Lecture 3: Binary image analysis

Thursday, Sept 6

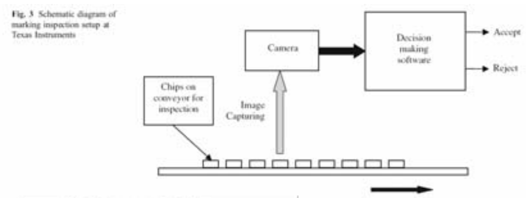
- Sudheendra's office hours
  - Mon, Wed 1-2 pm
  - ENS 31NR
- Forsyth and Ponce book

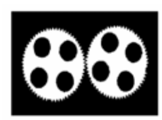
Binary images

- Two pixel values
- Foreground and background
- Regions of interest

1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	0	1	0	0	0	1
1	1	0	1	0	1	1	1

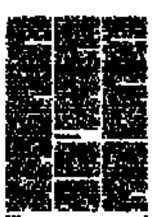
Constrained image capture setting

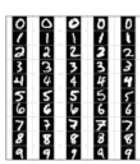




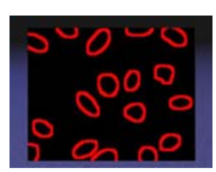
R. Nagarajan et al. A real time marking inspection scheme for semiconductor industries, 2006

Documents, text

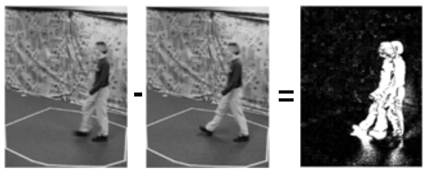




Medical, bio data



Intermediate low-level cues



Motion

Edges

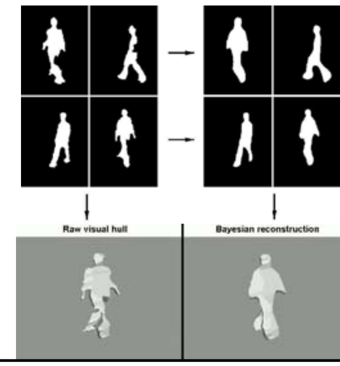
Orientation

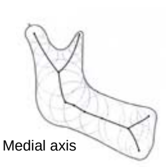
NASA robonaut
http://robonaut.jsc.nasa.gov/status/October_prime.htm

Shape

visual hulls

Silhouette





Medial axis

Outline

- Thresholding
- Connected components
- Morphological operators
- Region properties
  - Spatial moments
  - Shape
- Distance transforms
  - Chamfer distance

Thresholding

- Grayscale -> binary mask
- Useful if object of interest's intensity distribution is distinct from background

$$F_T[i,j] = \begin{cases} 1 & \text{if } F[i,j] \geq T \\ 0 & \textit{otherwise.} \end{cases}$$

$$F_T[i,j] = \begin{cases} 1 & \text{if } T_1 \leq F[i,j] \leq T_2 \\ 0 & \textit{otherwise.} \end{cases}$$

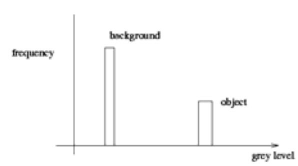
$$F_T[i,j] = \begin{cases} 1 & \text{if } F[i,j] \in Z \\ 0 & \textit{otherwise.} \end{cases}$$

- Example http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FITZGIBBON/simplebinary.html

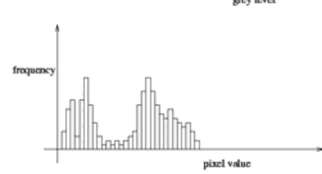
Selecting thresholds

- Partition a bimodal histogram
- Fit Gaussians
- Dynamic or local thresholds

A nice case: bimodal intensity histograms



Ideal histogram, light object on dark background



Actual observed histogram with noise

Images: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT2/node3.html

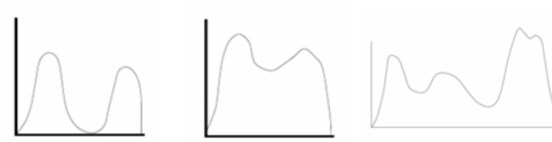
A nice case: bimodal intensity histograms

- Example
- Thresholding a bimodal histogram

- Otsu method (1979) : automatically select threshold by minimizing the weighted *within-group variance* of the two groups of pixels separated by the threshold.

