1：模拟图像转换为数字图像需要哪些处理过程？它们与数字图像分辨率有什么关系？

答：采样　　量化　　（编码）

采样：空间分辨率　衡量将模拟图像转化为数字图像的空间精度

量化：亮度分辨率　衡量图像亮度的量化精度

2：什么是图像直方图？直方图有哪些应用？

答：图像直方图是反映图像像素分布的统计表；

其中横坐标代表了图像像素的种类，可以是灰度的，也可以是彩色的；

纵坐标代表了每一种颜色值在图像中的像素总数或者占所有像素个数的百分比。

灰度直方图反映了图像灰度的分布（统计）特性

应用：

图像增强：直方图均衡

图像分割：根据直方图获取分割阈值

图像分类：直方图对比

直方图均衡：（推导过程＋实验代码）

（对输入图像采用变换函数 进行变换，换若输入和输出图像直方图 分别记为 和 ，试推导输出与输出图像直方图之间的关系式）

（给出实现图像直方图均衡的灰度变换关系式，并证明）

（什么是直方图均衡？实现直方图均衡包含哪些步骤？）

- **灰度 变换**　　$D_B = f(D_A)$
- **灰度直方图**　　$H_A(D)$：变换之前图像的直方图

　　$H_B(D)$：变换之后图像的直方图

$$H_B(D) = \frac{H_A(D)}{f'(D_A)}$$

灰度变换后图像直方图是变换前直方图与变换函数导数之比

直方图均衡化是将原图像通过某种变换，得到一幅灰度直方图为均匀分布的新图像的方法。直方图均衡化方法的基本思想是对在图像中像素个数多的灰度级进行展宽，而对像素个数少的灰度级进行缩减。从而达到清晰图像的目的。

步骤：计算直方图　计算新的灰度级　用新的灰度级替换原来的灰度级

$$D_B = \frac{D_m}{A_0} \cdot \sum_0^{D_A} H_A(D)$$

3：图像增强的主要方法有哪些?

答:

