实验三 模板操作

一、实验目的

- 熟悉像素的邻域定义和像素间的联系，即邻接、连接、连通的概念。
- 熟悉模板运算的基本概念。
- 理解不同空间滤波技术的基本原理，掌握算法实现

二、实验任务

- 分别使用3*3的邻域平均模板和高斯平均模板对灰度图yuv进行平滑滤波。
- 分别使用拉普拉斯算子和sobel算子对灰度图yuv进行锐化滤波,并实现图像增强。

三、实验材料

两个YUV格式图像文件，采样格式均为4:4:4，其图像参数如下：

文件名	宽Width	高Height
lena_noise.yuv	512	512
moon.yuv	464	538

四、实验步骤

- 基本的文件打开和读取操作。
- 定义二维数组存储图像的灰度信息，便于与模板进行运算。
- 逐像素与3*3的邻域平均模板进行运算，结果作为新的文件输出。平滑滤波重复进行2次、3次，对比滤波效果
- 使用播放器查看新的文件是否达到预期效果。
- 与高斯平均模板、拉普拉斯算子和sobel算子的运算重复上述步骤即可。其中高斯平均模板为

可采用的拉普拉斯算子分别有如下四种：

sobel算子如下，

五、结果分析

结合实验结果，对不同滤波器的理论和性能进行比较和分析，并写入在实验报告。

> ❓ *分别使用3*3的邻域平均模板和高斯平均模板对灰度图lena_noise.yuv进行平滑滤波。逐像素与3*3的邻域平均模板进行运算，结果作为新的文件输出。平滑滤波重复进行2次、3次，对比滤波效果

