目录

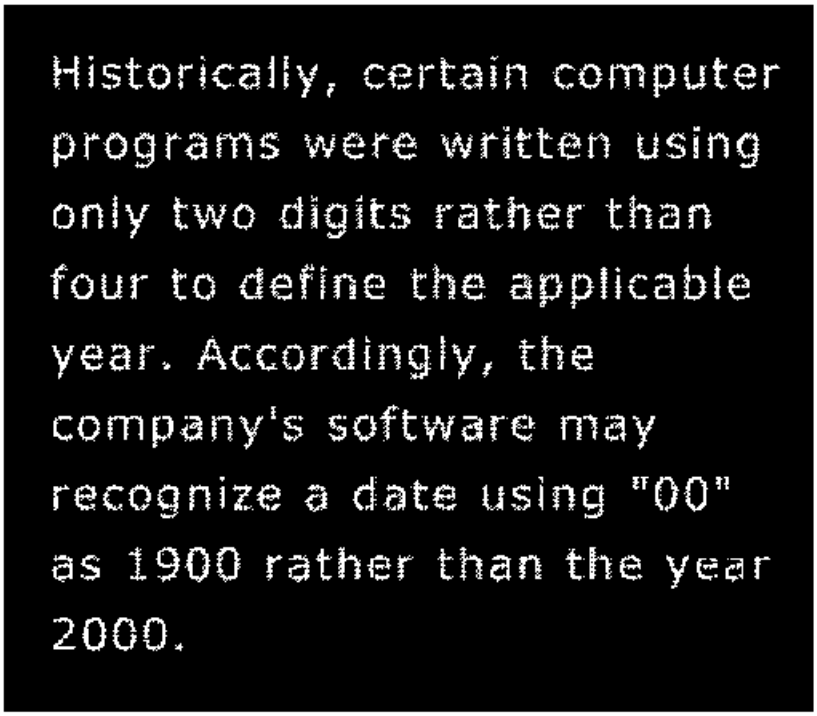
形态学图像处理

9.2 膨胀和腐蚀

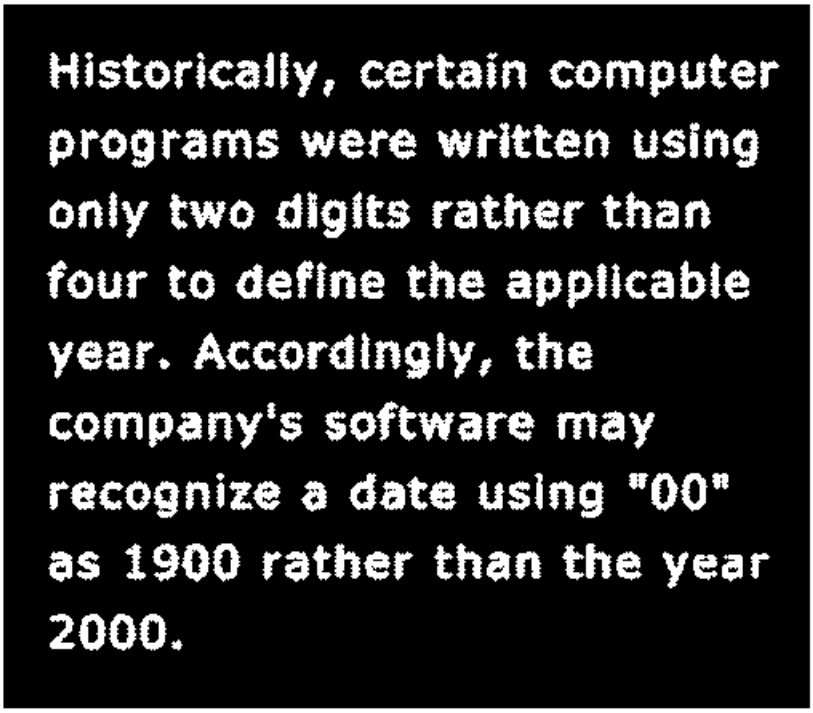
9.2.1 膨胀

例 9.1 膨胀应用

```
A = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0906(a)(broken-text).tif');
figure;
imshow(A, []);
```




```
B = [0 1 0; 1 1 1; 0 1 0];
A2 = imdilate(A, B); % 用 B 膨胀 A
figure;
imshow(A2, []); % 膨胀后 e a 被桥接
```



9.2.2 结构元素分解

结构元素如果能被分解为两个子结构的膨胀结果，[这里要写到笔记中]

9.2.3 strel 创建 SE

例 9.2 strel 分解结构元素

```
se = strel('diamond', 5);
se.Neighborhood
```

```
ans = 11×11 logical ##
   0   0   0   0   0   1   0   0   0   0   0
   0   0   0   0   1   1   1   0   0   0   0
   0   0   0   1   1   1   1   1   0   0   0
   0   0   1   1   1   1   1   1   1   0   0
   0   1   1   1   1   1   1   1   1   1   0
   1   1   1   1   1   1   1   1   1   1   1
   0   1   1   1   1   1   1   1   1   1   0
   0   0   1   1   1   1   1   1   1   0   0
   0   0   0   1   1   1   1   1   0   0   0
   0   0   0   0   1   1   1   0   0   0   0
     ⋮
```

```
decomp = getsequence(se); % 提取并检查分解中的单个结构元素
```

```
whos % 输出之前变量的信息
```

```
  Name          Size              Bytes  Class      Attributes

  A           444x509            225996  logical
  A2          444x509            225996  logical
  B             3x3                 72  double
  ans          11x11                121  logical
  decomp        4x1                576  strel
  se            1x1               2264  strel
```

```
% decomp size 为 4 * 1, 说明分解为 4 个结构了
decomp(1).Neighborhood
```

```
ans = 3×3 logical ##
   0   1   0
   1   1   1
   0   1   0
```

```
decomp(2).Neighborhood
```

```
ans = 3×3 logical ##
   0   1   0
   1   0   1
   0   1   0
```

```
decomp(3).Neighborhood
```

```
ans = 5×5 logical ##
   0   0   1   0   0
   0   0   0   0   0
   1   0   0   0   1
   0   0   0   0   0
   0   0   1   0   0
```

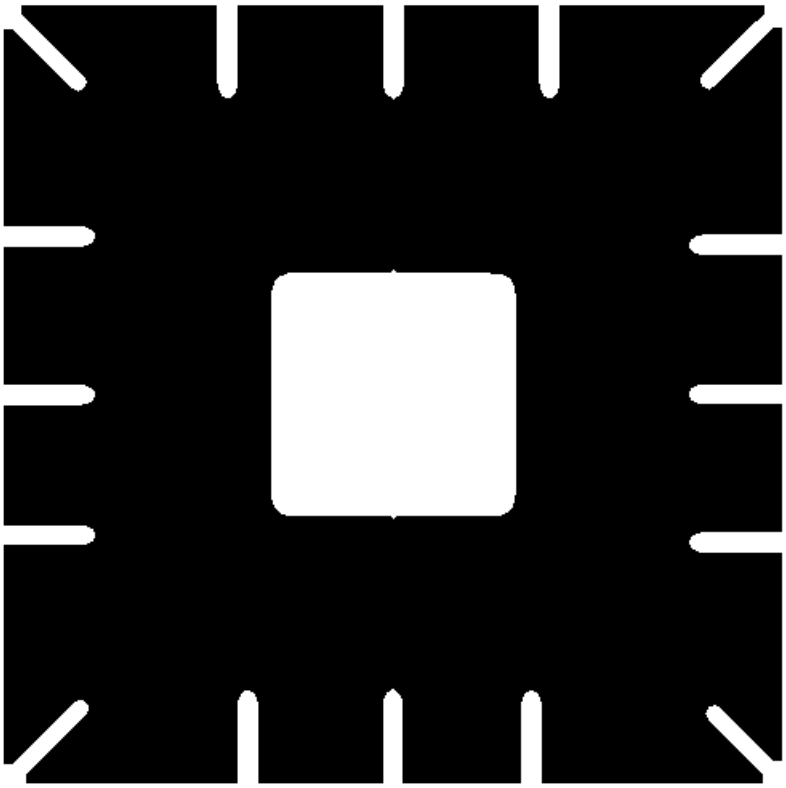
```
decomp(4).Neighborhood
```

```
ans = 3×3 logical ##
   0   1   0
   1   0   1
   0   1   0
```

