实验名称	**实验 4：图像去噪**
实验目的	1、掌握算术均值滤波器、几何均值滤波器、谐波和逆谐波均值滤波器进行图像去噪的算法 2、掌握利用中值滤波器进行图像去噪的算法 3、掌握自适应中值滤波算法 4、掌握自适应局部降低噪声滤波器去噪算法 5、掌握彩色图像去噪步骤
实验内容	1、均值滤波 具体内容：利用 OpenCV 对灰度图像像素进行操作，分别利用算术均值滤波器、几何均值滤波器、谐波和逆谐波均值滤波器进行图像去噪。模板大小为 5*5。（注：请分别为图像添加高斯噪声、胡椒噪声、盐噪声和椒盐噪声，并观察滤波效果） 2、中值滤波 具体内容：利用 OpenCV 对灰度图像像素进行操作，分别利用 5*5 和 9*9 尺寸的模板对图像进行中值滤波。（注：请分别为图像添加胡椒噪声、盐噪声和椒盐噪声，并观察滤波效果） 3、自适应均值滤波。 具体内容：利用 OpenCV 对灰度图像像素进行操作，设计自适应局部降低噪声滤波器去噪算法。模板大小 7*7（对比该算法的效果和均值滤波器的效果） 4、自适应中值滤波 具体内容：利用 OpenCV 对灰度图像像素进行操作，设计自适应中值滤波算法对椒盐图像进行去噪。模板大小 7*7（对比中值滤波器的效果） 5、彩色图像均值滤波 具体内容：利用 OpenCV 对彩色图像 RGB 三个通道的像素进行操作，利用算术均值滤波器和几何均值滤波器进行彩色图像去噪。模板大小为 5*5。

实验完成情况	1、 实验步骤：先为灰度图像添加高斯噪声、胡椒噪声、盐噪声和椒盐噪声，再分别利用算术均值滤波器、几何均值滤波器、谐波和逆谐波均值滤波器进行图像去噪。模板大小为 5*5。
	核心代码如下：
	添加各类噪声：
	IplImage* AddGuassianNoise(IplImage* src) //添加高斯噪声
	{
	IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
	IplImage* noise = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
	CvRNG rng = cvRNG(-1);
	cvRandArr(&rng,noise,CV_RAND_NORMAL,cvScalarAll(0),cvScalarAll(15));
	cvAdd(src,noise,dst);
	return dst;
	}
	IplImage* AddPepperNoise(IplImage* src) //添加胡椒噪声，随机黑色点
	{
	IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
	cvCopy(src, dst);
	for(int k=0; k<8000; k++)
	{
	int i = rand()%src->height;
	int j = rand()%src->width;
	CvScalar s = cvGet2D(src, i, j);
	if(src->nChannels == 1)
	{
	s.val[0] = 0;
	}
	else if(src->nChannels==3)
	{
	s.val[0]=0;
	s.val[1]=0;
	s.val[2]=0;
	}
	cvSet2D(dst, i, j, s);
	}
	return dst;
	}
	IplImage* AddSaltNoise(IplImage* src) //添加盐噪声，随机白色点
	{
	IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
	cvCopy(src, dst);
	for(int k=0; k<8000; k++)
	{
	int i = rand()%src->height;

```
            int j = rand()%src->width;
            CvScalar s = cvGet2D(src, i, j);
            if(src->nChannels == 1)
            {
                s.val[0] = 255;
            }
            else if(src->nChannels==3)
            {
                s.val[0]=255;
                s.val[1]=255;
                s.val[2]=255;
            }
            cvSet2D(dst, i, j, s);
        }
        return dst;
    }
    IplImage* AddPepperSaltNoise(IplImage* src)       //添加椒盐噪声，随机黑白点
    {
        IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
        cvCopy(src, dst);
        for(int k=0; k<8000; k++)
        {
            int i = rand()%src->height;
            int j = rand()%src->width;
            int m = rand()%2;
            CvScalar s = cvGet2D(src, i, j);
            if(src->nChannels == 1)
            {
                if(m==0)
                {
                    s.val[0] = 255;
                }
                else
                {
                    s.val[0] = 0;
                }
            }
            else if(src->nChannels==3)
            {
                if(m==0)
                {
                    s.val[0]=255;
                    s.val[1]=255;
                    s.val[2]=255;
```

```
                    }
                    else
                    {
                        s.val[0]=0;
                        s.val[1]=0;
                        s.val[2]=0;
                    }
                }
                cvSet2D(dst, i, j, s);
            }
            return dst;
        }
```

各类滤波器实现：

```
        //算术均值滤波器——模板大小 5*5
        IplImage* ArithmeticMeanFilter(IplImage* src)
        {
            IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
            cvSmooth(src,dst,CV_BLUR,5);
            return dst;
        }
        //几何均值滤波器——模板大小 5*5
        IplImage* GeometryMeanFilter(IplImage* src)
        {
            IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
            int row, col;
            int h=src->height;
            int w=src->width;
            double mul[3];
            double dc[3];
            int mn;
            //计算每个像素的去噪后 color 值
            for(int i=0;i<src->height;i++){
                for(int j=0;j<src->width;j++){
                    mul[0]=1.0;
                    mn=0;
                    //统计邻域内的几何平均值，邻域大小 5*5
                    for(int m=-2;m<=2;m++){
                        row = i+m;
                        for(int n=-2;n<=2;n++){
                            col = j+n;
                            if(row>=0&&row<h && col>=0 && col<w){
                                CvScalar s = cvGet2D(src, row, col);
                                mul[0] = mul[0]*(s.val[0]==0?1:s.val[0]);      //邻域内的非
零像素点相乘
```