Not so nice cases

- Threshold selection is an art, not a science



Two distinct modes   Overlapped modes

Shapiro and Stockman

Connected components

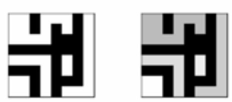
- Identify distinct regions

1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	0	1	0	0	0	1
1	1	0	1	0	1	1	1

a) binary image

1	1	0	1	1	1	0	2
1	1	0	1	0	1	0	2
1	1	1	1	0	0	0	2
0	0	0	0	0	0	0	2
3	3	3	3	0	4	0	2
0	0	0	3	0	4	0	2
5	5	0	3	0	0	0	2
5	5	0	3	0	2	2	2

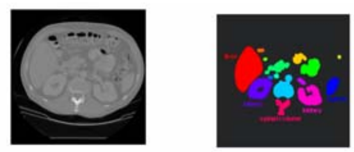
b) connected components labeling



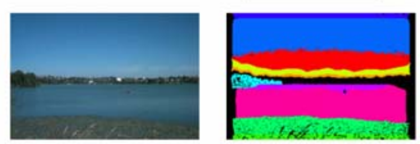
c) binary image and labeling, expanded for viewing

Shapiro and Stockman

Connected components



connected components of 1's from thresholded image

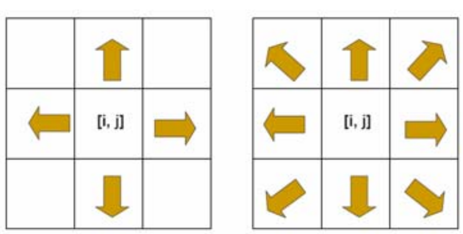


connected components of cluster labels

P. Duygulu

Connectedness

- Which pixels are considered neighbors



4-connected   8-connected

Image from http://www-ee.uta.edu/Online/Devarajan/ee6358/BIP.pdf

Connected components

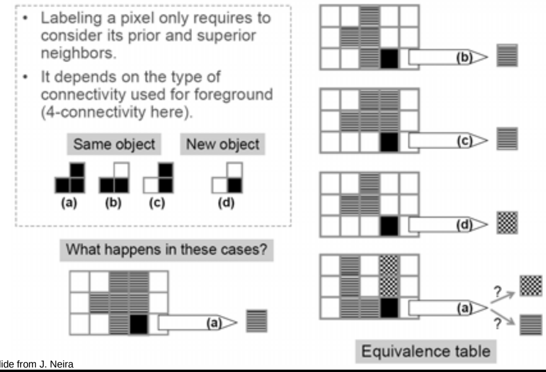
- Various algorithms to compute
  - Recursive (in memory)
  - Two rows at a time (image not necessarily in memory)
  - Parallel propagation strategy

Recursive connected components

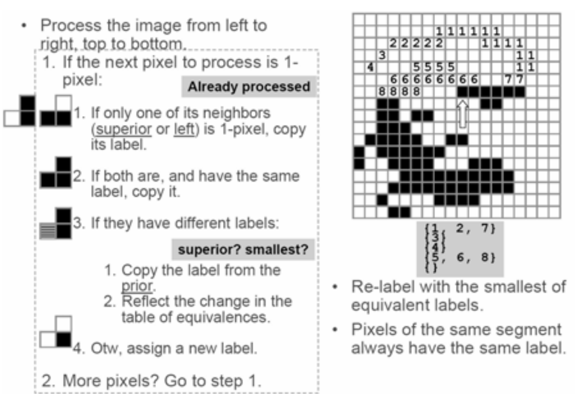
- Find an unlabeled pixel, assign it a new label
- Search to find its neighbors, and recursively repeat to find their neighbors til there are no more
- Repeat

- Demo http://www.cosc.canterbury.ac.nz/mukundan/covn/Label.html

Sequential connected components



Slide from J. Neira



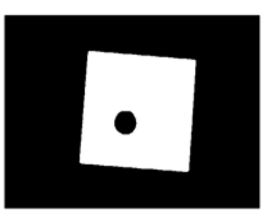
Morphological operators

- Dilation
- Erosion

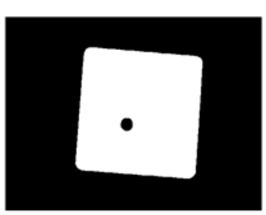
- Open, close

Dilation

- Expands connected components
- Grow features
- Fill holes



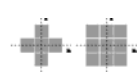
Before dilation



After dilation

Structuring elements

- Masks of varying shapes used to perform morphology



etc

- Scan mask across foreground pixels to transform the binary image

Dilation / Erosion

- Dilation: if current pixel is foreground, set all pixels under S to foreground in output (OR)
- Erosion: if every pixel under S is foreground, leave as is; otherwise, set current pixel to background in output