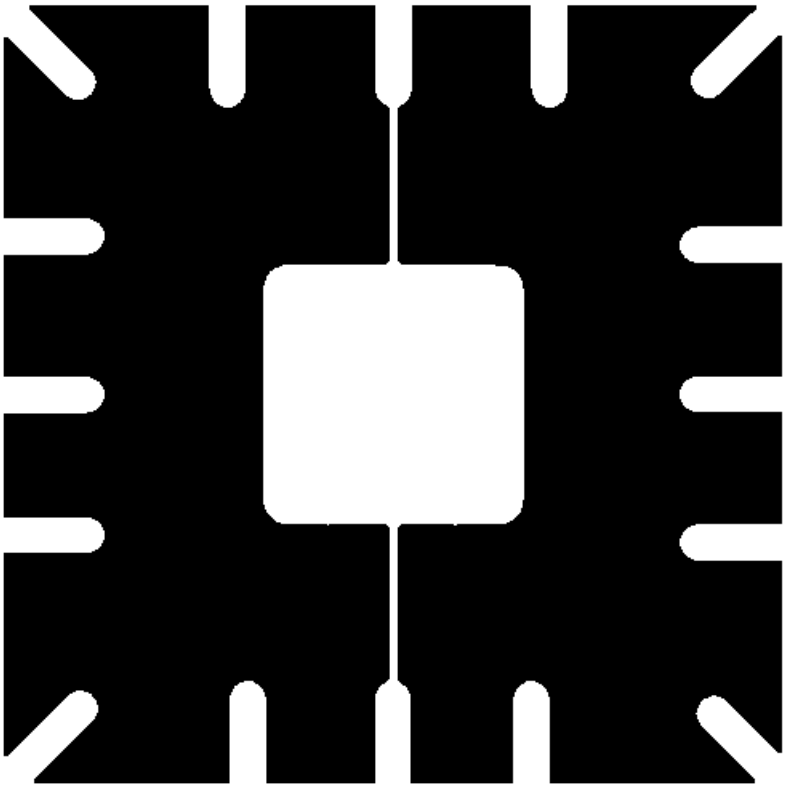
9.2.4 腐蚀

例 9.3 腐蚀的说明

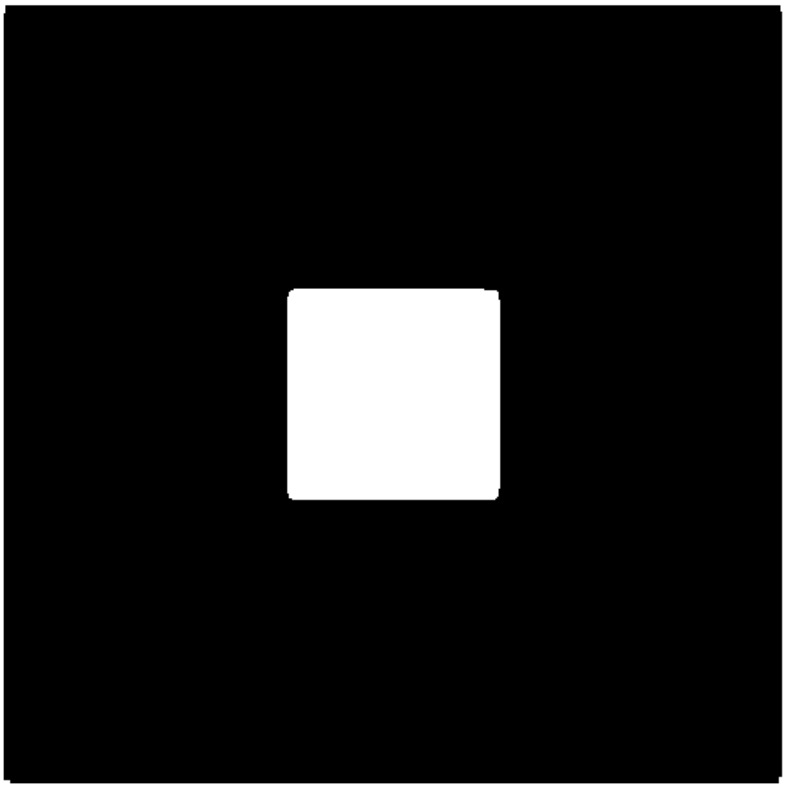
```
A = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0908(a)(wirebond-mask).tif');
SE = strel('disk', 10);
A2 = imerode(A, SE);
figure;
imshow(A2, []); % 这里用 半径为 10 圆盘腐蚀
```



```
SE = strel('disk', 5);
A2 = imerode(A, SE);
figure;
imshow(A2, []); % 这里用 半径为 5 圆盘腐蚀，粗的线没被腐蚀掉
```



```
SE = strel('disk', 20);
A2 = imerode(A, SE);
figure;
imshow(A2, []); % 这里用 半径为 20 圆盘腐蚀 只有中心方块保留
```

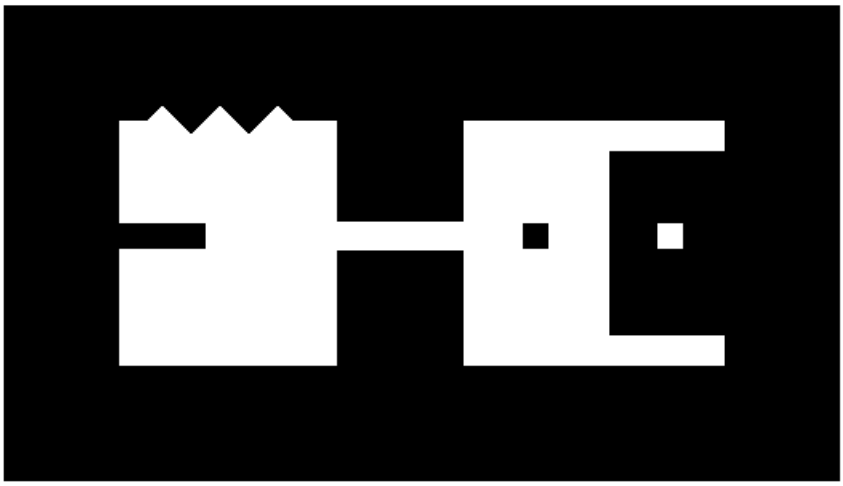


9.3 膨胀与腐蚀结合

9.3.1 开运算闭运算

例 9.4 imopen imclose

```
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0910(a)(shapes).tif');
figure;
imshow(f, []);
```



```
se = strel('square', 20);
se.Neighborhood
```

```
ans = 20×20 logical ##
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
     1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
       ⋮
```

```
fo = imopen(f, se); % 开运算
figure;
imshow(fo); % 回想一下几何解释，在图像内部平移，可以消除凸起，连接处也被断开
```

```
fc = imclose(f, se);
figure;
imshow(fc, []); % 回想一下几何解释，在图像外围平移，是不是会添补向内的空隙，保留向外凸起
```

```
foc = imclose(fo, se); % 开运算后再闭运算，可以平滑目标图像
figure;
imshow(foc, []);
```

```
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0911(a)(noisy-fingerprint).tif');
figure;
imshow(f, []); % 这是含噪的指纹图
```

```
se = strel('square', 3);
fo = imopen(f, se);
figure;
imshow(fo, []);% 利用开运算去除一些毛刺噪声，但是造成了一些内部的断裂(想一想几何解释，结构元再图像内部
```

```
foc = imclose(fo, se); % 再用闭运算填充缺口
figure;
imshow(foc, []);
```

9.3.2 击中 - 不击中变换

例 9.5 bwhitmiss

此处定义式在第三版，第四版有更新一种更好理解的。如果关于此实验例子不了解，翻看图 9.12

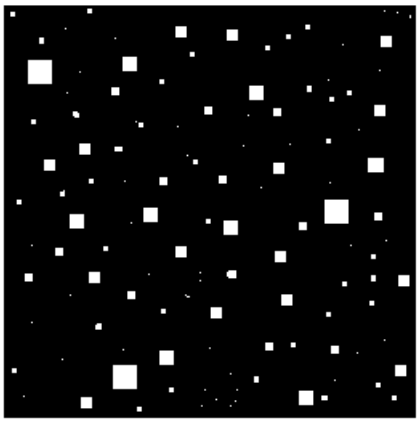
```
B1 = strel([0 0 0; 0 1 1; 0 1 0]);
B2 = strel([1 1 1; 1 0 0; 1 0 0]);
B1.Neighborhood % 要右边，下边领域像素，注意 B1 对应的 1 的位置
```

```
ans = 3×3 logical ##
   0   0   0
   0   1   1
   0   1   0
```

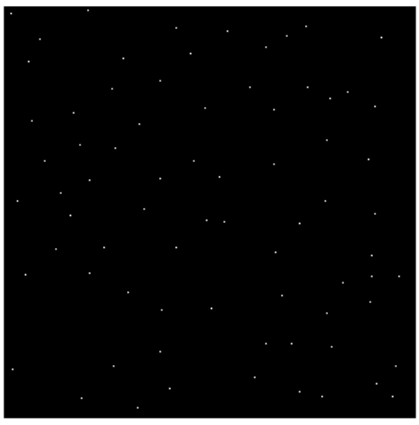
```
B2.Neighborhood % 不要左上|右上|左下|上，注意 B2 对应的 1 位置
```

```
ans = 3×3 logical ##
   1   1   1
   1   0   0
   1   0   0
```

```
% B1 关心击中，B2 关心不击中，B1 B2 右下角都为 0，说明这是不关心点。
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0913(a)(small-squares).tif');
figure;
imshow(f, []);
```



