Matlab Workshop 2008

by Sung-Hyuk Cha
On June 18th 2008
Graduate Center rm TBA, White Plains, NY

This tutorial introduces basic Matlab features.

Time	Topics
1:00AM ~1:50PM	Image Processing and Steganography
2:00PM ~3:30PM	Numerical Taxonomy, Clustering, etc...
3:30PM ~4:30PM	Discussion and curriculum development...

Documentation from Mathworks

- MATLAB Roadmap
- Image Processing Toolbox

Contact: scha@pace.edu

http://math.ucsd.edu/~driver/21d-s99/matlab-primer.html

http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml

http://www.mathworks.com/access/helpdesk/help/toolbox/images/images.shtml

Matlabs are installed in all computers at PLV Goldstein building Lab 315.
Matlabs are installed in all computers at Downtown Pace Plaza east Lab 314.

Any other location? Map a drive to \\sapps\matlab

Workshop resources

http://csis.pace.edu/~scha/MatlabWorkshop08.zip

Table of Content

```
%%%%%  Matlab as a simple calculator
2+3

2^2

A = 3

B = A + 5

C = [2,3,5]

mean(C)

mean([2,3,4])


%%%%%%  Matrix = array (Linear Algebra)

D = [2, 3 ,4; 3, 4, 5]

D(2, 1)

sum(D)
mean(D)
std(D)
var(D)

D'
E = sum(D')

F = 2 * D

G = D + E

%%%%%   Charts  (Excel spreadsheet)

H = [3, 2, 4, 1, 8, 3, 2]

bar(H)

pie(H)

pie(H,{'Manag','I-bus','PolSci', 'Theat', 'Finance', 'Bio', 'Chem'})



%%%%  Number conversion functions and .m files
%%%%%%%%   Hex for HTML?

dec2hex(10)
dec2bin(2)
dec2base(25, 8)

K = dec2bin(221,8)
K(8)
size(K)


%  Let's define   dec2oct  (must be located in the current directory)



%%%%%%%%  Conditional statements
%%%%   <   <=   >   >=   ==   ~=
%%%%   $ -> and      | ->  or
```

```
H = [3, 2, 4, 1, 8, 3, 2]
n = sum(H)

if n ~= 0
   P = H/n
else
   disp('error, divided by zero.')
end



%%  another example
n = 254

if mod(n,2) == 0
   disp('even number')
   B = dec2bin(n,8)
   B(1,8)
else
   disp('odd number')
   B = dec2bin(n,8)
   B(1,8)
end



%%%%%%  Loops   converting dec 2 bin
%%%   While loop

n = 254
b = zeros(1,8)
pos = 8;

while n > 0
   b(1,pos) = mod(n,2)
   n = floor(n/2)
   pos = pos - 1;
end


%%% for loop

H = [3, 2, 4, 1, 8, 3, 2]
[m, n] = size(H)
EO = zeros(m,n)
for i = 1:n
   EO(1,i) = mod(H(1,i),2)
end
```

```
%%%%%%%%%%   Distance Similarity Measures

X = [1,3,1,0,1,0,3,1]
Y = [1,1,2,1,1,1,1,2]

Z = [5,1,0,0,0,1,1,2]

lorentzian(X,Y)
citiblock(X,Y)
euclidean(X,Y)
minkowski(X,Y,3)
chebyshev(X,Y)
sorenson(X,Y)
soergel(X,Y)
canberra(X,Y)
gower(X,Y)
divergence(X,Y)
Czekanowski(X,Y)
wavehedges(X,Y)
intersection_s(X,Y)
intersection_d(X,Y)
IP_s(X,Y)
Cosine_s(X, Y)
tanimoto(X,Y)

n = sum(X)

matusita(X/n,Y/n)
bhattacharyya(X/n,Y/n)
harmonic(X/n,Y/n)
symchi(X/n,Y/n)
Jdivergence(X/n,Y/n)
JSdivergence(X/n,Y/n)
taneja(X/n,Y/n)

symgeo(X/n,Y/n)
KL(X/n,Y/n)
JD(X/n,Y/n)

pearson(X/n,Y/n)
neyman(X/n,Y/n)
tridisc(X/n,Y/n)
hellinger(X/n,Y/n)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear

close all
n= 20;
b = 8;

tn = 30;        % number of histograms
ns = 31;        % number of similarities
nl = 20;        % number of dendrograms

dmat = zeros(ns, tn);
cdist = zeros(nl,465);
for j = 1:nl
    J = zeros(1,b);
    for i = 1:n
        x = ceil (b * rand);
        J(1,x)= J(1,x) + 1;
    end
    H = zeros(tn, b);
    for k = 1:tn
        for i = 1:n
            x = ceil (b * rand);
            H(k,x)= H(k,x) + 1;
```

```
        end
    end

    for i = 1:tn
        dmat(1,i) = citiblock(J,H(i,:));
        dmat(2,i) = euclidean(J,H(i,:));
        dmat(3,i) = minkowski(J,H(i,:),3);
        dmat(4,i) = chebyshev(J,H(i,:));
        dmat(5,i) = avgCC(J,H(i,:));
        dmat(6,i) = sorenson(J,H(i,:));
        dmat(7,i) = euclidean2(J,H(i,:));
        dmat(8,i) = canberra(J,H(i,:));
        dmat(9,i) = gower(J,H(i,:));
        dmat(10,i) = divergence(J,H(i,:));
        dmat(11,i) = wavehedges(J,H(i,:));
        dmat(12,i) = tanimoto(J,H(i,:));
        dmat(13,i) = intersection_s(J,H(i,:));
        dmat(14,i) = IP_s(J,H(i,:));
        dmat(15,i) = Cosine_s(J,H(i,:));
        dmat(16,i) = jaccard(J,H(i,:));
        dmat(17,i) = morisita(J,H(i,:));
        dmat(18,i) = pearson(J/n,H(i,:)/n);
        dmat(19,i) = neyman(J/n,H(i,:)/n);
        dmat(20,i) = hellinger(J/n,H(i,:)/n);
        dmat(21,i) = bhattacharyya(J/n,H(i,:)/n);
        dmat(22,i) = Jdivergence(J/n,H(i,:)/n);
        dmat(23,i) = KL(J/n,H(i,:)/n);
        dmat(24,i) = matusita(J/n,H(i,:)/n);
        dmat(25,i) = JD(J/n,H(i,:)/n);
        dmat(26,i) = harmonic(J/n,H(i,:)/n);
        dmat(27,i) = JSdivergence(J/n,H(i,:)/n);
        dmat(28,i) = taneja(J/n,H(i,:)/n);
        dmat(29,i) = symchi(J/n,H(i,:)/n);
        dmat(30,i) = symgeo(J/n,H(i,:)/n);
        dmat(31,i) = tridisc(J/n,H(i,:)/n);
    end
    l = 1;
    for i = 1:ns
       for k = i+1:ns
         R=corrcoef(dmat(i,:),dmat(k,:));
         cdist(j,l) = 1 - abs(R(1,2));
         l = l + 1;
       end
    end
end


%dmatm = dmatm/10;


clustTreeEuc = linkage(mean(cdist),'average');

h101=figure(101);

[h,nodes] = dendrogram(clustTreeEuc,0,'ORIENTATION','right');

l={'CityBlock L1','Euclidean L2','Minkowski L3','Cheb
Loo','Avg(L1,Loo)','Sorenson','Qeuc','Canberra','Gower','divergence','wave
Hedges','Tanimoto','Intersection','Inner
Product','cosine','Jaccard','Dice','Peason K2','Neyman
K2','Hellinger','Bhattacha-','Jeffreys','KL','Matusita','Jensen diff','harmo
mean','JS','Taneja','symmetric K2','KJ','Triang Disc'};

hx=get(get(h101,'CurrentAxes'),'YTickLabel');

gx = zeros(ns,1);

for i=1:size(hx,1)
```

```
    gx(i)=str2double(hx(i,2));
    if hx(i,1) == ' '
        temp = hx(i,1);
    else
        gx(i) = gx(i) + 10* str2double(hx(i,1));
    end

end
set(get(h101,'CurrentAxes'),'YTickLabel',[l(gx(1)),l(gx(2)),l(gx(3)),l(gx(4)),l(g
x(5)),l(gx(6)),l(gx(7)),l(gx(8)),l(gx(9)),l(gx(10)),l(gx(11)),l(gx(12)),l(gx(13))
,l(gx(14)),l(gx(15)),l(gx(16)),l(gx(17)),l(gx(18)),l(gx(19)),l(gx(20)),l(gx(21)),
l(gx(22)),l(gx(23)),l(gx(24)),l(gx(25)),l(gx(26)),l(gx(27)),l(gx(28)),l(gx(29)),l
(gx(30)),l(gx(31))])
figure(h101);



%%%%%%%% Drawing hierachical clustering

ns = 9;
dmat = zeros(ns, tn);


for i = 1:tn
   dmat(1,i) = citiblock(J,H(i,:));
   dmat(2,i) = euclidean(J,H(i,:));
   dmat(3,i) = minkowski(J,H(i,:),3);
   dmat(4,i) = chebyshev(J,H(i,:));
   dmat(5,i) = IP_s(J,H(i,:));
   dmat(6,i) = Cosine_s(J,H(i,:));
   dmat(7,i) = tanimoto(J,H(i,:));
   dmat(8,i) = matusita(J/n,H(i,:)/n);
   dmat(9,i) = bhattacharyya(J/n,H(i,:)/n);
end

dmatn = dmat;

for i = 1:ns
     tmin1 = min(dmat(i,:));
     tmax1 = max(dmat(i,:));
     for l = 1:tn
        dmatn(i,l) = (dmat(i,l) - tmin1)/(tmax1 - tmin1);
     end
end




k = 1;
for i = 1:ns
  for j = i+1:ns
     k = ns * (i-1) + j;
     SUBPLOT(ns,ns,k), plot(dmatn(i,:), dmatn(j,:), 'r*')
%     AXIS([min(dmat(i,:)) max(dmat(i,:)) min(dmat(:)) max(dmat(:))])
     grid
  end
end



%%%  similarity to distance
dmatn(5,:) = ones(1,20) - dmatn(5,:);
dmatn(6,:) = ones(1,20) - dmatn(6,:);
dmatn(7,:) = ones(1,20) - dmatn(7,:);


eucD = pdist(dmatn,'euclidean');
```

```
clustTreeEuc = linkage(eucD,'average');
[h,nodes] = dendrogram(clustTreeEuc,0,'ORIENTATION','right');

figure

k = 1;
for i = 1:ns
  for j = i+1:ns
     meas(1,k) = sum(abs(dmatn(i,:) - dmatn(j,:)));
     k = k + 1;
  end
end


clustTreeEuc = linkage(meas,'average');
[h,nodes] = dendrogram(clustTreeEuc,0,'ORIENTATION','right');
```

