浙江大学

C++程 序 设 计

大 程 序 报 告

2018~2019 春夏学期 2019 年 6 月 18 日

1、题目描述和题目要求

设计和实现一款带有图形界面的游戏或其他应用。要求基于 fltk，设计和实现图形界面，至少运用以下技术和知识：

1）. 菜单、按钮、鼠标、键盘、文件；

2） 体现 c++的内容； 给出选题名称，描述特色和功能，填写下表空缺部分，尤其是系统功能部分。菜单、按钮、鼠标、键盘、文件；

1. 考评项		分数	说明
菜单系统		10	菜单系统由上方的菜单、滑条、按钮组成，主要包括文件、图像变换、滤镜、智能功能、边缘检测、图片去噪、形态学处理、帮助几个部分；滑条则主要对应于对比度、亮度、噪声的调节，旋转、左右、上下、大小等图片基本显示调节；按钮包括回撤、前进操作，以及全屏。
图标工具栏		5	菜单中常用功能/命令，不少于 5 个
快捷键		5	与菜单功能对应，不少于 5 个
状态信息栏		5	在窗口底部,即时显示操作的中间状态/结果
功能（35）	图片基本操作	10	图片读取、删除、写入、添加滤镜、旋转、调节大小、左右、上下、回撤前进操作等。
	图片高级操作	15	多图片拼接、人脸识别、人脸美化、图片去噪、形态学处理、边缘检测等。
	窗口界面调节及基本响应	10	图片鼠标绘图、图片改动次数显示、图片鼠标拖动、选取界面颜色、全屏调节自适应大小、文件夹读取等。
链表		5	设计链表用于系统数据的组织
文件	文件读取	10	保存图元信息到文本文件；从文件中读取图形
	多文件组	5	系统程序必须采用多文件组织的方式

	织		
大程序报告	分析和设计	5	软件简介，功能结构，全局、函数及重要算法说明，源程序中功能、函数、文件的组织关系
	部署和运行	5	编译安装、运行测试、用户使用手册
	运行结果	5	效果展示
	分析	5	系统开发亮点和应用知识点总结
总分	100		

选题：基于 FLTK 与 OPENCV 的多功能图像处理软件

特色：使用 FLTK 与 OPENCV 联合编译，FLTK 显示，多功能处理图片

2、需求分析

2.1 功能需求

- 用户通过本软件可以访问本电脑文件夹，选择 JPG 或 PNG 图片格式的图片进行操作。
- 本软件对文件可以实现的操作有：文件的读取、保存、删除；
- 对图像的操作有：图像变换： 线性灰度增强、指数变化、对数变化、直方图均衡化、显示直方图；滤镜：高斯模糊、均值模糊、毛玻璃效果、灰度图、怀旧色；智能功能：人脸识别、人脸美颜、图片拼接、阈值化；边缘检测：canny 算子、sobel 算法；图片去噪：算数均值滤波、集合均值滤波、谐波滤波、中值滤波；形态学处理：腐蚀、膨胀、开操作、闭操作、形态梯度；图片鼠标涂鸦；对比度、亮度、噪声调节；图片旋转；
- 本软件对显示可以实现的操作：全屏显示；图片左右、上下、大小滑条操作；鼠标拖动图片移动；软件背景颜色自由变化；钟表显示；
- 为用户提供帮助文档；

2.2 性能需求

- 系统应保证运行稳定，避免出现崩溃；

2.3 可视化需求

- 用户在保存图片后，系统成功保存后发出提示；
- 系统显示用户对一张图片的操作次数；
- 用户在打开文件夹选择时，系统显示预期图片效果；

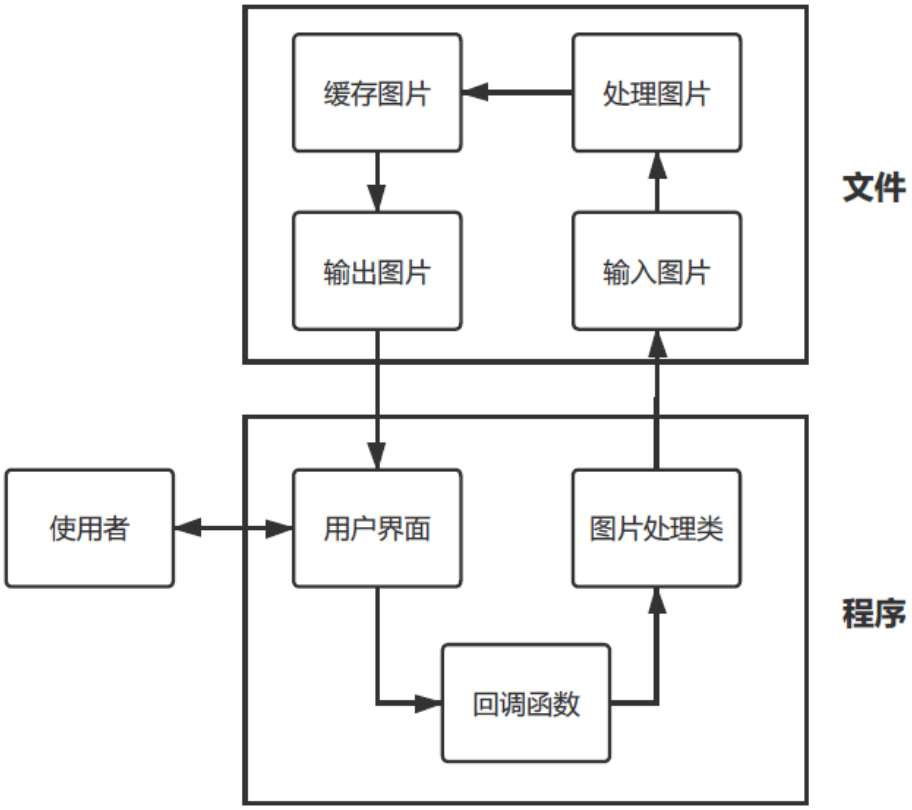
- 用户在进行人脸识别、人脸美颜操作时，若没有检测到人脸，系统发出提示；

2.4 防护需求

- 当文件格式错误时，系统提出警告；
- 当用户没有打开照片直接进行操作时，系统提出警告；
- 撤回前进：当用户处于第一张照片并进行回退，或处于最后一张照片并进行前进时，系统提出警告并停止操作；
- 当用户误操作点击退出时，系统提出警告。

3、总体设计

总体文件与程序交互流程如下，箭头代表信息传输方向。



3.1 功能模块设计

3.1.1 基本文件处理

- **打开图片：**打开文件夹访问用户电脑文件，选择处理的图片（.jpg 或.png 格式），点击确定后，图片会在主界面中进行显示。
- **删除图片：**对现在处理的图片进行删除，同时界面不显示该图片；
- **关闭图片：**主界面不显示目前处理图片；
- **保存图片：**对目前处理的图片进行保存；

- **撤销：**返回图片上次操作状态，同时可以在此状态上进行重做，当前是第一次状态时则无法撤销；
- **前进：**若已经进行过撤销操作，前进可以进入当前状态的下一状态，同时可以在此状态上进行重做，当前是最后一次状态时则无法前进；

3.1.2 界面显示

- **图片显示：**主界面会实时显示正在操作的图片状态；
- **操作次数显示：**左下角显示框显示对当前图片操作次数，若删除则清零重新计数；
- **图片移动：**可以通过鼠标拖拽图片移动，或直接操作滑条左右、上下平移图片。
- **界面大小调整及全屏显示：**可以直接拖拽界面的实现大小变化，点击上方菜单的全屏可实现全屏显示，给用户更舒适的操作体验。
- **界面颜色切换：**点击左侧菜单栏圆形色盘，可进入选色界面，用户可以随意将界面切换成为自己喜爱的颜色。
- **钟表显示：**左侧菜单栏显示有钟表，与系统时间同步显示时间。

