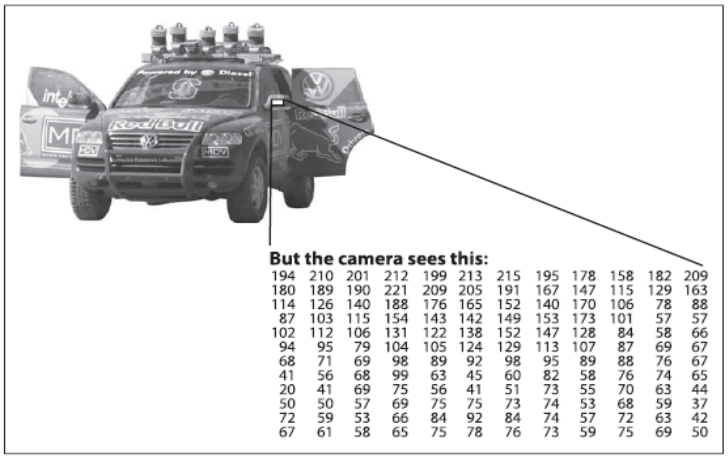
BAB II
LANDASAN TEORI

2.1 Computer Vision

Computer Vision adalah suatu proses transformasi atau perubahan dari data yang berasal dari kamera *video* maupun foto/gambar kedalam sebuah hasil keputusan ataupun sebuah presentasi yang baru, dimana hasil dari kegiatan transformasi tersebut memiliki kepentingan untuk mencapai suatu tujuan. Data yang dimasukan kedalam kegiatan transformasi tersebut memungkinkan untuk memiliki beberapa informasi yang terkonstektual seperti halnya sebuah foto/gambar yang didalamnya terdapat berbagai objek. Dengan demikian akan didapatkan keputusan-keputusan yang akan diambil pada gambar, misalnya berupa "adakah wajah seseorang pada gambar tersebut?" atau "siapa sajakah orang yang terdapat pada foto tersebut?". Adapun perubahan kedalam presentasi yang baru seperti perubahan gambar menjadi grayscale atau juga pemotongan objek pada gambar.

Berbeda halnya dengan manusia yang memiliki pemikiran untuk meneliti, memahami, dan membandingkan informasi pada objek secara langsung dengan informasi dari pengalaman-pengalaman yang didapatkan selama bertahun-tahun hidup di dunia. Dalam sistem penglihatan mesin (*machine vision*) komputer hanya mampu mendapatkan informasi berupa kumpulan angka-angka dari media input data seperti kamera atau disket. Berikut contoh gambar 2.1 untuk mengilustrasikannya.



Gambar 2. 1 Ilustrasi penglihatan komputer

Dari gambar berikut terlihat bahwa kaca samping pada gambar mobil hanyalah berupa kumpulan angka-angka jika dilihat oleh computer. Tentunya angka-angka ini masih memiliki banyak *noise* atau gangguan sehingga informasi yang terkandung sangatlah sedikit. Hal ini disebabkan dari penglihatan komputer yang terbatas hanya 2 dimensi (2D) namun objek yang diproses ialah objek 3 dimensi (3D) dunia nyata, sehingga informasi yang didapat mengalami banyak sekali gangguan dan kendala dari berbagai fenomena di kehidupan nyata (cuaca, cahaya, bayangan, refleksi cahaya, dan gerakan). Dengan demikian hal tersebut menjadi permasalahan yang terus-menerus diteliti untuk ditemukan cara atau tehnik penyelesaiannya (Bradsky, 2008).

2.2 Sistem Pengenalan Wajah

Sistem pengenalan wajah adalah sistem yang dibuat untuk dapat mengolah citra agar mampu mengidentifikasi dan memverifikasi wajah seseorang yang terdapat pada citra gambar. Sistem pengenalan wajah ini cukup populer, sehingga sering ditemukan dalam beberapa aplikasi untuk bidang keamanan dan bidang identifikasi pelaku kejahatan.

Untuk proses pengenalan wajah yang dilakukan oleh komputer tidak secepat dan sesimpel pengenalan yang dilakukan oleh manusia. Proses pengenalan wajah pada computer berjalan cukup lambat dan rumit, ditambah lagi dengan kesulitan

yang terjadi akibat beberapa faktor seperti cahaya, bayangan, kondisi gelap, serta objek lain yang menghalangi wajah.

Penelitian tentang pengenalan wajah telah berlangsung dari tahun 1960 hingga sekarang dengan beragan metode yang digunakan. Namun dari beragam metode tersebut, terdapat dua metode utama yang digunakan untuk pengenalan wajah, yaitu (Atalay, 1996):

1. **Pengenalan berbasis fitur**, ekstraksi yang berdasarkan pada ciri-ciri/karakter dari individu yang terletak pada wajah seperti mata, hidung dan mulut, serta hubungan mereka satu sama lain.
2. **Pengenalan berbasis pendekatan eigenfaces**, ekstraksi yang berdasarkan informasi yang paling relevan dan terkandung didalam 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 pencocokan citra yang diproses dengan citra yang telah disimpan kedalam database.

