BAB II

DASAR TEORI

2.1 Visi Komputer (*Computer Vision*)

Visi komputer merupakan ilmu yang mempelajari tentang bagaimana komputer dapat mengamati dan mengobservasi suatu benda tertentu. Hal ini bertujuan untuk meniru visualisasi yang dapat dilakukan oleh manusia sehingga dapat dilakukan oleh komputer, yang dalam aplikasinya dapat dilakukan pemrosesan lebih lanjut. Visi komputer terbagi menjadi 2 bidang ilmu, yaitu :

1. Pengolahan Citra (*image Processing*)

   Pengolahan Citra merupakan salah satu ilmu yang mempelajari tentang teknik perbaikan kualitas citra yang kurang baik menjadi citra yang lebih baik dan diterima sebagi citra yang mudah dikenali.

2. Pengenalan pola (*Pattern recognition*)

   Pengenalan pola merupakan pengidentifikasian objek berdasarkan proses pengelompokkan data numerik dan simbolik (termasuk citra), yang dilakukan secara otomatis oleh komputer.[5]

2.2 Face Recognition

Face recognition atau pengenalan wajah adalah salah satu teknologi biometrik yang telah banyak diaplikasikan dalam sistem keamanan selain pengenalan retina mata, pengenalan sidik jari dan iris mata. Dalam implementasinya, pengenalan wajah karyawan dengan menggunakan sebuah

webcam untuk menangkap wajah seseorang kemudian dibandingkan dengan wajah sebelumnya yang telah disimpan di dalam database tertentu.[14]

2.3 Metode *Eigenface*

Eigenface adalah sebuah algoritma *face detection* yang mudah untuk diimplementasikan.[1] Biasanya *eigenface* ini digunakan sebagai bahan pembelajaran karena merupakan metode pertama yang digunakan untuk mendeteksi benda sehingga metode ini yang paling sering digunakan. Pengenalan citra wajah dengan metode *eigenface* dilakukan berdasarkan pada pengurangan dimensi ruang wajah dengan menggunakan *Principal Component Analysis* (PCA) untuk memperoleh ciri wajah. Tujuan utama penggunaan metode PCA pada pengenalan wajah dengan menggunakan *eigenfaces* adalah membentuk (ruang wajah) dengan cara mencari vektor eigen yang berkoresponden dengan nilai eigen terbesar dari citra wajah. [2]

2.4 Proses Pengenalan dengan *Eigenface*

Dalam proses pengenalan dengan *eigenface*, terdapat beberapa tahapan yang dilakukan sehingga sebuah citra atau gambar dapat dikenali atau dideteksi, diantaranya adalah *grayscale, tresshold,* pengubahan dimensi citra, penyusunan *flatvector,* perhitungan rataan *flatvector,* proyeksi image ke *facespace,* ekstraksi PCA (Principal Component Analysis), dan pencarian *Distance minimum image test* dan *image* hasil *training*.

2.4.1 *Grayscale*

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 [9]. Berikut ini adalah persamaan *grayscaling* citra:

$$\boldsymbol{x} = \frac{(\boldsymbol{R}+\boldsymbol{G}+\boldsymbol{B})}{\boldsymbol{3}} \quad \text{..... (2.1)}$$

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

2.4.2 Threshold

Thresholding adalah proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk obyek dan *background* dari citra secara jelas. Citra hasil *thresholding* biasanya digunakan lebih lanjut untuk proses pengenalan obyek serta ekstraksi fitur.

$$g(x,y) = \begin{cases} 1 \; Jika \; f(x,y) \geq T \\ 0 \; Jika \; f(x,y) > T \end{cases} \quad \text{..... (2.2)}$$

T = *Threshold*

f(x,y)≥T disebut *object point*

2.4.3 Perubahan Dimensi Citra

Pada tahap ini, nilai dari tiap piksel yang disusun berupa matriks, akan di rubah dari 2D menjadi 1D. misalnya image yang disimpan berukuran (H x W) dalam bentuk matriks dikalikan dengan 1 maka persamaan tersebut (H x W) x 1 hasil tersebut akan berupa nilai piksel dalam bentuk satu baris. 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.

