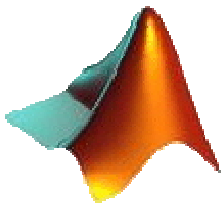
MATLAB TUTORIAL



Dasar Pengolahan Citra menggunakan MATLAB

TENTANG MATLAB

MATLAB adalah sebuah bahasa dengan (high-performance) kinerja tinggi untuk komputasi masalah teknik. Matlab mengintegrasikan komputasi, visualisasi, dan pemrograman dalam suatu model yang sangat mudah untuk pakai dimana masalah-masalah dan penyelesaiannya diekspresikan dalam notasi matematika yang familiar. Penggunaan Matlab meliputi bidang–bidang:

- Matematika dan Komputasi
- Pembentukan Algorithm
- Akusisi Data
- Pemodelan, simulasi, dan pembuatan prototipe
- Analisa data, explorasi, dan visualisasi
- Grafik Keilmuan dan bidang Rekayasa

MATLAB merupakan suatu sistem interaktif yang memiliki elemen data dalam suatu array sehingga tidak lagi kita dipusingkan dengan masalah dimensi. Hal ini memungkinkan kita untuk memecahkan banyak masalah teknis yang terkait dengan komputasi, kususnya yang berhubungan dengan matrix dan formulasi vektor, yang mana masalah tersebut merupakan momok apabila kita harus menyelesaikannya dengan menggunakan bahasa level rendah seperti Pascall, C dan Basic.

Nama MATLAB merupakan singkatan dari *matrix laboratory*. MATLAB pada awalnya ditulis untuk memudahkan akses perangkat lunak matrik yang telah dibentuk oleh LINPACK dan EISPACK. Saat ini perangkat MATLAB telah menggabung dengan LAPACK dan BLAS library, yang merupakan satu kesatuan dari sebuah seni tersendiri dalam perangkat lunak untuk komputasi matrix.

Dalam lingkungan perguruan tinggi teknik, Matlab merupakan perangkat standar untuk memperkenalkan dan mengembangkan penyajian materi matematika, rekayasa dan kelimuan. Di industri, MATLAB merupakan perangkat pilihan untuk penelitian dengan produktifitas yang tingi, pengembangan dan analisanya.

Fitur-fitur MATLAB sudah banyak dikembangkan, dan lebih kita kenal dengan nama *toolbox*. Sangat penting bagi seorang pengguna Matlab, toolbox mana yang mandukung untuk *learn* dan *apply* technologi yang sedang dipelajarinya. Toolbox toolbox ini merupakan kumpulan dari fungsi-fungsi MATLAB (M-files) yang telah dikembangkan ke suatu lingkungan kerja MATLAB untuk memecahkan masalah dalam kelas particular. Area-area yang sudah bisa dipecahkan dengan toolbox saat ini meliputi pengolahan sinyal, system kontrol, neural networks, fuzzy logic, wavelets, dan lain-lain.

Kelengkapan pada Sistem MATLAB

Sebagai sebuah system, MATLAB tersusun dari 5 bagian utama:

1. **Development Environment.** Merupakan sekumpulan perangkat dan fasilitas yang membantu anda untuk menggunakan fungsi-fungsi dan file-file MATLAB. Beberapa perangkat ini merupakan sebuah graphical user interfaces (GUI). Termasuk didalamnya adalah MATLAB desktop dan Command Window, command history, sebuah editor dan debugger, dan browsers untuk melihat help, workspace, files, dan search path.
2. **MATLAB Mathematical Function Library.** Merupakan sekumpulan algoritma komputasi mulai dari fungsi-fungsi dasar sepertri: sum, sin, cos, dan complex arithmetic, sampai dengan fungsi-fungsi yang lebih kompek seperti matrix inverse, matrix eigenvalues, Bessel functions, dan fast Fourier transforms.
3. **MATLAB Language.** Merupakan suatu high-level matrix/array language dengan control flow statements, functions, data structures, input/output, dan fitur-fitur object-oriented programming. Ini memungkinkan bagi kita untuk melakukan kedua hal baik "pemrograman dalam lingkup sederhana " untuk mendapatkan hasil yang cepat, dan "pemrograman dalam lingkup yang lebih besar" untuk memperoleh hasil-hasil dan aplikasi yang komplek.
4. **Graphics.** MATLAB memiliki fasilitas untuk menampilkan vector dan matrices sebagai suatu grafik. Didalamnya melibatkan high-level functions (fungsi-fungsi level tinggi) untuk visualisasi data dua dikensi dan data tiga dimensi, image processing, animation, dan presentation graphics.

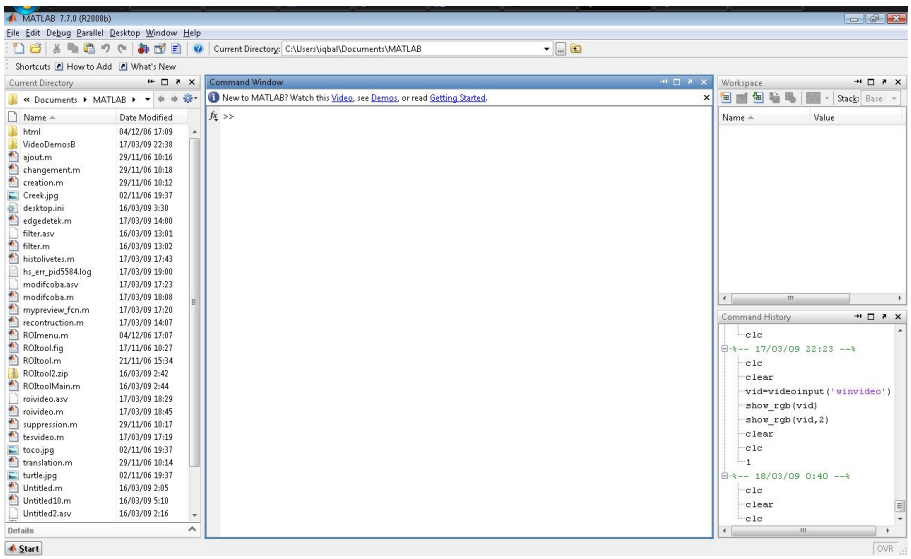
Ini juga melibatkan fungsi level rendah yang memungkinkan bagi anda untuk membiasakan diri untuk memunculkan grafik mulai dari benutk yang sederhana sampai dengan tingkatan graphical user interfaces pada aplikasi MATLAB anda.

5. **MATLAB Application Program Interface (API).** Merupakan suatu library yang memungkinkan program yang telah anda tulis dalam bahasa C dan Fortran mampu berinterakasi dengan MATLAB. Ini melibatkan fasilitas untuk pemanggilan routines dari MATLAB (dynamic linking), pemanggilan MATLAB sebagai sebuah computational engine, dan untuk membaca dan menuliskan MAT-files.

Memulai Matlab

Perhatikan Dekstop pada layar monitor PC, anda mulai MATLAB dengan melakukan *double-clicking* pada *shortcut icon*  MATLAB.

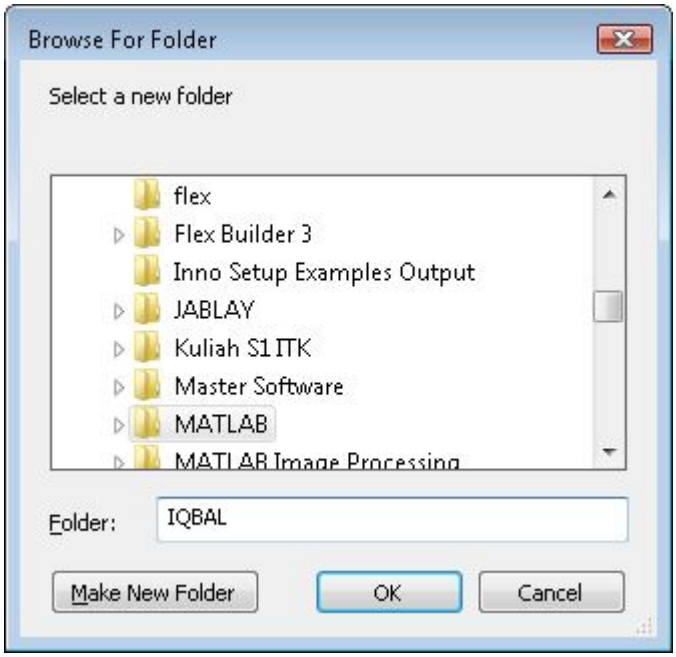
Selanjutnya anda akan mendapatkan tampilan seperti pada Gambar berikut ini.



Sedangkan untuk mengakhiri sebuah sesi MATLAB, anda bisa melakukan dengan dua cara, pertama pilih **File -> Exit MATLAB** dalam window utama MATLAB yang sedang aktif, atau cara kedua lebih mudah yaitu cukup ketikkan type ***quit*** dalam Command Window.

Menentukan Direktori Tempat Bekerja

Anda dapat bekerja dengan MATLAb secara default pada directory Work ada di dalam Folder MATLAB. Tetapi akan lebih bagus dan rapi jika anda membuat satu directory khusus dengan nama yang sudah anda kususkan, “IQBAL” atau nama yang lain yang mudah untuk diingat. Hal ini akan lebih baik bagi anda untuk membiasakan bekerja secara rapi dan tidak mencampur program yang anda buat dengan program orang lain. Untuk itu Arahkan pointer mouse anda pada kotak bertanda … yang ada disebelah kanan tanda panah kebawah (yang menunjukkan folder yang sedang aktif). Pilih new directory, selanjutnya ketikkan “IQBAL”, dan diikuti dengan click Ok.



