

# 项目三报告

12313605 吕梓翀

2025 年 5 月 11 日

## 目录

<b>1 项目概览</b>	<b>2</b>
<b>2 功能演示</b>	<b>2</b>
2.1 运行环境 . . . . .	2
2.2 库使用说明 . . . . .	3
2.3 图片读写 . . . . .	3
2.4 图片预览 . . . . .	4
2.5 亮度调整与对比度调整 . . . . .	4
2.6 反转 . . . . .	5
2.7 混合图像与重塑大小 . . . . .	6
2.8 剪切 . . . . .	7
2.9 灰度化 . . . . .	7
2.10 错误处理 . . . . .	8
<b>3 功能实现</b>	<b>9</b>
3.1 数据结构 . . . . .	9
3.2 图像处理功能实现 . . . . .	10
3.2.1 图像读取与写入 . . . . .	10
3.2.2 亮度调整 . . . . .	11
3.2.3 混合 . . . . .	11
3.2.4 图片预览 . . . . .	11
3.2.5 比例重构 . . . . .	12

<b>1 项目概览</b>	<b>2</b>
<b>4 科学优化</b>	<b>12</b>
4.1 并行优化 . . . . .	12
4.2 SIMD . . . . .	13
4.3 编译器的优化 . . . . .	14
4.4 避免索引的重复计算 . . . . .	14
4.5 性能测试 . . . . .	14
<b>5 总结</b>	<b>14</b>
5.1 AI工具 . . . . .	14
5.2 总结的总结 . . . . .	15

## 1 项目概览

本次项目是一个C++语言实现的图像处理库，这是我第一次动手做一个库，也是第一次大规模使用C++的类与对象特性进行编程的经历。在编写的过程中，先前学过的一些特性，在本次项目的编写与优化的过程中，我都得到了更深刻的理解与体会。相比于项目三实现的四个功能，本次我又添加了一些新的功能。我在编写的过程中，多处参考了开源图像处理库OpenCV的设计，以期获得更加优雅的结构和更加优秀的性能。由于JPG,PNG图片涉及到一些压缩的处理，有些太复杂了，我并没有在项目中加入，我支持了五种最常见的BMP图像：32位真彩图，24位真彩图，8位彩色图，8位灰度图，黑白图。下面我将就项目本身展开演示与报告。

## 2 功能演示

### 2.1 运行环境

本次项目是用C++语言编写的。使用Intel i7的处理器，通过WSL在Linux系统下运行。通过Cmake构建动态链接库，然后在程序编译时使用命令连接库：

```
1      g++ demo.cpp -o demo -I./MyImageLib/include -L./
          MyImageLib/build -lMyImageLib -Wl,-rpath=./build
```

动态库的结构如下图所示：build中为动态库和中间文件，src中为库文件，include中为头文件

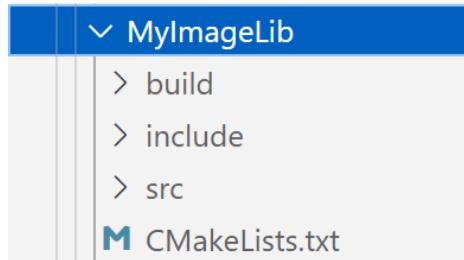


图 1: 动态库结构

## 2.2 库使用说明

在程序中，调用函数help();可以快速地得到该库的简单使用说明。通

```
@ lvzichong@LAPTOP-VH76P8LQ:/mnt/d/C_and_Cpp/project/project_4/src$ ./demo
[Function Overview]
Supports the following image processing features, applicable to BMP images (without OpenCV):
1. imread(filename)           // Read image from BMP file, returns a Mat object
2. imwrite(filename, Mat)     // Write a Mat image to a BMP file
3. Mat::adjustBrightness(delta) // Adjust brightness, delta ∈ [-255, 255]
4. Mat::adjustContrast(alpha)  // Adjust contrast, alpha ∈ [0.0, ∞)
5. Mat::toGrayscale()         // Convert image to grayscale
6. Mat::resize(newW, newH)    // Resize image to new dimensions
7. Mat::crop(x, y, w, h)      // Crop a region starting from (x, y) with width w and height h
8. Mat::flip()                // Flip image vertically or horizontally (user inputs H or V)
9. blend(Mat a, Mat b)       // Blend two images a and b
10. imshow(Mat)               // Display image Mat in a popup window (via temporary BMP)
11. size_of()                 // Get the size of Mat
```