2.4.4 Penyusunan *Flatvector*

Tahap penyusunan *flatvector* adalah menyusun seluruh *training image* menjadi suatu matriks tunggal. Misalnya *image* yang disimpan berukuran H x W piksel dan jumlahnya N buah, maka akan dimiliki *vector* ciri dengan dimensi N x (W x H). Misalnya dalam *training image* terdapat 3 *image* dengan ukuran 3 x 3 piksel maka kita akan mempunyai *eigenvector* ukuran 3 x 9. Seperti yang diketahui N adalah jumlah citra, W adalah lebar suatu citra, dan H adalah panjang suatu citra.

2.4.5 Perhitungan Rataan *Flatvector*

Perhitungan rataan *flatvector* diperoleh dengan menjumlahkan seluruh baris tiap baris dari *flatvector*, sehingga diperoleh matriks berukuran 1 x (W x H) kemudian dilakukan pembagian pada tiap elemen matriks yang dihasilkan dengan jumlah *image* N untuk mendapatkan nilai rata-rata *vector* ciri.

2.4.6 Proyeksi *Image ke facespace*

Proses untuk memproyeksikan *Image* ke *facespace* diaplikasikan dengan melakukan perkalian image pada basis *eigenface*. Proyeksi vektor wajah akan dibandingkan dengan vektor yang sesuai. Dengan menggunakan nilai rata-rata *vector* ciri, maka dapat dihitung *eigenface* untuk matriks *vector* ciri yang telah disusun, yaitu dengan mengurangi baris-baris pada matriks *vector* ciri dengan nilai rata-rata *vector* ciri. Jika hasil yang diperoleh merupakan nilai dibawah nol, maka nilai tersebut akan digantikan dengan nol.

2.4.7 Ekstraksi PCA

Proses ekstraksi PCA adalah (Principal Component Analysis), dimana hasil proyeksi tersebut diektraksi dengan perhitungan PCA untuk mendapatkan *feature* dari image. *Feature* adalah komponen–komponen penting dari *image-image training* yang didapatkan dari proses *training*. *Feature* inilah yang nanti akan digunakan untuk mengidentifikasikan *image* yang akan dikenali. Kalkulasi nilai *eigenface* untuk matriks *testface,* dengan cara yang sama dengan penentuan *eigenface* untuk *vector* ciri. Dengan demikian pada ekstrasi PCA ini terdapat tahap – tahap untuk menghitung ekstrasi fitur dengan persamaan-persamaan berikut:

2.4.7.1 Pembentukan matriks data citra wajah

Tahapan awal dari ekstraksi PCA adalah pembentukan matriks yang datanya diambil dari *pixel* setiap citra. Data citra direpresentasikan dalam ukuran matrik m x n, dimana m adalah jumlah citra yang dilatih dan n merupakan

dimensi dari citra tersebut, kemudian dilakukan proses ekstraksi menjadi citra dengan dimensi yang lebih kecil yang hasilnya diproyeksikan menjadi sebuah matriks seperti pada persamaan berikut : [6]

$$x = \begin{bmatrix} x_{1,1}x_{1,2}\ x_{1,3} \dots & x_{1,n} \\ x_{2,1}x_{2,2}x_{2,3} & x_{2,n} \\ x_{3,1}\ x_{3,2}x_{3,3} & x_{3,n} \\ \dots & \dots \\ x_{m,1}x_{m,2}x_{m,3} \dots & x_{m,n} \end{bmatrix} \quad \text{..... (2.2)}$$

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: [6]

$$\tau_i = \frac{(x_{1,i} + x_{2,i} + x_{3,i} + \dots + x_{m,i})}{m}$$
$$= \frac{\sum_{j=1}^{m} x_{j,i}}{m} = [\tau_1, \tau_2, \dots \dots \dots, \tau_n] \quad \text{..... (2.3)}$$

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.

