BAB II

TINJAUAN PUSTAKA

2.1 Mata

Mata adalah indra terbaik yang dimiliki oleh manusia sehingga citra (gambar) memegang peranan penting dalam perspektif manusia. Namun mata manusia memeliki keterbatasan dalam menangkap sinyal elektromagnetik. [5]

Mengedipkan mata adalah suatu proses penutupan dan pembukaan dari kelopak mata dalam waktu yang cepat. Proses tersebut merupakan suatu fungsi yang penting untuk memberikan air mata pada bagian-bagian mata dan untuk memindahkan benda yang tidak diinginkan dari permukaan kornea mata dan selaput mata.

2.2 Pengolahan Citra *(Image Processing)*

Pengolahan citra (*image processing*) adalah pengolahan suatu citra (gambar) dengan menggunakan komputer secara khusus, untuk menghasilkan suatu citra yang lain. [9]

Sesuai dengan perkembangan komputer itu sendiri, pengolahan citra mempunyai dua tujuan utama, yaitu sebagai berikut:

1. Memperbaiki kualitas citra, dimana citra yang dihasilkan dapat menampilkan informasi secara jelas. Hal ini berarti manusia sebagai pengolah informasi *(human perception).*

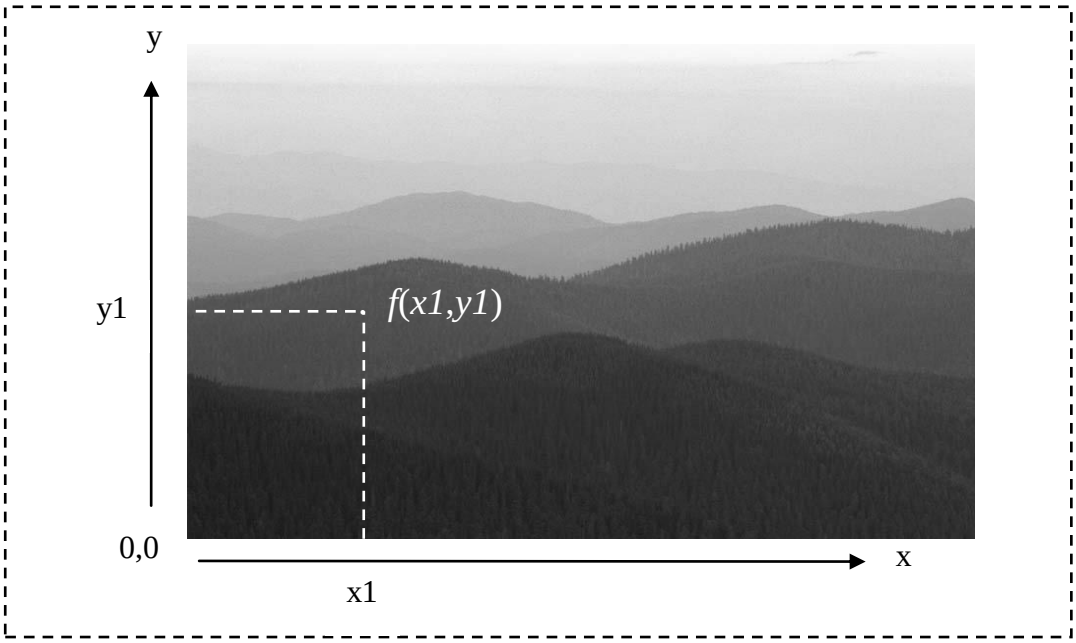
2. Mengekstraksi informasi ciri yang menonjol pada suatu citra, dimana hasilnya adalah informasi citra dimana manusia mendapatkan informasi ciri dari citra secara numerik atau dengan kata lain *computer* (mesin) melakukan interprestasi terhadap informasi yang ada pada citra melalui besaran-besaran data yang dapat dibedakan secara jelas (berupa besaran numerik).

Dalam perkembangan lebih lanjut, *image processing* dan *computer vision* digunakan sebagai pengganti mata manusia dengan perangkat *input image capture* seperti kamera dan scanner dijadikan sebagai mata dan mesin komputer dijadikan sebagai otak yang mengolah informasi. Sehingga muncul beberapa pecahan bidang yang menjadi penting dalam *computer vision* antara lain: *pattern recognition* (pengenalan pola), *biometric* (pengenalan identifikasi manusia berdasarkan ciri-ciri biologis yang tampak pada badan manusia), *content based image and video retrieval* (mendapatkan kembali citra atau video dengan informasi tertentu), *video editing* dan lain-lain.

2.2.1 Dasar – Dasar Pengolahan Citra Digital

Citra digital diasumsikan dengan persamaan f(x,y) dimana x menyatakan nomor baris, y menyatakan nomor kolom, dan f menyatakan nilai derajat keabuan dari citra. Sehingga (x,y) adalah posisi dari piksel dan f adalah nilai derajat keabuan pada titik (x,y). Kecerahan setiap citra disimpan dengan cara pemberian nomor pada setiap piksel. Makin tinggi nomor piksel maka makin gelap (hitam) piksel tersebut. Begitu juga sebaliknya makin rendah nilai piksel tersebut maka

makin terang. Sistem yang umum memiliki 256 tingkat kecerahan untuk setiap piksel, yang paling terang adalah 255 dan yang paling gelap adalah 0.



Gambar 0.1 Citra Digital

Citra atau gambar terbagi dalam tiga tipe adalah sebagai berikut: [5]

a. Gambar *Grayscale*

Citra yang terdiri dari satu layer warna dengann derajat keabuan tertentu. Dinyatakan dalam suatu fungsi :

$$f(x,y) \Sigma [0 \ldots 255] \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

b. Gambar Biner

Citra yang hanya memiliki dua nilai yaitu 1 dan 0. Dinyatakan dalam suatu fungsi :

$$f(x,y) \Sigma \{0,1\} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

c. Gambar berwarna

Citra yang terdiri dari tiga layer warna yaitu RGB (*Red-Green-Blue*) dimana *R-layer* adalah matrik yang menyatakan derajat kecerahan untuk warna merah, *G-layer* adalah matrik yang menyatakan derajat kecerahan untuk warna hijau, dan *B-layer* adalah matrik yang menyatakan derajat kecerahan untuk warna biru. Representasi dalam citra digital dinyatakan dalam persamaan :

$$\begin{aligned} &f_R(x,y)\ \Sigma\ [0\ldots255] \\ &f_G(x,y)\ \Sigma\ [0\ldots255] \\ &f_B(x,y)\ \Sigma\ [0\ldots255] \end{aligned} \qquad \text{……..(2.3)}$$

Proses pengolahan citra digital dengan menggunakan komputer digital adalah terlebih dahulu mentransformasikan citra ke dalam bentuk besaran-besaran diskrit dari nilai tingkat keabuan pada titik-titik elemen citra. Bentuk citra ini disebut citra digital. Elemen-elemen citra digital apabila ditampilkan dalam layar monitor akan menempati sebuah ruang yang disebut dengan piksel (*picture elemen/pixel*).

