华侨大学 Huaqiao university

《面向对象程序设计》

设计报告

题目：BMP 图像处理

院（系）：信息科学与工程学院

专　　业：通信工程

届　　别：2023 级

班　　级：通信工程 1 班

学　　号：2315102026

姓　　名：宋嘉诚

任课老师：彭盛亮

摘要

本课程设要求设计一款 24 位 BMP 图像处理软件，能够对 24 位 BMP 图像进行中值滤波，任意比例缩小，任意中心点任意角度旋转三个任务，同时具有简单的操作界面。

针对这些要求，本文设计并开发了基础的命令行形式的操作界面，用于完成三个任务。同时也利用 QT 开发环境，编写了一套具有 UI 的操作界面，能够更加直观的进行 BMP 图像的处理。

对于 UI 操作界面的开发，本设计的思路是，先用基础的操作界面完成对图像处理的三个任务，完成基本算法的源代码，随后通过 QT 这套环境，完成算法的移植，并通过编写 UI 界面，将不同的源代码功能赋予 UI 界面，让用户可以通过 UI 界面直接操作相关算法。

目前已经完善整套系统，可以正常的完成对 BMP 图像处理的三个任务，并拥有了一套较为美观，符合逻辑的操作系统。

目录

本设计所有代码都已上传 Github 仓库并开源，记录从开始写 QT 后所有代码的编写过程

https://github.com/Saturday-365/PhotoShop-Lite。

第一章 设计任务

本项目将设计一套基础的用终端显示的简单菜单驱动下的 BMP 处理软件，该软件可以做到基本的菜单显示、中值滤波、图像任意比例缩小、图像任意角度任意中心点旋转的功能。并在此基础上利用完成后的源代码算法，配套完成一套以 QT 为基础的 UI 菜单界面，为使用者提供一个美观，便捷符合逻辑的操作界面，以完成相应任务，并实时显示出被改变后的图片，让使用者能更加直观的对图片进行操作。

第二章 系统整体设计

2.1 系统设计方案

本系统采用 Qt 框架进行开发，使用 C++ 语言编写代码。Qt 框架提供了丰富的类库和工具，能够方便地实现图形界面设计、事件处理、文件操作等功能。系统主要由主窗口和子窗口组成，主窗口作为系统的核心界面，负责显示图像和提供基本操作按钮，用户可以通过点击这些按钮执行不同的操作。子窗口则用于输入图像缩小或旋转的参数，为用户提供了一个简洁、直观的参数输入界面。本系统本社可以通过调用 Qt 的相关类库实现图像的读取、处理和显示，但是为了学习相关算法原理，本系统只使用了 QT 中对图像的显示，而读取，写入，缩小，滤波等功能全部使用自己编写的代码来进行操作。

2.2 工作原理

系统启动后，首先显示主窗口。用户可以通过点击主窗口上的按钮执行不同的操作，如打开图片、保存图片、设置保存路径等。当用户选择图像缩小或旋转功能时，系统会弹出子窗口，用户在子窗口中输入相应的参数后，点击确定按钮，系统将根据用户输入的参数对图像进行处理，并在主窗口中显示处理后的图像。在图像读取和保存过程中，系统会自动处理文件格式和数据存储，确保图像的正确读写。在图像处理过程中，系统会调用相应的算法对图像进行处理，如中值滤波、图像缩小和旋转等。

2.3 各功能模块划分

主窗口模块：负责显示图像和提供基本操作按钮，如打开图片、保存图片、设置保存路径等。该模块是用户与系统交互的主要界面，通过合理布局和设计，为用户提供了一个简洁、易用的操作环境。

图像读取模块：负责从文件中读取图像数据，并将其显示在主窗口中。该模块使用 Qt 的 QImage 类，通过 Qlable 显示图像数据，支持多种常见的图像文件格式，如 JPEG、PNG 等，但是由于图像处理算法是针对 24 位的 BMP 格式文件编写的所以在本程序中只允许 BMP 格式的文件被打开。

图像保存模块：负责将处理后的图像保存到指定的文件中。该模块同样使用 Qt 的 QImage 类来保存图像数据，用户可以选择保存的文件格式和路径。

中值滤波模块：对图像进行中值滤波处理，去除图像中的噪声。该模块使用中值滤波算法对图像的每个像素进行处理，通过对像素邻域内的像素值进行排序，取中间值作为该像素的新值，从而有效去除噪声。

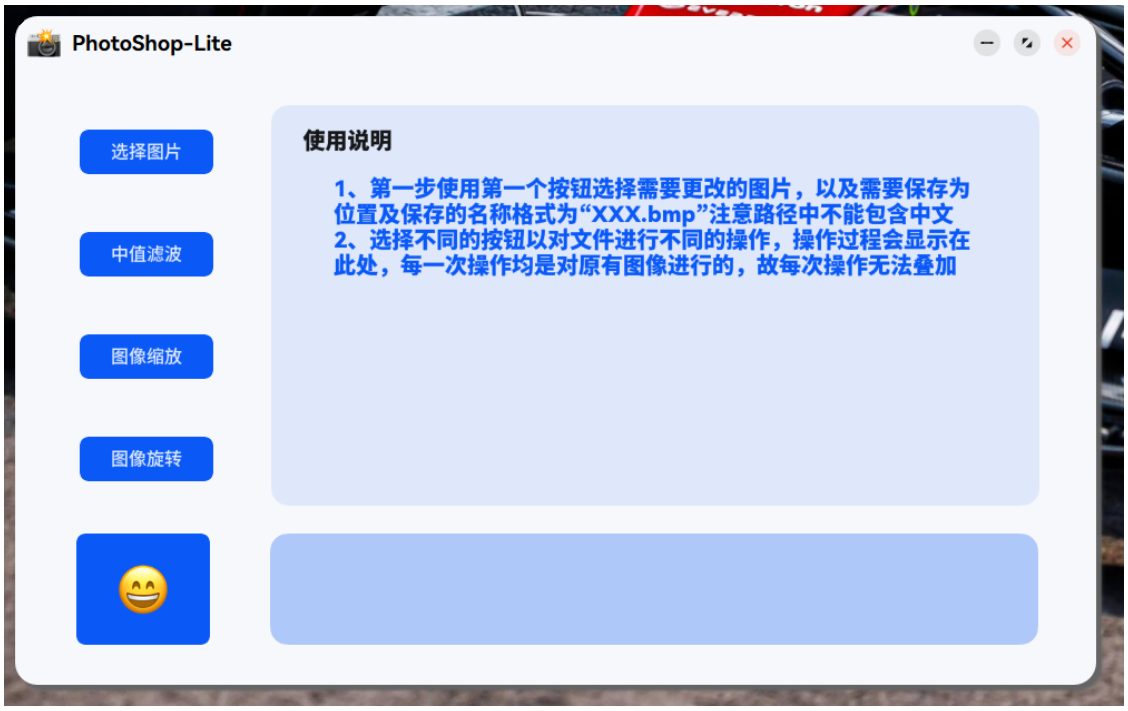
图像缩小模块：根据用户输入的参数，对图像进行缩小处理。该模块通过对不同位置的像素点进行映射来达到对图像进行缩放的功能，理论上也可以使用更加高级优秀的算法来优化缩放功能，使缩放后的图像更加自然，清晰。由于本课程设计主要目标在于 QT 的相关使用，故使用简单算法完成相印的功能。

图像旋转模块：根据用户输入的参数，对图像进行旋转处理。该模块同样使用映射的思想，通过对图像旋转后的像素位置进行映射，即计算旋转后图像对应原图像的位置来对每个像素进行映射，得到旋转后图像的像素值，实现图像的旋转效果。

第三章 功能模块设计

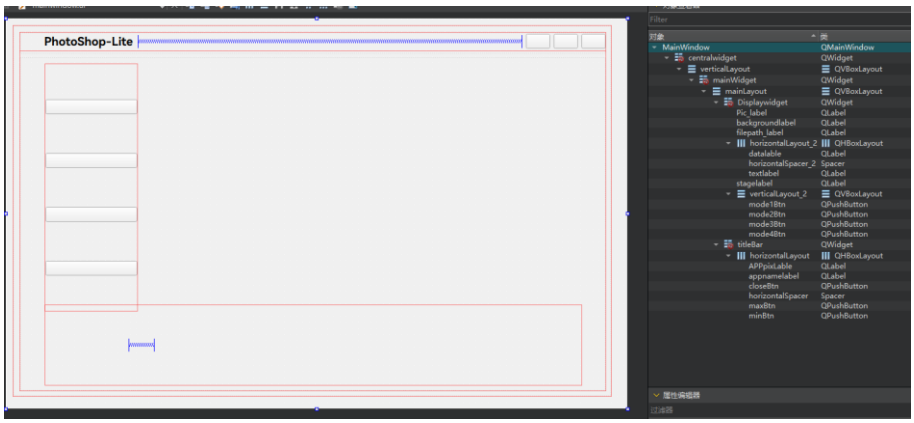
3.1 各功能模块介绍

3.1.1 主窗口模块



主窗口模块是系统的核心模块，负责显示图像和提供基本操作按钮。主窗口包含以下组件：

1、顶部显示及工具栏：包含软件图标、软件名称、以及基本的窗口操作。由于想要对软件进行圆角处理，而必须舍弃 windows 本身的窗口，所以需要为了实现圆角这一窗口，一是需要对 ui 界面进行重构（图 3.1.1），需要对主 widget 进行透明处理，同时添加一个 displaywidget 进行圆角的显示。二是由于设置了无边框窗口，以及透明窗口，（图 3.1.2、图 3.1.3）所以原本的软件图标以及软件名称，最小化最大化关闭键都消失，需要自己重新完成。（图 3.1.4）三是设置为透明窗口后，界面中鼠标的拖动事件时效，需要自己重新编写，以达到能够拖动的效果



