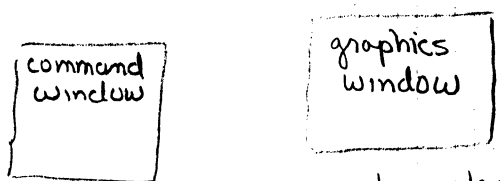
MATLAB primer (Ref: D.M. Etter, Engineering Problem Solving w/MATLAB)

very simple syntax Fortran, Basic
basic

matlab prompt >>
basic commands demo - list of demos to run
quit
exit
save - save variables in workspace.

limits of student edition
vector or matrix limited to 1024 elements.
graphics post processing not available (i.e. Post Script)



clc - clear command window

clg - clear graphics window

clear - clear all variables

^c - abort

MATLAB is case sensitive casesen off
casesen

who - list defined variables
whos - gives more information

% precedes comments

help - list of help topics

m-file <name>.m MATLAB program file
also called script file

to run an m-file enter the name of the m-file without the extension in the command window

echo cause m-files to be viewed as they execute

what lists all m-files on your computer
type <name> lists the contents of <name>.m

2.3 Matrices, Vectors and Scalars.

Explicitly defining matrices:

A = [3.5] ; suppresses printing of matrix

B = [1.5, 3.1] ;

C = [-1, 0, 0 ; 1, 1, 0 ; 1, -1, 0 ; 0, 0, 2]

end of row

can also do

```
C = [-1, 0, 0
      1, 1, 0
      1, -1, 0
      0, 0, 2] ;
```

continue for large matrices

F = [1, 52, 64, 197, 42, -42, 55, 82, 22, 109]

or F = [1, 52, 64, 197, 42, -42, ... indicates continue on next lin
55, 82, 22, 109] ;

using other matrices

B = [1.5 3.1]

S = [3.0 B]

gives S = [3.0 1.5 3.1].

S(2) references the 1.5

All MATLAB subscripts begin with 1.

Saving/loading matrices

easiest for images

save data1 x y ; saves matrices x and y in binary format

load data1 ; restores matrices

can also read/write ASCII files

save data1, dat z /ascii ; matrix row by row colon operator

- when used in a matrix it represents all the rows or all the columns.

$x = data1(:,1);$

all rows column 1

data1 = (0 , 0
.01 , .1255
.02 , .2507);

$y = data1(:,2);$

new matrices all rows column 2.

x & y will be column vectors.

- can also be used to generate numbers.

H = 1:8 generates [1, 2, 3, 4, 5, 6, 7, 8]

TIME = 0.0 : 0.5 : 5.0 generates numbers from 0.0 to 5.0 in increments of 0.5

- can be used to select submatrices

$$C = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

c_1 = C(:,2:3); where $C\_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 2 \end{bmatrix}$

all the rows and column

columns 2 to 3.

c_2 = C(3:4,1:2); $C\_2 = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}$

rows 3 & 4 columns 1 & 2.

Simple program

```
% Powers of a complex number
clear, clg                    % clears all variables & graphics
j= sqrt(-1)                   % define j
z1= 1.1* exp (j*2*pi/16);     % assign complex points z1 & z2
z2= 0.9* exp (j*2*pi/16);
z1powers = z1.^[1:32];        % raises point z1 to powers 1 thru 32
                              % i.e. z1powers = [z1**1 z1**2 ... z1**
x = [1:32]  % creates vector x = [1, 2, ... 32].
z2powers = z2.^x     % creates z2**1 z2**2, etc.
axis ('normal')   % (opt) 1:1 plot aspect ratio
plot (z1powers,'or') % plots each point of z1powers w/red circles
hold on                 % put more stuff on same plot
plot (z2powers,'.g')  % plots each point of z2powers w/green d
grid                    % put a grid on graph.
hold off
```

element by element

The dot operator is for element by element operations

for example

```
>> A.*B     %computes ABij = aij bij
>> A_2 = A.^2    %squares each element of A
>> A^2           % will compute A*A.
>> 2.^A  raises 2 to a matrix power.
>> 2_.^A  raises 2 to the power of each element in A.
```

if $A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$

$$2.\hat{}A = 1.0 \times 10^4 \begin{bmatrix} .7962 & 1.8029 & 2.8097 \\ .9782 & 2.2154 & 3.4523 \\ 1.1603 & 2.6276 & 4.0950 \end{bmatrix}.$$

$$2\_.\hat{}A = \begin{bmatrix} 2 & 16 & 128 \\ 4 & 32 & 256 \\ 8 & 64 & 512 \end{bmatrix}.$$

Images in MATLAB.