图 2: HELP指令输出

过运行这个函数，开发者可以获得一些简单的库使用指导

## 2.3 图片读写

```
1     Mat imread(const std::string& filename);
2     bool imwrite(const std::string& filename, const Mat&
               mat);
```

这两个函数实现了本库中bmp图片的读写部分。imread可以将支持的图片都转化为一种统一的存储方式，然后让你进行处理。处理结束后，可以用imwrite将图片数据写入磁盘中。在读取8位的灰度图时，程序会询问你，要将灰度图保存为什么形式，可以存为灰度图或者真彩图，如果你输入了错误的内容，就会触发default自动存为真彩图。然后灰度图在输出时会随

之构造一个标准灰度表一起写入磁盘。我在实际的自我测试中，也测试了其他的BMP类型，但是由于他们并不很好在网上找到，我使用python生成了一些，他们丑陋且无聊，我将不会在此处进行额外的展示。

## 2.4 图片预览

参考opencv的设计，我增加了一个图片预览的功能。通过一个简单的弹窗，可以让你检查运行中的某个图像。方便程序开发者去找到一些细节的问题，也能够方便我们的代码演示。在展示出图片后，程序会暂停，等待你关闭页面，输入回车后再继续进行。

```
1         imshow(Mat a);
```



图 3: 弹窗展示

## 2.5 亮度调整与对比度调整

在我的代码中我同时实现了调整亮度和对比度的功能。亮度功能为输入一个整数然后调整亮度，对比度功能为输入浮点数然后调整。

```
1         Mat bright = img.adjustBrightness(100);
2         Mat up_con = img.adjustContrast(1.5f);
```



(a) 原图



(b) 亮度增加



(c) 对比度增强

我们可以明显的看出，增加亮度是针对全部图片的像素点的，让图片变得很亮。对比度也是针对全部点的，但是他会让像素间的差距更加明显，使人物更突出于背景存在

## 2.6 反转

```
1     Mat flipped = img.flip();
```

当程序运行到这里时，会输出指令等待输入

```
1      Flip image. Enter 'H' for horizontal or 'V' for
2          vertical flip: W
3      [ERROR] Invalid input. Please enter 'H(Horizontal)'
4          or 'V(Vertical)'.
5      Flip image. Enter 'H' for horizontal or 'V' for
6          vertical flip: T
7      [ERROR] Invalid input. Please enter 'H(Horizontal)'
8          or 'V(Vertical)'.
9      Flip image. Enter 'H' for horizontal or 'V' for
10         vertical flip: I
11     [ERROR] Invalid input. Please enter 'H(Horizontal)'
12         or 'V(Vertical)'.
13     Flip image. Enter 'H' for horizontal or 'V' for
14         vertical flip: P
15     [ERROR] Invalid input. Please enter 'H(Horizontal)'
16         or 'V(Vertical)'.
17     Flip image. Enter 'H' for horizontal or 'V' for
18         vertical flip: H
```

当你输入错误的内容时，程序会提示错误然后等待新输入，直到你输入了正确的可以运行的结果。



图 5: 反转图示

## 2.7 混合图像与重塑大小

这两个功能常常是一起使用的。我们从互联网上下载的图片经常不是同一大小的，所以我们要用resize调整一下。在这里我决定偷个懒，使用前后反转的图片进行混合作为示例。这样就生成了一个颇具足球小将风格的照片。



图 6: 混合图示

### 2.8 剪切

指定区域然后进行裁剪。一个运动员的健壮大腿就被从图片中分离下来了。

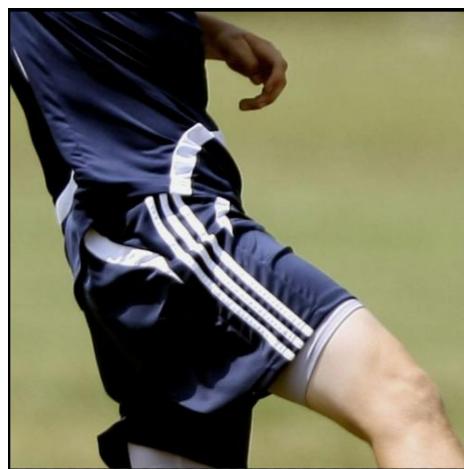


图 7: 混合图示

### 2.9 灰度化

通过人眼对色彩的敏感程度，我进行了灰度化的操作。利用加权平均数  $0.299R + 0.587G + 0.114B$  计算亮度（Lu minance Preserving，原理基于人眼对光的敏感度）。然后将这个结果统一赋值给BGR通道。