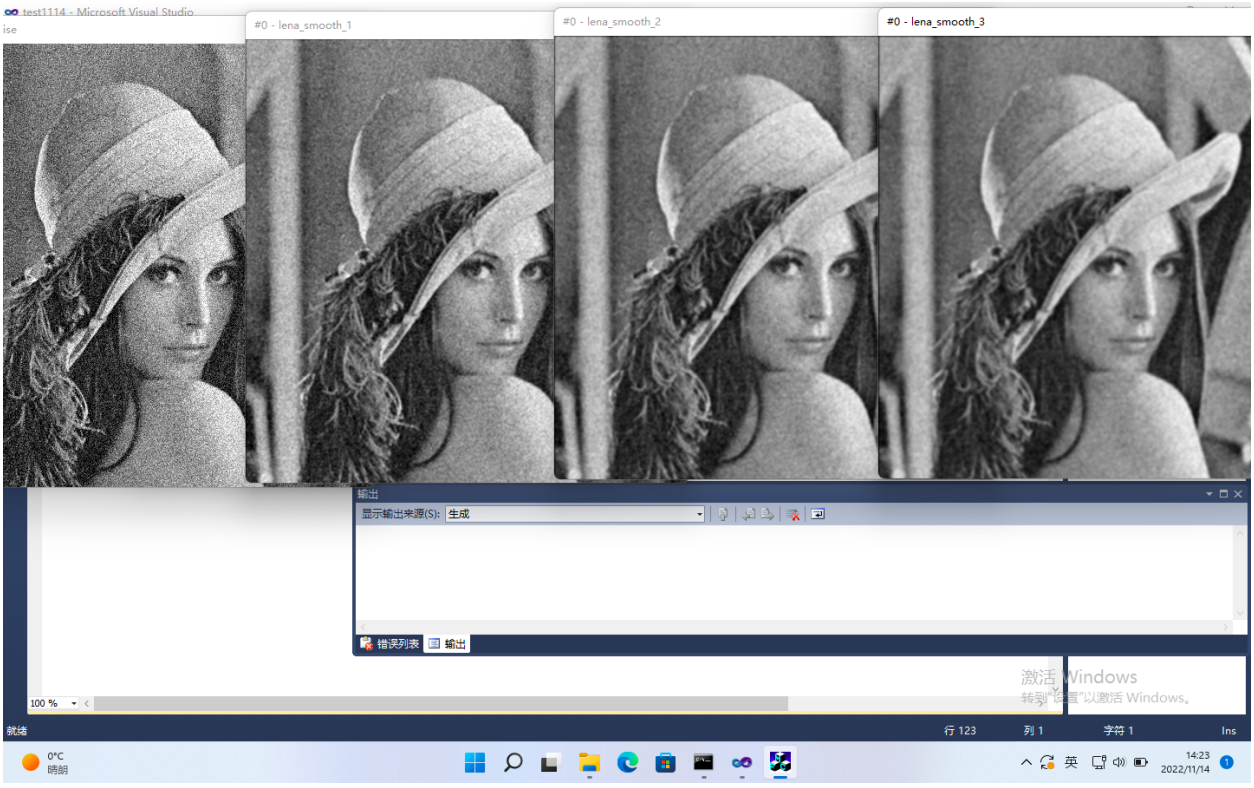
（1）3*3的邻域平均模板对灰度图lena_noise.yuv进行平滑滤波

第一次平滑（第二三次代码同理，此处不展示）

```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdio.h>
#include <stdlib.h>

//模板卷积
int Mask(unsigned char* ini,int i,int j, int  height, int width,int* w,int order)
{
```

```
  int sum = 0;
  int ini_x = i - 1;//(ini_x, ini_y), (i, j) 图像中3*3区域中心点（进行卷积的点）
  int ini_y = j - 1;
  for (int m = 0; m < order; m++)//3*3模板
  {
    for (int n = 0; n < order; n++)
    {
      if ((ini_x + m < 0) || (ini_y + n < 0) || (ini_x + m + 1 > height) || (ini_y + n + 1 > width))
        sum += 0;   //边界置零
      else
        sum += ini[(ini_x + m) * width + ini_y + n] * w[m * order + n]; //图像中3*3区域 与 模板矩阵中逐个元素相乘  ini_x + m 第某行第m； w[m * orde
      //width--宽的像素个数；height--高的像素个数
    }
  }
  if (sum>=0)
    return sum;
  else
    return abs(sum);
}

int main()
{
  //定义各个模板的数值
  int Smooth[9]={1,1,1,1,1,1,1,1,1};
  int Gausian[9]={1,2,1,2,4,2,1,2,1};
  int Laplace[9]={1,1,1,1,-8,1,1,1,1};
  int Sobel[9]={-1,-2,-1,0,0,0,1,2,1};

  int lena_width = 512;
  int lena_height = 512;
  int moon_width = 464;
  int moon_height = 538;

  int lena_size = lena_height * lena_width * 3;
  int moon_size = moon_height * moon_width * 3;

  FILE *fin;
  fin = fopen("C:\\lena_noise.yuv", "rb");
   if (NULL == fin)
   {
    perror("open file is failed\n");