开始界面

主界面

3.1.3 图片处理基本功能

3.1.3.1 图像变换

图像增强的目的：除去图象中的噪声,使边缘清晰以及突出图象中的某些性质，提高目标与背景的对比度，增强或抑制图像的某些细节。

- **线性灰度增强**

  图像模糊不清或曝光过度的情况下，可以通过线性灰度增强来对图像内的像素进行线性扩展，改变图片的显示效果。

原图

灰度增强后

- **对数变换**

可以用来扩展图像中的暗像素值，同时压缩亮像素值。

原图

对数变换后

- **指数变换**

指数变换的作用是扩展图像的高灰度级、压缩低灰度级。

原图

指数变换后

- **直方图均衡化**

直方图的图形高低不齐，直方图均衡化就是用一定的算法使直方图大致平和。

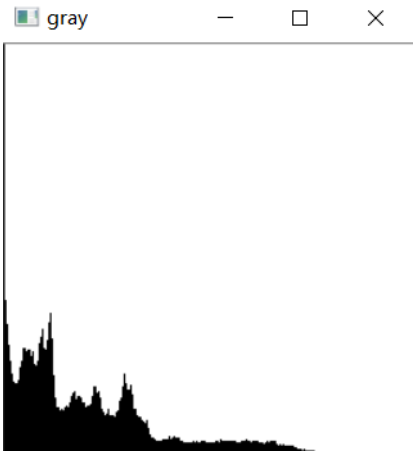
原图

直方图均衡化后

- **显示灰度直方图**

图像的灰度直方图就描述了图像中灰度分布情况，能够很直观的展示出图像中各个灰度级所占的多少。

原图



灰度直方图

3.1.3.3 对比度及亮度调整

$$g(x) = \alpha f(x) + \beta$$

两个参数$\alpha > 0$和$\beta > 0$一般称作增益和偏置参数。我们往往用这两个参数来分别控制对比度和亮度 。把f(x)看成源图像像素，把g(x)看成输出图像像素。即为：

$$g(i,j) = \alpha f(i,j) + \beta$$

其中，i 和 j 表示像素位于第 i 行和第 j 列 。α 可以调整图像的对比度，β 可以调整图像的亮度。

直接控制滑条即可改变图像的对比度与亮度大小。

原图

对比度调整后

原图

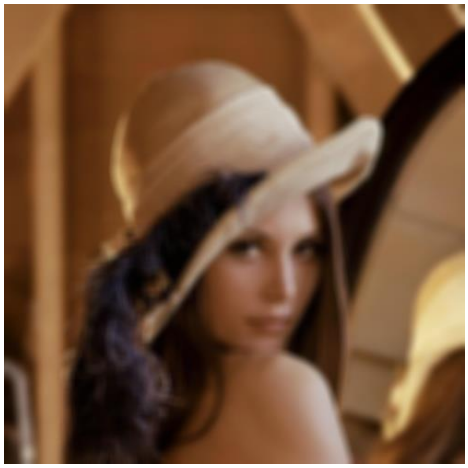
亮度调整后

3.1.3.5 滤镜

- **高斯模糊**

模糊即为拿一个矩阵对原图从左到右从上到下进行卷积，将卷积值赋给当前卷积的中心元素。高斯卷积核矩阵值服从二维高斯函数。

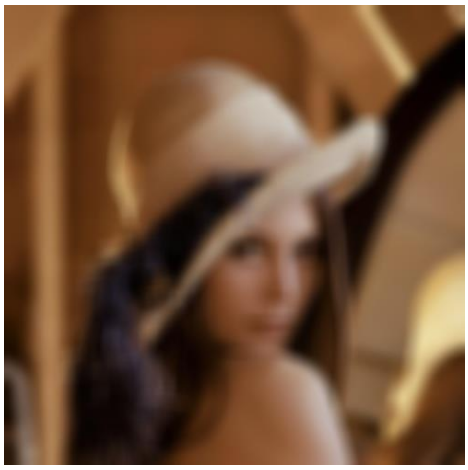
原图



高斯模糊后

- **均值模糊**

均值模糊，也称为均值滤波，相当于卷积核的矩阵值全部为 1/(卷积 SIZE)



原图　　均值模糊后

- **毛玻璃效果**

毛玻璃效果的实现通过用像素点邻域内随机一个像素点的颜色替代当前像素点的颜色实现。使得图像有通过玻璃后观察的视感。


原图　　毛玻璃效果处理后

- 灰度图


原图　　灰度图

- 怀旧色


原图　　怀旧色

3.1.4 图像处理高级功能

3.1.4.1 边缘检测

提供了 canny 算子与 sobel 算法两种边缘检测方法

- **Canny 算子**

Canny 算子边缘检测非常细致

原图

canny 算子边缘检测

- **Sobel 算法**

Sobel 算法倾向于提取清晰的轮廓

原图

x 向 sobel

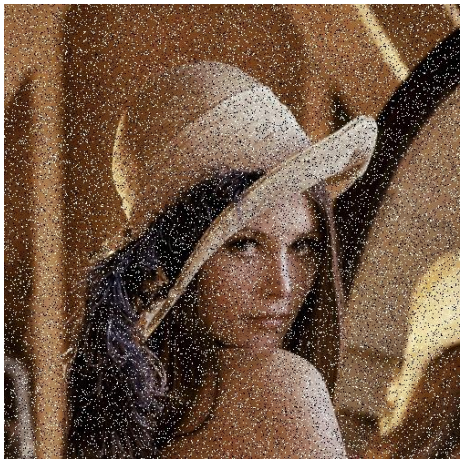
y 向 sobel **xy 向合成后边缘检测**

3.1.4.2 图片噪声&去噪

- **椒盐噪声**

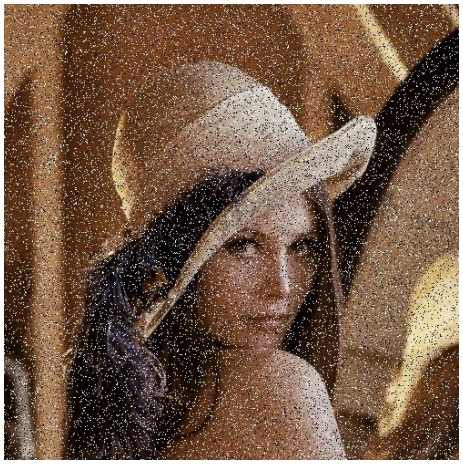
椒盐噪声是一种很简单的噪声，即随机将图像中一定数量的像素点设置为 0（黑）或 255（白）。由于看起来好像在图像上撒了椒盐一样，故被称为椒盐噪声。