图 8: 混合图示

## 2.10 错误处理

我的库中包含大量的错误处理，包含输入输出图像的可能错误处理（完全借鉴自己的项目三），函数可能的输入错误处理，并且都在处理后给出了一定的错误提示。下面是部分错误处理的例子，这样的处理保证了程序拥有良好的鲁棒性。

```
1      if (a.width != b.width || a.height != b.height  
2          || a.channels != b.channels)  
3          throw std::runtime_error("blend: image  
4              dimensions must match");  
5  
6      while (true)  
7      {  
8          std::cout << "Flip image. Enter 'H' for  
9              horizontal or 'V' for vertical flip: ";  
10         std::cin >> mode;  
11         if (mode == 'H' || mode == 'h' || mode == 'V'  
12             || mode == 'v')  
13             break;  
14         std::cerr << "[ERROR] Invalid input. Please  
15             enter 'H(Horizontal)' or 'V(Vertical)'.\n  
16             ";
```

```
14         x + new_width > width || y + new_height > height
15     )
16     throw std::runtime_error("Invalid crop
17         parameters");
18
19     if (new_width <= 0 || new_height <= 0)
20     throw std::invalid_argument("Invalid dimensions
21         for resize.");
22
23     if (!file)
24     throw std::runtime_error("Cannot open BMP file:
25         " + filename);
26     if (fileHeader.bfType != 0x4D42)
27     throw std::runtime_error("Not a BMP file.");
28     if (fileHeader.biCompression != 0)
29     throw std::runtime_error("Compressed BMP not
30         supported.");
```

### 3 功能实现

#### 3.1 数据结构

在数据结构的方面，我阅读并参考了opencv的Mat类，设计了我自己的mat类，在mat中，我储存了长宽数据和通道数，然后使用了一个智能指针来管理线性的数据。除了图片的像素数据，其余的属性都没有几乎在外部更改（不设置set函数）。所有的构造函数相关部分我都使用了浅拷贝构造，用智能指针来防止内存泄漏。并且在构造的时候添加了检查，防止循环引用的情况出现。线性的数据结构也方便我在后续的操作中进行SIMD优化。

```

class Mat {
public:
    Mat(int width, int height, int channels);
    Mat(const Mat& other); // 浅拷贝构造
    Mat operator=(const Mat& other); // 浅拷贝赋值
    ~Mat() = default; // 智能指针自动释放

    Mat adjustBrightness(int value) const;
    friend Mat blend(const Mat& a, const Mat& b);
    Mat adjustContrast(float alpha) const;
    Mat resize(int new_width, int new_height) const;
    void size_of() const;
    Mat flip() const;
    Mat toGrayscale() const;
    Mat crop(int x, int y, int new_width, int new_height) const;
    Mat clone() const; // 深拷贝

    int getWidth() const;
    int getHeight() const;
    int getChannels() const;
    unsigned char* getWritableData(); // 可修改数据访问
    const unsigned char* getData() const; // 只读数据访问

private:
    int width, height, channels;
    std::shared_ptr<unsigned char[]> data;
};

//接口
Mat imread(const std::string& filename);
bool imwrite(const std::string& filename, const Mat& mat);
void imshow(const Mat& img);
void help();
#endif

```

图 9: Mat类实现

此外，我将混合函数设计为一个友元函数，这样操作起来更加符合常规的逻辑，去混合两个图像。为了防止成员函数对内部内容的错误修改，我多处使用了const，来防止对Mat对象值的改变。

关于文件头的内容，我在使用opencv后用Windows的工具去查看了输出图片的十六进制形式，我发现他处理过后的文件头很多地方都为0.也就是说记默认值，或者文件读取器会自动调整的值，所以我就没有将文件头设计在Mat里面，而是在imread和imwrite的地方使用结构体，短暂的处理了一下。这样就大大的节省了我的存储空间。当然了，就像项目三的内容一样，我使用了内存对齐去处理这部分文件头。

