2.9

a)

$$t = \frac{500 * 1024 * 1024 * (8+2)}{3 * 10^6} = 1747.62s$$

b)

$$t = \frac{500 * 1024 * 1024 * (8+2)}{3 * 10^{10}} = 0.17s$$

2.14

a) 不是 4 邻接
b) 是 8 邻接
c) 是 m 邻接

2.18

a)
4 通路：不存在
8 通路：4
m 通路：5
b)
4 通路：6
8 通路：4
m 通路：6

3.1

设f为灰度变换函数，且原图中，灰度值为L_i的像素频率为p_i ，转换后的灰度值$L_i{}'$为$\left[(L-1)\sum_{n=0}^{i} p_n\right]$

即

$$f(L) = \left[(L-1)\sum_{n=0}^{i} p_n\right]$$

，其中[ ]为取整符号。

3.5

1)
将低有效比特平面设为 0，会使得图像的细节信息有损失，但对总体信息影响

不大，会使得直方图整体像素略偏小，比原有直方图左移。
2)
将高有效比特平面设为 0，则会损失较多的总体信息，并且使得像素灰度值范围缩减一半，直方图中低灰度占比大大增高。

3.29

1) 忽略边界效应，考虑一个3×3的均值滤波器，每次滤波使得像素中心点是周围九格的均值，反复应用该滤波器将使得整张图所有像素均为同一值，该值应为原图的平均灰度值
2) 考虑边界效应，由于在边界不停补 0，最终使图像的值均为 0

3.31

设核为$K_{h\times w}$，被处理图像周围用零补全后记为 $A_{m\times n}$ ，记

$$\sum_{p=1,q=1}^{h,w} K_{p,q} = 1$$

$$\sum_{i=1,j=1}^{m,n} A_{i,j} = S$$

命题等价于

$$S = \sum_{i=1,j=1,p=1,q=1}^{m,n,h,w} A_{i+p,j+q}\, K_{p,q}$$

右式展开为

$$\sum_{i=1,j=1,p=1,q=1}^{m,n,h,w} A_{i+p,j+q}\, K_{p,q} = \sum_{p=1,q=1}^{h,w} K_{p,q} \sum_{i=1,j=1}^{m,n} A_{i+p,j+q}$$

$p \in [1,h]$ 且 $q \in [1,w]$时，有：

$$\sum_{i=1,j=1}^{m,n} A_{i+p,j+q} = \sum_{i=1,j=1}^{m,n} A_{i,j} = S$$

可得：

$$\sum_{p=1,q=1}^{h,w} K_{p,q} \times S = S$$

证毕

3.37

1) 将其中元素进行排序，取第$\text{int}(\frac{n^2}{2})$个元素

2) 在 1)中对每个数据进行下标，记录其所属列数，剔除 1)中第一列元素，对新加入的一列元素进行排序。

3.40

中心为-4 的拉普拉斯尽让图像在水平与竖直两个方向产生了锐化效果；而中心为-8 的拉普拉斯在对角线方向上产生了额外的锐化效果。

4.38

$$1. (f \star h)(x,y) \Leftrightarrow (F \cdot H)(u,v)$$

$$\begin{aligned}
\mathcal{F}(f(x,y) \star h(x,y)) &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[(f(x,y) \star h(x,y))e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right] \\
&= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n)\right)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left[f(m,n)\left(\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} h(x-m,y-n)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right)\right] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}[f(m,n)\mathcal{F}(h(x-m,y-n))] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left[f(m,n)H(u,v)e^{-j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}\right] \\
&= F(u,v)H(u,v)
\end{aligned}$$

$$2. (f \cdot h)(x,y) \Leftrightarrow (1/MN)[(F \star H)(u,v)]$$

$$\begin{aligned}
\mathcal{F}^{-1}((1/MN)[(F \star H)(u,v)]) &= \frac{1}{(MN)^2}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left[(F(u,v) \star H(u,v))e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right] \\
&= \frac{1}{(MN)^2}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left[\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F(m,n)H(u-m,v-n)\right)e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right] \\
&= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left[F(m,n)\left(\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} H(u-m,v-n)e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}\right)\right] \\
&= f(x,y)h(x,y)
\end{aligned}$$