Memulai Perintah Sederhana

Langkah kita yang pertama adalah dengan menentukan variable scalar dengan cara melakukan pengetikan seperti berikut:

```
» x = 2  (selanjutnya tekan “Enter”)
x =
     2

» y = 3
y =
     3
» z = x + y
z =
     5
```

Tidak terlalu menjadi masalah bagi anda? Saya percaya anda tidak mengalami kesulitan,
sebab anda adalah orang yang sangat cerdas. Nah bagaimana dengan yang satu berikutnya ini? Disini kita mulai dengan mendefinisikan dua buah vector, yaitu vector x dan vector y:

```
» x = [1 2 3]
x =
1 2 3
» y = [4 5 6]
y =
4 5 6
```

Selajutnya ketik:

```
>> y(1)
ans =
4
```

dan ulangi untuk y(2) and y(3).

Matlab menggunakan integer positif untuk index array. Elemen pertama adalah **`y(1)`**, elemen kedua adalah **`y(2)`**, dan seterusnya. Nol atau bilangan negatif tidak diperbolehkan untuk indek array. Sekarang kita jumlahkan keduanya:

```
» x+y
ans =
5 7 9
```

dan sekarang hitung inner product:

```
» x*y'
ans =
      32
```

Jawabannya adalah 1*4 + 2*5 + 3*6 = 32! Catat, bahwa **y'** adalah transpose pada **y** dan merupakan suatu vector kolom. Untuk memeriksanya, ketikkan perintah berikut:

```
>> y'
ans =
     4
     5
    6
```

Cara lain pada pengkombinasian dua vector adalah diakukan melalui perkalian elementdemi- element:

```
>> x.*y
ans =
    4 10 18
```

Catat periode sebelum perkalian simbol. Sekarang kita dapat mendefinisikan suatu matrix:

```
» A = [1 2 3
       4 5 6
       7 8 9];
```

Catat bahwa matrik tidak diulang kalau kita menggunakan semi colon. Kita sekarang kalikan A dengan transpose dari x:

```
» A*x'
ans =
      14
      32
      50
```

Sekarang kita harus mentranspose x untuk memenuhi perkalian suatu matrik dan suatu vector kolom. Matrik-matrik ini dapat juga dikalikan satu sama lain diantara mereka:

```
» B = [1 2 3 4
       5 6 7 8
       7 6 5 4];
» A*B
ans =
      32 32 32 32
      71 74 77 80
      110 116 122 128
```

Sekarang coba anda lakukan penjumlahan antara A dan B:

```
» A+B
??? Error using ==> +
Matrix dimensions must agree.
```

Baiklah, kita tidak dapat menambah suatu matrik 3 kali 3 dengan matrix 3 kali 4 , dan Matlab akan mendeteksi dimensi yang mismatch dan selanjutnya memeberikan pesan error. Sekarang kita cari cara lain untuk mendefinisikan matrik dan vektor. Sebagai contoh suatu matrik nol dengan dimensi 3 baris dan 6 kolom dapat dinyatakan sebagai:

```
>> zeros(3,6)
ans =
       0 0 0 0 0 0
       0 0 0 0 0 0
       0 0 0 0 0 0
```

tentu saja jika anda tambahkan suatu ";" setelah zeros(3,6), jawabannya tidak akan ditampilkan di layar monitor anda. Angka pertama, 3 menunjukkan jumah baris, sedangkan angka kedua, 6, adalah jumlah kolom. Kita dapat pula melakukan hal yang sama untuk menampilkan angka satu seperti berikut:

```
>> ones(3,6)
ans =
       1 1 1 1 1 1
       1 1 1 1 1 1
       1 1 1 1 1 1
```

Pendefinisian Vektor-vektor Besar

Suatu vektor 1 kali 100 yang menyusun sample pada sinyal cosinus dapat dibangkitkan dengan

```
>> x = cos(0.1*pi*(0:99));
```

Untuk membangkitkan suatu "ramp" dari 1 sampai 50 coba:

```
>> x = [1:1:50];
```

bilangan kedua mengindikasikan step kenaikan dari dari 1 sampai 50. Untuk membangkitkan suatu fungsi "ramp" dari 1 sampai 50 coba berikut ini:

```
>> x = [1:1:50];
```

Ketika anda tidak memasukkan angka kedua pada perintah diatas, maka secara otomatis (default) step kenaikan ditetapkan bernilai “1”:

```
>> x = [1:50];
```

Anda bisa juga secara khusus mendefinisikan suatu rentang nilai pada x sebagai berikut::

```
>> x(51:100) = [50:-1:1]
```

Ini merupakan metode yang sangat bermanfaat untuk mensepsifikasi nilai “waktu” untuk penggambaran. Sebagai contoh, ditetapkan interval sampling dalam contoh diatas adalah 1 detik. Selanjutnya anda dapat mendefisnisikan seperti berikut:

```
>> time = [0:0.001:0.099];
```

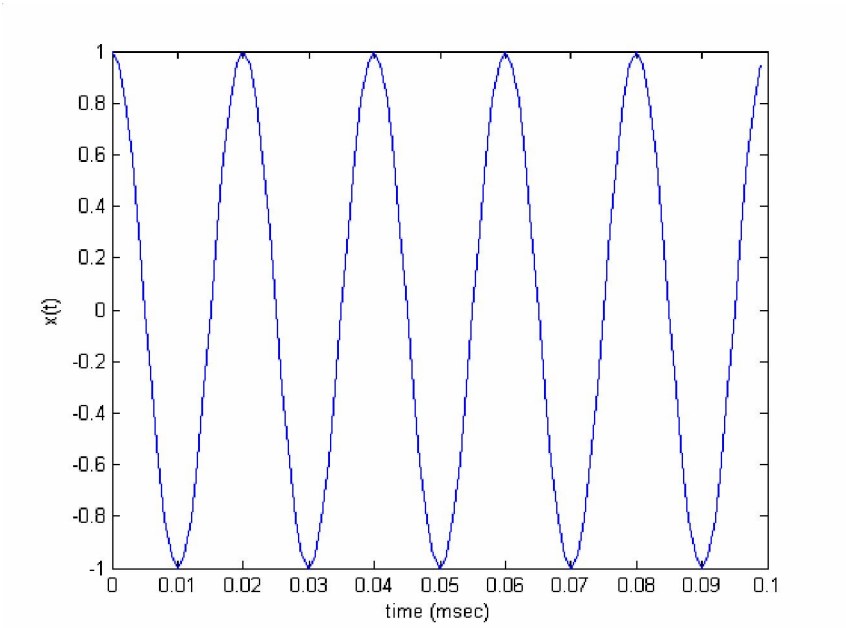
Penggambaran Grafik

Salah satu kelebihan dari Matlab adalah kemudahan dalam mengolah grafik. Sehingga anda tidak perlu kesulitan untuk melihat suatu respon system, misalnya pada kasus melijhat bentuk sinyal dalam domain waktu anda cukup mengikuti langkah berikut.

Sekarang ketikkan:

```
>> time = [0:0.001:0.099];
>> x = cos(0.1*pi*(0:99));
>> plot(time,x)
>> xlabel('time (msec)')
>> ylabel('x(t)')
```

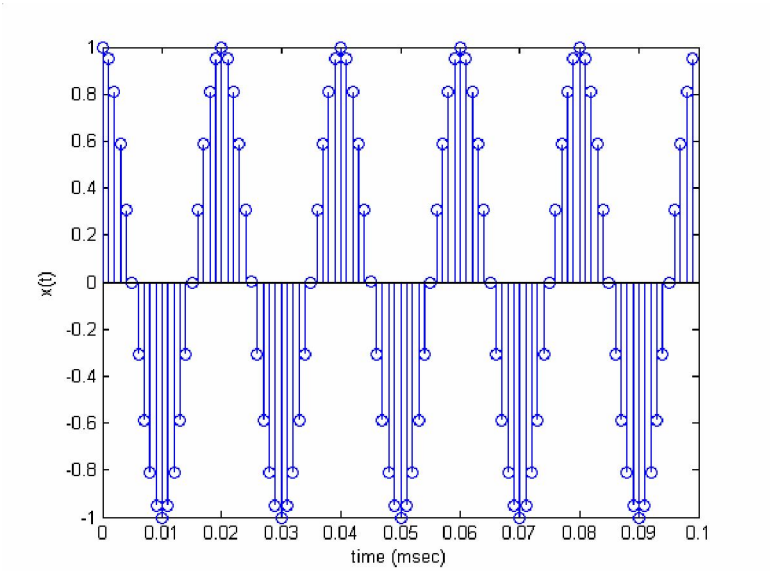
ini akan menghasilkan gambar seperti berikut:



Sedangkan cara untuk menampilkan sederetan nilai fungsi waktu diskrit adalah dengan menggunakan perintah "stem". Dari contoh deretan perintah coba anda rubah beberapa bagian dengan perintah berikut

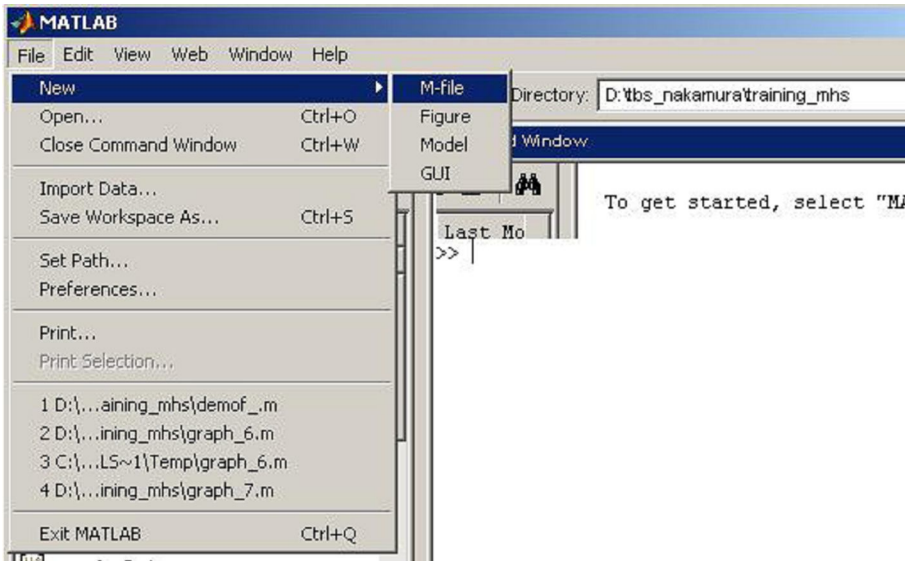
```
>> stem(time,x)
>> xlabel('time (msec)')
>> ylabel('x(t)')
```

Apakah hasilnya seperti berikut ini?

