北京郵電大學

本 科 毕 业 设 计（论文）

题目：基于 SVD 算法的图像去块效应处理平台

姓　　名　熊国庆

学　　院　电子工程学院

专　　业　光信息科学与技术

班　　级　2011211208

学　　号　2011211004

班内序号　17

指导教师　李巍海

2015 年　5 月

基于 SVD 算法的图像去块效应处理平台

摘　要

图像处理是用计算机对图像进行分析，以达到所需结果的技术，又称影像处理。21 世纪是一个充满信息的时代，图像作为人类感知世界的视觉基础，是人类获取信息、表达信息和传递信息的重要手段。如何在已获得的有限资源的条件下将图像显示得更为清晰就成为了人们迫切的需求。而现在又是一个富媒体时代，信息都以图像等形式展现，如何快速地体现图像，将图像处理加速又是人们另一个关注的热点。

图像去块效应处理平台就是一个将图像处理而显示得更为清晰的系统，本文围绕“图像去块效应处理平台”进行软件的设计和开发,详细介绍了在 VS2012 的环境下开发本系统的过程。针对图像去块效应耗时长和复杂度高的问题，经过详细的调查研究，开发出了一套操作简单、功能实用的基于 SVD 算法的图像去块效应处理平台。其中将有块状效应的图像应用 SVD 算法进行图像去块效应处理，再使用 Low-Rank 等处理方法使得图像噪声下降，显示更为清晰。并对算法的运行时间进行控制，利用多线程和复写算法等方式完善了程序细节及时间复杂度。最后设计并完成图像的选择，导入，选择参数，图像压缩，去块处理，显示，导出以及平台相关的系统设置等功能。

从系统架构的角度讲，本系统基于 MFC 架构。使用 C++和 OPENCV 作为开发工具。系统工作在 windows 平台上。经过多种黑盒测试，该系统基本可以满足图像去块效应的科学和高效的要求。运行稳定，可以在实际应用中取得很好的效果。

关键词：图像去块效应 SVD 算法 OPENCV MFC

Image deblocking process platform based on SVD algorithm

ABSTRACT

Image processing, also known as media processing, uses computer to analyze images, in order to achieve the required results. The 21st century is an era full of information, the image is a foundation for human perception of world vision, is an important means of human to obtain information, express information and passing information. How to obtain the image display clearer under the condition of the limited resources has become a pressing needs of people. And now it is a rich media era, information is expressed in image form, such as how to show the image quickly and the image processing speed is another focus of people.

Image deblocking process platform is a system can do image processing and display pictures more clear. This paper focus on "Image deblocking process platform" in the software design and development. Under the environment of VS2012, the development process of this system was introduced in details. In view of the image to block effect time consuming and the problem of high complexity, more through detailed investigation and study, developed a simple operation, function and practical image processing platform to block effect based on SVD algorithm. Which will have a massive effect using SVD algorithm for image processing to block effect, using Low Rank processing methods such as image noise down, show more clearly. Based on the control to improve the program details of processing time and the time complexity, I used multithreading method and rewrote some algorithm. Finally the image selection, import, select parameters, image compression, image deblocking, display, export and platform-dependent system Settings, and other functions is designed and completed.

From the perspective of system architecture, this system is based on the MFC framework. Using C++ and OPENCV as a development tool. The system works on Windows platform. Through a variety of black box testing, the system can meet the requirements of scientific and efficient of the image deblocking process. The program is running stably, can achieve very good effect in practical application.

KEY WORDS: Image deblocking SVD OPENCV MFC

目 录

第一章 绪论 1
1.1 块效应产生的原因 1
1.2 图像去块效应发展现状 3
1.4 论文结构安排 5
1.5 本章小结 5
第二章 相关知识介绍 5
2.1 C++及 visual studio2012 介绍 5
2.1.1 C++介绍 5
2.1.2 Visual Studio 平台特点 6
2.2 OpenCV 介绍 7
2.3 MFC 框架 7
第三章 系统整体框架设计 8
3.1 窗口设计 8
3.2 动态创建 9
3.3 消息映射 9
3.4 消息传递 9
3.5 MFC 窗口类 CWnd 10
3.5.1 窗口创建函数 10
3.5.2 用于设定、获取、改变窗口属性函数 11
3.5.3 用于完成窗口动作函数 11
3.6 MFC 下创建一个窗口对象 11
3.7 MFC 窗口的使用 12
3.8 功能详情 14
3.8.1 开始页面 14
3.8.2 选择图像 14
3.8.3 选择量化表参数 15
3.8.4 图像压缩 15
3.8.5 去块效应 16
3.8.6 保存图像 16
第四章 算法设计 17
4.1 图像输入处理 17
4.2 图像去块效应算法处理 17
4.2.1 图像降维 17
4.2.2 图像块匹配 18
4.2.3 奇异值分解 18
4.3 算法结构 20
第六章 优化方法 21
6.1 行列赋值 21
6.2 SVD 分解 22
6.3 openmp 多线程并行操作 22
第七章 图像数据集测试 23
7.1 图像处理时间比较 23
7.2 图像去块效应平台测试 24
第八章 总结 26
参考文献 1
致 谢 1

第一章　绪论

以数字图像应用为主的多媒体技术是21世纪最具时代特征和最富有活力的研究与应用领域之一。一方面，人们对获取数字图像信息的执着、对多媒体通信的追求将会越来越强烈，这种追求不断推动着通信、计算机、信息处理等技术的发展；另一方面,这些技术的广泛应用进一步激起人们对数字图像处理、对数字图像服务质量更高的要求。

图像可以按其内容的运动状态分为静止图像和运动图像两类，其中按照电视技术术语，运动图像亦被称为视频[1]。目前，关于静止图像和视频图像压缩编码技术的研究已经日趋成熟，并且出现了一系列静止图像和视频图像压缩编码标准，实现了较高的压缩比。出于计算复杂度等方面的考虑，目前图像压缩编码标准大部分是基于图像分块的压缩编码，需要在低码率情况下、实现高压缩比，编码算法需要丢弃大量被认为视觉上较为次要的信息，导致解码后的重构图像中图像块边缘处的亮度值发生跳变，即块效应。块效应的存在，严重地影响图像的主观视觉质量。因此，如何去除块效应、提高图像的主观视觉质量，成为了图像处理、图像压缩领域中另一个新的研究课题。

1.1 块效应产生的原因

变换编码是图像压缩编码中最常使用的方法之一，该编码技术是将一组像素值经过某种形式的正交变换转换为一组变换系数，图像经过正交变换之后，变换系数之间的相关性变的非常小，而且大部分能量集中在少数的变换系数上，然后根据图像能量的分布以及人眼视觉系统对不同图像能量的敏感程度来对各变换系数进行不同精度的量化，然后对量化系数进行编码，以此来消除像素间的空间冗余。目前常见的正交变换有离散傅立叶变换、DCT变换、K．L变换等，其中K．L变换是最小均方差意义下最佳正交变换，但是K．L变换没有快速算法，与K-L变换性能最接近的是DCT变换，由于该变换存在快速算法，利于硬件实现，因此目前绝大部分国际静态图像编码标准或视频图像编码标准中都采用了DCT变换。但是在基于DCT的压缩编码系统中，如果对整幅图像做DCT变换，由于变换后的DCT系数和图像中的每个像素都相关，会使得运算量相当大，因此，目前多数图像编码都是采用基于图像分块的DCT变换(BDCT)编码，即将图像分割成互不重叠的小方块，然后对这些小方块进行DCT变换，将其从空域转换到频域，然后对其DCT系数进行量化、熵编码，以此来去除图像的空域冗余[2]。在解码端再对编码的系数进行熵解码，再对每个图像块进行DCT反变换，最后将操作完成后的图像块组成一幅完整的图像， JPEG以及H．26x系列与MPEGⅨ系列这些国际编码标准都是采用这种图像基本编码框架。从理论上讲，DCT变换是无损的，但是由于对DCT系数进行量化的时候，是将DCT系数除以量化步长QP后取整，因此，该运算是有损的.