## 3.2 图像处理功能实现

### 3.2.1 图像读取与写入

读取文件头，判断是否是BMP文件以及能否被该程序处理。然后检查图片有没有调色板，如果有的话读取时会以调色板为准进行读取。最后，分辨清楚格式后，就开始读取了。读取时要去除padding位，这会影响很多功能的操作实现。在写入时，会补齐padding位。此外，我将所有的图像读取都在内存中设为BGR储存，这样无论是读取还是输出我都不需要进行顺

序的调换。height的正负也可能影响结果，我将height统一变为正数存储，如果是负数就从下向上读取。

### 3.2.2 亮度调整

我将输入的整数先判断符号然后取绝对值，然后截断为一个(0, 255)的数字。然后就开启了亮度的调整。超过255的部分都会被设为255，小于0的部分都会被设为0.所以亮度调整数如果很大图片就会变成白色的，很小就会变成黑色的。

### 3.2.3 混合

混合函数的实现也很简单，只需要加和取平均值就可以了。但是要注意输入的函数尺寸需要相同，如果不同则错误处理机制会抛出错误，无法混合。可以在输入之前用给定的size\_of函数看一看对应图像的尺寸。

### 3.2.4 图片预览

图片预览的处理用到了更加复杂的一些内容。

```
1      #ifdef _WIN32
2          #include <windows.h>
3          #define OS_WINDOWS
4      #elif defined(__APPLE__)
5          #include <unistd.h>
6          #define OS_MAC
7      #elif defined(__linux__)
8          #include <unistd.h>
9          #define OS_LINUX
10         #endif
```

我先检测了运行项目的系统，然后根据不同的系统（支持Windows, Linux, macOS）进行操作。不过需要提前安装好一些系统自带的图片读取器。我使用了Linux系统的eog。然后我先创建一个临时的图片，然后显示弹窗，在使用者关闭弹窗并且按下回车后，系统才会正常的往下运行。避免了多个弹窗同时不停弹出的情况。

### 3.2.5 比例重构

这部分我使用了双线性插值的办法，通过临近点来计算，实现了图像比例的放缩。

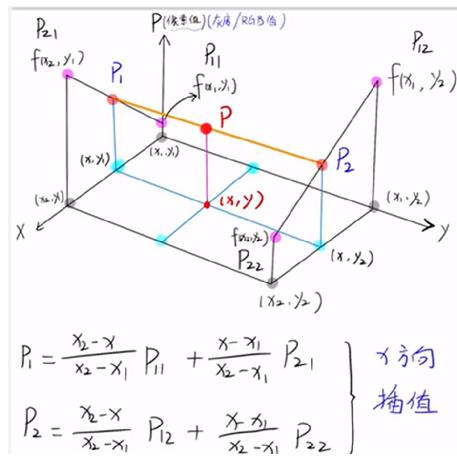


图 10: 双线性插值原理示意图

其余的部分都比较的简单，不做过多的赘述了。

## 4 科学优化

如何科学的优化是本次项目的一个小小的重点，下面我将展示一下我的优化过程。

### 4.1 并行优化

在考虑优化的时候，我本来想考虑试一试cuda，结果一番了解过后让我非常的失望。我没有一台拥有英伟达GPU的电脑，我的电脑上仅仅只有一个intel的集成显卡。所以我还是选择了openMP进行优化。我在很多地方都使用了并行优化的策略。在上次的项目中，我发现对I/O过程对性能的影响比较大，所以我在I/O部分做了进一步的处理，有时候任务规模比较小的时候，如果还是执意要并行处理，那么任务的分配和调度可能会让I/O的时间变得很长，所以我进行的优化，当输入规模不大时，我就不用并行处理。这样一来就可以让程序变得更加的迅速。为了防止程序变得

很冗余，我还将处理部分封装成了一个inline的函数，方便我在出现问题时即时高效的修改。至于封装成inline的原因，因为函数调用也是一比不小的开心，在并行处理时如果经常调用，可能会让程序的开销变大，所以我就使用了inline。（虽然编译器可能还是会在函数调用次数过多的时候自动将他设置为inline，但是那也只是可能）。此外，并行处理有时候可能也会导致一定的错误，比如同时使用file.read，等等，我在GPT的帮助下排除了这些问题。