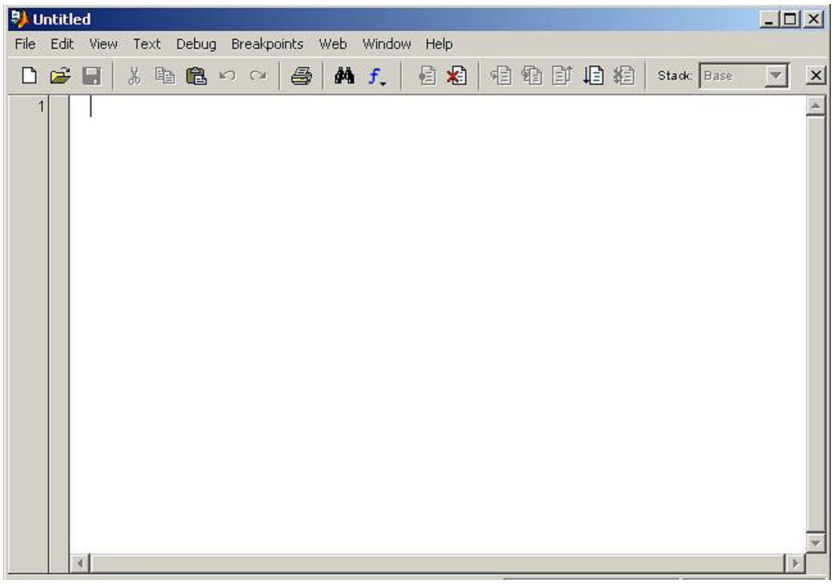


Menyusun Progam Sederhana

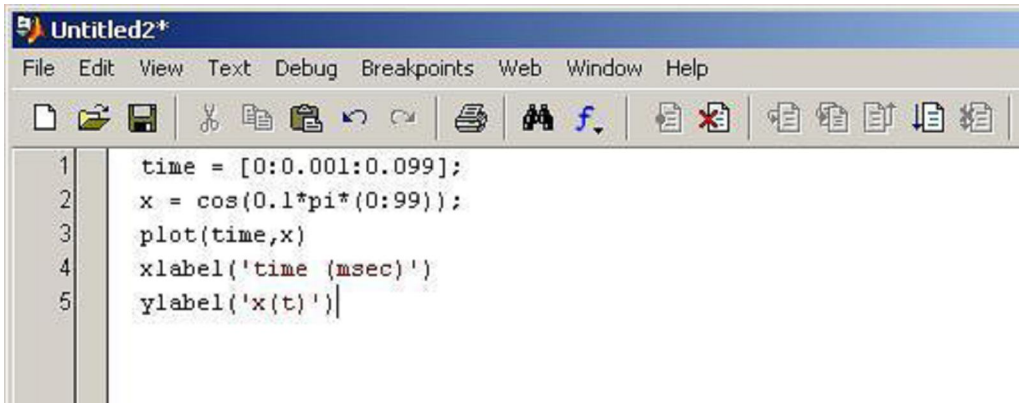
Anda dapat mengedit suatu file text yang tersusun dari beberapa perintah Matlab. Ini dapat dilakukan dengan menekan double-click pada icon "New M-File" icon in the Matlab toolbar.

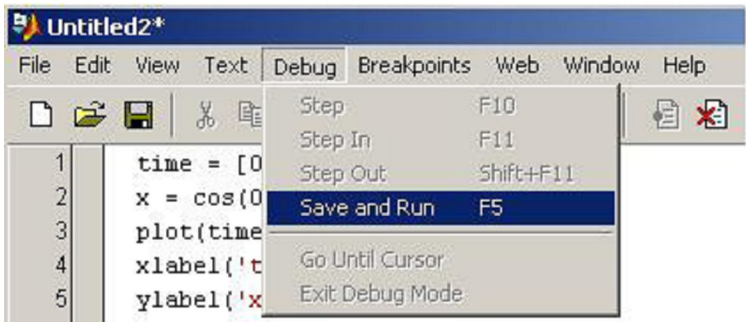


Selanjutnya anda akan mendapatkan sebuah tampilan Matlab Editor yang masih kosong seperti ini.



Selanjutnya anda buat program seperti pada contoh sebelumnya





Lanjutkan dengan menekan toolbar Debug, dan jangan lupa anda pilih Save anda Run. Disitu anda harus menuliskan nama program. Anda tuliskan coba_1, secara otomatis akan menjadi file coba_1.m dan akan anda lihat tampilan hasilnya. Seperti apa ya?

Program kedua anda

Cobalah untuk membuat program seperti berikut ini pada Matlab editor, dan jangan lupa anda simpan dengan nama coba_2

```
x(1:52) = [0 0 1:1:50];
x(53:102) = [50:-1:1];
h = [1 2];
for n = 3:101,
y(n) = 0;
for m = 1:2,
y(n) = y(n) + h(m)*x(n-m);
end
end
plot(y)
```

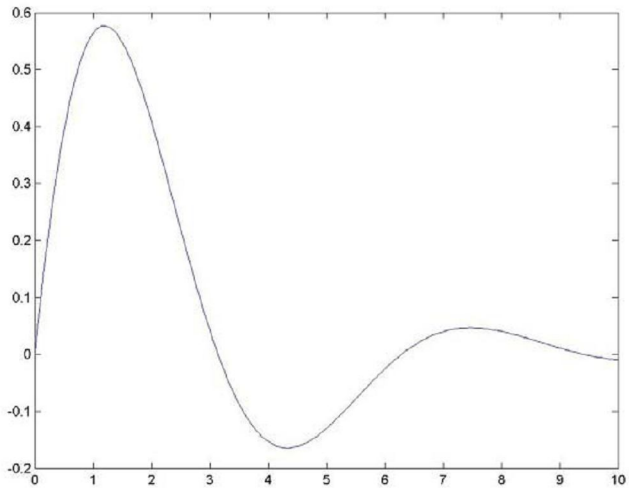
Hasil apa yang anda dapatkan ?

Dalam hal ini anda harus memahami arti setiap perintah yang anda tuliskan dalam Matlab, tidak ada salahnya anda bertanya kepada instruktur apa arti perintah-perintah tersebut.

Program ketiga anda

Satu contoh lain program untuk for adalah pembangkitan gambar seperti berikut.

```
%File Name:coba_3.m
n=201;
delx=10/(n-1);
for k=1:n
x(k)=(k-1)*delx;
y(k)=sin(x(k))*exp(-0.4*x(k));
end
%plot(x,y)
plot(x,y,'linewidth',4)
title('Grafik yang pertama')
xlabel('x');ylabel('y');
```



Bagiamana hasilnya?

Fungsi dalam Matlab

Matlab juga mampu untuk menuliskan fungsi yang didefinisikan oleh pemakainya. Buat sebuah fungsi dengan menuliskan program berikut ini:

```
function y = x2(t)
y = t^2;
```

Anda simpan dengan nama "x2.m" selanjutnya anda dapat memanfaatkan fungsi tersebut melalui Matlab line command dengan cara berikut:

```
>>t=0:1:10;
>> y_2=x2(t)
```

Hasilnya adalah seperti berikut:

```
y_2 =
        0 1 4 9 16 25 36 49 64 81 100
```

Anda bisa juga memanggil fungsi tersebut melalui program pada panggil_1.m file yang anda buat seperti berikut:

```
t=0:1:10;
y_2=x2(t)
```

Hasilnya adalah sama seperti menggunakan command line window.

Pendahuluan

Paper ini akan menjelaskan tentang dasar-dasar pengolahan citra menggunakan MATLAB. Seperti telah diketahui bahwa MATLAB merupakan bahasa komputasi yang memilki banyak sekali fungsi *built-in* berkaitan dengan matrik dan persamaan-persamaan yang biasa digunakan pada bidang tertentu sehingga sangat memudahkan pengguna untuk melakukan pemrograman sehingga pengguna tidak terlalu dipusingkan dengan logika pemrograman dan lebih fokus terhadap logika penyelesaian masalah yang dihadapi.

Apa itu digital image processing?

Image atau gambar adalah representasi spasial dari suatu objek yang sebenarnya dalam bidang dua dimensi yang biasanya ditulis dalam koordinat kartesian x-y, dan setiap koordinat merepresentasikan satu sinyal terkecil dari objek yang biasanya koordinat terkecil ini disebut sebagai piksel. Karena merupakan sistem koordinat yang memiliki nilai maka biasanya image dianggap sebagai sebuah matrik x-y yang berisi nilai piksel.