    return -2;
   }

   unsigned char *lena = (unsigned char *)malloc(lena_size);           // 转换后数据存放的位置
   if (NULL == lena)
   {
    fprintf(stderr, "malloc data failed\n");
    fclose(fin);
    return -3;
   }

    /* 配置 y u v 数据位置 */
  unsigned char *y = lena;      // y数据存放的位置
  unsigned char *u = lena+lena_width*lena_height;       // u数据存放的位置
  unsigned char *v = u+lena_width*lena_height;       // v数据存放的位置
  unsigned char *buff =(unsigned char *)malloc(lena_size);//设置缓存区

  if (NULL ==lena)
  {
    fprintf(stderr, "malloc buff failed\n");
    fclose(fin);
    return -4;
  }
    //printf("hi!\n");




int i = 0, j = 0;
fread(buff, 1, lena_size, fin);//读取fin，每次读取一个字节，读取lena_size个项，最终放入lena中


for (i = 0; i < lena_height; i++)
{
  for (j = 0; j < lena_width; j++)
  {

    lena[i * lena_width + j] = Mask(buff, i, j, lena_height, lena_width, Smooth, 3) / 9;
```

```
  }

}


//将u和v分量直接置为128
  for (i = 0; i < lena_height; i += 1)
  {
    for (j = 0; j< lena_width ; j += 1)
    {
      *u++ = 128;   /* u0 */
      *v++ = 128;   /* v0 */

    }
  }
printf("good\n");

 FILE *fout;
 fout = fopen("C:\\test\\output\\lena_smooth_ 1.yuv", "wb");

 if (NULL == fout)
 {
  perror("open newfile is failed\n");
  return -3;
 }

 fwrite(lena, 1,lena_size, fout);//将lena中lena_size写入fout
 fclose(fout);

 return 0;
}
```

