuk mencari nilai rata-rata nol (*zero mean*), setelah dibentuk menjadi matriks 1xn terlebih dahulu. Namun, ukuran matriks nilai rata-rata tidak berkesesuaian dengan matriks data citra sampel yang memiliki ukuran mxn, maka perlu dilakukan penggandaan nilai rata-rata.

c. Penggandaan nilai rata-rata.

Pengandaan nilai rata-rata dilakukan untuk menyamakan dimensi nilai rata-rata dengan dimensi data citra sampel sebanyak m, sehingga rata-rata seluruh citra data sampel memiliki dimensi (mxn). Matriks rata-rata citra data sampel yang telah digandakan sebanyak m kali dapat ditulis menggunakan persamaan:

$$\mu = a\ x\ \tau \qquad (2.5)$$

$$\mu = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ \ldots \\ a_{m,1} \end{bmatrix} x \begin{bmatrix} \tau_{1,1}\ \tau_{1,2} \ldots\ \tau_{1,n} \end{bmatrix} = \begin{bmatrix} \mu_{1,1}\ \mu_{1,2} \ldots\ \mu_{1,n} \\ \mu_{2,1}\ \mu_{2,2} \ldots\ \mu_{2,n} \\ \ldots \quad \ldots \quad \ldots \quad \ldots \\ \mu_{m,1}\mu_{m,2} \ldots\ \mu_{m,n} \end{bmatrix}$$

a adalah matriks kolom dengan dimensi mx1 dan a untuk keseluruhan nilai adalah bernilai 1. Nilai dari $[\mu_{i,1}, \mu_{i,2}, \ldots\ldots\ldots, \mu_{i,n}]$ pada baris ke i = nilai baris ke i+1 dan berlaku $\forall j, j \in 1, 2, 3, \ldots, m-1$, maka hasil persamaan diatas dapat digunakan untuk menghitung rata-rata nol.

d. Perhitungan nilai rata-rata nol

Perhitungan nilai rata-rata nol, berfungsi untuk menghilangkan *noise* yang dapat menganggu keakuratan pada perhitungan PCA. Perhitungan ini dapat dimodelkan menggunakan persamaan:

$$\phi = x - \mu \qquad (2.6)$$

Atau dapat dimodelkan menggunakan matriks seperti pada persamaan berikut:

$$\phi = \begin{bmatrix} x_{1,1}x_{1,2} \dots x_{1,n} \\ x_{2,1}x_{2,2} \dots x_{2,n} \\ \dots \quad \dots \quad \dots \quad \dots \\ x_{m,1}x_{m,2} \dots x_{m,n} \end{bmatrix} - \begin{bmatrix} \mu_{1,1}\,\mu_{1,2} \dots \mu_{1,n} \\ \mu_{2,1}\,\mu_{2,2} \dots \mu_{2,n} \\ \dots \quad \dots \quad \dots \quad \dots \\ \mu_{m,1}\mu_{m,2} \dots \mu_{m,n} \end{bmatrix} \tag{2.7}$$

$$\phi = \begin{bmatrix} x_{1,1} - \mu_{1,1}\; x_{1,2} - \mu_{1,2} \quad \dots \dots \dots \; x_{1,n} - \mu_{1,n} \\ x_{2,1} - \mu_{2,1}\; x_{2,2} - \mu_{2,2} \quad \dots \dots \dots \; x_{2,n} - \mu_{2,n} \\ \dots \dots \dots \quad \dots \dots \dots \quad \dots \dots \dots \quad \dots \dots \dots \\ x_{m,1} - \mu_{m,1}\; x_{m,2} - \mu_{m,2} \; \dots \dots \dots \; x_{m,n} - \mu_{m,n} \end{bmatrix}$$

$$\phi = \begin{bmatrix} x_{1,1} - \mu_{1,1}\; x_{1,2} - \mu_{1,2} \quad \dots \dots \dots \; x_{1,n} - \mu_{1,n} \\ x_{2,1} - \mu_{1,1}\; x_{2,2} - \mu_{1,2} \quad \dots \dots \dots \; x_{2,n} - \mu_{1,n} \\ \dots \dots \dots \quad \dots \dots \dots \quad \dots \dots \dots \quad \dots \dots \dots \\ x_{m,1} - \mu_{1,1}\; x_{m,2} - \mu_{1,2} \; \dots \dots \dots \; x_{m,n} - \mu_{1,n} \end{bmatrix}$$