Pada dasarnya, meskipun sistem pengenalan wajah didasari dengan metode-metode yang berbeda, sistem pengenalan wajah memiliki tahapan-tahapan yang hampir sama, berikut tahapan-tahapan yang pada umumnya terdapat pada sistem pengenalan wajah :

1. *Grayscaling* citra gambar
2. Pendeteksian wajah atau *tracking* wajah
3. Ekstraksi wajah
4. Pengenalan/pencocokan wajah

Tahapan diatas memiliki keterkaitan antara satu sama lain, diantaranya adalah tahap grayscaling citra yang bertujuan untuk mengubah parameter warna kedalam satu warna yaitu derajat keabuan yang akan membantu dalam mempermudah proses pendeteksian wajah. Kemudian setelah pendeteksian wajah dilakukan maka akan dilakukan tahap ekstraksi wajah yang bertujuan untuk memisahkan objek

wajah yang terdeteksi untuk dapat dilakukannya tahap selanjutnya yaitu tahap pengenalan/pencocokan wajah.

2.2.1 Grayscaling

Grayscaling pada citra merupakan tahap pengkonversian citra gambar warna yang terdiri dari tiga parameter warna yaitu merah, hijau, dan biru (RGB). Jika citra gambar RGB diproses kedalam sistem pengenalan wajah, maka akan lebih sulit untuk diproses karena mengingat citra gambar RGB memiliki tiga parameter warna, oleh karena itu diperlukan suatu penyempitan parameter kedalam warna abu sehingga menjadi lebih mudah dalam hal penyamaan parameter yang akan digunakan pada sistem pengenalan wajah.

Untuk mengkonversikan citra yang memiliki warna RGB kedalam derajat keabuan dapat menggunakan persamaan (2.1) atau (2.2)

$$Gray = (R + G + B)/3 \quad (2.1)$$

atau

$$Gray = 0.114 * R + 0.587 * G + 0.299 * B \quad (2.2)$$

Maka hasil dari proses *grayscaling* citra dapat dilihat pada gambar 2.2.



Gambar 2. 2 Proses *grayscaling* citra

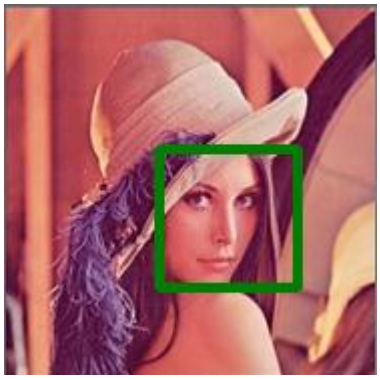
2.2.2 Normalisasi Citra

Normalisasi citra adalah proses yang bertujuan untuk mengolah citra kedalam suatu citra yang baru dengan tujuan agar citra baru tersebut dapat mengandung

informasi yang lebih banyak, sehingga proses ini dapat meningkatkan keakuratan suatu informasi yang terkandung didalam suatu citra.

2.2.3 Pendeteksian Wajah

Pendeteksian wajah adalah proses yang bertujuan untuk melakukan seleksi area wajah dari sebuah citra gambar yang dimasukan kedalam sistem. Proses ini bekerja dengan cara mendeteksi citra gambar yang telah dimasukan, apakah terdapat obek wajah pada citra gambar tersebut atau tidak. Apabila objek wajah telah ditemukan maka, citra wajah tersebut akan ditandai oleh sistem dan akan dilakukan tahapan-tahapan selanjutnya misalnya pengekstraksian wajah tersebut dari citra gambar sebelumnya. Selain itu, jika masukan pada sistem adalah bearbentuk video maka proses yang akan dilakukan adalah proses *face tracking*. Secara umum, *proses face* tracking dan proses pendeteksian wajah mempunyai fungsi yang sama. Berikut gambar 2.3 setelah dilakukannya proses pendeteksian wajah.



Gambar 2. 3 Proses pendeteksian wajah

2.2.4 Ekstraksi Wajah

Ekstraksi wajah merupakan proses yang berfungsi untuk mendapatkan informasi yang berguna dari hasil proses pendeteksian wajah terhadap citra gambar, yaitu berupa informasi mengenai citra wajah yang telah terdeteksi. Selanjutnya proses terdiri dari beberapa tahapan yaitu berupa :

1. Pemotongan citra
2. *Resizing* citra wajah
3. Penyimpanan kedalam *database*

2.2.4.1 Pemotongan Citra

Tahapan ini bertujuan untuk memisahkan citra gambar dengan citra gambar masukannya. Sehingga sistem dapat memperoleh citra wajah yang diperlukannya. Berikut gambar 2.4 citra wajah hasil dari tahapan-tahapan sebeleumnya dimulai dari tahap *grayscaling* hingga tahap pemotongan citra.



Gambar 2. 4 Hasil pemotongan citra

2.2.4.2 Resizing Citra

Pada tahap *resizing* citra ini 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.

2.2.4.3 Penyimpanan Citra

Proses penyimpanan adalah tahapan terakhir dari proses pelatihan citra wajah. Proses ini bertujuan untuk menyimpan informasi citra wajah yang telah didapat kedalam *file* atau *database*, yang nantinya akan digunakan sebagai pembanding citra wajah yang diuji pada proses pengenalan wajah.

