Image Processing

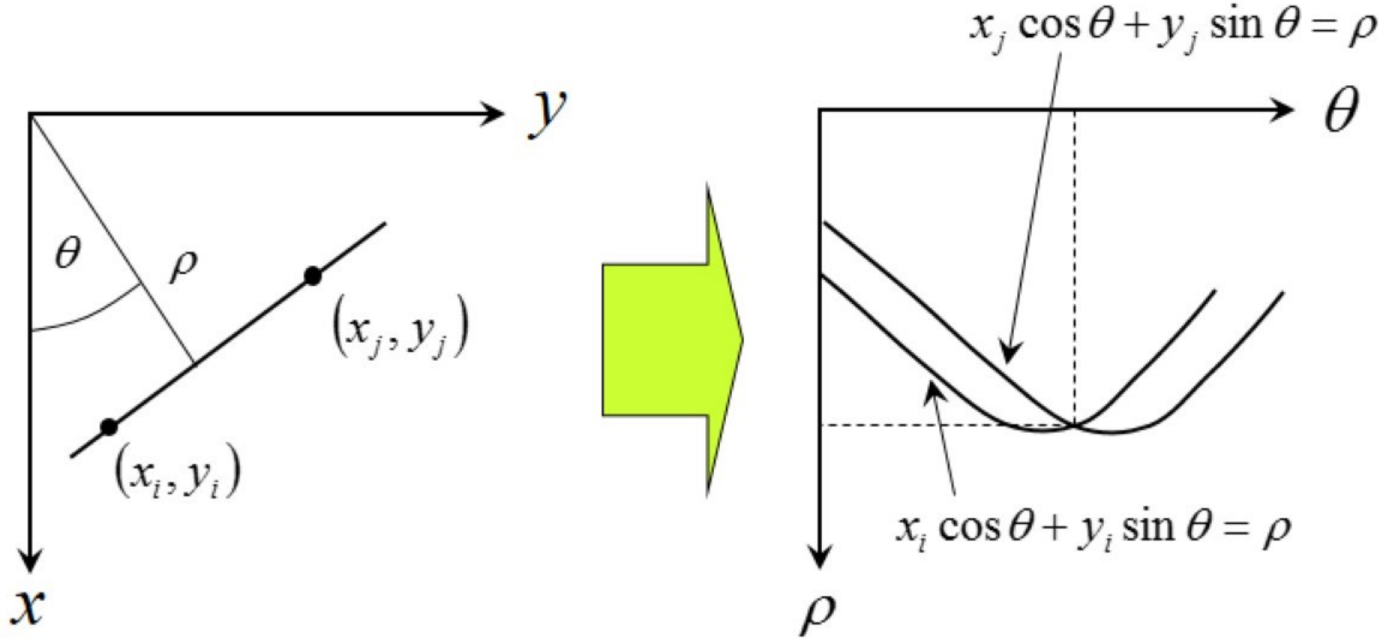
Image Segmentation (Part II)

Pattern Recognition and Image Processing Laboratory (Since 2012)

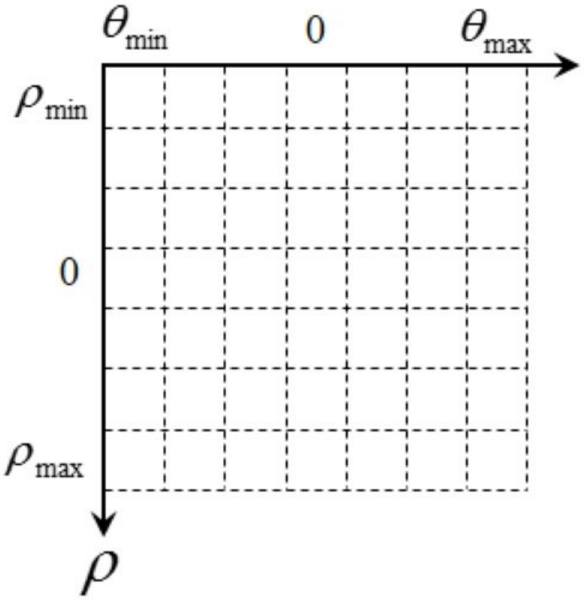
Line Detection Using the Hough Transform

One approach that can be used to find and link line segments in an image is the Hough transform.