$$\phi = \begin{bmatrix} \phi_{1,1}\phi_{1,2} \; \dots \; \phi_{1,n} \\ \phi_{2,1}\phi_{2,2} \; \dots \; \phi_{2,n} \\ \dots \quad \dots \quad \dots \quad \dots \\ \phi_{m,1}\phi_{m,2} \dots \phi_{m,n} \end{bmatrix}$$

e. Pembentukan matrik kovarian

Hasil perhitungan nilai rata-rata nol digunakan untuk mendapatkan nilai matriks kovarian. Matriks kovarian merupakan matriks reduksi dari proses ekstraksi PCA, yang memiliki dimensi lebih kecil dibandingkan dimensi matriks citra asli. Berikut ini adalah persamaan matriks kovarian:

$$C = \frac{1}{m-1}\phi_{j,i} * \phi^{T}{}_{j,i} \tag{2.8}$$

Persamaan 2.6 dapat diuraikan menggunakan matriks seperti pada persamaan:

$$C = \frac{1}{m-1}\begin{bmatrix} \phi_{1,1}\phi_{1,2} \; \dots \; \phi_{1,n} \\ \phi_{2,1}\phi_{2,2} \; \dots \; \phi_{2,n} \\ \dots \quad \dots \quad \dots \quad \dots \\ \phi_{m,1}\phi_{m,2} \dots \phi_{m,n} \end{bmatrix} * \begin{bmatrix} \phi_{1,1}\phi_{1,2} \; \dots \; \phi_{1,n} \\ \phi_{2,1}\phi_{2,2} \; \dots \; \phi_{2,n} \\ \dots \quad \dots \quad \dots \quad \dots \\ \phi_{m,1}\phi_{m,2} \dots \phi_{m,n} \end{bmatrix}^{T} \tag{2.9}$$

$$C = \frac{1}{m-1}\begin{bmatrix} c(x_1,x_1)\ c(x_1,x_2) & \dots\dots\dots & c(x_1,x_m) \\ c(x_2,x_1)\ c(x_2,x_2) & \dots\dots\dots & c(x_2,x_m) \\ \dots\dots\dots & \dots\dots\dots & \dots\dots\dots \\ c(x_m,x_1)\ c(x_m,x_2) & \dots\dots\dots & c(x_m,x_m) \end{bmatrix}$$

f. Penentuan matriks *eigenvalue* dan *eigenvector*

Setelah matriks kovarian didapatkan maka langkah selanjutnya adalah menentukan matriks *eigenvalue* dan matriks *eigenvector*.

Matriks *eigen* dapat dicari menggunakan persamaan:

$$C - Z = |C - \lambda I| \qquad (2.10)$$
$$|C - \lambda I| = 0$$

Dimana *C* adalah matriks kovarian. *Z* adalah matriks *eigenvalue* dengan λ sebagai *scalar* pembentuknya dan *I* sebagai matriks identitas, persamaan tersebut jika diimplementasikan ke bentuk matriks menjadi persamaan sebagai berikut:

$$\begin{bmatrix} C_{11}C_{12} & \dots & C_{1,n} \\ C_{21}C_{2,2} & \dots & C_{2,n} \\ \dots & \dots & \dots \\ C_{m,1}C_{m,2} & \dots & C_{m,n} \end{bmatrix} - \lambda \begin{bmatrix} 1\,0 & \dots & 0 \\ 0\,1 & \dots & 0 \\ \dots\dots & \dots & \dots \\ 0\,0 & \dots & 1 \end{bmatrix} = 0 \qquad (2.11)$$

$$\begin{bmatrix} C_{11}C_{12} & \dots & C_{1,n} \\ C_{21}C_{2,2} & \dots & C_{2,n} \\ \dots & \dots & \dots \\ C_{m,1}C_{m,2} & \dots & C_{m,n} \end{bmatrix} - \begin{bmatrix} \lambda\,0 & \dots & 0 \\ 0\,\lambda & \dots & 0 \\ \dots\dots & \dots & \dots \\ 0\,0 & \dots & \lambda \end{bmatrix} = 0$$

$$\begin{bmatrix} C_{11}-\lambda & C_{12} & \dots & C_{1,n} \\ C_{21} & C_{2,2}-\lambda & \dots & C_{2,n} \\ \dots & \dots & \dots & \dots \\ C_{m,1} & C_{m,2} & \dots & C_{m,n}-\lambda \end{bmatrix} = 0$$