4.48

直接逆变换

$$
\begin{aligned}
\mathcal{F}^{-1}(H) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} H\,(u,v) e^{j2\pi(ux+vy)} dudv \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A\, e^{j2\pi(ux+vy)-\frac{u^2+v^2}{2\sigma^2}} dudv \\
&= A\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{j2\pi\left(ux-\frac{u^2}{2\sigma^2}+vy-\frac{v^2}{2\sigma^2}\right)} dudv \\
&= A\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{j2\pi\left(ux-\frac{u^2}{2\sigma^2}\right)}\, e^{j2\pi\left(vy-\frac{v^2}{2\sigma^2}\right)} dudv \\
&= A\int_{-\infty}^{\infty} e^{j2\pi\left(ux-\frac{u^2}{2\sigma^2}\right)} du \int_{-\infty}^{\infty} e^{j2\pi\left(vy-\frac{v^2}{2\sigma^2}\right)} dv
\end{aligned}
$$

求积分：

$$
\int_{-\infty}^{\infty} e^{j2\pi\left(ux-\frac{u^2}{2\sigma^2}\right)} du = \sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2x^2}
$$

带入得证

4.49

K 次处理等价于在频域中乘以$e^{\frac{-KD^2(u,v)}{2D_0^2}}$

$$
\lim_{K\to\infty} e^{\frac{-KD^2(u,v)}{2D_0^2}} = 0 \text{ or } 1
$$

当且仅当$u = v = 1$时取 1。随着 K 增大，图像将越来越接近于单色常值图像，且其值为灰度平均值。

4.56

$$
H_{LP} + H_{HP} = 1
$$

$$
1 - \frac{1}{1+\left(\frac{D(u,v)}{D_0}\right)^{2n}} = \frac{1}{1+\left(\frac{D_0}{D(u,v)}\right)^{2n}} = H(u,v)
$$

5.21

计算$h(x,y)$的傅里叶变换：

$$\begin{aligned}\mathcal{F}(h(x,y)) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h\,(x,y)e^{-j2\pi(ux+vy)}dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x-a)^2-(y-b)^2}\, e^{-j2\pi(ux+vy)}dxdy \\ &= \mathcal{F}_x^u\left(e^{-(x-a)^2}\right)\mathcal{F}_y^v\left(e^{-(y-b)^2}\right) \\ &= \pi e^{-\pi u(\pi u+2ja)}e^{-\pi v(\pi v+2jb)}\end{aligned}$$

计算$f(x,y)$的傅里叶变换：

$$\begin{aligned}\mathcal{F}(f(x,y)) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f\,(x,y)e^{-j2\pi(ux+vy)}dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta\,(x-a)e^{-j2\pi(ux+vy)}dxdy \\ &= \int_{-\infty}^{\infty} e^{-j2\pi(ua+vy)}\,dy \\ &= e^{-j2\pi ua}\mathcal{F}_y^v(1) \\ &= \delta(v)e^{-j2\pi ua}\end{aligned}$$

计算$G(u,v) = F(u,v)H(u,v)$

$$\begin{aligned}G(u,v) &= F(u,v)H(u,v) \\ &= \delta(v)\pi e^{-j2\pi ua}e^{-\pi u(\pi u+2ja)}e^{-\pi v(\pi v+2jb)}\end{aligned}$$