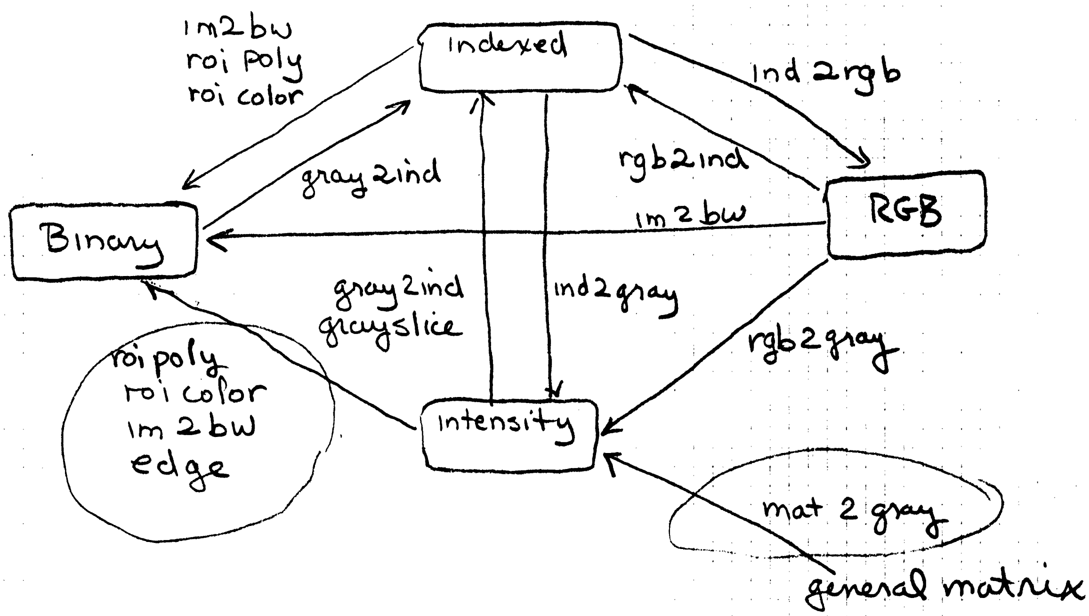
indexed images — uses color map. COLOR = [R G B]
n x 3 matrix for an image containing n colors.

intensity images — what we will use
double precision 0 (black) to 1 (white)

binary images — 0 (black), 1 (white).

RGB images — scanners, etc.
uses three separate matrices

image deck - similar to MRI image slice.



Reading & writing images

GIF (Graphics Interchange Format)

[X, map] = gifread ('img.gif');

X: indexed image X
map: with associated colormap map.

gifwrite (X, map, '<filename>');

TIFF (tagged image file format).

[r, g, b] = tiffread ('rgb.tif') — returns R,G,B for rgb image. returns image & color map for indexed file

type = tiffread ('<filename>')

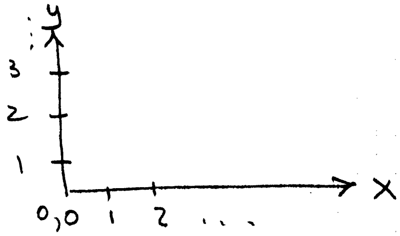
1 = binary
8 = indexed image
24 = RGB

tiffwrite (X, map, '<filename>');

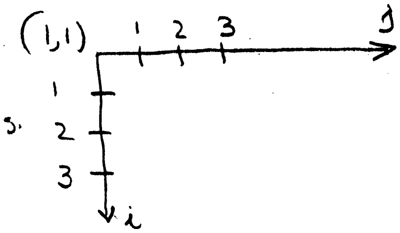
Can also do
- HDF
- BMP MS windows.
- PCX ZSoft Paint
- XWD (X-windows).

all work with indexed image matrices & colormaps.

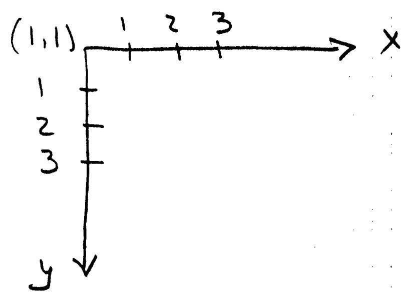
Coordinates:



cartesian
(matlab graphics routines).



matrix
matrix subscripts.



pixel
coordinate system used by image processing for almost everything

to read in an image.

load forest ← typically stored as X.

I = ind2gray(X, map) %convert to intensity

imhist (I, n) ← plot histogram.

\# of bins

B = imresize (I, [mrows, ncols], 'method')

nearest
bilinear
bicubic

B = imrotate (I, angle)

imshow (I, n) default is 256.

B = mfilter2 (h, A, filtmask).

output.