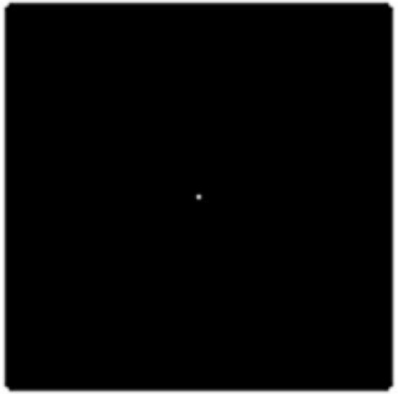
Line Detection Using the Hough Transform



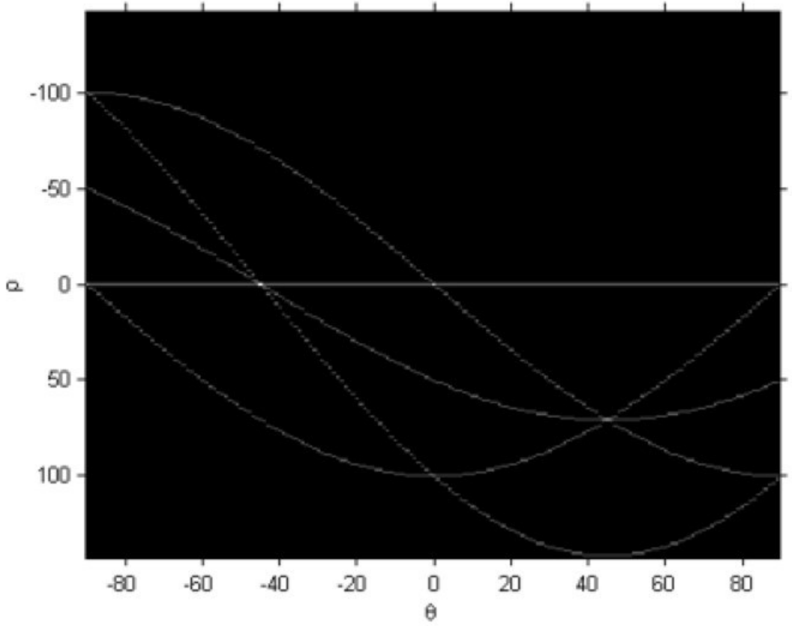
Line Detection Using the Hough Transform