Hasil dari matriks tersebut akan menghasilkan suatu persamaan, sebagai berikut:

$$\lambda^m + C\lambda^{m-1} + \dots + C = 0 \qquad (2.12)$$

Setelah itu dilakukan proses faktorisasi, maka akan didapatkan nilai $(\lambda_1, \lambda_2, \dots, \lambda_{i+1})$ yang nantinya akan digunakan untuk mencari nilai *eigenvector*, dengan persamaan:

$$(C - \lambda_i I)\mathrm{X_i} = 0 \qquad (2.13)$$

X_i merupakan matriks kolom dengan elemen (x_1, x_2, …, x_i) didalamnya, dan matriks inilah yang disebut *eigenvector*. Persamaan (2.13) dapat ditulis menjadi bentuk matriks sebagai berikut:

$$\begin{bmatrix} C_{11}-\lambda & C_{12} & \dots & C_{1,n} \\ C_{21} & C_{2,2}-\lambda & \dots & C_{2,n} \\ \dots & \dots & \dots & \dots \\ C_{m,1} & C_{m,2} & \dots & C_{m,n}-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = 0 \qquad (2.14)$$

Kedua matriks tersebut dikalikan hingga didapatkan sebuah persamaan sebagai berikut:

$$((C_{11} - \lambda_1)x_1 + C_{12}x_2 + \dots + C_{mn}x_n) = 0 \; ; dst. \qquad (2.15)$$

Dari persamaan (2.14) didapatkan nilai (x_1, x_2, …, x_{n+1}) yang nantinya diproyeksikan kedalam suatu matriks baris atau kolom, yang disebut matriks *eigenvector*.

g. Pencarian *eigenface* untuk proses pengenalan wajah

Eigenface didapatkan dari hasil perkalian matriks X atau *eigenvector* dengan matriks ϕ yang merupakan matriks hasil perhitungan rata-rata nol pada persamaan (2.7), yang dibagi oleh akar Z, sebagai *eigenvalue*.

$$Eigenface = \frac{1}{\sqrt{Z}} * X * \phi \qquad (2.16)$$

2.2.1.10 Pencocokan Fitur Menggunakan Metode Nearest Neighbours

Nearest neighbors adalah pendekatan untuk mencari kasus dengan menghitung kedekatan antara kasus baru dengan kasus lama, yaitu berdasarkan pada pencocokan bobot dari sejumlah fitur yang ada. Terdapat banyak metode pengukuran jarak pada *nearest neighbor* ini, salah satunya *euclidean distance* yang akan digunakan pada proses pengenalan wajah ini. Berikut persamaan dari metode *euclidean distance:*

$$d_v = \sqrt{\sum_{j=1}^{m} (Eigenface_{train\ ke\ j} - Eigenface_{Uji})^2} \quad (2.17)$$

Diketahui, $Eigenface_{train\ ke\ j}$ merupakan matriks fitur yang dihasilkan oleh pelatihan ke j, dengan $j \epsilon 1..m$ dan m=k*s. $Eigenface_{Uji}$ merupakan matriks fitur citra yang diuji. Hasil pengukuran kemiripan yang didapatkan merupakan nilai minimal dari d_v atau ditulis dengan persamaan:

$$d = min\ (d_v) \quad (2.18)$$

Pada tahap resizing citra, terjadi proses normalisasi dimensi citra wajah, yaitu proses pembesaran atau pengecilan dimensi citra wajah menjadi dimensi yang telah ditentukan. Tujuannya, untuk menyamakan dimensi wajah dari tiap citra yang dimasukan, sehingga pada proses ekstraksi citra nanti tidak ada perbedaaan dimensi dari matriks data citra wajah.