2 D filter

0's and 1's to mask where

imshow – display image.

imshow (X, map) indexed images

imshow (I, 64) display intensity image I with 64 gray levels.

imshow (BW, 2) binary images.

imshow (~BW, 2) display inverted image

imshow (R,G,B)

simple program

```
load kids
subplot (1,2,1), imshow (X, map), title ('Before Rotation')
subplot (1,2,2), imshow (imrotate (X, 35, 'crop'), map)
title ('After Rotation')
```

'crop' optional

subplot (m, n, 1) — makes first subarea active

divide graphics window into m x n sub areas.

B = imrotate (A, angle) — rotates by angle in ccw direction

B = imrotate (A, angle, 'method')

B = imrotate (A, angle, 'method', 'crop')

method:

- nearest — nearest neighbor interpolation
- bilinear — bilinear interpolation
- bicubic — bicubic interpolation

which we will talk about

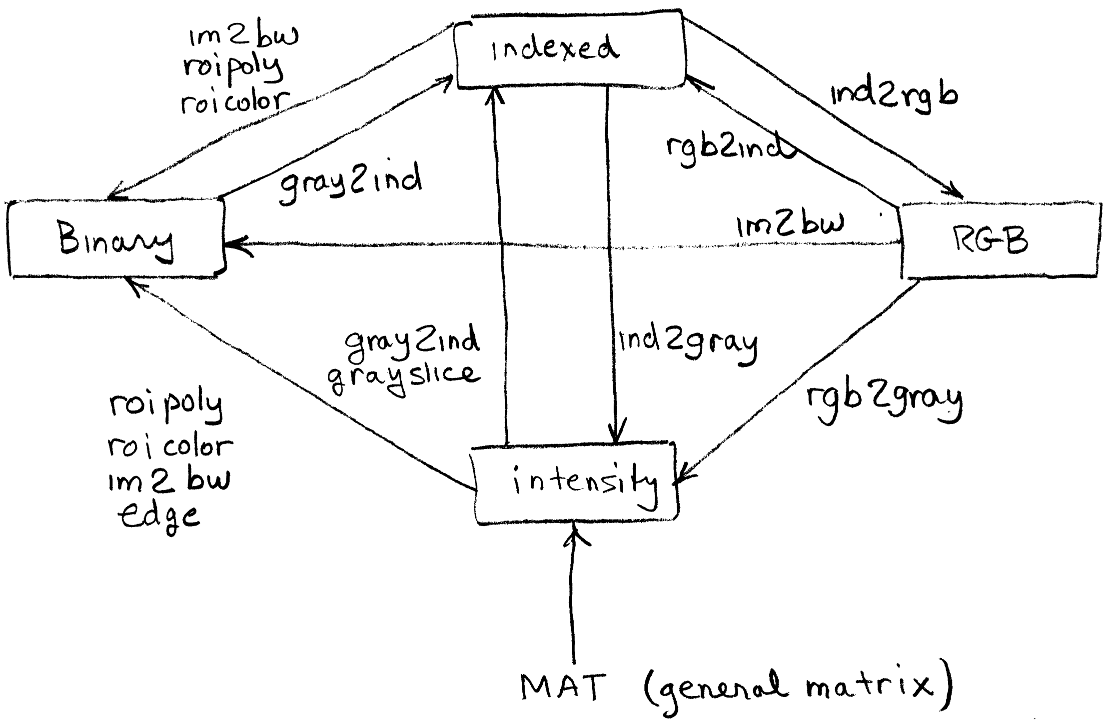
'crop' rotates but only returns central valid section which is same size as A.

imrotate (A, angle, ...) <u>displays</u> rotated image in current figure

```
load tire
Y = imrotate (X, 135, 'crop')
imshow (Y, map).
```

Images in MATLAB

- indexed images — pseudo color images which use a color map to convert intensity to color
- intensity images — double precision ϕ (black) to 1.0 white
- binary images — 0 = black, 1 = white
- RGB images — [R G B] three separate matrices



Reading and writing files

[x, map] = gifread ('img.gif');

indexed image X

associated color map

Graphics Interchange Format (AOL)

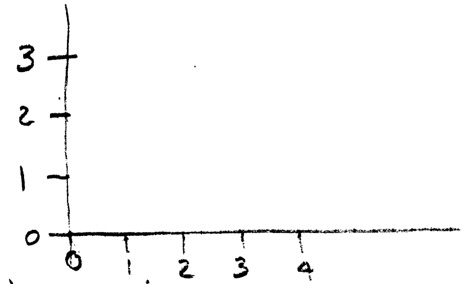
[r, g, b] = tiffread ('rgb.tif'); returns R,G,B for rgb image

returns image and color map for index file

type = tiffread ('img.tif'); 1 = binary, 8 = indexed, 24 = rgb

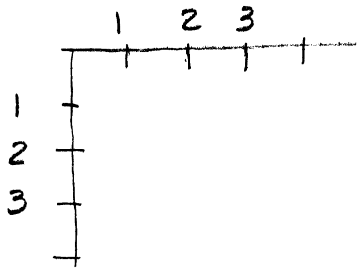
tiffwrite (x, map, 'rgb.tif');