2.2.2 Model Citra Digital

Citra ada dua macam yaitu citra kontinu dan citra diskrit(citra digital).Citra kontinu dihasilkan sistem optik yang menerima sinyal analog, misalnya mata manusia dan kamera analog. Sedangkan citra diskrit dihasilkan melalui proses digitalisasi sehingga mampu menghasilkan citra diskrit (citra digital) misalnya kamera digital dan *scanner*. Citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi, dinyatakan dalam bentuk fungsi ***f(x ,y)*** dimana :[3]

(x, y) : koordinat pada bidang dua dimensi

f(x, y) : intensitas cahaya (*brightness*) pada titik *(x, y)*, besarnya $0 < f(x,y) < \infty$

Intensitas cahaya, *f*(*x, y*), sebenarnya merupakan hasil kali antara jumlah cahaya yang berasal dari sumbernya (*illumination*) dengan derajat kemampuan obyek memantulkan cahaya (*reflectance*), dinyatakan dalam bentuk persamaan berikut :

$$\boldsymbol{f(x, y) = i(x, y) \,.\, r(x, y)} \qquad \textbf{……(2.4)}$$

dimana :

i(*x, y*) : *illumination*, besarnya $0 < i(x, y) < \infty$

r(x, y) : *reflectance*, besarnya $0 < r(x, y) < 1$

Nilai *i*(*x, y*) ditentukan oleh sumber cahaya. Contoh nilai *i*(*x, y*) :

1. Pada hari cerah, matahari menghasilkan $i(x, y) \approx 9000$ *foot candles*
2. Pada hari mendung, matahari menghasilkan $i(x, y) \approx 1000$ *foot candles*
3. Pada malam bulan purnama, sinar bulan menghasilkan $i(x, y) \approx 0.01$ *foot candles*

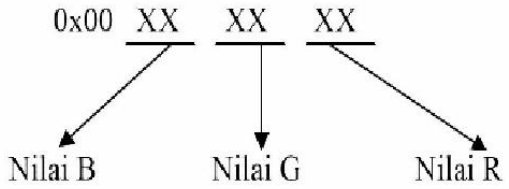
Nilai *r*(*x, y*) ditentukan oleh karakteristik obyek di dalam gambar.

Contoh nilai *r*(*x, y*) :

1. Benda hitam mempunyai $r(x, y) = 0.01$
2. Dinding putih mempunyai $r(x, y) = 0.8$
3. Benda logam *stainless steel* mempunyai $r(x, y) = 0.65$

2.2.3 Model Warna RGB

Dasar dari pengolahan citra adalah pengolahan warna RGB pada posisi tertentu. Dalam pengolahan citra, warna dipresentasikan dengan nilai hexadesimal dari 0x00000000 sampai 0x00ffffff. Warna hitam adalah 0x00000000 dan warna putih adalah 0x00ffffff. Definisi nilai 8 warna di atas seperti Gambar **2.2**, variabel 0x00 menyatakan angka dibelakangnya adalah hexadesimal.



Gambar 0.2 Nilai warna RGB dalam heksadesimal

Pengkodean warna RGB dapat ditulis dalam angka hexadesimal (basis 16) untuk masing-masing komponen R, G atau B. Misalkan:

1. Untuk Hitam murni akan ditulis dengan kode #000000 (R=00, G=00, B=00)
2. Untuk Putih sempurna akan ditulis dengan kode #FFFFFF (R=FF, G=FF, B=FF)
3. Untuk Biru murni akan ditulis dengan kode #0000FF (R=00, G=00, B=FF)

Model Warna RGB adalah sebuah model warna *additif* dimana pancaran warna r*ed* (merah), *green* (hijau), dan *blue* (biru) ditambahkan bersama dengan cara yang bervariasi untuk mereproduksi susunan warna yang lebar. Warna *additif* digunakan untuk *lighting*, video, dan monitor. Monitor sebagai contoh,

menciptakan warna dengan memancarkan cahaya melalui merah, hijau dan biru fosfor.

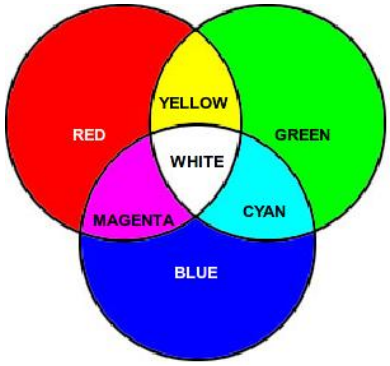
Tujuan utama dari model warna RGB adalah untuk mempresentasikan ulang, dan menampilkan gambar dalam sistem elektronik, misalnya dalam televisi dan komputer. Model warna RGB juga digunakan dalam fotografi konvensional.

RGB adalah sebuah ruang warna yang sifatnya bergantung kepada perangkat. Perangkat yang berbeda akan mendeteksi atau mereproduksi nilai RGB secara berbeda. Untuk membentuk warna dengan RGB, tiga cahaya berwarna balok (satu merah, satu hijau, dan satu biru) harus ditumpangkan (misalnya dengan emisi dari layar hitam, atau dengan refleksi dari layar putih). Masing-masing dari tiga balok disebut sebagai komponen warna, dan masing-masing dapat memiliki intensitas yang berbeda.

RGB sering digunakan dalam perangkat input seperti: TV berwarna dan kamera video, *scanner*, dan kamera digital. Perangkat output seperti: TV dalam berbagai teknologi (CRT, LCD, plasma, dll), komputer dan layar HP, video proyektor, layar LED multiwarna, dan layar lebar seperti JumboTron dan lain-lain.

RGB juga umumnya digunakan dalam pembuatan desain sebuah *website*. Awalnya kedalaman warna yang terbatas pada sebagian besar perangkat keras video memicu keterbatasan palet warna menjadi 216 warna RGB, hal itu ditetapkan oleh *Netscape Color Cube*. Namun, dengan keunggulan menampilkan

24-bit, penggunaan penuh dari 16.7 juta warna dari kode warna RGB HTML tidak lagi menimbulkan masalah bagi sebagian besar pengunjung website.



Gambar 0.3 Model warna RGB

2.2.4 Pengenalan Pola (*Pattern Recognition*)

Pengenalan pola (*Pattern Recognition*) adalah suatu ilmu untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif fitur (ciri) atau sifat utama dari suatu objek. [5]

Pengenalan pola mengelompokkan data numeric dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini computer). Tujuan pengelompokan adalah untuk mengenali suatu objek di dalam citra. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek di alam sehingga mampu membedakan suatu objek dengan objek lainnya. Kemampuan sistem visual manusia inilah yang dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut, dan memberikan keluaran berupa deskripsi objek didalam citra.



Gambar 0.4 Pengenalan Pola

Contoh pengenalan pola misalnya citra pada Gambar **2.5** adalah citra mata yang digunakan sebagai data masukan untuk mengenali mata. Dengan menggunakan suatu algoritma pengenalan pola, diharapkan komputer dapat mengenali bahwa citra tersebut adalah mata.