做傅里叶逆变换得到图像

$$\begin{aligned}\mathcal{F}^{-1}(G) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G\,(u,v)e^{j2\pi(ux+vy)}dudv \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta\,(v)\pi e^{-j2\pi ua}e^{-\pi u(\pi u+2ja)}e^{-\pi v(\pi v+2jb)}e^{j2\pi(ux+vy)}dudv \\ &= \pi\int_{-\infty}^{\infty} \delta\,(v)e^{-\pi v(\pi v+2jb)}e^{j2\pi vy}\left(\int_{-\infty}^{\infty} e^{-j2\pi ua}e^{-\pi u(\pi u+2ja)}e^{j2\pi ux}du\right)dv \\ &= \sqrt{\pi}e^{-(x-2a)^2}\int_{-\infty}^{\infty} \delta\,(v)e^{-\pi v(\pi v+2jb)}e^{j2\pi vy}dv \\ &= \sqrt{\pi}e^{-(x-2a)^2}\end{aligned}$$

5.29

1. 求模糊图像的频谱图$G(u,v)$中除了十字区域部分的灰度平均值
2. 构造一张图像$f(x,y)$，大小和模糊图像一致，先用上一步求得到的灰度平均值填充，然后按题干信息构造十字，并求其频谱图$F(u,v)$
3. 求$H(u,v) = \frac{G(u,v)}{F(u,v)}$

