**南京理工大学计算机科学与工程学院**

**数字图像处理 报告**

**指导教师 XXX**

**学生姓名 XX 学生学号 XXXXXXXXXXXX**

目 录

[1 项目简介 2](#_Toc101713180)

[2 算法介绍 2](#_Toc101713181)

[2.1 高斯滤波 2](#_Toc101713182)

[2.2 拉普拉斯锐化 7](#_Toc101713186)

[2.3 分离傅里叶变换 7](#_Toc101713187)

[2.4 基于Sobel算子的边缘检测 10](#_Toc101713188)

[2.5 直方图均衡化 11](#_Toc101713189)

[2.6 区域生长 12](#_Toc101713190)

[3 运行结果 14](#_Toc101713191)

[3.1 快速高斯滤波 15](#_Toc101713192)

[3.2 拉普拉斯锐化 15](#_Toc101713193)

[3.3 分离傅里叶变换得到频谱 16](#_Toc101713194)

[3.4 分离傅里叶变换得到相位谱 16](#_Toc101713195)

[3.5 基于Sobel算子的边缘检测 17](#_Toc101713196)

[3.6 直方图均衡化 18](#_Toc101713197)

[3.7 智能抠图 19](#_Toc101713198)

# 项目简介

本项目使用QT编写图形界面，调用纯c++编写的数字图像处理算法的可执行文件。项目主要实现了六种算法：高斯滤波（以及快速高斯滤波）、拉普拉斯锐化、直方图均衡化、傅里叶变换和基于Sobel算子的边缘检测、区域生长。