Gambar 0.5 Citra mata yang digunakan sebagai masukan untuk pengenalan mata

2.2.5 Penskalaan Citra (Scaling)

Penskalaan citra (*scaling*) adalah sebuah operasi geometri yang memberikan efek memperbesar atau memperkecil ukuran citra *input* sesuai dengan variabel penskalaan citranya.

Penskalaan citra (*scaling*) digunakan untuk memperbesar (*zoom-in*) atau memperkecil (*zoom-out*) citra. Rumus yang digunakan untuk penskalaan citra adalah :

$$x' = Sh\ x$$
$$y' = Sv\ y$$

……..……(2.5)

Keterangan :

Sh = faktor skala horizontal

Sv = faktor skala vertikal

Gambar Asli



400 x 200 *pixel*

Gambar Hasil *Scaling*



200 x 100 *pixel*

Gambar 0.6 Penskalaan Citra

2.2.6 Grayscale

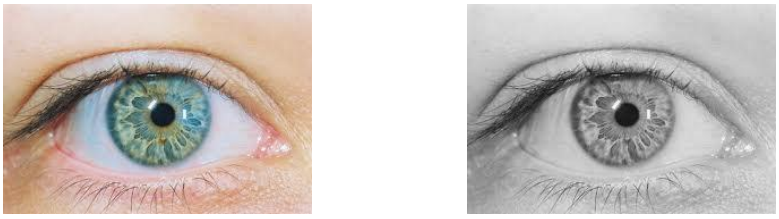
Citra *Grayscale* adalah citra yang hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian *Red* = *Green* = *Blue*. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan dan putih. Tingkat keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. [5]

Pengubahan dari citra berwarna ke bentuk *grayscale* mengikuti aturan sebagai berikut :

$$\mathbf{Grayscale} = \mathbf{0,2989\ R + 0,5870\ G + 0,1140\ B} \qquad \text{.....(2.6)}$$

Contoh gambar hasil *grayscale* :

Gambar asli ***Gambar grayscale***



Gambar 0.7 Citra *Grayscale*

2.2.7 Tresholding

Tresholding digunakan untuk mengatur jumlah derajat keabuan yang ada pada citra.. Proses *tresholding* ini pada dasarnya adalah proses pengubahan kuantisasi pada citra, sehingga untuk melakukan *tresholding* dengan derajat keabuan dapat digunakan rumus:

$$x = \frac{w}{b} \qquad \text{……..……..……(2.7)}$$

Keterangan :

w = Nilai derajat keabuan sebelum thresholding

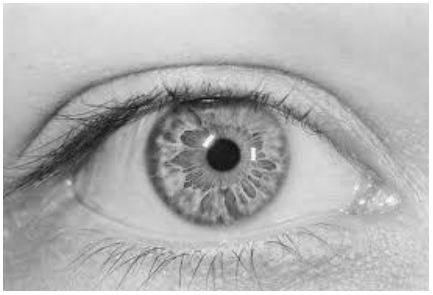
b = Jumlah keabuan { b = (256)}

x = Nilai derajat keabuan setelah *thresholding*

Proses *tresholding* akan menghasilkan citra biner, yaitu citra yang memiliki dia nilai tingkat keabuan yaitu hitam dan putih. Secara umum proses *tresholding* citra *grayscale* untuk menghasilkan citra biner adalah sebagai berikut:

$$g(x, y) = \begin{cases} 1 \text{ if } f(x,y) \geq T \\ 0 \text{ if } f(x,y) < T \end{cases} \qquad \text{……(2.8)}$$

Berikut ini contoh thresholding untuk mengubah citra grayscale menjadi 2 warna yaitu hitam dan putih :





Gambar 0.8 Citra *Tresholding*

2.3 *Sistem Real Time*

Sistem *real time* merupakan suatu sistem yang menangkap citra, memindahkan bingkai ke dalam memori computer, melakukan analisis dan perhitungan, dan menghasilkan citra lain atau lebih lagi sering digunakan untuk melakukan aksi, misalnya memberi predikat pada obyek yang diambil citranya seperti pada sistem sortasi, atau menggerakkan manipulator untuk memetik buah pada robot pemanen buah, dan sebagainya. [10]

Menurut kamus "*Oxford Dictionary of Computing*", Real-time system dapat didefinisikan sebagai: "Sistem apapun dalam hal waktu dimana suatu keluaran dihasilkan adalah penting. Hal ini biasanya dikarenakan suatu masukan yang berhubungan dengan suatu pergerakan dalam dunia fisik, dan keluarannya harus tetap memiliki hubungan dengan pergerakan tersebut. Keterlambatan dari waktu masuk sampai waktu keluar harus cukup kecil dan memenuhi batasan waktu yang dapat diterima".

Secara umum sistem *real time* merupakan suatu ukuran kinerja suatu sistem tertentu yang mana menyangkut batasan kinerja sistem dan tahapan perancangan sistem. Dalam sebuah sistem *real time* yang mana memiliki permasalahan waktu harus nyata, maksudnya harus memiliki persyaratan pewaktuan yang sesuai dengan keadaan nyata yang dijadikan parameter dalam sistem itu. Untuk mengkalibrasi citra *real time* ke dalam computer ialah dengan cara :

1. Mengidentifikasi sejumlah titik acuan atau *reference points*, dengan syarat koordinat titik – titik citra pada kamera diketahui dan koordinat-koordinat pada citra *realtime* juga diketahui, sehingga :

$$\boxed{\begin{aligned} \mathbf{U} &= \mathbf{u/t} \\ \mathbf{V} &= \mathbf{v/t} \end{aligned}} \qquad \textbf{……(2.9)}$$

Dimana : (U,V) ➔ koordinat – koordinat bidang citra

(u, v, t) ➔ koordinat – koordinat homogenus

2. Konversi koordinat – koordinat citra ke dalam dunia nyata dengan matriks 3 x 4 berikut :

$$\boxed{(x, y, z, 1) \begin{bmatrix} a1 & b1 & c1 \\ a2 & b2 & c2 \\ a3 & b3 & c3 \\ a4 & b4 & c4 \end{bmatrix} = (u, v, t)} \qquad \textbf{……(2.10)}$$

Jika dilanjutkan maka akan diperoleh :

$$\begin{aligned} x.a1 + y.a2 + z.a3 + a4 &= u \\ x.b1 + y.b2 + z.b3 + b4 &= v \\ x.c1 + y.c2 + z.c3 + c4 &= t \end{aligned} \qquad \textbf{……(2.11)}$$

Bila keempat titik ini dikumpulkan, nilai – nilai di dalam matriks dapat diselesaikan menggunakan persamaan simultan.