----图 3.1.1 为了实现圆角重构 UI----

```
setWindowFlag(Qt::FramelessWindowHint);          //无边框
setAttribute(Qt::WA_TranslucentBackground);      //窗口透明

//直接初始化不起作用，需要定时器延时初始化
QTimer *t = new QTimer(this);
connect(t, &QTimer::timeout, this, [=](){Init();});
t->setSingleShot(true);
t->start(10);
```

----图 3.1.2 设置窗口模式----

```
void MainWindow::Init()
{
    const int cornerRadius = 20;                    //窗口倒角弧度
    QColor mainBackGround = QColor(247, 248, 251);  //背景色，默认白色
    //绘制遮罩
    QPainterPath path;
    path.addRoundedRect(ui->mainWidget->rect(),cornerRadius-1,cornerRadius-1);
    QRegion mask(path.toFillPolygon().toPolygon());
    ui->mainWidget->setMask(mask);
    //设置主界面样式
    QString mainStyle("QWidget#mainWidget{background-color:"
                      + mainBackGround.name()
                      + QString::asprintf(";border-radius:%dpx", cornerRadius)
                      + "}");
    ui->mainWidget->setStyleSheet(mainStyle);
    // //设置投影效果
    QGraphicsDropShadowEffect *windowShadow;        //阴影效果
    windowShadow = new QGraphicsDropShadowEffect(this);
    windowShadow->setBlurRadius(5);
    windowShadow->setColor(QColor(100, 100, 100));
    windowShadow->setOffset(5, 5);

    ui->centralwidget->setGraphicsEffect(windowShadow);
}
```

----图 3.1.3 为窗口添加圆角和阴影----

```
//最小化界面
void MainWindow::on_minBtn_clicked()
{
    this->showMinimized();
}

//关闭界面
void MainWindow::on_closeBtn_clicked()
{
    this->close();
}

//放大，缩小界面
void MainWindow::on_maxBtn_clicked()
{
    static bool max = false;
    static QRect location = this->geometry();
    if (max) {
        this->setGeometry(location);//回复窗口原大小和位置
        ui->maxBtn->setIcon(QIcon(":/MAX_.png"));
    }else {
        ui->maxBtn->setIcon(QIcon(":/minMAX.png"));
        location = this->geometry();     //最大化前记录窗口大小和位置
        //this->setGeometry(qApp->desktop()->availableGeometry());
        this->showFullScreen(); //设置窗口铺满全屏
    }
    max = !max;
}
```

----图 3.1.4 重新编写右上角三大控制按键----

2、工具栏：提供常用的操作按钮，如选择图片、中值滤波、图像缩小、图像旋转。工具栏的按钮布局紧凑，图标直观，方便用户快速执行常用操作。工具栏则由基础的槽函数构成，通过槽函数调用先前编写好的图像处理源代码来对图像进行操作。

3、图像显示区域：用于显示当前打开的图像。该区域能够自适应图像的大小，确保图像能够完整显示。对于此处，使用了两个大小不同的 Label 来实现相应功能，即一个蓝色区域，代表此处的功能（图 3.3.1），一个图像显示 label 用于显示图像（图 3.3.2）

```
QPixmap pixmapbackground(":/Icon/background.png");  //设置图片显示位置背景浅蓝色
if(!pixmapbackground.isNull()){
    ui->backgroundlabel->clear();
    //QSize lableSize = pixmapbackground.size(); //设置大小为图片大小
    QSize lableSize = ui->backgroundlabel->size();//设置大小为图标大小
    ui->backgroundlabel->setPixmap(
                    pixmapbackground.scaled(lableSize,Qt::KeepAspectRatio,Qt::SmoothTransformation));
    //将图片按照原来的宽高比进行缩放到指定lable的大小
    ui->backgroundlabel->setAlignment(Qt::AlignCenter);//图片居中这个lable
}
```

----图 3.3.1 绘制背景浅蓝色----

```
ui->filepath_label->setText(FilePath);
ui->filepath_label->setAlignment(Qt::AlignCenter);//图片居中这个lable

if(!FilePath.isNull()){
    QPixmap pixmapin(FilePath);
    if(!pixmapin.isNull()){
        QSize lableSize = ui->Pic_label->size();   // 获取当前窗口大小
        ui->Pic_label->setPixmap(pixmapin.scaled(lableSize,Qt::KeepAspectRatio,Qt::SmoothTransformation));
        ui->Pic_label->setAlignment(Qt::AlignCenter);//图片居中这个lable
    }
}
```

----图 3.3.2 通过获取的路径完成图像显示----

4、图像地址显示区域：用于显示用户选择的图片路径地址，方便用户了解输入的图片路径中是否包含中文路径，如包含中文路径则程序无法正常进行。

3.1.2 图像读取模块

此模块对应“选择图片”的槽函数。此按钮被点击后，先调用 QFileDialog，选择原图片，并记录下原图片的路径，然后将源图片的路径转换为源代码中 io 流可以读取的路径。完成后同样对输出路径进行操作，（图 3.2.1）通过此按钮就得到了源代码中需要的两个参数，及图片输入路径和图片输出路径，再此过程中，源图片的路径被利用，在主串口中的图片显示 label 中显示（图 3.2.2、图 3.2.3）。

```
FilePath=QFileDialog::getOpenFileName(this,
"OpenPicture-File 打开你想要转换的BMP格式文件(不能包含中文路径)",":/Picture","BMP-img(*.bmp)");
sFilePath = FilePath.toStdString();
sFilePath=Process.convertPath(sFilePath);   //转化路径格式为io流可以读取的格式 /*  "/"->"\\" */
//ui->Pic_filepath_textEdit->insertPlainText(FilePath);// 显示打开的图片的路径

ui->filepath_label->setText(FilePath);
ui->filepath_label->setAlignment(Qt::AlignCenter);//图片居中这个lable

if(!FilePath.isNull()){
    QPixmap pixmapin(FilePath);
    if(!pixmapin.isNull()){
        QSize lableSize = ui->Pic_label->size();   // 获取当前窗口大小
        ui->Pic_label->setPixmap(pixmapin.scaled(lableSize,Qt::KeepAspectRatio,Qt::SmoothTransformation));
        ui->Pic_label->setAlignment(Qt::AlignCenter);//图片居中这个lable
    }
}
```

----图 3.2.1 路径或者和路径转换----

```
if(!FilePath.isNull()){
    QPixmap pixmapin(FilePath);
    if(!pixmapin.isNull()){
        QSize lableSize = ui->Pic_label->size();   // 获取当前窗口大小
        ui->Pic_label->setPixmap(pixmapin.scaled(lableSize,Qt::KeepAspectRatio,Qt::SmoothTransformation));
        ui->Pic_label->setAlignment(Qt::AlignCenter);//图片居中这个lable
    }
}
```

----图 3.2.2 居中显示图像----

```
QPixmap pixmapbackground(":/Icon/background.png");  //设置图片显示位置背景浅蓝色
if(!pixmapbackground.isNull()){
    ui->backgroundlabel->clear();
    //QSize lableSize = pixmapbackground.size(); //设置大小为图片大小
    QSize lableSize = ui->backgroundlabel->size();//设置大小为图标大小
    ui->backgroundlabel->setPixmap(
                    pixmapbackground.scaled(lableSize,Qt::KeepAspectRatio,Qt::SmoothTransformation));
    //将图片按照原来的宽高比进行缩放到指定lable的大小
    ui->backgroundlabel->setAlignment(Qt::AlignCenter);//图片居中这个lable
}
```

----图 3.2.3 设置图像显示区域背景----

3.1.4 中值滤波模块

此模块对应“中值滤波”的槽函数。此按钮被点击后，直接利用之前在“选择图片”中获取的两个地址，输入给先前编写的源代码，通过源代码完成对应图片的操作，在操作完成后，重新调用图片读取代码，重新对主窗口中的图片显示进行更新，将图片更新为经过代码运算后生成的新图片（图 3.2.3）。

```
void MainWindow::Button_medianFilter(){

    QPixmap pixmap1(":/Icon/doing.png");  //设置加载区图片显示位置背景
    if(!pixmap1.isNull()){
        ui->textlabel->clear();
        QSize lableSize = pixmap1.size(); //设置大小为图片大小
        //QSize lableSize = ui->textlabel->size();//设置大小为图标大小
        ui->textlabel->setPixmap(pixmap1.scaled(lableSize,Qt::KeepAspectRatio,Q
        ui->textlabel->setAlignment(Qt::AlignCenter);//图片居中这个lable
    }

    ui->stagelabel->setAlignment(Qt::AlignCenter);//居中这个lable
    // QThread::msleep(2000);//阻塞延时50ms
    Process.medianFilter(sFilePath,sFilePath_Out);//执行bmp处理函数

    QPixmap pixmap2(":/Icon/finish.png");  //设置加载区图片显示位置背景
    if(!pixmap1.isNull()){
        ui->textlabel->clear();
        QSize lableSize = pixmap2.size(); //设置大小为图片大小
        //QSize lableSize = ui->textlabel->size();//设置大小为图标大小
        ui->textlabel->setPixmap(pixmap2.scaled(lableSize,Qt::KeepAspectRatio,Q
        ui->textlabel->setAlignment(Qt::AlignCenter);//图片居中这个lable
    }
    reflash_PicShow();

}
```