原图



15%椒盐噪声

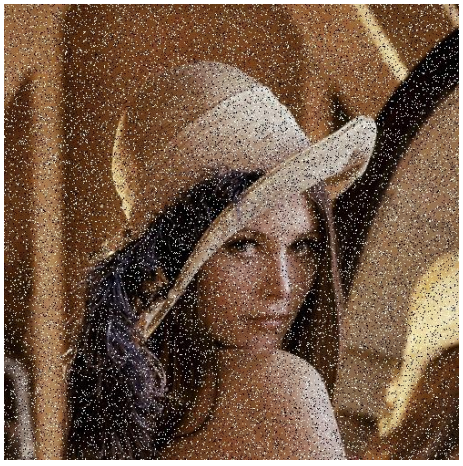
- **中值滤波**



原图

中值滤波后

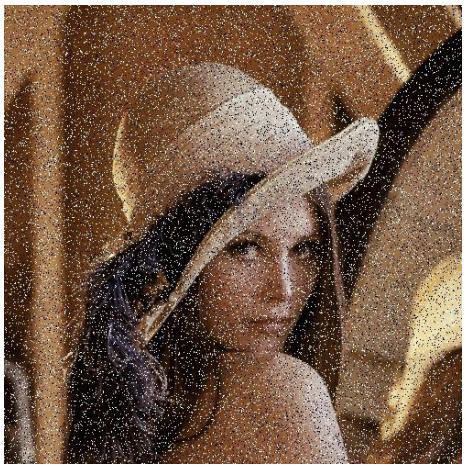
- **谐波滤波**



15%椒盐噪声

谐波滤波后

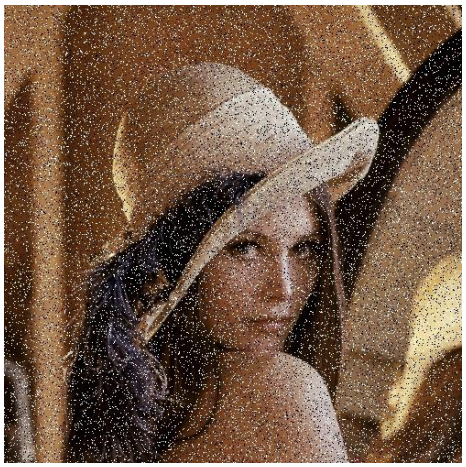
- **几何均值滤波**



15%椒盐噪声

几何均值滤波后

- **算数均值滤波**



15%椒盐噪声

算数均值滤波后

由此可见，对于椒盐噪声，中值滤波的效果最好。

3.1.4.3 形态学处理

- **腐蚀&膨胀**

腐蚀

膨胀

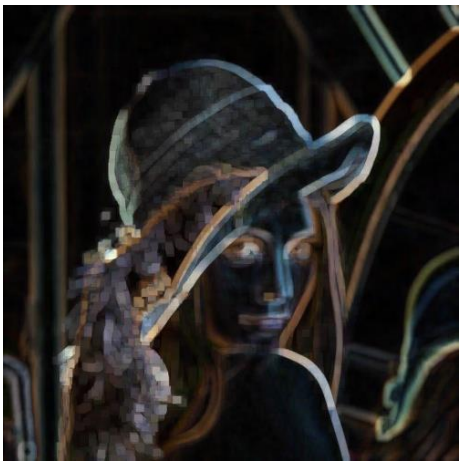
- 开操作&闭操作

开操作

闭操作

- 形态梯度

原图



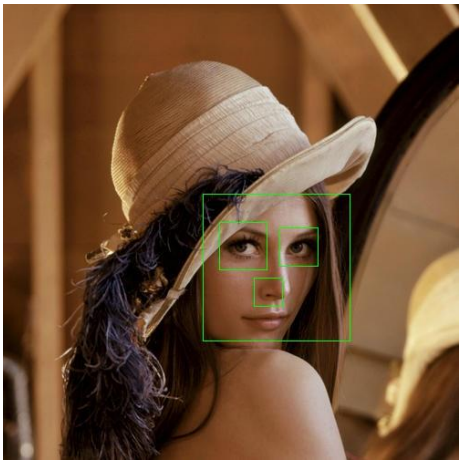
形态梯度后

3.1.5 图像处理智能功能

3.1.5.1 人脸识别

识别人脸及眼睛，并在图片上用绿色框框出来。

原图



人脸识别后

3.1.5.2 人脸美颜

对图像人脸进行识别，并对该区域进行美颜。

原图

人脸美颜后

3.1.5.3 图片拼接

选择另一张图片，则可以与当前图片自动调整大小进行拼接。

原图

图片拼接

3.1.5.4 图片阈值化

原图

阈值化处理

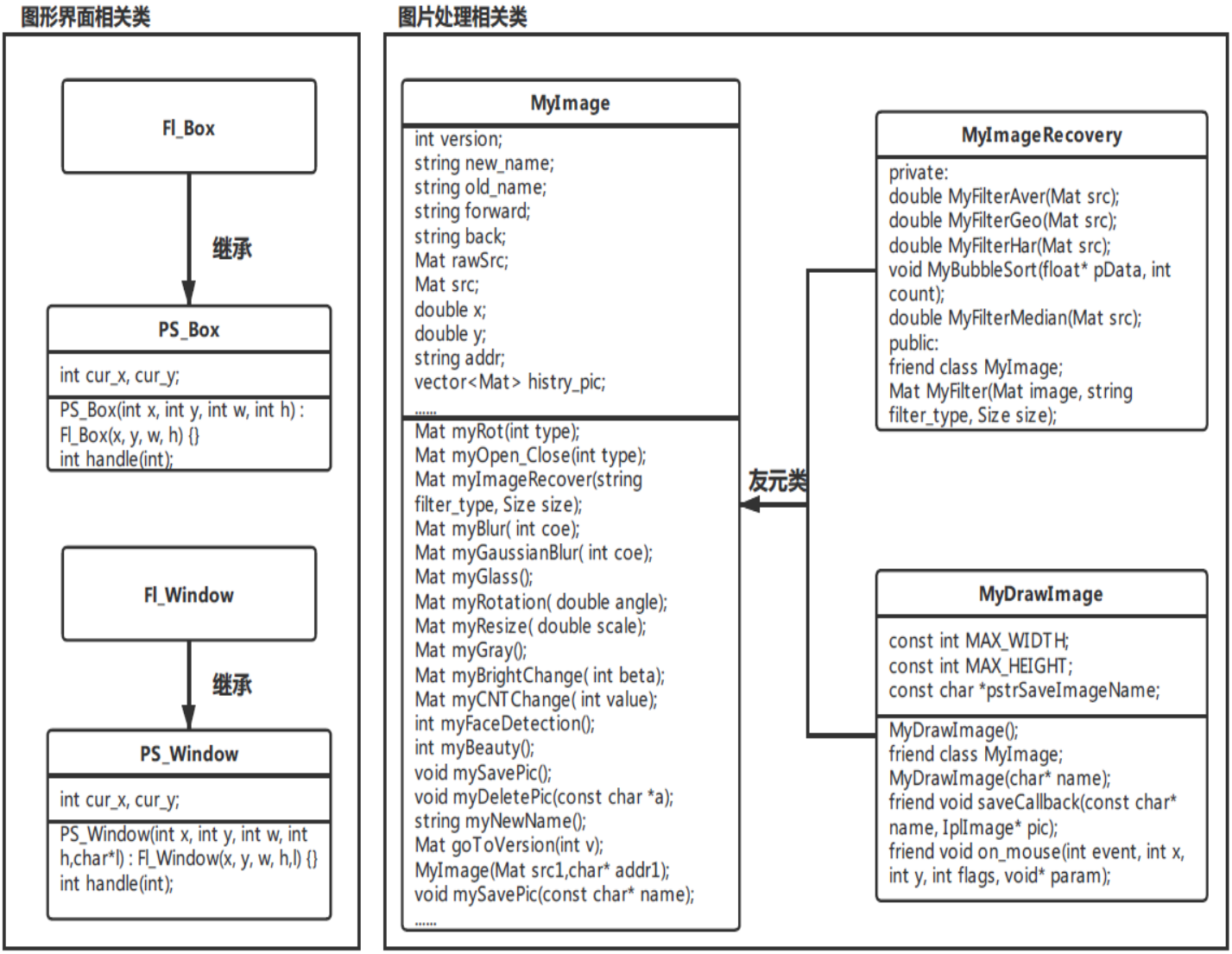
3.1.6 图片涂鸦

原图

涂鸦

3.2 类结构设计

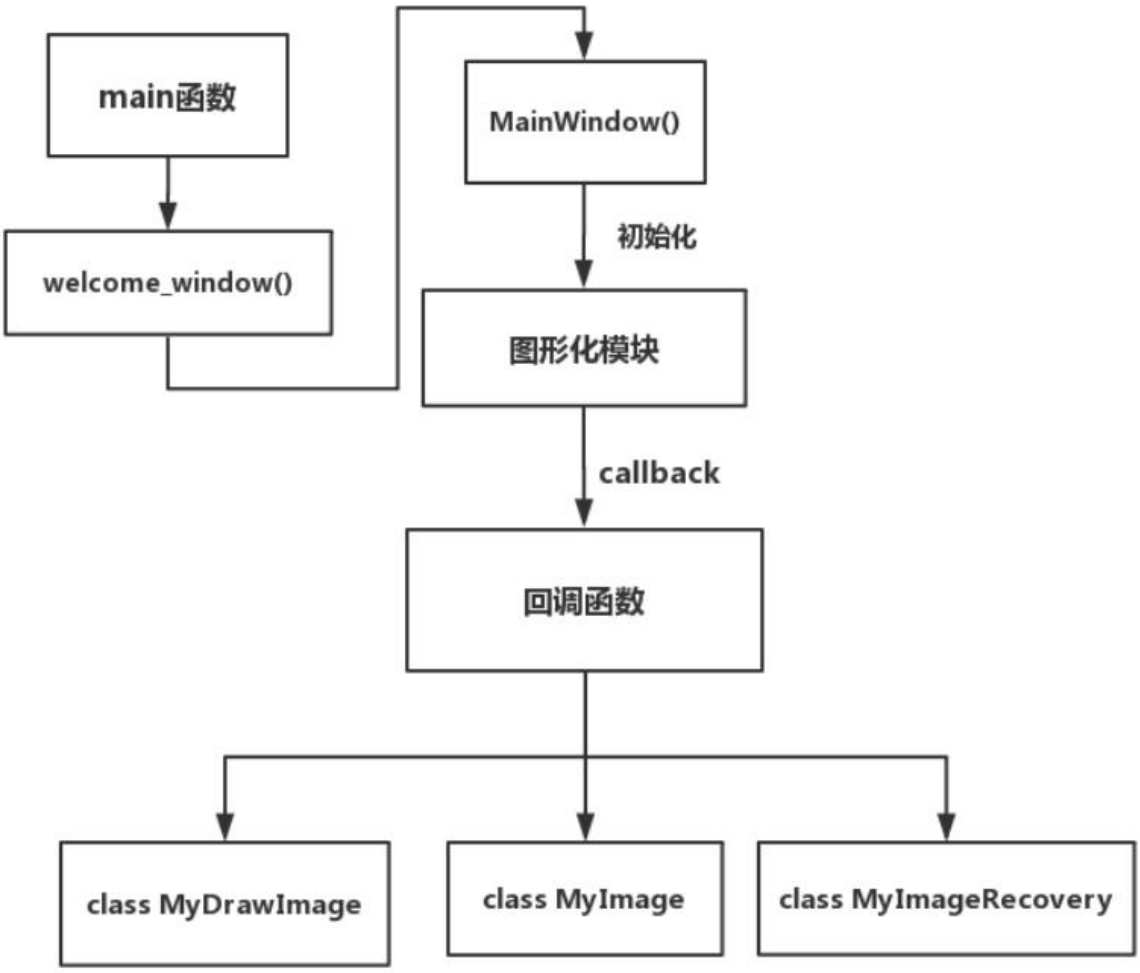
整体类结构设计如下图所示：



图形界面类由 PS_Box 与 PS_Window 类组成，分别继承 Fl_Box 与 Fl_Window，主要起初始化 box 与 window 的作用，因 button 以及 menubar 的 callback 函数不便于封装，故作为单独的函数起到初始化整体界面的作用。