Image Processing in Matlab

Prof. Sung-Hyuk Cha

Spring of 2008

School of Computer Science & Information Systems

PACE UNIVERSITY

Query by Image Content

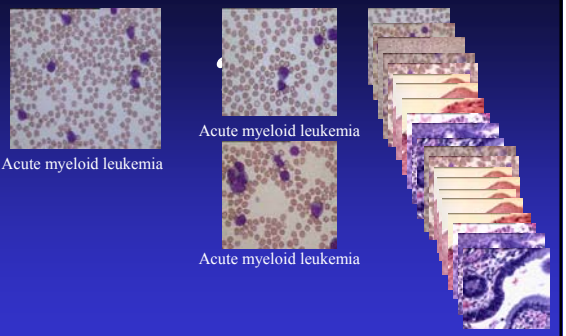
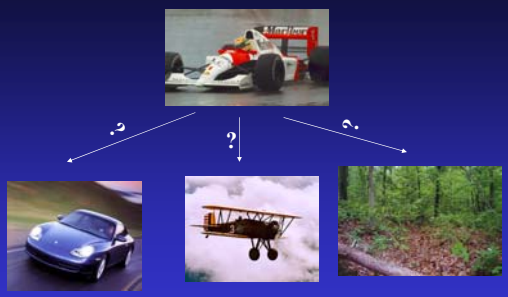