其中，量化矩阵的选择是由输入图像及图像显示设备的特性来决定的，JPEG中根据人眼视觉系统难以感觉到高频信息失真的特点，推荐了一个量化矩阵，如表1．2所示，为了控制图像的质量及压缩比，可以通过将量化矩阵乘以一定的量化因子来实现。以本文的图像去块效应平台程序的一级量化参数为例：

```
int a[8][8]={{50,60,70,70,90,120,255,255},
             {60,60,70,96,130,255,255,255},
             {70,70,80,120,200,255,255,255},
             {70,96,120,145,255,255,255,255},
             {90,130,200,255,255,255,255,255},
```

{120,255,255,255,255,255,255,255},
{255,255,255,255,255,255,255,255},
{255,255,255,255,255,255,255,255}};

式(1-1)

量化步长随着频率的升高而变大，对图像的高频分量量化较为粗糙，直接导致重构图像中一些细节信息的损失，造成图像失真；再加上BDCT变换都是将每个图像块作为单个整体分别处理，没有考虑相邻图像块之间的相关性，造成图像块非均匀失真，这样就导致重构图像块之间像素值出现不连续的现象，形成明显的块边界，看起来就像有很多“马赛克”一样，这就是块效应；当整体量化步长较低时，对DCT系数造的失真还比较轻微，从而块效应也不会很明显。但在低码率情况下，要将表1．2所示的量化矩阵乘以一个较大的量化因子，使得整体量化步长增大，对DCT系数造成较大的失真，很多代表图像高频信息的DCT系数直接被量化取整为零，导致图像块中更多高频信息的丢失，将原来相邻像素之间灰度的连续变化演变为“台阶”变化，在图像块边界处造成“虚假边缘”，形成明显的块效应，极大地损害图像的主观质量，如图1-1所示。总之，压缩比越大，形成的块效应越明显。

随着图像内容的不同，块效应也有不同的表现形式，主要有以下两种：梯形噪声和格形噪声。梯形噪声在图像的纹理区域出现，在低码率下，DCT的很多高频系数被量化为零，结果与边缘纹理信息有关的高频分量在变换域内不能完全体现，加上图像块的单独处理，不能保证穿过块边界的边缘纹理的连续性，导致在图像的边缘纹理处出现锯齿状噪声，如图1-1中人物的帽沿和肩膀边缘处，这种噪声称之为“梯形噪声”；格形噪声出现在图像的平坦区域，在变换域内，直流分量(DC)系数体现了图像块的平均亮度，所以这个系数包含了图像块的大部分能量，平坦区域中亮度的变化很小，但是如果有亮度是递增或递减，在量化取整时进行了四舍五入，可能导致DC系数越过相邻量化级的判决门限，造成在重建图像块边界处出现亮度突变，在视觉效果上表现为在平坦区域内出现片状轮廓，如图1-1 右图人物背后的背景部分，这种噪声称之为“格形噪声”。

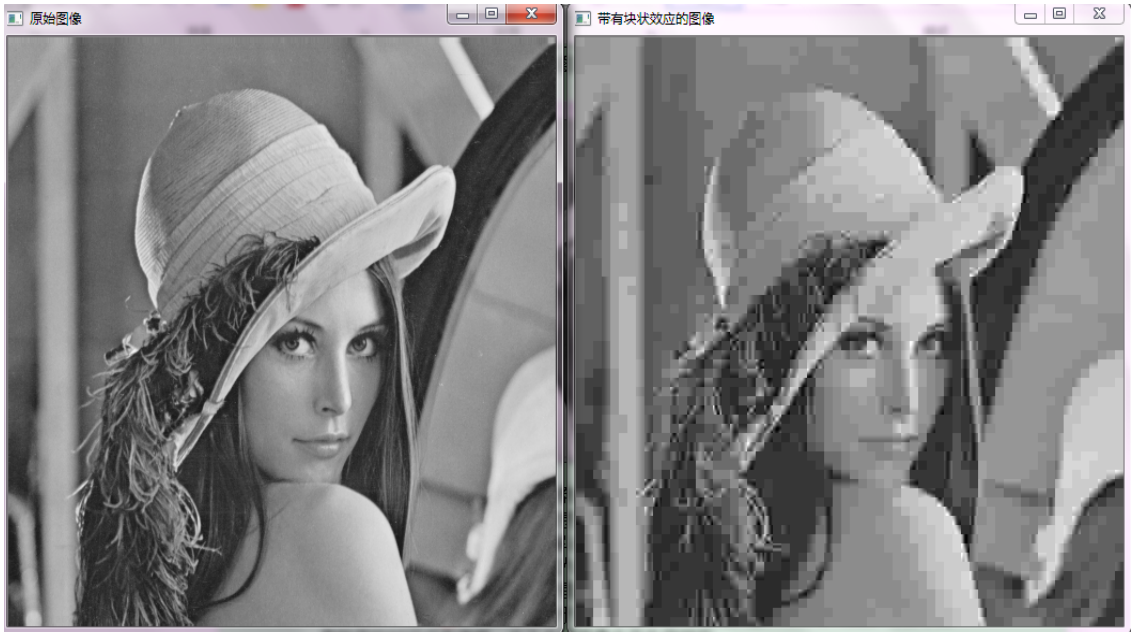

图 1-1 原始图像（左） 带有块状效应的图像（右）

根据人眼的视觉特性，对图像纹理区域中的梯形噪声相对来说不易察觉，但是对于平坦区域中的格形噪声却非常敏感，哪怕足很轻微的格形噪声都会大大影响人们对视频

质量的感觉。如何有效的解决这一问题，尽可能的降低或去除块效应成为刻不容缓的任务，因此，解决块效应成为图像处理中一个研究热点之一。尤其在低码率的情况下，低码率和块效应之间的矛盾比较突出，若要提高压缩比，图像中就必定会出现块效应。一般说柬，人们更倾向于获得较高的主观视觉质量，即，宁可更多地容忍图像稍微的时延或者停顿，也不希望图像中产生块效应。

1.2 图像去块效应发展现状

图像信息中包含巨大的数据量，以一幅中等分辨率的真彩色图像(分辨率为640×480，24比特／像素)为例，其比特数为640×480×24=7．37Mb(兆比特)，需占约O．9MB(兆字节)的存储空间。一1GB(1000MB)容量的硬盘只能存储1000多幅这样的图像，如果以每秒25帧的速度显示，其传输速率要求达到184Mb，s(兆比特／秒)[3]。如此庞大的数据量，给图像信息的传输、存储、处理等都带来巨大压力，因此对图像信息进行压缩编码势在必行。香农信息论[4]奠定了图像压缩编码的理论基础。根据香农信息论的观点，数据的信息应由其信息熵来表征、而多余信息均为冗余，图像数据中的冗余包括时域余(视频图像之中)、空域冗余、视觉冗余等。图像压缩编码技术正是将这些冗余去除，以达到压缩数据量的目的。

针对块效应问题，出现了许多不同的方法来解决这一问题。其中，主要的方法有两类：一是彻底改变原有编码模式，不再采用分块的处理方法，例如基于纹理的编码、小波变换编码等；另一类是在现有分块解决方案的基础上采用一些新的处理方法，例如进行低通滤波等。考虑到基于块编码方法的性能、复杂度、兼容性和市场需求等，以及目前国际标准中采用的主要还是基于块的编码方式。所以，对基于块效应的消除方法也得到了广泛的研究。对块效应的消除大都采用图像后处理技术，好处是不改变编码过程，并保持压缩编码率[5]。

去块效应后处理技术可以分为两种：基于图像恢复的方法和基于图像增强的方法。前者是将块效应的去除看作是对劣化图像的恢复问题，有很多经典的图像恢复算法可利用。基于图像恢复的算法考虑到很多先验知识，包括凸集投影法(POCS)、最大后验概率(MAP)等经典的图像恢复算法。基于图像增强的方法着重于对图像块效应的平滑，而不是将每一像素恢复到其原始值，主要有空域滤波和变换域滤波方法。

根据块效应产生的原因，一种最基本的想法就是改变图像的编码方法，例如在非图像块的基础上进行编码，如Malvar等人所提出的采用基于重叠块的编码方法；或采用其他的变换编码方法来代替DCT变换，如Xiong等人提出采用小波变换来代替DCT变换，但是综合性能、复杂度、兼容性和市场需求等多方面因素，涉及图像编码的国际标准，仍然倾向于采用咀基于块的DCT变换编码算法为基础的方法，因此，那些以通过改变图像的编码方式束击除块效应的算法渐渐演出了人们的视野，目前研究的重点是那些与国际图像编码标准相兼容的去块效应算法。

根据滤波器所处的位置可以分为编码环路中的环路去块滤波和解码端的后处理去块效应滤波两类。环路去块滤波是将滤波器置于编解码环路之内，主要用于对视频图像的去块效应处理：利用编解码过程中获得的信息，在编码端和解码端对每一帧重构图像进行去块滤波，将处理后的图像作为随后帧的参考帧，以此来去除视频图像中的块效应；为了使编解码器在处理数据时相匹配，要求编码器和解码器使用相同的滤波器，并且处于相对应的位置。环路滤波器由于使用了视频图像编码时相应的编码信息，使块效应分类判别更加准确，从而获得了较好的去块效果，并且由于使用滤波后的图像帧作为后继帧的参考帧，使得运动预测更加准确、预测残差更小且不会被向后传递，提高了压缩性

能。但是环路滤波器忽略了人眼视觉系统(HvS)对图像平坦区域与细节丰富区域的块效应敏感程度的差异，容易造成块边界高频信息损失，平坦区域的块内噪声也得不到有效处理。后处理去块效应滤波技术主要用来去除图像中的块效应，是在解码端对解码后的图像进行去块效应处理。当然，如果对计算量没有要求，也可以在视频序列经过环路滤波之后，再在解码端进行后处理去块滤波，以此更好地提高图像的主观视觉质量。由于后处理去块效应滤波只在解码器端提高重构图像的质量，不需要修改编解码的过程，因此后处理方法与现有的国际压缩编码标准完全兼容，也正是因为这个优势，使得后处理去块效应算法成为去块效应研究领域中最热门的研究方向。