coordinates

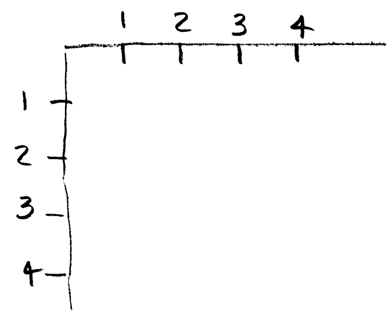


cartesian
MATLAB graphics routines

matrix
(matrix subscripts)



pixel coordinate
system used by
image processors.



MATLAB
- uses matrix only for direct matrix manipulation
- uses pixel coordinates for everything else.

Simple program to load in image

```
load forest;                          %stored as a matrix X
I = ind2gray (X, map);                % convert to intensity
imhist (I,n);                         % n = number of bins
B = imresize (I, [mrows, ncols], 'method')   % method = nearest
                                                          bilinear
                                                          bicubic
B= irotate (I, angle)
imshow (I,n)                          % n defaults to 256
B= mfilter 2 (h, A, filtmask);
```

output

2D filter

0's and 1's to mask the filter
usually want as all 1's

Date: Mon, 5 Feb 96 08:23:27 EST
From: dwilson@morph.EBME.CWRU.Edu (David Wilson - Department of Biomedical Engineering)
To: flm@po.cwru.edu
Subject: MATLAB template

Here is the template. I hope it works ...

dave

----- Begin Included Message -----

>From jabri Sun Feb 4 20:39:57 1996
Date: Sun, 4 Feb 96 20:39:41 EST
From: jabri (Kadri Jabri)
To: saa4@po, nab2@po, dlc2@po, rxh15@po, aoh2@po, tlh3@po,
pat@rosebud.lerc.nasa.gov, hxl2@po, txk17@po, jxl4@po, aap3@po,
jmr16@po, lbs5@po, dxw8@po, txw5@po, hxz@po, zxf2@po, ved@po,
derti@cibadiag.com, aod2@po, rsg2@po, yxh20@po, mdk5@po, rmm2@po,
dja4@po
Subject: MATLAB template
Cc: dwilson, flm@po, jabri
Content-Length: 12408

FROM: Dr. Wilson - EBME 512 announcement

You should find this template program useful for EBME 512/EEAP 431. It is a Matlab program that reads and image, rotates it, and displays the image before and after rotation. You do not require the image processing toolbox to run this code. Dr. Merat will assign a program that requires you to warp an image. This template program should get you started.

Below you will find Matlab *.m files and a description of where to find the code and images on servers. There is a description for both the MacIntosh and the PC. If you do not have access to images from the server, please email Kadri Jabri (jabri@morph.ebme.cwru.edu). He will email you the image.

Regards,
Dave

```
-----------------------------------cut here -----------------------------------
*How to access MATLAB on the CWRUnet:
        IBM PC
        Requirements:
        1.      The PC is on the CWRUnet.
        2.      The PC has Windows installed.
        Steps for running MATLAB:
        1.      under DOS, type "novell software win";
        2.      hit any key to continue until a menu window is on the screen;
        3.      select "Start Windows" from the menu;
        4.      select "F. Math&Statistics" from CWRUnet Software Library window
        5.      select "B. Matlab...".

        MACINTOSH
        Requirements:
                The Macintosh is on the CWRUnet.
        Steps for running MATLAB:
        1.      select "Chooser" from the Apple Menu;
        2.      select "Apple Share" at first, then "Library" in Apple Talk Zone          ,and
"Software Library" in file server;
```

3. select "Guest" and "OK" (The "CWRU Software" icon will be on the screen);
4. open "CWRU Software" folder, then "Math&Statistics" then"MATLAB"
5. click "MATLAB...".

Facilities in BME Department Computer Lab (Wickenden 425)
BME Department has one IBM PC with Windows installed and four PowerMac's available for running MATLAB.

*MATLAB template
All M-files and the test image "MRIbrain.mat" will be found on the EBME server (YEW).

IBM PC
1. Under DOS, type "novell yew ebme512";
2. password is "cluster".
3. type "cd H:\biomed\imaging\ebme512\PC".