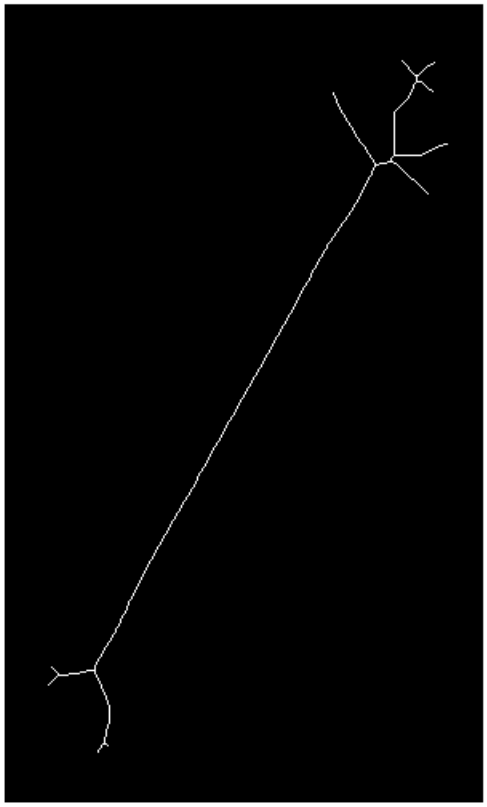
```
g = bwhitmiss(f, B1, B2);
figure;
imshow(g, []); % 可以对比一下击中 和 不击中分别在哪里，从而达到缩小目标图像的效果
```



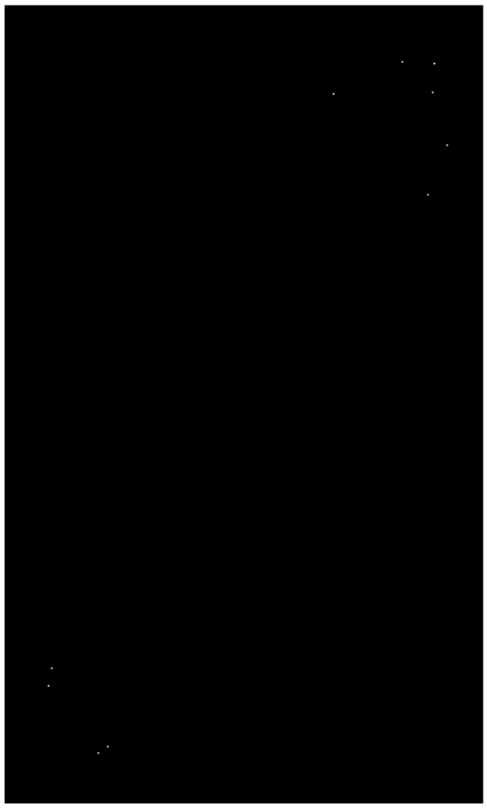
9.3.3 使用查找表

注意他前头说的例子，如何给一个形状分配一个唯一索引。

```
% 接下来查找端点
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0914(a)(bone-skel).tif');
figure, imshow(f, []);
```

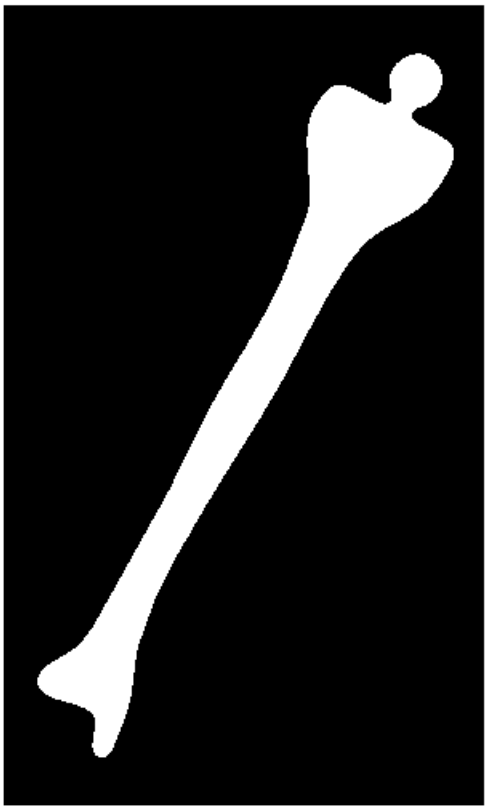


```
f_e = endpoints(f);
figure, imshow(f_e, []);
```

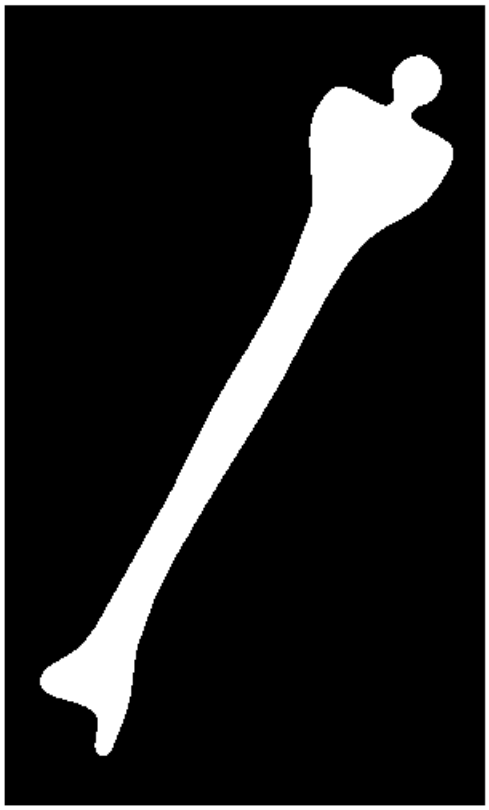


9.3.4 bwmorph 基于膨胀腐蚀查找表组合实现操作

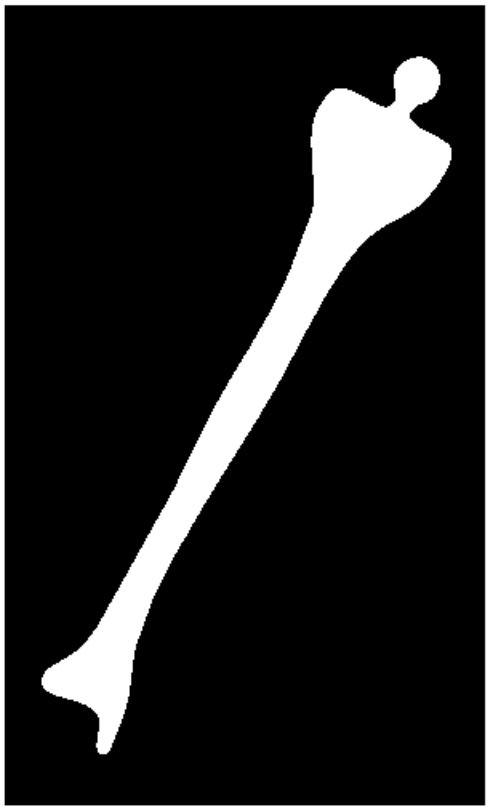
```
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0916(a)(bone).tif');
figure;
imshow(f, []);
```



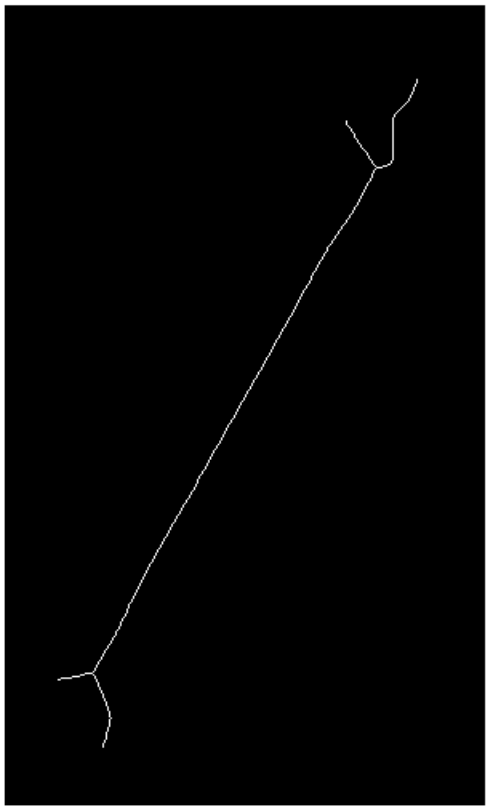
```
g1 = bwmorph(f, 'thin', 1); % 细化操作 1 次
g2 = bwmorph(f, 'thin', 2);
g_inf = bwmorph(f, 'thin', inf); % 无数次，直至不能再细化
figure, imshow(g1, []);
```