根据图像处理目的，后处理去块滤波又可以分以下为两类：基于图像恢复的去块滤波和与基于图像增强的去块滤波。

基于图像恢复的后处理去块滤波主要是在进行去块滤波时，尽量将解码图像恢复至原始图像。在滤波的过程中需要用到视频图像解码时的信息，以有效的判断块效应出现的情况和块效应的轻重程度，根据这些信息有效地通过去块滤波去除块效应。这种去块效应滤波通常会有比较好的效果，由于需要将后处理去块效应滤波器和解码器连接到一起使用，以利用编解码系统中的先验信息，所以，不适合对于已解码视频图像进行处理的情况。

基于图像增强的后处理去块滤波完全和解码器独立，可以在不知道解码信息的情况下对解码后图像进行去块滤波。该方法着重于对图像块效应的平滑，主要是根据人眼视觉效果有选择性的对图像的不同部分进行平滑处理，比如：在图像的平坦区域，人眼对块效应会更敏感，而在边缘区域由于人眼的掩蔽效应，人眼的敏感度降低；因此将图像分为不同的区域，然后根据图像的像素信息判断块效应的强度来进行去块处理。该方法可以较好的去除块效应，但是会造成图像的模糊。由于在实际情况中，一般无法获取视频图像的编解码信息，因此基于图像增强的后处理去块滤波成为了研究的热点。在本文中，我将主要以SVD分解算法作为图像增强的后处理方法，作为本程序的核心方法。

1.3 去块效应的评价标准

对去块效应算法的评价，主要通过比较原始图像、具有块效应的图像以及经过去块效应处理的图像来得出去块效应的效果。具体的评价标准分为主观质量评价标准和客观质量评价标准。

主观质量评价是从人眼视觉角度对原始图像、具有块效应的图像以及经过去块效应处理的图像进行观察比较，来判断块效应的去除效果。这种评价标准与观察者的特性及观察条件等因素有关，为保证主观评价在统计上有意义，选择观察者时既要考虑有未受过训练的“外行”观察者，又要考虑对图像技术有一定经验的“内行”观察者。因此，可以挑选一批人根据一定的标准来给图像打分，根据打分的
高低来评价块效应去除效果的好坏。客观质量评价是利用数学模型测量图像质量，与主观质量评价相比，具有速度快、费用低和可嵌入性等优点，比较实用。最经典的评价方法就是峰值信噪比(PSNR)方法，PSNR的计算简单快速，成为图像客观质量评价中较通用的方法[6]。PSNR是利用重构图像偏移原始图像的误差来衡量重构图像的质量，定义如下：给定一幅大小为M×N的原始数字图像f(x，y)和重构图像$f_0(x，y)$，则PSNR为：

$$\mathrm{PSNR}=10\lg\frac{f_{max}^2}{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[f(x，y)-f_0(x，y)]^2}$$

式(1-2)

PSNR可以有效地反映出重构图像与原始图像的像素值差别，是定量评价图像质量的主要指标，所以本文中将采用PSNR作为图像去块效应质量指标。

1.4 论文结构安排

文章一共分为八章，各章介绍如下：

第一章绪论：主要介绍图像去块效应处理的背景及发展趋势，国内外发展趋势，阐述本文的研究意义。

第二章相关知识介绍：介绍所用到的技术，包括：C++、OPENCV、MFC 框架。

第三章对系统整体进行了设计：对系统整体框架设计，窗口模块设计等等进行了阐述。

第四章算法设计：详细介绍了图像去块效应设计思路及具体的图像的 SVD 计算设计，算法流程的设计。

第五章详细设计：详细的对系统的每一个模块进行了介绍展示，并对主要的方法进行了简短的介绍。

第六章算法优化：针对图像处理算法的耗时长的问题及用户对于图像显示时间的需求优化程序运行时间。

第七章系统测试：一个系统要正式投入使用前必须经过认真的测试。本章从图像输入，量化参数设置等方面进行了测试。

第八章总结全文，并提出不足之处，并对将来进行展望。

1.5 本章小结

本章首先介绍了图像去块效应的发展及背景，然后详细介绍了国内外发展趋势；阐述了图像去块效应在人们生活中的重要性和必要性。最后对论文的安排进行了简单的介绍。

第二章 相关知识介绍

2.1 C++及 visual studio2012 介绍

2.1.1 C++介绍

C++是在 C 语言的基础上开发的一种通用编程语言，应用广泛。C++支持多种编程范式，面向对象编程、泛型编程和过程化编程。其编程领域众广，常用于系统开发，引擎开发等应用领域，是至今为止最受广大受用的最强大编程语言之一,支持类:类、封装、

重载等

C++设计成静态类型、和 C 同样高效且可移植的多用途程序设计语言。

C++设计直接的和广泛的支持多种程序设计风格（程序化程序设计、资料抽象化、面向对象程序设计、泛型程序设计）。

C++设计无需复杂的程序设计环境。

C++语言灵活，运算符的数据结构丰富、具有结构化控制语句、程序执行效率高，而且同时具有高级语言与汇编语言的优点，与其它语言相比 ，可以直接访问物理地址，与汇编语言相比又具有良好的可读性和可移植性。

总得来说，C++语言的主要特点表现在两个方面，一是尽量兼容 C,二是支持面向对象的方法。它操持了 C 的简洁、高效的接近汇编语言等特点，对 C 的类型系统进行了改革的扩充，因此 C++比 C 更安全，C++的编译系统能检查出更多的类型错误。另外，由于 C 语言的广泛使用，因而极大的促进了 C++的普及和推广。

C++语言最有意义的方面是支持面向对象的特征。虽然与 C 的兼容使得 C++具有双重特点，但他在概念上完全与 C 不同，更具面向对象的特征。

2.1.2 Visual Studio 平台特点

Microsoft Visual Studio（简称 VS）[7]是美国微软公司的开发工具包系列产品。VS 是一个基本完整的开发工具集，它包括了整个软件生命周期中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(IDE)等等。所写的目标代码适用于微软支持的所有平台，包括 Microsoft Windows、Windows Mobile、Windows CE、.NET Framework、.NET Compact Framework 和 Microsoft Silverlight 及 Windows Phone。

Visual Studio 是目前最流行的 Windows 平台应用程序的集成开发环境。

Visual Studio 2012 在三个方面为开发人员提供了关键改进：

快速的应用程序开发

高效的团队协作

突破性的用户体验

Visual Studio 2012 提供了高级开发工具、调试功能、数据库功能和创新功能，帮助在各种平台上快速创建当前最先进的应用程序。

Visual Studio 2012 包括各种增强功能，例如可视化设计器（使用 .NET Framework 3.5 加速开发）、对 Web 开发工具的大量改进，以及能够加速开发和处理所有类型数据的语言增强功能。Visual Studio 2012 为开发人员提供了所有相关的工具和框架支持，帮助创建引人注目的、令人印象深刻并支持 AJAX 的 Web 应用程序。

开发人员能够利用这些丰富的客户端和服务器端框架轻松构建以客户为中心的 Web 应用程序，这些应用程序可以集成任何后端数据提供程序、在任何当前浏览器内运行并完全访问 ASP NET 应用程序服务和 Microsoft 平台。

Visual Studio 2012 还使开发人员能够从同一开发环境内创建面向多个 .NET

Framework 版本的应用程序。开发人员能够构建面向 .NET Framework 2.0、3.0 或 3.5 的应用程序，意味他们可以在同一环境中支持各种各样的项目。

2.2 OpenCV 介绍

OpenCV[8]的全称是：Open Source Computer Vision Library。OpenCV 是一个基于（开源）发行的跨平台计算机视觉库，可以运行在 Linux、Windows 和 Mac OS 操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了 Python、Ruby、MATLAB 等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

OpenCV 于 1999 年由 Intel 建立，如今由 Willow Garage 提供支持。OpenCV 是一个基于 （开源）发行的跨平台计算机视觉库，可以运行在 Linux、Windows 和 Mac OS 操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了 Python、Ruby、MATLAB 等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

OpenCV 拥有包括 500 多个 C 函数的跨平台的中、高层 API。它不依赖于其它的外部库——尽管也可以使用某些外部库。

OpenCV 为 Intel® Integrated Performance Primitives (IPP) 提供了透明接口。 这意味着如果有为特定处理器优化的 IPP 库， OpenCV 将在运行时自动加载这些库。

OpenCV 致力于真实世界的实时应用，通过优化的 C 代码的编写对其执行速度带来了可观的提升，并且可以通过购买 Intel 的 IPP 高性能多媒体函数库(Integrated Performance Primitives)得到更快的处理速度。

2.3 MFC 框架

MFC(MicrosoftFoundationClasses)[9]是微软基础类库的简称，是微软公司实现的一个C++类库，主要封装了大部分的 windows API 函数，VC++是微软公司开发的 C/C++的集成开发环境，所谓集成开发环境，就是说利用它你可以编辑，编译，调试，而不是使用多种工具轮换操作，灵活性较大。以 C++类的形式封装了 Windows API，并且包含一个应用程序框架，以减少应用程序开发人员的工作量。其中包含的类包含大量 Windows 句柄封装类和很多 Windows 的内建控件和组件的封装类。