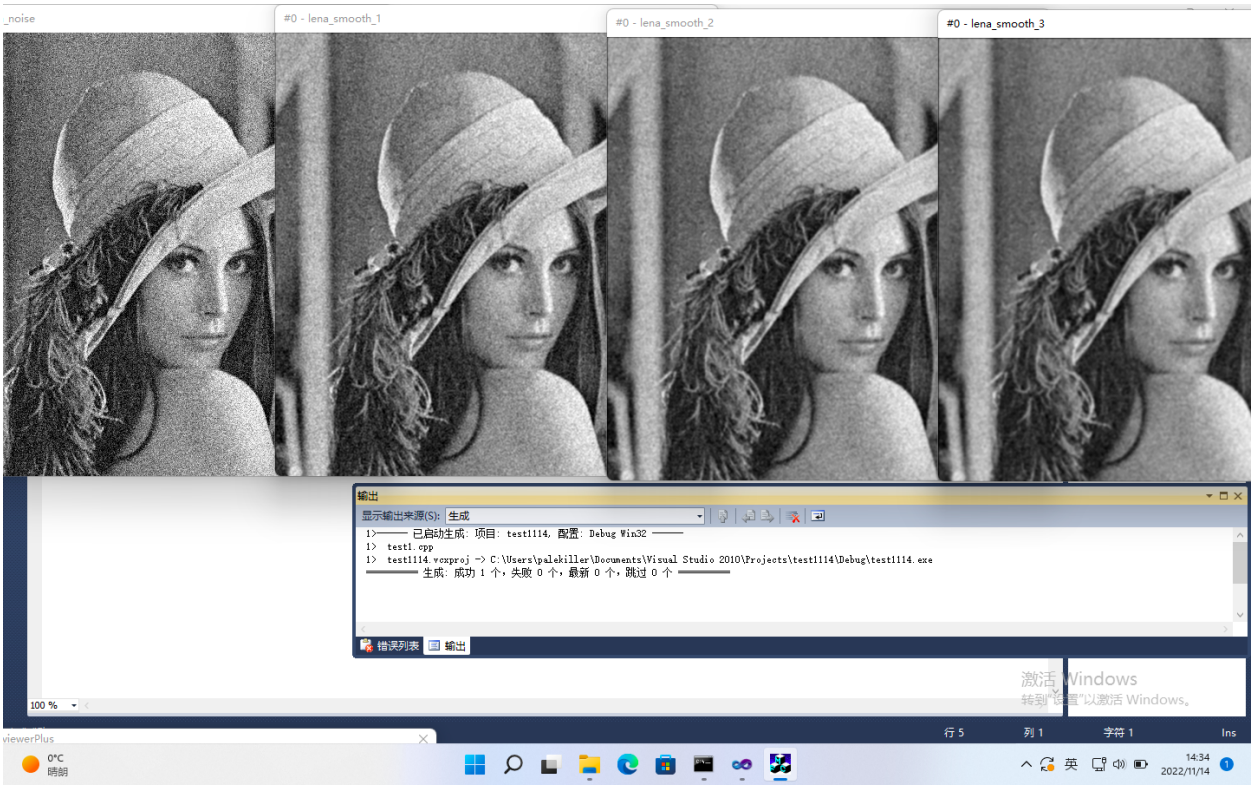


（2）3*3的高斯平均模板对灰度图lena_noise.yuv进行平滑滤波

第一次平滑（第二三次代码同理，此处不展示）

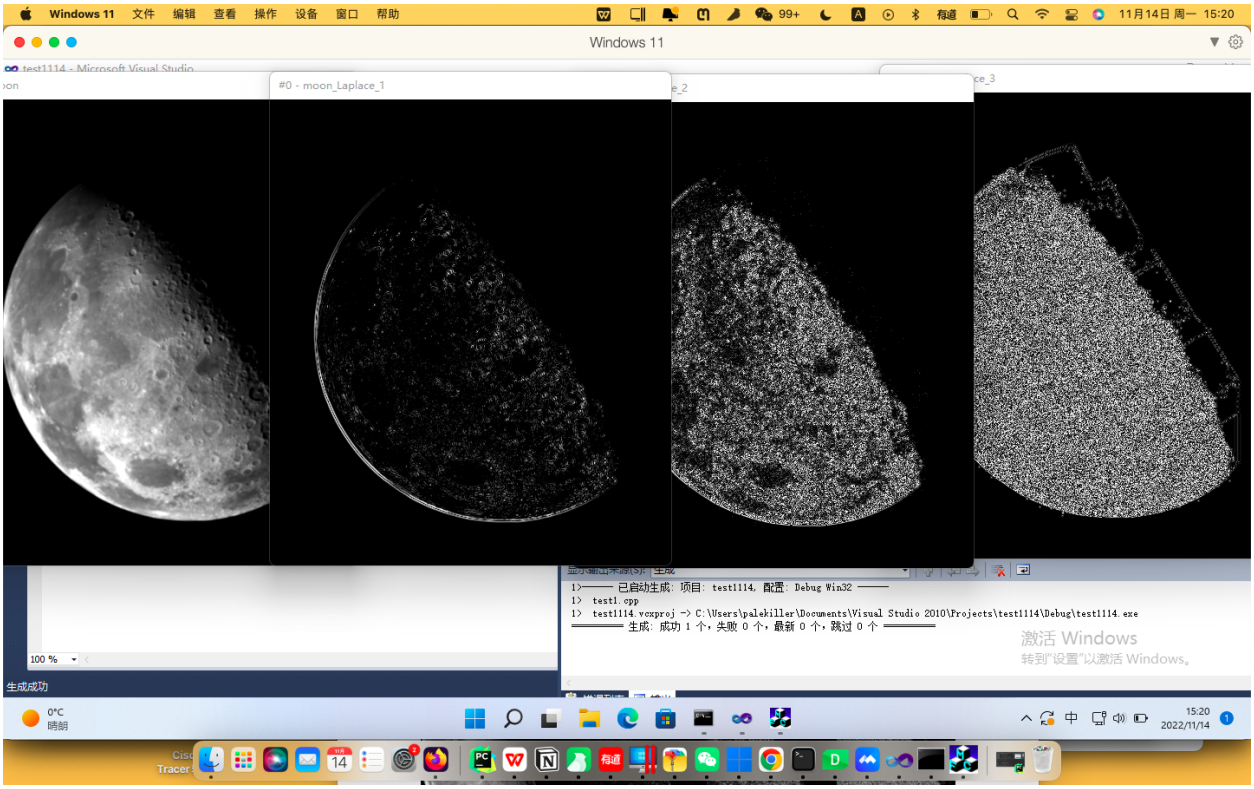
```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdio.h>
#include <stdlib.h>

//模板卷积
int Mask(unsigned char* ini,int i,int j, int  height, int width,int* w,int order)
{
  int sum = 0;
  int ini_x = i - 1;//(ini_x, ini_y), （i，j）图像中3*3区域中心点（进行卷积的点）
  int ini_y = j - 1;
  for (int m = 0; m < order; m++)//3*3模板
  {
    for (int n = 0; n < order; n++)
    {
      if ((ini_x + m < 0) || (ini_y + n < 0) || (ini_x + m + 1 > height) || (ini_y + n + 1 > width))
        sum += 0;   //边界置零
      else
        sum += ini[(ini_x + m) * width + ini_y + n] * w[m * order + n]; //图像中3*3区域 与 模板矩阵中逐个元素相乘  ini_x + m 第某行第m； w[m * orde
      //width--宽的像素个数；height--高的像素个数
    }
  }
  if (sum>=0)
    return sum;
  else
    return abs(sum);
}

int main()
{
  //定义各个模板的数值
  int Smooth[9]={1,1,1,1,1,1,1,1,1};
  int Gausian[9]={1,2,1,2,4,2,1,2,1};
  int Laplace[9]={1,1,1,1,-8,1,1,1,1};
  int Sobel[9]={-1,-2,-1,0,0,0,1,2,1};

  int lena_width = 512;
  int lena_height = 512;
  int moon_width = 464;
  int moon_height = 538;

  int lena_size = lena_height * lena_width * 3;
  int moon_size = moon_height * moon_width * 3;

  FILE *fin;
  fin = fopen("C:\\test\\output\\lena_smooth_2.yuv", "rb");
   if (NULL == fin)
   {