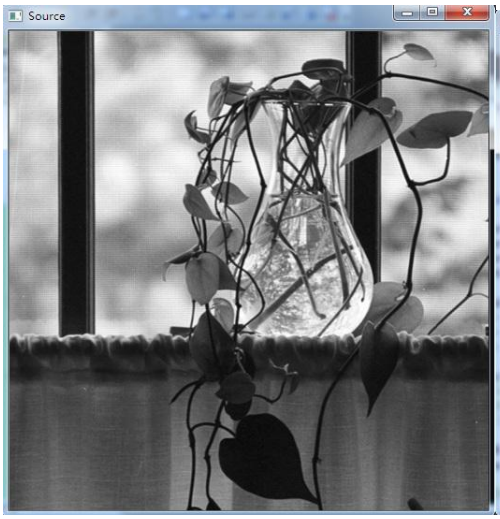
```
                                mn++;
                            }
                        }
                    }
                    //计算 1/mn 次方
                    CvScalar d;
                    dc[0] = pow(mul[0], 1.0/mn);
                    d.val[0]=dc[0];
                    //统计成功赋给去噪后图像。
                    cvSet2D(dst, i, j, d);
                }
            }
            return dst;
        }
        //谐波均值滤波器——模板大小 5*5
        IplImage* HarmonicMeanFilter(IplImage* src)
        {
            IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
            int row, col;
            int h=src->height;
            int w=src->width;
            double sum[3];
            double dc[3];
            int mn;
            //计算每个像素的去噪后 color 值
            for(int i=0;i<src->height;i++){
                for(int j=0;j<src->width;j++){
                    sum[0]=0.0;
                    mn=0;
                    //统计邻域,5*5 模板
                    for(int m=-2;m<=2;m++){
                        row = i+m;
                        for(int n=-2;n<=2;n++){
                            col = j+n;
                            if(row>=0&&row<h && col>=0 && col<w){
                                CvScalar s = cvGet2D(src, row, col);
                                sum[0] = sum[0]+(s.val[0]==0?255:255/s.val[0]);
                                mn++;
                            }
                        }
                    }
                    CvScalar d;
                    dc[0] = mn*255/sum[0];
                    d.val[0]=dc[0];
```

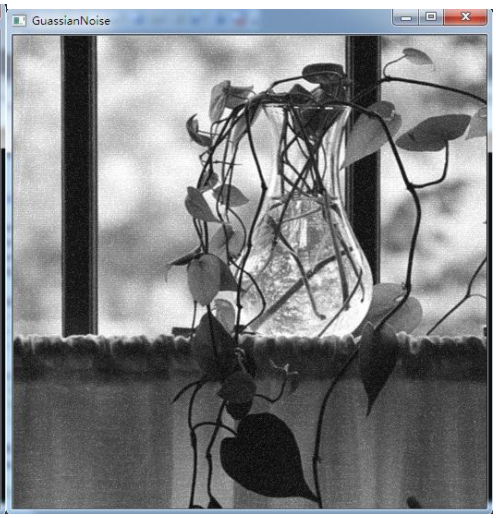
```
                    //统计成功赋给去噪后图像。
                    cvSet2D(dst, i, j, d);
                }
            }
            return dst;
        }
        //逆谐波均值大小滤波器——模板大小 5*5
        IplImage* InverseHarmonicMeanFilter(IplImage* src)
        {
            IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
            //cvSmooth(src,dst,CV_BLUR,5);
            int row, col;
            int h=src->height;
            int w=src->width;
            double sum[3];
            double sum1[3];
            double dc[3];
            double Q=2;
            //计算每个像素的去噪后 color 值
            for(int i=0;i<src->height;i++){
                for(int j=0;j<src->width;j++){
                    sum[0]=0.0;
                    sum1[0]=0.0;
                    //统计邻域
                    for(int m=-2;m<=2;m++){
                        row = i+m;
                        for(int n=-2;n<=2;n++){
                            col = j+n;
                            if(row>=0&&row<h && col>=0 && col<w){
                                CvScalar s = cvGet2D(src, row, col);
                                sum[0] = sum[0]+pow(s.val[0]/255, Q+1);
                                sum1[0] = sum1[0]+pow(s.val[0]/255, Q);
                            }
                        }
                    }
                    //计算 1/mn 次方
                    CvScalar d;
                    dc[0] = (sum1[0]==0?0:(sum[0]/sum1[0]))*255;
                    d.val[0]=dc[0];
                    //统计成功赋给去噪后图像。
                    cvSet2D(dst, i, j, d);
                }
            }
            return dst;
```