MFC 中的各种类结合起来构成了一个应用程序框架，它的目的就是让程序员在此基础上来建立 Windows 下的应用程序，这是一种相对 SDK 来说更为简单的方法。

因为总体上，MFC 框架定义了应用程序的轮廓，并提供了用户接口的标准实现方法，程序员所要做的就是通过预定义的接口把具体应用程序特有的东西填入这个轮廓。

Microsoft Visual studio 提供了相应的工具来完成这个工作：AppWizard 可以用来生成初步的框架文件（代码和资源等）；资源编辑器用于帮助直观地设计用户接口；ClassWizard 用来协助添加代码到框架文件；最后，编译，则通过类库实现了应用程序特定的逻辑。

第三章　系统整体框架设计

3.1　窗口设计

一个应用程序在创建某个类型的窗口前，必须首先注册该“窗口类”(Windows Class)。注意，这里不是 C++类的类。Register Window 把窗口过程、窗口类型以及其他类型信息和要登记的窗口类关联起来。

“窗口类”是 Windows 系统的数据结构，可以把它理解为 Windows 系统的类型定义，而 Windows 窗口则是相应“窗口类”的实例。Windows 使用一个结构来描述“窗口类”，其定义如下：

```
typedef struct _WNDCLASSEX {
UINT cbSize; //该结构的字节数
UINT style; //窗口类的风格
WNDPROC lpfnWndProc; //窗口过程
int cbClsExtra;
int cbWndExtra;
HANDLE hInstance; //该窗口类的窗口过程所属的应用实例
HICON hIcon; //该窗口类所用的像标
HCURSOR hCursor; //该窗口类所用的光标
HBRUSH hbrBackground; //该窗口类所用的背景刷
LPCTSTR lpszMenuName; //该窗口类所用的菜单资源
LPCTSTR lpszClassName; //该窗口类的名称
HICON hIconSm; //该窗口类所用的小像标
} WNDCLASSEX;
```

式(3-1)

从“窗口类”的定义可以看出，它包含了一个窗口的重要信息，如窗口风格、窗口过程、显示和绘制窗口所需要的信息，等等。关于窗口过程，将在后面消息映射等有关章节作详细论述。

Windows 系统在初始化时，会注册(Register)一些全局的“窗口类”，例如通用控制窗口类。应用程序在创建自己的窗口时，首先必须注册自己的窗口类。

AfxRegisterClass 函数是 MFC 全局函数。AfxRegisterClass 的函数原型：

BOOL AFXAPI AfxRegisterClass(WNDCLASS *lpWndClass);

参数 lpWndClass 是指向 WNDCLASS 结构的指针，表示一个“窗口类”。

首先，AfxRegisterClass 检查希望注册的“窗口类”是否已经注册，如果是则表示已注册，返回 TRUE，否则，继续处理。

接着，调用::RegisterClass(lpWndClass)注册窗口类；

然后，如果当前模块是 DLL 模块，则把注册“窗口类”的名字加入到模块状态的域

m_szUnregisterList 中。该域是一个固定长度的缓冲区，依次存放模块注册的“窗口类”的名字（每个名字是以“\n\0”结尾的字符串）。之所以这样做，是为了 DLL 退出时能自动取消(Unregister)它注册的窗口类。

最后，返回 TRUE 表示成功注册。

3.2 动态创建

动态创建就是运行时创建指定类的对象，在 MFC 中大量使用。如框架窗口对象、视对象，还有文档对象都需要由文档模板类对象来动态的创建。我觉得这是每个 MFC 的学习者很希望理解的问题。

初次接触 MFC 的时候，很容易有这样的迷惘。MFC 的几大类不用我们设计也就罢了，但最疑惑的是不用我们实例化对象。本来最直观的理解就是，我们需要框架的时候，亲手写上 CFrameWnd myFrame；需要视图的时候，亲自打上 CView myView;……但 MFC 不给我们这个机会，致使我们错觉窗口没有实例化就弹出来了！就象画了张电视机的电路图就可以看电视一样令人难以置信。其实不然，写 MFC 程序的时候，我们几乎要对每个大类进行派生改写。换句话说，MFC 并不知道我们打算怎样去改写这些类，当然也不打算全部为我们“静态”创建这些类了。即使静态了创建这些类也没有用，因为我们从来也不会直接利用这些类的实例干什么事情。我们只知道，想做什么事情就往各大类里塞，不管什么变量、方法照塞，塞完之后，我们似乎并未实例化对象，程序就可以运行.

3.3 消息映射

消息映射与命令传递体现了 MFC 与 SDK 的不同。在 SDK 编程中，没有消息映射的概念，它有明确的回调函数中，通过一个 switch 语句去判断收到了何种消息，然后对这个消息进行处理。所以，在 SDK 编程中，会发送消息和在回调函数中处理消息就差不多可以写 SDK 程序了。

在 MFC 中，看上去发送消息和处理消息比 SDK 更简单、直接，但可惜不直观。举个简单的例子，如果我们想自定义一个消息，SDK 是非常简单直观的，用一条语句：SendMessage(hwnd,message/*一个大于或等于 WM_USER 的数字*/,wparam,lparam)，之后就可以在回调函数中处理了。但 MFC 就不同了，因为你通常不直接去改写窗口的回调函数，所以只能亦步亦趋对照原来的 MFC 代码，把消息放到恰当的地方。这确实是一样很痛苦的劳动。

3.4 消息传递

有了消息映射表之后，我们得讨论到问题的关键，那就是消息发生以后，其对应的响应函数如何被调用。大家知道，所有的 MFC 窗口，都有一个同样的窗口过程——

AfxWndProc(…)。在这里顺便要提一下的是，看过 MFC 源代码的朋友都知道，从 AfxWndProc 函数进去，会遇到一大堆曲折与迷团，因为对于这个庞大的消息映射机制，MFC 要做的事情很多，如优化消息，增强兼容性等，这一大量的工作，有些甚至用汇编语言来完成，对此，我们很难深究它。所以我们要省略大量代码，理性地分析它。

3.5 MFC 窗口类 CWnd

在 Windows 系统里，一个窗口的属性分两个地方存放：一部分放在“窗口类”里头，如上所述的在注册窗口时指定；另一部分放在 Windows Object 本身，如：窗口的尺寸，窗口的位置（X，Y 轴），窗口的 Z 轴顺序，窗口的状态（ACTIVE，MINIMIZED，MAXMIZED，RESTORED…），和其他窗口的关系（父窗口，子窗口…），窗口是否可以接收键盘或鼠标消息，等等。

为了表达所有这些窗口的共性，MFC 设计了一个窗口基类 CWnd。有一点非常重要，那就是 CWnd 提供了一个标准而通用的 MFC 窗口过程，MFC 下所有的窗口都使用这个窗口过程。

CWnd 提供了一系列成员函数，或者是对 Win32 相关函数的封装，或者是 CWnd 新设计的一些函数。这些函数大致如下。

3.5.1 窗口创建函数

主要是函数 Create 和 CreateEx。它们封装了 Win32 窗口创建函数::CreateWindowEx。Create 的原型如下：

```
BOOL CWnd::Create(LPCTSTR lpszClassName,
LPCTSTR lpszWindowName, DWORD dwStyle,
const RECT& rect,
CWnd* pParentWnd, UINT nID,
CCreateContext* pContext)
```

式(3-2)

Create 是一个虚拟函数，用来创建子窗口（不能创建桌面窗口和 POP UP 窗口）。CWnd 的基类可以覆盖该函数，例如边框窗口类等覆盖了该函数以实现边框窗口的创建，视类则使用它来创建视窗口。

Create 调用了成员函数 CreateEx。CWnd::CreateEx 的原型如下：

```
BOOL CWnd::CreateEx(DWORD dwExStyle, LPCTSTR lpszClassName,
LPCTSTR lpszWindowName, DWORD dwStyle,
int x, int y, int nWidth, int nHeight,
HWND hWndParent, HMENU nIDorHMenu, LPVOID lpParam)
```

式(3-3)

CreateEx 有 11 个参数，它将调用::CreateWindowEx 完成窗口的创建，这 11 个参数

对应地传递给::CreateWindowEx。参数指定了窗口扩展风格、“窗口类”、窗口名、窗口大小和位置、父窗口句柄、窗口菜单和窗口创建参数。

窗口创建时发送 WM_CREATE 消息，消息参数 lParam 指向一个 CreateStruct 结构的变量，该结构有 11 个域，Windows 使用和 CreateEx 参数一样的内容填充该变量。

3.5.2 用于设定、获取、改变窗口属性函数

窗口属性函数

SetWindowText(CString tiltle)	设置窗口标题
GetWindowText()	得到窗口标题
SetIcon(HICON hIcon, BOOL bBigIcon)	设置窗口像标
GetIcon(BOOL bBigIcon)	得到窗口像标
GetDlgItem(int nID)	得到窗口类指定 ID 的控制子窗口
GetDC()	得到窗口的设备上下文
SetMenu(CMenu *pMenu)	设置窗口菜单
GetMenu()	得到窗口菜单

表 3-1

3.5.3 用于完成窗口动作函数

用于更新窗口，滚动窗口，等等。一部分成员函数设计成或可重载(Overloaded）函数，或虚拟(Overridden)函数，或 MFC 消息处理函数。

有关消息发送的函数：

SendMessage(UINT message,WPARAM wParam = 0, LPARAM lParam = 0);