空间域增强（直接对图像中像素的灰度级进行操作）

灰度变换　代数运算　空间域滤波

频域增强（图像进行傅里叶变换等，对变换后的系数进行操作）

频域滤波

→简述图像频域增强的基本原理

➢ 频域滤波原理

$$G(u,v)=H(u,v)\times F(u,v)$$

输出图像傅氏变换　系统频域响应　输入图像傅氏变换

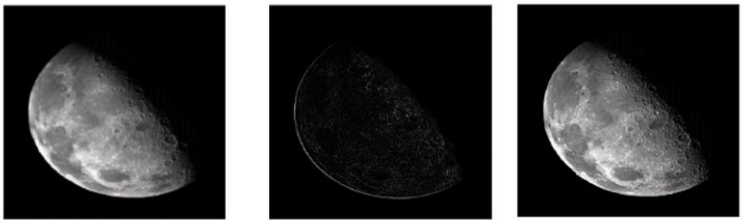
通过滤波系统“修正”输入图像频率成分从而达到图像增强目的

$H(u,v)$　低通滤波器　高通滤波器　……

➢ 频域图像增强

-- 高频滤波后，背景平均强度很小（高通滤波器滤除了傅里叶变换直流成分）

-- 原始图像与高频滤波后结果叠加



原图　拉普拉斯算子提取边缘　高频增强图像

→简述同态滤波的基本原理，给出同态滤波的步骤。

照射分量是在整个空间区域缓慢变化

反射分量在物体间的交界处急剧变化

➢ 同态滤波

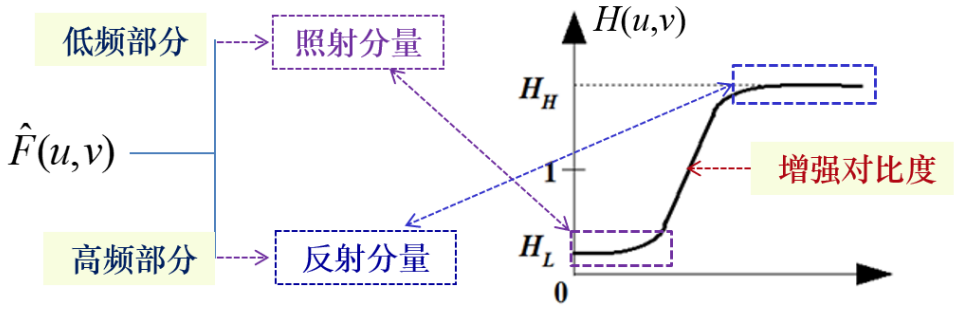
-- 基于图像成像模型

-- 在频域压缩灰度动态范围

-- 增强对比度

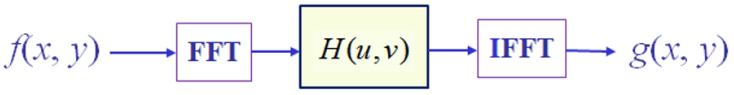
$$f(x,y)=I(x,y)\times R(x,y)$$

照射分量　反射分量

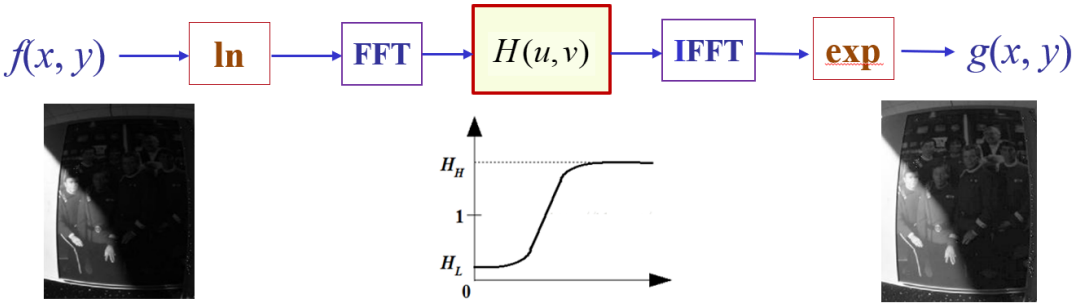


抑制低频，压缩了图像的动态范围
增强高频，加大了各部分间对比度

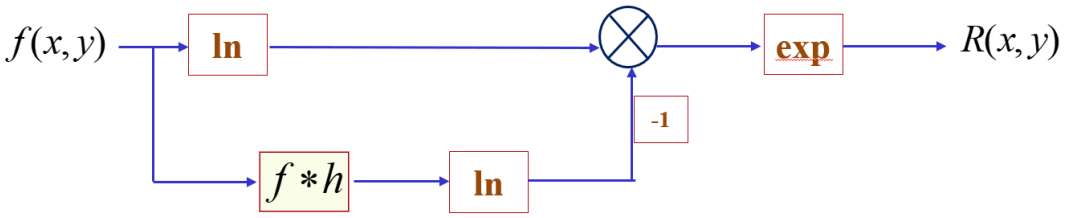
- 频域滤波

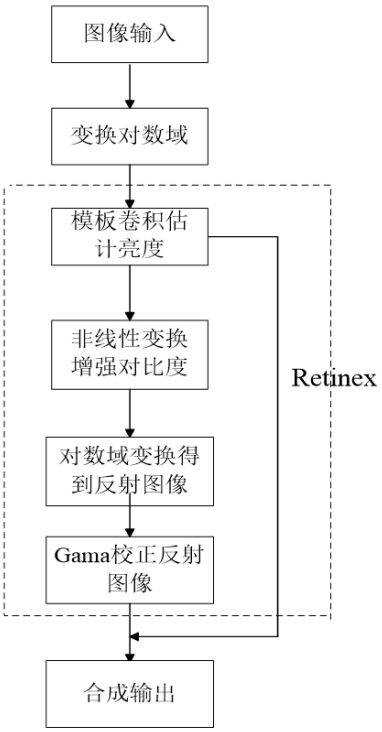


- 同态滤波

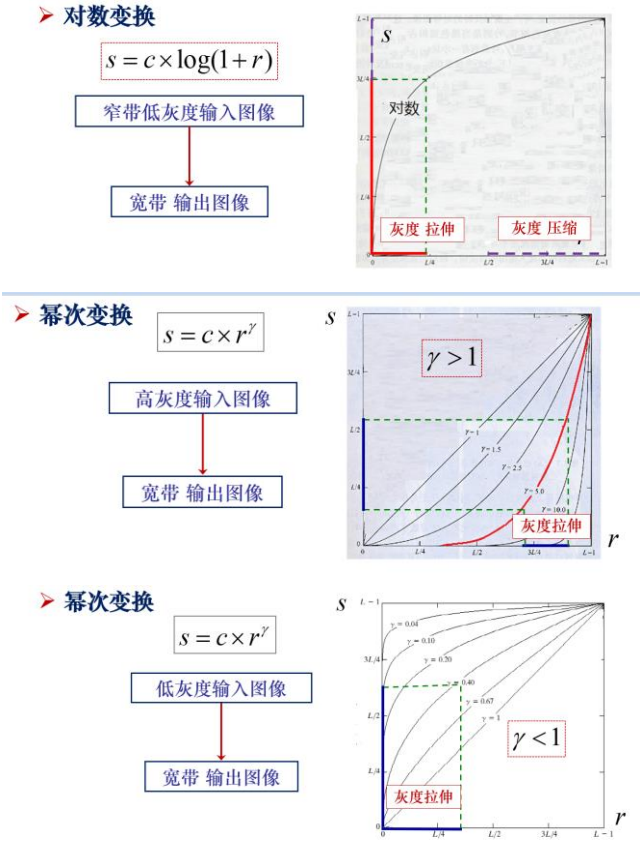


→简述基于 Retinex 图像增强的基本原理，给出增强系统的框图





4. 给出对数变换和指数变换的公式，并说明它们分别适合何种图像的增强？



5：实现图像平滑的滤波器有哪些？它们分别有何特点？

答：低通滤波器

均值滤波器：使用滤波器窗口内的像素的平均灰度值代替图像中的像素值

降低噪声的同时，也会模糊图像的边缘

【1 1 1】