    perror("open file is failed\n");
    return -2;
   }

   unsigned char *lena = (unsigned char *)malloc(lena_size);           // 转换后数据存放的位置
   if (NULL == lena)
   {
    fprintf(stderr, "malloc data failed\n");
    fclose(fin);
    return -3;
   }

    /* 配置 y u v 数据位置 */
  unsigned char *y = lena;      // y数据存放的位置
  unsigned char *u = lena+lena_width*lena_height;       // u数据存放的位置
  unsigned char *v = u+lena_width*lena_height;       // v数据存放的位置
  unsigned char *buff =(unsigned char *)malloc(lena_size);//设置缓存区

  if (NULL ==lena)
  {
    fprintf(stderr, "malloc buff failed\n");
    fclose(fin);
    return -4;
  }
    //printf("hi!\n");
```

```
int i = 0, j = 0;
fread(buff, 1, lena_size, fin);//读取fin，每次读取一个字节，读取lena_size个项，最终放入lena中


for (i = 0; i < lena_height; i++)
{
  for (j = 0; j < lena_width; j++)
  {

    lena[i * lena_width + j] = Mask(buff, i, j, lena_height, lena_width, Gausian, 3) / 16;



  }

}


//将u和v分量直接置为128
  for (i = 0; i < lena_height; i += 1)
  {
    for (j = 0; j< lena_width ; j += 1)
    {
      *u++ = 128;   /* u0 */
      *v++ = 128;   /* v0 */

    }
  }
printf("good\n");

 FILE *fout;
 fout = fopen("C:\\test\\output\\lena_smooth_3.yuv", "wb");

 if (NULL == fout)
 {
  perror("open newfile is failed\n");
  return -3;
 }

 fwrite(lena, 1,lena_size, fout);//将lena中lena_size写入fout
 fclose(fout);

 return 0;
}
```