2.4 *Computer Vision*

Terminologi lain yang berkaitan erat dengan pengolahan citra digital adalah *computer vision* atau *machine computer*. Pada hakikatnya, *computer vision* mencoba meniru cara kerja visual manusia (*Human Vision*). *Human Vision* sesungguhnya sangat kompleks, manusia melihat obyek dengan indera penglihatan (mata) lalu obyek citra diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti obyek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk mengambil suatu keputusan.

Sebagaimana layaknya mata dan otak, *computer vision* adalah suatu sistem yang mempunyai kemampuan untuk menganalisis obyek secara visual, setelah data obyek yang bersangkutan dimasukkan dalam bentuk citra (*image*). [10]

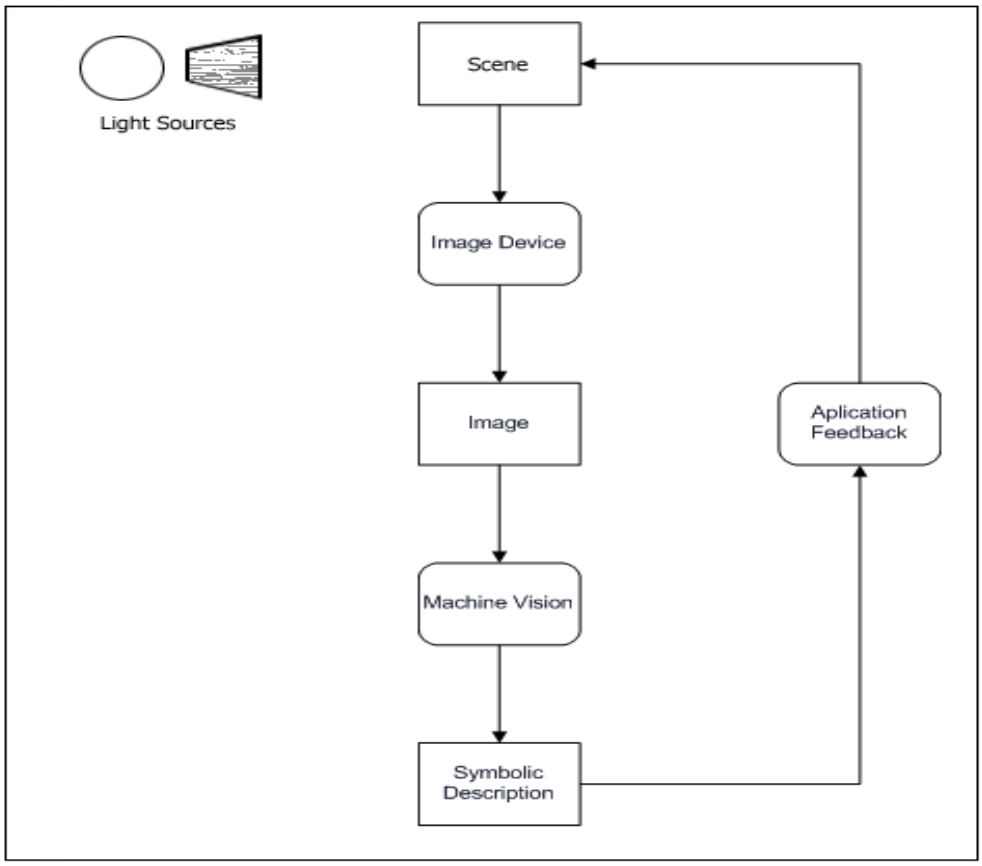
Proses-proses dalam *computer vision* dibagi dalam 3 (tiga) aktifitas :

1. Memperoleh atau mengakuisisi citra digital.
2. Melakukan teknik komputasi untuk memproses atau memodifikasi data citra.
3. Menganalisis dan menginterpretasi citra menggunakan hasil pemrosesan untuk tujuan tertentu, misalnya memandu robot, mengontrol peralatan, memantau manufaktur dan lain-lain.

Pengolahan citra merupakan proses awal (*preprocessing*) pada *computer vision*, sedangkan pengenalan pola merupakan proses untuk menginterpretasi citra. Teknik-teknik di dalam pengenalan pola memainkan peranan penting dalam *computer vision* untuk mengenali objek.

2.4.1 Elemen-Elemen *Computer Vision*

Gambar **2.9** di bawah ini adalah struktur yang mendasari elemen-elemen suatu mesin vision.



Gambar 0.9 Struktur *Computer Vision* secara umum

Keterangan gambar di atas adalah sebagai berikut :

a. *Light sources*, merupakan sumber cahaya yang digunakan sebagai sumber untuk aplikasi seperti layar laser, sistem robotika dan sebagainya.

b. *Scence*, merupakan kumpulan objek.

c. *Image Device*, merupakan alat yang digunakan untuk mengubah gambar menjadi suatu yang dimengerti oleh mesin.

d. *Image*, merupakan gambar dari suatu objek yang merupakan representasi dari keadaan sesungguhnya.

e. *Machine vision*, merupakan mesin yang menginterpretasikan gambar yang berkenaan dengan cirri-ciri pola maupun objek yang dapat ditelusuri oleh sistem.

f. *Symbolic description*, merupakan sistem yang dapat digunakan untuk menganalogikan struktur kerja sistem ke simbol-simbol tertentu yang dapat mimengerti oleh sistem.

g. *Aplication feedback*, merupakan suatu keadaan yang dapat memberikan respon untuk menerima gambar dari suatu sistem penglihatan.

2.5 Deteksi Mata *(Eye Detection)*

Untuk deteksi mata, pada penelitian ini menggunakan metode yang dibangun oleh *OpenCV* yaitu *cvHaar*. *cvHaar* adalah teknik modern yang mem-verifikasi roman wajah atau obyek yang lainnya. Selain untuk mendeteksi wajah, metode ini juga bisa digunakan untuk mendeteksi lokasi mata. Deteksi obyek oleh *OpenCV* diajukan oleh Paul Viola dan dikembangkan oleh Rainer Lienhart. *OpenCV* menggunakan tipe deteksi wajah disebut *Haar Cascade Classifier*. Dengan memberikan gambar yang berasal dari file maupun *live video*, detector ini menguji tiap lokasi gambar dan mengklasifikasi sebagai wajah atau bukan wajah

(dalam penelitian ini mata atau bukan mata). Klasifikasi dilakukan dengan menggunakan data yang disimpan dalam file XML, dimana berfungsi untuk memutuskan klasifikasi tiap lokasi gambar. Instaler *OpenCV* sudah termasuk didalamnya data XML, untuk deteksi wajah atau mata. Dalam aplikasinya cukup memberitahu software, dimana dapat menemukan file data yang ingin digunakan.

2.5.1 Integral Proyeksi

Integral proyeksi adalah suatu metode yang digunakan untuk mencari daerah lokasi dari obyek. Metode ini dapat kita gunakan untuk mendeteksi batas dari daerah gambar yang berbeda sehingga kita bisa mencari daerah lokasi kepala, tangan, kaki dan fitur – fitur lainnya. Integral ini juga di sebut dengan integral baris dan kolom dari pixel, karena integral ini menjumlahkan pixel per baris dan pixel per kolom. Dari metode ini akan dengan mudah untuk menemukan daerah lokasi objek yang diperlukan.