3.3 函数功能描述

整体流程如下：



3.3.1 main 函数

1)函数原型：

```
1. int main(int argc, char **argv);
```

2）功能描述：作为程序运行的入口，初始化开始界面，调用 fl_register_images(); 函数以支持标准图像格式，如 BMP，GIF，JPEG 和 PNG。

3）参数描述：int argc, char **argv。

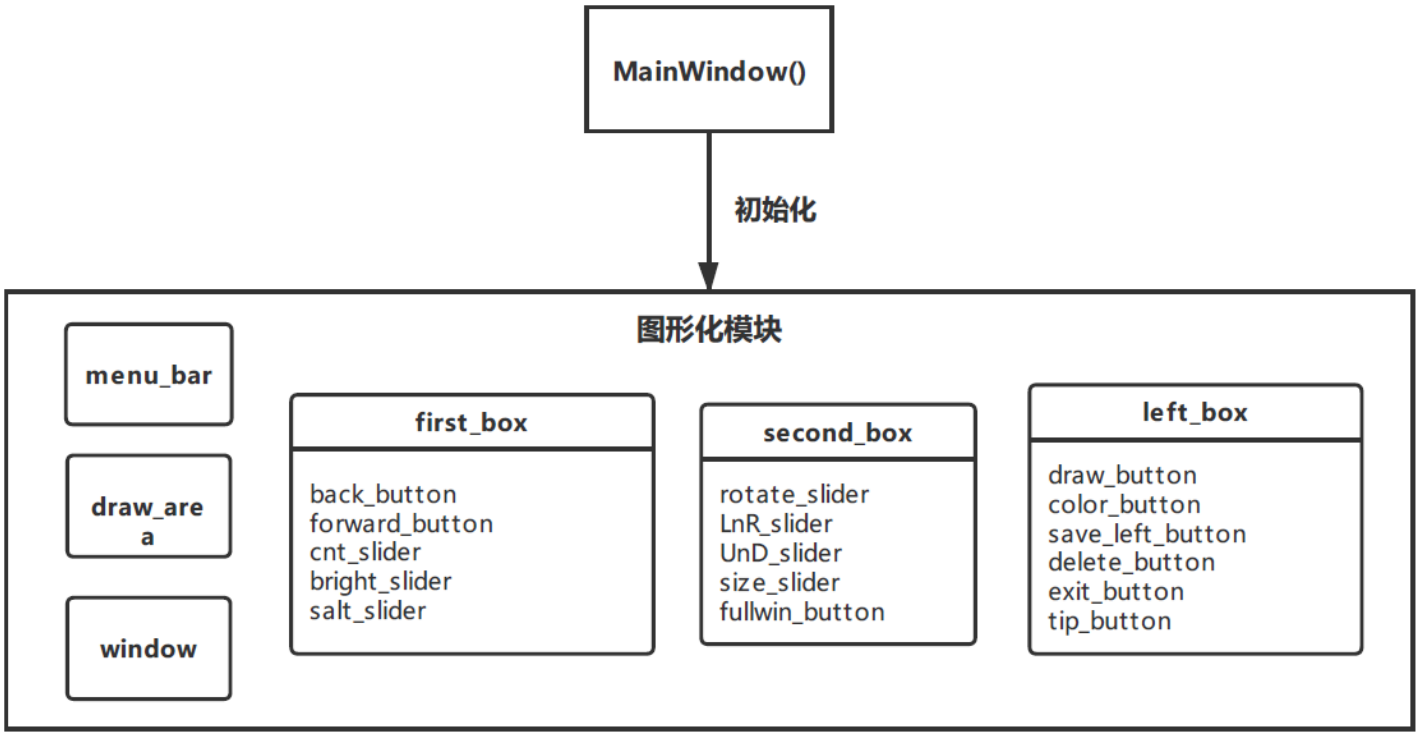
4）返回值描述：使程序一直运行因此 return Fl::run();

3.3.2 图形化界面初始化函数

1)函数原型：

```
1. void MainWindow();
```

2）功能描述：初始化所有图形化界面及各个图形模块。
3）参数描述：无参数
4）返回值描述：无返回值
5）函数算法描述：
整体初始化如下
其中首先初始化 window,作为主要显示窗口,之后由显示底部而上，链接到 fltk_images 库，加载主界面的背景。然后初始化其他 box、button、slider、clock 等组件，并设置其显示模式，调用回调函数。同时设置了窗口大小自适应。



3.3.3 回调函数

因回调函数较多，本文以一按键回调函数与滑块回调函数为例：

1、按键回调函数

1）函数原型：

```
1. void guassblur_pic_cb(Fl_Widget *o, void *);
```

2）功能描述：用户按下高斯模糊功能按键,则回调此函数。
3）参数描述：Fl_Widget *o, void *
4）返回值描述：无返回值
5）函数算法描述

- 首先运行保护机制，确定用户是否已经打开了图片。若没有打开，则调用 attention_pic_cb()函数对用户进行提醒并停止运行该函数，防止在无图片情况下操作导致访问空指针。
- 之后使用MyImage的对象访问其成员函数myGaussianBlur,对图片进行处理。
- 之后对图片进行保存，写入新地址，删除旧地址缓存的图片。同时删除