Represntasi dari matriks tersebut dapat ditulis sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Dan di MATLAB representasi ini biasa ditulis menjadi

$$\mathtt{f} = \begin{bmatrix} \mathtt{f(1,1)} & \mathtt{f(1,2)} & \cdots & \mathtt{f(1,N)} \\ \mathtt{f(2,1)} & \mathtt{f(2,2)} & \cdots & \mathtt{f(2,N)} \\ \vdots & \vdots & & \vdots \\ \mathtt{f(M,1)} & \mathtt{f(M,2)} & \cdots & \mathtt{f(M,N)} \end{bmatrix}$$

Yang perlu diperhatikan adalah bahwa indeks matriks pada MATLAB selalu dimulai dengan anggak 1 sehingga untuk f(0,0) akan sama dengan f(1,1) pada matlab.

Bentuk matrik ini kemudian diolah menurut teori-teori tertentu yang bertujuan untuk memecahkan masalah tertentu, bentuk matriks adalah perwujudan dari bentuk sinyal digital sehingga proses pemecahan dan pengolahan matriks dari gambar ini biasanya disebut dengan digital image processing.

Pembacaan Image

Pada matlab fungsi untuk melakukan pembacaan image standar yaitu:

imread('filename')

Perintah ini digunakan untuk membaca beberapa format file diantaranya:

Format	Deskripsi	Recognized Extension	
TIFF	Tagged Image File Format	.tif	.tiff
JPEG	Join Photographics Expert's Group	.jpg	.jpeg
GIF	Graphics Interchange Format	.gif	
BMP	Windows Bitmap	.bmp	
PNG	Portable Network Graphics	.png	
XWD	X-Window Dump	.xwd	

Hasil dari pembacaan imread('filename') bisa berupa matriks dua dimensi jika gambar yang dibaca adalah gambar grayscale dan matrik 3 dimensi jika berupa gambar 3 dimensi.

Ekstraksi Nilai Piksel Red, Green dan Blue (RGB)

Hampir setiap pengolahan citra yang berbasis warna perlu dilakukan pemisahan band-band yang ada pada citra khususnya citra RGB, MATLAB menyediakan fasilitas yang cukup baik dalam memisahkan ketiga warna RGB, yaitu sebagai berikut:

```
gambar=imread('gambarkoe.jpg'); %--------membaca file gambar
red=gambar(:,:,1); %memanggil matriks gambar yang hanya berisi piksel warna
merah
green=gambar(:,:,2);% memanggil matriks gambar yang hanya berisi piksel
warna merah
blue=gambar(:,:,3); %memanggil matriks gambar yang hanya berisi piksel
warna merah
%----------menampilkan gambar---------------------
imshow(gambar)
imshow(red)
imshow(green)
imshow(blue)
```

Terlihat bahwa untuk mengambil nilai piksel merah memiliki indeks 1, warna hijau memiliki indeks 2 dan warna biru memiliki indeks 3.

Konversi Gambar RGB ke Grayscale

Untuk merubah gambar RGB ke gambar grayscale di MATLAB disediakan fungsi khusus yaitu *rgb2gray(matrik_gambar)*, tetapi kadangkala diinginkan untuk perubahan bentuk grayscale ini tidak menggunakan fungsi MATLAB yang sudah ada yang merupakan nilai rata-rata piksel RGB tetapi masing-masing nilai RGB diberi nilai bobot yang berbeda-beda, hal ini dengan mudah dilakukan dengan menggunakan pemisahan nilai seperti yang telah dilakukan diatas seperti contoh berikut:

```
gambar=imread('gambarkoe.jpg'); %--------membaca file gambar
red=gambar(:,:,1); %memanggil matriks gambar yang hanya berisi piksel warna
merah
green=gambar(:,:,2);% memanggil matriks gambar yang hanya berisi piksel
warna merah
blue=gambar(:,:,3); %memanggil matriks gambar yang hanya berisi piksel
warna merah
gray2=0.3*red+0.5*green+0.2*blue ;
```

Membuat Histogram Image

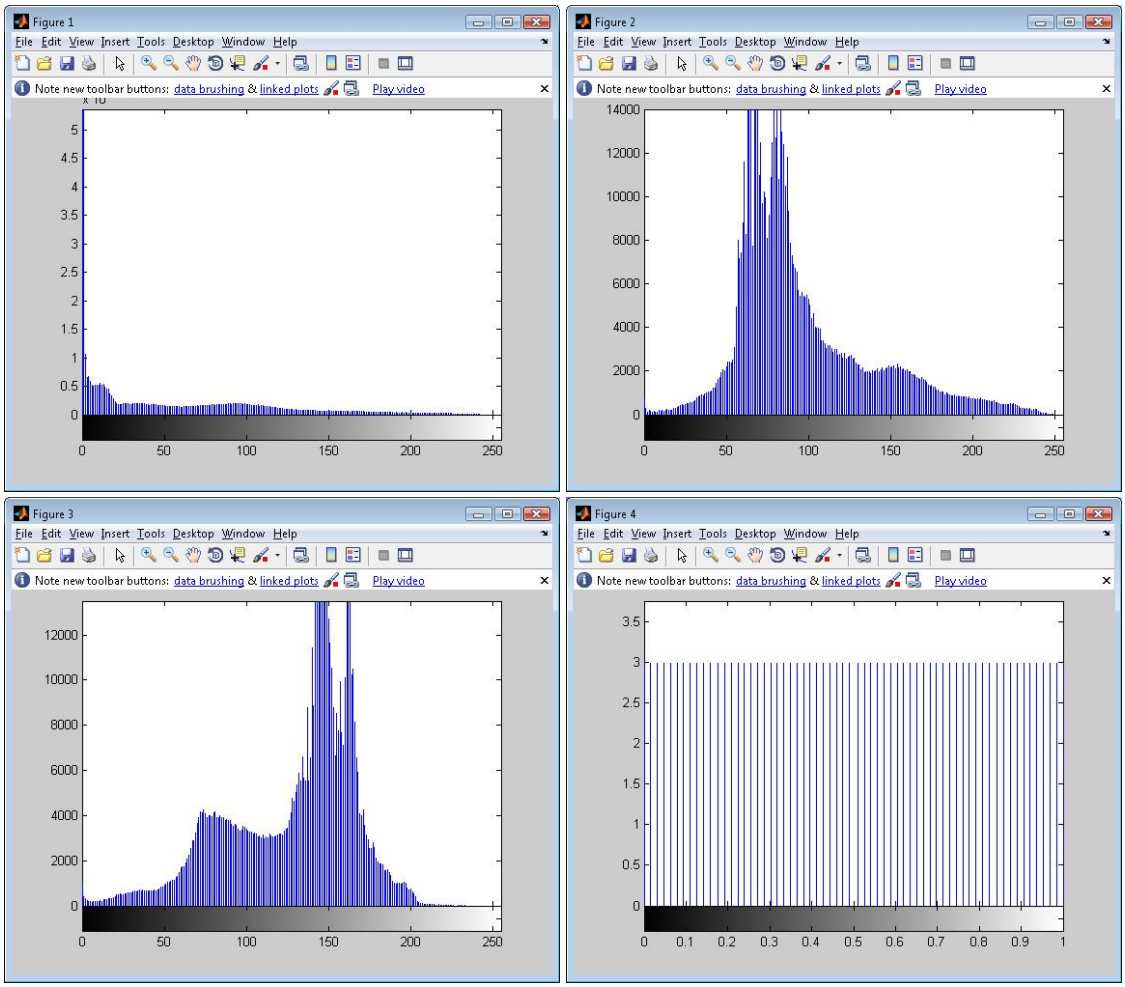
Fungsi yang disediakan MATLAB untuk membuat histogram dari gambar yaitu dengan fungsi *imhist(matrik_1_dimensi_image)*

Perlu diperhatikan bahwa *imhist* hanya dapat digunakan untuk matrik image 1 dimensi sehingga bila diimplementasikan pada matriks gambar maka hanya berupa matriks merah saja, hijua saja, biru saja atau grayscale.

Contoh penggunaan Histogram dari Image yaitu:

```
gambar=imread('gambarkoe.jpg'); %--------membaca file gambar
red=gambar(:,:,1); %memanggil matriks gambar yang hanya berisi piksel warna
merah
green=gambar(:,:,2);% memanggil matriks gambar yang hanya berisi piksel
warna merah
blue=gambar(:,:,3); %memanggil matriks gambar yang hanya berisi piksel
warna merah
merahgray2=0.3*red+0.5*green+0.2*blue ;
imhist(red)
imhist(green)
imhist(blue)
imhist(gray)
```

contoh Hasil:



Crop Image

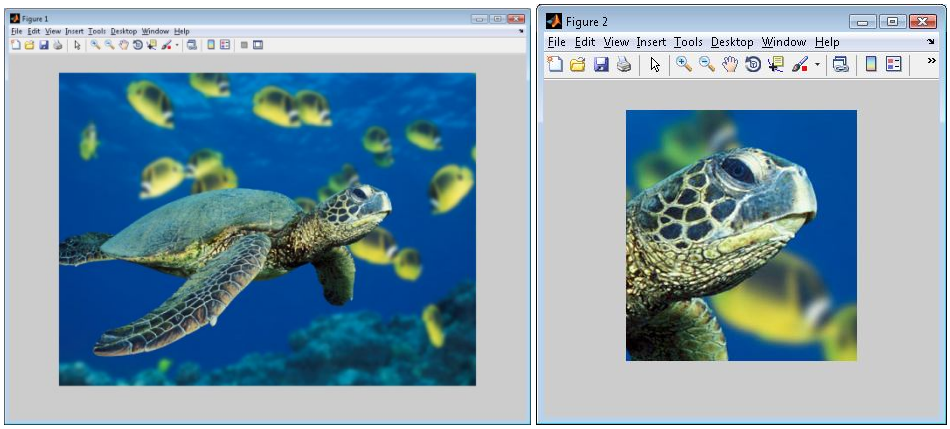
Matlab juga menyediakan fungsi untuk melakukan cropping (pemotongan bagian tertentu dari gambar menjadi matrik baru yang independen). Fungsi tersebut yaitu

