# 去噪算法

从统计学的观点来看，凡是统计特征不随时间变化的噪声称为平稳噪声，而统计特征随时间变化的噪声称为非平稳噪声。

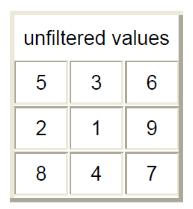
幅值基本相同，但是噪声出现的位置是随机的，称为椒盐噪声；

如果噪声的幅值是随机的，根据幅值大小的分布，有高斯型和瑞利型两种，分别称为高斯噪声和瑞利噪声。

## 均值滤波器： 取3X3/5X5 矩阵，求和再平均

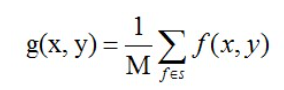
优点：算法简单，计算速度快；

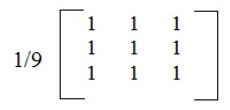
缺点：降低噪声的同时使图像产生模糊，特别是景物的边缘和细节部分。











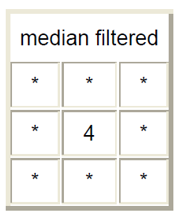
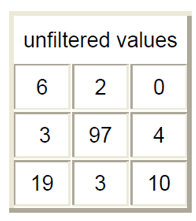
|  |
| --- |
| g = (f(x-1,y-1) + f(x,y-1)+ f(x+1,y-1) + f(x-1,y) + f(x,y) + f(x+1,y)  + f(x-1,y+1) + f(x,y+1) + f(x+1,y+1))/9 |

|  |
| --- |
| //边界没有处理  void meanFilter (unsigned char\* corrupted, unsigned char\* smooth, int width, int height)  {    memcpy ( smooth, corrupted, width\*height\*sizeof(unsigned char) );    for (int j=1;j<height-1;j++)  {  for (int i=1;i<width-1;i++)  {  smooth [ j\*width+i ] = ( corrupted [ (j-1)\*width+(i-1) ] + corrupted [ (j-1)\*width+i] + corrupted [ (j-1)\*width+(i+1) ] +  corrupted [ j\*width+(i-1) ] + corrupted [ j\*width+i] + corrupted [ j\*width+(i+1) ] +  corrupted [ (j+1)\*width+(i-1) ] + corrupted [ (j+1)\*width+i] + corrupted [ (j+1)\*width+(i+1) ] ) / 9;  }  }  } |

## 中值滤波： 取3X3/5X5 矩阵，按小到大排序，取中值

优点：抑制效果很好，画面的清析度基本保持；

缺点：对高斯噪声的抑制效果不是很好。





|  |
| --- |
| g = median[(x-1,y-1) + f(x,y-1)+ f(x+1,y-1)  + f(x-1,y) + f(x,y) + f(x+1,y)  +  f(x-1,y+1) + f(x,y+1) + f(x+1,y+1)] |



很明显中值滤波不仅是图像变得平滑，同时去除了椒盐噪声(图像最外圈的像素没有去除掉只是因为我没有从0-width处理而已)。从这里中值的逻辑来看， 我们做中值操作的时候， 那么白色(255)和黑色(0)因为是最大最小值， 除非周围的颜色都是黑色或者白色，不然一般都会被剔除掉， 这就是和均值最大的不同！ 所以在效果上要好很多。一般来说这个中值滤波是去除椒盐噪声的非常理想的选择。

|  |
| --- |
| void medianFilter (unsigned char\* corrupted, unsigned char\* smooth, int width, int height)  {  //memcpy ( smooth, corrupted, width\*height\*sizeof(unsigned char) );  for (int j=1;j<height-1;j++)  {  for (int i=1;i<width-1;i++)  {  int k = 0;  unsigned char window[9];  for (int jj = j - 1; jj < j + 2; ++jj)  for (int ii = i - 1; ii < i + 2; ++ii)  window[k++] = corrupted[jj \* width + ii];  // Order elements (only half of them)  for (int m = 0; m < 5; ++m)  {  int min = m;  for (int n = m + 1; n < 9; ++n)  if (window[n] < window[min])  min = n;  // Put found minimum element in its place  unsigned char temp = window[m];  window[m] = window[min];  window[min] = temp;  }  smooth[ j\*width+i ] = window[4];  }  }  } |

## 对称近邻均值滤波