2.3 Perangkat keras

Pada sub bab ini menjelaskan beberapa perangkata keras yang akan di gunakan seperti : webcam, mikrokontroler, EMLock600

2.3.1 Webcam

Webcam atau kamera web, pada dasarnya adalah sebuah kamera digital yang terhubung ke komputer, yang berfungsi untuk mengambil citra yang akan diolah oleh komputer. Pada awalnya *webcam* digunakan sebagai alat komunikasi yang menampilkan rentetan citra dan dapat diakses melalui *world wide web*. Namun, seiring perkembangannya *webcam* digunakan juga untuk keperluan lainnya.



Gambar 2.18. Webcam Logitech C270

Webcam Logitech C270 merupakan *webcam* yang memiliki kemampuan yang cukup memenuhi syarat untuk digunakan pada sistem pengenalan wajah secara *realtime*, karena:

a. Memiliki tingkat kecerahan dan ketajaman gambar yang cukup baik.
b. Memiliki kecepatan *frame rate* (60 fps) yang tinggi.
c. Memiliki kualitas *video* (1280x720pixel) yang baik.

2.3.2 Mikrokontroler

Mikrokontroler adalah sebuah komputer kecil dalam sebuah rangkaian terpadu yang berisi processor core, memory dan programmable input/output peripherals. Dalam Tugas Akhir ini digunakan mikrokontorler keluarga ATMEL, yakni ATmega8535.



Gambar 2.19 ATmega8535

Beberapa fitur ATmega8535, yakni:

1. Tegangan operasi: 4.5V – 5.5V
2. 8 Kbytes of In-System Self-Programmable Flash program memory
3. 512 Bytes EEPROM
4. 512 Bytes SRAM

5. Dual 8-bit Timer/Counter with Separate Prescalers and Compare Modes
6. Dual 16-bit Timer/Counter with Separate Prescalers, Compare Mode, and Capture Mode
7. 8-channel, 10-bit ADC (8 Single-ended Channels)
8. Programmable serial USARTs

Mikrokontroler ini memiliki 4 port dan yang di gunakan pada tuga akhir ini adalah port C. port c merupakan 8-bit bidirectional I/O port. Setiap pinnya dapat menyediakan internal pull-up resistor (dapat di atur per bit). Output buffer port C dapat memberi arus 20 mA. Data direction register port C (DDRC) harus di setting terlebih dahulu sebelum Port C digunakan. Bit-bit DDRC diisi 0 jika ingin mengfungsikan pin-pin port C yang bersesuaian sebagai input, atau diisi 1 jika sebagai output. Selain itu, dua pin port C (PC6 dan PC7) juga memiliki fungsi alternative lain sebagai oscillator untuk timer / counter 2.

2.3.3 Kunci Magnet

Kunci Magnet yang akan digunakan adalah EMLock 600. EMLock 600 merupakan sebuah kunci *magnetic* yang mampu menahan beban sampai 600lbs (272.16 Kg).



Gambar 2.20 EMLock 600

alat ini dapat bekerja pada 12volt 480mA dan 24volt 240mA dengan mensetting pin seperti pada gambar 2.19.



Gambar 2.21 setting pin

Alat ini bekerja menggunakan tegangan bila tegangan terputus alat ini tidak akan bekerja.

2.4 Perangkat Lunak

Pada sub bab ini akan menjelaskan tentang beberapa perangkat lunak yang akakn digunakan seperti : C#, Visual studio, CV AVR, *Emgucv*, *haarcascade frontal face* dan *haarcascade eye*.

2.4.1 C#

C Sharp (sering disingkat sebagai C#) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic.

Contoh program C Sharp(C#) sederhana:

```
serialPort1.PortName = "com8";
serialPort1.BaudRate = 9600;
serialPort1.DataBits = 8;
serialPort1.Open();
serialPort1.Close();
```

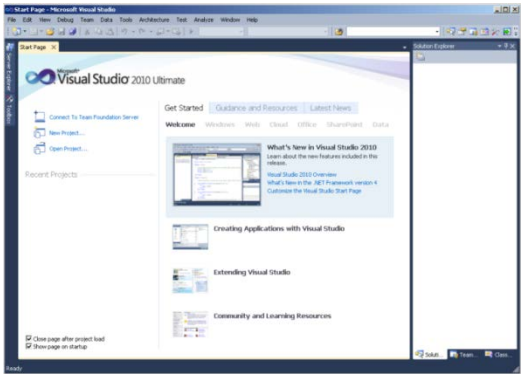
keterangan:

- serialPort1.portname = “com8”
  pernyataan ini berfungsi untuk pemberian nama pada port yang akan di pakai pada perintah ini port yang di pakai ada com8
- serialPort1.baudrate = 9600
  pada pernyataan ini berfungsi sebagai pengatur baudrate yang di gunakan pada perintah ini baud rate di set pada nilai 9600
- serialport1.open
  pada pernyataan ini berfungsi sebagai pengatur bahwa port siap di jalankan
- serialPort1.Close();

pada pernyataan ini berfungsi sebagai pengatur bahwa port yang di jalan kan akan di tutup atau di matikan

2.4.2 Visual Studio 2010

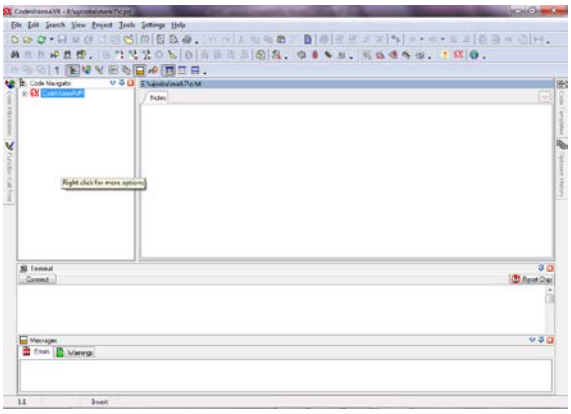
Visual Studio 2010 Ultimate merupakan IDE (*intergrated development Environtment*) atau lingkungan program yang sudah dikemas sebagai program aplikasi yang biasanya terdiri dari editor kode, *compiler, debugger,* dan antar muka grafis (GUI) builder. Visual studio juga dapat menggunakan beberapa bahasa seperti *Visual Basic, Visual C++, Visual Web Deploper, Visual C# Dan Visual F#.*



Gambar 2.22 Tampilan Visual Studio 2010

2.4.3 CV AVR

CV AVR adalah Integrated Development Environment (IDE) untuk menulis dan debugging aplikasi AVR pada Windows. CV AVR juga menyediakan alat manajemen, sumber Editor file proyek, simulator, assembler dan front-end untuk C / C + +, pemrograman, emulasi dan debugging on-chip.



Gambar 2.23 Tampilan CV AVR

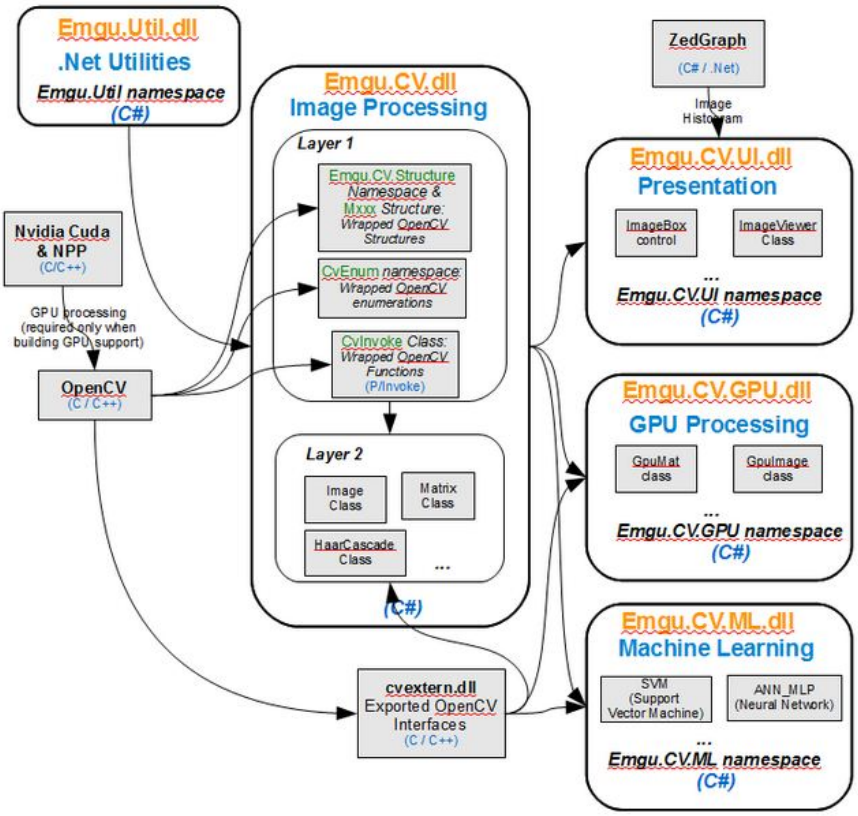
CodeVision AVR mempunyai suatu keunggulan dari compiler lain, yaitu adanya codewizard, fasilitas ini memudahkan kita dalam inisialisasi mikrokontroler yang akan kita gunakan.