2.4.7.2 Penggandaan nilai rata-rata.

Penggandaan 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: [6]

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

$$\mu = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ \dots \\ a_{m,1} \end{bmatrix} x \begin{bmatrix} \tau_{1,1}\ \tau_{1,2} \dots \tau_{1,n} \end{bmatrix} = \begin{bmatrix} \mu_{1,1}\ \mu_{1,2} \dots \mu_{1,n} \\ \mu_{2,1}\ \mu_{2,2} \dots \mu_{2,n} \\ \dots \quad \dots \quad \dots \quad \dots \\ \mu_{m,1}\mu_{m,2} \dots \mu_{m,n} \end{bmatrix} \quad \dots\dots\dots,\dots\dots\dots\dots \text{ (2.4)}$$

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 $\forall j, j \in 1, 2, 3, \dots, m-1$, maka hasil persamaan diatas dapat digunakan untuk menghitung rata-rata nol.

2.4.7.3 Perhitungan nilai rata-rata nol (nilai deviasi image)

Perhitungan nilai rata-rata nol, berfungsi untuk menghilangkan *noise* yang dapat menganggu keakuratan padaperhitungan PCA. Perhitungan ini dapat dimodelkan menggunakan persamaan: [6]

$$\phi = x - \mu \quad \dots\dots\dots\

$$\phi = \begin{bmatrix} \phi_{1,1}\phi_{1,2} & \cdots & \phi_{1,n} \\ \phi_{2,1}\phi_{2,2} & \cdots & \phi_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{m,1}\phi_{m,2} & \cdots & \phi_{m,n} \end{bmatrix} \quad \text{……..…….. (2.6)}$$

2.4.7.4 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: [6]

$$C = \frac{1}{m-1} \phi_{j,i} * \phi^{T}{}_{j,i} \quad \text{……..…….. (2.7)}$$

Persamaan diatas dapat diuraikan sebagai berikut:

$$C = \frac{1}{m-1} \cdot \begin{bmatrix} \phi_{1,1}\phi_{1,2} & \cdots & \phi_{1,n} \\ \phi_{2,1}\phi_{2,2} & \cdots & \phi_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{m,1}\phi_{m,2} & \cdots & \phi_{m,n} \end{bmatrix} * \begin{bmatrix} \phi_{1,1}\phi_{1,2} & \cdots & \phi_{1,n} \\ \phi_{2,1}\phi_{2,2} & \cdots & \phi_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{m,1}\phi_{m,2} & \cdots & \phi_{m,n} \end{bmatrix}$$

$$C = \frac{1}{m-1} \begin{bmatrix} c(x_1,x_1) & c(x_1,x_2) & \cdots\cdots & c(x_1,x_m) \\ c(x_2,x_1) & c(x_2,x_2) & \cdots\cdots & c(x_2,x_m) \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ c(x_m,x_1) & c(x_m,x_2) & \cdots\cdots & c(x_m,x_m) \end{bmatrix} \quad \text{…….. (2.8)}$$

2.4.7.5 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: [6]

$$C - Z = |C - \lambda I|$$
$$|C - \lambda I| = 0 \quad \text{……..……..…….. (2.9)}$$

Dimana *C* adalah matriks kovarian. Z adalah matriks *eigenvalue* dengan λsebagai *scalar* pembentuknya dan *I* sebagai matriks identitas, persamaan tersebut jika diimplementasikan ke bentuk matriks menjadi persamaan sebagai berikut: [6]

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

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

$$\begin{bmatrix} C_{11}-\lambda & C_{12} & \dots & C_{1,n} \\ C_{21} & C_{2,2}-\lambda & \dots & C_{2,n} \\ \dots & \dots & \dots & \dots \\ C_{m,1} & C_{m,2} & \dots & C_{m,n}-\lambda \end{bmatrix} = 0 \quad \text{……..……... (2.10)}$$