MACINTOSH
1. select "Chooser" from the Apple Menu;
2. select "Apple Share" at first, then "CWRUnet" in Apple Talk Zone ,and "YEW" in the file server;
3. user name: "ebme512", password: "cluster", click "OK", and select " VOL1";
4. open "VOL1", "BIOMED", "IMAGING", "ebme512", and "MAC" folders;

```
-----------------------------------cut here ----------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                         EBME 512 - Spring 96                            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  MATLAB template for processing of images                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% load image to be processed
load MRIbrain                              % load image data
image(brain)                               % display image (matrix "brain")
colormap(gray(256))                        % adjust colormap to 256 gray levels
axis('image')                              % Adjust axis to image viewing
                                           %  (square pixels)

%% perform a 45 degree rotation on the image by calling the function imrot %%
%% You can perform different operations on the image by calling other      %%
%% functions, including ones you write yourself.                           %%

rotbrain = imrot(brain,45,'crop');

%% display resultant image
figure(2)
image(rotbrain)
colormap(gray(256))
axis('image')

-----------------------------------cut here ----------------------------------
function bout = imrotate(arg1,arg2,arg3,arg4)
%IMROTATE Rotate image.
%       B = IMROTATE(A,ANGLE,'method') rotates the image A by ANGLE
%       degrees.  The image returned B will, in general, be larger
%       than A.  Invalid values on the periphery are set to one
%       for indexed images or zero for all other image types. Possible
%       interpolation methods are 'nearest','bilinear' or 'bicubic'.
%       'bilinear' is the default for intensity images, otherwise
```

```
%       'nearest' is used if no method is given.
%
%       B = IMROTATE(A,ANGLE,'crop') or IMROTATE(A,ANGLE,'method','crop')
%       crops B to be the same size as A.
%
%       Without output arguments, IMROTATE(...) displays the rotated
%       image in the current axis.
%
%       See also IMRESIZE, IMCROP, ROT90.

%       Clay M. Thompson 8-4-92
%       Copyright (c) 1992 by The MathWorks, Inc.
%       $Revision: 1.14 $  $Date: 1993/09/01 21:27:38 $

if nargin<2, error('Requires at least two input parameters.'); end
if nargin<3,
  if isgray(arg1), case = 'bil'; else case = 'nea'; end
  docrop = 0;
elseif nargin==3,
  if isstr(arg3),
    method = [lower(arg3),'   ']; % Protect against short method
    case = method(1:3);
    if case(1)=='c', % Crop string
      if isgray(arg1), case = 'bil'; else case = 'nea'; end
      docrop = 1;
    else
      docrop = 0;
    end
  else
    error('''METHOD'' must be a string of at least three characters.');
  end
else
  if isstr(arg3),
    method = [lower(arg3),'   ']; % Protect against short method
    case = method(1:3);
  else
    error('''METHOD'' must be a string of at least three characters.');
  end
  docrop = 1;
end

% Catch and speed up 90 degree rotations
if rem(arg2,90)==0 & nargin<4,
  phi = rem(arg2,360);
  if phi==90,
    b = rot90(arg1);
  elseif phi==180,
    b = rot90(arg1,2);
  elseif phi==270,
    b = rot90(arg1,-1);
  else
    b = arg1;
  end
  if nargout==0, imshow(b), else bout = b; end
  return
end

phi = arg2*pi/180; % Convert to radians

% Rotation matrix
T = [cos(phi) -sin(phi); sin(phi) cos(phi)];
```

```
% Coordinates from center of A
[m,n] = size(arg1);
if ~docrop, % Determine limits for rotated image
  siz = ceil(max(abs([(n-1)/2 -(m-1)/2;(n-1)/2 (m-1)/2]*T))/2)*2;
  uu = -siz(1):siz(1); vv = -siz(2):siz(2);
else % Cropped image
  uu = (1:n)-(n+1)/2; vv = (1:m)-(m+1)/2;
end
nu = length(uu); nv = length(vv);

blk = bestblk([nv nu]);
nblks = floor([nv nu]./blk); nrem = [nv nu] - nblks.*blk;
mblocks = nblks(1); nblocks = nblks(2);
mb = blk(1); nb = blk(2);

rows = 1:blk(1); b = zeros(nv,nu);
for i=0:mblocks,
  if i==mblocks, rows = (1:nrem(1)); end
  for j=0:nblocks,
    if j==0, cols = 1:blk(2); elseif j==nblocks, cols=(1:nrem(2)); end
    if ~isempty(rows) & ~isempty(cols)
      [u,v] = meshgrid(uu(j*nb+cols),vv(i*mb+rows));
      % Rotate points
      uv = [u(:) v(:)]*T'; % Rotate points
      u(:) = uv(:,1)+(n+1)/2; v(:) = uv(:,2)+(m+1)/2;
      if case(1)=='n', % Nearest neighbor interpolation