2.4.4 EmguCV

EmguCV adalah library lintas platform yang membiarkan fungsi OpenCV untuk di panggil atau digunakan dalam bahasa C#, VB, VC++ dan lainnya. Dan Emgu ini dapat di gunakan secara bebas.

Emgu CV terdiri dari 2 layer, yaitu basic layer dan second layer. Basic layer mengandung fungsi, struktur, dan enumerasi yang secara langsung merefleksikan apa yang ada di OpenCV. Dengan adanya layer inilah kita bisa memanggil fungsi-fungsi pada OpenCV dengan bahasa pemrograman C#. Sedangkan second layer mengandung kelas-kelas yang memanfaatkan keunggulan teknologi .NET.

Konsep kedua yang perlu dipahami yaitu mengenai code mapping, yaitu bagaimana kode-kode dalam OpenCV dipetakan ke dalam Emgu CV. Ada 3 tipe mapping kode, yaitu function mapping, structure mapping, dan enumeration mapping.



Gambar 2.24 Arsitektur EmguCV

Keterangan:

- Emgu.CV

  Pada komponen ini proses *image processing* dilakukan

- Emgu.CV.UI

  Pada komponen ini proses tampilan pada *image box* dilakukan

- Emgu.CV.Structure

  Komponen ini adalah bagian dari Emgu.CV yang bertugas memanggil struktur dari opencv

- Emgu.CV.CvEnum

  Komponen ini yang menerjemahkan bahasa c/c++ ke bahasa C#

2.4.5 Haarcascade FrontalFace

HaarCascade Frontal Face adalah suatu *library* berupa *file* templet berekstensi xml, yang diciptakan oleh Rainer Leinhart pada tahun 2000. *Library*

ini mengadopsi metode Viola-Jones yang memiliki fungsi untuk mendeteksi objek berupa wajah dengan posisi tegak lurus ke depan[4].