❓ 分别使用拉普拉斯算子和sobel算子对灰度图moon.yuv进行锐化滤波,并实现图像增强

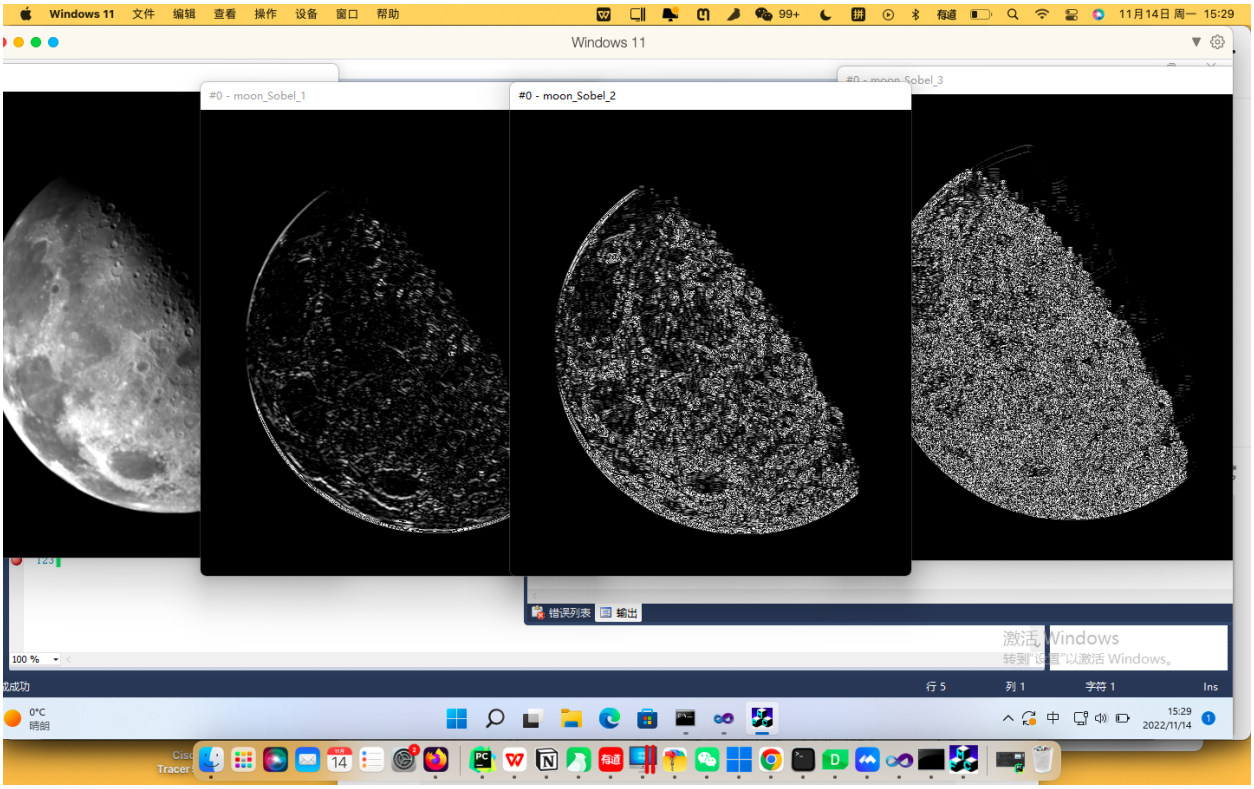
拉普拉斯算子




```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdio.h>
#include <stdlib.h>

//模板卷积
int Mask(unsigned char* ini,int i,int j, int  height, int width,int* w,int order)
{
  int sum = 0;
  int ini_x = i - 1;//(ini_x, ini_y), (i, j) 图像中3*3区域中心点 (进行卷积的点)
  int ini_y = j - 1;
  for (int m = 0; m < order; m++)//3*3模板
  {
    for (int n = 0; n < order; n++)
    {
      if ((ini_x + m < 0) || (ini_y + n < 0) || (ini_x + m + 1 > height) || (ini_y + n + 1 > width))
        sum += 0;   //边界置零
      else
        sum += ini[(ini_x + m) * width + ini_y + n] * w[m * order + n]; //图像中3*3区域 与 模板矩阵中逐个元素相乘  ini_x + m 第某行第m； w[m * orde
      //width--宽的像素个数；height--高的像素个数
    }
  }
  if (sum>=0)
    return sum;
  else
    return abs(sum);
}

int main()
{
  //定义各个模板的数值
  int Smooth[9]={1,1,1,1,1,1,1,1,1};
  int Gausian[9]={1,2,1,2,4,2,1,2,1};
  int Laplace[9]={1,1,1,1,-8,1,1,1,1};
  int Sobel[9]={-1,-2,-1,0,0,0,1,2,1};

  int lena_width = 512;
  int lena_height = 512;
  int moon_width = 464;
  int moon_height = 538;

  int lena_size = lena_height * lena_width * 3;
```

```
  int moon_size = moon_height * moon_width * 3;

  FILE *fin;
  fin = fopen("C:\\moon.yuv", "rb");
   if (NULL == fin)
   {
    perror("open file is failed\n");
    return -2;
   }

   unsigned char *moon = (unsigned char *)malloc(moon_size);           // 转换后数据存放的位置
   if (NULL == moon)
   {
    fprintf(stderr, "malloc data failed\n");
    fclose(fin);
    return -3;
   }

    /* 配置 y u v 数据位置 */
  unsigned char *y = moon;      // y数据存放的位置
  unsigned char *u = moon+moon_width*moon_height;       // u数据存放的位置
  unsigned char *v = u+moon_width*moon_height;       // v数据存放的位置
  unsigned char *buff =(unsigned char *)malloc(lena_size);//设置缓存区

  if (NULL ==moon)
  {
    fprintf(stderr, "malloc buff failed\n");
    fclose(fin);
    return -4;
  }
    //printf("hi!\n");




int i = 0, j = 0;
fread(buff, 1, moon_size, fin);//读取fin，每次读取一个字节，读取lena_size个项，最终放入lena中


for (i = 0; i <moon_height; i++)
{
  for (j = 0; j < moon_width; j++)
  {

    moon[i * moon_width + j] = Mask(buff, i, j,moon_height, moon_width,  Laplace, 3);



  }

}


//将u和v分量直接置为128
  for (i = 0; i <moon_height; i += 1)
  {
    for (j = 0; j<moon_width ; j += 1)
    {
      *u++ = 128;   /* u0 */
      *v++ = 128;   /* v0 */

    }
  }
printf("good\n");

 FILE *fout;
 fout = fopen("C:\\test\\output\\moon_Laplace_1.yuv", "wb");

 if (NULL == fout)
 {
  perror("open newfile is failed\n");
  return -3;
 }

 fwrite(moon, 1,moon_size, fout);//将lena中lena_size写入fout
 fclose(fout);

 return 0;
}
```