给窗口发送发送消息，立即调用方式

PostMessage((UINT message,WPARAM wParam = 0, LPARAM lParam = 0);

给窗口发送消息，放进消息队列

有关改变窗口状态的函数

MoveWindow(LPCRECT lpRect, BOOL bRepaint = TRUE);

移动窗口到指定位置

ShowWindow(BOOL)；显示窗口，使之可见或不可见

式(3-4)

3.6 MFC 下创建一个窗口对象

MFC 下创建一个窗口对象分两步，首先创建 MFC 窗口对象，然后创建对应的 Windows 窗口。在内存使用上，MFC 窗口对象可以在栈或者堆(使用 new 创建)中创建。具体表述如下：

创建 MFC 窗口对象。通过定义一个 CWnd 或其派生类的实例变量或者动态创建一个 MFC 窗口的实例，前者在栈空间创建一个 MFC 窗口对象，后者在堆空间创建一个

MFC 窗口对象。

调用相应的窗口创建函数，创建 Windows 窗口对象。

有 CMainFrame（派生于 CMDIFrame(SDI)或者 CMDIFrameWnd(MDI)）类。它有两个成员变量定义如下：

CToolBar m_wndToolBar;

CStatusBar m_wndStatusBar;

当创建 CMainFrame 类对象时，上面两个 MFC Object 也被构造。

CMainFrame 还有一个成员函数

OnCreate（LPCREATESTRUCT lpCreateStruct)，

它的实现包含如下一段代码，调用 CToolBar 和 CStatusBar 的成员函数 Create 来创建上述两个 MFC 对象对应的工具栏 HWND 窗口和状态栏 HWND 窗口。