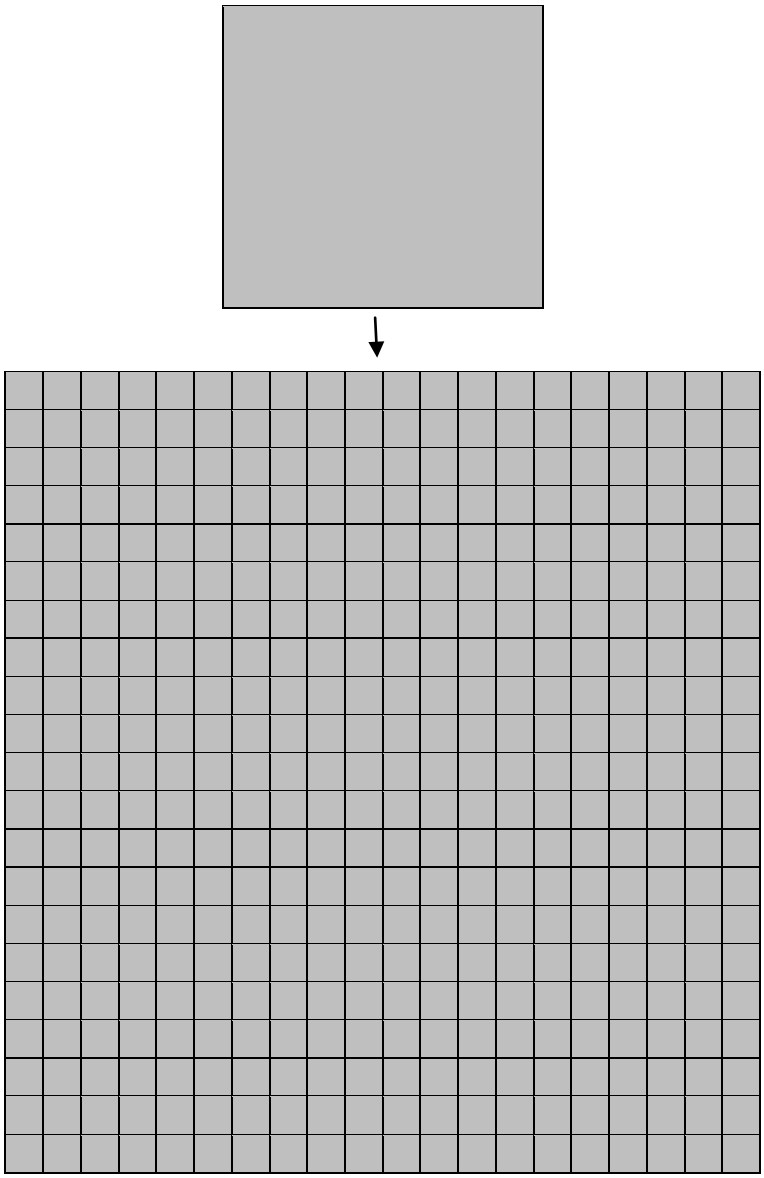
Setidaknya terdapat empat *file* templet *HaarCascade Frontal Face*, namun hanya satu yang akan digunakan penulis yaitu haarcascade_frontalface_alt.xml. *File* templet ini memiliki karakteristik sebagai berikut:

a. Menggunakan *sub-window* citra dengan dimensi 20x20
b. Memiliki 22 tahapan (*stage 0* sampai *stage* 21) *cascade classifier*
c. Memiliki 2133 *weak classifier* (*tree*)

Berikut ini adalah potongan kode *file* templet haarcascade_frontalface_alt2.xml

```
<?xml version="1.0"?>
<opencv_storage>
<haarcascade_frontalface_alt2 type_id="opencv-haar-classifier">
 <size>20 20</size>
 <stages>
  <_>
   <!-- stage 0 -->
   <trees>
    <_>
     <!-- tree 0 -->
     <_>
      <!-- root node -->
      <feature>
       <rects>
        <_>2 7 16 4 -1.</_>
        <_>2 9 16 2 2.</_></rects>
       <tilted>0</tilted></feature>
      <threshold>4.3272329494357109e-003</threshold>
      <left_val>0.0383819006383419</left_val>
      <right_node>1</right_node></_>
     <_>
```

Pada awal kode, tertera <*size*> yang mendefinisikan ukuran *sub-window* yang digunakan yaitu 20x20. Hal ini menunjukan bahwa setiap citra masukan akan dibentuk menjadi *sub-window* berukuran 20x20 sebelum masuk ketahapan deteksi wajah. Gambar 2.23 akan menjelaskan pembentukan *sub-window* 20x20.



Gambar 2.25. Pembentukan Sub-Window 20x20

Setelah *sub-window* terbentuk tahap selanjutnya adalah tahap klasifikasi yang melalui tingkatan-tingkatan (*stage*) *cascade classifier* dan *tree* sebagai *weak classifier*. Didalam *tree* terdapat satu *node* yaitu *root node*, yang isinya berbeda-beda disetiap *tree*. Didalam *node* ini, terdapat fitur *Haar* (*rects*), nilai ambang (*threshold*) proses deteksi wajah, nilai batas minimum (*left_val*) dan nilai batas maksimum (*right_val*) yang harus dipenuhi agar *sub-window* lolos ke tahap selanjutnya. Untuk dapat melewati suatu tahapan *sub-window* harus berhasil melewati setiap *tree* didalam suatu tahapan. Perhitungan nilai fitur dilakukan juga pada *tree*. Sebagai contoh, berikut ini adalah perhitungan fitur *Haar* (*rects*) pada suatu *tree*.

```
<rects>
<_>2 7 16 4 -1.</_>
<_>2 9 16 2 2.</_>
</rects>
```