边界填充支持三种填充形式：零填充、复制填充和镜像填充。

项目需要调用opencv库，但只用于图像读取和写入，其余算法自行实现。

# 算法介绍

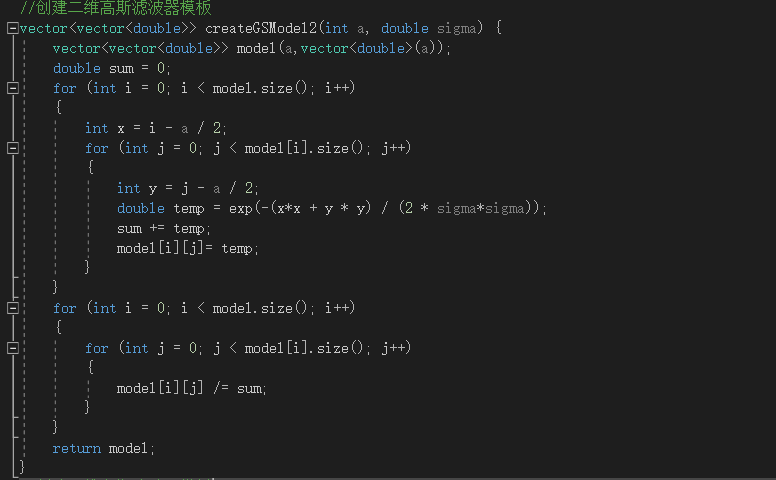
## 高斯滤波

高斯滤波器采用高斯函数作为加权函数。特点如下：

1. 二维高斯函数具有旋转对称性，保证滤波时各方向平滑程度相同。
2. 离中心点越远权值越小，确保边缘细节不被模糊。

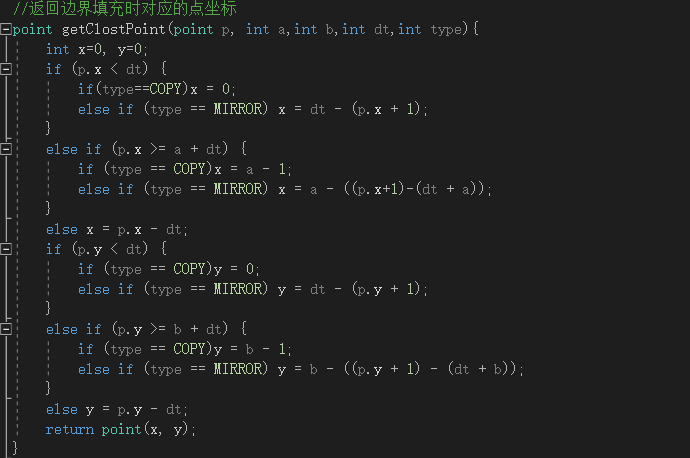
二维高斯分布函数表达式如下：

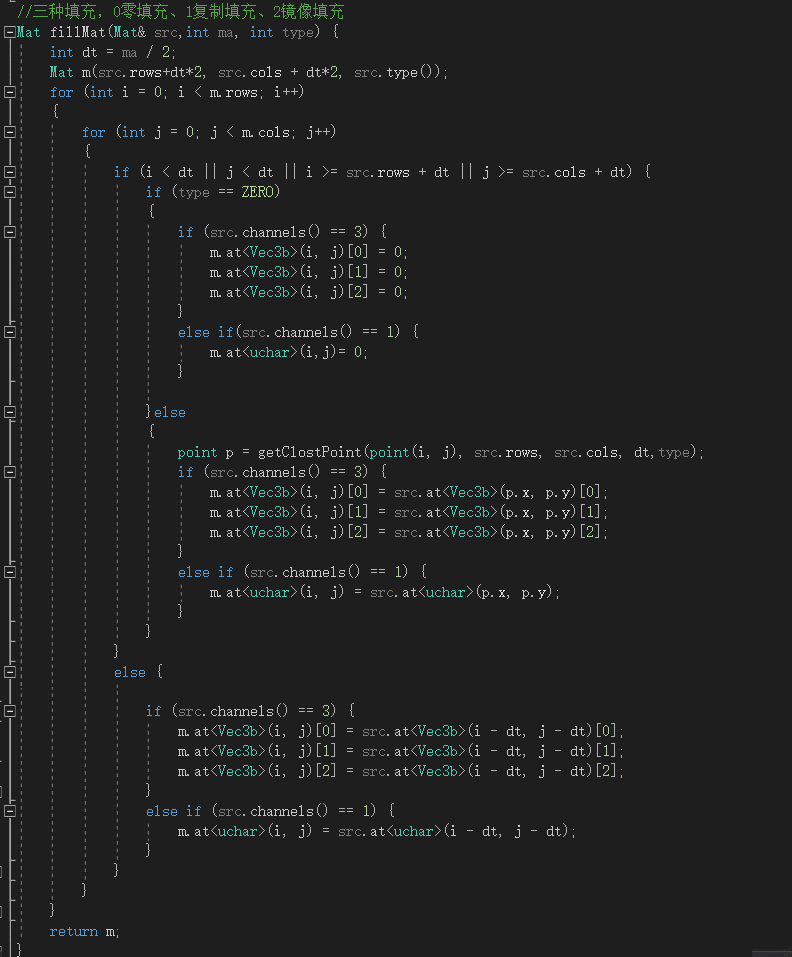


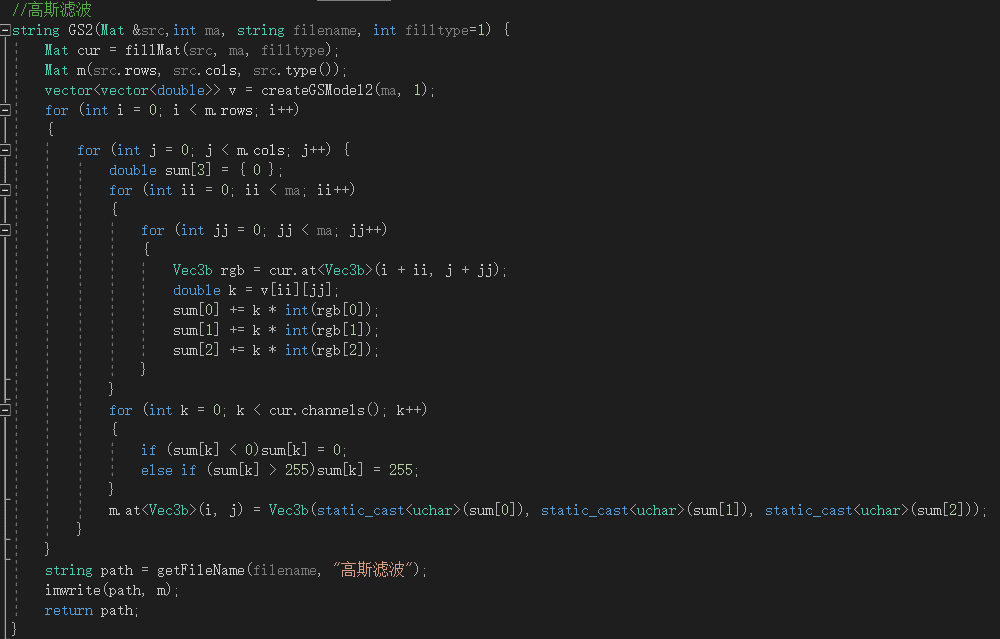


图表 1 二维高斯滤波器代码实现

在生成高斯滤波器时，因为最后需要实现0-255的分布，所以表达式前面的系数可以不用管。利用高斯滤波器进行卷积时，需要先对原图像进行边界填充。





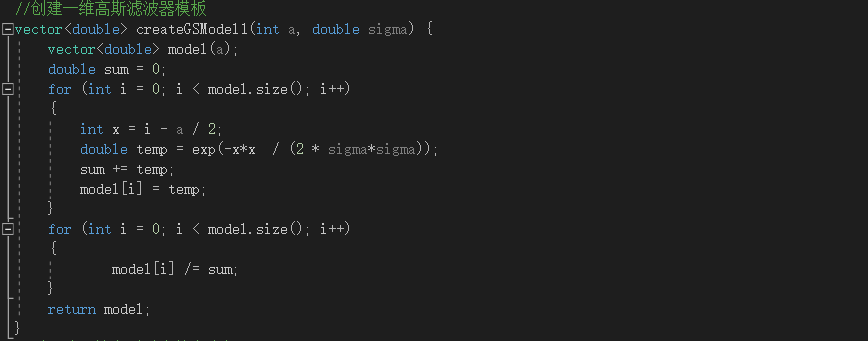
图表 2边界填充代码实现

图表 3 高斯滤波代码实现

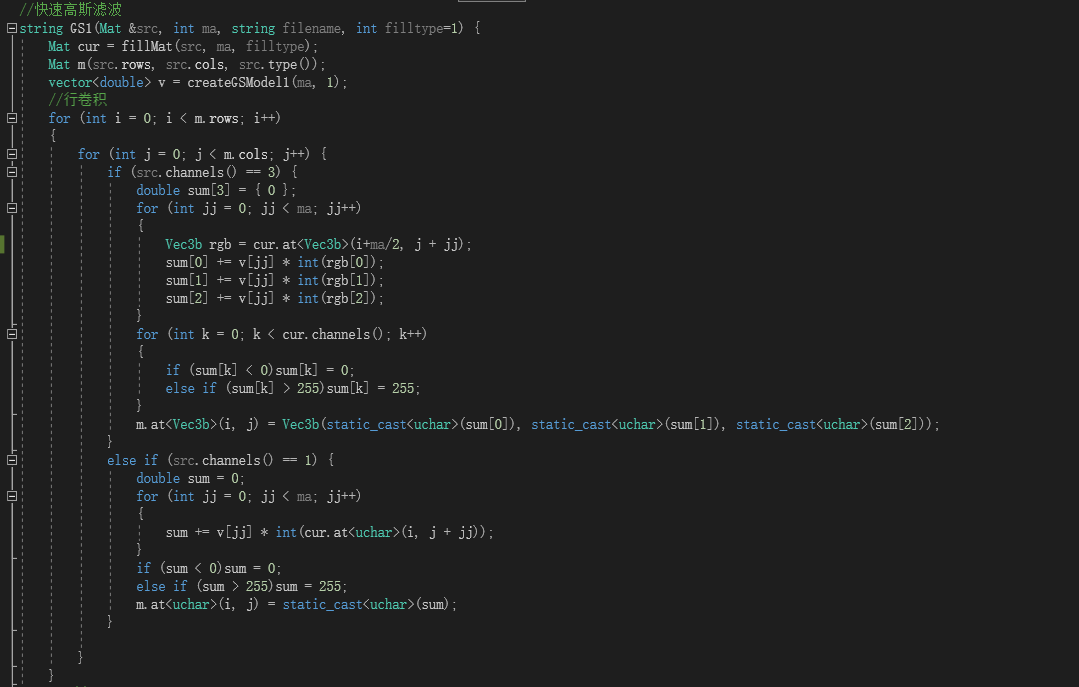
但是，这种算法实现的高斯滤波效率很低，假设输入一张图片大小为n\*n，则时间复杂度为o(n^4)。所以我采用更高效的快速高斯滤波，时间复杂度为o(n^3)。先计算一维高斯核Gx和Gy，再归一化Gx和Gy，使用Gx对图像进行行卷积，使用Gy对行卷积结果进行列卷积。