Imcrop(matrik_gambar,matrik_titiksudut_crop);

Contoh implementasinya adalah:

```
gambar=imread('turtle.jpg');
crop=imcrop(gambar,[627 237 230 250])
imshow(gambar), figure, imshow(crop)
```

contoh hasil keluaranya dari program diatas yaitu:



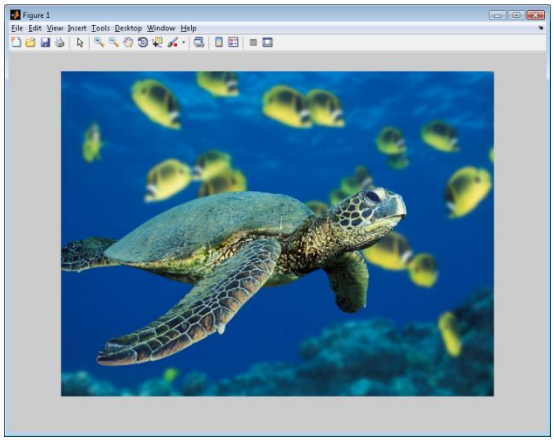
Penentuan titik yang akan diambil yaitu menggunakan *matrik_titiksudut_crop* yaitu yang merepresentasikan nilai [x,y,a,b] dimana x dan y adalah titik awal (sudut kiri atas) dari image yang akan dikrop sedangakan a adalah jumlah piksel memanjang kearah sumbu-x dan b adalah jumlah piksel ke arah sumbu-y.

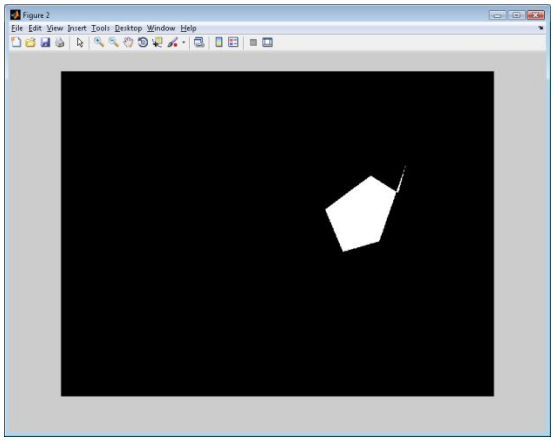
Region Of Interest (ROI)

Kelemahan proses cropping jika itu merupakan daerah yang kita tertarik maka proses cropping hanya dapat digunakan untuk bentuk kotak (*rectangular*). Untuk bentuk lain atau area yang berbentuk tidak beraturan yang ingin dipisahkan dari image induk maka didefinisikan sebagai ROI (*Region of Interest*) dimana di MATLAB terdapat banyak sekali fungsi yang bisa digunakan, salah satunya yaitu roipoly(I,c,r) dimana I adalah matrik gambar, c adalah matrik titik kolom daerah yang menjadi ROI dan r adalah matrik titik baris daerah yang menjadi ROI.

Contoh penggunaan ROI yaitu:

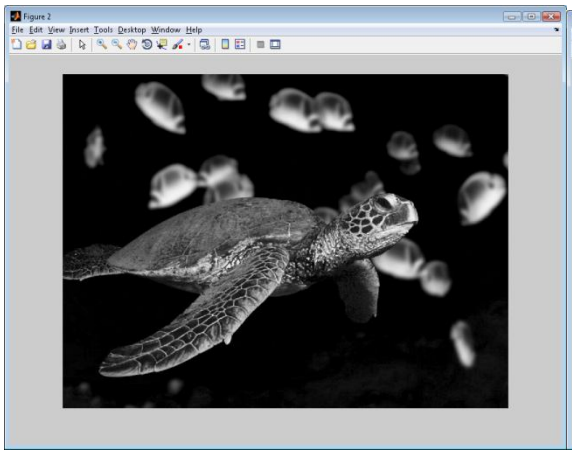
```
I=imread('turtle.jpg');
c = [625 685 733 798 816 753 667];
r = [327 282 247 288 221 402 427];
BW = roipoly(I,c,r);
figure, imshow(I)
figure, imshow(BW)
```

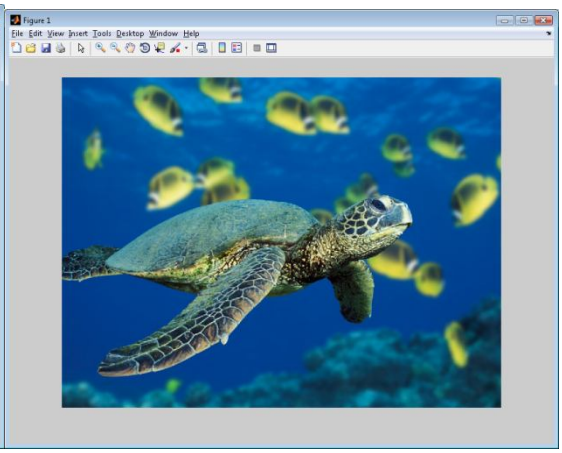


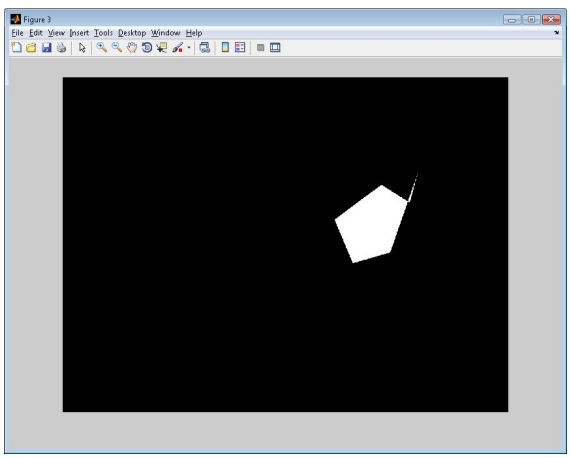


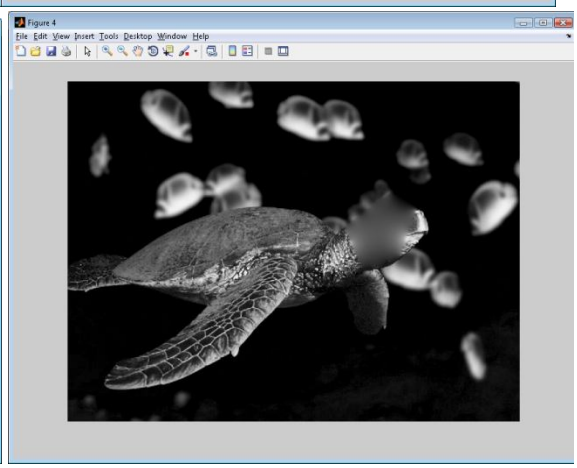
Contoh implementasi real ROI

```
gambar=imread('turtle.jpg');
I=gambar(:,:,1);
c = [625 685 733 798 816 753 667];
r = [327 282 247 288 221 402 427];
BW = roipoly(I,c,r);
j = roifill(I,c,r);
figure, imshow(gambar)
figure, imshow(I)
figure, imshow(BW)
figure, imshow(j)
```









Pengolahan Citra dengan Domain Frekuensi

Pada Domain frekuensi, citra dinyatakan sebagai kombinasi dari gelombang penyusun dengan frekuensi berbeda.

Beberapa fungsi MATLAB yang biasa digunakan untuk pengolahan sinyal untuk domain frekuensi yaitu:

- fft, fft2
- dct, dct2

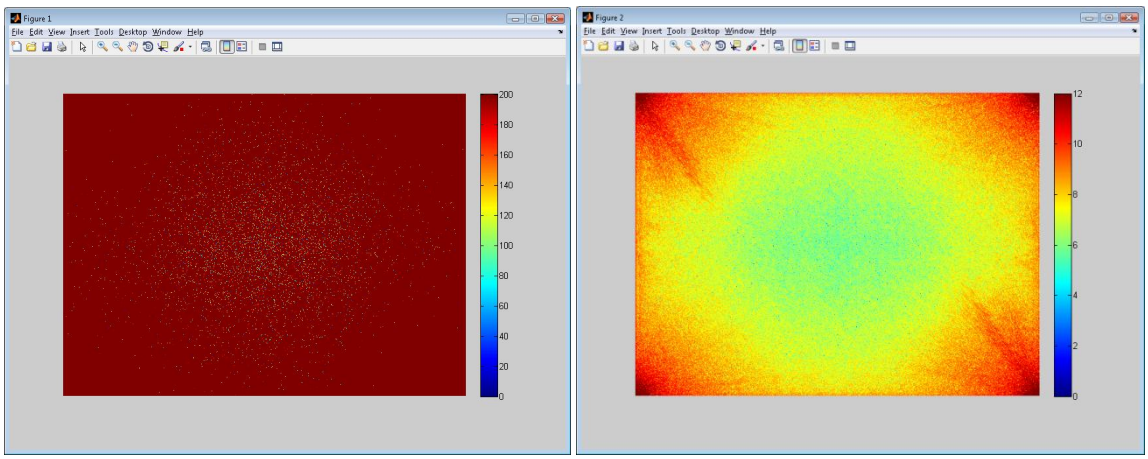
FFT (Fast Fourier Transform)