Fl_Shared_Image 获得改地址缓存的图片，并写入新图片，防止对旧图片的重复显示。

- 重画图片。

2、滑条回调函数

1）函数原型：

```
1. void light_pic_cb(Fl_Value_Slider*o, long);
```

2）功能描述：用户滑动滑条，则可以对图片亮度进行可逆大小调整。

3）参数描述：Fl_Value_Slider*o, long

4）返回值描述：无返回值

5）函数算法描述

- 首先运行保护机制，确定用户是否已经打开了图片。若没有打开，则调用 attention_pic_cb()函数对用户进行提醒并停止运行该函数，防止在无图片情况下操作导致访问空指针。
- 之后使用 MyImage 的对象访问其成员函数 myBrightChange，对图片亮度进行处理，同时使用 o->value()传入对应滑条的数值，进行亮度大小调整。
- 之后对图片进行保存，写入新地址，删除旧地址缓存的图片。同时删除 Fl_Shared_Image 获得改地址缓存的图片，并写入新图片，防止对旧图片的重复显示。
- 重画图片。

3.3.4 文件功能相关函数

1、button 回调函数

1）函数原型：

```
● void file_button_cb(Fl_Widget *, void *);
```

2）功能描述：用户打开文件，可以访问系统文件夹选择文件。

3）参数描述：Fl_Widget *, void *

4）返回值描述：无返回值

5）函数算法描述

- 主要使用 fltk 自带模块 fl_file_chooser 进行文件处理，调用回调函数对用户选择的文件名进行处理。

2、fl_file_chooser 回调函数

1）函数原型：

```
1. void file_cb(const char *n);
```

2）功能描述：fl_file_chooser 的回调函数。

3）参数描述：const char *n 传递了用户选择的文件名

4）返回值描述：无返回值

5）函数算法描述

- 首先判断是否是和已经打开文件相同，若相同则返回；

- 之后运行 load_file 函数，对文件进一步处理；
- 判断打开的是否是图片，如果是则用 Mat 读入图片，并运行 pic_save 函数进一步图形相关操作。

3、文件处理函数

1）函数原型：

```
2. void load_file(const char *n);
```

2）功能描述：对传入的文件名进一步判断及操作，并绘制图片在 box 中，若有老图片则擦除，若图片大于 box 则放缩图片。

3）参数描述：const char *n 传递了用户选择的文件名

4）返回值描述：无返回值

5）函数算法描述

- 首先判断是否已经打开了图片，若有则释放老图片；
- 判断打开的是否为文件夹，若为文件夹则不显示图片且返回；
- 若打开的是图片，使用 Fl_Shared_Image 获得该图片，若图片大于 box，则将图片改变大小进一步放缩。
- 在 window 中绘制图片。

4、图片初始化函数

1）函数原型：

```
1. void pic_save(char *n, Mat img)
```

2）功能描述：对于读入的图片进一步初始化。

3）参数描述：char *n, Mat img 传递了用户选择的文件名与 Mat 图片形式

4）返回值描述：无返回值

5）函数算法描述

- 首先保存传入的 img 图片，运行构造函数进行初始化；
- 初始化图像及图像地址；
- 重新绘制图像

5、重新绘制图片函数

1）函数原型：

```
1. void redraw_pic(Fl_Shared_Image *n);
```

2）功能描述：用户对图像重新操作后，更新主窗口显示的图像。

3）参数描述：Fl_Shared_Image *n , Fl_Shared_Image 类的指针

4）返回值描述：无返回值

5）函数算法描述

- 将显示图片的 draw_area 显示参数指向的图片，传入指针；
- 如果图片版本更新，则删除旧图片缓存；
- 更新全局变量的图片地址。
- 操作次数增加

3.3.5 MyImage 图像处理实现

对于图像处理操作利用面向对象的思想，将整个图像的相关处理封装成为一个类进行操作。同时将图像封装成类来处理，便于添加功能函数，方便进行读写、储存、删除、撤回、前进等文件操作，也提高了程序整体的封装性。

主要使用文件类为 OPENCV 的 Mat 类，以及 IplImage 类进行操作。Mat 类为 OPENCV 中用于矩阵处理的对象，用于表示一个多维度的单通道或多通道稠密数组，可以用来储存灰度图、彩色图等，将图像作为像素值的二维或三维矩阵进行处理。Mat 类一般在 OPENCV2 之后使用较多，而 IplImage 类在 OPENCV1 中经常使用。

MyImage 函数均以 Mat 类为中心进行，对该类进行继承，并添加了对图片处理的相关函数，对一些函数进行派生，添加更多的功能。

```
class MyImage: public Mat {
public:
    /*图片功能*/
    friend class MyImageRecovery;
    Mat myImageRecover(string filter_type, Size size);
    Mat myimageRotate1(  double angle, bool isClip);
    Mat myBlur(  int coe);
    Mat myGaussianBlur(  int coe);
    Mat myGlass();
    Mat myCNTChange(  int value);
    int myFaceDetection();
    int myBeauty();
    Mat myThreshold( double threshold_value, int threshold_type);
    Mat myBlend(Mat src2, double alpha);
    Mat mySalt(double salt_ratio);
    MatND getHistt();
    /*此处省略部分函数*/
    /*文件操作*/
    void mySavePic();
    void myDeletePic(const char *a);
    string myNewName();
    Mat goToVersion(int v);
    MyImage(Mat src1,char* addr1);
    void mySavePic(const char* name);
    int version = 0;
    Mat src;
    Mat rawSrc;
    int MyMaxVersion();
private:
    double x;
```

```
31.     double y;
32.     string addr;
33.     vector<Mat> histry_pic;
34. };
```

1、构造函数

1）函数原型：

```
1. MyImage(Mat src1,char* addr1);
```

2）功能描述：初始化图片类的对象。

3）参数描述：`Mat src1,char* addr1`，传入 Mat 类型的图片，传入 char*类型的图片地址。

4）返回值描述：无返回值

5）函数算法描述

- 将 char*类型的图片地址转换为 string 类型，之后初始化类的部分成员：图片的长宽、历史图片数组 `histry_pic`，图片地址，Mat 类型图片等。

2、析构函数

1）函数原型：

```
1. ~MyImage();
```

2）功能描述：释放图片类的对内存。

3）参数描述：析构函数无参数

4）返回值描述：无返回值

5）函数算法描述

- 释放存储的 Mat 类型的对象；
- 清空存放历史图像的数组。

3、保存图像

1）函数原型：

```
1. void mySavePic();
```

2）功能描述：将图片写入到新地址。

3）参数描述：无参数

4）返回值描述：无返回值

5）函数算法描述

- 使用 stringstream 将图片的版本写入到新地址中，并输出；
- 将图片写入到新地址；

4、删除图像

1）函数原型：

```
1. void MyImage::myDeletePic(const char * a);
```

2）功能描述：删除对应地址的图像。
3）参数描述：const char * a，为对应地址
4）返回值描述：无返回值
5）函数算法描述

- 使用 remove 函数删除对应地址的图像。

5、回退、前进操作

1）函数原型：

```
1. Mat MyImage::goToVersion(int v);
```

2）功能描述：回到任意一个版本的图片。
3）参数描述：int v，为版本数。
4）返回值描述：Mat 类型的图片。
5）函数算法描述

- 对类成员历史图片记录 histry_pic 直接访问下表读取图片即可。

6、图像基本操作

- 旋转：

1）函数原型：

```
1. Mat myRotation(double angle);
```

2）功能描述：对图片进行旋转操作。
3）参数描述：double angle，为旋转角度。
4）返回值描述：Mat 类型的图片。
5）函数算法描述

- 新图像中的像素点位置通过原图像中的坐标计算而来：

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix}$$

对各部分进行取整计算后，填入适当的像素值，即可完成旋转。

- 灰度化

1）函数原型：

```
1. Mat myGray();
```

2）功能描述：将 RGB 彩色图像转化为灰度图像。
3）参数描述：无参数。
4）返回值描述：Mat 类型的图片。
5）函数算法描述

- 灰度值选取为 RGB 三通道的平均值，之后在新图像的对应位置填入灰度值即可得到灰度图。

- 图像模糊

1）函数原型：
均值模糊：

```
1. Mat MyImage::myBlur(int coe);
```

高斯模糊：

```
1. Mat MyImage::myGaussianBlur(int coe);
```

2）功能描述：将图像通过小的滤波算子进行卷积实现低通与高通滤波，以减少噪声。
3）参数描述：滤波模板卷积核的尺寸大小。
4）返回值描述：Mat 类型的图片。
5）函数算法描述