2.2.5 Pengenalan Wajah

Proses ini bertujuan untuk membandingkan citra wajah yang dideteksi oleh sistem dengan citra wajah yang terdapat didalam *database*. Dari proses perbandingan wajah tersebut akan menghasilkan nilai jarak terdekat yang menandakan nilai wajah citra yang terdeteksi hampir menyamai dengan nilai citra wajah pada *database*. Nilai jarak ini akan menjadi nilai masukan untuk nilai kemiripan citra. Nilai kemiripan citra merupakan nilai tingkat kemiripan citra

yang dibandingkan, semakin besar nilainya menandakan bahwa orang yang sedang diamati adalah orang yang sama dengan orang yang citra wajahnya telah disimpan dalam *database*. Setelah itu dilakukan proses pencocokan, yaitu kondisi dimana hasil dari perbandingan wajah memberikan keputusan terhadap citra wajah yang terdeteksi apakah cocok dengan citra wajah yang telah dipilih dari *database*, sehingga hasil dari pencocokan adalah benar-benar hasil yang paling tepat dan akurat.

2.2.6 Metode Viola Jones

Metode Viola Jones adalah metode yang memiliki kapabilitas untuk mendeteksi wajah pada suatu citra gambar dengan tingkat keakuratan yang cukup tinggi. Metode ini berperan sebagai metode untuk mendeteksi objek wajah pada citra gambar masukan dalam sistem pengenalan wajah. Pada standarnya pengolahan citra pada gambar akan dilakukan dengan merubah skala ukuran gambar kedalam ukuran yang ditentukan dan melakukan pendeteksian dengan skala tetap pada gambar tersebut. Dengan hal tersebut simpulkan bahwa waktu untuk pendeteksian akan lebih cepat dibandingkan dengan melakukan pendeteksian terhadap citra gambar yang berukuran berbeda-beda.

Berbeda halnya dengan Metode Viola Jones yang terus merubah ukuran skala pendeteksian dibandingkan harus merubah ukuran dari skala gambar dan menjalankan banyak pendeteksian terhadap gambar dengan ukuran skala pendektesian yang berbeda tiap waktunya. Hal ini diperkirakan akan membutuhkan lebih banyak waktu, tapi Metode Viola Jones memiliki fitur pendeteksian yang mengkonstruksi gambar kedalam gambar integral dan beberapa fitur persegi yang simple yang disebut *Haar wavelets*. Sehingga hasil kalkulasi dalam skala pendeteksian akan sama berapapun ukuran dari gambar yang diproses (Jensen, 2008).

Langkah pertama dari pendeteksian ini yaitu dengan cara mengubah contoh gambar yang berdimensi 5x5 kedalam gambar integral. Yaitu dengan cara menjumlahkan tiap *pixel* dengan seluruh *pixel* yang terdapat di daerah sebelah kiri dan atas *pixel* tersebut, seperti yang terdapat pada contoh gambar 2.5 berikut.

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

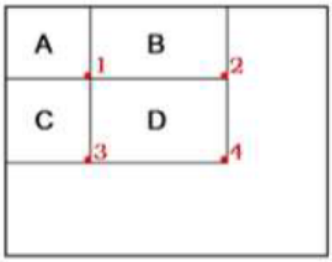
Input Image

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Integral Image

Gambar 2. 5 Gambar perhitungan integral

Dengan demikian, selanjutnya perhitungan dapat dilakukan dengan cara menjumlahkan 4 nilai *pixel* kedalam *pixel* yang telah dipilih. Berikut ilustrasi pada gambar 2.6.



Gambar 2. 6 Ilustrasi posisi *pixel* pada gambar

Dimana D adalah nilai *pixel* kanan bawah, A adalah nilai *pixel* kiri atas, B adalah *pixel* atas dari *pixel* D dan C adalah nilai *pixel* kiri dari *pixel* D. Dengan rumusan seperti berkut : D = D + A - (B + C). Dengan *integral image* ini maka perhitungan Haar akan menjadi lebih cepat. Berikut adalah perbandingan perhitungan haar pada gambar tanpa integral dengan gambar integral, dimana total *pixel* hitam dikurangi dengan total *pixel* putih :

Gambar tanpa integral

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

Input Image

Gambar 2. 7 Gambar tanpa integral

Sehingga nilai fitur Haar adalah sebagai berikut :

Nilai fiatur Haar = |(total *pixel* hitam)-(total *pixel* putih)|

=|(1+1+1+1+1+1+1+1)-(1+1+1+1+1+1+1+1)|

= |8-8|

= 0

Gambar integral

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Integral Image

Gambar 2. 8 Gambar dengan menggunakan interal

Sehingga nilai fitur Haar adalah sebagai berikut :