Line Detection Using the Hough Transform



Binary image with five dots



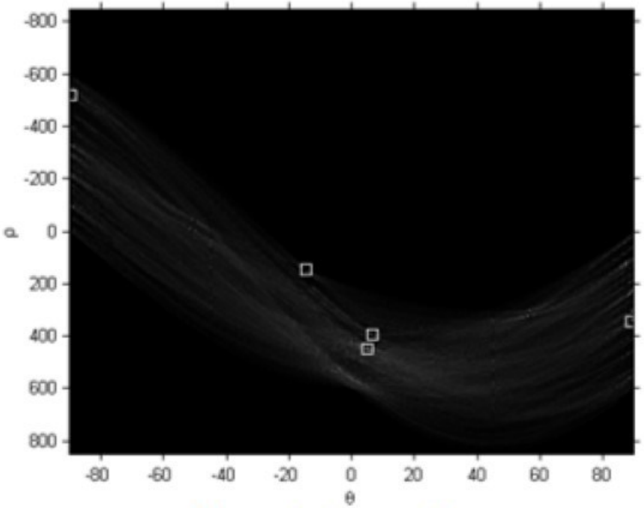
Hough transform

ex_hough_5dots.m

Line Detection Using the Hough Transform

- **Line segments corresponding to the Hough transform peaks**

Edge detection image



Hough transform

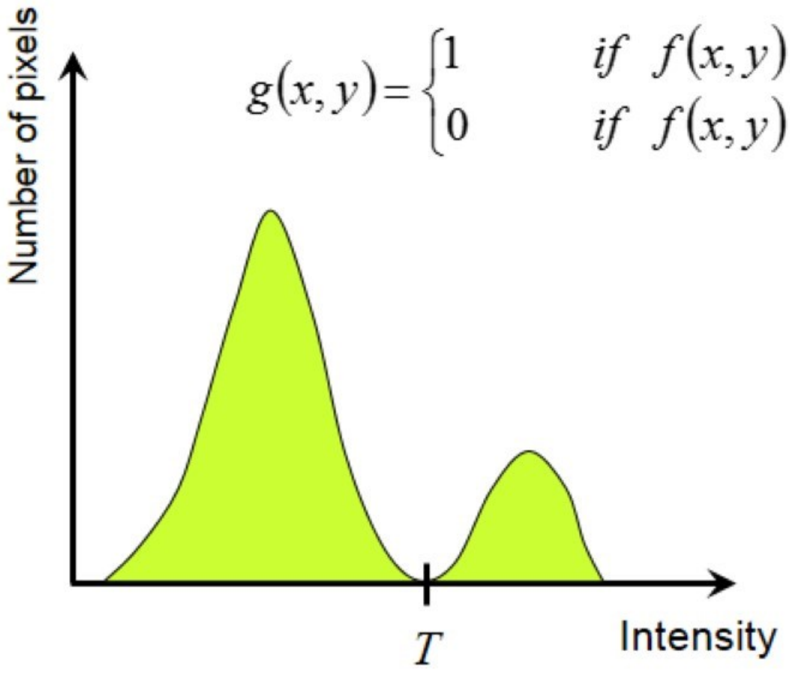
Line Detection Using the Hough Transform

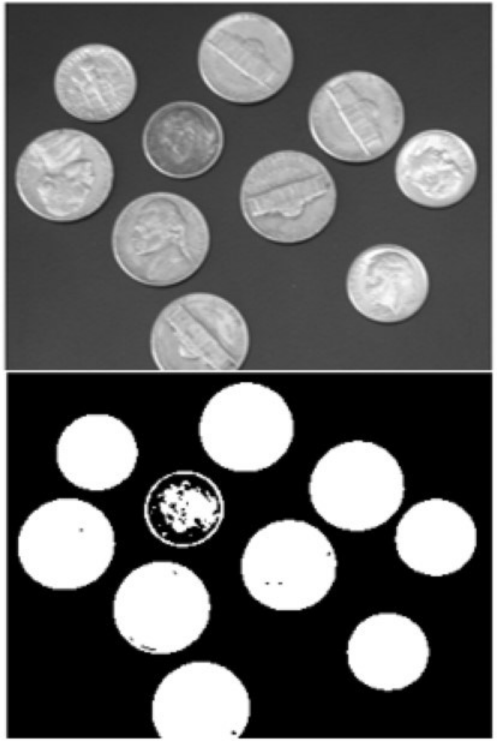
- Line segments corresponding to the Hough transform peaks

line_segment.m

Tresholding

$$g(x,y)=\begin{cases}1 & \text{if } f(x,y)\geq T\\ 0 & \text{if } f(x,y)<T\end{cases}$$



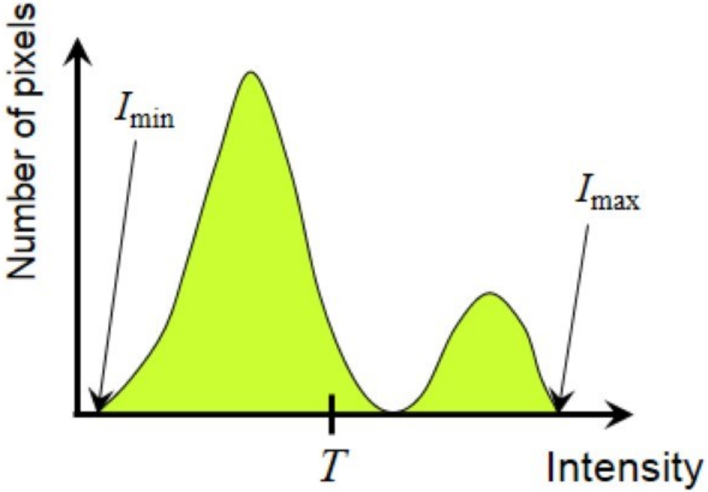


T is a specified threshold.

Tresholding

- **Global Thresholding**

  For choosing a threshold automatically, Gonzalez and Woods describe the following iterative procedure.