Example for Dilation (1D)

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element	1	1	1							
Output Image	1	1								

$$g(x) = f(x) \oplus SE$$

Adapted from T. Moeslund

Example for Dilation

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element	1	1	1							
Output Image	1	1								

Example for Dilation

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element		1	1	1						
Output Image	1	1	0							

Example for Dilation

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element			1	1	1					
Output Image	1	1	0	0						

Example for Dilation

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element				1	1	1				
Output Image	1	1	0	1	1	1				

Example for Dilation

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element					1	1	1			
Output Image	1	1	0	1	1	1	1			

Example for Dilation

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element						1	1	1		
Output Image	1	1	0	1	1	1	1	1		

Example for Dilation

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element							1	1	1	
Output Image	1	1	0	1	1	1	1	1		

Example for Dilation

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element								1	1	1
Output Image	1	1	0	1	1	1	1	1	1	1

The object gets bigger and holes are filled!

Erosion

- Erode connected components
- Shrink features
- Remove bridges, branches, noise

Before erosion

After erosion

Example for Erosion (1D)

Input image		1	0	0	0	1	1	1	0	1	1
Structuring Element	1	1	1								
Output Image		0									

$$g(x) = f(x) \ominus SE$$

Example for Erosion (1D)

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	1	1

$$g(x) = f(x) \Theta SE$$

Output Image

0	0								

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	1	1

Output Image

0	0	0							

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	1	1

Output Image

0	0	0	0						

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	1	1

Output Image

0	0	0	0	0					

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	1	1

Output Image

0	0	0	0	0	1				

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	1	1

Output Image

0	0	0	0	0	1	0			

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

						1	1	1	

Output Image

0	0	0	0	0	1	0	0		

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

							1	1	1

Output Image

0	0	0	0	0	1	0	0	0	

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

								1	1

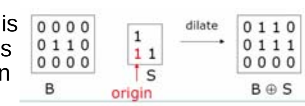
Output Image

0	0	0	0	0	1	0	0	0	1

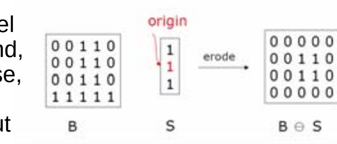
The object gets smaller

Dilation / Erosion

- Dilation: if current pixel is foreground, set all pixels under S to foreground in output (OR)



- Erosion: if every pixel under S is foreground, leave as is; otherwise, set current pixel to background in output