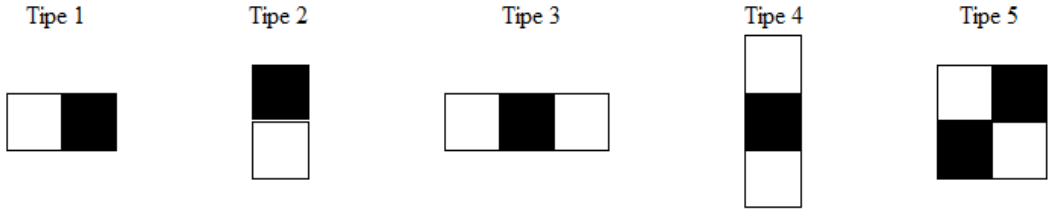
Nilai fiatur Haar = |(total *pixel* hitam)-(total *pixel* putih)|

=|(15+1-(3+5))-(25+3-(5+15))|

= |8-8|

= 0

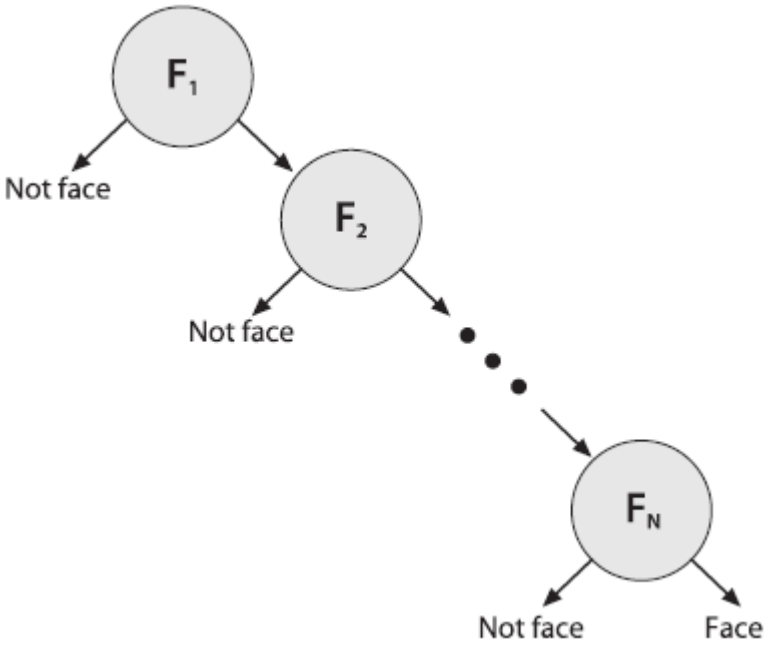
Kemudian hasil nilai tersebut akan menghasilkan fitur *sub-window* yang memiliki dua atau lebih kombinasi kotak hitam dan putih yang akan digunakan untuk tahap pendetaksian selanjutnya. Berikut contoh gambar 2.9 untuk tipe-tipe dari kombinasi kotak.



Gambar 2. 9 Tipe-tipe kombinasi kotak

Kombinasi kotak tersebut berfungsi sebagai acuan kriteria dalam penyeleksian fitur wajah, dengan cara mencari daerah dari pixel hitam dan pixel putih yang memenuhi kesamaan dengan kombinasi kotak yang telah ditentukan untuk pendeteksian wajah, dengan demikian ciri dari wajah dapat ditemukan dalam citra gambar.

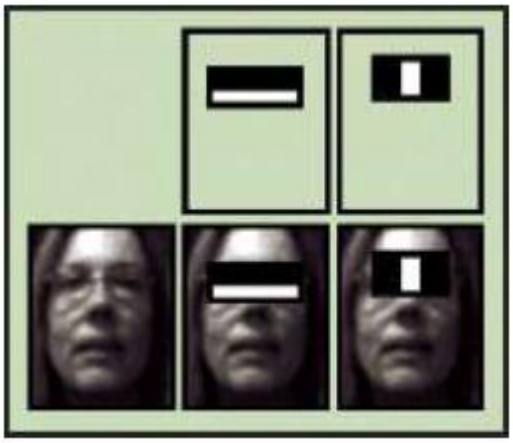
Selanjutnya, setelah nilai fitur didapatkan Viola-Jones menggunakan metode AdaBoost *machine-learning* untuk mengetahui suatu fitur apakah merepresentasikan ada tidaknya wajah dalam suatu citra masukan. Metode ini menggabungkan banyak *classifier* lemah menjadi satu *classifier* kuat. *Classifier* adalah informasi ciri yang menandakan adanya suatu objek pada suatu citra misalnya ciri pada wajah. Sedangkan *classifier* lemah adalah suatu jawaban benar namun memiliki tingkat kebenaran yang kurang akurat, maka jika digabungkan *classifier* lemah tersebut akan menghasilkan suatu *classifier* kuat. Berikut adalah gambar 2.10 dari metode AdaBoost yang digunakan untuk mendeteksi wajah.



Gambar 2. 10 Proses pendeteksian wajah AdaBoost (Bradsky, 2008)

Gambar berikut menjelaskan penolakan ciri menggunakan Viola Jones *Classifier*, dimana ciri yang memiliki kemungkinan kecil sebagai wajah akan ditolak terus menerus hingga *Classifier* akan menyisakan ciri wajah yang asli pada citra gambar.

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. 11 Fitur tingkat pertama pada Metode Viola Jones

Gambar 2.11 diatas mejelaskan fitur pendeteksian wajah menggunakan beberapa tipe kombinasi kotak untuk pencarian ciri wajah.