在本项目中创建方法如下：

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWndEx::OnCreate(lpCreateStruct) == -1)
        return -1;
    BOOL bNameValid;
        OnApplicationLook(theApp.m_nAppLook);
    if (!m_wndMenuBar.Create(this))
    {
        TRACE0("未能创建菜单栏\n");
        return -1;          // 未能创建
    }
```

式(3-5)

在 Windows 窗口的创建过程中，将发送一些消息，如：

在创建了窗口的非客户区(Nonclient area)之后，发送消息 WM_NCCREATE；

在创建了窗口的客户区（client area）之后，发送消息 WM_CREATE；

窗口的窗口过程在窗口显示之前收到这两个消息。

3.7 MFC 窗口的使用

MFC 提供了大量的窗口类，其功能和用途各异。在本次项目中直接使用 MFC 提供的窗口类或者先从 MFC 窗口类派生一个新的 C++类然后使用它，这些在通常情况下都不需要程序员提供窗口注册的代码。是否需要派生新的 C++类，视 MFC 已有的窗口类是否能满足使用要求而定。派生的 C++类继承了基类的特性并改变或扩展了它的功能，例如增加或者改变对消息、事件的特殊处理等。

在本次图像处理平台项目中主要使用或继承以下一些 MFC 窗口类

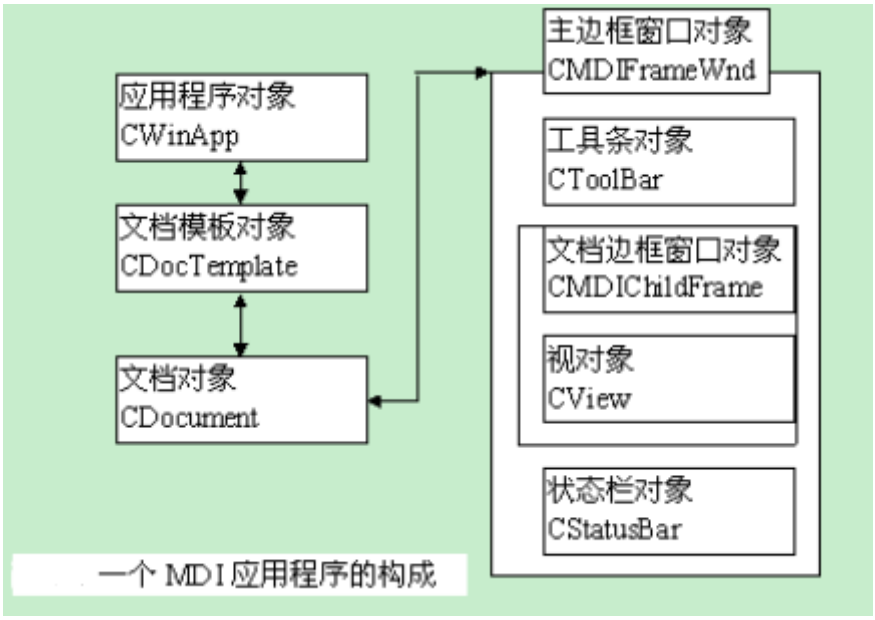

图 3-1 MFC 窗口类

视图 CView 和 CView 派生的有特殊功能的视图如：列表 CListView，编辑 CEditView，树形列表 CTreeView，支持 RTF 的 CRichEditView，基于对话框的视 CFormView 等等。

通常，都要从这些类派生应用程序的框架窗口和视窗口或者对话框。

其他各类控制窗口，如列表框 CList，编辑框 CEdit，组合框 CComboBox，按钮 Cbutton 等。在项目中直接使用了这些类。

CDocument 文档，负责内存数据与磁盘的交互。最重要的是 OnOpenDocument(读入)，OnSaveDocument（写盘），Serialize（序列化读写）。

CView 视图，负责内存数据与用户的交互。包括数据的显示、用户操作的响应（如菜单的选取、鼠标的响应等等）。最重要的是 OnDraw(重画窗口)，通常用 CWnd::Invalidate()来启动它。另外，它通过消息映射表处理菜单、工具条、快捷键和其他用户消息。你自己的许多功能都要加在里面，你打交道最多的就是它。

CDC 设备文本。无论是显示器还是打印机，都是画图给用户看。这图就抽象为 CDC。CDC 与其他 GDI（图形设备接口）一起，完成文字和图形、图像的显示工作。把 CDC 想象成一张纸，每个窗口都有一个 CDC 相联系，负责画窗口。CDC 有个常用子类 CClientDC（窗口客户区），画图通常通过 CClientDC 完成。

CWinApp 应用程序类。似于 C 中的 main 函数，是程序执行的入口和管理者，负责程序建立、消灭，主窗口和文档模板的建立。最常用函数 InitInstance（）：初始化。

CGdiObject 及子类，用于向设备文本画图。

3.8 功能详情

3.8.1 开始页面

可依次点击图标进行图像去块操作处理，也可点击菜单栏处理窗口设置。

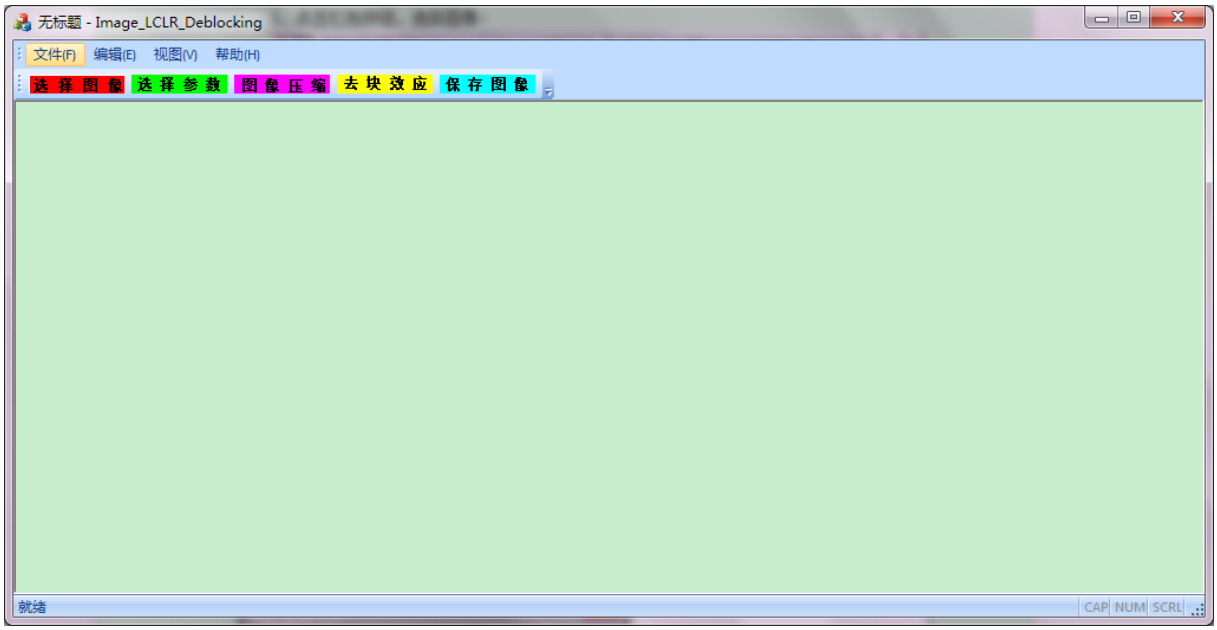

图 3-2 开始页面

3.8.2 选择图像

点击红色按钮选择图像，可自由选择系统中的测试图片。

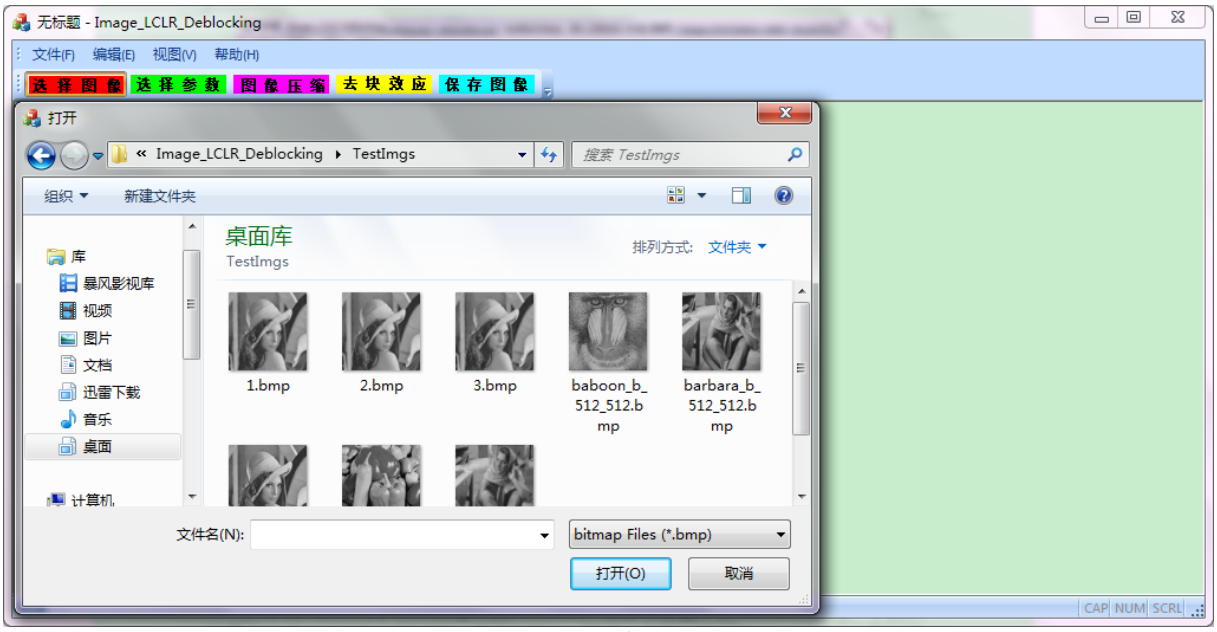

图 3-3 选择图像

3.8.3 选择量化表参数

通过选择量化参数等级，可使得图像压缩程度变化，块状效应更高。

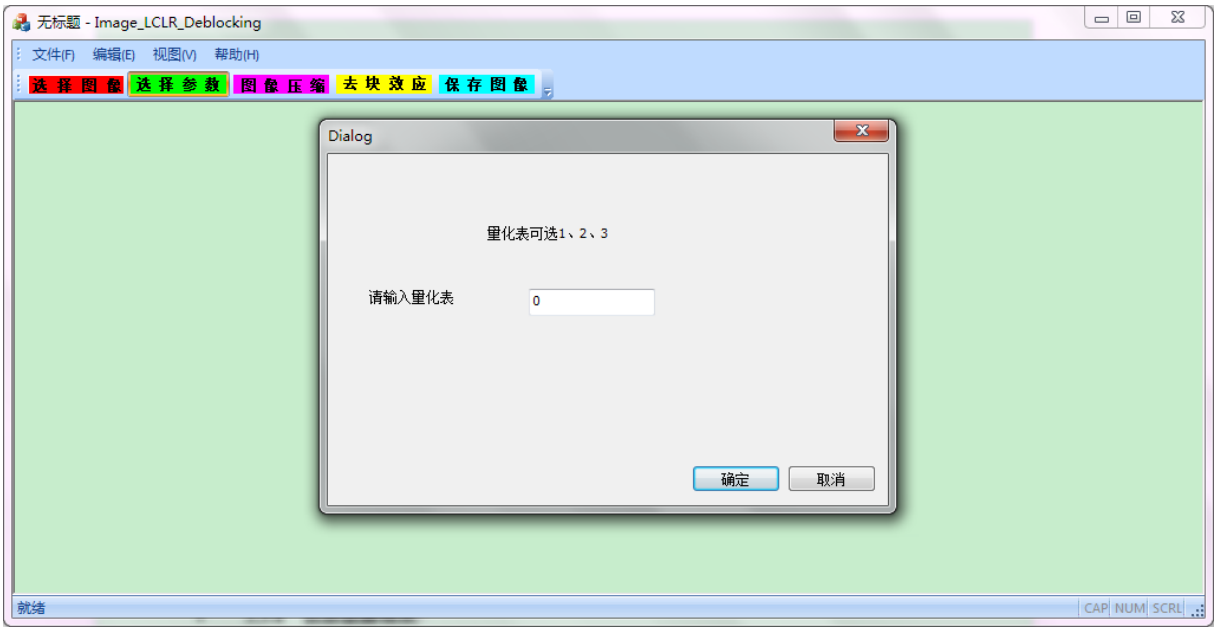

图 3-4 选择量化表参数

3.8.4 图像压缩

本模块可将清晰的测试图像压缩为块状效应图像，点击图像压缩按钮后，显示带有块状效应的图像，并可将图像保存。

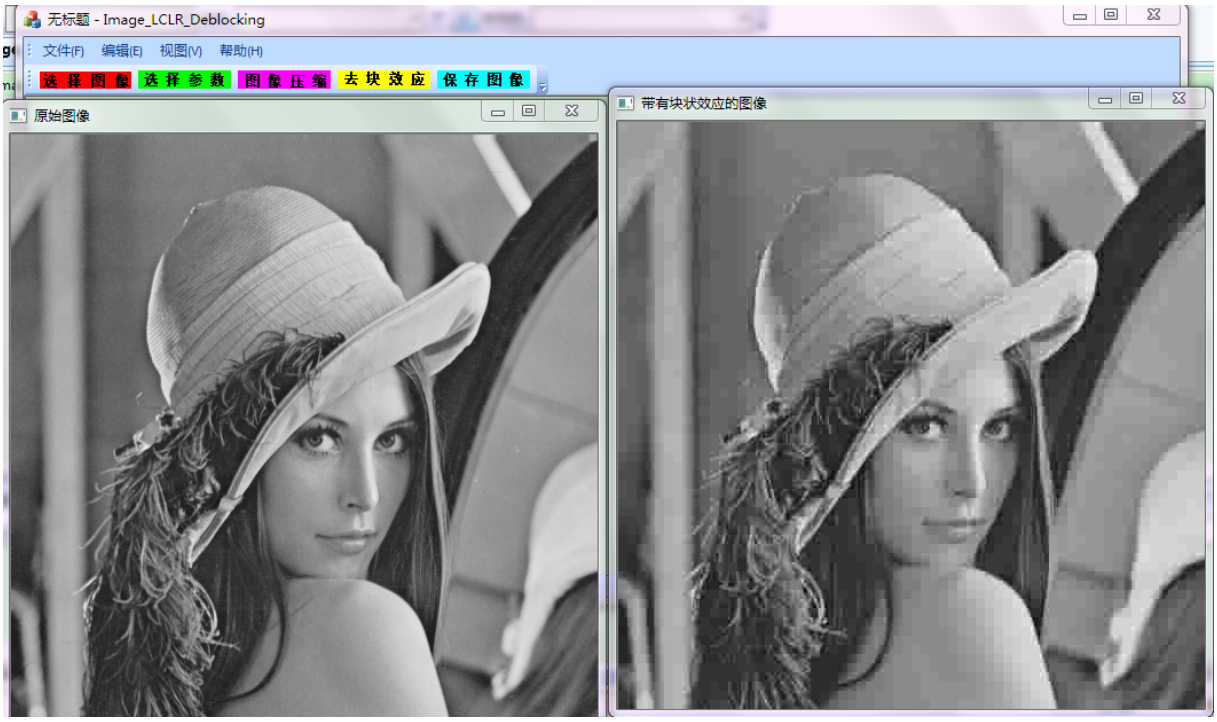

图 3-5 信息提醒模块

3.8.5 去块效应

点击去块效应按钮后，将会显示去块效应过程所耗时间，点击确定按钮后结束去块效应过程。

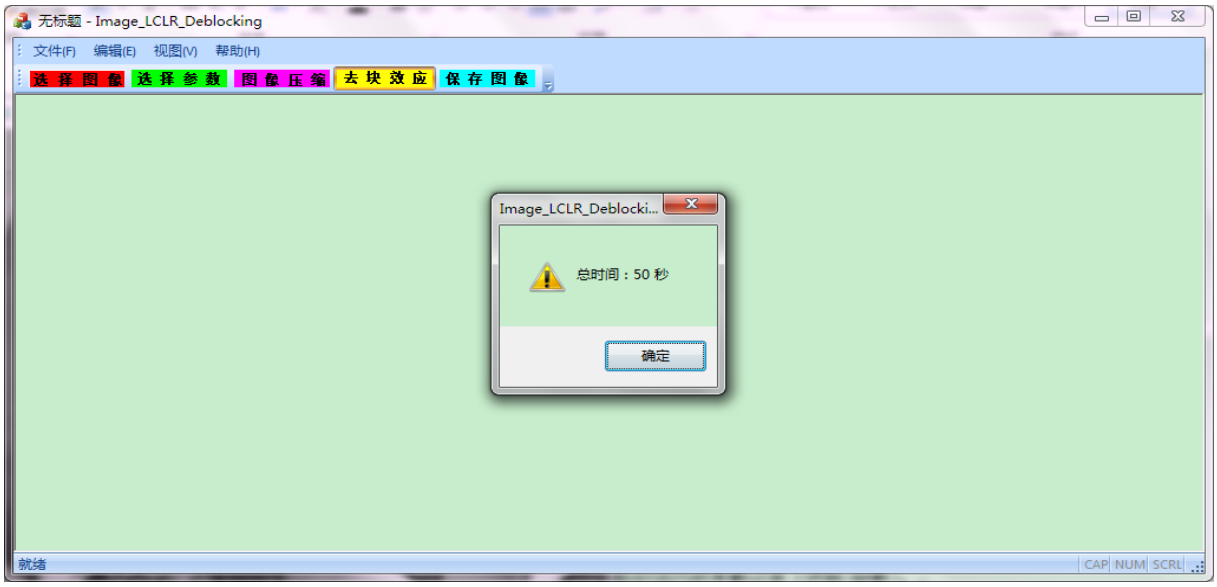