- **均值模糊**：使用滤波器如下：

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- **高斯模糊**：使用滤波器如下：

$$\begin{bmatrix} 0.095 & 0.118 & 0.095 \\ 0.118 & 0.148 & 0.118 \\ 0.095 & 0.118 & 0.095 \end{bmatrix}$$

由上述滤波器完成卷积操作。

- **椒盐噪声及去噪**

1）函数原型：
椒盐噪声：

```
1. Mat MyImage::mySalt(double salt_ratio);
```

去噪：去噪调用了友元类myImageRecover的函数。
友元类myImageRecover声明如下：

```
1. class MyImageRecovery {
2. private:
3.     double MyFilterAver(Mat src);
4.     double MyFilterGeo(Mat src);
5.     double MyFilterHar(Mat src);
6.     void MyBubbleSort(float* pData, int count);
7.     double MyFilterMedian(Mat src);
8. public:
9.     friend class MyImage;
10.    Mat MyFilter(Mat image, string filter_type, Size size);
11.};
```

对应MyImage类中的函数：

```
1. Mat MyImage::myImageRecover(string filter_type, Size size);
```

2）功能描述：给图像加入椒盐噪声，然后通过四种不同的滤波器进行滤波，以平滑图像去除噪声。
3）参数描述：滤波器的类型，卷积核的尺寸。
4）返回值描述：Mat 类型的图片。
5）函数算法描述

- **椒盐噪声：**获得图像的长宽像素值，在其中随机加入盐噪声（白色 255），以及椒噪声（黑色 0），即为随机分布黑白杂点。
- **滤波：**对应用户选择的不同滤波方法，调用友元类中不同的滤波函数。
- **算数均值滤波：**
  使用采样 kernal 函数进行卷积，通常是 3*3 的矩阵：

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- **几何滤波：**公式如下：

$$f(x,y) = \left[\prod_{(s,t) \in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

  滤波后的像素由模板窗口内像素的乘积的 1/mn 幂给出。与算数均值滤波器相比，几何均值滤波器能更好的取出高斯噪声，能够更多保留图像的边缘信息。
- **谐波滤波：**公式如下：

$$f(x,y) = \frac{mn}{\sum_{(x,y) \in S_{xy}} \frac{1}{g(s,t)}}$$

  谐波均值滤波器对于白噪声（盐噪声）效果较好。
- **中值滤波：**
  中值滤波与均值滤波唯一不同之处在于，不是用均值来替换每个像素，二十将周围像素与中心像素排序以后，取中值。

- **图像变换**

1）函数原型：

```
1. Mat MyImage::myLinear_Transform();//线性灰度增强
2. Mat MyImage::myLog();//对数变换
3. Mat MyImage::myIndex();//指数变换
4. Mat MyImage::myEqualize_hist();//直方图均衡化
5. MatND MyImage::getHist1();//获得直方图
```

2）功能描述：对图片进行变换，以达到增强细节、去除噪声等效果。
3）参数描述：无参数。
4）返回值描述：Mat 类型的图片。

5）函数算法描述

- **线性灰度增强：**

  图像模糊不清或曝光过度的情况下，可以通过线性灰度增强来对图像内的像素进行线性扩展，改变图片的显示效果。

  转换公式： $g(x,y)=\frac{d-c}{b-a}[f(x,y)-a]+c$

  其中，f(x,y)为转换前图像，其灰度范围为[a,b]，g(x,y)为转换后图像，其灰度范围为[c,d]。本程序中取 c=0，d=255。

- **对数变换：**

  对数变换可以拉伸范围较窄的低灰度值，同时压缩范围较宽的高灰度值。可以用来扩展图像中的暗像素值，同时压缩亮像素值。

  $$s=c\log(1+r)$$

  其中 c 为常数，r 加 1 可以使函数向左移一个单位，得到的 s 均大于 0。

- **指数变换**

  指数变换的基本表达式为：$y=b^{c(x-a)}-1$

  其中参数 b、c 控制曲线形状，参数 a 控制曲线的左右位置。

  指数变换的作用是扩展图像的高灰度级、压缩低灰度级。虽然幂次变换也有这个功能，但是图像经过指数变换后对比度更高，高灰度级也被扩展到了更宽的范围。

- **直方图均衡化**

  直方图均衡化是灰度变换的一个重要应用，广泛应用于图象增强处理中。图像的像素灰度变化是随机的，直方图的像素灰度变化是随机的，直方图的图形高低不齐，直方图均衡化就是用一定的算法使直方图大致平和。
  本文首先求出图像的直方图，然后算出平均值加在每个像素值上，最后通过 cv::LUT 创建矩阵，把一个像素值映射到另一个像素值上。

- **获得直方图**

  图像的灰度直方图就描述了图像中灰度分布情况，能够很直观的展示出图像中各个灰度级所占的多少。
  图像的灰度直方图是灰度级的函数，描述的是图像中具有该灰度级的像素的个数：其中，横坐标是灰度级，纵坐标是该灰度级出现的频率。
  本文通过 opencv 的 calcHist 函数，计算直方图。

- **人脸识别、美化**

  1）函数原型：

```
1. int MyImage::myFaceDetection();//人脸检测
```

```
2. int MyImage::myBeauty();//人脸美颜
```

2）功能描述：检测出人脸、眼睛，以及对人脸进行美颜。
3）参数描述：无参数
4）返回值描述：int，作为是否检测出人脸的反馈，若没有检测出人脸，将会为用户进行提示。
5）函数算法描述

- **人脸检测：**
    1. 首先对原图像灰度化，以便于进行人脸的识别
    2. 载入分类器（opencv 的一种数据类）
    3. 分类器中载入分类数据文件
    4. 检测人脸位置，并在对应位置绘制矩形。该原理是基于对于图像特征的学习（奇异值分解方法，获取人脸，基于当前人脸与特征人脸向量进行匹配，分析得到人脸位置）
- **人脸美化：**
    1. 和人脸检测相同，首先取出灰度图，并进行线性灰度增强，检测出人脸位置。
    2. 之后对于矩形框出的位置先进行高斯滤波，再双边滤波，再进行高斯滤波，进行非掩膜锐化，相当于对人脸进行去噪以及模糊。

- **形态学处理**

1）函数原型：

```
1. Mat MyImage::myRot(int type);//膨胀与腐蚀：1：膨胀 2：腐蚀
2. Mat MyImage::myOpen_Close(int type);//开闭运算 0：开 1：闭 3：形态梯度
```

2）功能描述：膨胀以为对象增加像素，平滑对象的边缘；腐蚀以为对象减少像素，平滑边缘。开运算实际是先腐蚀后膨胀的过程；闭运算是先膨胀再腐蚀的过程；形态学梯度就是膨胀图与腐蚀图之差，用于保留物体的边缘轮廓。
3）参数描述：int type，传入参数以确定运算。
4）返回值描述：Mat 类型的图片
5）函数算法描述

- **腐蚀与膨胀：**膨胀使用 opencv 自带函数 dilate，腐蚀使用函数 erode；
- **开运算与闭运算：**开运算使用函数 morphologyEx，传入参数 MORPH_OPEN；闭运算使用函数 morphologyEx，传入参数 MORPH_CLOSE。形态梯度则传入参数 MORPH_GRADIENT。

- **边缘检测**

1）函数原型：

```
1. Mat MyImage::myCanny();
2. Mat MyImage::mySobel();
```

2）功能描述：边缘检测是从不同视觉对象中提取有用的结构信息并大大减少要处理的数据量的一种技术。
3）参数描述：int type，传入参数以确定运算。
4）返回值描述：Mat 类型的图片
5）函数算法描述