2.2.7 Principal Component Analysis (PCA)

PCA pertama kali ditemukan oleh Karl Pearson pada tahun 1901. Dan sekarang ini sering digunakan sebagai alat eksplorasi dalam analisis data dan untuk membuat model prediksi, misalnya untuk mengenali wajah. 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 (Purnama, 2010). Misalkan 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.

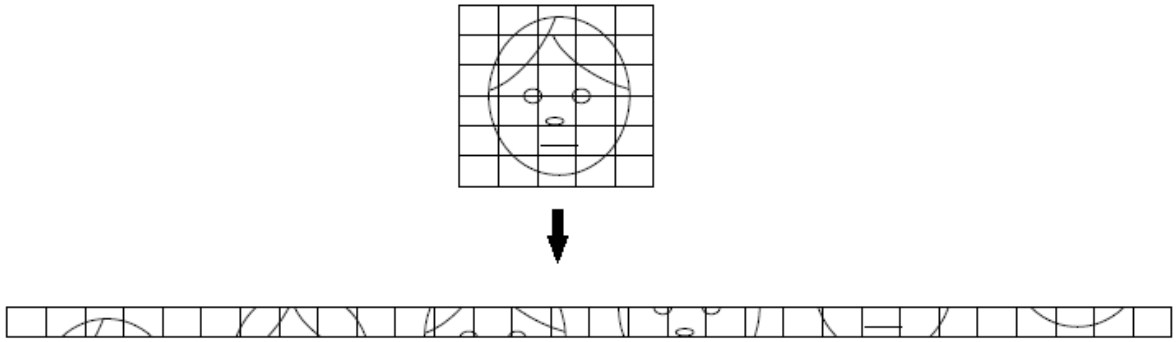
Sebagai contoh, misalnya terdapat data sampel 25 orang dengan 8 pose citra yang digunakan untuk pelatihan, maka m = 25x8 = 200. Jika tinggi citra 150 *pixel*

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 seperti pada persamaan (2.3) sebagai berikut.

$$x = \begin{bmatrix} x_{1,1}x_{1,2}x_{1,3} & \cdots & x_{1,n} \\ x_{2,1}x_{2,2}x_{2,3} & \cdots & x_{2,n} \\ x_{3,1}x_{3,2}x_{3,3} & \cdots & x_{3,n} \\ \vdots & & \vdots \\ x_{m,1}x_{m,2}x_{m,3} & \cdots & x_{m,n} \end{bmatrix} \quad (2.3)$$

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 2.12 berikut.



Gambar 2. 12 Representasi Perubahan Dimensi dari 6x5 Menjadi 1x30

2.2.8 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 persamaan (2.4).

$$\tau_i = \frac{\left(x_{1,i} + x_{2,i} + x_{3,i} + \cdots + x_{m,i}\right)}{m}$$

$$= \frac{\sum_{j=1}^{m} x_{j,i}}{m} = [\tau_1, \tau_2, \cdots\cdots\cdots, \tau_n] \qquad (2.4)$$

Terlihat bahwa hasil perhitungan persamaan diatas adalah merupakan sebuah *array* yang memuat nilai n, (τ1, τ2, … … …, τn). *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 (2.5).

$$\mu = a \; x \; \tau$$

$$\mu = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ \cdots \\ a_{m,1} \end{bmatrix} x \begin{bmatrix} \tau_{1,1} & \tau_{1,2} & \cdots & \tau_{1,n} \end{bmatrix} = \begin{bmatrix} \mu_{1,1} & \mu_{1,2} & \cdots & \mu_{1,n} \\ \mu_{2,1} & \mu_{2,2} & \cdots & \mu_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \mu_{m,1} & \mu_{m,2} & \cdots & \mu_{m,n} \end{bmatrix} \qquad (2.5)$$

Dimana *a* adalah matriks kolom dengan dimensi mx1 dan *a* untuk keseluruhan nilai adalah bernilai 1. Nilai dari [μi,1, μi,2, … … …, μi,n] pada baris ke i = nilai baris ke i+1 dan berlaku 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 (2.6).

$$\emptyset = x - \mu \tag{2.6}$$

Atau dapat dimodelkan menggunakan matriks seperti pada persamaan (2.7) berikut.

$$\emptyset = \begin{bmatrix} x_{1,1}x_{1,2} & \cdots & x_{1,n} \\ x_{2,1}x_{2,2} & \cdots & x_{2,n} \\ \cdots\ \cdots & \cdots & \cdots \\ x_{m,1}x_{m,2} & \cdots & x_{m,n} \end{bmatrix} - \begin{bmatrix} \mu_{1,1}\mu_{1,2} & \cdots & \mu_{1,n} \\ \mu_{2,1}\mu_{2,2} & \cdots & \mu_{2,n} \\ \cdots\ \cdots & \cdots & \cdots \\ \mu_{m,1}\mu_{m,2} & \cdots & \mu_{m,n} \end{bmatrix}$$