----图 3.2.3 中值滤波槽函数处理过程----

源代码中值滤波模块对图像进行中值滤波处理，去除图像中的噪声。该模块使用中值滤波算法对图像的每个像素进行处理，具体实现步骤如下：

首先遍历图像的每个像素，对于每个像素，获取其邻域内的像素值。对邻域内的像素值进行排序，取中间值作为该像素的新值。最后将处理后的像素值更新到图像中即完成了相应的滤波（图 3.4.2）。

```
uint32 num_width = data_size / height;
uint8 *new_data = new uint8[data_size];
memcpy(new_data, data, data_size);
const int FILTER_SIZE = 25;
uint8 values[FILTER_SIZE]; // 5x5邻域
for (int i = 2; i < height - 2; i++) {
  for (int j = 2; j < width - 2; j++) {
    for (int c = 0; c < 3; c++) {
      // 收集邻域像素值到数组
      int index = 0;
      for (int x = -2; x <= 2; x++) {
        for (int y = -2; y <= 2; y++) {
          values[index++] = *(data + (i + x) * num_width + (j + y) * 3 + c);
        }
      }
      // 使用冒泡排序
      for (int pass = 0; pass < FILTER_SIZE - 1; pass++) {
        for (int k = 0; k < FILTER_SIZE - pass - 1; k++) {
          if (values[k] > values[k + 1]) {
            uint8 temp = values[k];
            values[k] = values[k + 1];
            values[k + 1] = temp;
          }
        }
      }
      // 对应像素位取中值
      *(new_data + i * num_width + j * 3 + c) = values[12];
    }
  }
}
writeBMPInfo(new_name, width, height, data_offset, data_size, new_data, name);
```

----图 3.2.3 图像中值滤波过程----

3.1.5 图像缩小模块

此模块对应“图像缩小”的槽函数。由于此函数除了需要，输入图像和输出图像的地址外，还需要两个参数即 x 方向的缩小比例和 y 方向的缩小比例，因此此按钮被点击后，会产生一个子窗口，用于获取两个比例值，来输入给图像处理的源代码完成相应任务。

对于获取这两个值，我采用了两个滑条的形式来获取数据（图 3.5.1）用户通过滑动这两个滑条来输入相应参数，同时也拥有微调框（图 3.5.1），便于用户进行更精细的参数调节。这两个值获取后输入给相应函数，完成新图片的生成。

对于这个子窗口，同样需要一个圆角窗口，以满足整个软件的整体性，所以重新编写了关闭按钮等（图 3.5.1）以追求美观。最终是如何通过子窗口的按钮来控制主窗口中图片的显示，即如何在字窗口中，调用主窗口中的控件，在查阅资

料后，通过传递主窗口结构体（图 3.5.2），来让子窗口能够调用主窗口中的控件，最准实现了所有功能。

----图 3.5.1 子窗口界面----

```
private:
    Ui::rotateimage *ui;
    MainWindow *m_parent;                // 主窗口类的对象
```

```
m_parent = static_cast<MainWindow*>(parent);
```

----图 3.5.2 结构体传递----

源代码中对于缩小的实现，该模块通过对不同位置的像素点进行映射（图 3.5.3）来达到对图像进行缩放的功能，在实现过程中，由于 X,Y 参数的随机性，可能导致直接的简单的映射域的取值会让最终生成的 bmp 格式文件不符合 4 字节规范，从而导致图像显示的不正确，进而导致程序的崩溃，所以，源代码还添加了 4 字节检测（图 3.5.4），以及 4 字节自动补齐等功能，以保证图像的正确缩放，及正确生成。

```
// 计算新旧图像每行字节数
uint32 row_size = calculateRowSize(width);
uint32 new_row_size = calculateRowSize(new_width);
uint32 new_data_size = new_row_size * new_height;

uint8 *new_data = new uint8[new_data_size];
memset(new_data, 0, new_data_size);

for (int i = 0; i < new_height; i++) {
    for (int j = 0; j < new_width; j++) {
        // 计算源图像中的对应位置（使用最近邻插值）
        int src_i = static_cast<int>(i / ratio_y + 0.5);
        int src_j = static_cast<int>(j / ratio_x + 0.5);

        // 确保不越界
        src_i = min(src_i, static_cast<int>(height) - 1);
        src_j = min(src_j, static_cast<int>(width) - 1);

        for (int c = 0; c < 3; c++) {
            // 直接使用最近邻像素值
            *(new_data + i * new_row_size + j * 3 + c) = *(data + src_i * row_size + src_j * 3 + c);
        }
    }
}
```

----图 3.5.3 缩小函数算法----

```
// 计算每行像素数据的字节数（必须是4的倍数）
uint32 BMP_Process::calculateRowSize(uint32 width) {
    return 4 * ((width * 3 + 3) / 4); // 确保每行字节数是4的倍数
}
```

----图 3.5.4 4 字节检测函数----

3.1.6 图像旋转模块

此模块对应“图像旋转”的槽函数。同“图像缩放”此函数除了需要，输入图像和输出图像的地址外，还需要三个参数即旋转中心点的 X 坐标和 Y 坐标，以及旋转的度数，因此此按钮被点击后，同样会产生一个子窗口（图 3.6.1），用于获取旋转中心点坐标和旋转角度，来输入给图像处理的源代码完成相应任务。

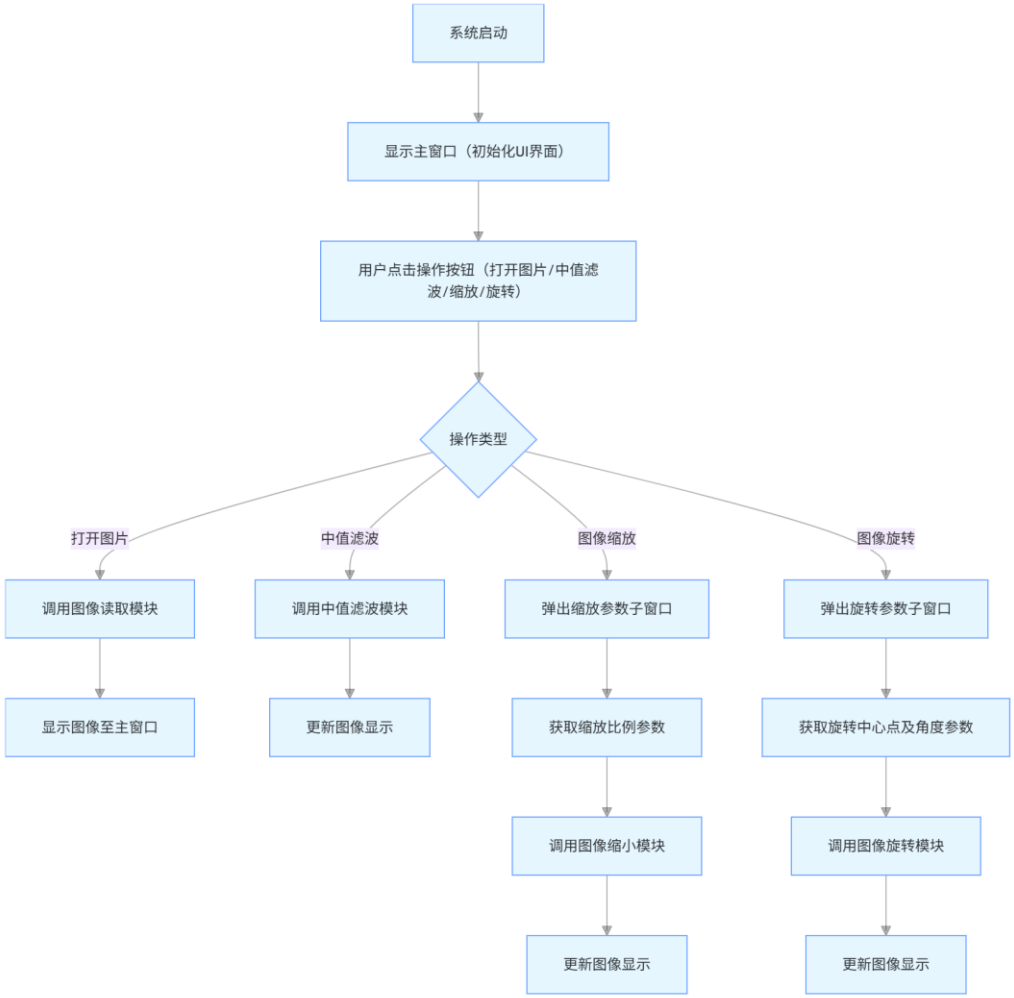
对于获取这三个个值，我采用了两个滑条的，和一个旋钮的形式来获取数据（图 3.6.1）用户通过滑动这两个滑条来输入相应参数，滑条和旋钮几乎一比一战术中心点和旋转度数，直观的展示给用户图像会出现的结构，便于用户对图像的操作。和用户想要旋转的度数，同时也拥有微调框（图 3.6.1），便于用户进行更精细的参数调节。这两个值获取后输入给相应函数，完成新图片的生成。

对于这个子窗口，同样“图像缩放”一样需要一个圆角窗口，以满足整个软件的整体性，详情可看“图像缩放”中的展示。

----图 3.6.1 旋转模块子窗口----

源代码中对于旋转图像，同样使用映射的思想，通过对图像旋转后的像素位置进行映射，即计算旋转后图像对应原图像的位置来对每个像素进行映射，得到旋转后图像的像素值，实现图像的旋转效果。

3.2 设计流程图


系统启动
显示主窗口（初始化UI界面）
用户点击操作按钮（打开图片/中值滤波/缩放/旋转）
操作类型
打开图片
中值滤波
图像缩放
图像旋转
调用图像读取模块
调用中值滤波模块
弹出缩放参数子窗口
弹出旋转参数子窗口
显示图像至主窗口
更新图像显示
获取缩放比例参数
获取旋转中心点及角度参数
调用图像缩小模块
调用图像旋转模块
更新图像显示
更新图像显示