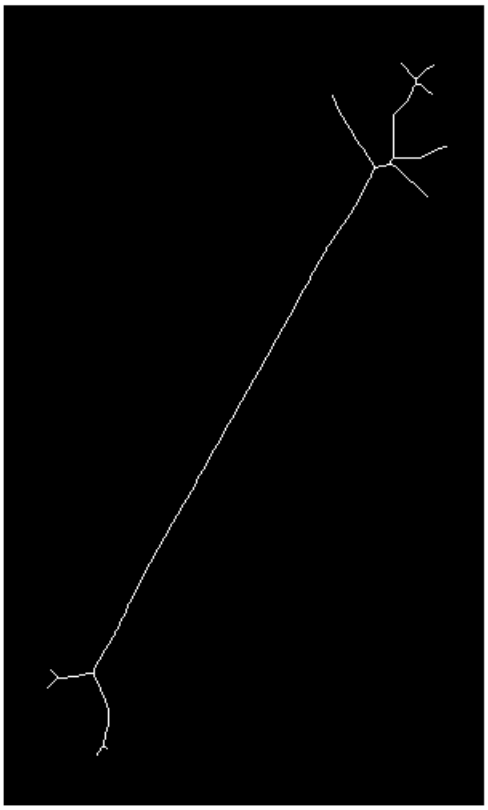
```
figure, imshow(g2, []);
```



```
figure, imshow(g_inf, []); % inf 细化会造成一些损失，不能喝骨骼化等同
```



```
fs = bwmorph(f, 'skel', inf); % 直接进行骨骼化
figure, imshow(fs, []);
```

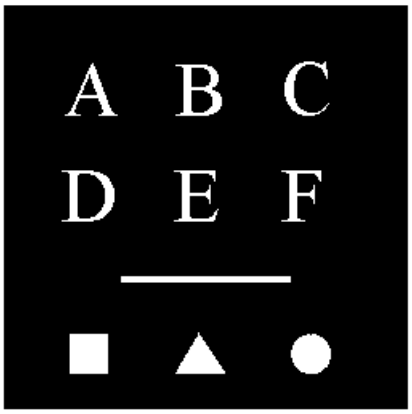


9.4 标注连接分量

```
% 4\8 邻接，4\8 连接(针对一条路径 详见 图 9.18
% 连接分量(或者说一个对象)，根据邻接方式的不同而变化 图 9.19
```

例 9.7 计算显示连接分量的质心

```
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0917(a)(ten-objects).tif');
figure, imshow(f);
```



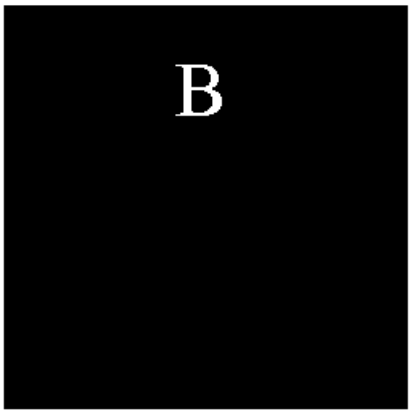

```
[L, n] = bwlabel(f); % 计算连通分量的个数，默认 8 连接分量，L 为标记矩阵
% figure;
L % 会标记好几个分量，从 1 开始标记
```

```
L = 252×252
     0     0     0     0     0     0     0     0     0     0     0     0     0···
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     ⋮
```

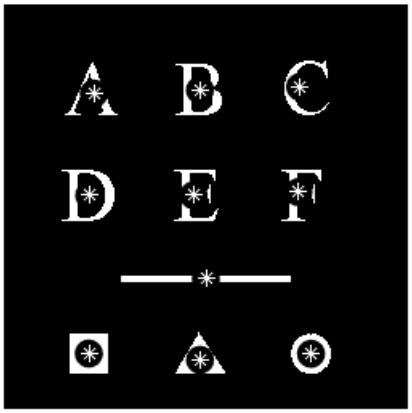
```
n % 分量数
```

```
n = 10
```

```
% 通过这个例子来说明如何找连通分量
[r, c] = find(L == 6); % 标记为 6 的分量
[M, N] = size(f);
t_f    = zeros(M, N);
t_f(r, c) = f(r, c);
figure, imshow(t_f, []);
```




```
% 接着标记质心
figure;
imshow(f);
hold on; %让后头的 plot 都作用在前一张图
for k = 1 : n
    [r, c] = find(L == k);
    rbar = mean(r);
    cbar = mean(c);
    % 想想 plot 函数时是 x, y, 而这里为什么要反过来? (坐标系差异
    plot(cbar, rbar, 'Marker', 'o', 'MarkerEdgeColor', 'k', 'MarkerFaceColor','k', 'MarkerSize
    plot(cbar, rbar, 'Marker', '*', 'MarkerEdgeColor', 'w');
end
hold off;
```



9.5 形态学重构

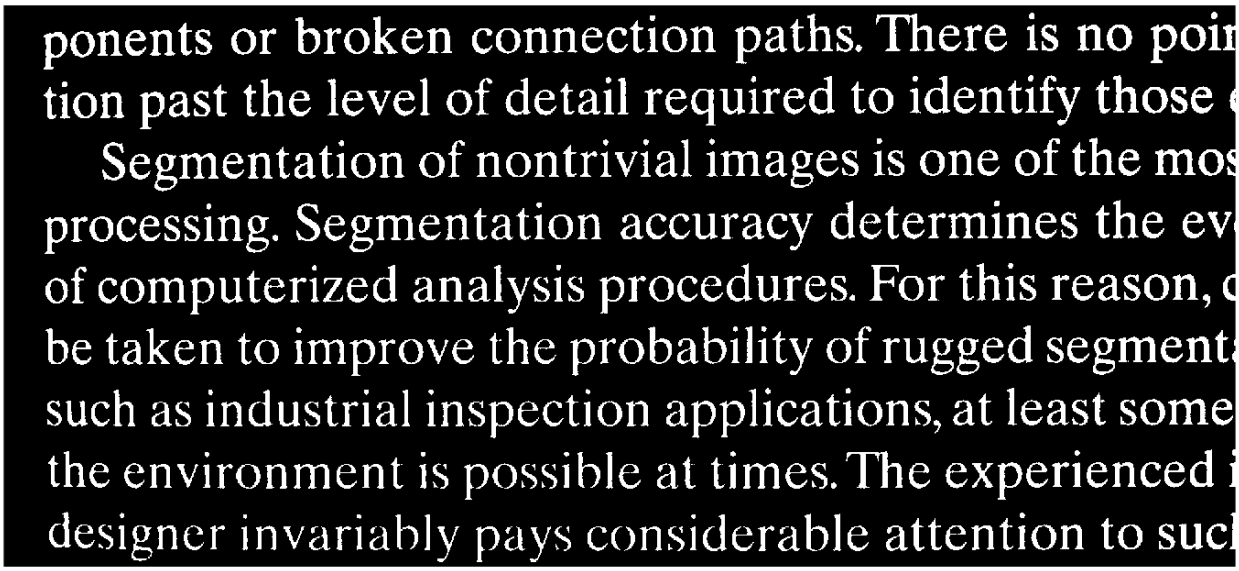
f 为标记，g 为模板, 不断迭代如下公式

$h_{k+1} = (h_k \oplus f) \cap g)$，直至 $h_{k+1} = h_k$

```
% imreconstruct(marker, mask)
```

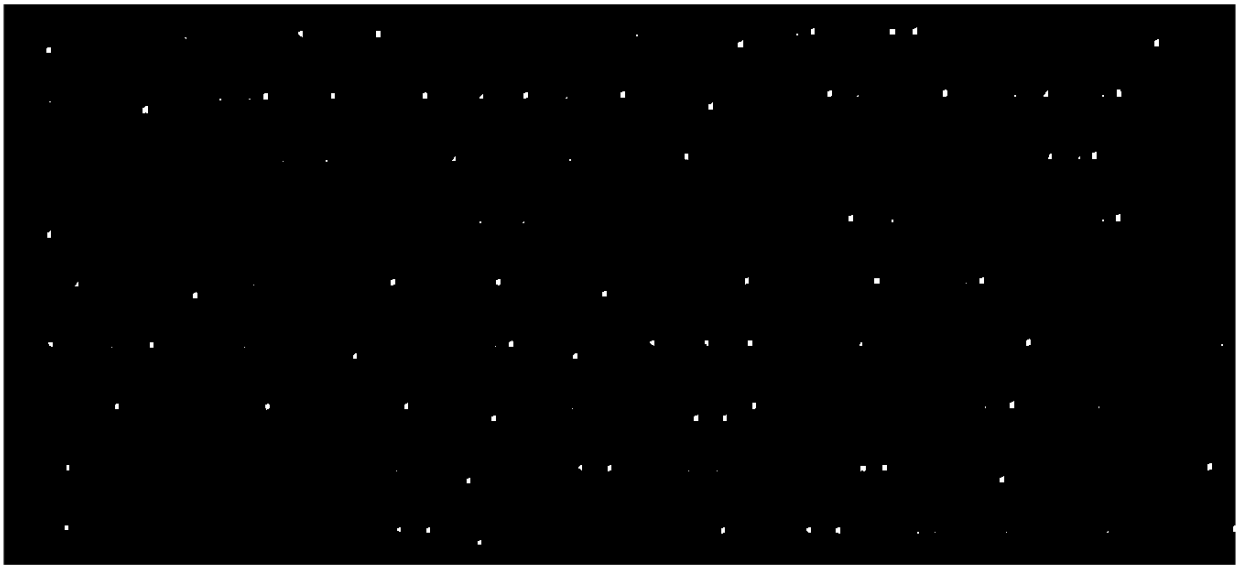
9.5.1 由重构做开运算

```
% 例 9.8 由重构做开运算
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0922(a)(book-text).tif');
figure;
imshow(f, []);
```