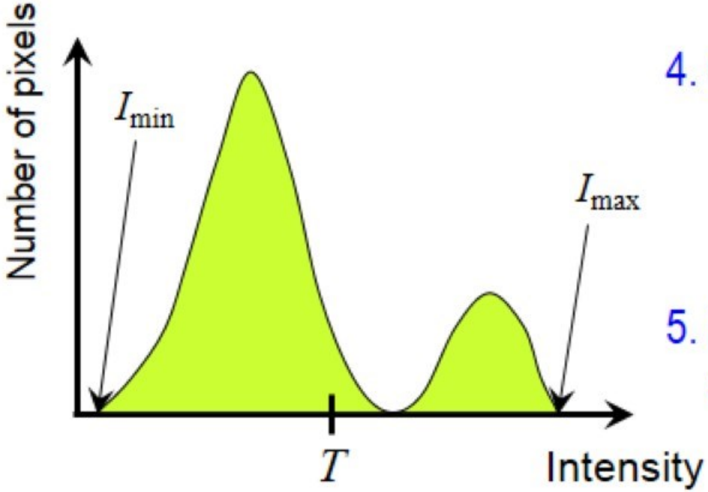
1. Select an initial estimate for T

$$T = \frac{I_{max} + I_{min}}{2}$$

2. Segment the image using T

Tresholding

- **Global Thresholding**



3. Compute the average intensity values μ_1 and μ_2 for the pixels in regions G_1 and G_2

4. Compute a new threshold T

$$T = \frac{(\mu_1 + \mu_2)}{2}$$

5. Repeat steps 2-4 until T is not change or less than a specified value.

Tresholding

- **Global Thresholding**

```
>> f =imread('rice.tif');
>> T =0.5*(double(min(f(:))) + double(max(f(:))));
>> done = false;

>> while ~done
>>       g =f >=T;
>>       Tnext = 0.5*(mean(f(g)) + mean(f(~g)));
>>       done = abs(T-Tnext)< 0.5;
>>       T = Tnext;
>> end
>> clc
>> T
>> To =graythresh(f)*255
```

Tresholding

- **Local Thresholding**

Local threshold

$$g(x,y)=\begin{cases}1 & if \;\; f(x,y)\geq T(x,y)\\ 0 & if \;\; f(x,y)<T(x,y)\end{cases}$$

where $T(x,y)=f_o(x,y)+T_o$

Morphological opening of f

Automatic threshold

Tresholding

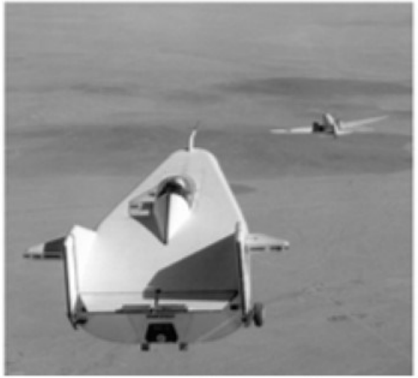
- **Local Thresholding: MATLAB code**

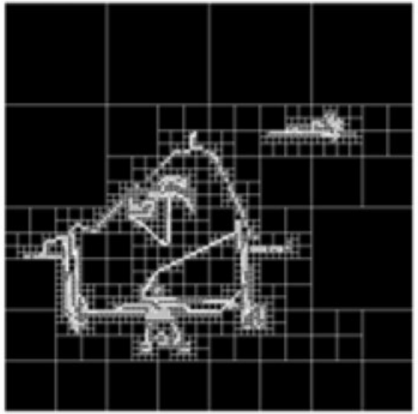
```
>> f =imread('rice.tif');
>> figure(1); imshow(f);
>> se =strel('disk', 10);
>> fo =imopen(f, se);
>> figure(2); imshow(fo);
>> To =graythresh(fo);
>> T =fo +(To*255);
>> figure(3); imshow(T);
>> [m, n]=size(f);
>> out = zeros(m, n);
>> out_idx = find(f >=T);
>> out(out_idx)) = 1;
>> figure(4); imshow(out);
```

Region-Based Segmentation

- **Basic Formulation**

Let R representation the entire image region. We may view segmentation as a process that partitions R into n subregions, $R_1, R_2, \ldots, R_n$, such that





Region-Based Segmentation

- **Basic Formulation**

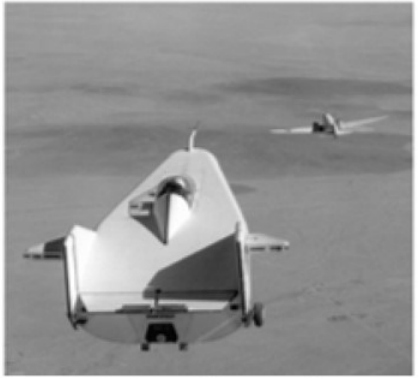
1 $\bigcup_{i=1}^{n} R_i = R$

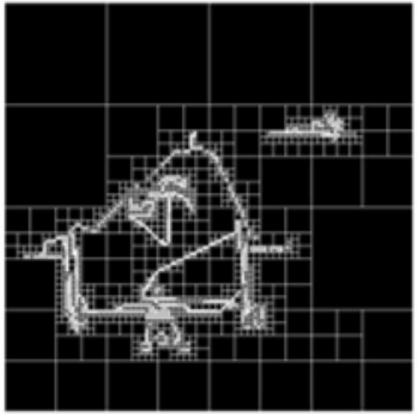
2 R_i is a connected region, $i = 1, 2, \ldots, n$

