C++个人报告

学号：3160104062

姓名：刘涵宇

一、基本任务

1.小组任务

我们小组选择的工程任务是实现一个图片处理软件，能够支持传入传出图片，对传入的图片进行特色化处理并保存，有基本的UI框架并实现实时的预览效果。小组的主要任务包括UI层的实现，图像算法的实现，以及对产品架构的设计把控。

2.个人任务

我的任务是编写图像算法，在core层与model层之间工作。我们的工作尽量减少了相互协调的部分，在core层部分我只需要测试好每一个算法的功能可用，并把它们封装成为静态库，把头文件、静态库、源码分别保存好。在model层内为每一个算法滤镜注册名，并写好API，以便更高结构调用这个算法滤镜。

二、接口设计

1.core

算法基本通过对Mat型变量内部的点变换来实现，具体的内容都是OpenCV的一些简单算法，在这里不做赘述。由于对传入传出变量的统一规定是Mat型变量，可以参考的几组设计方式有封装成滤镜类、封装成滤镜函数两种方式。考虑到我们的代码存在迭代，因而我希望能够从设计中体现出我们的滤镜更新，即一代滤镜类的更新。因而统一的滤镜类如下：

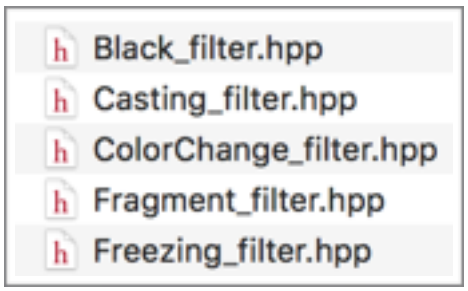
```
class OldMovie_filter {
public:
    OldMovie_filter() {}
    ~OldMovie_filter() {}

        double generateGaussianNoise(double mu, double sigma);//生成随机高斯噪点
        Mat AddNoise(Mat img, double mu, double sigma, int k);//图片添加高斯噪点
        Mat shift(Mat imge, int dir, int offset);//移动图像(FRAGMENT内置函数)

        Mat ColorChage(Mat img);//生成怀旧色图像
        Mat OldImage(Mat img);//生成怀旧色+噪点图像
        Mat OldMovieImage(Mat img);//生成怀旧色+噪点+暗边图像
        Mat FixedVignetting(Mat img, int color);//生成添加白色毛边或黑色毛边图像
        Mat FragmentImage(Mat imge, int Offset);//生成碎片模糊图像
        Mat FragmentImage(Mat imge);//缺参碎片模糊图像
        Mat FragmentLomoImage(Mat img);//生成碎片模糊+暗边效果图像
        Mat OldPaperImage(Mat img);//生成怀旧色+噪点+白边图像
};
```

后来发现这样的调用并没有节省编译时间，因为在model层无论调用哪一个滤镜都要生成一个新的类才能实现，这样反而容易弄混淆每一个滤镜的工作，因为它们都是属于一个类下的。

为了实现滤镜封装调用，并且方便以后添加新的算法，后面的滤镜算法我都是一个滤镜一个类进行封装的。这样的好处是对于上层在加载这个类的时候就能够迅速了解到这个滤镜是做什么的，不会出现调用错误的情况。



其实直接的封装函数也可以，但我比较喜欢类的封装特点。

接口设计基本从简，除了三个辅助函数：高斯模糊、高斯噪点、图像移动，其他的滤镜算法的接口基本都只需要输入一个Mat型就可以输出一个Mat型修改好的图片。在上层调用滤镜函数之前，需要创建一个相应的函数类。

```
class Black_filter{
public:
    Black_filter(){};
    ~Black_filter(){};
    Mat BlackImage( Mat img );
};
```

2.model

model主要为更上层的view服务，这一层由我和上层的同学合作完成，我只需要在写好算法之后在model层为每个算法写一个布尔函数来使用它。布尔函数内不需要传入Mat的参数，因为对于函数内的Mat型的直接调用私有类里当前编辑的图片；但其他类型的变量需要传入参数，比如对高斯模糊等函数：

```
bool gsfilter(double,int);
bool addnoise(double, double, int);
bool shift(int, int);
```