图 3-6 去块效应

3.8.6 保存图像

点击保存图像按钮可将之前去块效应得到的成果保存，保存完毕后将显示处理结果，可将其与之前的原始图像及带有块状效应的图像作对比。默认名为”Test”。

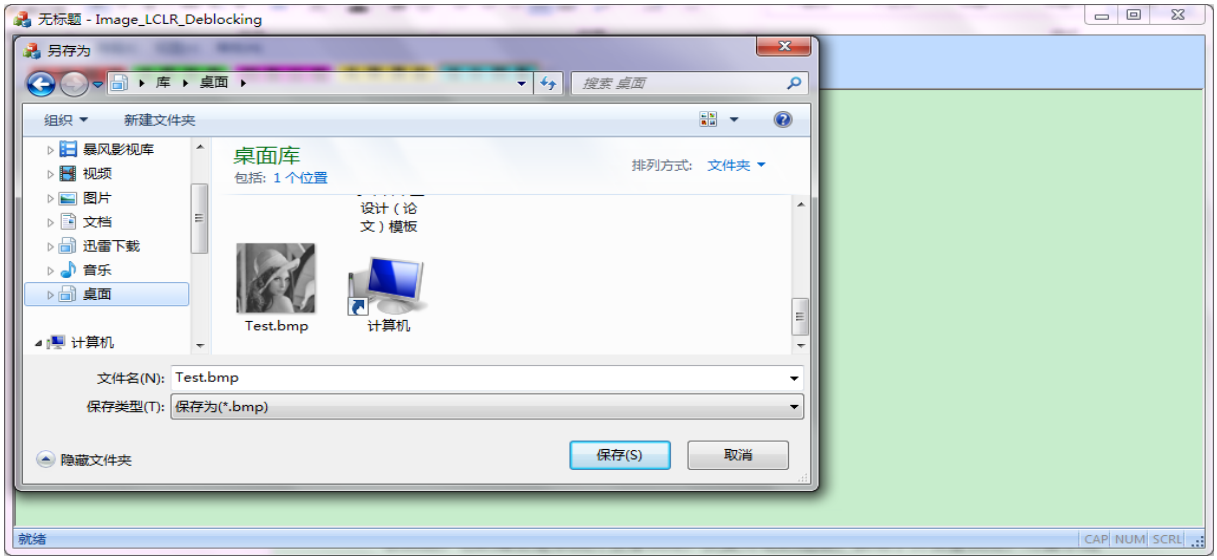

图 3-7 保存图像

第四章　算法设计

4.1　图像输入处理

在本次项目中，我使用 OpenCV 的 imread 函数和 imshow 函数读入和显示图像，设定的测试图像大小为 512*512，然后利用 OpenCV 的内置函数 convertTo 将图像存为 CV_64F 格式的矩阵，即 double 类型矩阵。

在用户选择完量化参数后，进行 DCT 变换、量化、光滑、反量化和 IDCT 变换。根据用户选择的量化参数选择量化表和噪声系数，进行完一系列三角函数计算后返回带有块状效应的图像矩阵。应用公式[10]如下：

对数据f[i][j]进行DCT变化生成F[u][v]
应用公式：
F(u,v) = alpha(u)*alpha(v)*sum
sum=f(x,y) * cos ((2*i+1)*u*PI/16)* cos ((2*j+1)*v*PI/16)
当 u=0 v=0 -->alpha(u)*alpha(v)=0.125 cos(u)=cos(v)=0
当 u=0 ||v!=0 -->alpha(u)*alpha(v)=sqrt(2.0)/8.0
当 u!=0&&v!=0 -->alpha(u)*alpha(v)=0.25

式(4-1)

对数据F[u][v]进行IDCT变化生成f[i][j]
应用公式：
f(x,y) =sum*alpha(u)*alpha(v)*F(u,v)*cos((2*i+1)*u*PI)* cos ((2*j+1)*v*PI)
F(u,v) = alpha(u)*alpha(v)*sum
Sum=f(x,y)*cos((2*i+1)*u*PI/16)* cos((2*j+1)*v*PI/16)

式(4-2)

得到带有块状效应图像后，将该图像作为输入数据，进入去块效应过程。

4.2　图像去块效应算法处理

去块效应算法部分的输入数据为 512*512 大小的经过压缩后具有块状效应的图像，先将图像数据进行降维处理，分成一个个小块图像，再将图像块进行块匹配后进行 SVD 算法，然后进行 Low-Rnak 和软阈值等操作将图像降噪，最后得到每个图像块合并后的原始图像，得到最后去块效应后的图像。

4.2.1　图像降维

先事先设置好图像块的合适大小为 6*6 的图像窗口。利用 OpenCV 里的方法将整个图像分为多个 6*6 的窗口图像，因为 OpenCV 和 C++的矩阵数组存储行列方式不同，利用 transpose 方式将每个图像块转置存储。然后利用 OpenCV 的 reshape 方式将每个图像块压为一维图像数据，即将图像数据存于一维向量中，便于后面的图像块的匹配。这样便得到新的图像矩阵，大小为 36*(M-f+1)(N-f+1)，只有前 507*507 的数据有相似图像块，

因为 512 不能完整被 6 整除。每个纵列为一个小图像块，整个矩阵构成了原来的图像，数据没有丢失。

4.2.2 图像块匹配

将之前的数据降维后的数据矩阵和原始图像矩阵输入图像块匹配过程中。本过程的目的是找到每个图像块周围最相似的 40 个图像块，查找范围预先设置为以 6*6 的图像块的左上角点为中心点的 22*22 的图像窗口。所以设计思想是事先将图像在上下左右方向每隔 6 个数据点定一个图像块起始点，在图像块起始点为中心作一个 22*22 的探测相似块窗口点，每个点代表一个 6*6 的图像块的左上角坐标，这样就可以完美地将 6*6 图像块周围的相似图像块都检测出来。若某些图像块起始点在图像的边缘部分，那就以边缘为准，检测相似图像块的数量就少些。

每个相似块都将左上角起始点的坐标存于图像降维步骤所处理好的新的图像矩阵中，即可将原始图像中图像块起始点坐标进行映射到新的图像矩阵的列号中，所以进行相似图像块匹配时所计算的图像像素差的平方是由一维图像矩阵决定的将 36*1 的图像矩阵相减再进行平方和操作，然后用 OpenCV 里的

cv::sortIdx(dis, ind, CV_SORT_EVERY_COLUMN+CV_SORT_ASCENDING)

式(4-3)

得到该图像块的相似图像块的像素差的升序排列，得到。最先相似的前 40 个图像块，将他们的序号映射到新的图像矩阵列号中，得到 36*40 的相似图像块矩阵。

4.2.3 奇异值分解

因为本项目是对图像进行去块效应，若一次去块效应处理结果不满意，可以再点击去块效应按钮进行迭代，直到得到用户想要的结果。其中一次迭代的输入参数的构造器为：