$$\emptyset = \begin{bmatrix} x_{1,1}-\mu_{1,1} & x_{1,2}-\mu_{1,2} & \cdots\cdots\cdots & x_{1,n}-\mu_{1,n} \\ x_{2,1}-\mu_{2,1} & x_{2,2}-\mu_{2,2} & \cdots\cdots\cdots & x_{2,n}-\mu_{2,n} \\ \cdots\cdots\cdots & \cdots\cdots\cdots & \cdots\cdots\cdots & \cdots\cdots\cdots \\ x_{m,1}-\mu_{m,1} & x_{m,2}-\mu_{m,2} & \cdots\cdots\cdots & x_{m,n}-\mu_{m,n} \end{bmatrix}$$

$$\emptyset = \begin{bmatrix} x_{1,1}-\mu_{1,1} & x_{1,2}-\mu_{1,2} & \cdots\cdots\cdots & x_{1,n}-\mu_{1,n} \\ x_{2,1}-\mu_{1,1} & x_{2,2}-\mu_{1,2} & \cdots\cdots\cdots & x_{2,n}-\mu_{1,n} \\ \cdots\cdots\cdots & \cdots\cdots\cdots & \cdots\cdots\cdots & \cdots\cdots\cdots \\ x_{m,1}-\mu_{1,1} & x_{m,2}-\mu_{1,2} & \cdots\cdots\cdots & x_{m,n}-\mu_{1,n} \end{bmatrix}$$

$$\emptyset = \begin{bmatrix} \emptyset_{1,1}\emptyset_{1,2} & \cdots & \emptyset_{1,n} \\ \emptyset_{2,1}\emptyset_{2,2} & \cdots & \emptyset_{2,n} \\ \cdots\ \cdots & \cdots & \cdots \\ \emptyset_{m,1}\emptyset_{m,2} & \cdots & \emptyset_{m,n} \end{bmatrix} \tag{2.7}$$

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 (2.8) matriks kovarian.

$$C = \frac{1}{\mathrm{m}-1}\emptyset_{j,i} * \emptyset_{j,i}^{T} \tag{2.8}$$

Persamaan (2.8) dapat diuraikan menggunakan matriks seperti pada persamaan (2.9).

$$C = \frac{1}{m-1}\begin{bmatrix} \emptyset_{1,1}\emptyset_{1,2} & \cdots & \emptyset_{1,n} \\ \emptyset_{2,1}\emptyset_{2,2} & \cdots & \emptyset_{2,n} \\ \cdots \cdots & \cdots & \cdots \\ \emptyset_{m,1}\emptyset_{m,2} & \cdots & \emptyset_{m,n} \end{bmatrix} * \begin{bmatrix} \emptyset_{1,1}\emptyset_{1,2} & \cdots & \emptyset_{1,n} \\ \emptyset_{2,1}\emptyset_{2,2} & \cdots & \emptyset_{2,n} \\ \cdots \cdots & \cdots & \cdots \\ \emptyset_{m,1}\emptyset_{m,2} & \cdots & \emptyset_{m,n} \end{bmatrix}^T$$

$$C = \frac{1}{m-1}\begin{bmatrix} c(x_1,x_1)\ c(x_1,x_2) & \cdots\cdots & c(x_1,x_m) \\ c(x_2,x_1)\ c(x_2,x_2) & \cdots\cdots & c(x_2,x_m) \\ \cdots\cdots\ \cdots\cdots & \cdots\cdots & \cdots\cdots \\ c(x_m,x_1)\ c(x_m,x_2) & \cdots\cdots & c(x_m,x_m) \end{bmatrix} \quad (2.9)$$

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 (2.10).

$$C - Z = |C - \lambda I|$$

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

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 (2.11) sebagai berikut.

$$\begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ C_{m,1} & C_{m,2} & \cdots & C_{m,n} \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,n} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ C_{m,1} & C_{m,2} & \cdots & C_{m,n} \end{bmatrix} - \begin{bmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \lambda \end{bmatrix} = 0$$

$$\begin{bmatrix} C_{1,1} - \lambda & C_{1,2} & \cdots & C_{1,n} \\ C_{2,1} & C_{2,2} - \lambda & \cdots & C_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ C_{m,1} & C_{m,2} & \cdots & C_{m,n} - \lambda \end{bmatrix} = 0 \quad (2.11)$$

Hasil dari matriks tersebut akan menghasilkan suatu persamaan (2.12) sebagai berikut.

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

Setelah itu dilakukan proses faktorisasi, maka akan didapatkan nilai (λ_1, λ_2, …, λ_{i+1}) yang nantinya akan digunakan untuk mencari nilai *eigenvector*, dengan persamaan (2.13).

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

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.

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

Kedua matriks tersebut dikalikan hingga didapatkan sebuah persamaan (2.15) sebagai berikut.

$$(C_{1,1} - \lambda_1)x_1 + C_{1,2}x_2 + \cdots + C_{m,n}x_n = 0; dst. \qquad (2.15)$$

Dari persamaan (2.14) didapatkan nilai (x1, x2, …, xn+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 *N* yang merupakan matriks hasil perhitungan rata-rata nol pada persamaan (2.7), yang dibagi oleh akar *Z*, sebagai *eigenvalue*.