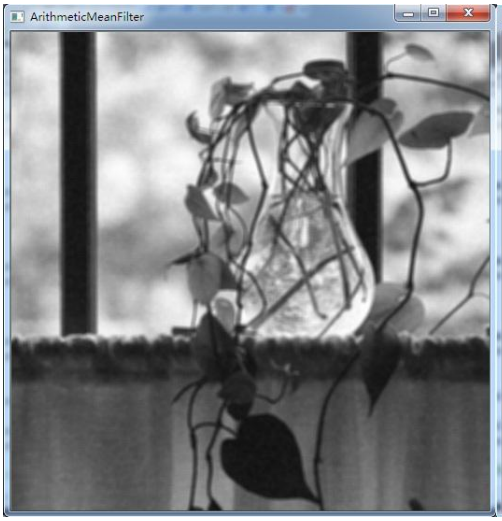
}

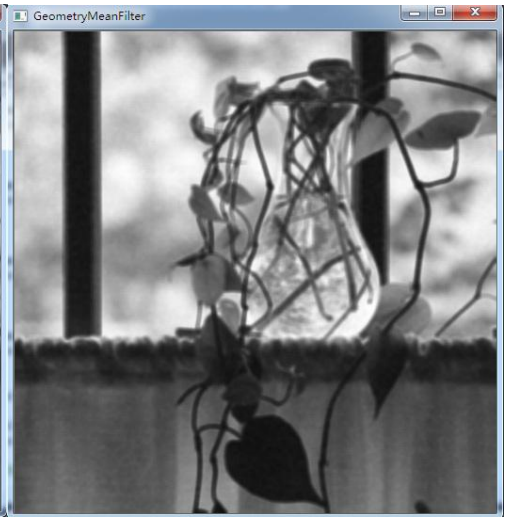
实验结果如图所示：(从左至右，从上至下分别为原图像、加噪图像、算术均值处理图像、几何均值处理图像、谐波均值处理图像、逆谐波均值处理图像)