Angka pada kolom pertama dan kedua menunjukan posisi piksel yang akan diklasifikasi (x, y) pada *sub-window*. Kolom ketiga dan keempat menunjukan lebar dan tinggi fitur. Sedangkan, angka pada kolom terakhir adalah konstanta yang akan dikalikan ke setiap fitur persegi tersebut, dimana angka ini berfungsi untuk menentukan piksel hitam dan piksel putih suatu fitur, dengan ketentuan jika nilai suatu piksel dalam suatu piksel < 0, maka piksel tersebut masuk ke daerah piksel hitam, jika tidak piksel tersebut masuk ke daerah putih.

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
(-1+2)=1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Gambar 2.26. Fitur Haar Pada Stage 0 – Tree 0

Gambar 2.26 menjelaskan gambaran fitur dalam *sub-window* pada kasus fitur *Haar* (*rects*) diatas. Sesuai dengan gambar tersebut, piksel hitam adalah piksel dengan nilai fitur = -1, sedangkan piksel putih adalah piksel dengan nilai fitur = (-1+2) = 1. Setelah daerah piksel hitam dan daerah piksel putih ditentukan, langkah selanjutnya adalah menghitung nilai fitur *Haar* keseluruhan caranya dengan mengurangi jumlah nilai fitur pada piksel area hitam dengan jumlah nilai fitur pada piksel area putih. Hasilnya akan dibandingkan dengan nilai yang tertera pada *threshold* pendeteksian wajah, jika nilainya berada dibawah *threshold* proses pendeteksian wajah maka fitur dianggap tidak ada, jika diatas *threshold* pendeteksian wajah fitur akan dilanjutkan ke *left_val* dan *right_val*. Apabila, fitur tersebut memiliki nilai diantara *left_val* dan *right_val*, maka *sub-window* dinyatakan berhasil melewati *tree* tersebut.

Untuk dapat melanjutkan ke tingkat *classifier* selanjutnya, seluruh *sub-window* harus dapat melewati seluruh *tree* dalam suatu tingkat, jika tidak maka secara otomatis, *sub-window* akan dieleminasi dan dianggap bukan wajah. Jika, berhasil melalui keseluruhan tingkat *classifier* maka *sub-window* dianggap sebagai wajah.

2.4.6 Haarcascade Eye

HaarCascade eye adalah suatu library berupa file templet berekstensi xml, yang salah satu penciptanya adalah Yusuf Bediz pada tahun 2009. Library ini mengadopsi metode Viola-Jones yang memiliki fungsi untuk mendeteksi objek berupa kedua mata.

Setidaknya terdapat delapan *file* templet *HaarCascade Frontal Eye*. namun hanya satu yang akan digunakan penulis yaitu frontalEyes35x16.xml *File* templet ini memiliki karakteristik sebagai berikut:

a. Menggunakan *sub-window* citra dengan dimensi 35x16
b. Memiliki 19 tahapan (*stage 0* sampai *stage* 18) *cascade classifier*
c. Memiliki 523 *weak classifier* (*tree*)

Berikut ini adalah potongan kode *file* templet frontalEyes35x16.xml:

```
<?xml version="1.0"?>
<opencv_storage>
<frontalEyes35x16 type_id="opencv-haar-classifier">
 <size>
  35 16</size>
 <stages>
  <_>
   <!-- stage 0 -->
   <trees>
    <_>
     <!-- tree 0 -->
     <_>
      <!-- root node -->
      <feature>
       <rects>
        <_>
         13 4 9 9 -1.</_>
        <_>
         16 4 3 9 3.</_></rects>
       <tilted>0</tilted></feature>
      <threshold>0.0102830100804567</threshold>
      <left_val>-0.8216220736503601</left_val>
      <right_val>0.5776131153106690</right_val></_></_>
```

Pada awal kode, tertera <*size*> yang mendefinisikan ukuran *sub-window* yang digunakan yaitu 35x16. Hal ini menunjukan bahwa setiap citra masukan akan dibentuk menjadi *sub-window* berukuran 35x16 sebelum masuk ketahapan deteksi mata. Tahapan selanjutnya hampir sama dengan proses haarcascade_frontalface_alt2.xml