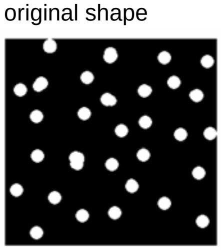


Images by P. Duygulu

Opening

- Erode, then dilate
- Remove small objects, keep original shape


Before opening


After opening

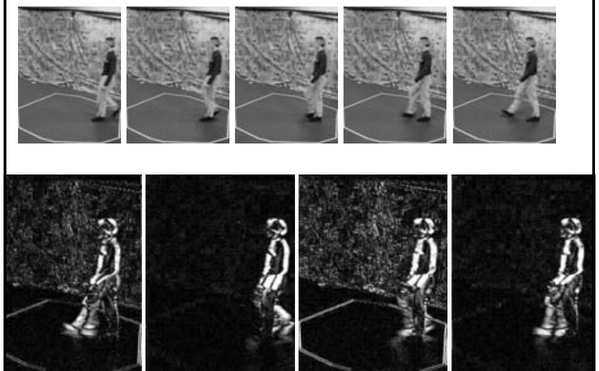
Closing

- Dilate, then erode
- Fill holes, but keep original shape

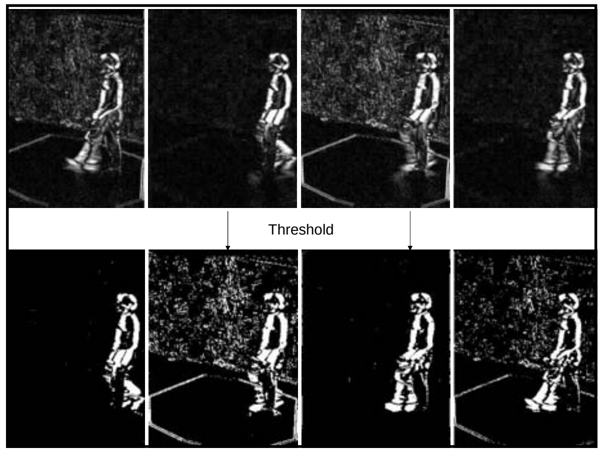

Before closing

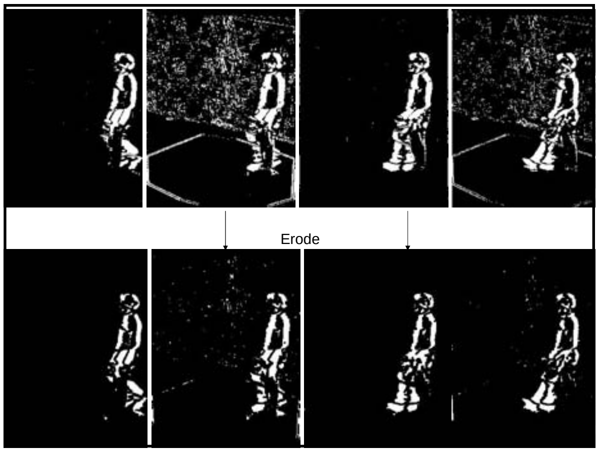

After closing

Application: blob tracking



Absolute differences from frame to frame

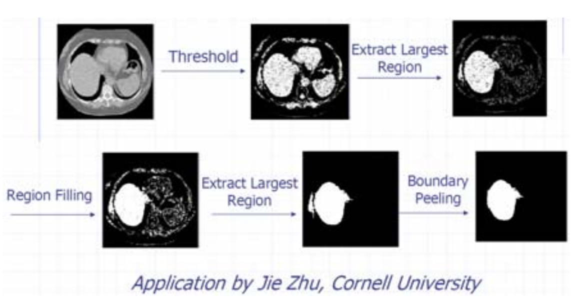




Application: blob tracking

- Background subtraction + blob tracking

Application: segmentation of a liver



Slide credit: Li Shen

Region properties

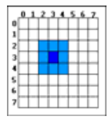
Some useful features can be extracted once we have connected components, including

- Area
- Centroid
- Extremal points, bounding box
- Circularity
- Spatial moments

Area and centroid

- We denote the set of pixels in a region by R.
- **assuming square pixels:**
  area:

$$A = \sum_{(r,c)\in R} 1$$



centroid:

$$\bar{r} = \frac{1}{A} \sum_{(r,c)\in R} r$$
$$\bar{c} = \frac{1}{A} \sum_{(r,c)\in R} c$$

- $(\bar{r}, \bar{c})$ is generally not a pair of integers.

Shapiro & Stockman

Circularity

a second measure uses variation off of a circle
circularity(2):

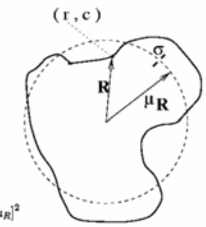
$$C_2 = \frac{\mu_R}{\sigma_R}$$

where μ_R and $\sigma_R{}^2$ are the mean and variance of the distance from the centroid of the shape to the boundary pixels (r_k, c_k).
mean radial distance:

$$\mu_R = \frac{1}{K} \sum_{k=0}^{K-1} \|(r_k, c_k) - (\bar{r}, \bar{c})\|$$

variance of radial distance:

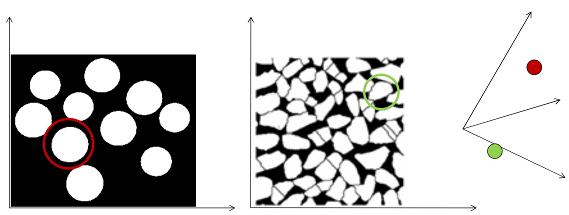
$$\sigma_R^2 = \frac{1}{K} \sum_{k=0}^{K-1} [\|(r_k, c_k) - (\bar{r}, \bar{c})\| - \mu_R]^2$$



[Haralick]

Shapiro & Stockman

Invariant descriptors



[a1, a2, a3,…]

[b1, b2, b3,…]

Feature space distance

Often want features independent of position, orientation, scale.

Central moments

S is a subset of pixels (region).
Central (j,k)th moment defined as:

$$\mu_{jk} = \sum_{(x,y)\in S} (x - \bar{x})^j (y - \bar{y})^k$$

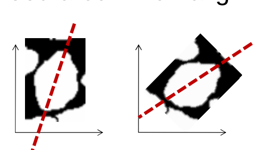
- Invariant to translation of S.