三、实现细节

图像算法一共实现了二十六种滤镜效果，每个效果经本地测试可用之后上传至仓库。为了确保程序可跨平台及配置正确，我先用Mac的Xcode平台搭载了OpenCV的环境，并在Xcode上编译通过后，在VS2017平台再次配置编译，发现可以运行。再通过VS2017生成静态库后，配置静态库编译运行，再次测试运行，可以运行后上传至仓库。

图像算法中一部分是色彩变幻的函数，实现的方式比较简单，只需要获取Mat型里的ptr堆，对里面每一个点的RGB颜色进行演算就可以了。只要知道颜色的变换函数，实现每个图像的变化都只需要O(N^2)的时间就可以。

还有一部分算法是用到了点的位置变化以及叠加。我发现到Mat型+Mat型这样一个重载运算符运算可以产生非常多有趣的效果，因而用它和反色等自带函数做了一些模糊、加边框的效果。

四、实验心得

这次的小组工程不同于以前我做的常规的C++工程，在这个学期我们学习了C++的面向对象特性之后，在面向对象程序设计课程、数据库原理、计算机图像学设计的课程作业中均有涉及到C++的小组编程项目。除了数据库原理有老师提供的非常明确的可参照分工之外，其他的两个工程的结构体系都非常的混乱，导致我们对小组分工非常不明确，往往都是一个部分的人不能做自己的部分，因为另外一个人的部分还没有做好，导致时间无法充分利用。

在这门课里我理解到了集体编程并不是一个纯线性的工作，它有很多需要考虑的问题。首先，我们在工程开始的阶段就与之前的小组工作完全不同，这款软件的制作过程中我比以往都有把握。由于这款软件的功能都需要我们自己决定，在最初开始阶段我们并没有对技术难度有直观的认识，很多的技能都没有掌握。因而我们的开始阶段着手于对可行性的分析、学习新的技术，同时我们根据学习程度来调节已经制定好的需求目标。这个过程对我的影响非常的大，因为很多时候我们做小组工程都喜欢“画饼”，在不考虑到难度和可行性的情况下，一味地考虑工程的美观强大，最后往往完不成这样的任务，反而降低了制作人员的工作积极性。但其实很多技术都是逐步提高的，我在最初甚至搭建不出来一个可以运行OpenCV的编译平台，但在后期我就可以娴熟地在ios平台上编写代码并调试，再将代码在win10平台上的Visual Stiduo上打包成静态库，并且全部测试运行成功。这些技能对我来说都是一步步更新的，如果一开始告诉我我需要用OpenCV写一个滤镜并打包成静态库，在我完全不了解的时候我会认为这个任务过于复杂而无法达到。但当我学习了一段时间OpenCV并且对Visual Stiduo的功能有了一定了解之后，这项工作对我来说就非常简单了。

其次，也是最重要的一点，这次的小组工作让我知道了，小组分工其实也可以很明确很简单，不需要互相等待，只需要完成自己的部分就可以了。回想数据库的miniSQL实验，我们之间的接口约定都非常地明确，所使用的数据结构类型也是事先约定好的，因而模块的交接非常地快。换言之，我们在做项目时其实不需要知道对方做了什么，只需要把这一部分实现写在自己的代码里就好了。

因而我们在工程结构上采用了MVVM结构，分为Model、View、ViewModel部分。这样的分工尤其是对界面设计的人员友好了，界面实现了独立测试。

最后，这次的小组工程我得到的体验非常的好，因为我们组长的经验非常丰富。在我们一起从事工作的这段时间中，很多时候我明明在学习这项技能，却始终无法推动进程。比如最开始由于种种平台配置的问题，我用了很多方法去装一堆软件，去配置每一个软件的编译环境，经常做到一半的时候抛锚，不知道自己在干什么了。这个时候总是组长在我身边询问我做到了哪一步，接下来要做什么，明确了我的工作状态，提醒我工作进程。其实对于很多小组项目来说，就是缺乏这样一个灵魂人物来催促每一个人，告诉每个人应该做什么，大家的进度才会得不到保障。因而我意识到，产品经理这样一个角色也是一个不可或缺的灵魂人物，可能他并不从事主要的编程工作，相对于其他的人来说看上去要轻松许多，但他对于整个工作的意义是其他人不可以代替的。