        b(i*mb+rows,j*nb+cols) = interp6(arg1,u,v);
      elseif all(case=='bil'), % Bilinear interpolation
        b(i*mb+rows,j*nb+cols) = interp4(arg1,u,v);
      elseif all(case=='bic'), % Bicubic interpolation
        b(i*mb+rows,j*nb+cols) = interp5(arg1,u,v);
      else
        error(['Unknown interpolation method: ',method]);
      end
    end
  end
end

d = find(isnan(b));
if length(d)>0,
  if isind(arg1), b(d) = ones(size(d)); else b(d) = zeros(size(d)); end
end

if nargout==0,
  imshow(b), return
end
bout = b;

-----------------------------------cut here -----------------------------------
function [mb,nb] = bestblk(siz,k)
%BESTBLK Best block size for block processing.
%       BLK = BESTBLK([M N],K) returns the 1-by-2 block size BLK
%       closest to but smaller than K-by-K for block processing.
%
%       [MB,NB] = BESTBLK([M N],K) returns the best block size
%       as the two scalars MB and NB.
%
%       [...] = BESTBLK([M N]) returns the best block size smaller
%       than 100-by-100.
%
%       BESTBLK returns the M or N when they are already smaller
%       than K.
```

```
%
%       See also BLKPROC, SIZE.

%       Clay M. Thompson
%       Copyright (c) 1993 by The MathWorks, Inc.
%       $Revision: 1.6 $  $Date: 1994/03/04 19:54:04 $

if nargin==1, k = 100; end % Default block size

%
% Find possible factors of siz that make good blocks
%

% Define acceptable block sizes
m = floor(k):-1:floor(min(ceil(siz(1)/10),k/2));
n = floor(k):-1:floor(min(ceil(siz(2)/10),k/2));

% Choose that largest acceptable block that has the minimum padding.
[dum,ndx] = min(ceil(siz(1)./m).*m-siz(1)); blk(1) = m(ndx);
[dum,ndx] = min(ceil(siz(2)./n).*n-siz(2)); blk(2) = n(ndx);

if nargout==2,
  mb = blk(1); nb = blk(2);
else
  mb = blk;
end

-----------------------------------cut here ----------------------------------
function b = blkproc(a,block,border,P0,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10)
%BLKPROC Process an image in blocks.
%       B = BLKPROC(A,[M N],'fun') processes the image A by applying
%       the function 'fun' to each distinct M-by-N block of A.  The
%       results are assembled to create B.  The function 'fun' should
%       operate on an M-by-N block, x, and return a matrix of the
%       same size, y: y = fun(x);
%
%       Up to 10 additional parameters can be passed to the function
%       using B = BLKPROC(A,[M N],'fun',P1,P2,P3,...) in which case
%       'fun' is called using y = fun(x,P1,P2,P3,...).  A is padded,
%       if necessary, so that its size is a multiple of M-by-N.
%
%       B = BLKPROC(A,[M N],[K L],'fun',...) processes the image A
%       in (M+2*K)-by-(N+2*L) blocks that are formed by extending
%       the original M-by-N block from A by K and L in the row and
%       column directions.  This border is padded, if necessary,
%       at the edges of A.  In this case, the function 'fun' should
%       operate on the (M+2*K)-by-(N+2*L) block and return either an
%       M-by-N result, or a (M+2*K)-by-(N+2*L) result whose M-by-N
%       center will be incorporated into B.  A border can be used to
%       reduce edge effects.
%
%       At the edges, the blocks are formed by padding with ones
%       if A is an indexed image or with zeros otherwise.
%
%       See also BLK2COL, COL2BLK, HOOD2COL, COLFILT, NLFILTER.

%       Clay M. Thompson 10-6-92
%       Copyright (c) 1992 by The MathWorks, Inc.
%       $Revision: 1.11 $  $Date: 1993/09/30 17:16:34 $

error(nargchk(3,14,nargin));
```

```
if nargin==3,
  FUN = border;
  border = [0 0];
  if ~any(FUN<48), fcall = [FUN,'(x)']; else fcall = FUN; end
else
  if isstr(P0),
    FUN = P0;
    % Form call string.
    params = [];
    for n=5:nargin
      params = [params,',P',int2str(n-4)];
    end
    if ~any(FUN<48), fcall = [FUN,'(x',params,')']; else fcall = FUN; end
  else
    FUN = border;
    border = [0 0];
    % Form call string and shift parameters
    params = [];
    for n=4:nargin
      params = [params,',P',int2str(n-3)];
      eval(['P',int2str(nargin-n+1),'=P',int2str(nargin-n),';']);
    end
    if ~any(FUN<48), fcall = [FUN,'(x',params,')']; else fcall = FUN; end
  end
end

% Expand A: Add border, pad if size(a) is not divisible by block.
[ma,na] = size(a);
mpad = rem(ma,block(1)); if mpad>0, mpad = block(1)-mpad; end
npad = rem(na,block(2)); if npad>0, npad = block(2)-npad; end
if isind(a),
  aa = ones(ma+mpad+2*border(1),na+npad+2*border(2));