sobel算子




```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdio.h>
#include <stdlib.h>

//模板卷积
int Mask(unsigned char* ini,int i,int j, int  height, int width,int* w,int order)
{
  int sum = 0;
  int ini_x = i - 1;//(ini_x, ini_y), (i, j) 图像中3*3区域中心点（进行卷积的点）
  int ini_y = j - 1;
  for (int m = 0; m < order; m++)//3*3模板
  {
    for (int n = 0; n < order; n++)
    {
      if ((ini_x + m < 0) || (ini_y + n < 0) || (ini_x + m + 1 > height) || (ini_y + n + 1 > width))
        sum += 0;   //边界置零
      else
        sum += ini[(ini_x + m) * width + ini_y + n] * w[m * order + n]; //图像中3*3区域 与 模板矩阵中逐个元素相乘  ini_x + m 第某行第m； w[m * orde
      //width--宽的像素个数；height--高的像素个数
    }
  }
  if (sum>=0)
    return sum;
  else
    return abs(sum);
}

int main()
{
  //定义各个模板的数值
  int Smooth[9]={1,1,1,1,1,1,1,1,1};
  int Gausian[9]={1,2,1,2,4,2,1,2,1};
  int Laplace[9]={1,1,1,1,-8,1,1,1,1};
  int Sobel[9]={-1,-2,-1,0,0,0,1,2,1};

  int lena_width = 512;
```

```
  int lena_height = 512;
  int moon_width = 464;
  int moon_height = 538;

  int lena_size = lena_height * lena_width * 3;
  int moon_size = moon_height * moon_width * 3;

  FILE *fin;
  fin = fopen("C:\\moon.yuv", "rb");
   if (NULL == fin)
   {
    perror("open file is failed\n");
    return -2;
   }

   unsigned char *moon = (unsigned char *)malloc(moon_size);           // 转换后数据存放的位置
   if (NULL == moon)
   {
    fprintf(stderr, "malloc data failed\n");
    fclose(fin);
    return -3;
   }

    /* 配置 y u v 数据位置 */
  unsigned char *y = moon;      // y数据存放的位置
  unsigned char *u = moon+moon_width*moon_height;       // u数据存放的位置
  unsigned char *v = u+moon_width*moon_height;       // v数据存放的位置
  unsigned char *buff =(unsigned char *)malloc(lena_size);//设置缓存区

  if (NULL ==moon)
  {
    fprintf(stderr, "malloc buff failed\n");
    fclose(fin);
    return -4;
  }
    //printf("hi!\n");




int i = 0, j = 0;
fread(buff, 1, moon_size, fin);//读取fin，每次读取一个字节，读取lena_size个项，最终放入lena中


for (i = 0; i <moon_height; i++)
{
  for (j = 0; j < moon_width; j++)
  {

    moon[i * moon_width + j] = Mask(buff, i, j,moon_height, moon_width,  Sobel, 3);



  }

}


//将u和v分量直接置为128
  for (i = 0; i <moon_height; i += 1)
  {
    for (j = 0; j<moon_width ; j += 1)
    {
      *u++ = 128;   /* u0 */
      *v++ = 128;   /* v0 */

    }
  }
printf("good\n");

 FILE *fout;
 fout = fopen("C:\\test\\output\\moon_Sobel_1.yuv", "wb");

 if (NULL == fout)
 {
  perror("open newfile is failed\n");
  return -3;
 }
```

```
 fwrite(moon, 1,moon_size, fout);//将lena中lena_size写入fout
 fclose(fout);

 return 0;
}
```