Image Classification



Dissimilarity (distance) / Similarity

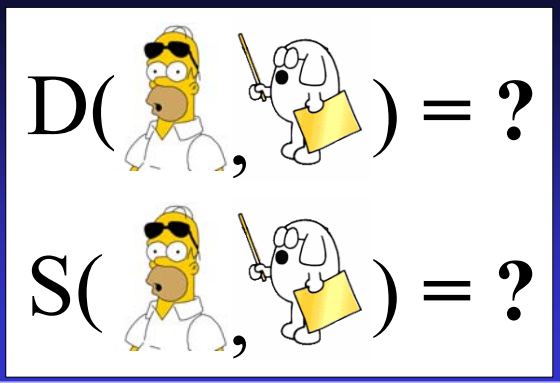


Image Indexing & Retrieval

Overview

- Human vision & illusion.
- Basic Image Processing & computer vision
- Machine Vision Applications.

Digital Image Processing vs. Computer Vision

CSIS

- There are no clear distinction
- Image processing
  - Applications where humans are in the loop.
  - Humans supply the intelligence
  - Image Analysis - extracting quantitative info.
    - Size of a tumor
    - distance between objects
    - facial expression
  - Image restoration. Try to undo damage
    - needs a model of how the damage was made
  - Image enhancement. Try to improve the quality of an image
  - Image compression. How to convey the most amount of information with the least amount of data

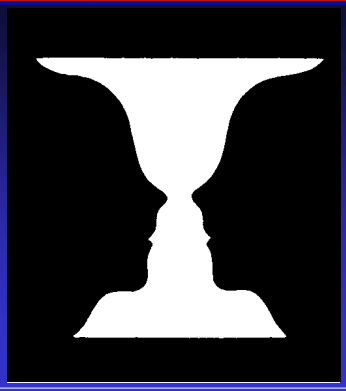
Digital Image Processing vs. Computer Vision

CSIS

- Computer Vision
  - Take the human out of the loop
  - The computer supplies the intelligence
  - Where does the computer get it's intelligence?

The figure-Ground Problem

CSIS



The Bunny/Duck illusion

CSIS

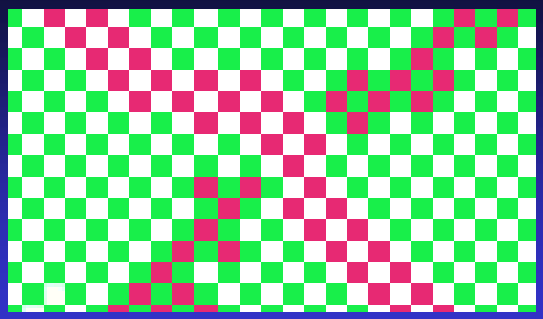
More illusions

CSIS

Squares or lines?

More illusions: How many colors?

CSIS

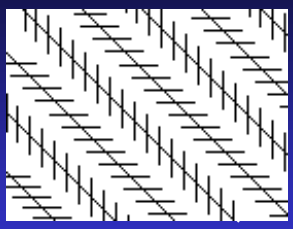


More illusions: parallel line

CSIS

More illusions

CSIS



More illusions

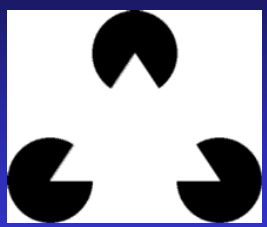
CSIS

More illusions

CSIS

More illusions

CSIS



Photometry

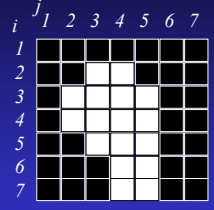
CSIS

Concerned with mechanisms for converting light energy into electrical energy.

World → Optics → Sensor → Signal → Digitizer → Digital Representation

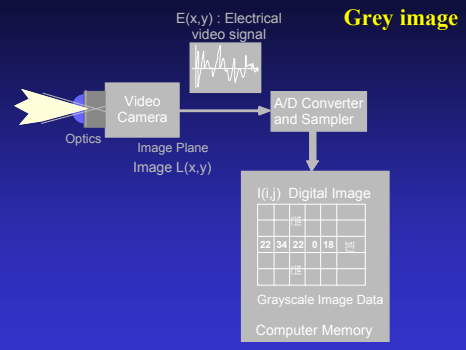
Binary image

CSIS



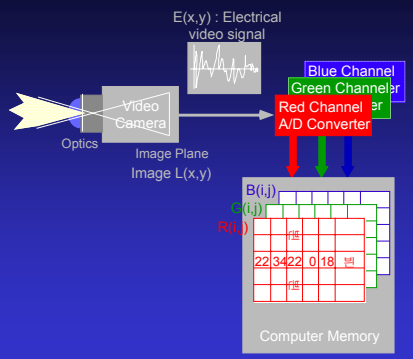
Grey image

CSIS



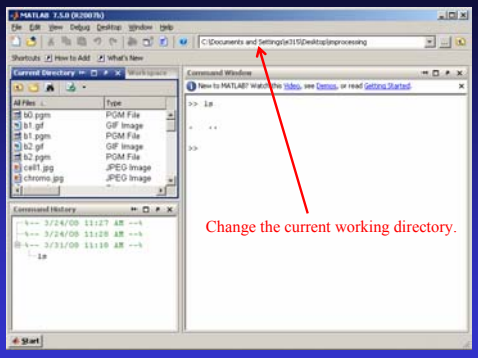
Color image

CSIS



Reading images in Matlab

CSIS



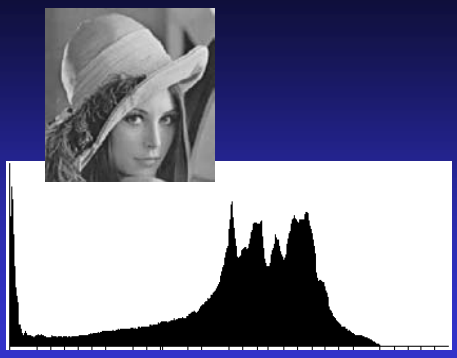
Reading and Displaying images in Matlab

CSIS

```
% To read and display...
I = imread('scha98.jpg', 'jpg');
imshow(I)
I(20,10,1)
I(20,10,:)
```