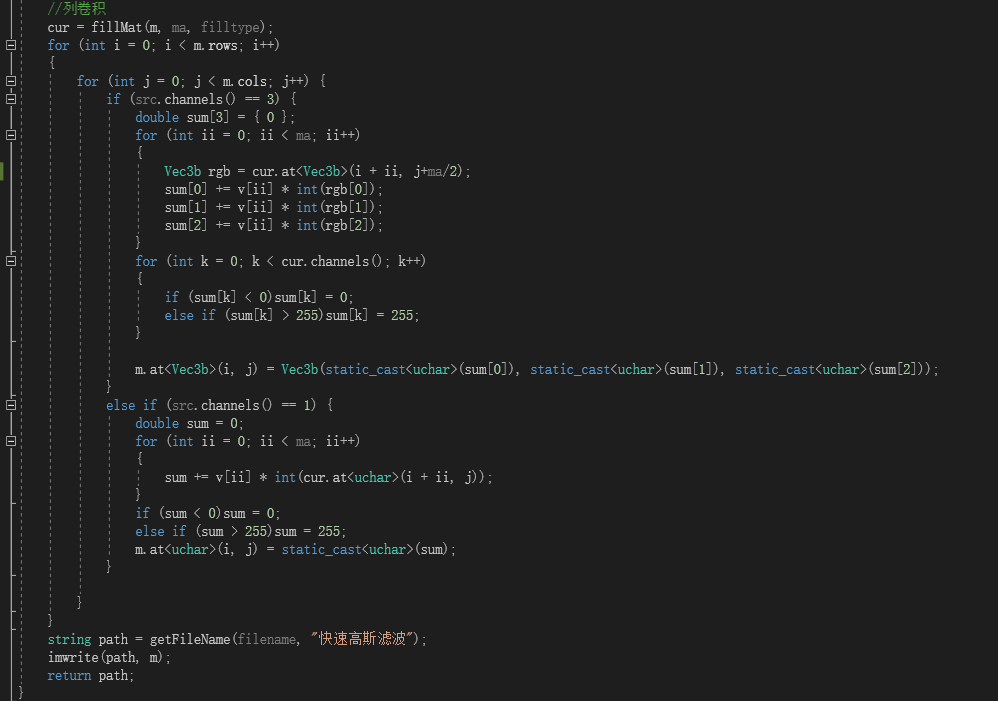






图表 4一维高斯滤波器代码实现





图表 5快速高斯滤波代码实现



## 拉普拉斯锐化

拉普拉斯算子使用二阶导数来锐化图像，与一阶导数相比，可增强更精细的细节。

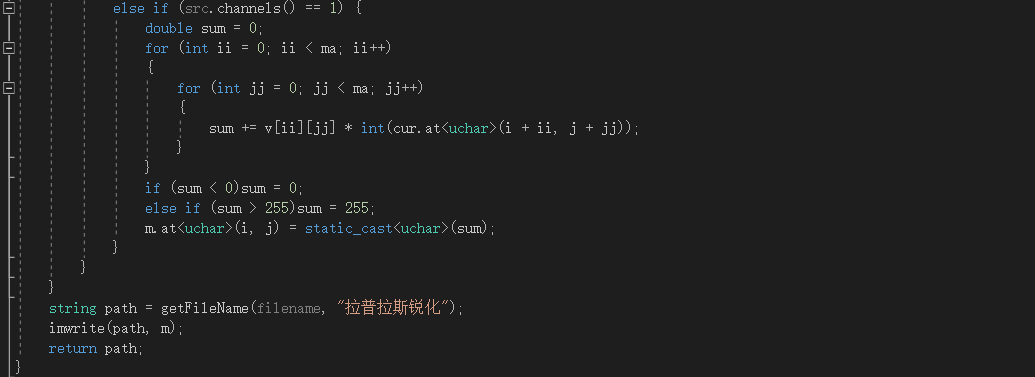
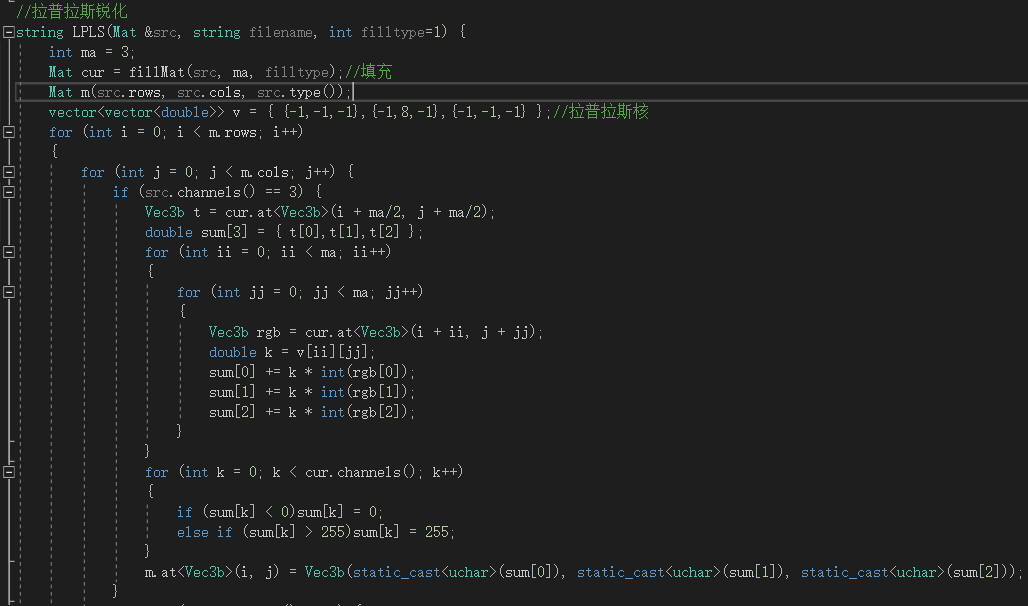




 拉普拉斯核

设f(x,y)和g(x,y)分别是原图和锐化后图像，则使用拉普拉斯锐化图像的基本方法如下：





图表 6拉普拉斯锐化代码实现

## 分离傅里叶变换

傅里叶变换:非周期函数用正弦函数/余弦函数乘以加权函数的积分来表示。

 傅里叶变换

 极坐标表示

 幅度谱

 相位谱

为了加快算法效率，采用分离傅里叶变换，二维DFT可分离为两次一维DFT。给定一幅 M\*N 大小的二维图像 f(x,y)，先按行对列变量 y 做一次长度为 N 的一维DFT，再将计算结果按列对行变量 x 做一次长度为 M 的一维DFT就可以得到该图像的傅里叶变换结果。