第四章 系统测试

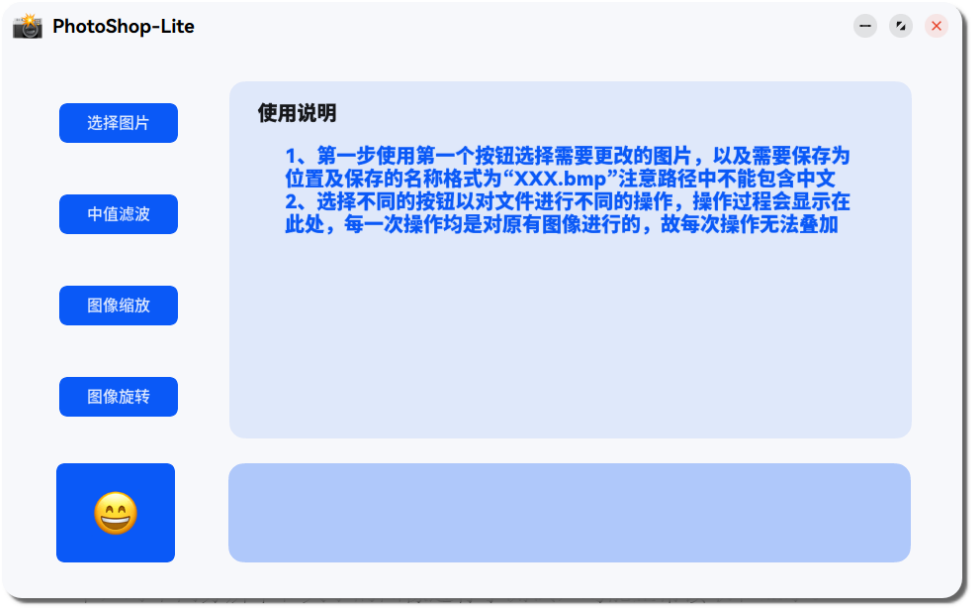
4.1 测试环境及设备

操作系统：Windows 11
开发环境：Qt Creator 16.0.0.1
编译器：MSVC 2022 64-bit

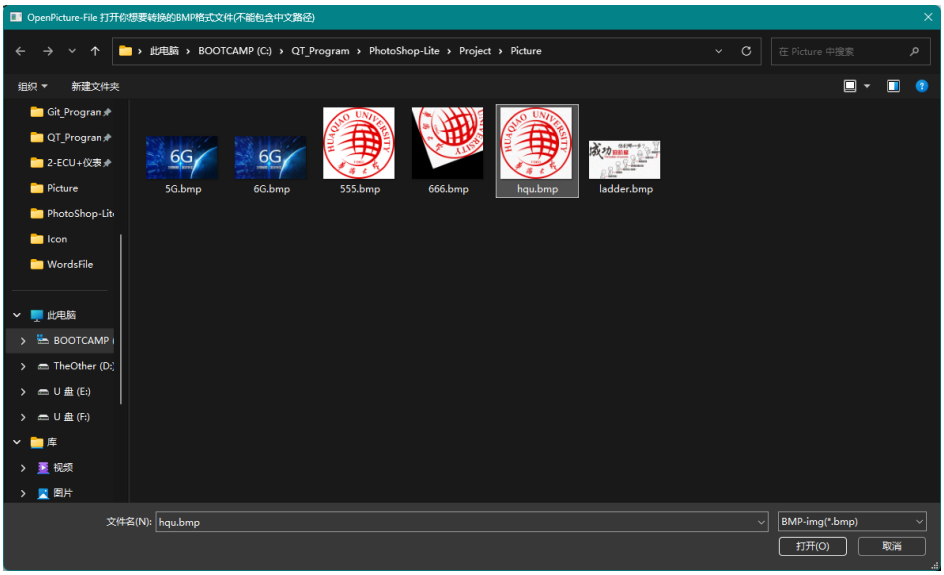
4.2 每个功能点 / 指标的测试流程和测试结果

4.2.1 图像读取功能

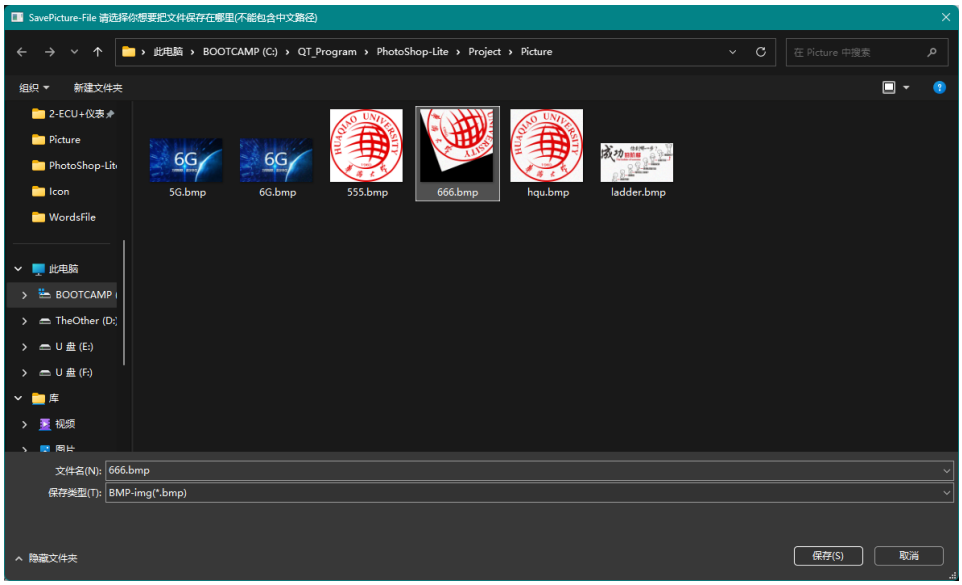
测试流程：



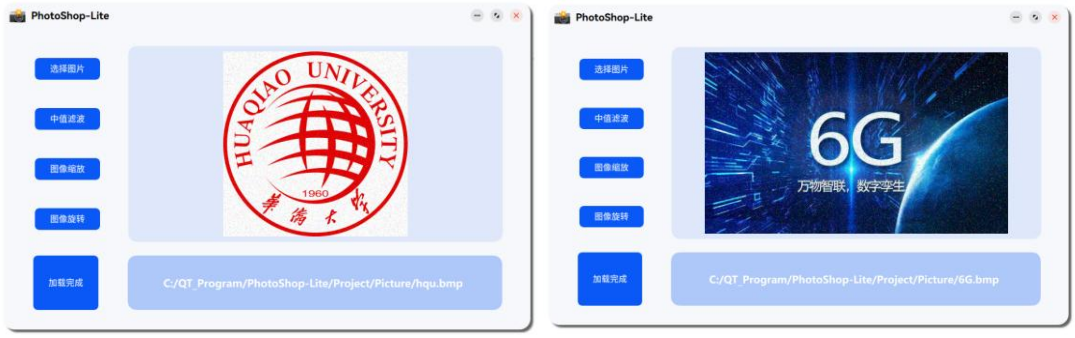
启动系统，点击 “选择图片” 按钮。



选择不同的 BMP 图像文件进行打开操作。



选择 BMP 图像文件的保存位置。



检查图像是否成功显示在主窗口中。

测试结果：能够成功读取图像文件，并在主窗口中显示。在测试过程中，对不同分辨率和大小的图像进行了测试，均能正常读取和显示。

4.2.3 中值滤波功能

测试流程：
打开一张含有噪声的图像。
点击 “中值滤波” 按钮。
观察滤波后的图像，检查噪声是否得到有效去除。

测试结果：对含有噪声的图像进行中值滤波处理后，噪声得到有效去除，图像质量明显提高。在测试过程中，对不同的噪声图像进行了测试，均能有效去除噪声。

4.2.4 图像缩小功能

测试流程：
打开一张图像。
点击 “图像缩小” 按钮，弹出子窗口。

在子窗口中通过滑块不同的缩小参数。

点击 “确定” 按钮，观察缩小后的图像是否显示正常。
测试结果：能够根据用户输入的参数对图像进行缩小处理，缩小后的图像显示正常，没有出现失程序闪退等现象。在测试过程中，对不同图像进行了不同缩小比例的缩小测试，均能正常缩小。

4.2.5 图像旋转功能

测试流程：
打开一张图像。
点击 “图像旋转” 按钮，弹出子窗口。

在子窗口中通过滑块和旋转控件控制不同的旋转中心点和旋转角度

点击 “确定” 按钮，观察旋转后的图像是否显示正常。

测试结果：能够根据用户输入的参数对图像进行旋转处理，旋转后的图像显示正常，没有出现变形或扭曲的现象。在测试过程中，对不同旋转角度及不同中心点进行了测试，均能正常旋转。

第五章 结论

5.1 课程设计总结

首先是知识方面，在之前的学习过程之中，我一直接触的是基础的嵌入式 C 语言，除了基础的语法函数，能基本懂一些指针，结构体，并熟练运用到竞赛之中。而通过本次课程设计，我深入学习了 C++中的更多语法结构，对我的代码能力有个很大的补充，对计算机语言也有了更加深入的了解。同时我也在本次的课程设计中浅浅的学习了 Qt 框架的使用，遗憾的是并没有系统的学习，更多是利用之前学习的经验，凭借自己的理解，需要什么功能，搜索相关资料，运用相关函数以及 QT 的函数联想来完成本次课程设计。后续可能会利用本次课程设计中学习到的相关 QT 的知识，开发一套更加美观，符合逻辑，深度定制的上位机，来配合我自己的一些嵌入式项目。

然后是学习能力方面，在本次的课程设计中，从 0 开始接触了 QT 的相关知识，从配置环境，到最后的完成本次课程设计，这一段学习经历让我的学习能力，得到了很大的提升。