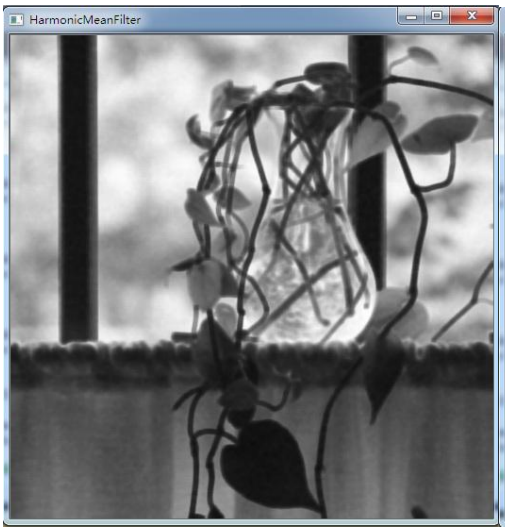
(1)高斯噪声：

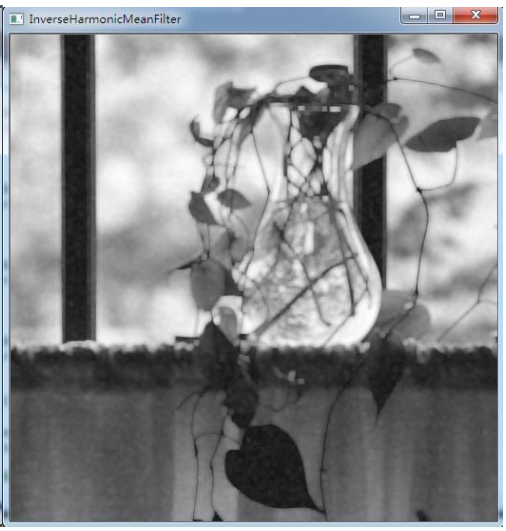




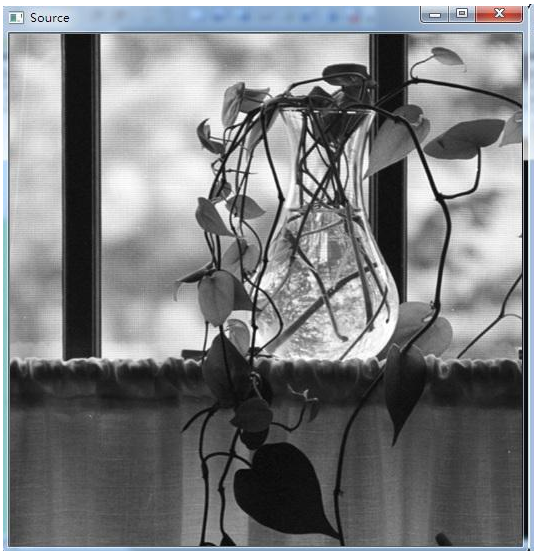


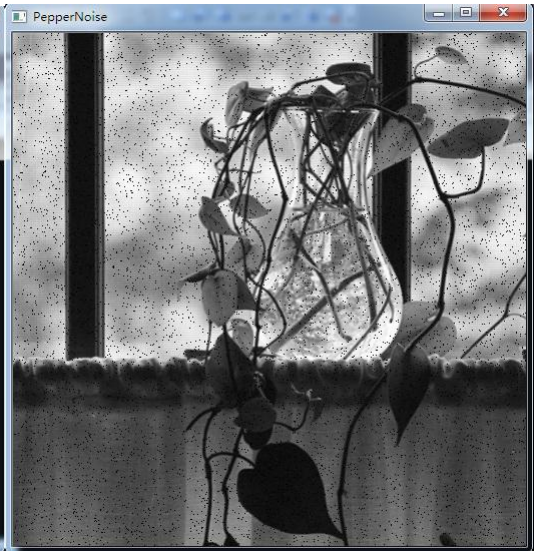


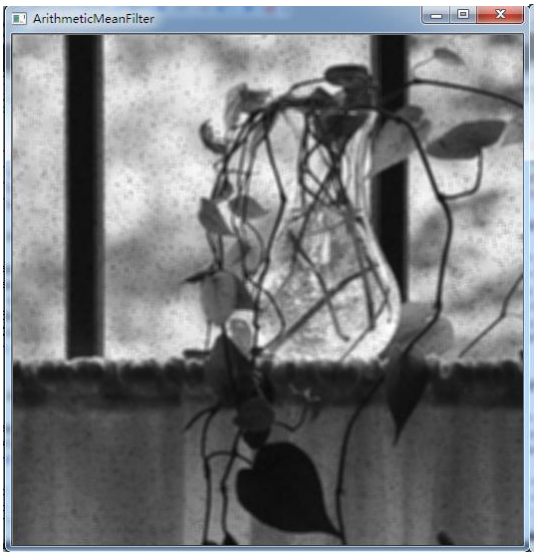




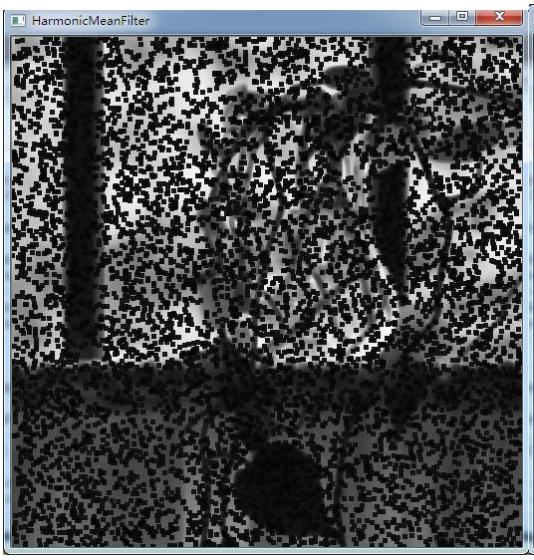
(2)胡椒噪声：

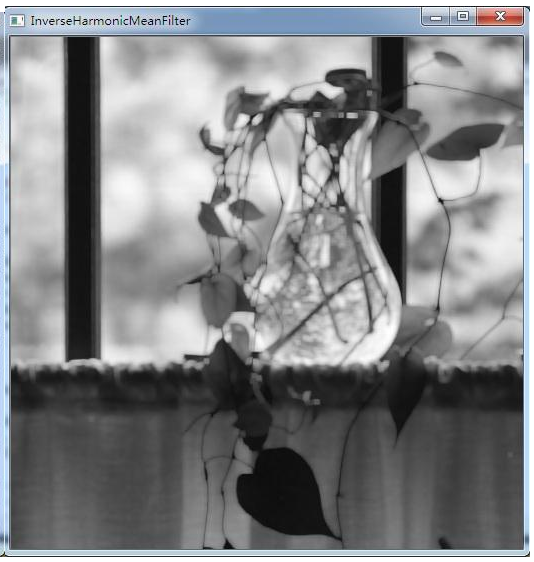

Source


PepperNoise

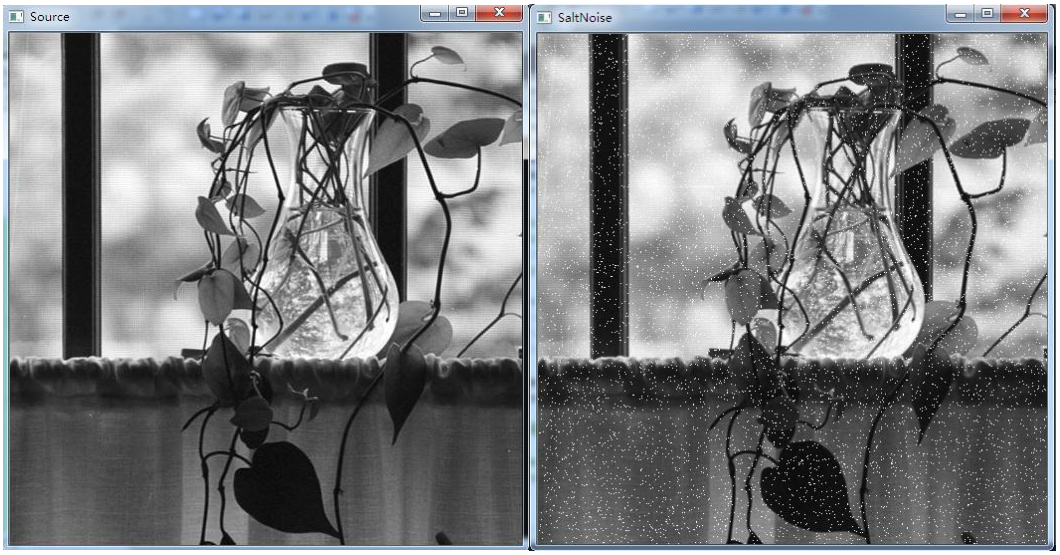

ArithmeticMeanFilter


GeometryMeanFilter

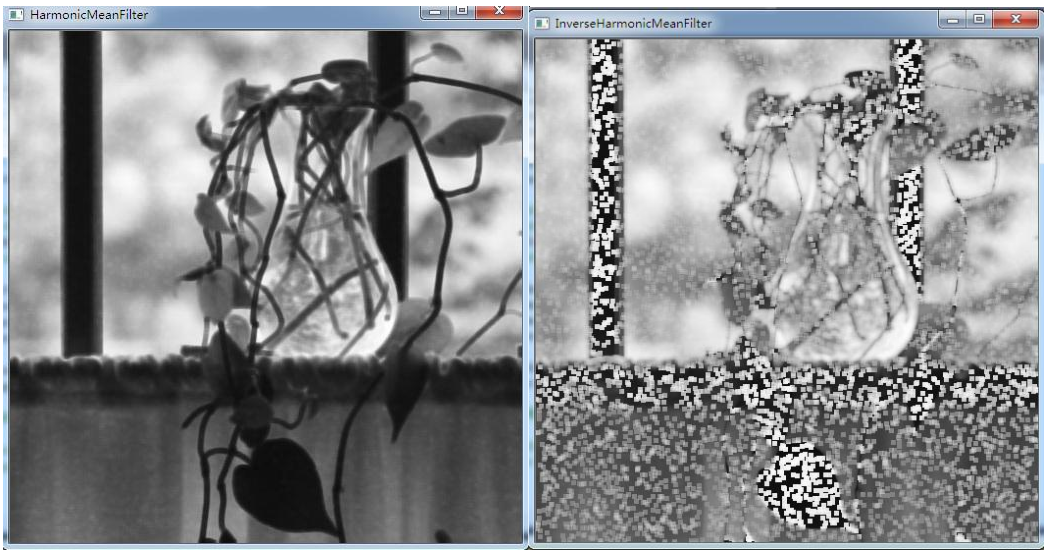

HarmonicMeanFilter


InverseHarmonicMeanFilter

(3)盐噪声