根据欧拉公式 (1.13)得：





为了实现频谱中心化，将输入图像乘以,再进行傅里叶变换。

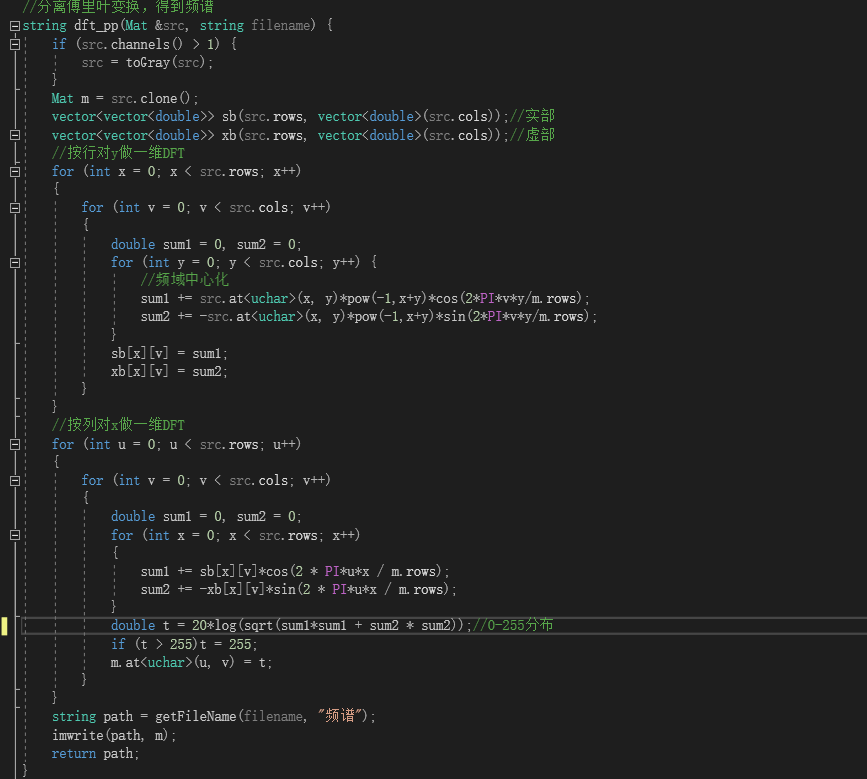
为了将得到的像素点值0-255分布，根据相关资料，

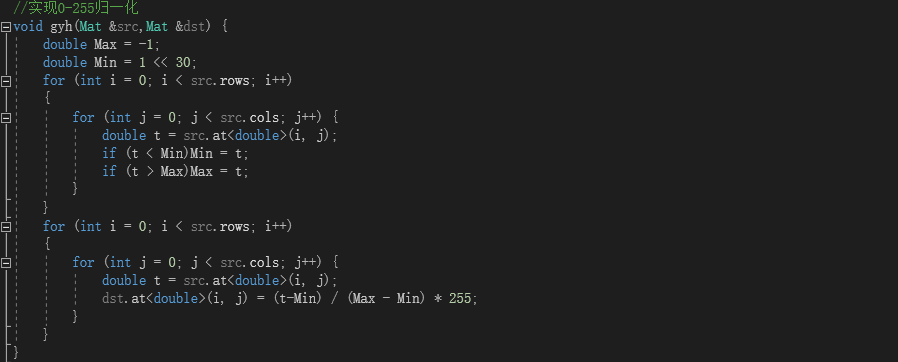
频谱中采用如下方法：



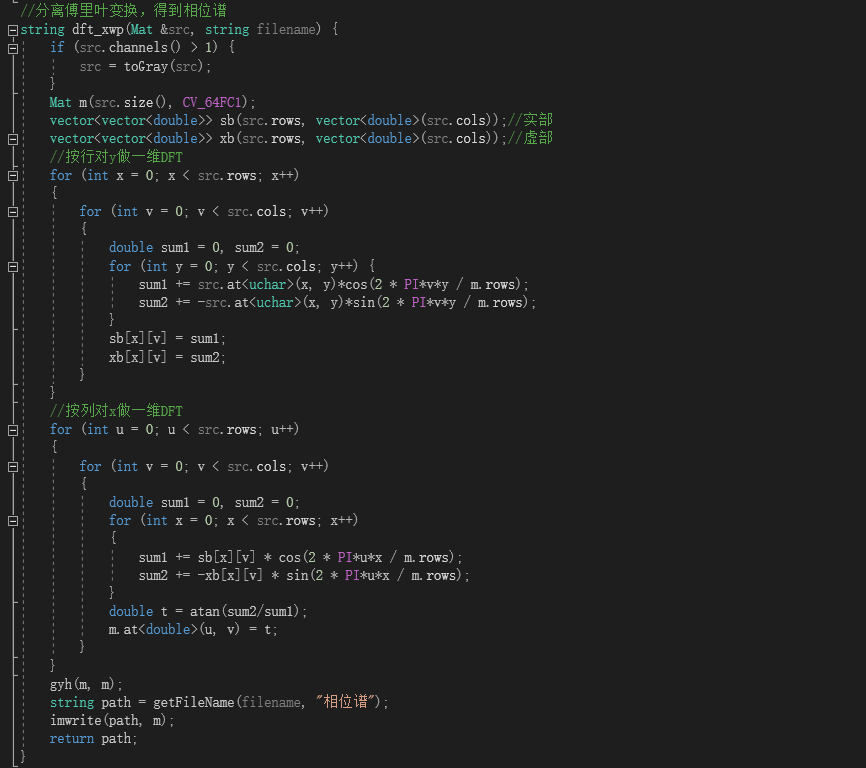
相位谱采用如下方法：





图表 7分离傅里叶变换得到频谱代码实现

图表 8像素点0-255分布代码实现



图表 9分离傅里叶变换得到相位谱

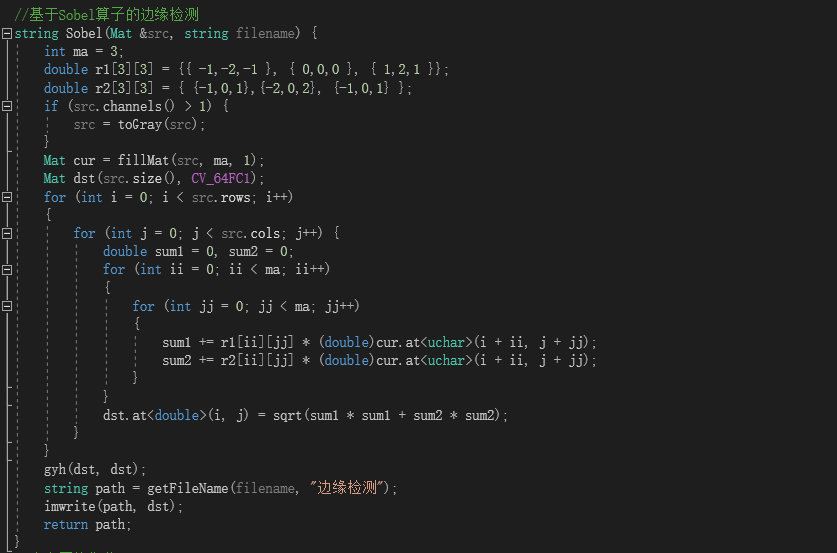
## 基于Sobel算子的边缘检测

基于梯度算子的边缘检测

 梯度

 梯度幅值

 Sobel算子



图表 10基于Sobel算子的边缘检测代码实现

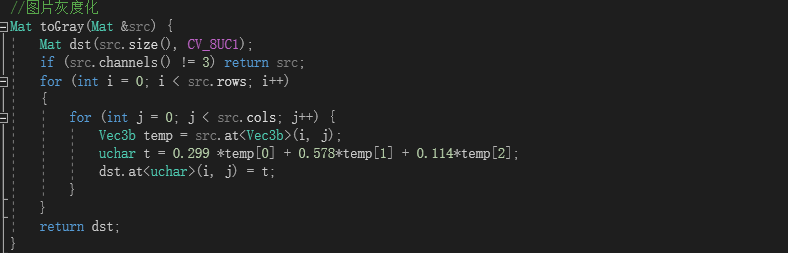