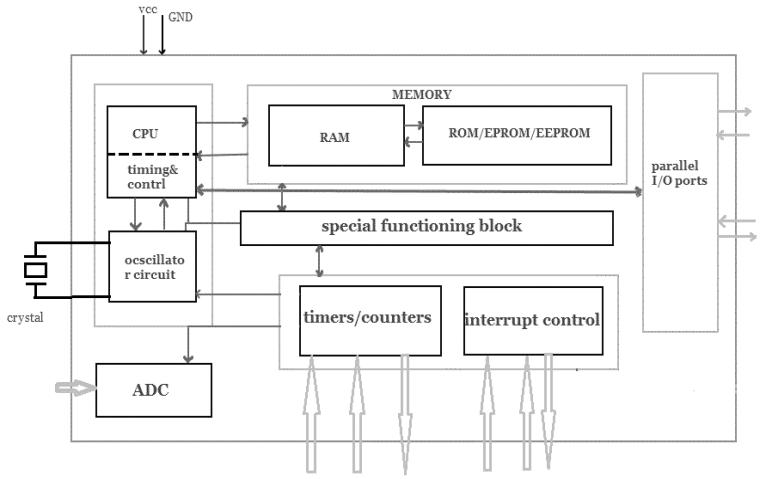
2.4.8 Pencarian *Distance Minimum.*

Proses akhir dari pengenalan wajah, biasanya dilakukan dengan membandingkan nilai *euclidean distance minimum* dari *image* yang di-*capture* dengan image yang sudah ada di *database*. Hal ini dilakukan dengan tujuan untuk mencari *distance* minimum antara *image test* dan *image* hasil *training.* Setelah nilai *eigenface* untuk *image test* diperoleh maka dapat dilakukan identifikasi dengan menentukan jarak (*distance*) terpendek dengan *eigenface* dari *eigenvector training image*.

2.5 Microcontroller

Microcontroller adalah sebuah alat pengendali yang berukuran mikro atau sangat kecil yang dikemas dalam satu chip.[7] *Microcontroller* juga dapat disebut sebagai komputer kecil, yang didalamnya terintegrasi beberapa komponen . *yang* terdiri dari CPU (*Central Processing Unit*), *memory*,

input/output ports, timers and counters, interrupt controls, analog to digital converters, serial interfacing ports dan *oscillatory circuits.*



Gambar 2.1 Blok Diagram Microcontroller [8]

Penjelasan singkat mengenai blok diagram *microcontroller* : *[8]*

a. CPU (*Central Processing Unit*)

CPU merupakan pusat pengontrol dari sebuah *microcontroller*, dapat juga dikatakan sebagai otak dari komputer. CPU bertugas dalam pengambilan instruksi (fetching), pengkodean (decodes), kemudian mengeksekusi setiap instruksi (execute). CPU menghubungkan setiap bagian yang ada pada *microcontroller*, menjadi sebuah kesatuan sistem. Fungsi utama dari CPU adalah pengambilan dan pengkodean instruksi. Instruksi yang telah diambil dari memori program, harus dilakukan pengkodean oleh CPU.

b. *Memory*

Fungsi dari memori dalam sebuah *microcontroller* hampir sama seperti mikroprosesor. Memori digunakan untuk menyimpan data dan program.

Sebuah mikroprosesor biasanya memiliki RAM dan ROM (EEPROM, EPROM, dll) atau memori *flash* untuk keperluan penyimpanan proram

c. *Parallel Input*/*Output Ports*

Parallel input/*output ports biasanya* digunakan untuk menghubungkan perangkat luar dengan *microcontroller (interface)* , seperti LCD, LED,dll.

d. *Serial Ports*

Serial port digunakan menghubungkan perangkat lain yang bersifat komunikasi serial.

e. *Timers / Counters*

Timers / counters merupakan salah satu fungsi yang sangat bermanfaat dalam *microcontroller*. Dalam satu *microcontroller,* bisa terdapat lebih dari satu *timer and counter*. *Timer and counter* menyediakan semua fungsi pewaktuan dan penghitungan pada sebuah *microcontroller*. Sebagian besar operasi yang berkaitan antara lain adalah fungis *clock*, modulasi, pembangkit pulsa, pengukuran frekuensi, dan pembuatan osilasi, dll.

f. ADC (*Analog to Digital Converter*)

ADC berfungsi untuk mengubah sinyal yang semula dalam bentuk *analog*, menjadi sinyal *digital*. Sinyal masukan dalam unit ini adalah sinyal *analog*, dan sinyal keluarannya adalah sinyal digital. Sinyal yang dihasilkan oleh ADC dapat diaplikasikan dalam beberapa aplikasi digital, seperti pembuatan alat ukur digital.

g. DAC (*Analog to Digital Converter*)