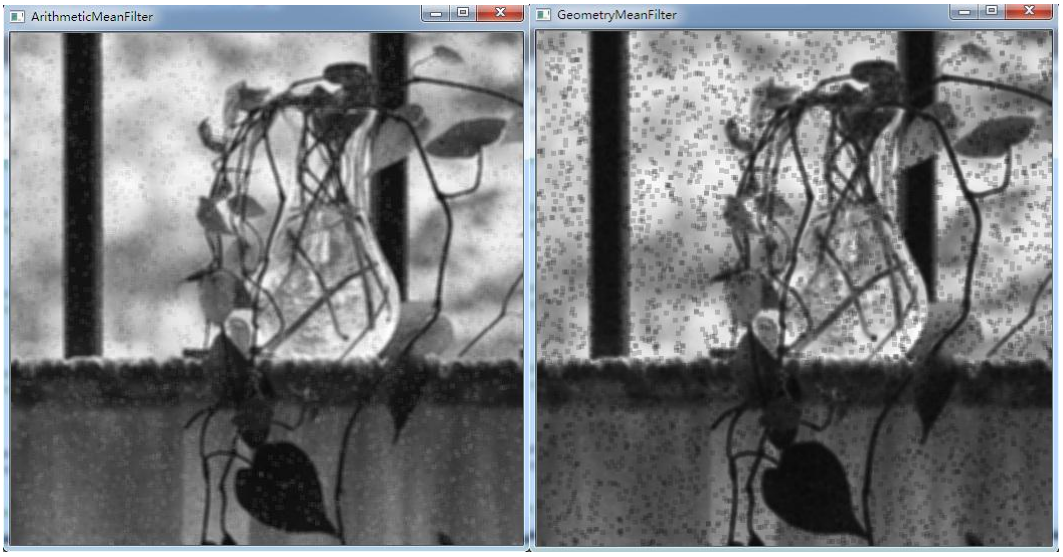

Source
SaltNoise

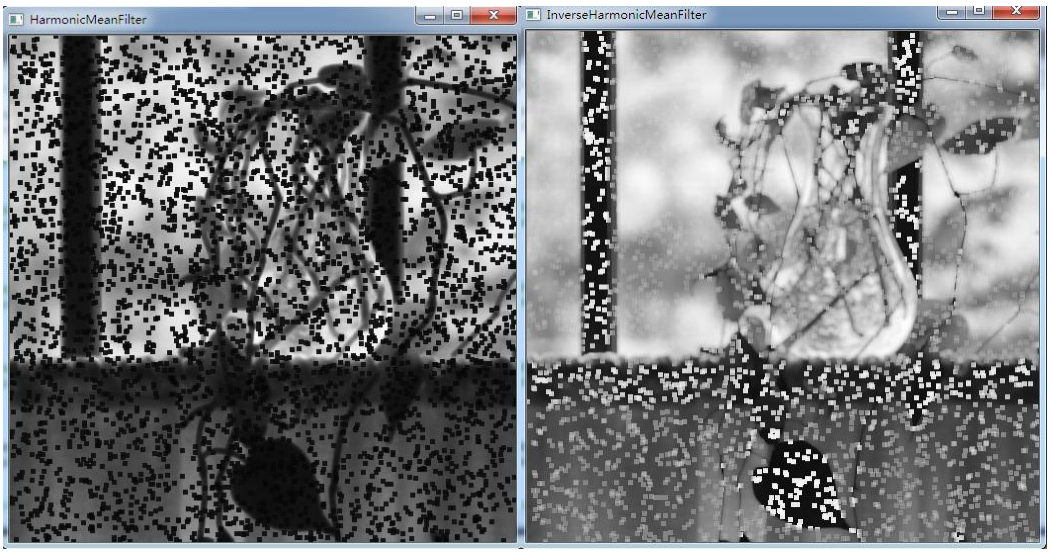

ArithmeticMeanFilter
GeometryMeanFilter


HarmonicMeanFilter
InverseHarmonicMeanFilter

(4)椒盐噪声


Source
PepperSaltNoise


ArithmeticMeanFilter
GeometryMeanFilter


HarmonicMeanFilter
InverseHarmonicMeanFilter

2、 实验步骤：先为灰度图像添加胡椒噪声、盐噪声和椒盐噪声，再分别利用 5*5 和 9*9 尺寸的模板对图像进行中值滤波。

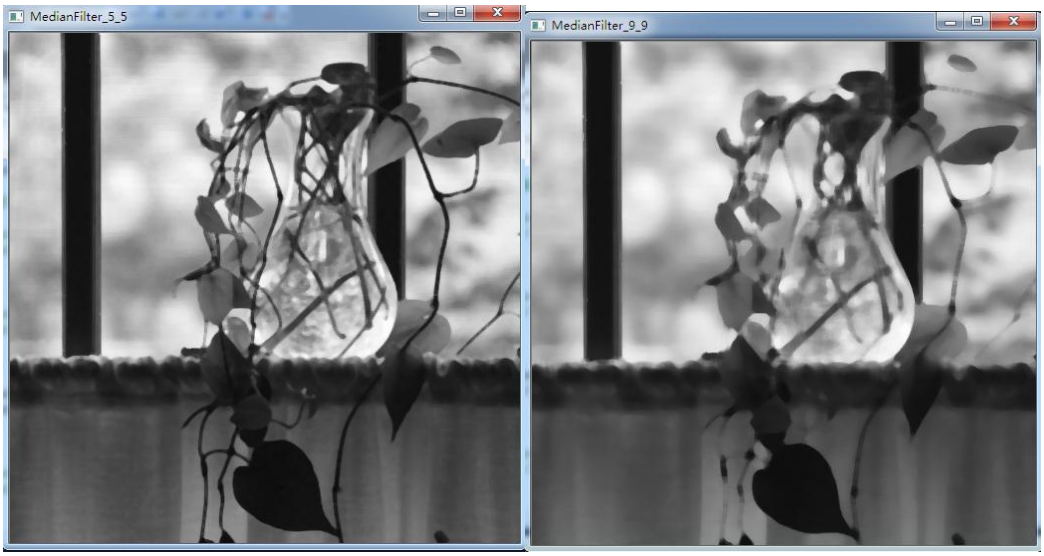
核心代码如下：

```
IplImage* MedianFilter_5_5(IplImage* src){
    IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