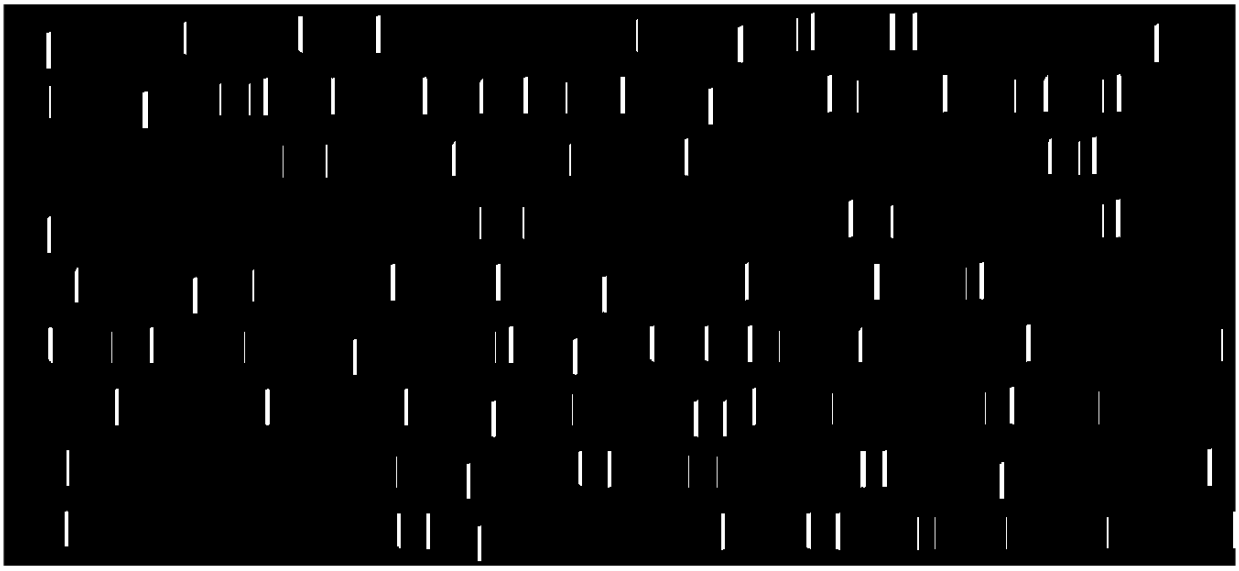
警告: 图像太大，无法在屏幕上显示；将以 67% 显示

```
fe = imerode(f, ones(51, 1)); % 用竖线腐蚀
% fe_hori = imerode(f, ones(1, 21));
figure, imshow(fe, []);
```



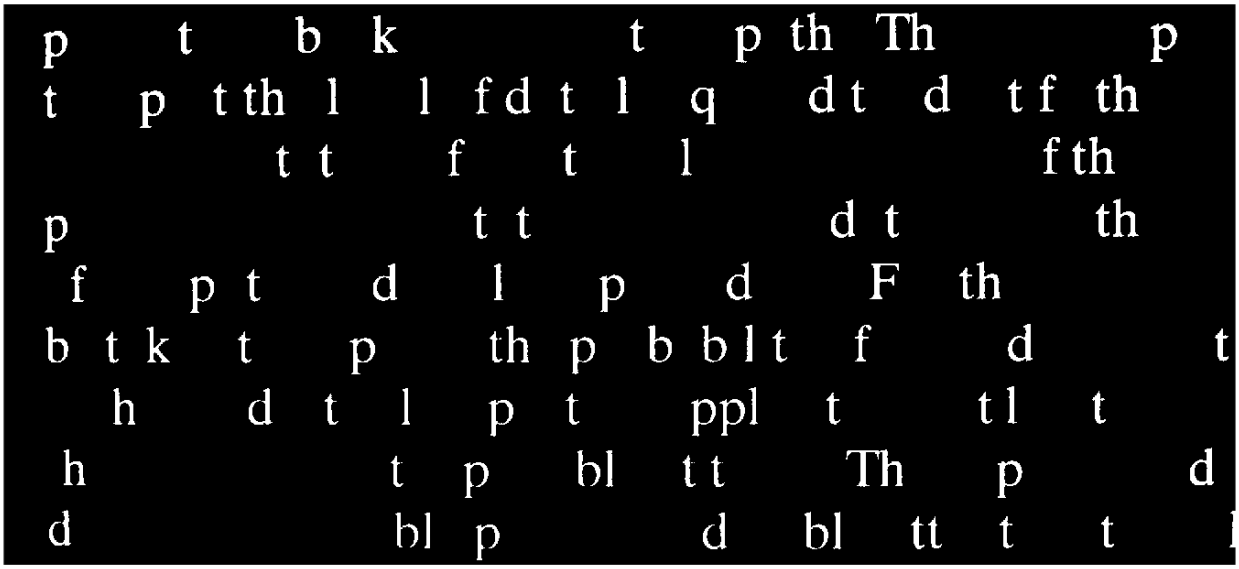
警告: 图像太大，无法在屏幕上显示；将以 67% 显示

```
fo = imopen(f, ones(51, 1)); % 开运算，在图像内部滚，保留了竖线
figure, imshow(fo);
```



警告: 图像太大，无法在屏幕上显示；将以 67% 显示

```
fobr = imreconstruct(fe, f); % 被腐蚀过的标记图像，模板图像为原图
figure, imshow(fobr, []) % 含有长条的字母被复原
```



警告: 图像太大，无法在屏幕上显示；将以 67% 显示

```
% figure, imshow(imreconstruct(fe_hori, f)); 可选
```

9.5.2 填充孔洞

利用 imfill 可以实现，注意回想填充孔洞的算法

```
% imfill
%
```

9.5.3 清除边界对象

使用 imclearborder(f, conn), conn 为 4 或 8 邻接

与图像边界相连的对象可以清除

```

```

9.6 灰度图像形态学

9.6.1 腐蚀和膨胀

其实还是可以用 imerode, 可以利用 strel 创建平坦或不平坦结构元

平坦元的灰度腐蚀是局部最小算子，所以会偏暗。灰度膨胀会偏亮

```
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0923(a)(aerial).tif');
figure;

subplot(2, 2, 1);
imshow(f);
```

```
% b = strel([1 1 1], [1 2 1]); % 不平坦结构元，灰度值作为 xy 平面高度
% f_b = imerode(f, b);
% subplot(2, 2, 2);
% imshow(f_b);
se = strel('square', 3);
gd = imdilate(f, se); % 平坦结构元膨胀图像
subplot(2, 2, 2);
imshow(gd);

ge = imerode(f, se);
subplot(2, 2, 3);
imshow(gd);

% 形态学梯度 - 膨胀结果和腐蚀结果的差值
morph_grad = imsubtract(gd, ge); % 图像间差值
subplot(2, 2, 4);
imshow(morph_grad); % 明显边缘部分更明显，因为梯度就用于凸显边缘的变化程度
```




9.6.2 开闭运算

在灰度开闭运算中，同前头说的，xy 平面上的一个高度值，是像素值，我们用一个剖面来讨论开运算，结构元在剖面底下，去除比结构元小的明亮细节；闭运算，结构元在剖面上。