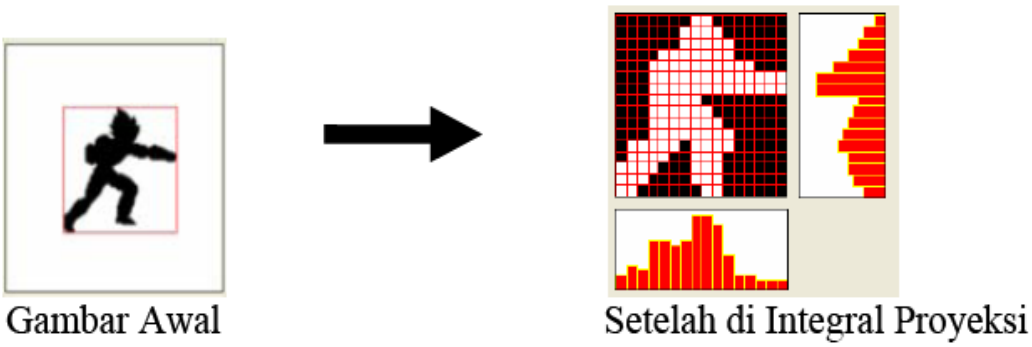
Rumus dari Integral Proyeksi adalah sebagai berikut :

$$h(j) = \sum_{i=1}^{Nbaris} x(i, j)$$

$$h(i) = \sum_{j=1}^{Nkolom} x(i, j)$$

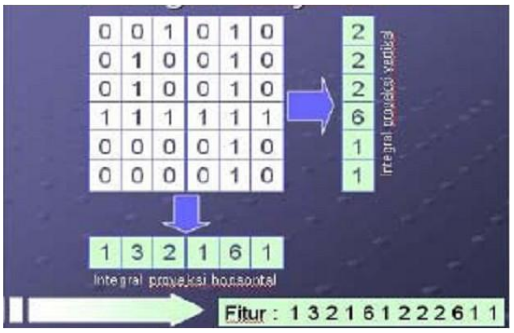
……(2.12)

Contoh penggunaan integral proyeksi dalam pengolahan image :



Gambar 0.10 Contoh *Integral Proyeksi*

Contoh Penghitungan dalam integral proyeksi :



Gambar 0.11 Contoh Penghitungan *Integral Proyeksi*

2.5.2 Metode *Haar Cascade Classifier*

Proses deteksi adanya citra mata dalam sebuah gambar pada OpenCV, menggunakan sebuah metode yang dipublikasikan oleh Paul Viola dan Michael Jones tahun 2001. Umumnya disebut metode Haar Classifier. Metode ini merupakan metode yang menggunakan *statistical model* (*classifier*). Pendekatan untuk mendeteksi objek dalam gambar menggabungkan empat konsep utama :

1. Training data
2. Fitur segi empat sederhana yang disebut fitur Haar.
3. Integral image untuk pendeteksian fitur secara cepat.
4. Pengklasifikasi bertingkat (Cascade classifier) untuk menghubungkan banyak fitur secara efisien.

2.5.2.1 *Training* data pada Haar

Metode ini memerlukan 2 tipe gambar objek dalam proses *training* yang dilakukan, yaitu :

1. *Positive samples*

   Berisi gambar obyek yang ingin di deteksi, apabila ingin mendeteksi mata maka *positive samples* ini berisi gambar mata, begitu juga obyek lain yang ingin dikenali.

2. *Negative samples*

   Berisi gambar obyek selain obyek yang ingin dikenali, umumnya berupa gambar *background* (tembok, pemandangan, lantai, dan gambar lainnya). Resolusi untuk sampel negatif disarankan untuk memiliki resolusi yang sama dengan resolusi kamera.

Training dari Haar menggunakan dua tipe sampel diatas. Informasi dari hasil training ini lalu dikonversi menjadi sebuah parameter model statistik.

2.5.2.2 Sistem Kerja Algoritma *Haar Cascade Classifier*

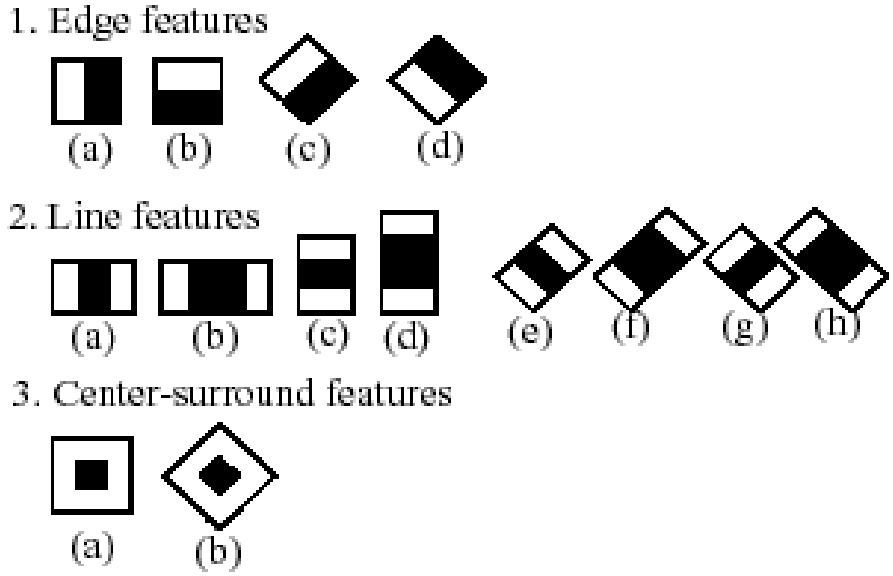
Algoritma Haar menggunakan metode statistikal dalam melakukan pengenalan mata. Metode ini menggunakan *simple haar-like features* dan juga *a cascade of boosted tree classifier.* Classifier ini menggunakan gambar berukuran tetap. Cara kerja dari haar dalam mendeteksi mata adalah menggunakan teknik *sliding window* pada keseluruhan gambar dan mencari apakah terdapat bagian dari gambar yang berbentuk seperti mata atau tidak. Haar juga memiliki kemampuan untuk melakukan *scalling* sehingga dapat mendeteksi adanya mata yang berukuran lebih besar ataupun lebih kecil dari gambar pada *classifier*.

Tiap *feature* dari *haar-like feature* didefinisikan pada bentuk dari *feature*, diantaranya koordinat dari *feature* dan juga ukuran dari *feature* tersebut.

2.5.2.3 *Haar Feature*

Haar Feature adalah fitur yang digunakan oleh Viola dan Jones didasarkan pada *Wavelet Haar*. *Wavelet Haar* adalah gelombang tunggal bujur sangkar (satu interval tinggi dan satu interval rendah). Untuk dua dimensi, satu terang dan satu gelap. Selanjutnya kombinasi-kombinasi kotak yang digunakan untuk pendeteksian objek visual yang lebih baik.