- Canny 算法：一般分为以下 5 个步骤：
    1. 使用高斯滤波器，平滑图像，消除噪声；
    2. 计算图像中每个像素点的梯度强度和方向；
    3. 应用非极大值抑制，以消除边缘检测的杂散效应；
    4. 应用双阈值检测来确定真实和潜在的边缘；
    5. 通过抑制孤立的弱边缘最终完成边缘检测。
- Sobel 算法：对传进来的图像像素做卷积，即为求出了梯度值；然后对生成的新像素灰度值做阈值运算，来确定边缘信息。
  其中，G_X是对原图 x 方向上的卷积，G_Y是对原图 y 方向上的卷积；

$$G_X = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathrm{A}$$

$$G_Y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathrm{A}$$

  原图中的作用点像素通过卷积之后为：

$$\mathrm{G} = \sqrt{{G_X}^2 + {G_Y}^2}$$

  之后通过平滑处理降低噪声。

3.3.6 鼠标绘图

类结构如下：

```
class MyDrawImage {
public:
    const int MAX_WIDTH = 500;
    const int MAX_HEIGHT = 400;
    const char *pstrSaveImageName = "MouseDraw.jpg";
    friend class MyImage;
    MyDrawImage();
    MyDrawImage(char* name);
    friend void saveCallback(const char* name, IplImage* pic);
    friend void on_mouse(int event, int x, int y, int flags, void* p
aram);
};
```

1、构造函数

1）函数原型：

```
MyDrawImage::MyDrawImage(char* name);
```

2）功能描述：初始化 OPENCV 的 window，读入正在主窗口操作的图片。同时设置鼠标回调函数，对应不同的事件进行回调，进而实现绘图功能。同时对于用户输入的不同有不同的响应。
3）参数描述：图片地址。
4）返回值描述：无返回值。
5）函数算法描述：

- 首先初始化窗口，显示图片。
- 调整窗口大小为图片大小，设置鼠标回调函数。
- 响应键盘输入值。对于输入 r，进行清空，屏幕设置为白色；对于输入 s，保存绘制的图片；

2、鼠标回调函数

1）函数原型：

```
1. void on_mouse(int event, int x, int y, int flags, void* param);
```

2）功能描述：对于鼠标事件进行响应，进而实现绘图功能。
3）参数描述：int event, int x, int y, int flags, void* param。
4）返回值描述：无返回值。
5）函数算法描述：

- 使用 switch-case，对于鼠标按下时，在(x,y)处绘制点；
- 鼠标松起之后，停止绘制；
- 对于鼠标移动情况，若鼠标按下，则绘制，否则停止绘制。

3.3.7 图片移动

类声明：

```
4    class PS_Box : public Fl_Box {
5        int handle(int);
6    public:
7        int cur_x, cur_y;
8        PS_Box(int x, int y, int w, int h) : Fl_Box(x, y, w, h) {}
9    };
```

1）函数原型：

```
1. int PS_Box::handle(int e);
```

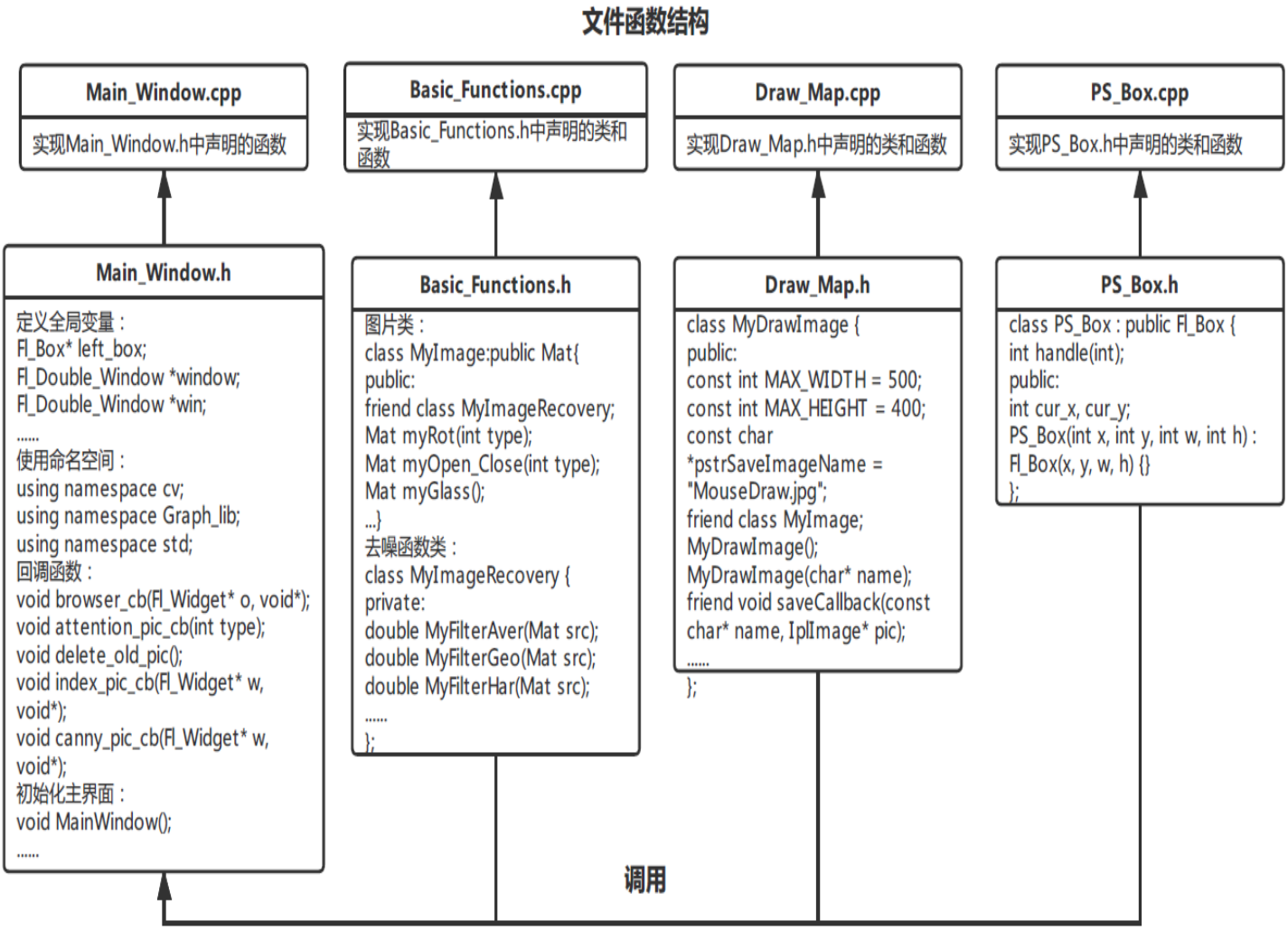
2）功能描述：继承 Fl_Box,派生 handle 函数，实现鼠标拖动图片移动。
3）参数描述：鼠标事件
4）返回值描述：int 型
5）函数算法描述：

- 若鼠标按下则记录当前的坐标作为初始坐标，初始化成员变量。
- 若鼠标拖动，则改变 box 的位置，以实现图片位置的变化。
- Window 重新绘制，擦除旧图像。

- 如果鼠标释放则直接显示。

3.4 程序文件结构

1）文件函数结构
文件函数结构如下图所示：



注：因使用 fltk 底层库，由于作为 ui 界面有大量按键回调函数不易于封装，因此使用了全局变量主要如下：

```
1.  Fl_Box* left_box;
2.  Fl_Box *welcome_box;
3.  Fl_Double_Window  *window;
4.  Fl_Window *wel_window;
5.  Fl_Shared_Image *img;
6.  PS_Box *draw_area;
```

2）多文件构成机制

- 本程序主要由 Main_Window.cpp，Main_Window.h，Basic_Functions.cpp，Basic_Functions.h，Draw_Map.cpp,Draw_Map.h，PS_Box.cpp，PS_Box.h，以及作为帮助文档的 txt 文档，jpg 或 png 格式的图片构成；
- Main_Window.cpp 调用了其他几个.h 文件，其他的 cpp 文件均为对应的.h 文件的实现；
- Txt 文档在 Main_Window.cpp 中被读入，作为帮助文档；
- 图片由 Fl_Shared_Image::get 读入，或由 OPENCV 的 imread 函数通过地址读入。