Central moments

- 2nd central moment: variance
- 3rd central moment: skewness
- 4th central moment: kurtosis

Axis of least second moment

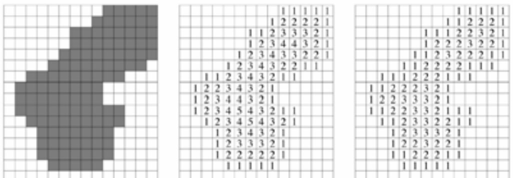
- Invariance to orientation?
  Need a common alignment



Axis for which the squared distance to 2d object points is minimized.

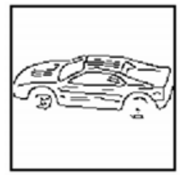
Distance transform

- Image reflecting distance to nearest point in point set (e.g., foreground pixels).



4-connected adjacency | 8-connected adjacency

Distance transform



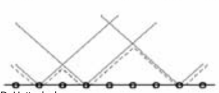
Edge image

Distance transform image

Distance transform (1D)

Two pass O(n) algorithm for 1D L_1 norm

1. Initialize: For all j
   $D[j] \leftarrow 1_P[j]$
2. Forward: For j from 1 up to n-1 | 1 | 0 |
   $D[j] \leftarrow \min(D[j], D[j-1]+1)$
3. Backward: For j from n-2 down to 0 | 0 | 1 |
   $D[j] \leftarrow \min(D[j], D[j+1]+1)$

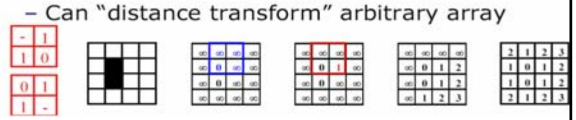


∞	0	∞	0	∞	∞	∞	∞	0	∞
∞	0	1	0	1	2	3	0	1	
1	0	1	0	1	2	1	0	1	

Adapted from D. Huttenlocher

Distance Transform (2D)

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to left
    - Bwd pass finds closest below and to right
- Note nothing depends on 0,∞ form of initialization
  - Can "distance transform" arbitrary array

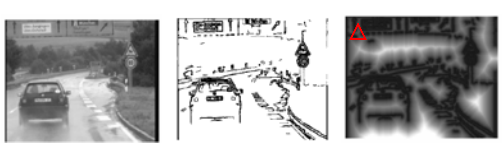


Adapted from D. Huttenlocher

Chamfer distance

- Average distance to nearest feature

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$



Edge image | Distance transform image

D. Gavrila, DAGM 1999

Chamfer distance

More on this and other distances later

Edge image | Distance transform image

D. Gavrila, DAGM 1999

Generalized distance transforms

- Same forward/backward algorithm applicable with different initialization
- Initialize with function values F(x,y):

Generalized Distance Transform

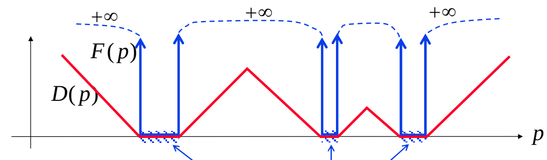
$$D(p) = \min_{q \in I} (\| p - q \| + F(q))$$

Distance Transform vs. Generalized Distance Transform

- Assuming $F(p) = \begin{cases} 0 & if\ \ pixel\ \ p\ is\ image\ \ feature \\ \infty & O.W. \end{cases}$

then $D(p) = \min_q \{\| p - q \| + F(q)\} = \min_{q:F(q)=0} \| p - q \|$

is standard *Distance Transform* (of image features)



Slide credit Y. Boykov

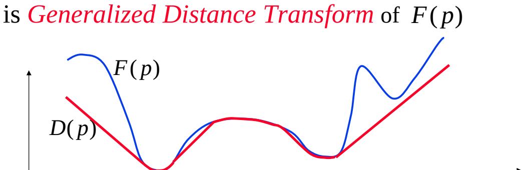
Distance Transform vs. Generalized Distance Transform

- For general $F(p)$

$$D(p) = \min_q \{\| p - q \| + F(q)\}$$

Location of q is close to p, and F(q) is small there

is *Generalized Distance Transform* of $F(p)$



F(p) **may represent non-binary image features (e.g. image intensity gradient)**

Slide credit Y. Boykov

Binary images

- Pros
  - Can be fast to compute, easy to store
  - Simple processing techniques available
  - Lead to some useful compact shape descriptors
- Cons
  - Hard to get “clean” silhouettes, noise common in realistic scenarios
  - Can be too coarse of a representation
  - Not 3d