Setiap *Haar-like feature* terdiri dari gabungan kotak - kotak hitam dan putih



Gambar 0.12 ***Haar Feature***

3 tipe kotak(rectangular) feature:

Tipe *two-rectangle feature* (horisontal/vertikal)

Tipe *three-rectangle feature*

Tipe *four-rectangle feature*

Adanya fitur Haar ditentukan dengan cara mengurangi rata-rata piksel pada daerah gelap dari rata-rata piksel pada daerah terang. Jika nilai perbedaannya itu diatas nilai ambang atau treshold, maka dapat dikatakan bahwa fitur tersebut ada.

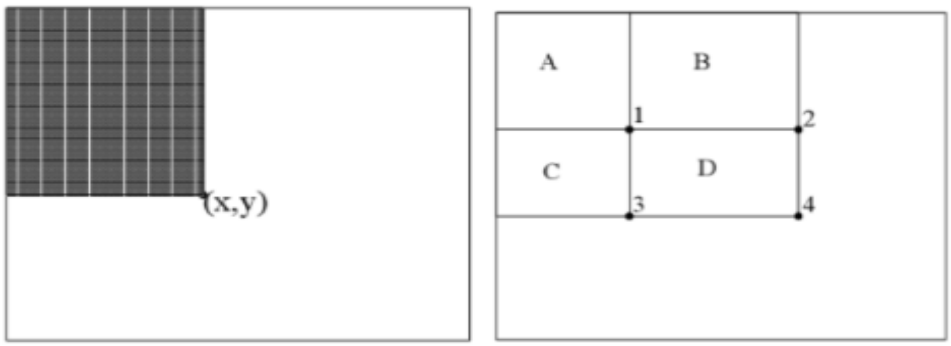
Nilai dari *Haar-like feature* adalah perbedaan antara jumlah nilai-nilai piksel *gray level* dalam daerah kotak hitam dan daerah kotak putih:

f(x)=Sumblack rectangle– Sumwhite rectangle	**……(2.13)**

Kotak Haar-like feature dapat dihitung secara cepat menggunakan "*integral image*"

2.5.2.4 *Integral Image*

Integral Image digunakan untuk menentukan ada atau tidaknya dari ratusan fitur Haar pada sebuah gambar dan pada skala yang berbeda secara efisien. Pada umumnya, pengintegrasian tersebut berarti menambahkan unit-unit kecil secara bersamaan. Dalam hal ini unit-unit kecil tersebut adalah nilai-nilai piksel. Nilai integral untuk masing-masing piksel adalah jumlah dari semua piksel-piksel dari atas sampai bawah. Dimulai dari kiri atas sampai kanan bawah, keseluruhan gambar itu dapat dijumlahkan dengan beberapa operasi bilangan bulat per piksel.



(a) **(b)**

Gambar 0.13 ***Integral Image***

Seperti yang ditunjukkan oleh gambar **2.13(a)** di atas setelah pengintegrasian, nilai pada lokasi piksel (x,y) berisi jumlah dari semua piksel di dalam daerah segiempat dari kiri atas sampai pada lokasi (x,y) atau daerah yang diarsir. Untuk menentukan nilai rata-rata piksel pada area segiempat (daerah yang diarsir) ini dapat dilakukan hanya dengan membagi nilai pada (x,y) oleh area segiempat.

$$P(x, y) = \sum i(x', y') \qquad \text{……(2.14)}$$

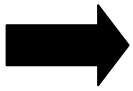
Untuk mengetahui nilai piksel untuk beberapa segiempat yang lain missal, seperti segiempat D pada gambar **2.13(b)** di atas dapat dilakukan dengan cara menggabungkan jumlah piksel pada area segiempat A+B+C+D, dikurangi jumlah dalam segiempat A+B dan A+C, ditambah jumlah piksel di dalam A. Dengan, A+B+C+D adalah nilai dari integral image pada lokasi 4, A+B adalah nilai pada lokasi 2, A+C adalah nilai pada lokasi 3, dan A pada lokasi 1. Sehingga hasil dari D dapat dikomputasikan :

$$D = (A+B+C+D)-(A+B)-(A+C)+A \qquad \text{……(2.15)}$$

Contoh Integral Image :

Image

0	**8**	**6**	**1**
1	**5**	**9**	**0**
0	**7**	**5**	**0**
2	**8**	**9**	**2**



Integral Image

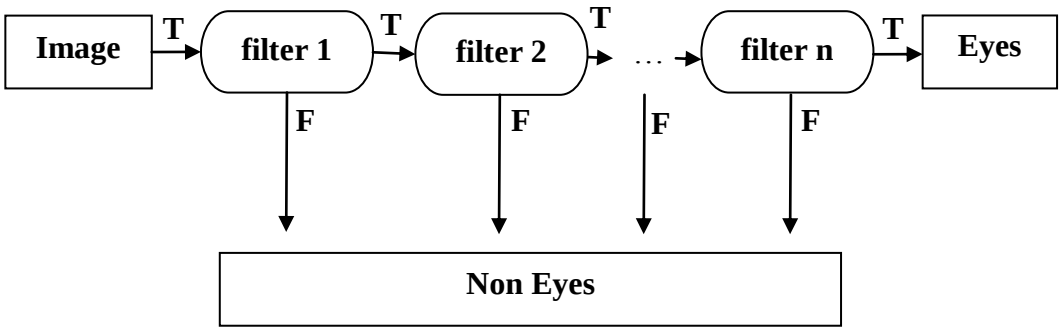
0	**8**	**14**	**15**
1	**14**	**29**	**30**
1	**21**	**41**	**42**
3	**31**	**60**	**63**

Gambar 0.14 Contoh *Integral Image*

2.5.2.5 *Cascade Classifier*

Cascade classifier adalah sebuah rantai *stage classifier*, dimana setiap *stage classifier* digunakan untuk mendeteksi apakah didalam image *sub window* terdapat obyek yang diinginkan (*object of interest*).

Stage classifier dibangun dengan menggunakan algoritma adaptive-boost (*AdaBoost*). Algoritma tersebut mengkombinasikan performance banyak weak classifier untuk menghasilkan *strong* classifier. *Weak* classifier dalam hal ini adalah nilai dari haar-like feature. Jenis *AdaBoost* yang digunakan adalah *Gentle AdaBoost*.



Gambar 0.15 *Cascade Classifier*

2.6 Tracking Mata *(Eye Tracking)*

Tracking mata dilakukan menggunakan metode *Eye Tracking*. Sistem akan mencari koordinat *pointer* , kemudian nilai koordinat tersebut akan dijumlahkan dengan titik tengah koordinat mata. Untuk menentukan koordinat titik tengah mata dapat dilakukan dengan aturan sebagai berikut :

$$\Delta Xn, \Delta Yn = ((Xen - Xe(n-1))/2, (Yen - Ye(n-1))/2) \quad \text{……(2.16)}$$

Kemudian untuk menentukan posisi pointer untuk melakukan pergerakan pointer menggunakan aturan sebagai berikut :

$$Xm', Ym' = ((\Delta Xn + Xm\text{-}1),(\Delta Yn + Ym\text{-}1)) \quad \text{.....(2.17)}$$

Dimana $Xm\text{-}1, Ym\text{-}1$ adalah 0,0.