FFT didefiniskan berdasarkan persamaan berikut:

$$F(\omega_1,\omega_2) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m,n) e^{-j\omega_1 m} e^{-j\omega_2 n}$$

Ada dua cara untuk menampilkan hasil FFT yaitu berdasarkan magnitude yaitu $|F(\omega_1, \omega_2)|$ dan log dari FFT yaitu log $|F(\omega_1, \omega_2)|$. Berikut contoh program implementasi penggunaan fft:

```
gambar=imread('Toco.jpg');
red=gambar(:,:,1);
green=gambar(:,:,2);
blue=gambar(:,:,3);
f=fft2(gambar);
ff=abs(f);
flog=log(ff);
imshow(ff(:,:,3),[0 200]), colormap(jet),colorbar
figure,imshow(ff(:,:,3),[0 12]), colormap(jet),colorbar
```



DCT (Discrete Cosine Transform)

Persamaan DCT biasanya ditulis seperti dibawah ini:

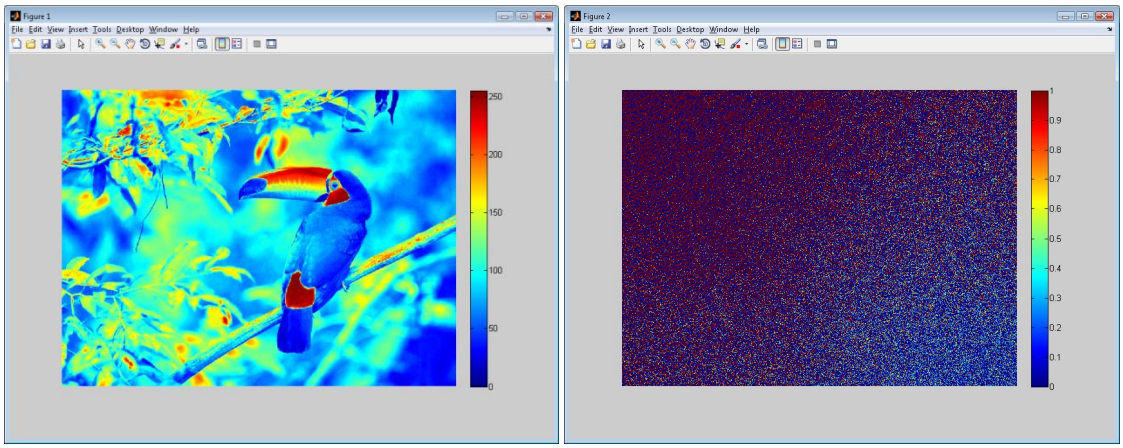
$$C_x(k) = \sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi}{2N} k(2n+1),\ 0 \le k \le N-1$$

DCT biasanya digunakan untuk kompresi karena mampu mengurangi terjadinya perulangan piksel yang sama pada daerah yang berdekatan.

Contoh penggunaan DCT yaitu:

```
gambar=imread('Toco.jpg');
gray=rgb2gray(gambar);
f=dct2(gray);
imshow(gray), colormap(jet),colorbar
figure,imshow(f), colormap(jet),colorbar
```

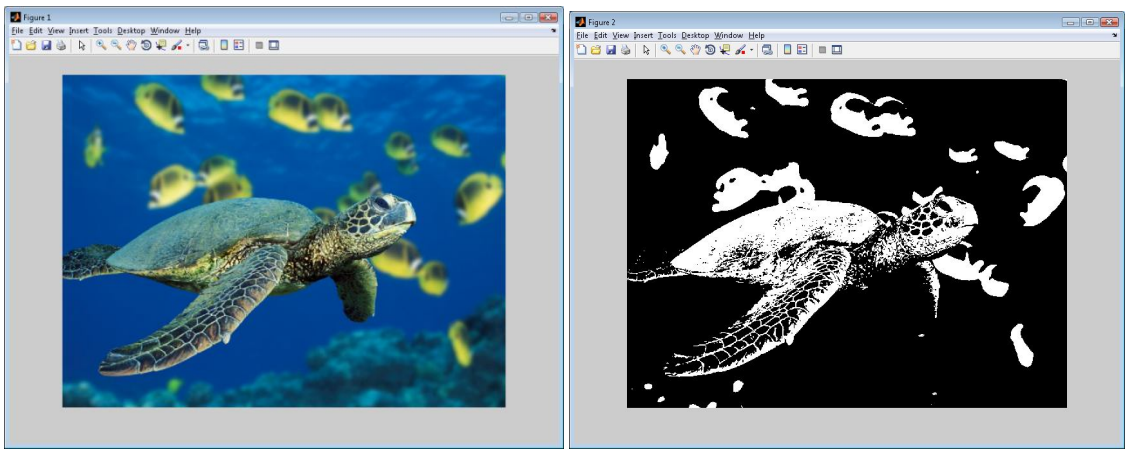
contoh hasilnya yaitu:



Konversi Citra ke Citra Biner

Binerisasi citra adalah salah satu proses penting yang biasanya dilakukan dalam pemrosesan image, MATLAB menyediakan beberapa fungsi untuk melakukan proses ini. Sebagai contoh proses tersebut seperti pada langkah dibawah ini:

```
gambar=imread('turtle.jpg');
gray=rgb2gray(gambar);
thresh=graythresh(gray);
imbw=im2bw(gray,thresh);
imshow(gambar)
figure, imshow(imbw)
```



Ada 2 fungsi penting dalam proses diatas yaitu `thresh=graythresh(gray);` yang digunakan untuk mendapatkan nilai ambang batas dan `imbw=im2bw(gray,thresh);` yang melakukan proses binerisasi citra itu sendiri.

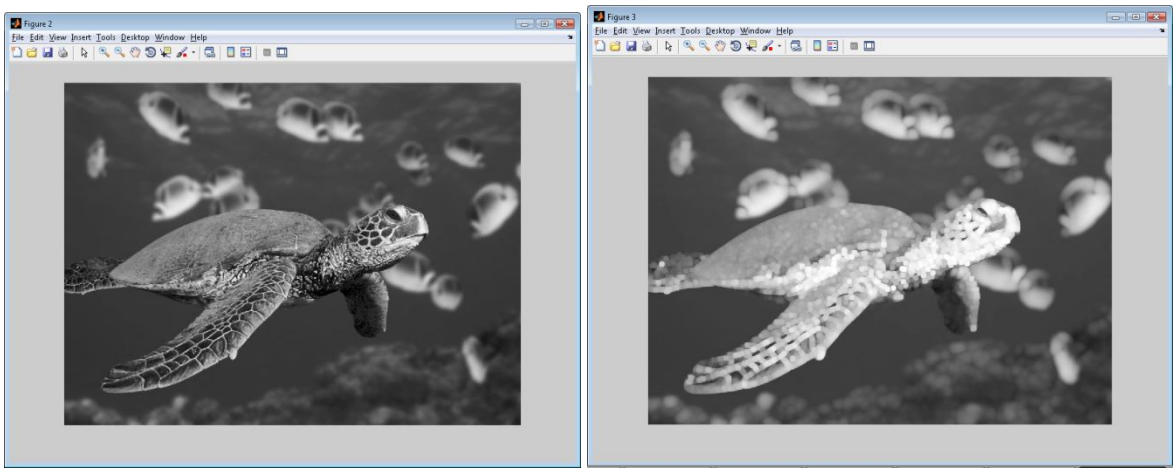
Morphological Image Processing

Merupakan pengolahan citra yang berhubungan dengan bentuk dan struktur dari suatu objek, ada beberapa contoh teknik yang digunakan seperti dilasi, erosi dan objek counting.

Dilasi

Contoh:

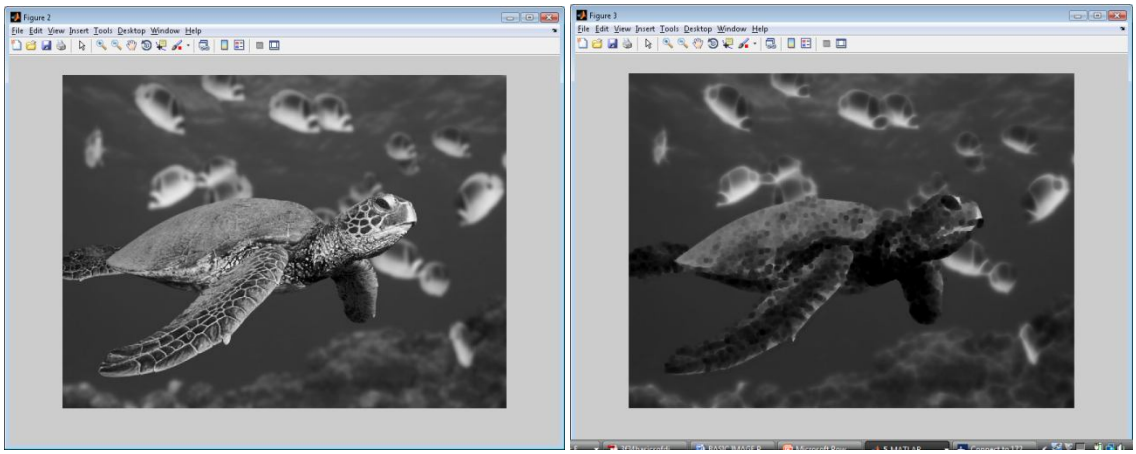
```
gambar=imread('turtle.jpg');
gray=rgb2gray(gambar);
se = strel('ball',5,5);
dilat=imdilate(gray,se);
imshow(gambar)
figure, imshow(gray)
figure, imshow(dilat)
```



EROSI

Contoh:

```
gambar=imread('turtle.jpg');
gray=rgb2gray(gambar);
se = strel('ball',5,5);
dilat=imerode(gray,se);
imshow(gambar)
figure, imshow(gray)
figure, imshow(dilat)
```



Object Counting

Yaitu proses menghitung objek berdasarkan konektivitasnya terhadapap piksel disekitarnya, bisa berdasarkan 4 piksel koneksi atau menggunakan 8 piksel koneksi.