DAC merupakan kebalikan dari ADC. DAC mengubah sinyal *digital* ke sinyal *analog*. Sinyal masukan dalam unit ini adalah sinyal *digital*, dan sinyal keluarannya adalah sinyal *analog*. Sinyal yang dihasilkan oleh ADC dapat diaplikasikan dalam beberapa aplikasi *analog*, seperti pengendalian beberapa perangkat, seperti *motor DC*.

h. *Interrupt Control*

Interrupt control digunakan untuk melakukan interupsi berupa *delay* dalam mengerjakan sebuah program. Ada 2 jenis intrupsi, yaitu intrupsi *internal* (dari instruksi pada kode program), dan interupsi *eksternal* (dari pin interup).

i. *Special Functioning Block*

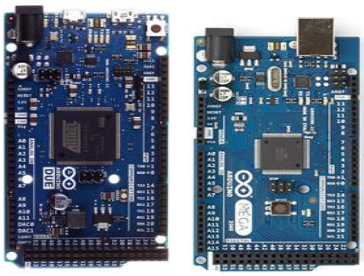
Special functioning block merupakan bagian yang berisikan port tambahan untuk melakukan beberapa operasi khusus.

Salah satu jenis *microcontroller* yang popular digunakan sekarang adalah *arduino. Arduino* adalah sebuah platform komputasi bersifat *open-source* dimana berbasis papan/board *microcontroller* sederhana. Hardwarenya menggunakan prosesor *atmel* AVR dan softwarenya memiliki bahasa pemrograman sendiri yaitu C++ *arduino*.

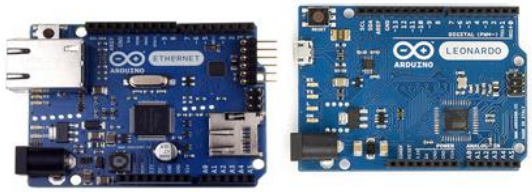
Jenis-jenis ini memiliki perbedaan dalam segi bentuk, spesifikasi dan performansi, secara singkatnya dijelaskan pada gambar 2.1 dan tabel 2.1.



Gambar 2.2 (a) *[11] Arduino UNO & Arduino Mega ADK*



Gambar 2.2 (b) *Arduino Due & Arduino Mega*



Gambar 2.2 (c) *Arduino Ethernet & Arduino LEONARDO*

Tabel 2.1 Spesifikasi *Arduino*

	processor	Processor voltage	Supply voltage	Flash	SRAM	Digital I/O Pins	PWM Pins	Analog Insputs	Hardware Serial Port	Dimensions
Uno	16MHz Atmega 328	5v	7-12V	32Kb	2Kb	14	6	6	1	2.1”x2.7” 53x75mm
Uno Ethernet	16MHz Atmega 328	5v	7-12V	32Kb	2Kb	14	6	6	1	2.1”x2.7” 53x75mm
Mega	16MHz Atmega 2560	5v	7-12V	256Kb	8Kb	54	14	16	4	2.1”x4” 53x102mm
Mega ADK	16MHz Atmega 2560	5v	7-12V	256Kb	8Kb	54	14	16	4	2.1”x4” 53x102mm
Leonardo	16MHz Atmega 32U4	5v	7-12V	32Kb	2.5Kb	20*	7	12*	1	2.1”x2.7” 53x75mm
Due	84MHz ARM SAM3X8E	3.3v	7-12V	512Kb	96Kb	54	12	12	4	2.1”x4” 53x102mm
Micro	16MHz Atmega 32U4	5v	5V	32Kb	2.5Kb	20*	7	12*	1	0.7”x1.9” 53x49mm
Flora	8MHz Atmega 32U4	3.3v	3.5-16V	32Kb	2.5Kb	8*	4	4*	1	1.75” dia 44.5mm dia

2.6 *Arduino* IDE (Integrated Development Environment)

Arduino IDE adalah sebuah *software* untuk menulis program, mengedit program sampai mengirim program tersebut ke *arduino.* Program *arduino* sering disebut *sketch. Arduino* IDE merupakan *software* editor yang berbasiskan *open*

source, yaitu cara penggunaannya dapat dilihat oleh orang banyak. *Software* ini digunakan untuk *microcontroller arduino*, dengan bahasa yang digunakan yaitu menggunakan bahasa C. *Software* ini dapat menampilkan data *interface* antara *microcontroller* dan *software*.[5]