## 直方图均衡化



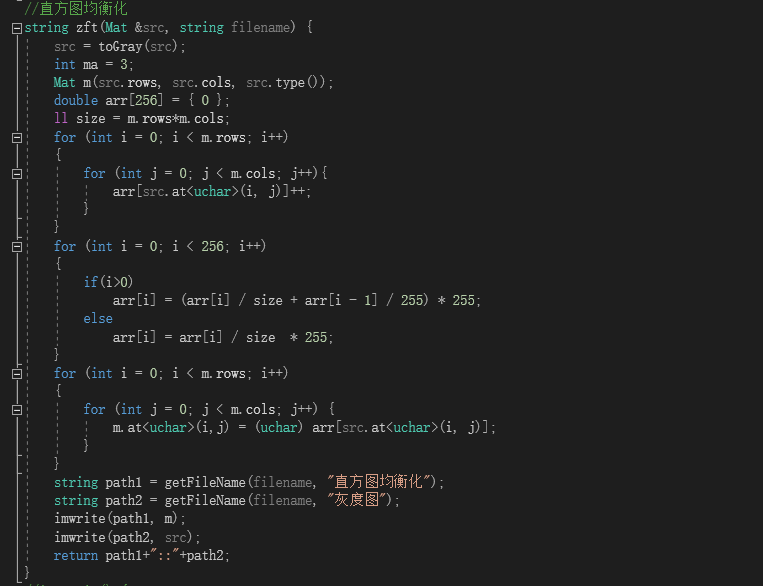
表示输出图像像素值，表示输入图像像素值，L表示图像大小，表示对应概率。

这种方法的直方图均衡化适用于灰度图，需要先将输入图像灰度化。





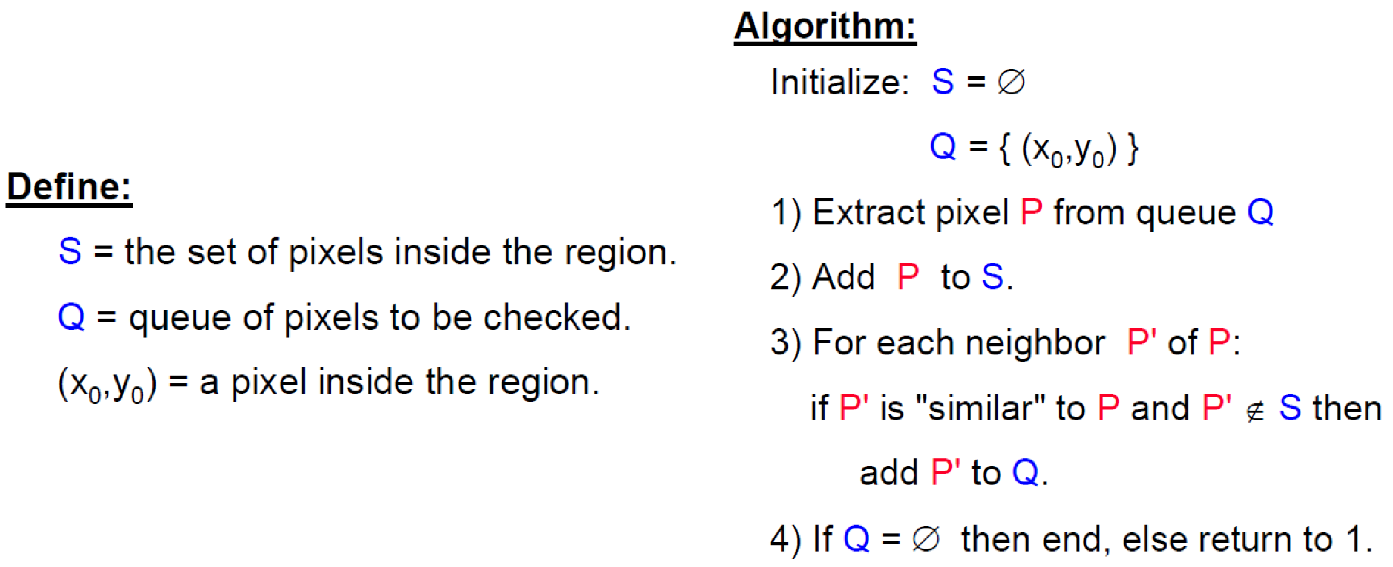
图表 11图像灰度化代码实现



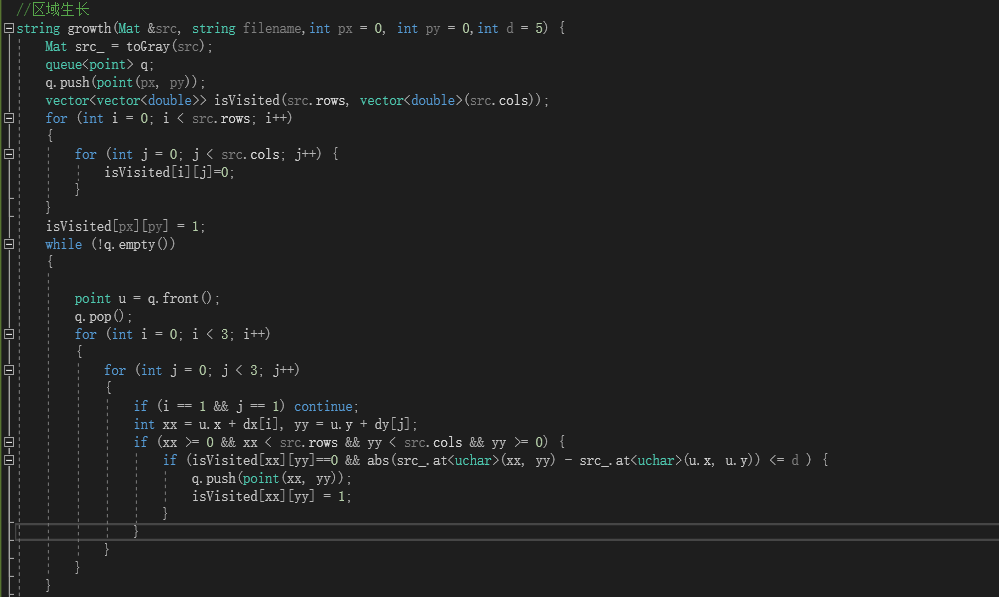
图表 12直方图均衡化代码实现

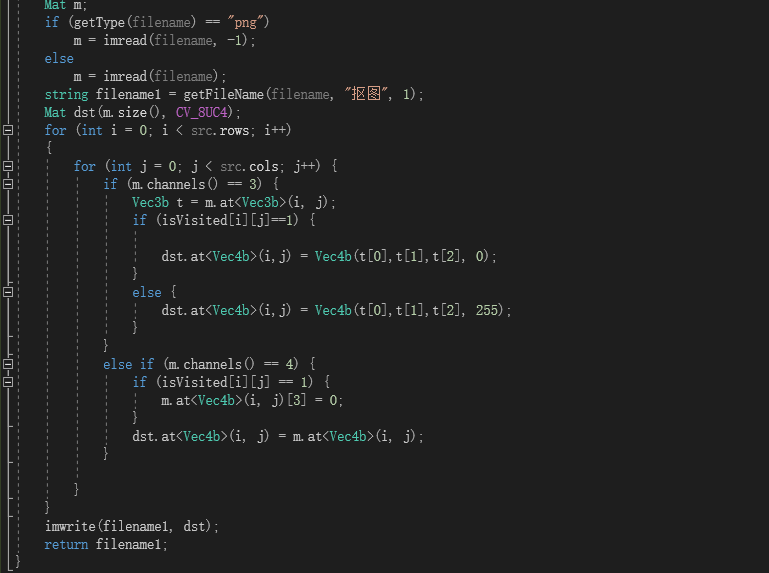
## 区域生长

智能抠图基于边缘检测和区域生长两种算法，先通过边缘检测将非孤立点的像素值置零，再把像素值为零的点作为初始点进行使用广度优先搜索算法区域生长，再将此区域点的透明度置零，从而实现背景透明化。初始点由用户通过交互界面给出，用户可以通过设置误差大小，对抠图效果进行调试。