（简述中值滤波器的基本思想及其特点）

中值滤波器：邻域内像素（包括原像素）灰度排序,取中间值

使突出的亮（暗）点更接近它周边的点，消除孤立的亮点或者暗点

去除噪声的同时，比较好地保留边缘

能够有效去除脉冲噪声

高斯滤波器：

无振铃现象

(哪些滤波器具有振铃现象？原因是什么？)

举例：理想低通滤波器；理想高通滤波器

图像处理中，对一幅图像进行滤波处理，若选用的频域滤波器具有陡峭的变化，则会使滤波图像产生“振铃”，所谓“振铃”，就是指输出图像的灰度剧烈变化处产生的震荡，就好像钟被敲击后产生的空气震荡

6：实现图像锐化的滤波器有哪些？它们分别有何特点？

答：高通滤波器

Laplace 滤波器：

对噪声敏感；不能检测边缘的方向；零交叉性质进行边缘定位

Sobel 滤波器：

对噪声敏感度较低；可以检测边缘的方向；无法进行边缘定位

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Laplace算子

−1	0	1
−2	0	2
−1	0	1

−1	−2	−1
0	0	0
1	2	1

Sobel 算子

代码实现

7. 形态学运算腐蚀和膨胀分别有何种效果？

答:

膨胀时将与物体接触的所有背景点合并到该物体中，使边界向外部扩张的过程；可以用来填补物体中的空洞

腐蚀是一种消除边界点，使边界向内部收缩的过程；用来消除小且无意义的物体

8. 什么是开运算和闭运算？

答:

开运算：先腐蚀再膨胀

用来消除小物体，在纤细处分离物体

平滑较大物体的边界同时并不明显改变其面积

闭运算：先膨胀再腐蚀

用来填充物体内细小空洞

连接邻近物体

平滑其边界的同时并不明显改变其面积

9. 简述图像分割的主要方法。

答:

1) 基于阈值的分割方法

（基于阈值的分割中，阈值选取可以采用哪些方法）

待分割图像 $f(x,y)$ 分割后图像 $g(x,y)$

$$g(x,y)=\begin{cases}1, & f(x,y)>T\\ 0, & f(x,y)\le T\end{cases}$$

T：阈值

✓ 全局阈值：T 取决于整幅图像阈值

✓ 局部阈值：T 取决于局部图像阈值

✓ 动态阈值：T 取决于空间坐标 (x,y)