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

2.2.9 Pencocokan Fitur Menggunakan Euclidean Distance

Euclidean distance adalah metode pendekatan untuk mencari kasus dengan menghitung kedekatan antara kasus baru dengan kasus lama, yaitu berdasarkan pada pencocokan bobot dari sejumlah fitur yang ada untuk digunakan pada proses pengenalan wajah ini. Berikut persamaan (2.17) dari metode *euclidean distance* yang berfungsi untuk mendapatkan nilai dari *eigen distance*.

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

Diketahui, $Eigenface_{train\ \ ke\ \ j}$ merupakan matriks fitur yang dihasilkan oleh pelatihan ke j, dengan j , 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 (2.18).

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

2.2.10 Photometric Normalization

Pada saat ini, performa dari sistem pengenalan wajah sudah mampu memberikan hasil yang baik dan memuaskan. Akan tetapi dalam beberapa situasi, sistem pengenalan wajah seringkali menghadapi beberapa persoalan seperti faktor pencahayaan, perubahan arah cahaya, wajah yang berkarakter hampir mirip dengan seseorang, facial tambahan pada wajah seperti kumis, gaya rambut, serta kacamata. Hal tersebut tentu saja sangat berpengaruh terhadap perubahan informasi yang terdapat pada gambar, sehingga akan mempengaruhi performa dari metode dan algoritma untuk pengenalan wajah.

Untuk mengatasi berbagai permasalahan yang timbul diakibatkan oleh situasi-situasi tersebut, maka banyak peneliti yang bekerja untuk menciptakan metode-metode dibidang pengolahan citra yang sesuai dan efektif dalam mengatasi setiap permasalahan. Metode tersebut kemudian disusun kedalam algoritma pemrograman, dengan demikian hingga saat ini algoritma tersebut salah satunya dikenal dengan *Algoritma Photometric Normalization*. Algoritma ini bertujuan untuk dapat memproses citra gambar terlebih dahulu sebelum dilakukannya

pendeteksian dan pengenalan wajah pada sistem. Sehingga algoritma ini diharapkan agar dapat meningkatkan performa algoritma pengenalan wajah. Adapun metode sebagai bagian dari algoritma ini yang telah dipilih untuk diterapkan kedalam sistem pengenalan wajah yaitu metode *isotropic smoothing* dan *multiscale retinex*.

2.2.10.1 Isotrophic Smoothing

Metode ini bekerja dengan cara mengolah pencahayaan yang dapat diperkirakan sebagai versi samar-samar (*blur*) dari gambar aslinya. Proses *blurring* ini dapat diimplementasikan dengan membuat fungsi pencahayaan sebagai *L* yang memiliki kesamaan dengan gambar asli *I* namun mengandung sebuah parameter berupa *smoothing constraint*. Fungsi pencahayaan dapat dibangun dengan meminimalkan nilai dari fungsi persamaan (2.19) berikut.

$$J(L) = \int_y \int_x (L - I)^2 dxdy + c \int_y \int_x \left(L_x^2 + L_y^2\right) dxdy \qquad (2.19)$$

Dimana *c* sebagai parameter yang menentukan kerelatifan nilai penting pada *smoothness constraint*. Permasalahan dari persamaan (2.19) dapat diselesaikan dengan menggunakan proses difusi *Euler-Lagrange* yang dapat dijabarkan pada persamaan (2.20) yang terlihat sebagai berikut.

$$I(x,y) = L(x,y) + c[\nabla L_N(x,y) + \nabla L_S(x,y) + \nabla L_E(x,y) + \nabla L_W(x,y)] \quad (2.20)$$

Dimana ∇L_D mengacu pada turunan untuk tiap masing-masing keempat pengaturan arah sekitar *pixel* (*neighbouring pixel*) dari *I* yaitu *pixel* yang dimaksud, contohnya lambang *N, S, E* dan *W* yang menunjukan arah dari sekitar *pixel*, sehingga $\nabla I_N(x,y)$ adalah nilai dari *(x , y) pixel* dikurang nilai dari *(x , y - 1) pixel*. Besaran dari nilai *smoothing* ditentukan oleh parameter *c*.

Kemudian untuk pembentukan citra yang baru dari hasil perhitungan maka digunakan metode untuk menentukan batas nilai maksimal dan minimal dalam perentangan jarak dinamis sebuah gambar yaitu dengan metode *Gain/offset correction*. Pada kasusnya diketahui sebuah jarak dinamis antara r_{max} dan r_{min}, dan

sebuah batas jarak tampilan dinamis d_{max}, kemudian persamaannya dapat dilihat sebagai berikut.

$$I_i'(x,y) = \frac{d_{max}}{r_{max} - r_{min}} \cdot (I_i(x,y) - r_{min}) \quad (2.21)$$

Proses ini mampu menghasilkan tampilan yang baik berdasarkan bentuk original sumber.

2.2.10.2 Multiscale Retinex

Adalah metode pencahayaan dengan sebuah gambar *I* yang didapatkan melalui camera maka akan memiliki dua komponen, refleksi cahaya $R(x,y)$ dan pencahayaan $L(x,y)$ (Horn, 1986).

$$I(x,y) = L(x,y).R(x,y) \quad (2.22)$$

Pencahayaan adalah besaran cahaya yang jatuh pada objek yang berasal dari sumber cahaya. Refleksi cahaya adalah besaran dari pantulan cahaya pada permukaan objek.