## 4.2 SIMD

使用SIMD优化对于计算密集型的项目来说非常有效且常见。我使用的是AVX的指令进行优化。在使用时我发现，并不是所有的输入类型都能找到一个对应的指令进行处理，有时候可能需要我们提前进行类型的转换。比如在进行亮度调整时，我需要将一个32位的整数提前转成一个八位无符号的整数，来方便指令进行加速处理。

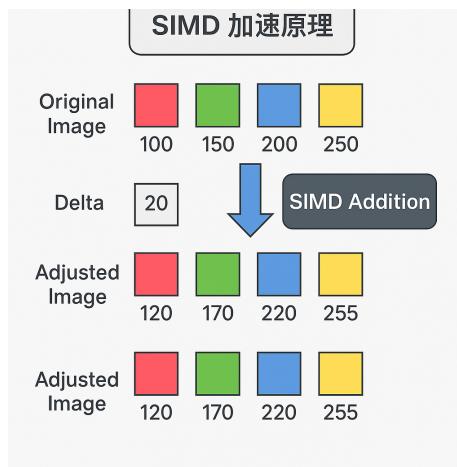


图 11: SIMD加速原理示意图

这个图示是由gpt生成的，展示了SIMD处理的原理，就是同时处理多个颜色点。

### 4.3 编译器的优化

在编译库的CmakeList文件中，我打开了-O3选项。经过我对GPT的拷问和直接查阅生成的汇编语言文件，我发现其实这种类矩阵运算，编译器会主动自动的变成SIMD的优化，这真的很好，所以我打开了-O3，希望他最大程度的帮我再优化优化。

### 4.4 避免索引的重复计算

我将所有的数组形式的内容都替换成裸指针的操作，这样避免了索引计算所花费的时间。更大程度的加速了程序的执行。但是实际上编译器也会帮我做，尤其是打开-O3选项的时候，不过我更信任我自己写的。

### 4.5 性能测试

在这次项目中，我不想测试关于图像处理功能的部分了。我想去试一试，动态库，静态库，还有不使用库对程序性能的影响。我将从内存问题出发。首先，使用动态库可以简化运行程序的内存占有情况，静态库时为90kb，无库时为89kb，动态库时为20kb，当然这是显而易见的因为没有包含动态库。动态库为63kb，这样来看即时合起来也是有微小的内存优化的。下面我们考虑性能情况，在分别测量十次运行时间，总的来看，运行时间并没有巨大的差别。取平均值分析后得出的结论是，动态库运行时间最大，而其余两者几乎相同。动态库要慢约9%。分析原因：首先，在启动程序时，要动态查找库，加载库。会稍慢一些，然后在运行时，函数调用可能通过间接跳转，加大了函数调用的开销。此外，还有一个比较重要的问题，就是运行程序无法通过调用函数的情况进行大规模优化，因为动态库的内部是不透明的。而另外两个的库都可被编译器阅读。这就是我分析出的可能的几个原因。不过库本身在编译时就有过编译器的优化了，这一点可能不是那么的明显。

## 5 总结

### 5.1 AI工具

本次项目我觉得AI工具真的非常好用。我发现有时候觉得ai不好用可能是你提问的话术太垃圾，如果你提问的方式足够细节，要求足够明确，

ai做出的东西在某些程度上要强于我。他的鼓励式提问令我很喜欢，每次向他提出新的要求，他都会说你的观察真的很细致！或者你的程序真的写的非常好了！（当然很大可能这是在骗我。）此外关于动态库的部分，其实我本来在这方面学习时没有很留心，我不是特别清楚里面的细节。我花了一下午时间拷打gpt让他给我讲清楚了一些。

## 5.2 总结的总结

这次项目我体会到了一种老板的快乐，我指挥AI帮我完成了许多的内容，而且还是免费的。尤其是我给他看了我的项目三内容，希望他学习学习我的处理风格。不过说到底我觉得，这次项目也让我感受到了编程的快乐。我添加了许多新的功能，然后他实现了之后出现了我想要的样子我真的挺开心的。还有一点就是，我在学习Java的时候很不认真，很多类的部分我都处于一种一知半解的状态，在这次项目之后，我学会了很多，包括我该如何构建我的代码结构，以及如何封装，而不是漏洞百出。