2.7 Deteksi Kedipan *(Blink Detection)*

Berdasarkan konstren geometri wajah, mata terletak pada setengah bagian atas wajah. Karena pada umumnya, ketika seseorang berkedip kedua matanya bergerak bersamaan, maka dalam deteksi kedipan mata ini, sistem megidentifikasi kedipan masing – masing mata, baik mata sebelah kiri ataupun kanan.

Dari hasil segmentasi daerah mata yang diperoleh pada tahap sebelumnya, dapat diketahui posisi mata. Untuk itu citra warna pada daerah mata diubah menjadi citra keabuan (*grayscale*), yaitu dengan menghitung nilai keabuan tiap piksel (x,y). Dimana gray (x,y) adalah nilai intensitas keabuan pada piksel (x,y), sedangkan RGB adalah komponen citra warna. Proses selanjutnya adalah melakukan *equalisasi histogram* pada citra yang bertujuan untuk mengatasi masalah pencahayaan.

Tahap berikutnya yang penting adalah operasi morfologi. Dalam penelitian ini operasi morfologi digunakan untuk menghilangkan *noise* yang ada. Sebelum dilakukan operasi morfologi, dipilih daerah tertentu saja (daerah yang di-bounding) yang diproses atau biasa disebut ROI (*Region of Interest*) untuk mempercepat proses komputasi. Setelah menetapkan daerah ROI (pada penelitian ini ROI adalah daerah mata), pada daerah tersebut dilakukan proses *tresholding*,

dalam penelitian ini dipilih nilai ambang batas 5. Bila nilai intensitas keabuan piksel dibawah atau sama dengan 5, hasil deteksinya ditandai dengan warna putih, sedangkan bila nilai intensitas keabuannya di atas 5, hasil deteksinya ditandai dengan warna hitam.

2.7.1 Morfologi

Morfologi merupakan ilmu yang mengkaji bentuk dan struktur. Di dalam *computer vision*, bidang ini mengkaji tentang region – region atau bentuk, bagaimana mereka diubah dan dihitung, dan bagaimana area – area tersebut dievaluasi [8]

Secara umum tujuan dari operasi morfologi pada citra biner adalah untuk memperbaiki bentuk obyek agar dapat menghasilkan fitur – fitur yang lebih akurat ketika analisis dilakukan terhadap obyek. [9]

Operasi morfologi merupakan teknik pengolahan citra yang didasarkan pada bentuk segmen atau region dalam citra. Karena difokuskan pada bentuk obyek, maka operasi ini biasanya diterapkan pada citra biner. Biasanya segmen tersebut didasarkan pada obyek yang menjadi perhatian. Segmentasi dilakukan dengan membedakan antara obyek dan latar, antara lain dengan memanfaatkan operasi pengambangan yang mengubah citra warna dan skala keabuan menjadi citra biner. Nilai biner dari citra hasil merepresentasikan dua keadaan: obyek dan bukan obyek (latar). Meskipun lebih banyak dipakai pada citra biner, operasi morfologi sering pula digunakan pada citra skala keabuan dan warna. Hasil operasi morfologi dapat dimanfaatkan untuk pengambilan keputusan dengan

analisis lebih lanjut. Operasi ini antara lain meliputi pencarian batas/kontur, dilasi, erosi, penutupan (*closing*), pembukaan (*opening*), pengisian (*filling*), pelabelan, dan pengerangkaan (*skeletonization*).

2.7.1.1 Erosi

Erosi merupakan proses penghapusan titik-titik batas objek menjadi bagian dari latar, berdasarkan *structuring element* yang digunakan. Pada operasi ini, ukuran obyek diperkecil dengan mengikis sekeliling objek. Ada 2 cara yang dapat dilakukan untuk proses erosi, yaitu :

1. Dengan mengubah semua titik batas menjadi titik latar
2. Dengan menset semua titik di sekeliling titik latar menjadi titik latar

Contoh gambar proses erosi :



Gambar 0.16 Proses Erosi

2.7.1.2 Dilasi

Dilasi merupakan proses penggabungan titik-titik latar menjadi bagian dari objek, berdasarkan *structuring element* yang digunakan. Proses ini adalah kebalikan dari erosi, yaitu merubah latar disekeliling objek menjadi bagian dari objek tersebut. Ada 2 cara untuk melakukan operasi ini, yaitu :

1. Dengan cara mengubah semua titik latar yang bertetangga dengan titik batas menjadi titik obyek, atau lebih mudahnya set setiap titik yang tetangganya adalah titik obyek menjadi titik obyek.
2. Dengan mengubah semua titik di sekeliling titik batas menjadi titik obyek, atau lebih mudahnya set semua titik tetangga sebuah titik obyek menjadi titik obyek.

Contoh gambar proses dilasi :



Gambar 0.17 Proses Dilasi

2.8 *ROI (Region of Interest)*

Region of Interest (ROI) merupakan salah satu fitur yang tersedia dalam JPEG2000. ROI memungkinkan dilakukannya pengkodean secara berbeda pada area tertentu dari citra dijital, sehingga mempunyai kualitas yang lebih baik dari area sekitarnya (*background*). Fitur ini menjadi sangat penting, bila terdapat bagian tertentu dari citra dijital yang dirasakan lebih penting dari bagian yang lainnya.

2.9 *DFD (Data Flow Diagram)*

Data Flow Diagram (DFD) adalah sebuah diagram yang menggunakan notasi untuk menggambarkan arus data dari sistem. Penggunaan notasi ini sangat membantu dalam komunikasi dengan pemakai sistem untuk memahami sistem secara logika.

Penggunaan DFD sebagai modelling tools dipopulerkan oleh Tom De Marco (1978) dan Gane & Sarson (1979) dengan menggunakan pendekatan metoda analisis sistem terstruktur (*structured system analysis methode*).

2.9.1 Simbol-Simbol Pada DFD

Berikut ini merupakan beberapa simbol yang digunakan untuk pemodelan menggunakan DFD :

1. *Eksternal Entity*

Entity bisa berupa orang atau organisasi yang berada diluar sistem yang memberikan data kepada sistem atau yang menerima informasi dari sistem. Berikut ini adalah gambar eksternal entity pada DFD :

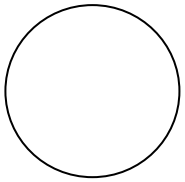


Gambar 0.18 Eksternal Entity

2. Proses

Menggambarkan apa yang dilakukan oleh sistem. Berfungsi mentransformasikan satu atau beberapa data masukan menjadi satu atau beberapa

data keluaran sesuai dengan spesifikasi yang diinginkan. Berikut ini adalah gambar proses pada DFD :



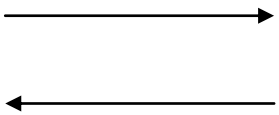
Gambar 0.19 Proses

3. *Data Flow*

Data Flow menggambarkan aliran data dari suatu entity ke entity lainnya. Arah panah menggambarkan aliran data. Aliran data bisa terjadi antara :

a. Dua proses yang berurutan
b. Dari data store ke proses dan sebaliknya.
c. Dari source ke proses
d. Dari proses ke link.