在这一次的课程设计中，我也对 AI 协助编写代码有了一些经验，对于 AI 解决编程中的问题有了一些体会。

如在尝试编写通过子窗口调用主窗口 UI 控件时，我通过查找相关资料，实现了子窗口的调用，但是由于传递了主窗口的结构体，导致子窗口于主窗口融合，失去了子窗口的意义，查找各种资料无果后，询问 AI，得知是子窗口类型设置文件，增加半行代码后问题得到解决。

还有就是在我通过自己尝试完成本次设计中的源代码后，我尝试让 AI 来编写相应任务的源代码，我提供给 AI 我之前编写的含有基本命令行菜单界面的代码，和老师提供的 BMP 处理例程，两个文件，以及告诉他本次的题目，结果 AI 在第一次生成中就几乎很好的完成了相应的任务，而且使用了更加简洁的函数，达到了我使用更多行完成的任务，后续经过几次额外的提示词修改，AI 很完美的完成了源代码的编写，而我只是给了它两个文件，及几句话。

在这一次经历中，丰富了我使用 AI 的经历，可能会大幅加快我之后的设计效率，我认为 AI 也是我在学习过程中的一位良师，它告诉了我，完成同样功能，可以使用更加简便的算法，有更直接的函数。我认为，要做好一件事，首先要学会的就是模仿，只有知道什么是好的代码，什么是优秀简洁的代码，学会去模仿，学会去阅读，模仿，使用，理解，才能一步一步走向由模仿到超越的结果。而 AI 就便捷迅速的为我们提供一个很好的模范标准。

最后，在本次课程设计中，我深刻体会到了 UI 相关代码编写的困难程度，浅浅入门了 QT 以及 C++这门语言，希望我能在之后的日子中一步一步提升自己的代码便携能力，提升自己的逻辑思维，算法思维，能更加深入的理解 C++这门语言。

5.2 对本课程的意见和建议

首先，在本课程中，我学到了很多知识，很大程度上丰富了我的软件经历。学到了更独属于 C++的知识，对 C++有了更加深入的了解。

其次，对于本课程的课程设计的形式，而非考试，我个人是非常认同的，学到的知识只有付诸于实践，付诸于每一个项目之中，才能被得到真正的检验。

最后是个人对于本课程的一点小小的建议，即对于平时的作业题目，我认为可以引入一些在线测评类似 noi.openjudge.cn 及各种书籍自带的在线测评系统。关于此，一个方面是，仅靠自己输入一些测试代码有的是不能很好的发现代码中的 bug，使用在线测评系统则能应对此问题。伴随其发生的问题可能会有，题目选择受限，以及直接能在网路中搜索到答案，但是我认为，大部分同学还是会利用好类似的系统完善自己的代码，而不是直接抄袭。另一个方面是，在代码学习的过程中，需要一些正反馈，为了 AC 的过程中，不断修改自己的代码，一步一步进化自己的代码，直到最终 AC，是一个很好的正反馈过程，让同学们更加能投入进来，有成就感。同时也能告诉我们，有这么多的题目可以练习，有这样的渠道可以了解。

参考文献

https://blog.csdn.net/sinat_36420785/article/details/81218049
https://blog.csdn.net/keanight/article/details/79150637
https://blog.csdn.net/qq_33485434/article/details/80680506 qstring 和 string 转化
https://blog.csdn.net/u012790503/article/details/119855465 槽函数相关探究
https://www.cnblogs.com/LifeoFHanLiu/p/9978425.html getOpenFileName getSaveFileName
https://blog.csdn.net/jkjijijkv/article/details/119969325 弹出新窗口
https://blog.csdn.net/yanghz/article/details/135004353 Qt6.5 类库详解：QTextBrowse
https://blog.csdn.net/qq_45652092/article/details/111183979 Qt 中的 Qlabel 类（标签类）详解
https://blog.csdn.net/qq_14945437/article/details/98730805 Qt 之 QSlider 介绍
https://blog.csdn.net/Sakuya__/article/details/105885764 QT 在子窗口中调用主窗口的 UI
https://blog.csdn.net/yy_xzz/article/details/148299320 Qt 窗口标志（Window Flags）
https://blog.csdn.net/qq_44084616/article/details/109612685
https://juejin.cn/post/7186248005496864824 无边框窗口圆角
https://blog.csdn.net/qq_44084616/article/details/109612685 无边框窗口移动
https://blog.csdn.net/hiwoshixiaoyu/article/details/122468086 ui 界面布局
https://blog.csdn.net/rong11417/article/details/104000236 Qt
延时/等待写法 阻塞延时/不阻塞延时/耗时代码的处理

附录

本设计所有代码都已上传 Github 仓库并开源，记录从开始写 QT 后所有代码的编写过程

https://github.com/Saturday-365/PhotoShop-Lite。

-----图像处理算法核心代码------

```
#include <conio.h>
#include <cstring>
#include <fstream>
// #include <iomanip>
#include "BMP_Process.h"
#include <QString>
#include <iostream>
#include <math.h>
#include <sstream>
#include <string>


#define M_PI 3.1415926; // 定义圆周率常量

string BMP_Process::convertPath(const string &path) {
  stringstream result;
  for (char c : path) {
    if (c == '/') {
      result << "\\\\";
    } else {
      result << c;
    }
  }
  return result.str();
}

void BMP_Process::readBMPInfo(string name, uint32 &width, uint32 &height,
                              uint32 &data_offset, uint32 &data_size,
                              uint8 *&data) {
  fstream bmpdata;
  bmpdata.open(name, ios::binary | ios::in); // 以二进制只读模式打开文件
  if (bmpdata.fail()) {                      // 检查文件是否打开失败
    // cout << "原始图像读取失败";
    exit(5); // 若失败则退出程序
  }
  bmpdata.seekg(18, ios::beg);
  bmpdata.read((char *)(&width), sizeof(width));
```

```
  bmpdata.seekg(22, ios::beg);
  bmpdata.read((char *)(&height), sizeof(height));
  bmpdata.seekg(10, ios::beg);
  bmpdata.read((char *)(&data_offset), sizeof(data_offset));
  bmpdata.seekg(34, ios::beg);
  bmpdata.read((char *)(&data_size), sizeof(data_size));
  data = new uint8[data_size];
  bmpdata.seekg(data_offset, ios::beg);
  bmpdata.read((char *)data, data_size);
  bmpdata.close(); // 关闭文件
}

// 写入 BMP 图像信息的函数
void BMP_Process::writeBMPInfo(string new_name, uint32 width, uint32
height,
                               uint32 data_offset, uint32 data_size,
                               uint8 *data, string old_name) {
  fstream bmpw;
  bmpw.open(new_name, ios::binary | ios::out); // 以二进制写模式打开文件
  if (bmpw.fail()) {                           // 检查文件是否打开失败
    // cout << "新图像写入失败";
    exit(4); // 若失败则退出程序
  }
  fstream bmpdata;
  bmpdata.open(old_name, ios::binary | ios::in);
  uint8 *tmp = new uint8[data_offset];
  bmpdata.seekg(0, ios::beg);
  bmpdata.read((char *)tmp, data_offset);
  bmpw.write((char *)tmp, data_offset);
  delete[] tmp;    // 释放临时内存
  bmpdata.close(); // 关闭原始文件
  bmpw.seekp(18, ios::beg);
  bmpw.write((char *)(&width), sizeof(width));
  bmpw.seekp(22, ios::beg);
  bmpw.write((char *)(&height), sizeof(height));
  bmpw.seekp(34, ios::beg);
  bmpw.write((char *)(&data_size), sizeof(data_size));
  bmpw.seekp(data_offset, ios::beg);
  bmpw.write((char *)data, data_size);
  bmpw.close();
}

// 中值滤波
void BMP_Process::medianFilter(string name, string new_name) {
```

```
  uint32 width, height, data_offset, data_size;
  uint8 *data;

  readBMPInfo(name, width, height, data_offset, data_size, data);

  uint32 num_width = data_size / height;
  uint8 *new_data = new uint8[data_size];
  memcpy(new_data, data, data_size);
  const int FILTER_SIZE = 25;
  uint8 values[FILTER_SIZE]; // 5x5 邻域
  for (int i = 2; i < height - 2; i++) {
    for (int j = 2; j < width - 2; j++) {
      for (int c = 0; c < 3; c++) {
        // 收集邻域像素值到数组
        int index = 0;
        for (int x = -2; x <= 2; x++) {
          for (int y = -2; y <= 2; y++) {
            values[index++] = *(data + (i + x) * num_width + (j + y) * 3 +
c);
          }
        }
        // 使用冒泡排序
        for (int pass = 0; pass < FILTER_SIZE - 1; pass++) {
          for (int k = 0; k < FILTER_SIZE - pass - 1; k++) {
            if (values[k] > values[k + 1]) {
              uint8 temp = values[k];
              values[k] = values[k + 1];
              values[k + 1] = temp;
            }
          }
        }
        // 对应像素位取中值
        *(new_data + i * num_width + j * 3 + c) = values[12];
      }
    }
  }
  writeBMPInfo(new_name, width, height, data_offset, data_size, new_data,
name);
  delete[] data;
  delete[] new_data;
}
// 计算每行像素数据的字节数（必须是 4 的倍数）
uint32 BMP_Process::calculateRowSize(uint32 width) {
    return 4 * ((width * 3 + 3) / 4); // 确保每行字节数是 4 的倍数
```

```
}
// 图像缩小函数
void BMP_Process::shrinkImage(string name, string new_name, double
ratio_x,double ratio_y) {
  uint32 width, height, data_offset, data_size;
  uint8 *data;
  readBMPInfo(name, width, height, data_offset, data_size, data);

  uint32 new_width = static_cast<uint32>(width * ratio_x);
  uint32 new_height = static_cast<uint32>(height * ratio_y);

  // 计算新旧图像每行字节数
  uint32 row_size = calculateRowSize(width);
  uint32 new_row_size = calculateRowSize(new_width);
  uint32 new_data_size = new_row_size * new_height;

  uint8 *new_data = new uint8[new_data_size];
  memset(new_data, 0, new_data_size);

  for (int i = 0; i < new_height; i++) {
      for (int j = 0; j < new_width; j++) {
          // 计算源图像中的对应位置（使用最近邻插值）
          int src_i = static_cast<int>(i / ratio_y + 0.5);
          int src_j = static_cast<int>(j / ratio_x + 0.5);

          // 确保不越界
          src_i = min(src_i, static_cast<int>(height) - 1);
          src_j = min(src_j, static_cast<int>(width) - 1);

          for (int c = 0; c < 3; c++) {
              // 直接使用最近邻像素值
              *(new_data + i * new_row_size + j * 3 + c) = *(data + src_i *
row_size + src_j * 3 + c);
          }
      }
  }

  writeBMPInfo(new_name, new_width, new_height, data_offset,
new_data_size,new_data, name);
  delete[] data;
  delete[] new_data;
}


// 图像旋转
```

```
void BMP_Process::rotateImage(string name, string new_name, double angle,
                              int center_x, int center_y) {
  uint32 width, height, data_offset, data_size;
  uint8 *data;
  readBMPInfo(name, width, height, data_offset, data_size, data);

  // 计算好角度值
  double rad = angle / 180.0 * M_PI;
  double cos_angle = cos(rad);
  double sin_angle = sin(rad);

  // 计算原始图像每行数据的字节数
  uint32 num_width = data_size / height;
  // 动态分配内存用于存储旋转后的图像数据
  uint8 *new_data = new uint8[data_size];
  memset(new_data, 0, data_size);

  // 遍历图像的每个像素
  for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
      // 计算新像素相对于旋转中心的坐标
      int new_x = j - center_x;
      int new_y = i - center_y;

      // 旋转公式计算原图像中的坐标
      int old_x = (int)(new_x * cos_angle - new_y * sin_angle) + center_x;
      int old_y = (int)(new_x * sin_angle + new_y * cos_angle) + center_y;

      // 边界检查
      if (old_x >= 0 && old_x < width && old_y >= 0 && old_y < height) {
        for (int c = 0; c < 3; c++) {
          *(new_data + i * num_width + j * 3 + c) =
              *(data + old_y * num_width + old_x * 3 + c);
        }
      }
    }
  }

  writeBMPInfo(new_name, width, height, data_offset, data_size, new_data,
name);
  delete[] data;
  delete[] new_data;
}
```

```
#ifndef BMP_PROCESS_H
#define BMP_PROCESS_H

#include <conio.h>
#include <cstring>
// #include <iomanip>
#include <QString>
#include <math.h>
#include <string>


using namespace std;
typedef unsigned char uint8;
typedef unsigned short uint16;
typedef unsigned long uint32;

class BMP_Process{
private:
  void readBMPInfo(string name, uint32 &width, uint32 &height,
                   uint32 &data_offset, uint32 &data_size, uint8 *&data);
  void writeBMPInfo(string new_name, uint32 width, uint32 height,
                    uint32 data_offset, uint32 data_size, uint8 *data,
                    string old_name);
  uint32 calculateRowSize(uint32 width);

public:
  string convertPath(const string &path);
  void medianFilter(string name, string new_name);
  void shrinkImage(string name, string new_name, double ratio_x,double
ratio_y);
  void rotateImage(string name, string new_name, double angle, int center_x,
                   int center_y);
};

#endif
```