    cvSmooth(src,dst,CV_MEDIAN,5);
    return dst;
}
IplImage* MedianFilter_9_9(IplImage* src){
    IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
    cvSmooth(src,dst,CV_MEDIAN,9);
    return dst;
}
```

实验结果如下图（灰度图像和加噪图像第一问中已给出，下面只列出分别利用 5*5 和 9*9 尺寸的模板对图像进行中值滤波后的图像）：

灰度图像加胡椒噪声，分别利用 5*5 和 9*9 尺寸的模板对图像进行中值滤波。

灰度图像加盐噪声，分别利用 5*5 和 9*9 尺寸的模板对图像进行中值滤波。



灰度图像加椒盐噪声，分别利用 5*5 和 9*9 尺寸的模板对图像进行中值滤波。

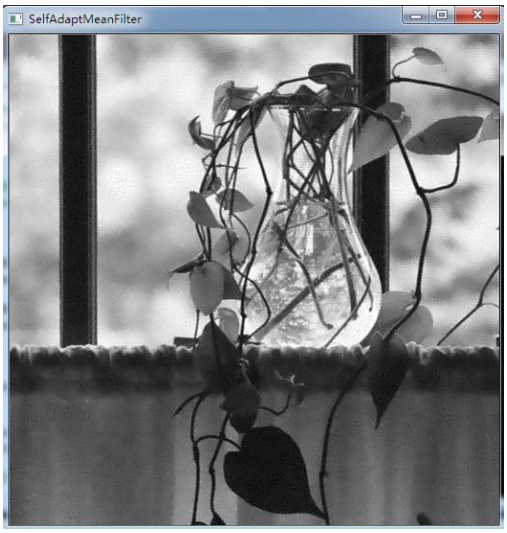
3、 实验步骤：自适应均值滤波（以高斯噪声为例），先为灰度图像添加高斯噪声，再利用 7*7 尺寸的模板对图像进行自适应均值滤波。