阈值选取：直方图技术、最小误差阈值法、最大方差阈值法

2) 基于边缘的分割方法

给出几种边缘提取算子：

➢ 一阶算子

Robert算子

-1	0	0	-1
0	1	1	0

Prewitt 算子

-1	0	1	-1	-1	-1
-1	0	1	0	0	0
-1	0	1	1	1	1

Sobel 算子

-1	0	1	-1	-2	-1
-2	0	2	0	0	0
-1	0	1	1	2	1

➢ 二阶算子

Laplace算子

0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

3) 基于区域的分割方法

利用图像像素的空间性质，分割出有相似性质，属于同一个区域的像素

包括区域生长法、形变模型、分裂合并法

（简述区域增长法）

➢ 区域生长法

-- 对每个需要分割的区域找一个种子像素作为生长的起点

-- 判断种子像素周围邻域中与种子像素是否具有相似性质

-- 若具有相似性质，则将该像素合并到种子像素所在的区域

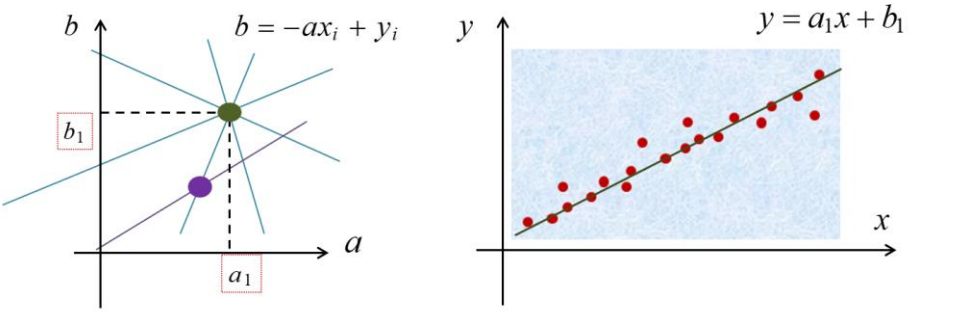
-- 将这些新像素当作新的种子像素继续进行上面的过程

-- 直到再没有满足条件的像素可被包括进来

4) 基于学习的分割方法

10. 简述 Hough 变换的基本思想

◆ Hough变换基本思想：转换到参数平面



在参数平面中相交直线最多的点，对应的 x，y 平面的直线即为所求的直线。

◆ **Hough变换**

✓ 优点

-- 抗噪声能力强

-- 能够在信噪比较低的条件下检测出直线或解析曲线。

✓ 缺点

-- 需要首先做二值化以及边缘检测等图像预处理工作

-- 损失掉原始图像中的许多信息。

- 代码的关键部分

处理的像素点

Image[i*width+j]

图像的尺度变换

$$normImg(j,i) = image(col,row)$$

$$row = i \times \frac{height}{newHei} \qquad col = j \times \frac{width}{newWid}$$

图像的二值化

```
for (i = 0; i<height; i++)
{
    for (j = 0; j<width; j++)
    {
        if (image[i*width + j]>150)
            outImg[i*width + j] = 160;
        else
            outImg[i*width + j] = 10;
    }
}
```

计算直方图

```
for (i = 0; i<height; i++)
{
    for (j = 0; j<width; j++)
    {
        gray = image[i*width + j];
        hist[gray]++;
    }
}
```

灰度变换

直方图均衡

```
void CSampDispView::hisEqualiz(BYTE*image,int w,int h,BYTE*outImg)
{
    int his[256];
    int n,i,j;

    for( n=0; n<256; n++)//initialize
        his[n]=0;

    // compute histogram of image
    for( i=0; i<h; i++)
        for( j=0; j<w; j++)
            his[ image[i*w+j] ] ++;

    //accumulation
    for( n=1; n<256;n++)
        his[n] += his[n-1];

    BYTE gray[256]; //compute the new graylevel
    float cons;
    cons = 255./his[255];

    for( n=0; n<256; n++)
        gray[n] = (BYTE)(cons*his[n]);
}
```

$$D_B = \frac{D_m}{A_0}\sum_{0}^{D_A} H_A(D)$$

第一步：计算直方图

第二步：计算新的灰度级

第三步：用新的灰度级替换原来的灰度级

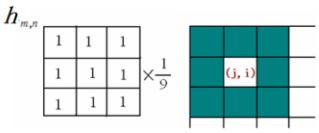
```
for( i=0; i<h; i++)
    for( j=0; j<w; j++)
        outImg[i*w+j]=gray[image[i*w+j]];
```

低通滤波----------均值滤波

```
int smth[9];
int i, j, m,n;    for( i=0; i<9; i++)
BYTE block[9];        smth[i] = 1;

int value;
for( i=0; i<hei; i++)
    for( j=0; j<wid; j++)
    {
        if(i==0||j==0||i==wid-1||j==hei-1)
            outImg[i*wid+j]=0;
        else
        { //pick up a 3x3 block
            for( m=-1; m<2; m++)
                for( n=-1; n<2; n++)
                    block[(m+1)*3+n+1] = image[(i+m)*wid+j+n];
            value = convolution( smth, block);
            outImg[i*wid+j] = BYTE( value/9.+0.5);
        }
    }
```

```
// add image processing function
void smooth(BYTE*, int, int, BYTE*);
int convolution( int *, BYTE*);
```



$$y(j,i) = \sum_{m=-1}^{1}\sum_{n=-1}^{1}\frac{1}{9}x(j-m,i-n)$$

```
int CImgProView::convolution( int *operatr, BYTE*block)
{
      int value;
      int i, j;
      value = 0;
      for( i=0; i<3; i++)
          for( j=0; j<3; j++)
              value += operatr[i*3+j]*block[i*3+j];

       return value;
}
```

低通滤波----------高斯滤波

1	2	1
2	4	2
1	2	1

高斯滤波

低通滤波----------中值滤波：

−1	0	1
−2	0	2
−1	0	1

−1	−2	−1
0	0	0
1	2	1

冒泡排序

边缘提取：其实意思一样，只不过算子不同

Sobel 算子:

膨胀腐蚀：其实意思一样

膨胀：周边有图形点，该点就赋值 255

腐蚀：周边有 3 个以上的非图像点，该点就赋值为 0