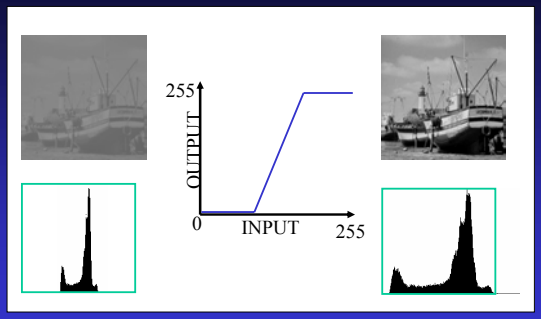

grey image and its histogram

CSIS



Linear Stretching

CSIS



Histogram

CSIS

```
% Reading, histogram, and writing
I3 = imread('tryColor-L.bmp', 'bmp');
imshow(I3)
I3(20,10,1)
I3(20,10,:)
figure, imhist(I3)
I2 = histeq(I3);
figure, imshow(I2)
figure, imhist(I2)
imwrite(I2, 'hey.jpg');
```

I4 = imadjust(I3, stretchlim(I), [0 1]);

Histogram

CSIS

```
% Color image histogram
A = imhist(I)
R = imhist(I(:,:,1))
G = imhist(I(:,:,2))
B = imhist(I(:,:,3))
imhist(I)
figure, bar(R)
figure, bar(G)
figure, bar(B)
```


Resizing

CSIS

```
I2 = imresize(I, 2.5);
figure, Imshow(I2)
```

```
I3 = imresize(I,[100 150]);
figure, imshow(I3)
```

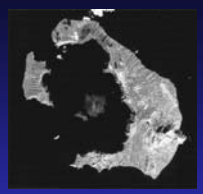

Rotation

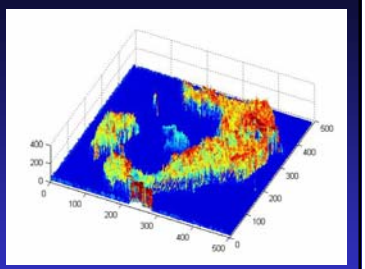
CSIS

```
I = imread('scha2.jpg', 'jpg');
J = imrotate(I, 180, 'bilinear');
imshow(J);
```


Mesh and Surf

CSIS






```
imshow(I);
figure, mesh(double(I));
figure, surf(double(I));
figure, mesh(double(I)),zlim([0,200]);
```

Panorama and Mosaics

CSIS

Panorama

CSIS

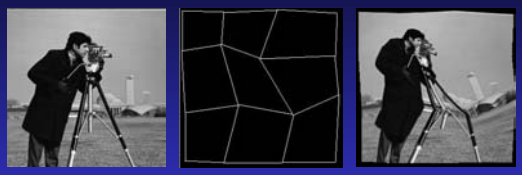
```
% Panorama
I2 = imread('scha1.jpg', 'jpg');
J1 = imrotate(I, 90, 'bilinear');
J2 = imrotate(I2, 90, 'bilinear');
T = [J1;J2];
J1 = imrotate(T, 270, 'bilinear');

% Warping
[x,y,z] = cylinder;
J2 = imrotate(J1, 180, 'bilinear');
warp(x,y,z,J2)
```

C = [I;I2];
C = [I2,I];

Warping

CSIS



http://www.doctorwarp.com/index.php?ID=23&flx=world

Warping

CSIS

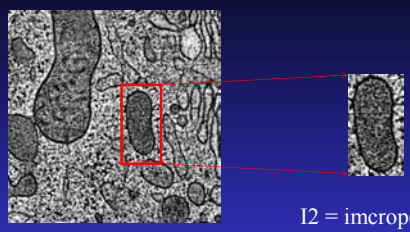
```
% Warping
[x,y,z] = cylinder;
warp(x,y,z,I)

[x,y,z] = sphere;
warp(x,y,z,I)

[x,y,z] = cylinder;
J2 = imrotate(J1, 180, 'bilinear');
warp(x,y,z,J2)
```


Crop

CSIS

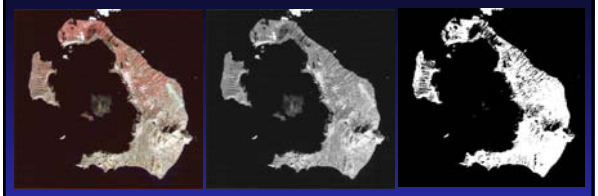


I2 = imcrop(I);

```
I = imread('cell1.jpg');
I2 = imcrop(I,[60 40 100 90]);
figure, imshow(I)
figure, imshow(I2)
```


Conversion

CSIS

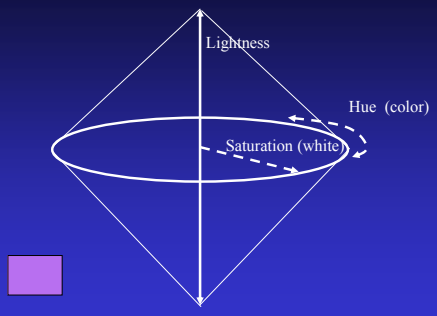


```
G = rgb2gray(I);
BW = im2bw(I);

HSV = rgb2hsv(I);
H = HSV(:,:,1)
```