Fungsi yang digunakan untuk menghitung objek yaitu:

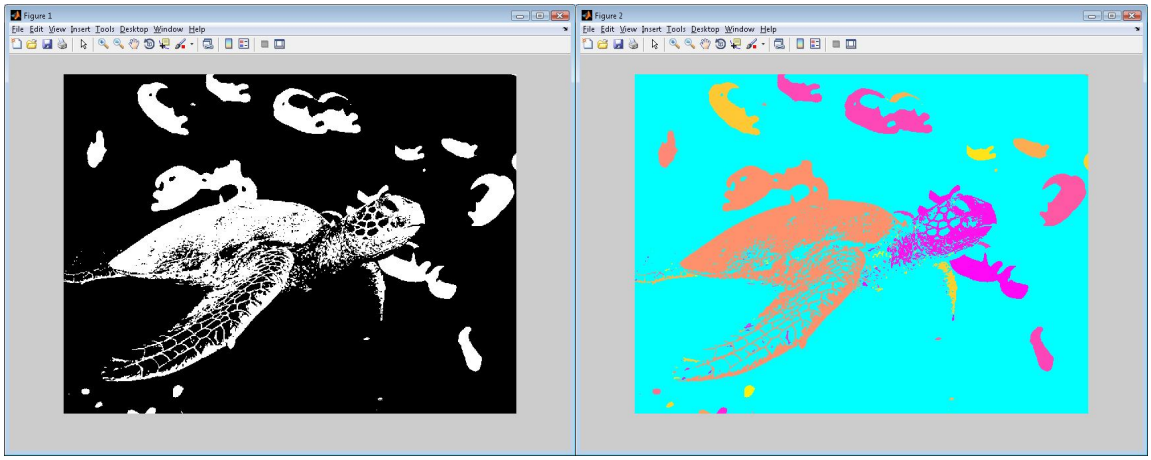
[labeled,numObjects] = bwlabel(imbw,4);

Sedangkan fungsi yang digunakan untuk memberi label dan warna yang berbeda pada setiap objek yaitu:

imlabel = label2rgb(labeled, @spring, 'c', 'shuffle');

untuk contoh implementasi dapat dilihat pada kode berikut:

```
gambar=imread('turtle.jpg');
gray=rgb2gray(gambar);
thresh=graythresh(gray);
imbw=im2bw(gray,thresh);
[labeled,numObjects] = bwlabel(imbw,8);
imlabel = label2rgb(labeled, @spring, 'c', 'shuffle');
imshow(imbw)
figure,imshow(imlabel)
```

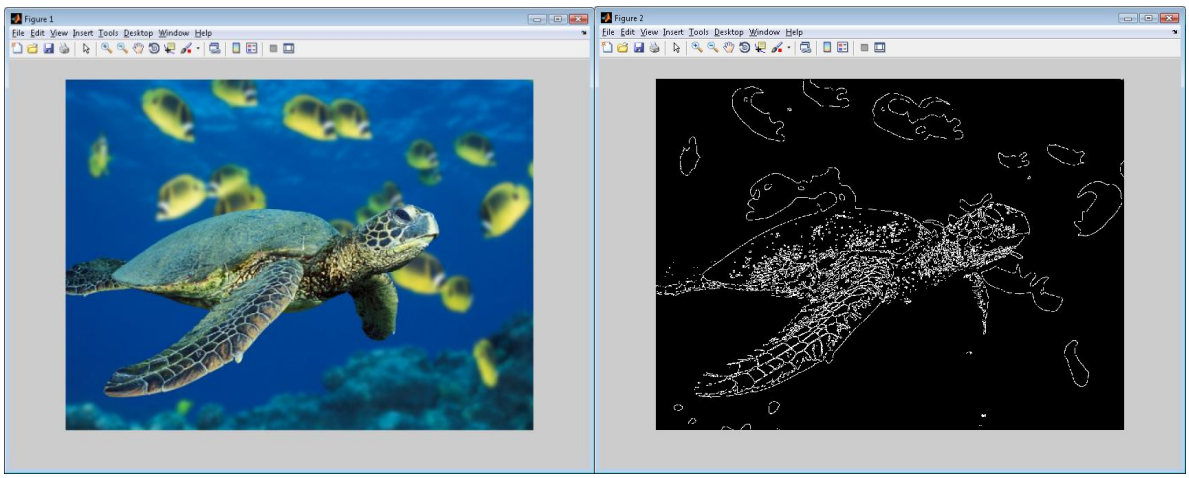


Konvolusi Image

Konvolusi adalah salah satu proses filtering image yang sering dilakukan pada proses pengolahan gambar. Pada MATLAB terdapat banyak sekali cara yang dapat dilakukan untuk melakukan proses konvolusi. Proses konvolusi dilakukan dengan menggunakan matriks yang biasa disebut *mask* yaitu matriks yang berjalan sepanjang proses dan digunakan untuk menghitung nilai representasi lokal dari beberapa piksel pada image.

Contoh implementasi konvolusi ini yaitu:

```
gambar=imread('turtle.jpg');
mask = [-1 -1 -1; -1 8 -1; -1 -1 -1];
gray=rgb2gray(gambar);
thresh=graythresh(gray);
imbw=im2bw(gray,thresh);
hasil=conv2(double(imbw),mask,'valid');
imshow(gambar)
figure, imshow(hasil)
```



Filtering Image

Proses filtering secara khusus oleh matlab menggunakan fungsi built-in fspecial(special filter) dimana syntax umumnya adalah

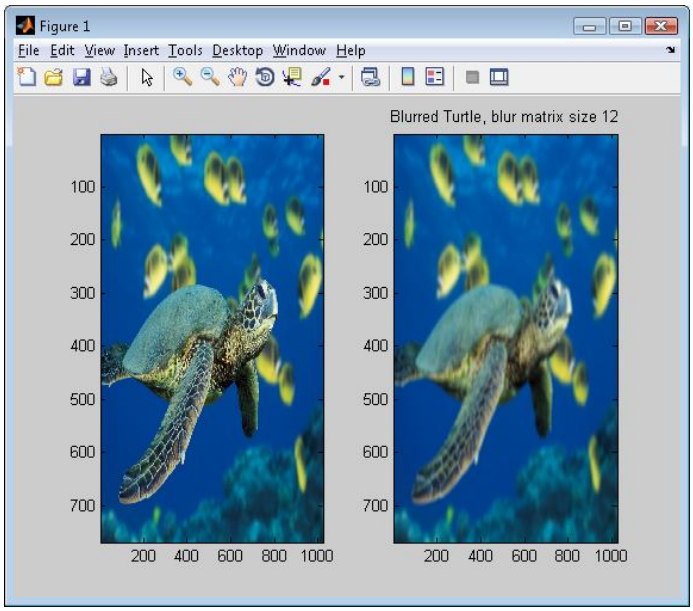
fspecial(filtername,parameter,..)

dimana:

- fspecial adalah jenis filter yang digunakan
- average = filter rata-rata
- disk = circular averaging filter
- gaussian = filter gauss
- laplacian = aproximasi operator 2-D laplace
- log= laplacian of gaussian filter
- motion= motion filter
- prewitt : Prewitt horizontal edge-emphasizing filter
- sobel : Sobel horizontal edge-emphasizing filter
- unsharp : unsharp contrast enhancement filter

filter yang tersusun diatas kemudian diimplementasikan pada fungsi *imfilter* untuk image RGB (3-D) dan *filter2* untuk image grayscale atau 2-D. Adapun contoh penggunaanya seperti contoh berikut dimana filter yang digunakan adalah filter gaussian dengan matriks 12x12, dan terlihat bahwa gambar hasil menjadi blur.

```
gambar=imread('turtle.jpg');
gaussianFilter = fspecial('gaussian', [12, 12], 5)
hasil = imfilter(gambar, gaussianFilter, 'symmetric', 'conv');
subplot(1,2,1), image(gambar);
subplot(1,2,2), image(hasil), title('Blurred Turtle, blur matrix size 12');
```



Deteksi Tepi

Seleksi objek biasanya selanjutnya dilakukan langkah deteksi tepi dalam proses pengolahan citra, di MATLAB proses pendeteksian tepi dilakukan dengan perintah/fungsi *edge*. Ada beberapa metode yang dapat dilakukan pada deteksi tepi menggunakana MATLAB yaitu metode sobel, prewitt, roberts, laplacian of gaussian, metode zero cross, dan Canny.

Yang penting diperhatikan pada deteksi tepi bahwa hanya dapat dilakukan menggunakan citra grayscale atau citra 2-D.