核心代码如下：

```
IplImage* SelfAdaptMeanFilter(IplImage* src){
    IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
    cvSmooth(src,dst,CV_BLUR,7);
    int row, col;
    int h=src->height;
    int w=src->width;
    int mn;
    double Zxy;
    double Zmed;
    double Sxy;
    double Sl;
```

```
        double Sn=100;
        for(int i=0;i<src->height;i++){
            for(int j=0;j<src->width;j++){
                CvScalar xy = cvGet2D(src, i, j);
                Zxy = xy.val[0];
                CvScalar dxy = cvGet2D(dst, i, j);
                Zmed = dxy.val[0];
                Sl=0;
                mn=0;
                for(int m=-3;m<=3;m++){
                    row = i+m;
                    for(int n=-3;n<=3;n++){
                        col = j+n;
                        if(row>=0&&row<h && col>=0 && col<w){
                            CvScalar s = cvGet2D(src, row, col);
                            Sxy = s.val[0];
                            Sl = Sl+pow(Sxy-Zmed, 2);
                            mn++;
                        }
                    }
                }
                Sl=Sl/mn;
                CvScalar d;
                d.val[0]=Zxy-Sn/Sl*(Zxy-Zmed);
                cvSet2D(dst, i, j, d);
            }
        }
        return dst;
}
```

实验结果如图：



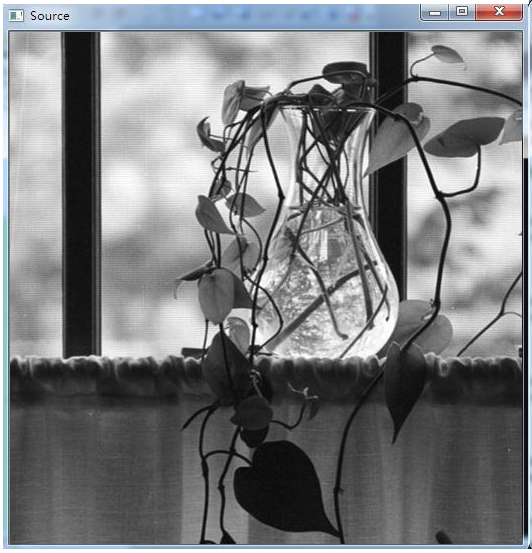
4、 实验步骤：自适应中值滤波（以椒盐噪声为例），先为灰度图像添加椒盐噪声，再利用 7*7 尺寸的模板对图像进行自适应中值滤波。