-----QT 主窗口代码------

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include <QIcon>
#include <QTimer>
#include <QPixmap>
#include <QPalette>
#include <QWidget>
```

```
#include <QMouseEvent>
#include <QPointerEvent>
#include <QPainter>
#include <QPainterPath>
#include <QGraphicsDropShadowEffect>
#include "./SourseLib/BMP_Process.h"
#include <QThread>

#define MARGIN 2

QString FilePath,FilePath_Out;
string sFilePath,sFilePath_Out;
BMP_Process Process;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    //
setWindowFlags(Qt::FramelessWindowHint|Qt::WindowCloseButtonHint|Qt::Win
dowMinMaxButtonsHint);
    // setWindowFlag(Qt::FramelessWindowHint);
    // setAttribute(Qt::WA_TranslucentBackground);
    setWindowFlag(Qt::FramelessWindowHint);         //无边框
    setAttribute(Qt::WA_TranslucentBackground);     //窗口透明

    //直接初始化不起作用，需要定时器延时初始化
    QTimer *t = new QTimer(this);
    connect(t, &QTimer::timeout, this, [=](){Init();});
    t->setSingleShot(true);
    t->start(10);

    QPixmap pixmapAPPpix(":/Icon/image 1760.png");  //设置左上角图标
    //QPixmap pixmapclosepix(":/Icon/close.png");  //设置左上角图标
    if(!pixmapAPPpix.isNull()){
        ui->APPpixLable->clear();
        //QSize lableSize = pixmapAPPpix.size(); //设置大小为图片大小
         QSize lableSize = ui->APPpixLable->size();//设置大小为图标大小
        ui->APPpixLable->setPixmap(pixmapAPPpix.scaled(lableSize,Qt::Keep
AspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行缩放到
指定 lable 的大小
        ui->APPpixLable->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    }
```

```
    QPixmap pixmapdatalableground(":/Icon/smile.png");  //设置加载区图片显
示位置背景
    if(!pixmapdatalableground.isNull()){
        ui->textlabel->clear();
        QSize lableSize = pixmapdatalableground.size(); //设置大小为图片大
小
        //QSize lableSize = ui->textlabel->size();//设置大小为图标大小
        ui->textlabel->setPixmap(pixmapdatalableground.scaled(lableSize,Q
t::KeepAspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进
行缩放到指定 lable 的大小
        ui->textlabel->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    }

    QPixmap pixmaptextlableground(":/Icon/blue690x100.png");  //设置图片地
址显示位置背景浅蓝色
    if(!pixmaptextlableground.isNull()){
        ui->datalable->clear();
        QSize lableSize = pixmaptextlableground.size(); //设置大小为图片大
小
        //QSize lableSize = ui->datalable->size();//设置大小为图标大小
        ui->datalable->setPixmap(pixmaptextlableground.scaled(lableSize,Q
t::KeepAspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进
行缩放到指定 lable 的大小
        // ui->datalable->setAlignment(Qt::AlignCenter);//图片居中这个
lable
    }

    QPixmap pixmapbackground(":/Icon/things.png");  //设置图片显示位置背景
浅蓝色
    if(!pixmapbackground.isNull()){
        ui->backgroundlabel->clear();
        //QSize lableSize = pixmapbackground.size(); //设置大小为图片大小
        QSize lableSize = ui->backgroundlabel->size();//设置大小为图标大小
        ui->backgroundlabel->setPixmap(pixmapbackground.scaled(lableSize,
Qt::KeepAspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进
行缩放到指定 lable 的大小
        ui->backgroundlabel->setAlignment(Qt::AlignCenter);//图片居中这个
lable
    }
    // QPixmap pixmapbarbackground(":/Icon/barbackground.png");  //顶部 bar
透明黑色背景
    // if(!pixmapbackground.isNull()){
    //     ui->label->clear();
```

```
    //     //QSize lableSize = pixmapbarbackground.size(); //设置大小为图片
大小
    //     QSize lableSize = ui->label->size();//设置大小为图标大小
    //     ui->label->setPixmap(pixmapbarbackground.scaled(lableSize,Qt::
IgnoreAspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行
缩放到指定 lable 的大小
    //     ui->label->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    // }
    // setWindowTitle("PhotoShop Lite"); //设置窗口标题
    // setFixedSize(960,540);
    //setBackGround(":/Picture/mclaren senna wallpaper.jpg"); //设置背景
    initButtons();
    connect(ui->mode1Btn,&QPushButton::clicked,this,&MainWindow::Button_
OpenFile);//按钮链接槽函数 打开图片文件
    connect(ui->mode2Btn,&QPushButton::clicked,this,&MainWindow::Button_
medianFilter);//按钮链接槽函数 实现中值滤波并重新显示新的图片

}

void MainWindow::Init()
{
    const int cornerRadius = 20;                    //窗口倒角弧度
    QColor mainBackGround = QColor(247, 248, 251);  //背景色，默认白色
    //绘制遮罩
    QPainterPath path;
    path.addRoundedRect(ui->mainWidget->rect(),cornerRadius-1,cornerRadi
us-1);
    QRegion mask(path.toFillPolygon().toPolygon());
    ui->mainWidget->setMask(mask);
    //设置主界面样式
    QString mainStyle("QWidget#mainWidget{background-color:"
                      + mainBackGround.name()
                      + QString::asprintf(";border-radius:%dpx",
cornerRadius)
                      + "}");
    ui->mainWidget->setStyleSheet(mainStyle);
    // //设置投影效果
    QGraphicsDropShadowEffect *windowShadow;        //阴影效果
    windowShadow = new QGraphicsDropShadowEffect(this);
    windowShadow->setBlurRadius(5);
    windowShadow->setColor(QColor(100, 100, 100));
    windowShadow->setOffset(5, 5);

    ui->centralwidget->setGraphicsEffect(windowShadow);
```

```
}

void MainWindow::setBackGround(const QString & filename)  //设置背景
{
    QPixmap pixmap(filename);//创建照片
    QSize windowsSize = this->size(); // 获取当前窗口大小
    QPixmap scalePixmap =
pixmap.scaled(windowsSize,Qt::IgnoreAspectRatio,Qt::SmoothTransformation
); //讲图片所放到当前窗口的大小
    QPalette palette = this->palette();   //创建 pattle 对象并设置背景照片 调
色板
    palette.setBrush(QPalette::Window,QBrush(scalePixmap));
    this->setPalette(palette);   //将调色板应用到 windows
}