else
  aa = zeros(ma+mpad+2*border(1),na+npad+2*border(2));
end
aa(border(1)+(1:ma),border(2)+(1:na)) = a;

%
% Process each block in turn.
%
m = block(1) + 2*border(1);
n = block(2) + 2*border(2);
mblocks = (ma+mpad)/block(1);
nblocks = (na+npad)/block(2);
b = zeros(ma+mpad,na+npad);
arows = 1:m; acols = 1:n;
brows = 1:block(1); bcols = 1:block(2);
xrows = brows + border(1); xcols = bcols + border(2);
mb = block(1); nb = block(2);
for i=0:mblocks-1,
  for j=0:nblocks-1,
    x = aa(i*mb+arows,j*nb+acols);
    c = eval(fcall);
    if all(size(c)==size(x)),
      b(i*mb+brows,j*nb+bcols) = c(xrows,xcols);
    elseif all(size(c)==size(x)-2*border),
      b(i*mb+brows,j*nb+bcols) = c;
    else
      error('Block returned by FUN is the wrong size.');
    end
  end
end
```

```
b = b(1:ma,1:na);

-----------------------------------cut here -----------------------------------
function y = isgray(x)
%ISGRAY True for intensity images.
%       ISGRAY(A) returns 1 if A is an intensity image and 0 otherwise.
%       An intensity image contains values between 0.0 and 1.0.
%
%       See also ISIND, ISBW.

%       Clay M. Thompson 2-25-93
%       Copyright (c) 1993 by The MathWorks, Inc.
%       $Revision: 1.4 $  $Date: 1993/08/18 03:11:32 $

y = min(min(x))>=0 & max(max(x))<=1;

-----------------------------------cut here -----------------------------------
function y = isind(x)
%ISIND True for indexed images.
%       ISIND(A) returns 1 if A is an indexed image and 0 otherwise.
%       An indexed image contains integer values that are indices into
%       an associated colormap.
%
%       See ISGRAY, ISBW.

%       Clay M. Thompson 2-25-93
%       Copyright (c) 1993 by The MathWorks, Inc.
%       $Revision: 1.6 $  $Date: 1994/03/04 19:53:26 $

y = (min(min(x))>=1 & max(max(x))<=256) & all(all(abs(x-floor(x))<eps));

------------------------------------ END ---------------------------------------


----- End Included Message -----
```

```
  if l ~= m*n, l, error('HSI image file is wrong length'), end
  % Image elements are colormap indices, so start at 1.
  X = X'+1;
  fclose(fp);
else
  error('Image file name must end in "raw".')
end
```

```
function E = canny(I, sd, th1, th0);
%CANNY  Edge detection.
%       E = canny(I) finds the edges in a gray scaled image I using the Canny
%       method, and returns an image E where the edges of I are marked by
%       non-zero intensity values.
%
%       E = canny(I, SD) uses SD as the standard deviation for the gaussian
%       filtering phase. Default is 1 pixel.
%
%       E = canny(I, SD, TH1) uses TH1 for the higher hysteresis threshold.
%       Default is 0.5 times the strongest edge. Setting TH1 to zero will
%       avoid the (sometimes time consuming) hysteresis.
%
%       E = canny(I, SD, TH1, TH0) uses TH1 for the lower hysteresis threshold.
%       Default is 0.1 times the strongest edge.
%
%       See also EDGE (in the Image Processing toolbox).

%       Oded Comay, Feb 23, 1996 - Original version.
%       Tel Aviv University
%
%       Oded Comay, Feb 27, 1997 - Hysteresis added.


if nargin<2 sd=  1; end; if isempty(sd),  sd=  1; end;
if nargin<3 th1= .5; end; if isempty(th1), th1= .5; end;
if nargin<4 th0= .1; end; if isempty(th0), th0= .1; end;

x= -5*sd:sd*5;
g= exp(-0.5/sd^2*x.^2);                  % Create a normalized Gaussian
g= g(g>max(g)*.005); g= g/sum(g(:));
dg= diff(g);                             % Gaussian first derivative

dx= abs(conv2(I, dg, 'same'));           % X/Y edges
dy= abs(conv2(I, dg', 'same'));

[ny, nx]= size(I);
                                         % Find maxima
dy0= [dy(2:ny,:); dy(ny,:)]; dy2= [dy(1,:); dy(1:ny-1,:)];
dx0= [dx(:, 2:nx) dx(:,nx)]; dx2= [dx(:,1) dx(:,1:nx-1)];
```

```
peaks= find((dy>dy0 & dy>dy2) | (dx>dx0 & dx>dx2));
e= zeros(size(I));
e(peaks)= sqrt(dx(peaks).^2 + dy(peaks).^2);

e(:,2)   = zeros(ny,1);    e(2,:)= zeros(1,nx); % Remove artificial edges
e(:,nx-2)= zeros(ny,1); e(ny-2,:)= zeros(1,nx);
e(:,1)   = zeros(ny,1);    e(1,:)= zeros(1,nx);
e(:,nx)  = zeros(ny,1);   e(ny,:)= zeros(1,nx);
e(:,nx-1)= zeros(ny,1); e(ny-1,:)= zeros(1,nx);
e= e/max(e(:));

if th1 == 0, E= e; return; end                  % Perform hysteresis
E(ny,nx)= 0;

p= find(e >= th1);
while length(p)
  E(p)= e(p);
  e(p)= zeros(size(p));
  n= [p+1 p-1 p+ny p-ny p-ny-1 p-ny+1 p+ny-1 p+ny+1]; % direct neighbors
  On= zeros(ny,nx); On(n)= n;
  p= find(e > th0 & On);
end
```