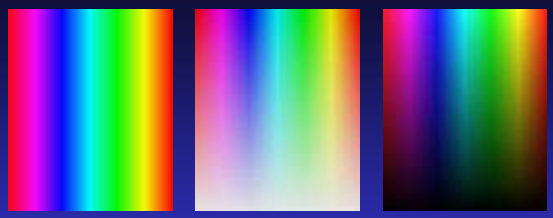
HSL Color Space

CSIS



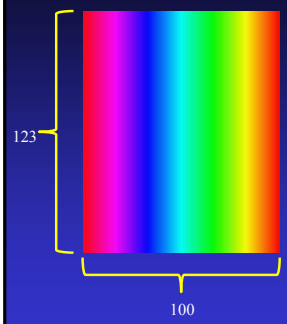
HSL Color Space

CSIS



HSL Color Space

CSIS

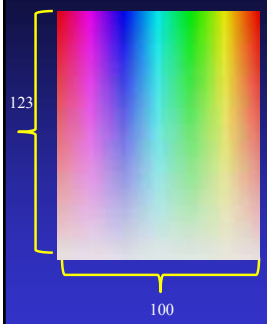



```
for i = 1:123
   for j = 1:100
      HSV(i,j,1) = 1 - j * 0.01;
      HSV(i,j,2) = 0.97;
      HSV(i,j,3) = 0.97;
   end
end

I2 = hsv2rgb(HSV);
imshow(I2)
```


HSL Color Space

CSIS

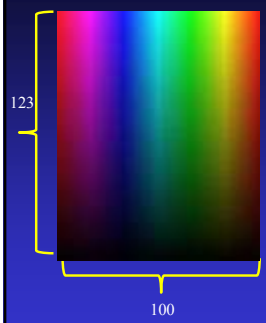



```
for i = 1:123
  for j = 1:100
    HSV(i,j,1) = 1 - j * 0.01;
    HSV(i,j,2) = 1.0 - i * 0.008;
    HSV(i,j,3) = 0.9;
  end
end

I2 = hsv2rgb(HSV);
figure, imshow(I2)
```


HSL Color Space

CSIS

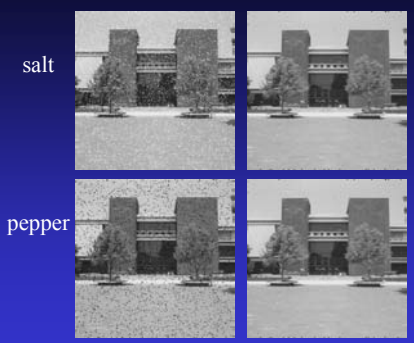



```
for i = 1:123
  for j = 1:100
    HSV(i,j,1) = 1 - j * 0.01;
    HSV(i,j,2) = 0.9;
    HSV(i,j,3) = 1.0 - i * 0.008;
  end
end

I2 = hsv2rgb(HSV);
figure, imshow(I2)
```


Noise Removal

CSIS



Segmentation

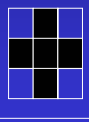
CSIS

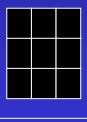
Connected Components/ Image Labeling

CSIS

- Goal: To find clusters of pixels that are similar and connected to each other
- How it works:
  - Assign a value to each pixel
  - Define what similar values mean
    - e.g., 10 +/- 2
  - Determine if like pixels are connected

4- connected 8-connected

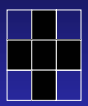




Connected Components/ Image Labeling

CSIS

1	1	1	1	1	1
1	0	0	1	1	1
1	1	1	0	1	1
1	2	2	0	0	1
1	2	2	0	0	1

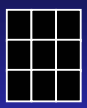


A	A	A	A	A	A
A	B	B	A	A	A
A	A	A	C	A	A
A	D	D	C	C	A
A	D	D	C	C	A

Connected Components/ Image Labeling

CSIS

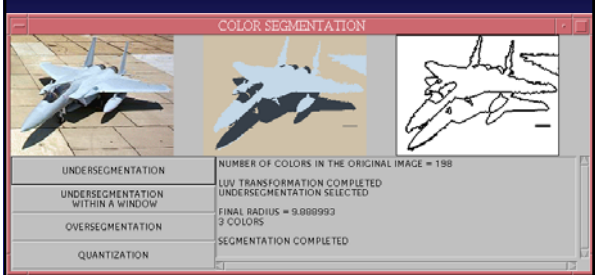
1	1	1	1	1	1
1	0	0	1	1	1
1	1	1	0	1	1
1	2	2	0	0	1
1	2	2	0	0	1



A	A	A	A	A	A
A	B	B	A	A	A
A	A	A	B	A	A
A	C	C	B	B	A
A	C	C	B	B	A

Segmentation

CSIS

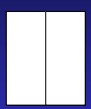


Edge Detection

CSIS

convolution

mask



$$\mathbf{B} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Machine Vision

CSIS

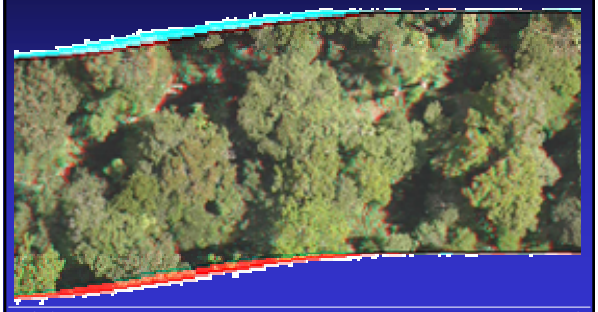
Biometric Verification

CSIS

From the movie, “Minority Report”: a robot identifies a suspect.