void MainWindow::setButtonStyle(QPushButton * button , const QString &
filename)
{
    button->setFixedSize(136,48);//设置按钮大小
    button->setIcon(QIcon(filename));//设置按钮图标
    button->setIconSize(QSize(button->width(),button->height()));//设置按
钮图标大小
    button->setStyleSheet("background-color:transparent");
}

void MainWindow::setbarButtonStyle(QPushButton * button , const QString &
filename)
{
    button->setFixedSize(30,30);//设置按钮大小
    button->setIcon(QIcon(filename));//设置按钮图标
    button->setIconSize(QSize(button->width(),button->height()));//设置按
钮图标大小
    button->setStyleSheet("background-color:transparent");
}
void MainWindow::initButtons()
{
    setButtonStyle(ui->mode1Btn, ":/Icon/choosebtn.png");
    setButtonStyle(ui->mode2Btn, ":/Icon/flitbtn.png");
    setButtonStyle(ui->mode3Btn, ":/Icon/shrinkbtn.png");
    setButtonStyle(ui->mode4Btn, ":/Icon/rottbtn.png");
    setbarButtonStyle(ui->maxBtn, ":/Icon/maxbtn.png");
    setbarButtonStyle(ui->minBtn, ":/Icon/minbtn.png");
    setbarButtonStyle(ui->closeBtn, ":/Icon/close.png");
```

```
}
void MainWindow::reflash_PicShow(){
    QPixmap pixmapin(FilePath_Out);
    if(!pixmapin.isNull()){
        ui->Pic_label->clear();
        // QSize lableSize = pixmapin.size();
        QSize lableSize = ui->Pic_label->size();
        ui->Pic_label->setPixmap(pixmapin.scaled(lableSize,Qt::KeepAspect
Ratio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行缩放到指定
lable 的大小
        ui->Pic_label->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    }

    QPixmap pixmap1(":/Icon/finish.png");  //设置加载区图片显示位置背景
    if(!pixmap1.isNull()){
        ui->textlabel->clear();
        QSize lableSize = pixmap1.size(); //设置大小为图片大小
        //QSize lableSize = ui->textlabel->size();//设置大小为图标大小
        ui->textlabel->setPixmap(pixmap1.scaled(lableSize,Qt::KeepAspectR
atio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行缩放到指定
lable 的大小
        ui->textlabel->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    }

}
void MainWindow::Button_OpenFile(){  //打开图片文件槽函数，返回这个文件的路径

    QPixmap pixmapbackground(":/Icon/background.png");  //设置图片显示位置
背景浅蓝色
    if(!pixmapbackground.isNull()){
        ui->backgroundlabel->clear();
        //QSize lableSize = pixmapbackground.size(); //设置大小为图片大小
        QSize lableSize = ui->backgroundlabel->size();//设置大小为图标大小
        ui->backgroundlabel->setPixmap(pixmapbackground.scaled(lableSize,
Qt::KeepAspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进
行缩放到指定 lable 的大小
        ui->backgroundlabel->setAlignment(Qt::AlignCenter);//图片居中这个
lable
    }

    FilePath=QFileDialog::getOpenFileName(this,"OpenPicture-File 打开你想
要转换的 BMP 格式文件(不能包含中文路径)",":/Picture","BMP-img(*.bmp)");
    sFilePath = FilePath.toStdString();
```

```
    sFilePath=Process.convertPath(sFilePath);   //转化路径格式为 io 流可以读取的格式 /*  "/"->"\\" */
    //ui->Pic_filepath_textEdit->insertPlainText(FilePath);// 显示打开的图片的路径

    ui->filepath_label->setText(FilePath);
    ui->filepath_label->setAlignment(Qt::AlignCenter);//图片居中这个 lable

    if(!FilePath.isNull()){
        QPixmap pixmapin(FilePath);
        if(!pixmapin.isNull()){
            QSize lableSize = ui->Pic_label->size();   // 获取当前窗口大小
            ui->Pic_label->setPixmap(pixmapin.scaled(lableSize,Qt::KeepAspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行缩放到指定 lable 的大小
            ui->Pic_label->setAlignment(Qt::AlignCenter);//图片居中这个 lable
        }
    }
    FilePath_Out=QFileDialog::getSaveFileName(this,"SavePicture-File 请选择你想要把文件保存在哪里(不能包含中文路径)",":/Picture","BMP-img(*.bmp)");
    sFilePath_Out= FilePath_Out.toStdString();
    sFilePath_Out=Process.convertPath(sFilePath_Out);   //转化路径格式为 io 流可以读取的格式 /*  "/"->"\\" */

    QPixmap pixmap1(":/Icon/finish.png");  //设置加载区图片显示位置背景
    if(!pixmap1.isNull()){
        ui->textlabel->clear();
        QSize lableSize = pixmap1.size(); //设置大小为图片大小
        //QSize lableSize = ui->textlabel->size();//设置大小为图标大小
        ui->textlabel->setPixmap(pixmap1.scaled(lableSize,Qt::KeepAspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行缩放到指定 lable 的大小
        ui->textlabel->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    }

}
void MainWindow::Button_medianFilter(){

    QPixmap pixmap1(":/Icon/doing.png");  //设置加载区图片显示位置背景
    if(!pixmap1.isNull()){
        ui->textlabel->clear();
        QSize lableSize = pixmap1.size(); //设置大小为图片大小
        //QSize lableSize = ui->textlabel->size();//设置大小为图标大小
```

```
        ui->textlabel->setPixmap(pixmap1.scaled(lableSize,Qt::KeepAspectR
atio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行缩放到指定
lable 的大小
        ui->textlabel->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    }

    ui->stagelabel->setAlignment(Qt::AlignCenter);//居中这个 lable
    // QThread::msleep(2000);//阻塞延时 50ms
    Process.medianFilter(sFilePath,sFilePath_Out);//执行 bmp 处理函数

    QPixmap pixmap2(":/Icon/finish.png");  //设置加载区图片显示位置背景
    if(!pixmap1.isNull()){
        ui->textlabel->clear();
        QSize lableSize = pixmap2.size(); //设置大小为图片大小
        //QSize lableSize = ui->textlabel->size();//设置大小为图标大小
        ui->textlabel->setPixmap(pixmap2.scaled(lableSize,Qt::KeepAspectR
atio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行缩放到指定
lable 的大小
        ui->textlabel->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    }
    reflash_PicShow();

}

void MainWindow::on_mode3Btn_clicked()
{
    shrinkWindow = new Shrinkmage(this);
    shrinkWindow->setWindowFlags(shrinkWindow->windowFlags() |
Qt::Window);//在子窗口的构造函数或者显示之前，明确设置 Qt::Window 标志：
    shrinkWindow->show();
}
void MainWindow::setPicWindow(QLabel *imageLabel,QPixmap *pixmap)
{
    imageLabel->setStyleSheet("border:1px solid #ccc;border-radius:6px");
    //QPalette palette = this->palette(); //创建 pattle 对象并设置背景照片 调
色板
    //palette.setBrush(QPalette::Window,QBrush(scalePixmap));
    //this->setPalette(palette);          //将调色板应用到 windows
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

```
void MainWindow::on_mode4Btn_clicked()
{
    rotateWindow = new rotateimage(this);
    rotateWindow->setWindowFlags(rotateWindow->windowFlags() |
Qt::Window);//在子窗口的构造函数或者显示之前，明确设置 Qt::Window 标志：
    rotateWindow->show();

}


//最小化界面
void MainWindow::on_minBtn_clicked()
{
    this->showMinimized();
}

//关闭界面
void MainWindow::on_closeBtn_clicked()
{
    this->close();
}

//放大，缩小界面
void MainWindow::on_maxBtn_clicked()
{
    // static bool max = false;
    // static QRect location = this->geometry();
    // if (max) {
    //     this->setGeometry(location);//回复窗口原大小和位置
    //     ui->maxBtn->setIcon(QIcon(":/MAX_.png"));
    // }else {
    //     ui->maxBtn->setIcon(QIcon(":/minMAX.png"));
    //     location = this->geometry(); //最大化前记录窗口大小和位置
    //     //this->setGeometry(qApp->desktop()->availableGeometry());
    //     this->showFullScreen();  //设置窗口铺满全屏
    // }
    // max = !max;
}


//鼠标按下事件
/*
```

```
 *作用:
 *1.判断是否时左键点击 _isleftpressed
 *2.获取光标在屏幕中的位置 _plast
 *3.左键按下时光标所在区域 _curpos
 */
void MainWindow::mousePressEvent(QMouseEvent *event)
{
    Q_UNUSED(event);
    if (event->button() == Qt::LeftButton)
    {
        this->_isleftpressed = true;
        QPoint temp = event->globalPos();
        _plast = temp;
        _curpos = countFlag(event->pos(), countRow(event->pos()));
    }
}

//鼠标释放事件
/*
 *作用:
 *1.将_isleftpressed 设为false
 *2.将光标样式恢复原样式  setCursor(Qt::ArrowCursor);
 */
void MainWindow::mouseReleaseEvent(QMouseEvent *event)
{
    Q_UNUSED(event);
    if (_isleftpressed)
        _isleftpressed = false;
    setCursor(Qt::ArrowCursor);
}

//鼠标移动事件
void MainWindow::mouseMoveEvent(QMouseEvent *event)
{
    Q_UNUSED(event);
    if(this->isFullScreen()) return;    //窗口铺满全屏，直接返回，不做任何操
作
    int poss = countFlag(event->pos(), countRow(event->pos()));
    setCursorType(poss);
    if (_isleftpressed)//是否左击
    {
        QPoint ptemp = event->globalPos();
        ptemp = ptemp - _plast;
        if (_curpos == 22)//移动窗口
```

```
        {
            ptemp = ptemp + pos();
            move(ptemp);
        }
        else
        {
            QRect wid = geometry();
            switch (_curpos)//改变窗口的大小
            {
            case 11:wid.setTopLeft(wid.topLeft() + ptemp); break;//左上角
            case 13:wid.setTopRight(wid.topRight() + ptemp); break;//右上角
            case 31:wid.setBottomLeft(wid.bottomLeft() + ptemp); break;//
左下角
            case 33:wid.setBottomRight(wid.bottomRight() + ptemp); break;//
右下角
            case 12:wid.setTop(wid.top() + ptemp.y()); break;//中上角
            case 21:wid.setLeft(wid.left() + ptemp.x()); break;//中左角
            case 23:wid.setRight(wid.right() + ptemp.x()); break;//中右角
            case 32:wid.setBottom(wid.bottom() + ptemp.y()); break;//中下角
            }
            setGeometry(wid);
        }
        _plast = event->globalPos();//更新位置
    }
}

//获取光标在窗口所在区域的 列  返回行列坐标
int MainWindow::countFlag(QPoint p,int row)//计算鼠标在哪一列和哪一行
{
    if(p.y()<MARGIN)
        return 10+row;
    else if(p.y()>this->height()-MARGIN)
        return 30+row;
    else
        return 20+row;
}

//获取光标在窗口所在区域的 行   返回行数
int MainWindow::countRow(QPoint p)
{
    return (p.x()<MARGIN) ? 1 : (p.x()>(this->width() - MARGIN) ? 3 : 2);
}

//根据鼠标所在位置改变鼠标指针形状
```

```
void MainWindow::setCursorType(int flag)
{
    switch(flag)
    {
    case 11:
    case 33:
        setCursor(Qt::SizeFDiagCursor);
        break;
    case 13:
    case 31:
        setCursor(Qt::SizeBDiagCursor);break;
    case 21:
    case 23:
        setCursor(Qt::SizeHorCursor);break;
    case 12:
    case 32:
        setCursor(Qt::SizeVerCursor);break;
    case 22:
        setCursor(Qt::ArrowCursor);
        QApplication::restoreOverrideCursor();//恢复鼠标指针性状
        break;
    }
}
```