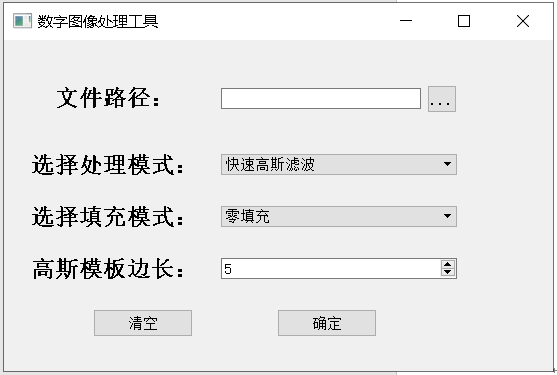
图表 13算法伪代码



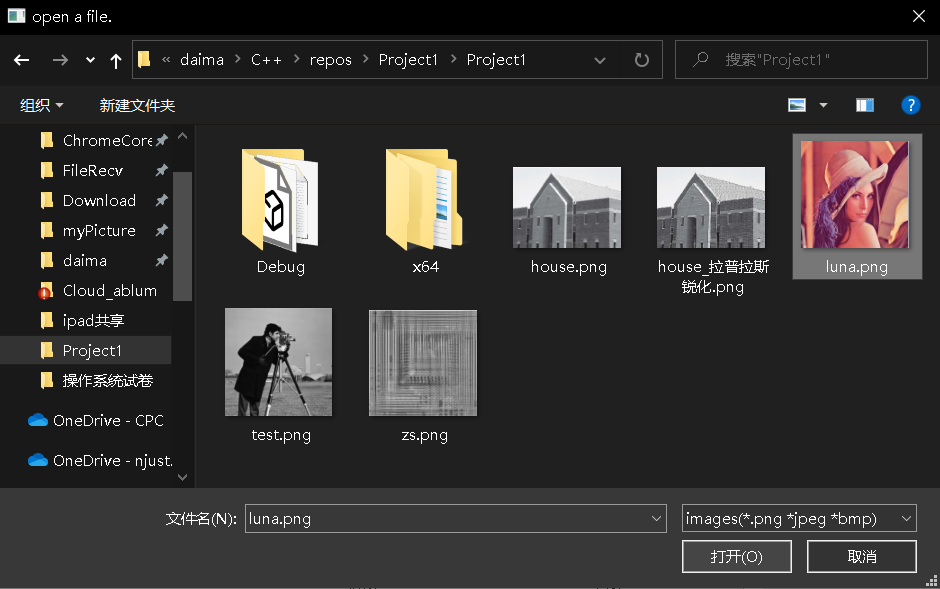


图表 14区域生长代码实现

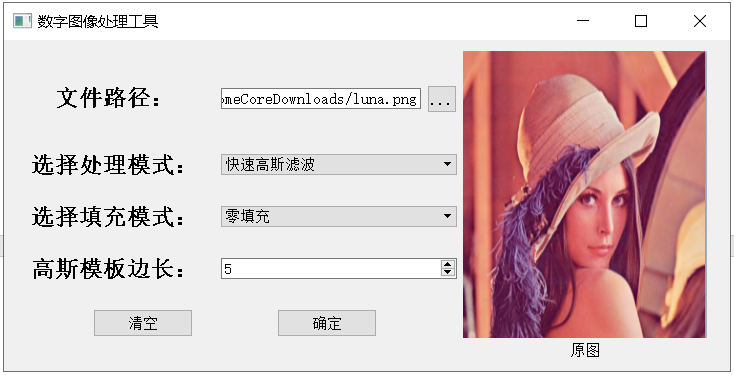
# 运行结果



图表 15开始界面

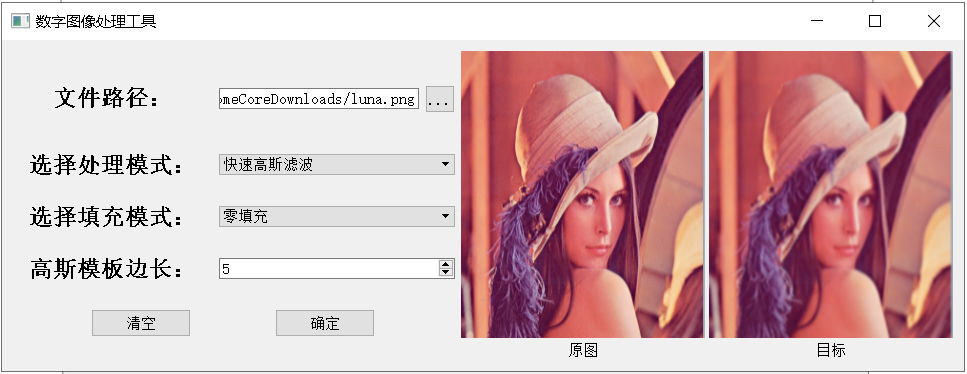


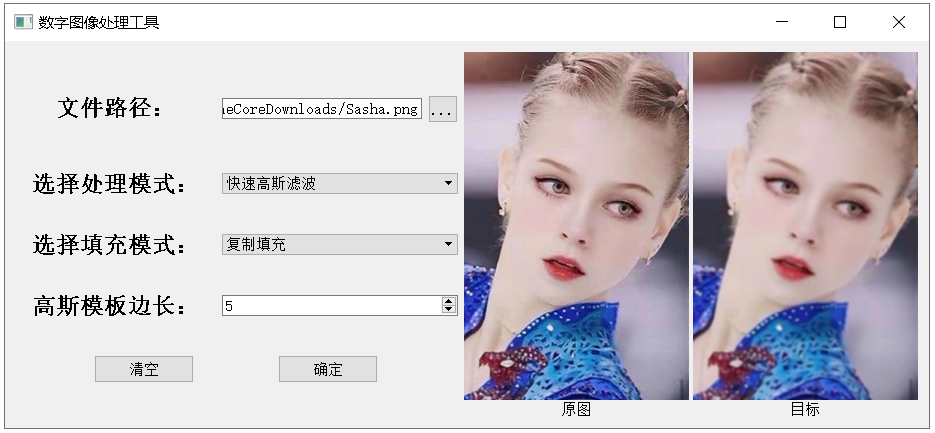
图表 16选择图像文件



图表 17打开图像文件

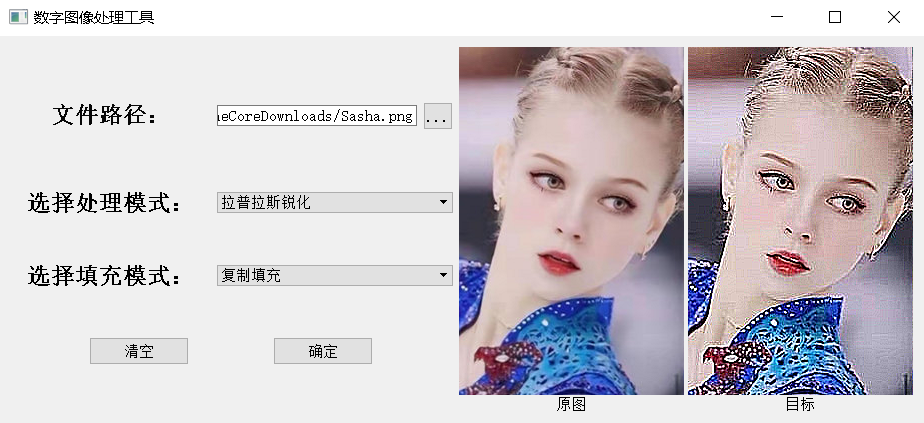