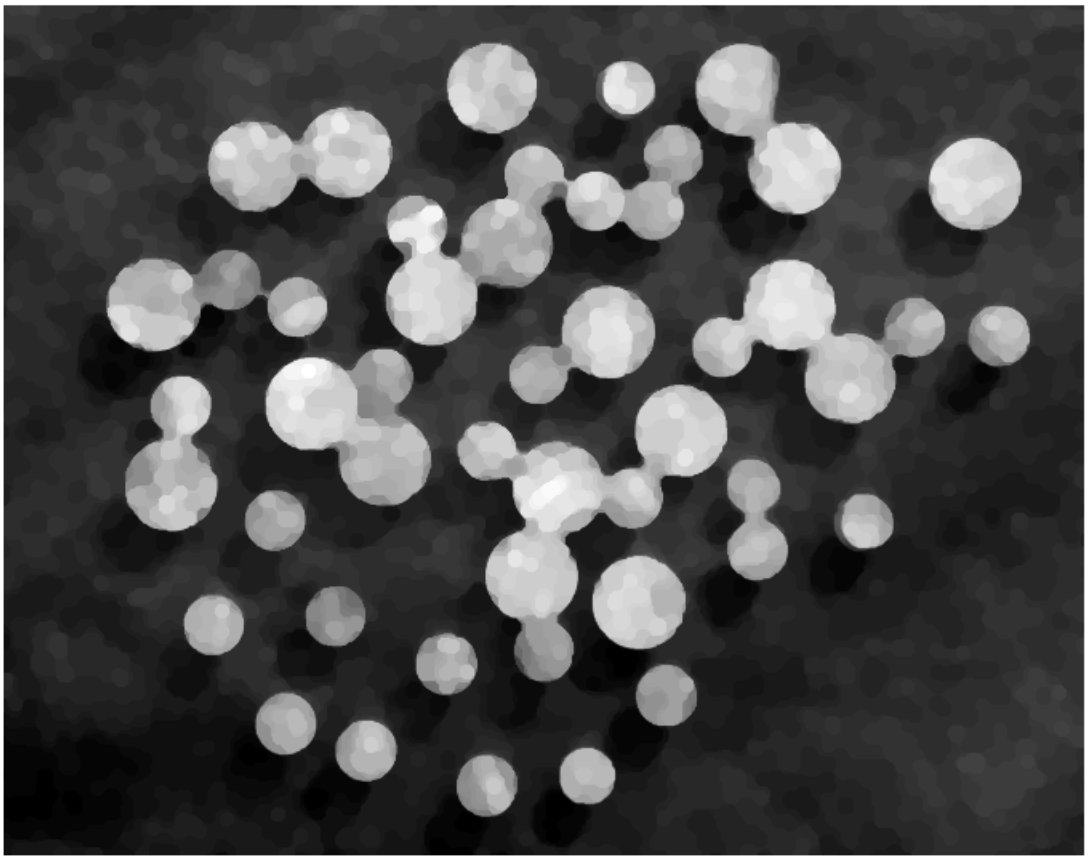
详见 图 9.24

例 9.9 开闭做形态学平滑

```
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0925(a)(dowels).tif');
figure, imshow(f, []);
```

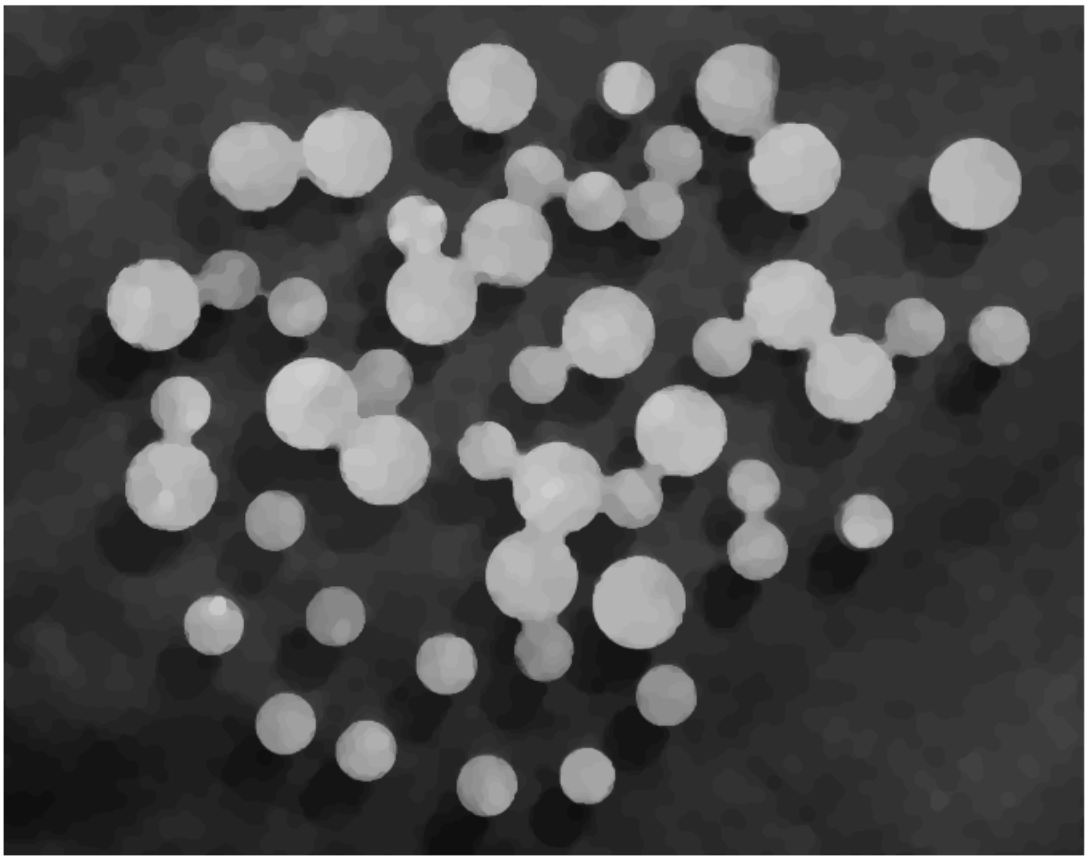
```
se = strel('disk', 5);
fo = imopen(f, se);
figure, imshow(fo, [])
```

```
foc = imclose(fo, se);
figure, imshow(foc, []) % 从灰度剖面角度想，就知道为什么图中的圆产生了黏连
```



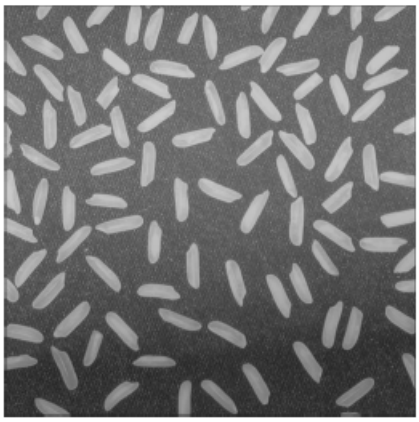
```
% 还有一种交替滤波方式，结构元不断增大
fasf = f;
for k = 2 : 5
    se = strel('disk', k);
    fasf = imclose(imopen(fasf, se), se); % 先开后闭
end

figure;
imshow(fasf) % 将过亮过暗的平滑
```



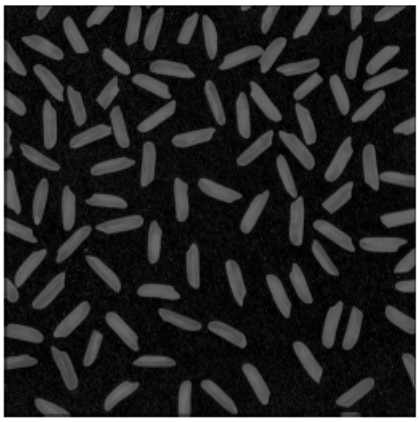
例 9.10 顶帽变换

```
% 顶帽中 保留被去除的亮细节
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0926(a)(rice).tif');
figure, imshow(f);
```



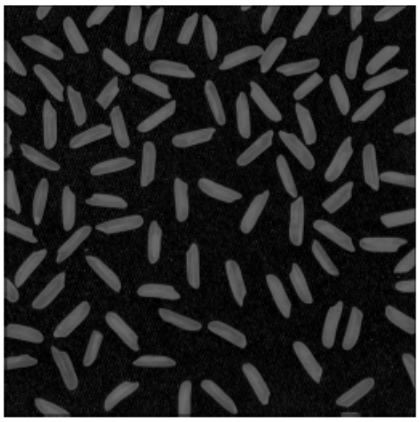
```
se = strel('disk', 10);
fo = imopen(f, se); % 先开运算
figure, imshow(fo);
```

```
f2 = imsubtract(f, fo); % 顶帽运算
figure, imshow(f2); % 明显 米粒被提取出来了
```



当然你用 imtophat 也可以

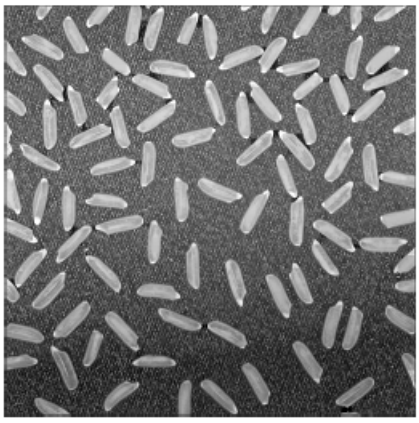
```
f2 = imtophat(f, se);
figure, imshow(f2)
```



用底帽和顶帽增强对比度

底帽针对亮背景的暗目标

```
se = strel('disk', 3);
g = imsubtract(imadd(f, imtophat(f, se)), imbothat(f, se));
figure, imshow(g)
```