-----QT 子窗口 1 核心代码------

```
#include "rotateimage.h"
#include "ui_rotateimage.h"
#include "./SourseLib/BMP_Process.h"
#include "mainwindow.h"
#include <QMouseEvent>
#include <QGraphicsDropShadowEffect>
int rotate_angle_int=0;
int xdata_int=0;
int ydata_int=0;
extern BMP_Process Process;
extern QString FilePath,FilePath_Out;
extern string sFilePath,sFilePath_Out;
#define MARGIN 2

rotateimage::rotateimage(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::rotateimage)
{
```

```
    ui->setupUi(this);
    setWindowTitle("设置缩小系数"); //设置窗口标题
    setWindowFlag(Qt::FramelessWindowHint);         //无边框
    setAttribute(Qt::WA_TranslucentBackground);     //窗口透明


    QPixmap labelpixmapin(FilePath);
    if(!labelpixmapin.isNull()){
        ui->label_pic->clear();
        // QSize lableSize = pixmapin.size();
        QSize lableSize = ui->label_pic->size();
        ui->label_pic->setPixmap(labelpixmapin.scaled(lableSize,Qt::KeepA
spectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进行缩放到
指定 lable 的大小
        ui->label_pic->setAlignment(Qt::AlignCenter);//图片居中这个 lable
    }
    QPixmap labelpixbackmapin(":/Icon/glooo130x80.png");
    if(!labelpixbackmapin.isNull()){
        ui->label_pic_Back->clear();
        // QSize lableSize = pixmapin.size();
        QSize lableSize = ui->label_pic_Back->size();
        ui->label_pic_Back->setPixmap(labelpixbackmapin.scaled(lableSize,
Qt::KeepAspectRatio,Qt::SmoothTransformation));  //将图片按照原来的宽高比进
行缩放到指定 lable 的大小
        ui->label_pic_Back->setAlignment(Qt::AlignCenter);//图片居中这个
lable
    }
    ui->closebut->setFixedSize(30,30);//设置按钮大小
    ui->closebut->setIcon(QIcon(":/Icon/close111.png"));//设置按钮图标
    ui->closebut->setIconSize(QSize(ui->closebut->width(),ui->closebut->
height()));//设置按钮图标大小
    ui->closebut->setStyleSheet("background-color:transparent");

    ui->confirmbutton->setFixedSize(77,30);//设置按钮大小
    ui->confirmbutton->setIcon(QIcon(":/Icon/confirmbtnicon.png"));//设置
按钮图标
    ui->confirmbutton->setIconSize(QSize(ui->confirmbutton->width(),ui->
confirmbutton->height()));//设置按钮图标大小
    ui->confirmbutton->setStyleSheet("background-color:transparent");

    connect(ui->horizontalSlider_xdata,SIGNAL(valueChanged(int)),ui->spi
nBox_xdata,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算
    connect(ui->spinBox_xdata,SIGNAL(valueChanged(int)),ui->horizontalSl
ider_xdata,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算
```

```
    connect(ui->verticalSlider_ydata,SIGNAL(valueChanged(int)),ui->spinB
ox_ydata,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算
    connect(ui->spinBox_ydata,SIGNAL(valueChanged(int)),ui->verticalSlid
er_ydata,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算

    connect(ui->dial_angledata,SIGNAL(valueChanged(int)),ui->spinBox_ang
ledata,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算
    connect(ui->spinBox_angledata,SIGNAL(valueChanged(int)),ui->dial_ang
ledata,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算

    connect(ui->confirmbutton,&QPushButton::clicked,this,&rotateimage::B
utton_rotateImage);//按钮链接槽函数 进行缩小图片的计算

    m_parent = static_cast<MainWindow*>(parent);
}
void rotateimage::Button_rotateImage(){
    rotate_angle_int = ui->dial_angledata->value();
    double rotate_angle=rotate_angle_int*1.0;
    xdata_int = ui->verticalSlider_ydata->value();
    ydata_int = ui->horizontalSlider_xdata->value();
    Process.rotateImage(sFilePath, sFilePath_Out, rotate_angle,
xdata_int,ydata_int);
    m_parent->reflash_PicShow();

}
```

-----QT 子窗口 2 核心代码------

```
#include "shrinkmage.h"
#include "ui_shrinkmage.h"
#include "./SourseLib/BMP_Process.h"
#include "mainwindow.h"
#include <QMouseEvent>
#include <QGraphicsDropShadowEffect>
double shrink_xfactor=0;
int shrink_xfactor_int=10;
double shrink_yfactor=0;
int shrink_yfactor_int=10;
extern BMP_Process Process;
extern QString FilePath,FilePath_Out;
extern string sFilePath,sFilePath_Out;
#define MARGIN 2

Shrinkmage::Shrinkmage(QWidget *parent)
    : QWidget(parent)
```

```
    , ui(new Ui::Shrinkmage)
{
    ui->setupUi(this);
    setAutoFillBackground(true);
    setWindowTitle("设置缩小系数"); //设置窗口标题
    setFixedSize(400,225);
    setWindowFlag(Qt::FramelessWindowHint);         //无边框
    setAttribute(Qt::WA_TranslucentBackground);     //窗口透明

    ui->closebtn->setFixedSize(30,30);//设置按钮大小
    ui->closebtn->setIcon(QIcon(":/Icon/close111.png"));//设置按钮图标
    ui->closebtn->setIconSize(QSize(ui->closebtn->width(),ui->closebtn->
height()));//设置按钮图标大小
    ui->closebtn->setStyleSheet("background-color:transparent");

    ui->ConfirmButton->setFixedSize(77,30);//设置按钮大小
    ui->ConfirmButton->setIcon(QIcon(":/Icon/confirmbtnicon.png"));//设置
按钮图标
    ui->ConfirmButton->setIconSize(QSize(ui->ConfirmButton->width(),ui->
ConfirmButton->height()));//设置按钮图标大小
    ui->ConfirmButton->setStyleSheet("background-color:transparent");

    connect(ui->QSLShrinkxFactor,SIGNAL(valueChanged(int)),ui->QSPShrink
xFactor,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算
    connect(ui->QSPShrinkxFactor,SIGNAL(valueChanged(int)),ui->QSLShrink
xFactor,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算

    connect(ui->QSLShrinkyFactor,SIGNAL(valueChanged(int)),ui->QSPShrink
yFactor,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算
    connect(ui->QSPShrinkyFactor,SIGNAL(valueChanged(int)),ui->QSLShrink
yFactor,SLOT(setValue(int)));//按钮链接槽函数 进行缩小图片的计算

    connect(ui->ConfirmButton,&QPushButton::clicked,this,&Shrinkmage::Bu
tton_shrinkImage);//按钮链接槽函数 进行缩小图片的计算

    m_parent = static_cast<MainWindow*>(parent);
}


void Shrinkmage::Button_shrinkImage(){
    shrink_xfactor_int = ui->QSPShrinkxFactor->value();
    shrink_xfactor=shrink_xfactor_int/100.0;
    // shrink_xfactor=0.5;
    shrink_yfactor_int = ui->QSPShrinkyFactor->value();
    shrink_yfactor=shrink_yfactor_int/100.0;
```

```
    // shrink_yfactor=0.5;
    //ui->FactorShow->setNum(shrink_factor_int);  //测试用
    Process.shrinkImage(sFilePath,sFilePath_Out,shrink_xfactor,shrink_yf
actor);
    m_parent->reflash_PicShow();
}
```