## 快速高斯滤波





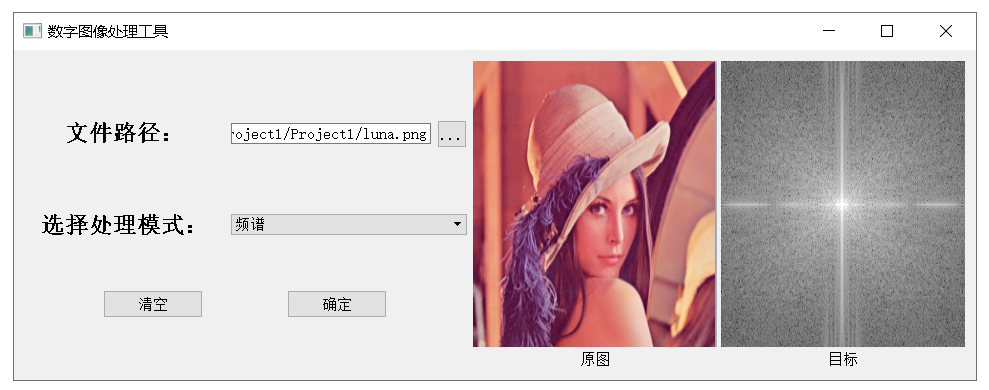
图表 18快速高斯滤波

## 拉普拉斯锐化



图表 19拉普拉斯锐化

## 分离傅里叶变换得到频谱



图表 20分离傅里叶变换得到频谱

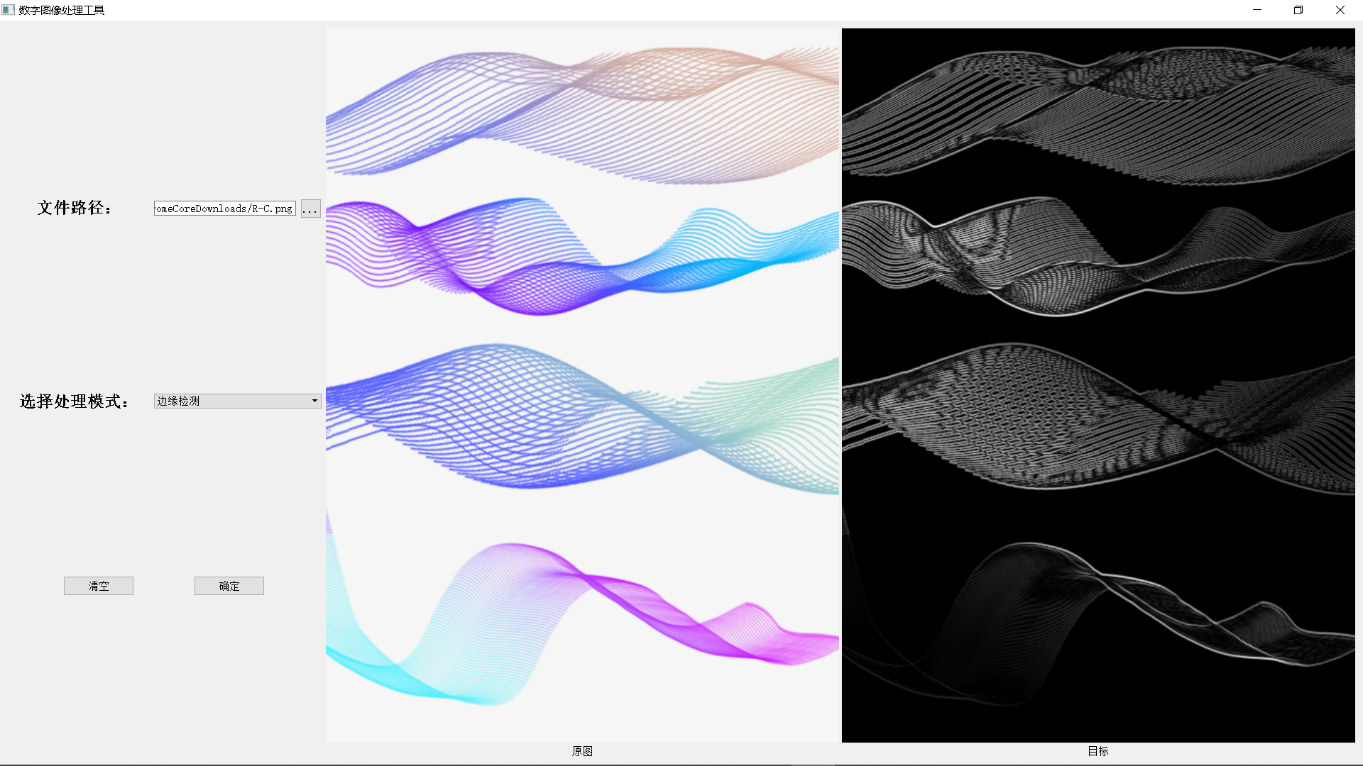
## 分离傅里叶变换得到相位谱



图表 21分离傅里叶变换得到相位谱

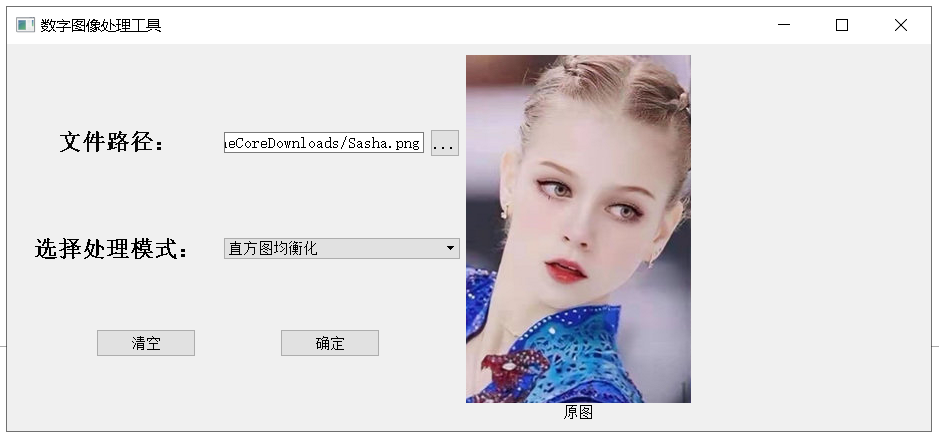
## 基于Sobel算子的边缘检测

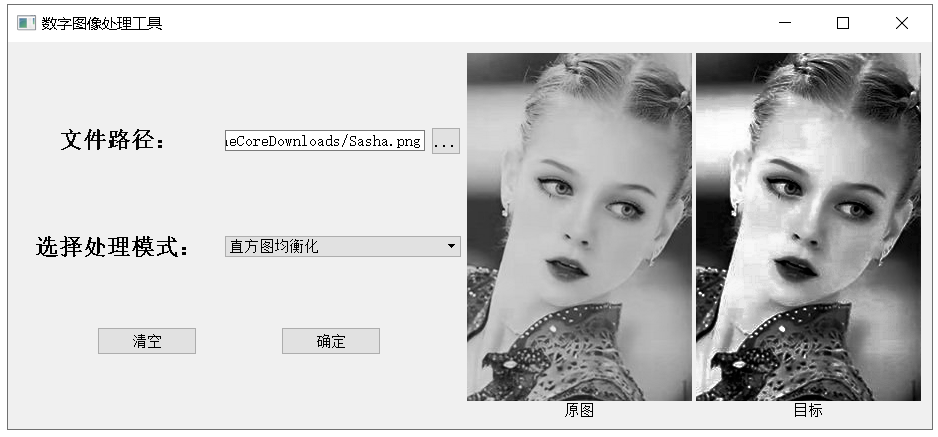