3 $R_i \cap R_j = \varnothing$ for all i and j, $i \neq j$

4 $P(R_i) = \text{TRUE}$ for $i = 1, 2, \ldots, n$

5 $P(R_i \cup R_j) = \text{FALSE}$ for any adjacent regions R_i and R_j.





$P(R_i)$ is a logical predicate.

Segmentation Using the Watershed Transform

Segmentation Using the Watershed Transform

- **Watershed Segmentation Using the Distance Transform**

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

0.0	0.0	1.0	2.0	3.0
0.0	0.0	1.0	2.0	3.0
1.0	1.0	1.4	2.0	2.2
1.4	1.0	1.0	1.0	1.4
1.0	0.0	0.0	0.0	1.0

It is the distance from every pixels to the nearest nonzero-valued pixel.

```
D =bwdist(x)
```

Segmentation Using the Watershed Transform

Example: Segmenting a binary image using the distance and Watershed Transforms.

```
>> watershed_dt.m % See demo
```

Segmentation Using the Watershed Transform

- **Watershed Segmentation Using Gradients**

The key concept of this method is that the gradient magnitude is used often to preprocess a gray-scale image prior to using the Watershed Transform for segmentation.

Segmentation Using the Watershed Transform

Example: Segmenting a gray-scale image using gradients and the Watershed Transform.

```
>> watershed_g.m % See demo
```

Segmentation Using the Watershed Transform

- **Marker Controlled Watershed Segmentation**

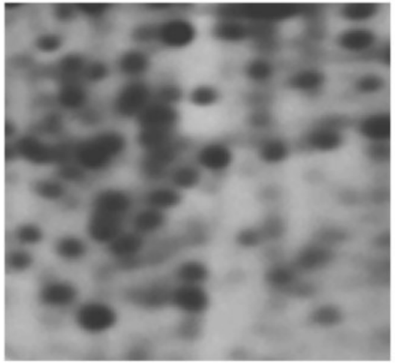
?

Direct application of the watershed transform to a gradient image usually leads to over-segmentation due to noise and other local irregularities of the gradient.

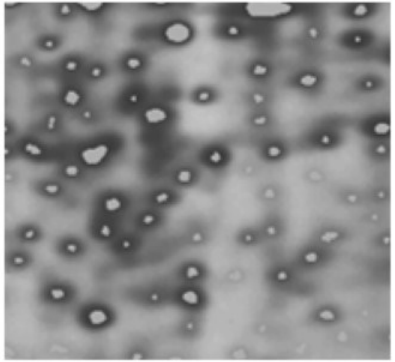
Segmentation Using the Watershed Transform

- **Marker Controlled Watershed Segmentation**

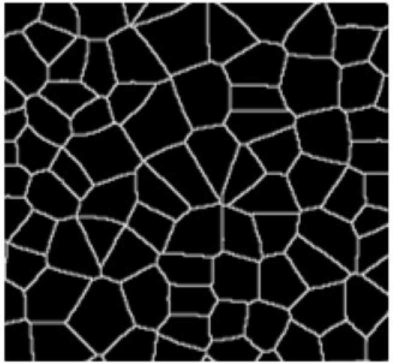
  An approach used to control over-segmentation is based on the concept of markers