Berikut ini adalah gambar *data flow* pada DFD :



Gambar 0.20 *Data Flow*

4. *Data Store*

Menggambarkan tempat penyimpanan data. Proses dapat mengambil data dari atau memberi data ke store. Berikut ini adalah gambar *data store* pada DFD :



Gambar 0.21 *Data Store*

2.10 Bahasa Pemrograman C#

2.10.1 Sejarah Bahasa C#

Salah satu bahasa pemrograman terdini adalah bahasa *assembly*, di mana pemrogram membuat programnya dengan menuliskan instruksi- instruksi prosesor tertentu (biasa disebut *low level programming*). Karena segala seluk-beluk program berada 100% di tangan pemrogram, maka pemrogram yang handal mampu menciptakan program yang efisiensi dan kecepatannya sangat tinggi.

Ada beberapa kelemahan yang dimiliki bahasa *assembly*. Keharusan untuk hardware-hardware lain menyebabkan bahasa ini susah dipelajari dan dipakai. Kelemahan lain dari bahasa *assembly* adalah produktifitas pemrogram yang sangat rendah dan susahnya mengelola program berskala menengah ke atas. Untuk melakukan hal yang paling sederhana sekalipun diperlukan berbaris-baris kode dalam bahasa *assembly*. Semakin besar ukuran program kita, kesulitan untuk mengelolanya akan meningkat secara eksponen. Karena itulah muncul bahasa-bahasa pemrograman lain yang tingkat abstraksinya lebih tinggi.

Selain itu, kebutuhan akan penggunaan program tidak lagi hanya terbatas pada peralatan berupa personal computer, program juga digunakan pada *consumer-electronic devices* (cell phones dan PDA) sehingga menciptakan persoalan dan kebutuhan yang baru. Integrasi komponen software yang berbeda bahasa juga semakin sulit. Kebutuhan pengembangan aplikasi berbasis Web yang terkoneksi dengan Internet juga sudah semakin banyak. Pengembang software

sadar bahwa mereka membutuhkan software yang dapat digunakan siapa saja dan dapat digunakan pada semua perangkat yang ada (PC,*cell phones*, PDA, dll).

Untuk menjawab semua permasalahan dan kebutuhan diatas pada tahun 2000 Microsoft meluncurkan bahasa pemrograman baru yang diberi nama *C# Programming Language*. C# dikembangkan oleh Microsoft oleh tim yang dipimpin oleh Anders Hejlsberg dan Scott Wiltamuth. C#

2.10.2 Pengertian Bahasa C#

Bahasa C# adalah sebuah bahasa pemrograman modern yang bersifat general-purpose, berorientasi objek, yang dapat digunakan untuk membuat program di atas arsitektur Microsoft .NET Framework. Bahasa C# ini memiliki kemiripan dengan bahasa Java, C dan C++ .

Bahasa pemrograman ini dikembangkan oleh sebuah tim pengembang di Microsoft yang dipimpin oleh **Anders Hejlsberg**, seorang yang telah lama malang melintang di dunia pengembangan bahasa pemrograman karena memang ialah yang membuat Borland Turbo Pascal, Borland Delphi, dan juga Microsoft J++.

Kini, C# telah distandarisasi oleh European Computer Manufacturer Association (ECMA) dan juga International Organization for Standardization (ISO) yang mendukung beberapa fitur baru semacam Language Integrated Query (LINQ) dan lain-lainnya.

2.11 *Software* Pendukung

2.11.1 *Microsoft Visual Studio 2010*

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).

Microsoft kini merilis Microsoft Visual Studio 2010 dan Microsoft .NET Framework 4.0. Dua tool yang paling banyak digunakan untuk mengembangkan program di atas Windows, Windows Mobile, Web (ASP.NET), Silverlight, dan beberapa platform lainnya.

2.11.2 *OpenCV 2.2*

OpenCV 2.2 adalah suatu *library* gratis yang dikembangkan oleh Intel Corporation. *Library* ini terdiri fungsi-fungsi *computer vision* dan *image processing* tingkat tinggi. OpenCV diusulkan kepada para programmer untuk dapat menciptakan aplikasi yang handal, kuat dibidang digital vision. Karena *library* ini bersifat cuma-cuma dan terbuka, OpenCV tidak dipesan khusus untuk pengguna arsitektur Intel, tetapi dapat dibangun hampir semua arsitektur.

Semua kode program ditulis dalam C/C++/C# bahasa dan di-*compile* dengan gcc/g++/gsharp. Suatu pengetahuan yang umum tentang C programming adalah penting untuk memahami metoda programming digunakan di OpenCV. Kita harus mempunyai suatu pemahaman yang baik tentang bahasa program C sebab OpenCV semua aplikasinya ditulis dalam bahasa C.

Opencv menggunakan struktur *Iplimage* untuk menciptakan dan menangani gambar. Sebagai contoh :

.*width.* adalah lebar dari Image

.*height* adalah tinggi dari Image

.*nChannels*. jumlah *channel*

Masing – masing fungsi di OpenCV dimulai dengan .cv., sebagai contoh .cvCreateImage., . cvSobel., . cvAdd.. . .

Anda harus menambahkan .cv. untuk menggunakan fungsi apapun dari OpenCV. OpenCV juga menyediakan fungsi-fungsi GUI yang disebut highGUI (*high graphical user interface*).

2.12 *Hardware* Pendukung

2.12.1 *Web Camera (Webcam)*

Kamera digital digunakan sebagai sarana masukan untuk memberikan data objek gambar ke komputer. Sebuah kamera digital dapat menjangkau hingga resolusi yang cukup. Biasanya resolusi yang mendukung mencapai 320x460 sampai 768x1024 pixel. Kemampuan suatu kamera untuk mengambil banyak gambar utuh (frame) dalam satu detiknya menjadi penting.

Untuk menghasilkan video yang memiliki pergerakan yang lancar, suatu kamera setidaknya harus memiliki kemampuan mengambil gambar utuh sebanyak 30 frame setiap detiknya. Semakin banyak gambar utuh yang diambil setiap detiknya makin mulus gambar yang dihasilkan. Ada dua jenis kamera yang bisa digunakan, pertama adalah tipe WebCam yang dihubungkan melalui USB untuk tipe ini, citra yang dihasilkan kurang bagus. Tipe kedua adalah tipe VCR (Video Camera Recorder) atau kamera video digital yang dihubungkan melalui port fireware. Untuk jenis kedua, citra yang dihasilkan baik, dengan resolusi tinggi.



Gambar 2.22 *Web Camera*