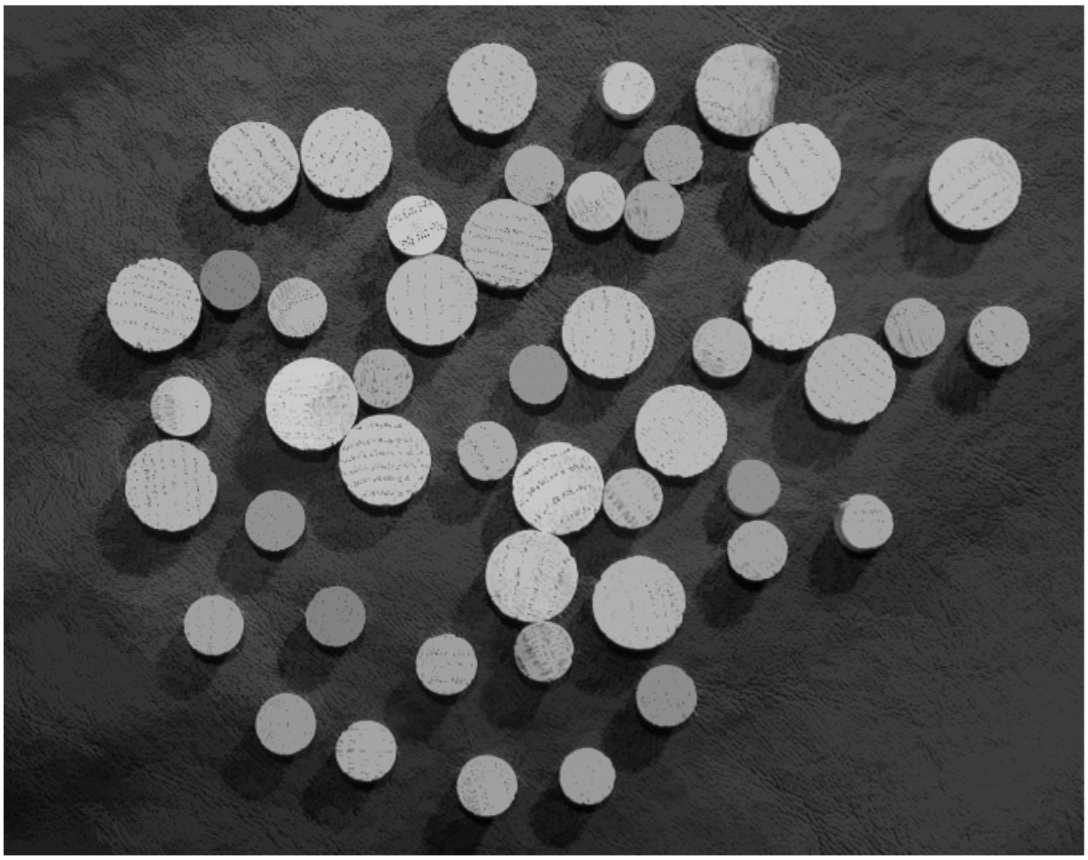
例 9.11 颗粒分析

跳过

9.6.3 重构

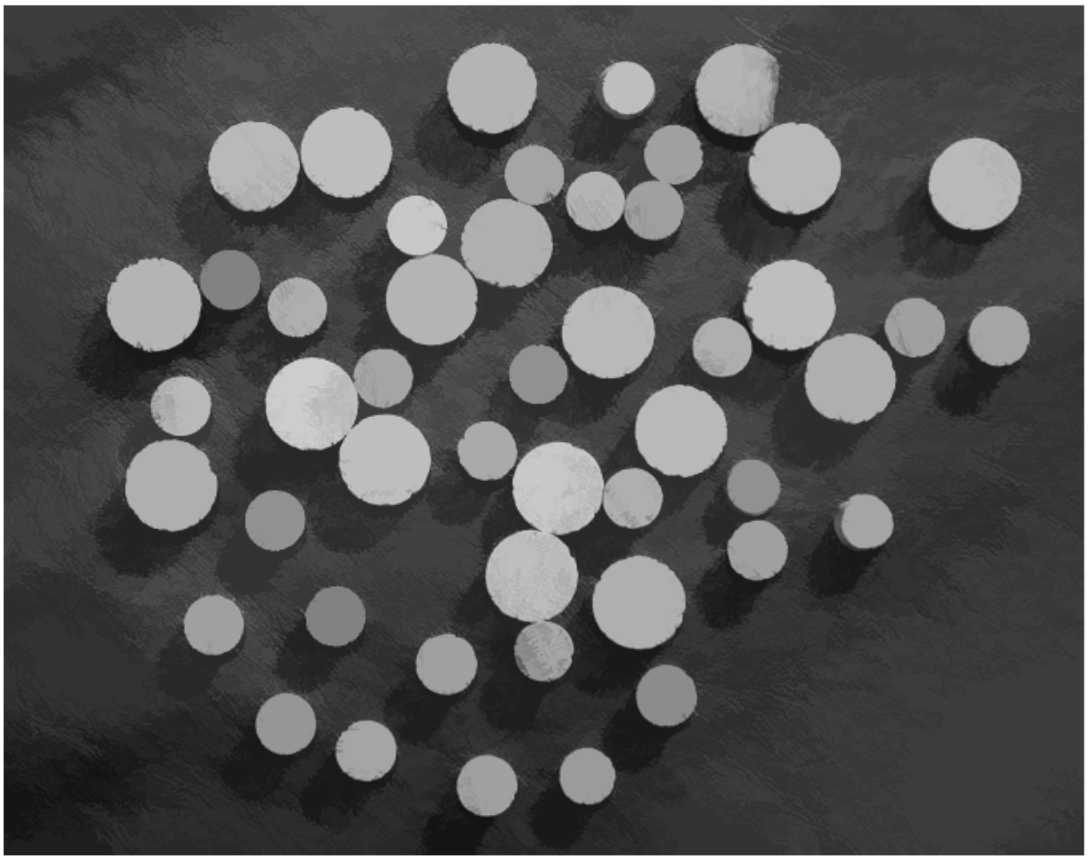
开运算重构，腐蚀的图像做标记图像，原图像做模板

```
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0925(a)(dowels).tif');
se = strel('disk', 5);
fe = imerode(f, se);
fobr = imreconstruct(fe, f);
figure, imshow(fobr);
```



重构闭运算

```
fobrc = imcomplement(fobr); % 对 fobr 求补
fobrce = imerode(fobrc, se);
fobrcbr = imcomplement(imreconstruct(fobrce, fobrc)); % 对重构开运算求补。重构开运算的模板是原图像
figure, imshow(fobrcbr);
```