Contoh penggunaan metode deteksi tepi:

```
I = imread('turtle.jpg');
gray=rgb2gray(I);
BW1 = edge(gray,'prewitt');
BW2 = edge(gray,'canny');
BW3 = edge(gray,'sobel');
BW4 = edge(gray,'roberts');
imshow(BW1);
figure, imshow(BW2)
figure, imshow(BW3)
figure, imshow(BW4)
```

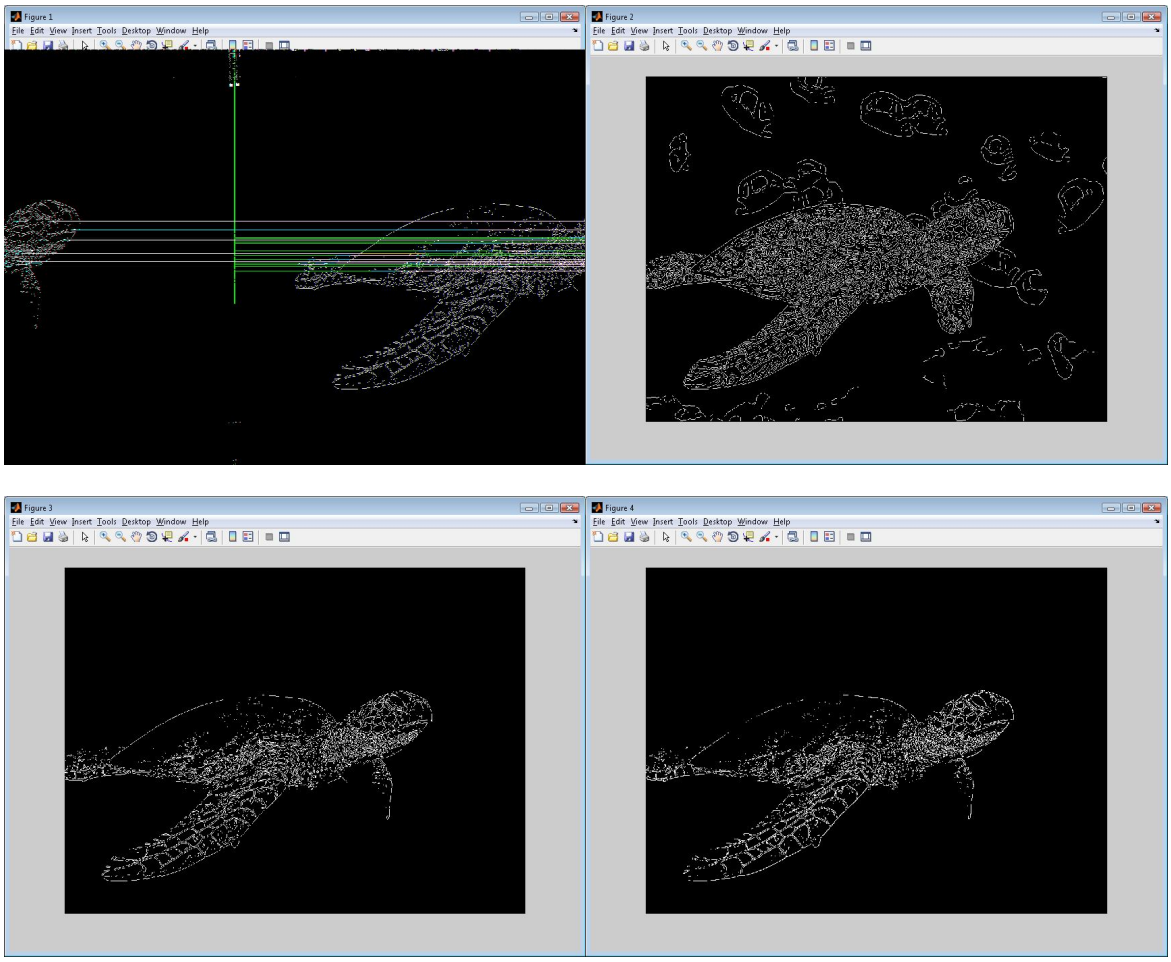
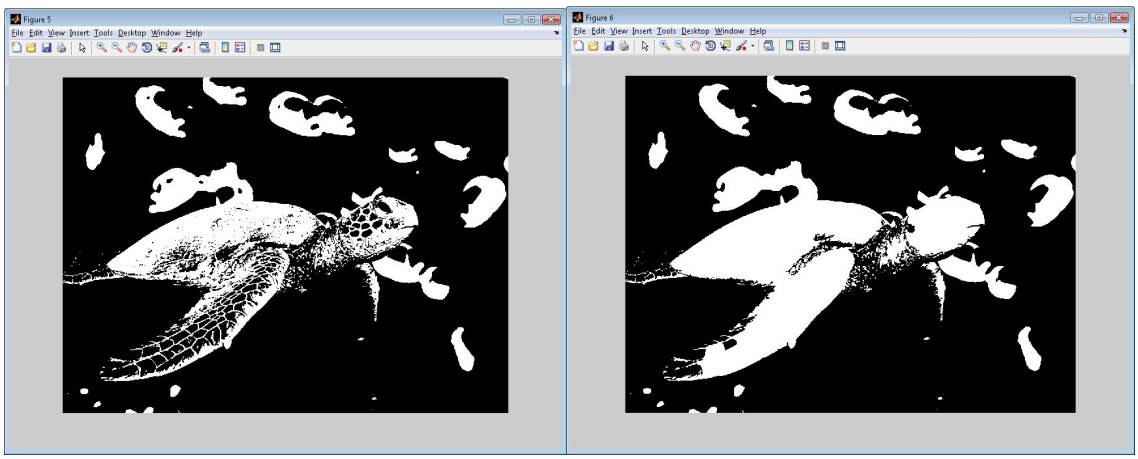


Image Reconstruction

Pada banyak kasus pengolahan citra baik proses binerisasi maupun deteksi tepi menghasilkan citra yang pada umumnya masih belum baik, oleh karena itu perlu dilakukan perbaikan citra / reconstruksi citra kembali. Di matlab proses rekonstruksi dilakukan menggunakan fungsi *imfill* .

Contoh penggunaan rekonstruksi image yaitu:

```
gambar = imread('turtle.jpg');
[X,map] = rgb2ind(gambar, 128);
I = ind2gray(X,map);
thresh=graythresh(gray);
imbw=im2bw(gray,thresh);
Ifill = imfill(imbw,'holes');
figure, imshow(imbw);figure, imshow(Ifill)
```



Terlihat gambar diatas gambar ke-2 terlihat lebih baik jika dibandingkan dengan gambar hasil binerisasi.

Dasar Pemrosesan Image di MATLAB berbasis Video Camera

Mendeteksi Jenis Video Camera

Sebelum memulai mengambil data dari video camera perlu dipastikan bahwa adapter kamera kita sudah terbaca dengan baik oleh matlab, untuk melakukan hal tersebut dapat digunakan perintah `imaqhwinfo`

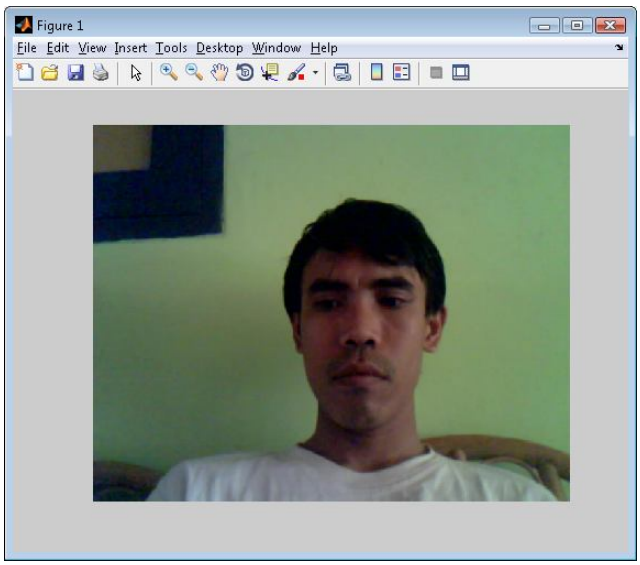
```
>> imaqhwinfo
ans =
    InstalledAdaptors: {'coreco'  'winvideo'}
        MATLABVersion: '7.7 (R2008b)'
          ToolboxName: 'Image Acquisition Toolbox'
       ToolboxVersion: '3.2 (R2008b)'
```

Terlihat dari hasil diatas bahwa kamera yang terinstal pada komputer bermerk coreo dan menggunakan driver winvideo.

Mengambil dan menampilkan Gambar Video Camera

Proses selanjutnya dalam pemrosesan video yaitu mengambil data dengan menggunakan perintah *videoinput* dan menentukan parameter seperti video resolusi, lebar, panjang dan band dari video kemudian membuat sebuah handle yang menampung image frame tersebut. Setelah *handle* image didapatkan maka biasanya video dapat ditampilkan menggunakan perintah *preview*. Contoh sederhana dari menampikan video pada matlab seperti dibawah ini.

```
vid = videoinput('winvideo');
vidRes = get(vid, 'VideoResolution');
imWidth = vidRes(1);
imHeight = vidRes(2);
nBands = get(vid, 'NumberOfBands');
hImage = image( zeros(imHeight, imWidth, nBands) );
preview(vid, hImage);
pause(30);
stoppreview(vid);
delete(vid)
clear vid
```



Live Histogram

Histogram sangat penting dalam pengolahan citra termasuk video oleh karena itu pembuatan histogram secara live juga sangat dibutuhkan dalam proses interpretasi objek yang akan dianalisis. Contoh pembuatan histogram pada matlab dilakukan seperti dibawah ini:

```
utilpath = fullfile(matlabroot, 'toolbox', 'imaq', 'imaqdemos', 'helper');
addpath(utilpath);
vidobj = videoinput('winvideo');
set(vidobj, 'ReturnedColorSpace', 'grayscale')
vidRes = get(vidobj, 'VideoResolution');
f = figure('Visible', 'off');
imageRes = fliplr(vidRes);
subplot(1,2,1);
hImage = imshow(zeros(imageRes));
```

```
axis image;
setappdata(hImage,'UpdatePreviewWindowFcn',@update_livehistogram_display);
preview(vidobj, hImage);
pause(30);
stoppreview(vidobj);   delete(f);
delete(vidobj)
clear vidobj
```