Matlab

- N = HIST(Y,M)
- L = BWLABEL(BW,N);
- STATS = REGIONPROPS(L,PROPERTIES) ;
  - 'Area'
  - 'Centroid'
  - 'BoundingBox'
  - 'Orientation‘, …
- IM2 = imerode(IM,SE);
- IM2 = imdilate(IM,SE);
- IM2 = imclose(IM, SE);
- IM2 = imopen(IM, SE);
- [D,L] = bwdist(BW,METHOD);

- Everything is matrix

```
%%%% Matrix Definition 1 %%%%
A = [1 2 3 4;5 6 7 8];

%%%% Matrix Definition 2 %%%%
A = [1:1:4; 5:1:8];

%%%% Matrix Definition 3 %%%%
for i = 1:2
    for j = 1:4
        A(i,j) = (i-1)*4+j;
    end
end

%%%% Matrix Definition 4 %%%%
A = [];
A = zeros(n,n); %ones zeros eye rand randn
```

Tutorial adapted from W. Freeman, MIT 6.896

- Matrix index

```matlab
%%%% Matrix index
A = magic(4);
    >> A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
A(1:2, 1:2)
    >> ans =
    16     2
     5    11
A(:,1)
    >> ans =
    16
     5
     9
     4
A(1,:)
    >> ans =
    16     2     3    13
A([6 7])
    >> ans =
    11     7
```

- Manipulate matrices

```matlab
A = [1 NaN 2 NaN;3 Inf 4 Inf]; %1/0=Inf 0/0=NaN
%%%% replace Nan/Inf with 0
for i = 1:size(A,1)
    for j = 1:size(A,2)
        if isnan(A(i,j)) | isinf(A(i,j))
            A(i,j) = 0;
        end
    end
end

%%%% 'find'
a = [0 1 0 2;0 3 0 4]
    >> a =
     0     1     0     2
     0     3     0     4
find(a)
    >> ans = [3 4 7 8]'
[ii jj] = find(a);
    >> ii = [1 2 1 2]
    >> jj = [2 2 4 4]

nanflag = isnan(A)
    >> nanflag =
     0     1     0     1
     0     0     0     0
infflag = isinf(A)
    >> infflag =
     0     0     0     0
     0     1     0     1
```

- Manipulate matrices

```matlab
%%%% Matrix operation
A = [1 2;3 4];
B = [1 1;1 -1];
A*B
    >> ans =
     3    -1
     7    -1
A.*B
    >> ans =
     1     2
     3    -4
A/B %= A * inv(B)
    >>ans =
   1.5  -0.5
   3.5  -0.5
A./B
    >> ans =
     1     2
     3    -4
```

```matlab
%%%% Matrix concatenation
A = [1 2];B = [3 4];
C = [A B]
    >>C =
     1     2     3     4
C = [A;B]
    >>C =
     1     2
     3     4
```

```matlab
A = [1 2];
A*A' = 5
A.*A =
     1     4
sum(A.*A) = 5
```

- Scripts and functions
  - Scripts are m-files containing MATLAB statements
  - Functions are like any other m-file, but they accept arguments
  - Name the function file the same as the function name

myfunction.m

```matlab
function y = myfunction(x)
% Function of one argument with one return value

a = [-2 -1 0 1];              % Have a global variable of the same name
y = a + x;
```

myotherfunction.m

```matlab
function [y, z] = myotherfunction(a, b)
% Function of two arguments with two return values

y = a + b;
z = a - b;
```

- Try to code in matrix ways

```matlab
%use for loops
A = [1 2 3 4];
for i = 1:4
    A(i) = A(i)^2;
end
%or use matrix
A = [1 2 3 4];
A = A.^2;
```

```matlab
%use for loops
A = [1 2 3 4;5 6 7 8];
sum(A)
    >> ans =
     6     8    10    12
ASUM = sum(A,2)
    >> ASUM =
    10
    26
for i = 1:size(A,1)
    for j = 1:size(A,2)
        APROB(i,j) = A(i,j)/ASUM(i);
    end
end

%use matrix
A = [1 2 3 4;5 6 7 8];
ASUM = sum(A,2)
APROB = A./repmat(ASUM, 1, size(A,2));

repmat(ASUM, 1, size(A,2))
    >> ans =
    10    10    10    10
    26    26    26    26
```

- whos
- help
- lookfor
- clear / clear x
- save
- load