图表 22基于Sobel算子的边缘检测

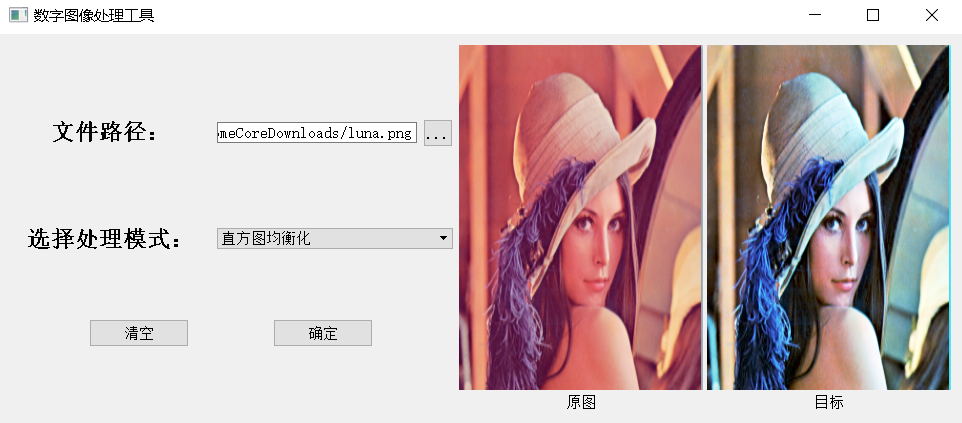
## 直方图均衡化





图表 23直方图均衡化

如果不对彩色图像进行灰度化，把rgb分别进行直方图均衡化，再合并，效果并不好，颜色的失真比较严重的。



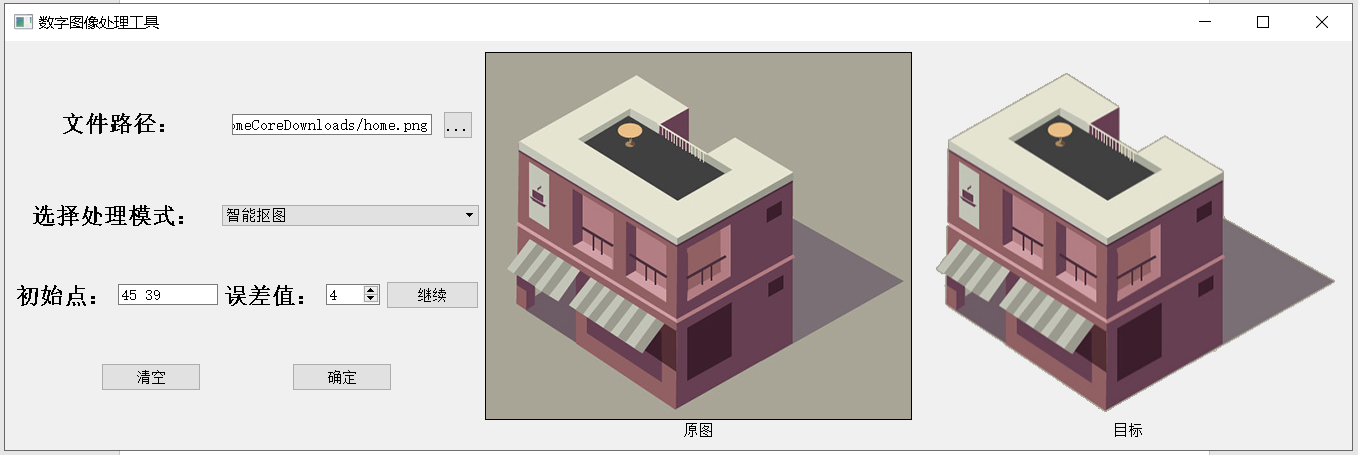
图表 24彩色图像直接进行直方图均衡化

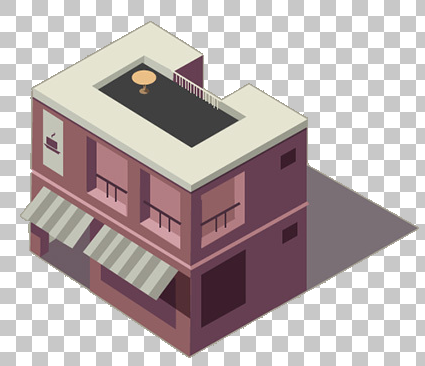
## 智能抠图

1.选择初始点



2.点击确定进行抠图





图表 25智能抠图