2.7 Kamera *Webcam*

Kamera *webcam*, merupakan salah komponen yang digunakan untuk menangkap objek gambar yang akan diolah. Perangkat ini dihubungkan dengan komputer menggunakan komunikasi serial USB. Resolusi dari spesifikasi kamera *webcam*, menentukan kualitas citra yang dihasilkan. Perangkat ini juga dapat menampilkan gambar secara langsung atau *realtime* pada monitor jika dihubungkan dengan komputer.[3]



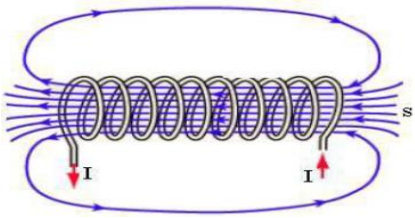
Gambar 2.3 Kamera *Webcam* C170

2.8 *Solenoid*

Solenoid adalah peralatan yang dipakai untuk mengkonversi sinyal elektrik atau arus listrik menjadi gerak mekanik. *Solenoid* dibuat dari kumparan dan inti besi yang dapat digerakkan yang berfungsi sebagai aktuator untuk membuka kunci otomatis pada brankas.

Secara skematik bentuk dari *solenoid* dapat dilihat pada Gambar 2.16, dimana selonoid terdiri dari *n* buah lilitan kawat berarus listrik I, medan magnet yang

dihasilkan memiliki arah seperti pada gambar dibawah, dimana kutub utara magnet mengikuti aturan tangan kanan. [4]



Gambar 2.4 *Solenoid* Dengan Banyak Lilitan *n* [4]

Kuat medan magnet pada selonoid dengan jumlah lilitan persatuan panjang *n* adalah dapat dihitung dengan menggunakan persamaan dibawah ini.

$B = \mu 0 \, . \, n \, . \, I$

Dimana : $n=Nl$

Keterangan :

B = Medan Magnet

μ0 = Konstanta Permeabilitas Udara

n = Jumlah Lilitan Persatuan Panjang

N = Jumlah Lilitan

l = Panjang Lilitan

I = Arus

Prinsip kerja dari *Solenoid* berdasarkan pada penghantar yang membawa arus kedalam kumparan sehingga kumparan akan menimbulkan medan magnet. Medan magnet ini dibuat sedemikian rupa sehingga keadaannya selalu tolak menolak antara medan magnet. Gambar 2.16 dibawah menunjukkan bentuk fisik *solenoid*.



Gambar 2.5 Bentuk Fisik *Solenoid* [4]

2.9 MATLAB 2015

MATLAB adalah sebuah bahasa dengan kemampuan tinggi untuk komputasi teknis. MATLAB menggabungkan **komputasi, visualisasi, dan pemrograman** dalam satu kesatuan yang mudah digunakan di mana masalah dan penyelesaiannya diekspresikan dalam notasi matematik yang sudah dikenal. Pemakaian MATLAB meliputi :

- Matematika dan komputasi
- Pengembangan algoritma
- Akuisisi data
- Pemodelan, simulasi dan *prototype*
- Grafik saintifik dan *engineering*
- Perluasan pemakaian, seperti graphical *user interface* (GUI).

MATLAB adalah system interaktif yang mempunyai basis data *array* yang tidak membutuhkan dimensi. Ini memungkinkan kita dapat menyelesaikan banyak masalah komputasi teknis, khususnya yang berkaitan dengan formulasi matrik dan *vector*.

Nama MATLAB merupakan singakatan dari *matrix labolatory* . MATLAB awalnya dibuat untuk memudahkan dalam mengakses *software* matriks yang telah dikembangkan oleh LINPACK dan EISPACK. Dalam perkembangannya, MATLAB mampu mengintegrasikan beberapa *software*

matriks sebelumnya dalam satu software untuk komputasi matriks. Tidak hanya itu, MATLAB juga mampu melakukan komputasi simbolik yang biasa dilakukan oleh MAPLE.