6.4

在黑白相机拍摄时，分别使用能够通过 R、G、B 分量的滤波片，则在底片上出现白色显著的是对应 R、G、B 颜色的零件。

6.5

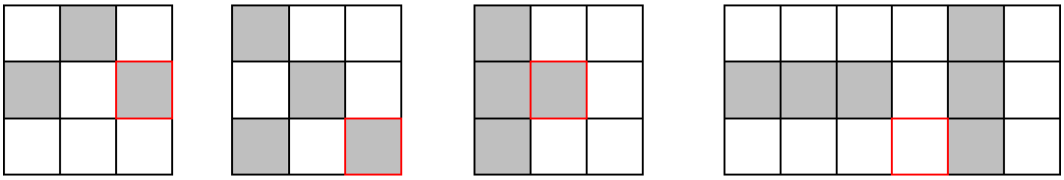
RGB=(128,255,128)，亮绿色

6.28

式子可化为$0.125r^2+g^2+b^2=D_0$，即一个 RGB 空间的椭球面

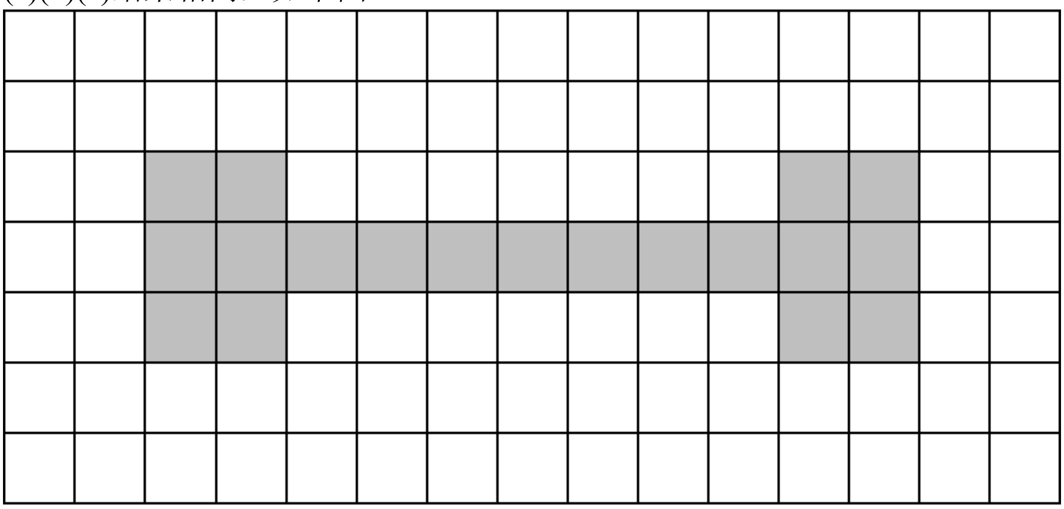
9.1

如下图所示



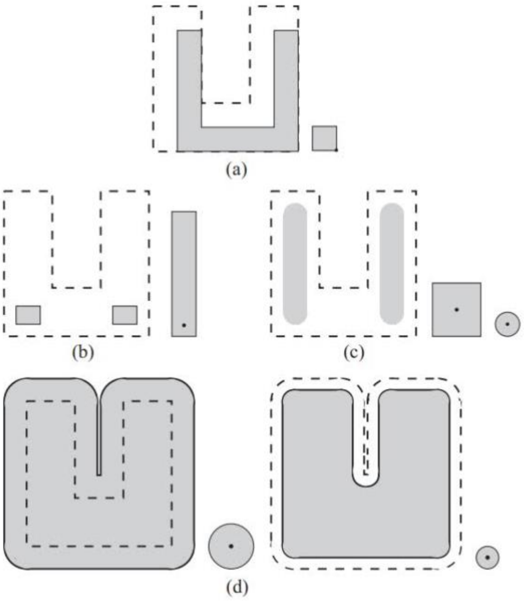
9.2

(a)(b)(c)结果相同，如下图：



9.8

如下图所示，(a)采用方型核腐蚀；(b)采用长条核腐蚀；(c)先用一个大方核腐蚀，再用一个小圆核膨胀；(d)先采用一个大圆核膨胀，再用一个小圆核腐蚀。



9.30

通过轮廓检测判断图像是否具有闭合轮廓，有轮廓的是 Lake；没有轮廓的图形为 Bay 或 Line，连接图形的端点，如果连线与图形的交点只有这两个端点，则是 Bay，否则是 Line。

或者使用行扫描，根据每行的像素累计值也可以区分三种图形。

9.37

s1 拷贝待提取图像
s2 在待提取图像中任选一个像素值为 1 的点，提取其连通域
s3 拷贝到待提取图像的备份中，然后在待提取图像中把该连通域置 0
s4 循环 s2 与 s3，直到待提取图像完全置 0

10.3

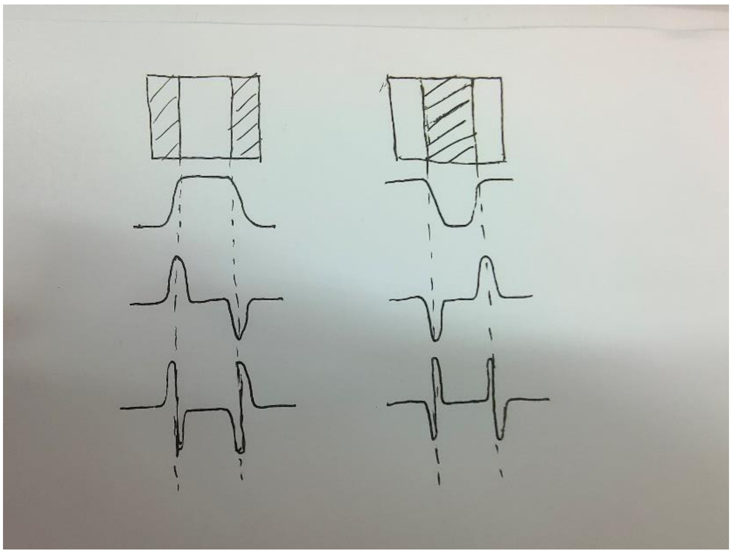
3x3 kernels 如下所示：

水平：$\begin{matrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{matrix}$　　垂直：$\begin{matrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{matrix}$

45°：$\begin{matrix} 0 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & 0 \end{matrix}$　　-45°：$\begin{matrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{matrix}$

10.9

如图所示



10.26

(1)由于 x=0,y=0,则 ρ=0，θ=any 因此是一条横向直线。
(2)是的。原因：(x, y)不为(0,0)时，ρ=xcosθ+ysinθ，对于确定的 x, y 而言，ρ, θ 满足三角函数关系。
(3) ρ=xcosθ+ysinθ，当 θ=+/-90°，ρ=+/-y，因此 reflective adjacent.