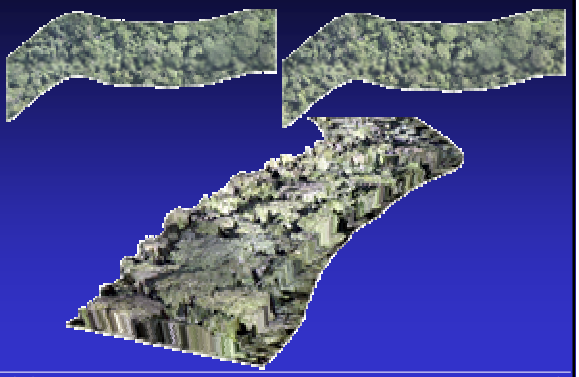
Stereo

CSIS



Stereo vision

CSIS

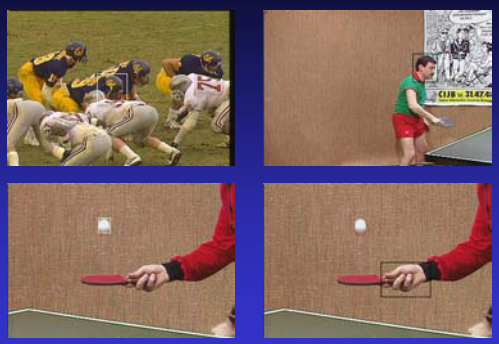


Subtraction

CSIS

Object Tracking

CSIS



Object Tracking for Surveillance

CSIS

Input

Output

Fingerprint Verification System

CSIS

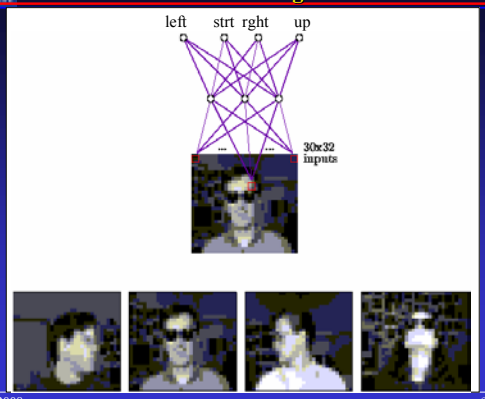
Individuality of Faces

CSIS

Each person has different faces.

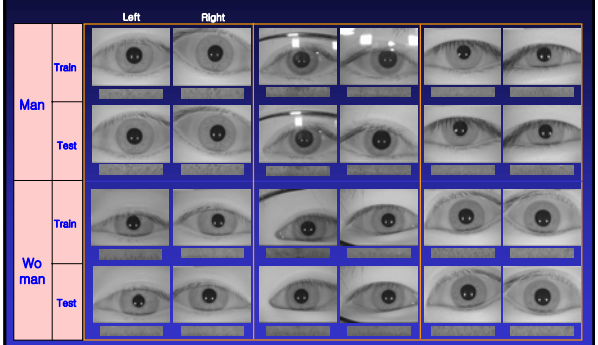
Head Pose Recognition

CSIS



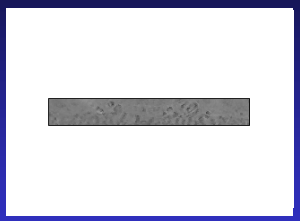
Iris authentication

CSIS



Iris authentication

CSIS



Complex Pattern Recognition Applications

CSIS

Sargur N. Srihari

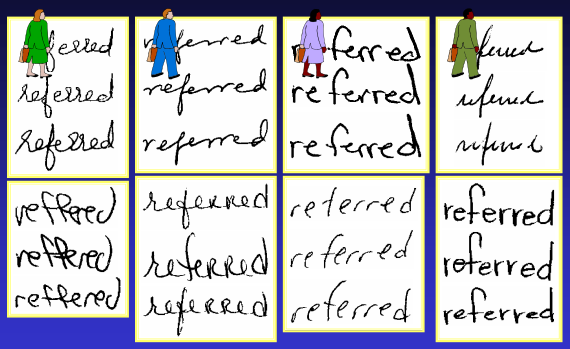
520	Lee Entrance	STE	202
f_5 primary number	f_6 street name	f_7 secondary designator abbr.	f_8 secondary number

Amherst	NY	14228	-	2583
f_1 city name	f_2 state abbr.	f_3 5-digit ZIP Code		f_4 4-digit ZIP+4 add-on

• Delivery point 142282583

Handwriting

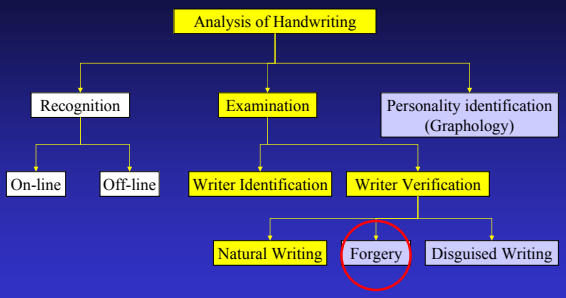
CSIS



Each person writes differently.