Sistem pada MATLAB

Matlab dapat digunakan dan dimanfaatkan untuk beberapa keperluan sebagai berikut

- **Analisis Data dan Visualisasi**

MATLAB merupakan aplikasi yang menyediakan alat untuk memperoleh, menganalisis, dan memvisualisasikan data, juga memungkinkan untuk mendapatkan informasi tentang data secara cepat spreadsheet atau bahasa pemrograman tradisional.

- **Mengakuisisi data**

MATLAB memungkinkan pengaksesan data dari *file*, aplikasi lain, database, dan perangkat eksternal. Hal ini berlaku untuk data dari format *file* populer seperti Microsoft Excel, *file* teks atau biner, gambar, suara, dan *file* video dan *file* ilmiah seperti netCDF dan HDF. Fungsi *file* I/O MATLAB dapat bekerja dengan *file* data dalam format apapun.

MATLAB juga dapat memungkinkan perolehan data dari perangkat keras serperti port serial komputer, *soundcard*, secara langsung, dimana data tersebut dapat diukur dan divisualisasikan untuk dianasis. Matlab juga dapat melakukan komunikasi dengan beberapa instrument seperti osiloskon, generator fungsi, dan penganalisa sinyal.

- **Menganalisis data**

MATLAB dapat memungkinkan pengelolaan, filter, dan *preprocess* data. Matlab dapat menyediakan fitur yang mendukung analisis data eksplorasi untuk mengungkap tren, asumsi tes, dan membangun model deskriptif. MATLAB menyediakan fungsi untuk menyaring dan *smoothing* , interpolasi , konvolusi , dan transformasi Fourier (FFTs). *Add-on* produk menyediakan kemampuan untuk kurva dan ketepatan permukaan, statistik multivariat, analisis spektral, analisis citra, identifikasi sistem, dan tugas-tugas analisis lainnya.

- **Visualisasi data**

MATLAB menyediakan *fitur build-in 2-D* dan *3-D*, Perencanaan fungsi, serta fungsi visualisasi volume. Anda dapat menggunakan fungsi ini untuk memvisualisasikan dan memahami data dan mengkomunikasikan hasil-hasil. Plot dapat disesuaikan baik secara interaktif atau pemrograman.

MATLAB memiliki Galeri Plot yang menyediakan banyak contoh serta cara untuk menampilkan data grafis dalam MATLAB.

- **Mendokumentasikan dan Berbagi Hasil**

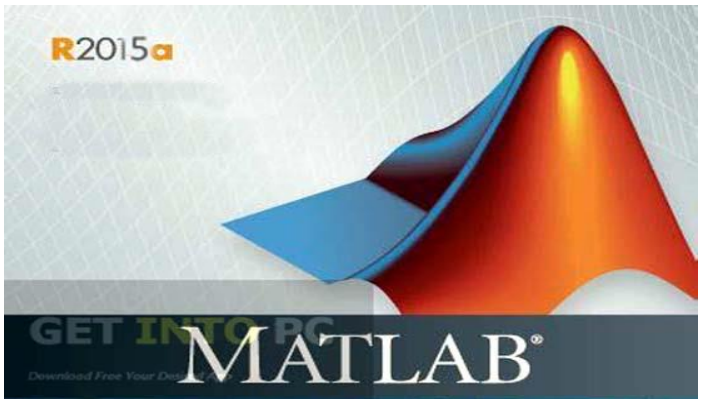
MATLAB dapat mngeluarkan hasil dalam bentuk plot atau laporan lengkap. Plot dapat dihasilkan, dapat disesuaikan untuk memenuhi spesifikasi publikasi dan disimpan ke format grafis dan data *file* yang umum .

MATLAB dapat secara otomatis menghasilkan laporan program dijalankan. Laporan tersebut berisi kode , komentar , dan hasil program yang

dibuat, termasuk plot. Laporan dapat diterbitkan dalam berbagai format , seperti HTML , PDF , Word, atau LaTeX .

- **Pemrograman dan Pengembangan Algoritma**

MATLAB menyediakan bahasa dan pengembangan tingkat tinggi alat yang memungkinkan pengembangan dan analisis algoritma secara cepat dan efektif.



Gambar 2.6 Tampilan awal *LabVIEW* 2015