Reading Assignment

Chapter 2

2.5 Imaging Geometry Problem 2.16

Chapter 5

5.9 Geometric Transformations Problem 5.16 ?

Some MATLAB functions for input/output

input

[x, y] = ginput (n) gets n points

displays the graph window
puts up a cross hair
gets n coordinates from graph window, press mouse button to get data

[x, y] = ginput gathers points until you press the return key

other forms

plot (x, y) plots vector x versus vector y

plot (x1, y1, x2, y2) plots two lines

plot (x1, y1, ':', x2, y2, '+') plots dotted line for first curve
point for second curve

plot (x1, y1, 'r', x2, y2, '+g') plots solid red line for first curve
green + symbols for second curve

line-types		point types		color	
solid	-	point	.	red	r
dashed	--	plus	+	green	g
dotted	:	star	*	blue	b
dash	-.	circle	o	white	w
		x-mark	x		

imshow - display image.

imshow (X, map)   indexed images

imshow (I, 64)   display intensity image I with 64 gray levels.

imshow (BW, 2)   binary images.

imshow (~BW, 2)   display inverted image

imshow (R, G, B)

simple program

```
load kids
subplot (1,2,1), imshow (X, map), title ('Before Rotation')
subplot (1,2,2), imshow (imrotate (X, 35, 'crop'), map)
title ('After Rotation')
```

'crop' — optional

subplot (m, n, 1) — 1 makes first subarea active

m, n: divide graphics window into m x n sub areas.

B = imrotate (A, angle) — rotates by angle in CCW direction

B = imrotate (A, angle, 'method')

B = imrotate (A, angle, 'method', 'crop')

method:
- nearest — nearest neighbor interpolation
- bilinear — bilinear interpolation
- bicubic — bicubic interpolation

which we will talk about

'crop' rotates but only returns central valid section which is same size as A.

imrotate (A, angle, ...)   <u>displays</u> rotated image in current figure

```
load tire
Y = imrotate (X, 135, 'crop')
imshow (Y, map).
```

Here is a little MATLAB program you can run to see histogram ~~equalization~~, stretching

```
load forest
I = ind2gray (X, map) ;     % forest is an indexed image
                              this converts it
J = imadjust (I, [0. 0.5], [], []) ;
```

input image

range of values in input image

bottom : top in output image.
empty matrices mean use defaults
[0 1], [1]

matrix for color.

value of γ
uses transform
$y = x^{\gamma}$
$\gamma = 1$ does linear stretching

```
subplot (2,2,1) , imhist (I, 128);
subplot (2,2,2) , imshow (I, 128);
subplot (2,2,3), imhist (J, 128);
subplot (2,2,4), imshow (J, 128);
```

1	2
3	4

2x2

for equalization use.

```
J = histeq (I, n)
```

# of gray levels.
default levels

MATLAB filter types

h = fspecial ('type', params)

'gaussian', n, sigma $\qquad$ n = n x n
sigma is mainlobe width σ in pixels.

'sobel' $\qquad$ gives 3x3 for vertical derivative $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
h' gives 3x3 for horiz. derivative
i.e. h' = fspecial ('sobel')

'laplacian', alpha $\qquad$ $0 \le \alpha \le 1$ controls shape.

$$\nabla^2 \approx \frac{4}{\alpha+1} \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

'log', n, sigma $\qquad$ n x n laplacian of Gaussian
with Gaussian mainlobe σ pixels

'prewitt' $\qquad$ $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$ to rotate use h'

'averaging', n $\qquad$ returns n x n averaging filter
$\frac{\text{ones}(n,n)}{n^2}$?

'unsharp', alpha

$$\frac{1}{\alpha+1} \begin{bmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{bmatrix}$$

Example

```
load trees
I = ind2gray (X, map);
h = fspecial ('sobel');
A = filter2 (h, I);
```

h: 2 dimensional mask.
I: input image