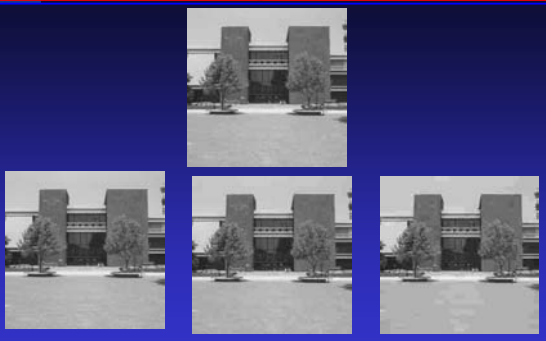
Handwriting Analysis Taxonomy

CSIS



Compression

CSIS



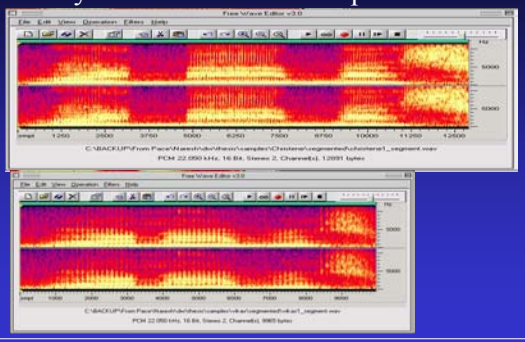
Speech Recognition System

CSIS

Speaker Individuality

CSIS

"My name is" from Two Speakers



```
% Understanding Images...

% To read, display, and write...
I = imread('scha98.jpg', 'jpg');
imshow(I)
I(20,10,1)
I(20,10,:)


%--------------- see from Robot file (Histograms)------ %

I3 = imread('tryColor-L.bmp', 'bmp');
imshow(I3)
I3(20,10,1)
I3(20,10,:)
figure, imhist(I3)
I2 = histeq(I3);
figure, imshow(I2)
figure, imhist(I2)
imwrite(I2, 'hey.jpg');

I4 = imadjust(I3, stretchlim(I), [0 1]);
figure, imshow(I4);



% Color image histogram
A = imhist(I)
R = imhist(I(:,:,1))
G = imhist(I(:,:,2))
B = imhist(I(:,:,3))
imhist(I)
figure, bar(R)
figure, bar(G)
figure, bar(B)


%------------------------

% Resizing
I2 = imresize(I, 2.5);
figure, Imshow(I2)
I3 = imresize(I,[100 150]);
figure, Imshow(I3)
% good for size invarient recognizer...



clear, close all


% Rotation
I = imread('scha2.jpg', 'jpg');
J = imrotate(I, 180, 'bilinear');
imshow(J);


%  mesh and surf
imshow(I);
figure, mesh(double(I));
figure, surf(double(I));
figure, surf(double(I)),zlim([0,200]);
figure, surf(double(I(1:8:end,1:8:end,:,3))),zlim([0 255]);
figure, mesh(double(I(1:8:end,1:8:end,:,3))),zlim([0 255]);


% Panorama
I2 = imread('scha1.jpg', 'jpg');
```

```
J1 = imrotate(I, 90, 'bilinear');

J2 = imrotate(I2, 90, 'bilinear');
T = [J1;J2];
J1 = imrotate(T, 270, 'bilinear');

C = [I;I2];

C = [I2,I];

% Warping
[x,y,z] = cylinder;
J2 = imrotate(J1, 180, 'bilinear');
warp(x,y,z,J2)


% Cropping


I2 = imcrop(I,[60 40 100 90]);
figure, imshow(I)
figure, imshow(I2)

I2 = imcrop(I);

island.jpg


% Conversion

I = imread('scha98.jpg', 'jpg');

G = rgb2gray(I);

BW = im2bw(I);

HSV = rgb2hsv(I);

HSV(:,:,1)


% Understanding HSV

for i = 1:123
  for j = 1:100
    HSV(i,j,1) = 1 - j * 0.01;
    HSV(i,j,2) = 0.97;
    HSV(i,j,3) = 0.97;
  end
end


I2 = hsv2rgb(HSV);
imshow(I2)

for i = 1:123
  for j = 1:100
    HSV(i,j,1) = 1 - j * 0.01;
    HSV(i,j,2) = 1.0 - i * 0.008;
    HSV(i,j,3) = 0.9;
  end
end


I2 = hsv2rgb(HSV);
figure, imshow(I2)
```

```
for i = 1:123
  for j = 1:100
    HSV(i,j,1) = 1 - j * 0.01;
    HSV(i,j,2) = 0.9;
    HSV(i,j,3) = 1.0 - i * 0.008;
  end
end


I2 = hsv2rgb(HSV);
figure, imshow(I2)


%%%%%%%%%%
%%%%%%%%%%%%%%%%%  Image Processing summary %%%%%
imread
imwrite
imshow
imhist
imresize
imrotate
imadjust
imcrop

rgb2gray
im2bw
rgb2hsv
hsv2rgb


histeq
mesh
surf

figure
bar

double
cylinder
warp

clear
close all
==================
Try help(function name) on the Matlab prompt. e.g.,
help(mesh)
```

```
%%%%%   Steganography
%%%%%%%%%%%%%%%%  Hiding


I = imread('scha.bmp', 'bmp');
imshow(I)

BW = imread('info.bmp', 'bmp');
imshow(BW)

w=size(I,2);
h=size(I,1);

C = double(I);
D = double(BW);

for i = 1:h
  for j = 1:w
    if mod(C(i,j,1),2) == 0
         if D(i,j) == 1
            if C(i,j,1) > 0
               C(i,j,1) = C(i,j,1) - 1;
            else
               C(i,j,1) = C(i,j,1) + 1;
            end
         end
    else
         if D(i,j) == 0
            if C(i,j,1) < 255
               C(i,j,1) = C(i,j,1) + 1;
            else
               C(i,j,1) = C(i,j,1) - 1;
            end
         end
    end

  end
end

E = I;

for i = 1:h
  for j = 1:w
     E(i,j,1) = C(i,j,1);
  end
end

figure, imshow(E)

imwrite(E, 'haha.bmp')



%%%%%%%%%%%%%%%%  Decoding
clear, close all

I = imread('haha.bmp', 'bmp');

w=size(I,2);
h=size(I,1);
BW = zeros(h,w);

C = double(I);

for i = 1:h
  for j = 1:w
    if mod(C(i,j,1),2) == 0
         BW(i,j) = 0;
```

```
    else
         BW(i,j) = 1;
    end
  end
end


figure, imshow(mat2gray(BW))



%%%%%%%%%%%%%%%%%  with a simple function   (Sender side)
clear, close all
I = imread('scha.bmp', 'bmp');
imshow(I)

BW = imread('info2.bmp', 'bmp');
imshow(BW)

E = hide(I, BW);

imwrite(E, 'haha.bmp')

%%%%%%%%%%%%%%%  Receiver only receives haha.bmp
clear, close all

I = imread('haha.bmp', 'bmp');
Decodeinfo(I);



%%%%%%%%%%%%%%%%  Hiding two messages   (Sender side)
clear, close all
I = imread('scha.bmp', 'bmp');
imshow(I)

B1 = imread('info.bmp', 'bmp');
figure, imshow(B1)
B2 = imread('info2.bmp', 'bmp');
figure, imshow(B2)

E = hide2(I, B1, B2);

imwrite(E, 'Booya.bmp')

%%%%%%%%%%%%%%%  Receiver only receives Booya.bmp
clear, close all

I = imread('Booya.bmp', 'bmp');
Decodeinfo2(I, 1);



clear, close all

I = imread('haha.bmp', 'bmp');
Decodeinfo2(I, 2);



%%%%%%%%%%%%%%%%  nicer hider
clear, close all
I = imread('scha.bmp', 'bmp');
imshow(I)
```

```
BW = imread('info2.bmp', 'bmp');
imshow(BW)

E = hide3(I, BW, 1);

Decodeinfo2(E, 1);
%%%% compare

Decodeinfo2(I, 1);


imwrite(E, 'Ohboy.bmp')
```

M-Files: Creating Your Own Scripts and Functions

Users are able to tailor MATLAB by creating their own functions and scripts of MATLAB commands and functions. Both scripts and functions are ordinary ASCII text files external to MATLAB. The name of each file must end in ``.m'' and must be found on MATLAB's search path. (It is easiest to start MATLAB from the directory containing your M-files. See also the command `path`.) By default while an M-file is executing its statements are not displayed. This default behavior can be changed using the `echo` command.