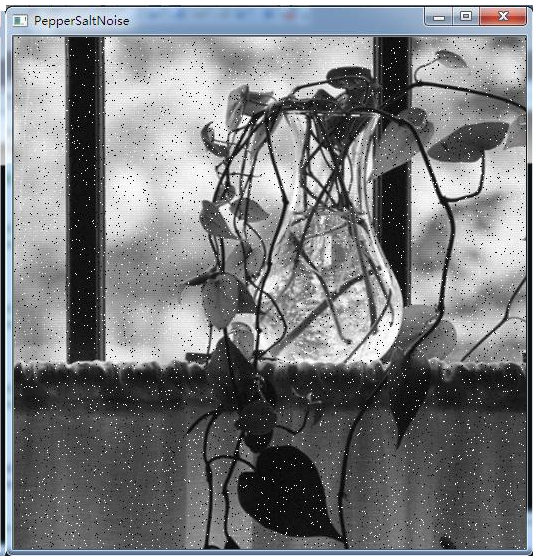
核心代码如下：

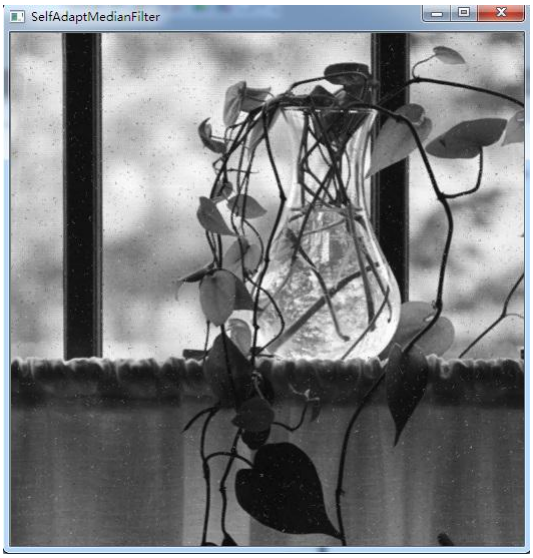
```
IplImage* SelfAdaptMedianFilter(IplImage* src){
    IplImage* dst = cvCreateImage(cvGetSize(src),src->depth,src->nChannels);
    int row, col;
    int h=src->height;
    int w=src->width;
    double Zmin,Zmax,Zmed,Zxy,Smax=7;
    int wsize;
    //计算每个像素的去噪后 color 值
    for(int i=0;i<src->height;i++){
        for(int j=0;j<src->width;j++){
            //统计邻域
            wsize=1;
            while(wsize<=3){
                Zmin=255.0;
                Zmax=0.0;
                Zmed=0.0;
                CvScalar xy = cvGet2D(src, i, j);
                Zxy=xy.val[0];
                int mn=0;
                for(int m=-wsize;m<=wsize;m++){
                    row = i+m;
                    for(int n=-wsize;n<=wsize;n++){
                        col = j+n;
                        if(row>=0&&row<h && col>=0 && col<w){
                            CvScalar s = cvGet2D(src, row, col);
                            if(s.val[0]>Zmax){
                                Zmax=s.val[0];
```

```
                                    }
                                    if(s.val[0]<Zmin){
                                        Zmin=s.val[0];
                                    }
                                    Zmed=Zmed+s.val[0];
                                    mn++;
                                }
                            }
                        }
                        Zmed = Zmed/mn;
                        CvScalar d;
                        if((Zmed-Zmin)>0 && (Zmed-Zmax)<0){
                            if((Zxy-Zmin)>0 && (Zxy-Zmax)<0){
                                d.val[0]=Zxy;
                            }else{
                                d.val[0]=Zmed;
                            }
                            cvSet2D(dst, i, j, d);
                            break;
                        } else {
                            wsize++;
                            if(wsize>3){
                                CvScalar d;
                                d.val[0]=Zmed;
                                cvSet2D(dst, i, j, d);
                                break;
                            }
                        }
                    }
                }
            }
            return dst;
        }
```

实验结果如图：





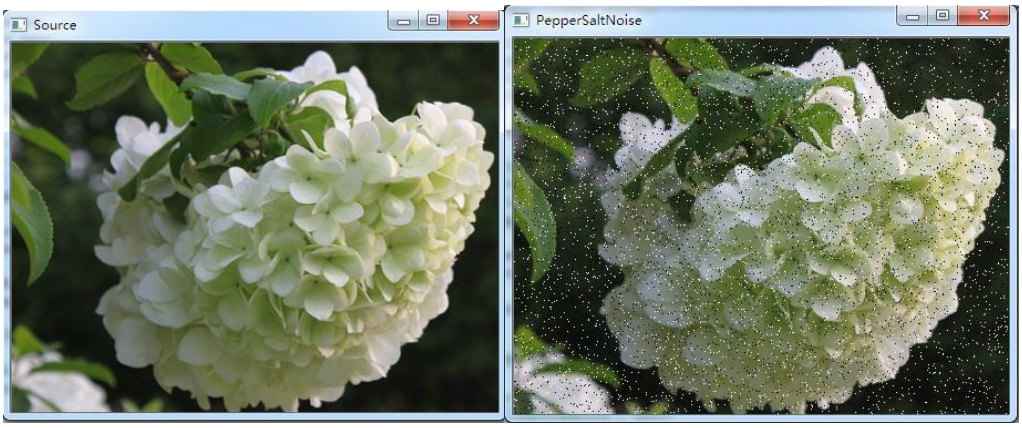


5、 实验步骤：利用 OpenCV 对彩色图像 RGB 三个通道的像素进行操作，利用算术均值滤波器和几何均值滤波器进行彩色图像去噪。模板大小为 5*5。

实验代码参照问题一，选择彩色图片、算术均值滤波器和几何均值滤波器进行彩色图像去噪。

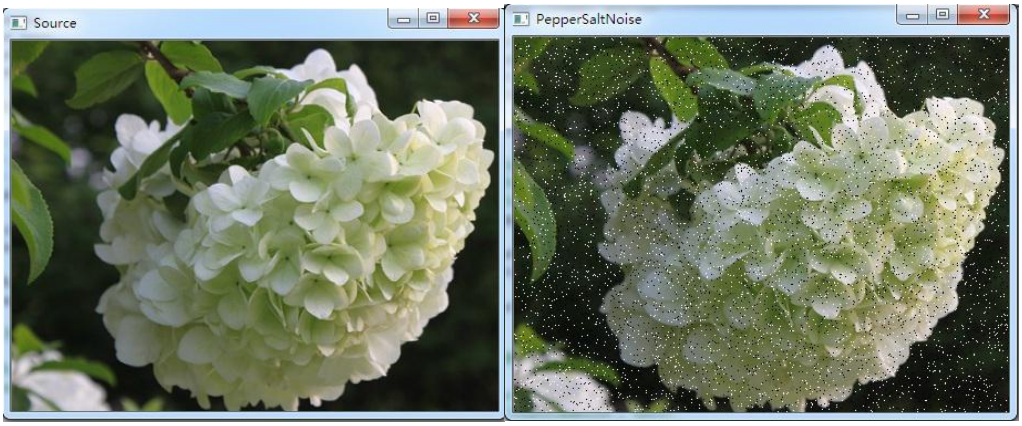
实验结果如下图（以椒盐噪声为例）：

利用算术均值滤波器：


Source
PepperSaltNoise


ArithmeticMeanFilter

利用几何均值滤波器：


Source
PepperSaltNoise

	
实验中的问题	实验问题：几何均值滤波以及谐波、逆谐波滤波没有对应的库函数 解决方法：通过学习书本对应章节，根据公式写出程序
实验结果	