Original image



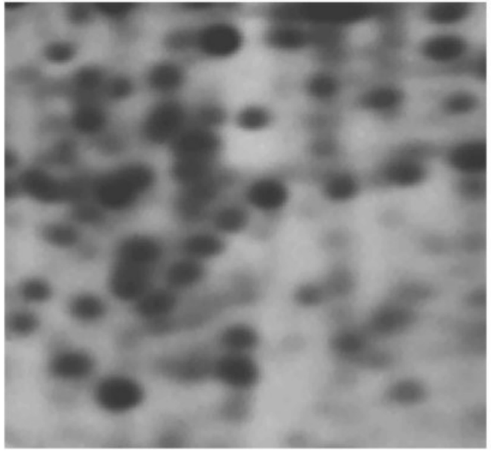
Internal marker image

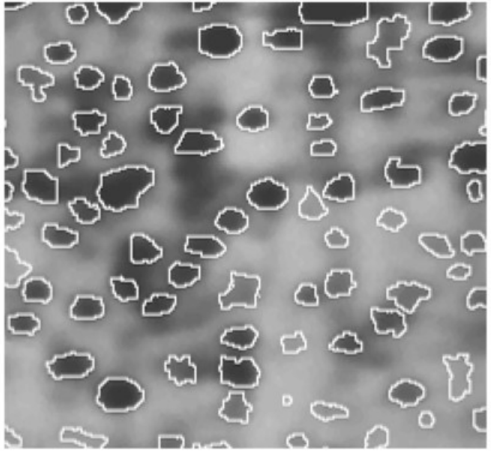


External marker image

Segmentation Using the Watershed Transform

- **Marker Controlled Watershed Segmentation**


Original image


Segmented image

last_ex.m

The end of
part II

ex_hough_5dots.m

```matlab
% ------------------------------------------
f = zeros(101, 101);
f(1, 1) = 1;
f(101, 1) = 1;
f(1, 101) = 1;
f(101, 101) = 1;
f(51, 51) = 1;

[H, theta, rho] = hough(f);
figure(1); imshow(H, []);
figure(2); imshow(H,'XData', theta, 'YData',rho);
axis on, axis normal;
xlabel('\theta'), ylabel('\rho');
```

```matlab
%% line_segment.m
%% ------------------------------------------------

f = imread('building.tif');

g_canny_best = edge(f, 'canny', [0.04 0.10], 1.5);
figure(1); imshow(g_canny_best)

[H, theta, rho] = hough(g_canny_best, 0.5);
figure(2);
imshow(H,'XData', theta, 'YData',rho); axis on, axis normal;
xlabel('\theta'), ylabel('\rho');

[r, c] = houghpeaks(H, 10);

hold on
plot(theta(c), rho(r), 'linestyle', 'none', 'marker', ...
    's', 'color', 'w');

lines = houghlines(g_canny_best, theta, rho, r, c);
figure(3); imshow(g_canny_best), hold on
for k = 1:length(lines)
    xy = [lines(k).point1 ; lines(k).point2];
    plot(xy(:, 2), xy(:, 1), 'LineWidth', 4, 'Color', ...
        [.6 .6 .6]);
end
```

```matlab
%% watershed_dt.m
%% ---------------------------------------------------

clear all
close all

f = imread('bwdowel.tif');
figure, imshow(f);

gc = ~f;
figure, imshow(gc);

D = bwdist(gc);
figure, imshow(mat2gray(D));

L = watershed(-D);
figure, imshow(~L);

w = (L==0);
g2 = f & ~w;
figure, imshow(g2);
```

```
%% watershed_g.m
%% ------------------------------------------------

clear all
close all

f = imread('blobs.tif');
figure(1), imshow(f);

h = fspecial('sobel');
fd = double(f);
g = sqrt(imfilter(fd, h, 'replicate').^2 + ...
         imfilter(fd, h', 'replicate').^2);
figure(2), imshow(mat2gray(g));

L = watershed(g);
wr = (L==0);
figure(3), imshow(wr);

g2 = imclose(imopen(g, ones(3,3)), ones(3,3));
L2 = watershed(g2);
wr2 = (L2==0);
figure(4), imshow(wr2);
```

```matlab
%% last_ex.m
%% -----------------------------------------------
clear all
close all

f = imread('gel.tif');
figure(1),imshow(f)

h = fspecial('sobel');
fd = double(f);
g = sqrt(imfilter(fd, h, 'replicate').^2 + ...
         imfilter(fd, h','replicate').^2);

g1 = log2(1+double(g));
figure, imshow(mat2gray(g1));

L = watershed(g);
wr = L==0;
figure, imshow(wr);

rm = imregionalmin(g);
figure, imshow(rm)

im = imextendedmin(f, 2);
fim = f;
fim(im) = 175;
figure, imshow(fim)

Lim = watershed(bwdist(im));
em = Lim == 0;
figure, imshow(em);

%g2=imimposemin(f,mark)
gg = mat2gray(g1);
g2 = imimposemin(gg, em | im);
% g22 = log(1+double(g2));
% figure, imshow(mat2gray(g22))
figure, imshow(g2)

L2 = watershed(g2);
f2 = f;
f2(L2 == 0) = 255;
figure, imshow(f2)
```