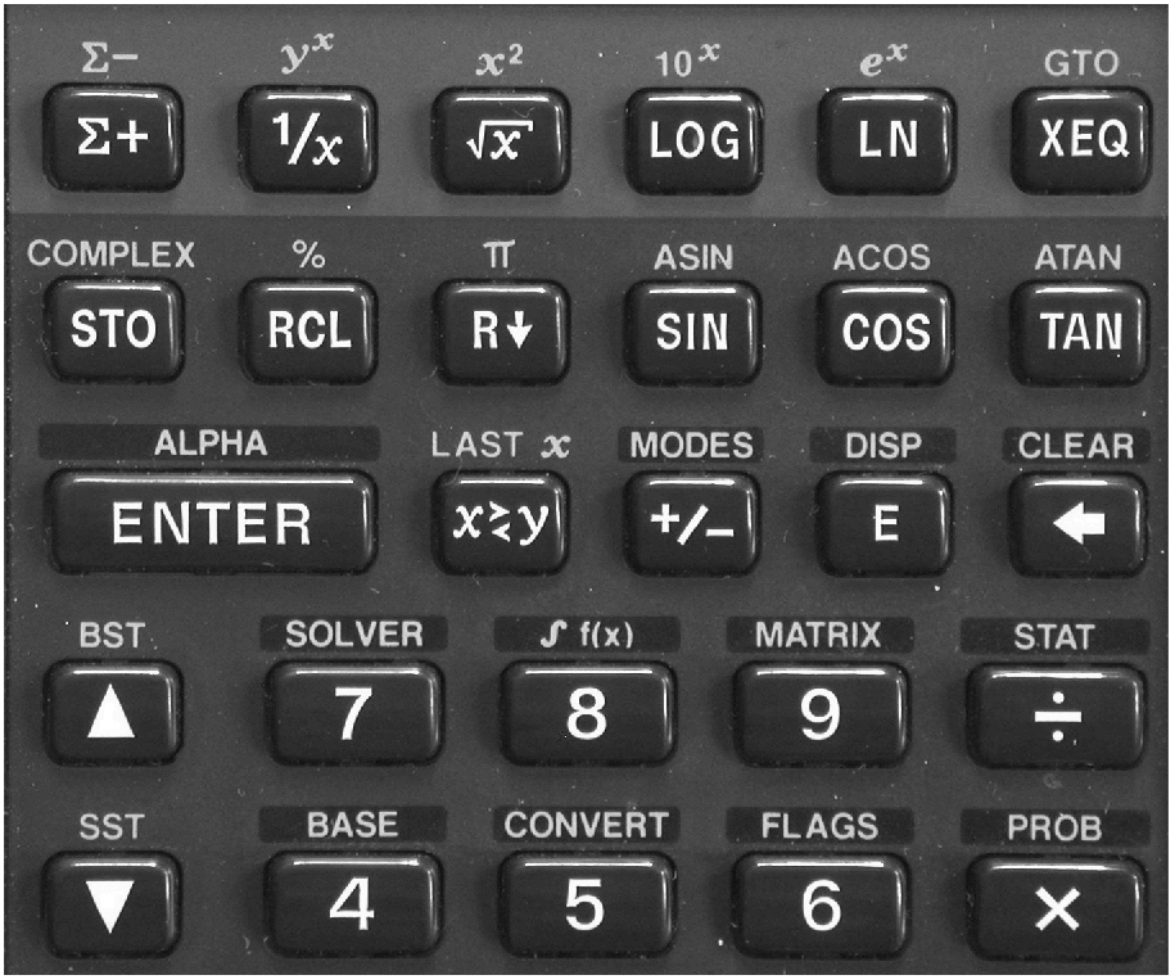
例 9.12 使用重构删除 复杂图像背景

~~例子里头的重建运算已经由 imreconstruct 实现，但是基于此的一些技巧，在本例中展现出来~~

关于重构开运算，前头已经提供了。

如：什么用作标记图像，什么用作模板图像。

```
f = imread('./DIP-Ex/pic/dipum_images_ch09/Fig0930(a)(calculator).tif');
figure, imshow(f);
```

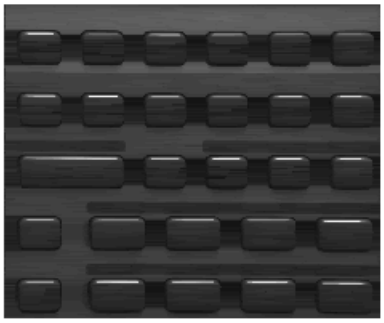


警告: 图像太大，无法在屏幕上显示；将以 67% 显示

横向反射光比按键上的字都宽，用长条结构元做开运算

会把按键都腐蚀，但亮光会被重新膨胀回来从而保留

```
f_obr = imreconstruct(imerode(f, ones(1, 71)), f); % 对 f 进行大小为 1 的重建开运算
f_o   = imopen(f, ones(1, 71));
figure;
subplot(1, 2, 1), imshow(f_obr);
subplot(1, 2, 2), imshow(f_o); % 开运算重构比开运算效果更好一些
```



顶帽重构

```
f_thr = imsubtract(f, f_obr);
figure, imshow(f_thr);
```

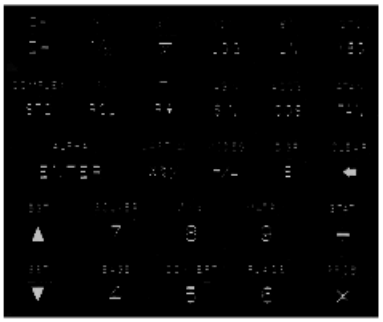
警告：图像太大，无法在屏幕上显示；将以 67% 显示

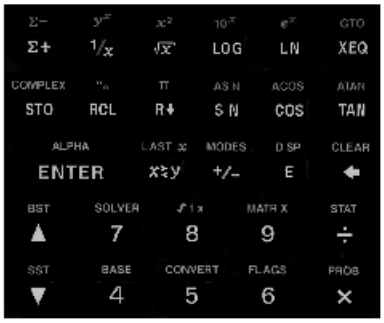
明显，垂直光还是存在的。

利用短水平线结构元再进行开运算重构

我对比了腐蚀和开运算重建，因为我把两者概念混了，只考虑了腐蚀水平反射光，忘记了腐蚀会破坏更多。

```
figure;
g_obr = imreconstruct(imerode(f_thr, ones(1, 11)), f_thr); % 把腐蚀过的图像作为标记，f_thr 做模板
subplot(1, 2, 1);
imshow(imerode(f_thr, ones(1, 11)));
subplot(1, 2, 2);
imshow(g_obr);
```



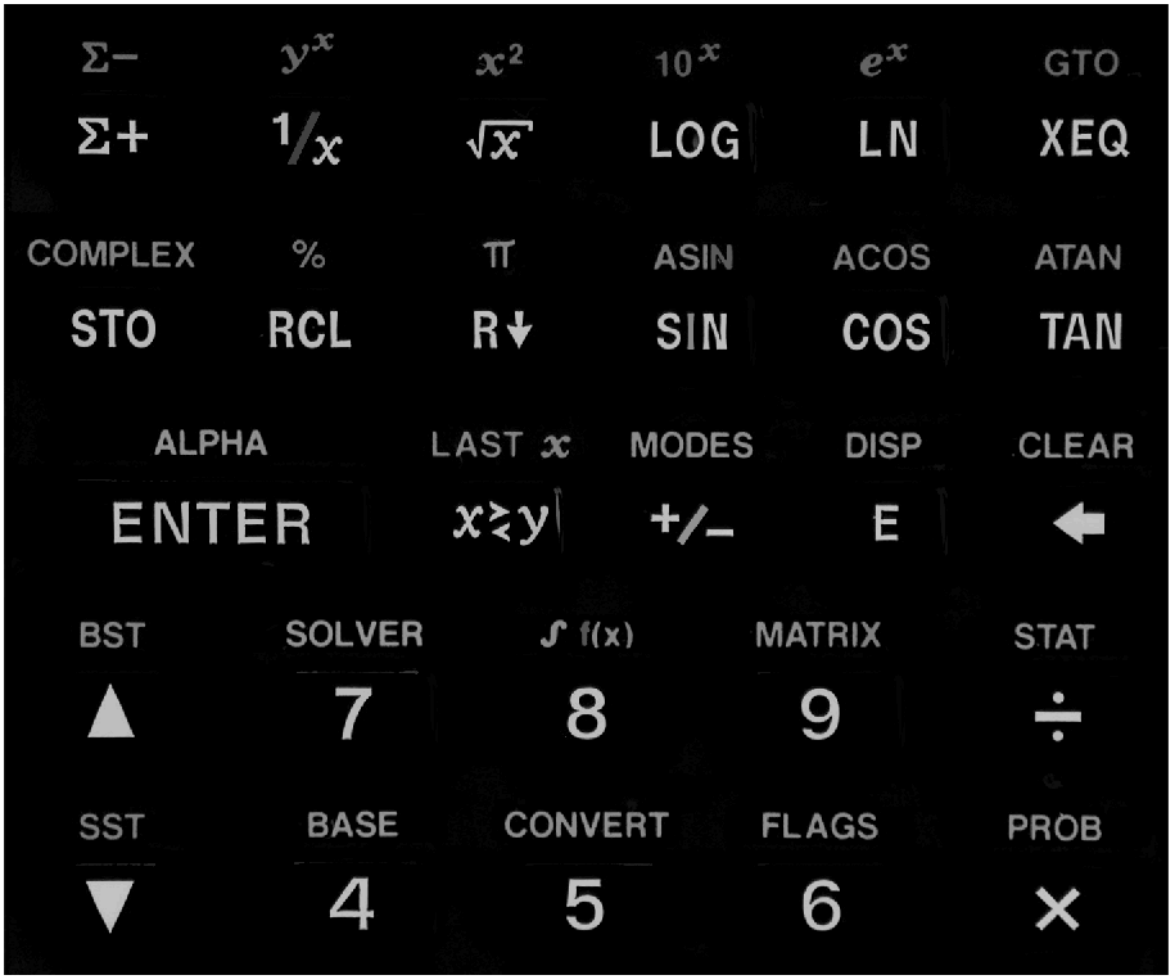


可以发现 i 和 % 都有缺失，因为近似竖直结构元且比它小，被腐蚀了。我们要恢复他们。

用短水平线膨胀 g_obr，使得被腐蚀字符的相邻字符被膨胀，使得被腐蚀区域被重叠(这就有了在此区域的像素，为重建打基础

将顶帽重建后的图像用作模板(基于它重构，才能恢复回去)，用最小化图像用作标记图像，进行重建

```
g_obrd = imdilate(g_obr, ones(1, 21));
f2 = imreconstruct(min(g_obrd, f_thr), f_thr);
figure, imshow(f2)
```



警告: 图像太大，无法在屏幕上显示；将以 67% 显示

函数自定义区

```
function g = endpoints(f)
    persistent lut % 定义持久变量
    if isempty(lut)
        lut = makelut(@endpoint_fcn, 3); % 传入一个判别函数，得到一个查找表，原理查看实验书 9.3.3。
    end
    g = applylut(f, lut);
end

% 查看此像素是否为端点
function is_end_point = endpoint_fcn(nhood)
    % nhood 是 3 * 3 的二值矩阵表示领域，如果中心元素是端点，就返回 1
```

```
%     [0 0 0
%      0 1 0
%      0 0 1]
%      [1 0 0
%      0 1 0
%      0 0 0]
%   类似如上的都算端点
    is_end_point = nhood(2, 2) & (sum(nhood(:)) == 2);
end
```