4.运行结果

4.1 编译安装运行说明

4.1.1 IDE 介绍

本项目使用的 IDE 为 Visual Studio 2015 Community，区别于 Professional 和 Enterprise 两个版本。Microsoft Visual Studio（简称 VS）是微软公司的开发工具包系列产品，包括了软件生命周期所需要的大部分工具，如 UML 工具、代码管控工具、集成开发环境等，所写的目标代码适用于微软支持的所有平台，包括 Microsoft Windows、Windows Phone、Windows CE、.NET Framework、.NET Compact Framework 和 Microsoft Silverlight。Visual Studio .NET 是用于快速生成企业级 ASP.NET Web 应用程序和高性能桌面应用程序的工具。VS 包含基于组建的开发工具（如 Visual C#、Visual J#、Visual Basic 和 Visual C++），以及许多用于简化基于小组的解决方案设计、开发和部署的其他技术。

4.1.2 Compiler 介绍

MSVC(Microsoft Visual C++)是 Windows 下的使用最广泛的主流编译器，是专为 Windows 平台开发的一套工具集，支持 C、C++/CLI 等编程语言。MSVC 集成了便利的除错工具，尤其是微软 Windows 视窗操作系统应用程序接口（Windows API）、三维图形 DirectX API、Microsoft .NET 框架。

MSVC 与另外几款编译器的比较：

- GCC(GNU Compiler Collection)：一套由 GNU 开发的编程语言编译器，在所有平台上都使用同一个前端处理程序，产生一样的中介码，通常是跨平台软件的编译器首选，几乎支持 99%的 C/C++标准的内容。
- ICC：没有专门的前端，而是针对 Intel 体系结构上的专门优化。
- Clang：一个 C、C++、Objective-C 和 Objective-C++编程语言的编译器前端。它采用了底层虚拟机(LLVM)作为其后端，目标是提供一个 GNU 编译器套装(GCC)的替代品。特性是编译速度快，内存占用小，兼容 GCC，设计清晰简单、容易理解，易于扩展增强；基于库的模块化设计，易于 IDE 集成及其他用途的重用；出错提示更友好。
- MSVC：Windows 平台上最常用的编译器，常被人诟病的是对标准的支持不够新不够快，随着微软发布基于 Clang/C2，MSVC 与 Clang/C2 将并行。

4.1.3 使用库介绍

4.1.3.1 FLTK

FLTK(Fast Light Tool Kit)是一种使用 C++开发的 GUI 工具包。

它可以应用于 Unix,Linux,MS-Windows95/98/NT/2000 和 MacOS 操作系统平台，相对于其它的许多图形接口开发工具包（如 MFC，GTK，QT 等），它具有体积很小、速度比较快，且有着更好的移植性的优势。并提供了如下功能：

- 提供丰富的跨平台 GUI 构件(Widget)。有按钮，菜单，窗口等。
- 支持 OpenGL，提供 FI_GL_Window，支持 OpenGL 相关的操作。
- 提供界面设计工具 FLUID，非常方便进行界面的设计。
- 良好的跨平台移植性。
- 支持多种 C++编译器，Gcc，BC，VC 等等。

4.1.3.2 OPENCV

Opencv 是 Intel 开源计算机视觉库。

它由一系列 C 函数和少量 C++类构成，实现了图像处理和计算机视觉方面的很多通用算法。其拥有包括 300 多个 C 函数的跨平台的中高层 API，且不依赖于其它的外部库。 并提供了如下功能：

- 提供丰富的函数，强大的图像和矩阵运算功能。
- 良好的平台无关性
- 程序运行的实时性
- 具有方便灵活的用户接口
- 支持 Ch SDK 等扩展
- 统一的结构和功能定义

4.1.4 环境搭建

4.1.4.1 FLTK

- 下载源码包（版本 v1.3.4），并解压到对应目录下。
- 由于 FLTK 是跨平台的 C++ GUI 库，以源代码形式发布，因此在使用前需要编译。将 FLTK 编译成静态链接库的形式使用。打开 fltk.sln，使用 VS2015 进行编译。

- 打开 VS2015 的 VC++目录，将编译生成的静态链接库文件拷贝到 VC++的 lib/x86 目录下，将 FL 文件夹拷贝到 include 目录下。

- 定义 VS2015 的预处理器为 WIN32，并添加依赖库。

4.1.3.2 OPENCV

- 下载并解压 opencv(版本 2.4.13)，并新建 VS2015 控制台应用程序。
- 配置管理器为 x86(32 位)。
- 设置系统环境变量，添加.\opencv2.4\opencv\build\x86\vc12\bin 至环境变量。
- 添加 build\include 目录及其子目录为包含目录，并将 x86\vc12\lib 添加为库目录。
- 添加附加依赖项。（依赖项见附录）

4.2 典型测试情况

（选取测试阶段典型的案例，说明如何设计测试数据，发现和定位错误的，测试结果可以含有屏幕截图。）

4.2.1 回撤和前进错误

在测试阶段，发现回撤后继续操作时，图片总会先显示之前操作的结果，再继续新的操作。开始怀疑图片存储读取机制具有问题，而经过不断调试与查找资料后发现，是因为读取图片 Fl_Shared_Image::get 具有缓存功能，对于一个地址的图片，如果读入没有释放的话，会无法读入新图片。因此在即将读入新图片时，需要对原来的地址存储的图片进行释放，才能读入新图片，正确显示回撤后新的操作。

4.2.2 亮度、噪声滑条控制错误

在测试阶段，发现控制滑条移动后（以噪声条为例），如果滑条滑回原来位置，不会返回原来位置对应值的效果，而会进一步叠加。这与我的图片操作机制有关系，因此每次在噪声的操作时，会在总版本数之外，增加一个新成员噪声控制版本数，如果噪声控制版本数和总版本保持一致，说明此时用户在连续操作滑条。同时增加一个新成员 Mat 图片形式，保存连续操作之前的原始图片。当检测到用户在连续操作时，直接在原始图片上操作，而不是延续上一步操作，以实现滑条回退撤销的效果。

5. 总结

5.1 程序亮点或创新之处

5.1.1 程序亮点之处：基于 fltk 的图形化界面，与 opencv 联合编译，利用了面向对象的思想，继承了 opencv 的图像类 Mat，构造一个可以对图像读入、储存、删除、修改的类 MyImage，同时与 fltk 读取图片的类相结合，实现了对图片的丰富操作。

5.1.2 程序创新之处：

本程序创新之处在于 opencv 与 fltk 文件读取、存储的实现。实现思路如下：

1、图片版本：图片版本是暂存图片的标志，随着修改次数递增，图片命名即为原图片+版本数，原始图片版本为 0。同时图片版本将显示在界面上。

2、地址的获取：MyImage 中有函数生成对应当前版本数的路径地址。

2、图片打开与存储：对于主界面中图片的显示，使用了 fltk 的 Fl_Shared_Image 类，然后图片地址传入 MyImage 图片类，使用 opencv 的 imread 函数读取图片地址进而获得图片，并转化为 Mat 类型的图片。在结束操作后，根据生成的新地址写入图片，同时删除老的缓存图片（保存图片除外），然后 fltk 的 Fl_Shared_Image 进一步读取新的地址与图片并进行显示。

3、图片修改历史：图片修改历史通过 MyImage 类中类型为 Mat 的 vector 数组进行储存。

4、撤销重做：通过下标直接访问图片修改历史达到目的。

5.2 应用知识点　：

继承与派生：图像处理类 MyImage 对 Mat 进行继承以及派生；PS_Box 类对 Fl_Box 进行继承以及派生。

友元函数：图像处理类之间互为友元类，并且互相调用。

使用容器：使用 Mat 类型的 Vector 对历史图像进行储存。

虚函数及重载：对 handle 函数进行重载，以实现新的系统响应功能。

5.3 心得体会

本次大作业，我独立完成了一个图像处理软件。基于 FLTK 以及 OPENCV 库，运用了大量面向对象的知识，学习了很多图像处理领域的相关算法与知识，收获非常多。

6、参考文献和资料

（列出参考的书籍、论文、网站的信息和地址等。）

参考网站：

https://blog.csdn.net/baimafujinji/article/details/50500757

https://www.jianshu.com/p/2a06c68f6c14

https://blog.csdn.net/zhu_hongji/article/details/80984341

https://blog.csdn.net/csdnforyou/article/details/82216301

https://blog.csdn.net/zhangqipu000/article/details/53260647