Kemudian gambar didekomposisi kedalam suatu pengrefleksian dan pencahayaan, dengan memperkirakan pencahayaan sebagai versi *low-pass* dari gambar asli, maka dari itu untuk menemukan refleksi dari cahaya maka gambar akan dibagi berdasarkan dari fungsi pencahayaan. Kemudian tahap tersebut dioptimalkan dengan cara memperkirakan pencahayaan sebagai kombinasi dari gambar yang dihasilkan oleh *low-pass filtering* terhadap gambar asli menggunakan variasi ukuran lebar *Gaussian*.

$$L(x,y) = \sum_{s=1}^{S} w_s(I(x,y) * G_s(x,y)) \quad (2.23)$$

Dimana w_s adalah masa dari ukuran pelebaran dan *Gaussian* sebagai $G_s(x,y)$ yang memiliki lebar σ_s pada skala *s* dan K untuk meningkatkan nilai e pada pemangkatan suatu nilai d atau K = e^d seperti yang didefinisikan sebagai persamaan (2.24) berikut.

$$G_s(x,y) = Ke^{-\frac{x^2+y^2}{\sigma_s^2}} \quad (2.24)$$

Kemudian untuk proses pembentukan gambar yang baru dari hasil perhitungan maka dapat kembali menggunakan persamaan (2.21).

2.3 C#

C# adalah bahasa pemrograman baru yang diciptakan oleh *Microsoft* dan dikembangkan dibawah kepemimpinan Anders Hejlsberg yang telah menciptakan berbagai macam bahasa pemrograman termasuk Borland Turbo C++ dan orland Delphi. Bahasa C# juga telah di standarisasi secara internasional oleh ECMA. Seperti halnya bahasa pemrograman yang lain, C# bisa digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis *windows* (desktop) dan aplikasi berbasis web serta aplikasi berbasis *web services*.

Keunggulan bahasa C# adalah :

1. program dapat di eksekusi di mesin komputer sendiri atau di transmisikan melalu web dan di eksekusi di komputer lainnya
2. memiliki sekumpulan perintah yang sama dengan C++ yang kaya akan fitur yang lengkap tetapi dengan gaya bahasa yang lebih diperhalus sehingga memudahkan penggunanya
3. C# memodifikasi perintah yang sepenuhnya sama dengan C++ dan memberitahu dimana letak kesalahan kita bila ada kesalahan dalam aplikasi, hal ini dapat mengurangi waktu kita dalam mencari error
4. menyediakan bantuan yang dibutuhkan untuk membuat tampilan yang rumit dengan *tools* seperti *frames, dropdown, tabbed windows, group button , scroll bar, background image*, dan lainnya

Banyaknya kelebihan dari bahasa C# ini bukan berarti bahasa ini tidak memiliki kekurangan, berikut adalah kekurangan bahasa C# :

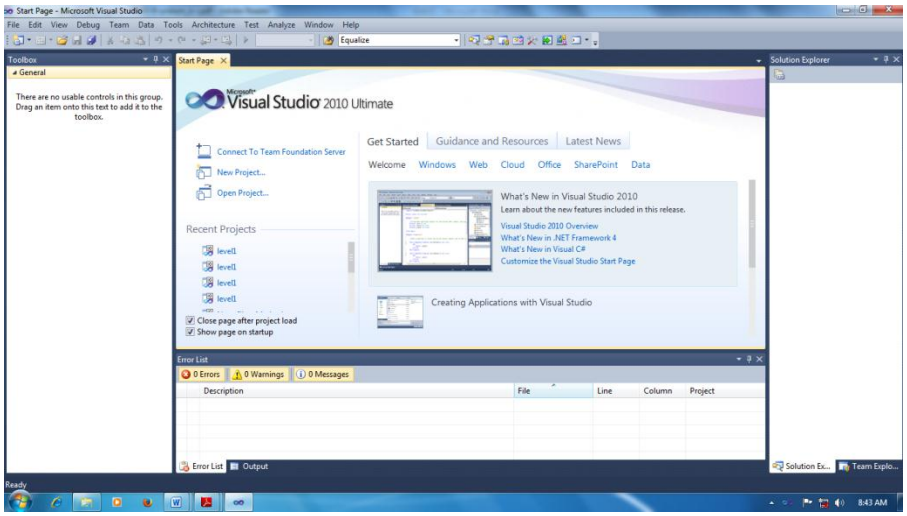
Banyaknya operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai, terutama bagi pemula yang pada umumnya akan kesulitan menggunakan *pointer*.

2.4 MS Visual 2010 Ultimate

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi *console*, aplikasi *Windows*, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas *Windows*) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* di atas .NET *Framework*). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas .NET *Compact Framework*).

Berikut gambar 2. 13 tampilan pada *MS Visual Studio 2010 Ultimate*.



Gambar 2. 13 Tampilan IDE Microsoft Visual Studio 2010