11.3

(c)寻找归一化链码就是把链码看成一个自然数，找到这个自然数按顺序排列的最小值。链码 11076765543322 的排列为 07676554332211 时，其值最小，因此是归一化链码，归一化起点为 0.

11.4

(b) 0101030303323232212111 应当看作循环码，则起始差分是最后一位与第一位的差分。一次差分：3131331313031313031300。

其它问题

圆检测方法

思路一：使用标准霍夫圆检测，将平面上所有非零像素点坐标(x, y)投影到三维参数空间(a, b, r)上，参数空间的圆之间相交时，交点的次数超过阈值时，认为该交点是平面上一个圆的参数。这种方法比较慢。

思路二：参考 OpenCV 内置的霍夫梯度检测原理，使用 Canny 检测平面上的边缘，然后使用 Sobel 算子计算各边缘点上的梯度，根据梯度求直线，当所有边缘点的梯度上的直线交于某个点的次数超过阈值时，认为该交点是一个圆心，并采用投票法，将出现次数最多的距离作为圆的半径。这种方法比较快，但容易受到图像噪声的影响。

边缘检测 Roberts, Perwitt, Laplacian, Sobel, Canny 的代码(python opencv)：

```python
import cv2
import numpy as np


def edge_Roberts(img):
    # Roberts 算子
    kernelx = np.array([[-1, 0], [0, 1]], dtype=int)
    kernely = np.array([[0, -1], [1, 0]], dtype=int)

    x = cv2.filter2D(img, cv2.CV_16S, kernelx)
    y = cv2.filter2D(img, cv2.CV_16S, kernely)

    X = cv2.convertScaleAbs(x)
    Y = cv2.convertScaleAbs(y)
    img_roberts = cv2.addWeighted(X, 0.5, Y, 0.5, 0)
    return img_roberts


def edge_Prewitt(img):
    # Prewitt 算子
    kernelx = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]], dtype=int)
    kernely = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]], dtype=int)

    x = cv2.filter2D(img, cv2.CV_16S, kernelx)
    y = cv2.filter2D(img, cv2.CV_16S, kernely)

    # 转 uint8 ,图像融合
```

```python
    X = cv2.convertScaleAbs(x)
    Y = cv2.convertScaleAbs(y)
    img_prewitt = cv2.addWeighted(X, 0.5, Y, 0.5, 0)
    return img_prewitt


def edge_Sobel(img):
    # Sobel 算子
    x = cv2.Sobel(img, cv2.CV_16S, 1, 0)
    y = cv2.Sobel(img, cv2.CV_16S, 0, 1)

    # 转 uint8 ,图像融合
    X = cv2.convertScaleAbs(x)
    Y = cv2.convertScaleAbs(y)
    img_sobel = cv2.addWeighted(X, 0.5, Y, 0.5, 0)
    return img_sobel


def edge_laplace(img):
    # Laplacian
    dst = cv2.Laplacian(img, cv2.CV_16S, ksize=3)
    img_laplace = cv2.convertScaleAbs(dst)
    return img_laplace


def edge_canny(img):
    # Canny
    dst = cv2.Canny(img, 45, 90)
    img_canny = dst
    return img_canny


if __name__ == "__main__":
    img = cv2.imread('lena.jpg')
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, img_thres = cv2.threshold(img_gray, 100, 255, cv2.THRESH_BINARY)
    img_thres = img_gray

    img_roberts = edge_Roberts(img_thres)
    img_prewitt = edge_Prewitt(img_thres)
    img_sobel = edge_Sobel(img_thres)
    img_laplace = edge_laplace(img_thres)
    img_canny = edge_canny(img_thres)
```

```
    cv2.imshow('img_roberts', img_roberts)
    cv2.imshow('img_prewitt', img_prewitt)
    cv2.imshow('img_sobel', img_sobel)
    cv2.imshow('img_laplace', img_laplace)
    cv2.imshow('img_canny', img_canny)
    cv2.waitKey(0)
```