A script may contain any sequence of MATLAB statements, including references to other M-files. It is invoked like any other command without arguments and acts on the variables of the workspace globally. Each command in the file is executed as though you had typed it into MATLAB. Most of the demos provided by MATLAB are simply scripts.

The first line of a function M-file starts with the word *function* and declares the name of the function and its input and output parameters. All input parameters and variables defined within the file are local to the function. Figure 1 provides a simple example from the *MATLAB User's Guide*.

```
function y = mymean(x)
% mymean:  Average or mean value.
% For vectors, mymean(x) returns the mean value.
% For matrices, mymean(x) is a row vector containing the mean value of
each column.
[m,n] = size(x);
if m == 1
        m = n;
end
y = sum(x)/m;
```

Figure 1: A User Defined Function.

If we type this function into a file called mymean.m, then we can call `mymean` like any other MATLAB function.

```
>> mymean(1:99)

ans =
     50
```

The parameter supplied to mymean is passed by value (not reference) to the local variable `x`. The variables `m,n,y` are also local to `mymean` and do not exist after it returns. An M-file function does

not alter the value of variables in the workspace unless explicit *global* declarations are made. To explicitly declare a global variable, include the statement `global var-name` in any function (and possibly the MATLAB base workspace) in which the variable is to be accessed globally.

The MATLAB function `eval` accepts a single string argument, which it interprets as a MATLAB expression or statement. The following example demonstrates how you can use `eval` to pass one function to second function as an argument and have the second function call the first function within its body.

Suppose the following function is in the M-file `goo.m`

```
function val = goo(x)
val = x.^2;
```

and the following function is in the M-file `foo.m`.

```
function val = foo(fname, a)
% calls another function, whose name is passed as a string to
% argument fname, with the parameter a
val = eval([fname,'(a)']);
%          ^^^^^^^^^^^  Constructs a string interpreted as an expression
%                       or statement by eval.
```

Using these functions, the following command

```
>> foo('goo', 3)
```

evaluates to

```
ans =
     9
```

For Loops, While Loops, and Conditional Statements

MATLAB also provides a programming language that includes looping statements (`for` and `while`), conditional statements (`if`), and relational and logical operators for constructing conditions.

The execution of an `if` or `while` depends upon the evaluation of a *condition*. To construct conditions, MATLAB provides six relational operators

<	less than	≤	less than or equal	==	equal
>	greater than	≥	greater than or equal	~=	not equal

and three logical operators.

&	and	|	or	~	not

Note that the relational operator == compares two arguments, while = is used to assign a value to a variable.

When a condition compares scalar quantities it evaluates to 1 if true and 0 if false. Relational operators can also be applied to matrices of the same dimension. In this case a condition evaluates to a matrix of 0s and 1s representing the result of applying the operator to the individual elements of the matrix operands. A condition evaluating to a matrix is interpreted by `if` or `while` to be true if each element of the matrix is nonzero. (The MATLAB function `any` is useful when you want a different interpretation of the matrix resulting from the evaluation of a condition.)

The simplest form of an `if` statement is `if` *condition*
statements
`end`

For example,

```
>> A = [6 3 9];
>> n = 3;
>> if n ~= 0
     A = A/n;
   end
>> A
```

results in the output

```
  A =
     2     1     3
```

(Each semicolon in the previous example suppresses the output of a statement's result.)

As in other programming languages, `if` statements can also be used to choose between 2 or more alternatives. For example, the statement

```
>> if n < 0
     x = [x,abs(n)];
   elseif (rem(n,2) == 0) & (n ~= 0)
     x = [x,n/2];
   else
     x = [x,n+1];
   end
```

adds one of three possible elements to the end of the existing row vector `x`, depending on the value of scalar `n`. Note that `elseif` is one word. Writing `else` and `if` separately would create a second `if` statement requiring its own `end`.

When working with matrices and vectors, the ability to repeat statements with `for` and `while` statements is essential. The general form of the `for` statement given below permits the statements in its body to be repeated a specific number of times. `for` *variable* = *expression*
statements
`end`

The columns of *expression* are assigned to *variable* one by one and then the statements are executed. As an example, the following statements construct a vector containing the squares of the integers 2 through 5.

```
>> x = [];
>> low = 2;
>> hi = 5;
>> for i = low:hi
     x = [x, i*i];
   end
```
Of course, you could also write this `for` statement on a single line by separating its components by commas,
```
>> for i = low:hi, x = [x,i*i], end
```
but using separate lines with appropriate indentation usually improves readability. The following statement produces a vector with the same elements as `x`, but they are arranged in the reverse order.
```
>> y = [];
>> for i = hi:-1:low
     y = [y, i*i];
   end
```

You can also nest `for` statements. The following statements create an $m \times n$ Hilbert matrix, `H`.

```
>> for i = 1:m
     for j = 1:n
       H(i,j) = 1/(i+j-1)
     end
   end
```

Finally, MATLAB also has its own version of the while loop, which repeatedly executes a group of statements while a condition remains true. `while` *condition*
  statements
`end`

Given a positive number `val` the following statements compute and display the even powers of 2 less than or equal to `val`.

```
>> n = 0;
>> while 2^n <= val
     if rem(n,2) == 0
       2^n
       n = n + 1;
     else
       n = n + 1;
     end
   end
```
(The previous example increments `n` by 1 each iteration (instead of by 2) and exhibits other elements of poor programming style so that we can illustrate the nesting of an `if` statement inside a `while`.)