```
CLCLR_Deblocking::CLCLR_Deblocking(Mat m,int it)
{
    m_nR        =        m.rows;              // 矩阵行数
    m_nC        =        m.cols;              // 矩阵列数
    nim=m.clone();
    im_out=m.clone();
    par=new CParameter(it);
    iter=0;
    MainOperation();
}
```

式(4-4)

多次迭代的构造器为：

```
CLCLR_Deblocking::CLCLR_Deblocking(Mat or,Mat m,Mat bl,int it,CParameter *p)
{
    m_nR        =        m.rows;              // 矩阵行数
    m_nC        =        m.cols;              // 矩阵列数
    nim=or.clone();                          //原始图像
    im_out=m.clone();                        //输出图像
    par=p;                                   //原始图像
```

```
    iter=it;                                        //原始参数
    blk_arr=bl.clone();
    MainOperation();
}
```

式(4-5)

然后将输入的图像矩阵数据进行迭代正则化，取 lamada 参数为 0.63，这是一个预先设置的参数，可将前次迭代数据和原始图像进行加权叠加，获得正则化后的真正的输入图像矩阵。

另一个输入参数是 36*40 的相似图像块矩阵，先将该矩阵按行求均值，得到 36 个数据点的均值，然后原矩阵减去将该均值，实行去均值操作，得到 SVD 分解所需数据矩阵。

再将量化噪声更新，使得上一次的噪声数据矩阵可以更新到下一次迭代中使用。

在本次项目中输入的矩阵为 36*40 的矩阵。

M 是一个 m×n 阶矩阵，其中的元素全部属于域 K，也就是 实数域或复数域。如此则存在一个分解使得

$$M = U \Sigma V^*$$

式(4-6)

其中 U 是 m×m 阶酉矩阵；Σ是半正定 m×n 阶对角矩阵；而 V*，即 V 的共轭转置，是 n×n 阶酉矩阵。这样的分解就称作 M 的奇异值分解。Σ对角线上的元素 Σi,i 即为 M 的奇异值。

U 的列(columns)组成一套对 M 的正交"输入"或"分析"的基向量。这些向量是 MM* 的特征向量。

V 的列(columns)组成一套对 M 的正交"输出"的基向量。这些向量是 M*M 的特征向量。

Σ对角线上的元素是奇异值，可视为是在输入与输出间进行的标量的"膨胀控制"。这些是 M*M 及 MM*的奇异值，并与 U 和 V 的行向量相对应。

因为 U 和 V 向量都是单位化的向量, 我们知道 U 的列向量$u_1,...u_m$组成了 K 空间的一组标准正交基。同样，V 的列向量$v_1,... v_n$也组成了 K 空间的一组标准正交基(根据向量空间的标准点积法则).

线性变换[11] T: K → K，把向量N_x变换为M_x。考虑到这些标准正交基，这个变换描述起来就很简单了: T(vi) = σi u_i, for i = 1,...,min(m,n), 其中σi 是对角阵Σ中的第 i 个元素; 当 i > min(m,n)时，T(vi) = 0。

这样，SVD 理论的几何意义就可以做如下的归纳：对于每一个线性映射 T: K → K，T 把 K 的第 i 个基向量映射为 K 的第 i 个基向量的非负倍数,然后将余下的基向量映射为零向量。对照这些基向量，映射 T 就可以表示为一个非负对角阵。

在完成 SVD 奇异值分解后，得到了对于每个图像块周围相似图像块的奇异值分解的过程。对图像的各个方向的分量做分析，可以得到图像的像素移动趋势，可以在特征空间中分解奇异值从而得到多个特征值和权值，调整特征值和权值后，在还原的空间中得到最符合该图像块的每个像素的取值，从而使得该图像块完美嵌入其他的图像块，使得图像平滑，块状图像马赛克效应消失，特别是使得图像的不同物体边缘趋于平滑，图像识别率更高。

然后利用图像处理中的软阈值将图像降噪和 SVD 分解中的低秩近似表示。将 SVD 算法输出的最奇异值用软阈值处理，再进行降秩处理，使得重构后的图像更为低噪化。其中SVD的参数如下：

SVD参数

double value	最奇异值	Sigma0
double sv	显著方差值	sv
double thr	软阈值	thr
Mat S(Sigma0.rows,1,CV_64F)	降秩操作结果值	S

表 4-1

4.3 算法结构

算法描述：　　Image_LCLR_Deblocking
输入：带有块状效应的编码图像 Y

$$Y = \text{imread}(\text{imgpath}, 0);$$

输出：对于图像 Y 的解码图像 X
用Y初始化X^0　　k=0
While　k<max Iter do　//Iter 是用户想要的迭代次数
1、图像块聚集

Im2Patch(im_out);
Block_matching(im_out);

式(4-7)

2、迭代正则化

im_out　=　　im_out + lamada*(nim - im_out);

式(4-8)

3、量化噪声更新

sv=value*value/B.cols-(sigma)*(sigma);

式(4-9)

4、SVD 分解

CLow_rank_SSC　SplitUV(udata,vdata,m_data,0);

式(4-10)

5、阈值更新

par->sigma　= sqrt(fabs(vd))*par->lamada;

式(4-11)

6、图像更新

CLow_rank_SSC　thr=c1*sigma*sigma/sqrt(sv);

式(4-12)

k=k+1
End
Return X^k

对于整个项目的算法结构的架构我是这样完成的：

2 月 1 号到 3 月 20 号

完成了对论文的理解以及 Matlab 程序的理解，并且在 Visual Studio 下完成了整个程序，程序结果与 Matlab 一致，但是运行时间过慢，主要的问题是没有把 Matlab 的思想转换到 C++的思想、并且矩阵类是完全自己编写的，源 Matlab 程序有大量的对不连续行列的赋值拷贝，并且还有大量的排序、SVD 等算法后续采用 OpenCV 重新完成该项目

3 月 20 日到 4 月 20 日

这个一个月，我完成了在 OpenCV 下完成整个程序，OpenCV 有很多集成好的复制拷贝以及计算的方法，但是复制拷贝函数只能针对连续行列，为了提升速度我经过多次试验终于得到一种快速赋值的方法，并且将相似块查找部分时间提升很多，但是在 SVD 部分仍然时间很长，用 OpenCV 自带 SVD 函数需要跑三分钟，用纯 C++代码也需要一分钟

4 月 20 日至 5 月 10 日

在这一个月，我的工作重心在如何优化程序，我选择了用多线程并行运行程序，相似快查找部分在 2 s 内便可以跑完，SVD 部分在 10 几秒内可以跑完，相比源程序整体时间提升了 40%~50%，根据对现在一些论文的查询发现，例如一张 512*512 的图片，要对 40*36 的矩阵进行 10609 次 SVD 分解，这个时间是不可能少于 10s，由于 SVD 算法的本质导致。

第六章　优化方法

在调用之前的 OpenCV 库函数 SVD 分解后，发现耗时过长，这对于用户体验太差，于是我决定做些时间上的优化。方法所示如下：

6.1 行列赋值

Matlab 中有很多集成好的的行列赋值语句，特别是对几个不连续行、几个不连续列进行赋值拷贝特别迅速，首先，OpenCV 没有对不练续行不连续列的赋值拷贝，只有对连续行列的赋值拷贝，其次，即便是连续行列赋值，OpenCV 的函数也非常慢，所以源程序中大大量赋值语句必须需要自己写行列赋值语句。下面分析一下对矩阵 M 某一个元素（i,j）进行操作的方法。

方法一：M.at<float>(i, j)

方法二：M.ptr<float>(i)[j]

方法三：用一个指向 M.data 的指针 pdata
方法四：直接用 M.data 的指针
通过对 10000×10000 进行初试赋值计算时间得到以下列表

行列赋值结果

	Debug	Release
方法一	139.06ms	2.51ms
方法二	66.28ms	2.50ms
方法三	4.95ms	2.28ms
方法四	5.11ms	1.37ms

表 6-1

由于方法中经常有几万到几十万次几千次循环执行赋值，只能选择方法最快方法三和方法四，为了节省空间，选择方法四

6.2 SVD 分解

奇异值分解[12] (sigular value decomposition,SVD) 是另一种正交矩阵分解法；SVD 是最可靠的分解法，但是它比 QR 分解法要花上近十倍的计算时间。[U,S,V]=SVD(A)，其中 U 和 V 代表二个相互正交矩阵，而 S 代表一对角矩阵。 和 QR 分解法相同者， 原矩阵 A 不必为正方矩阵。使用 SVD 分解法的用途是解最小平方误差法和数据压缩
例如： 用 512×512 的图像，每一次迭代要对 40×36 的矩阵进行 10609 次 SVD 分解

SVD 分解结果

	Debug
matlab	21.1s
Opencv自带函数	312.5s
自己编写的SVD函数	58.8s
并行SVD函数	11.2s

表 6-2

6.3 openmp 多线程并行操作

OpenMP 是作为共享存储标准而问世的。它是为在多处理机上编写并行程序而设计的一个应用编程接口。它包括一套编译指导语句和一个用来支持它的函数库[13]。

对于一般单一执行绪（single thread）的程式，多核心的处理器并没有办法提升它的处理效能；不过对於多执行绪（multi thread）的程式，就可以透过不同的核心同时计算，来达到加速的目的了！简单的例子，以单执行绪的程式来说，一件事做一次要十秒的话，要做十次，都丢给同一颗核心做的话，自然就是 10 秒 * 10 次，也就是 100 秒了；但是以多执行绪的程式来说，它可以把这一件事，分给两颗核心各自做，每颗核心各做 5 次，所以所需要的时间就只需要 50 秒！

在经过一系列的优化后，我终于把去块效应的运行时间从 80 多秒优化到了 50 秒内，这是一个对于我个人项目的不小的突破。

第七章　图像数据集测试

结果测试(测试五次的平均值)

其中的图像均为收集完成的标准的初始图像集，图像处理成大小为 512*512 或者 256*256 的清晰图像集，均为经典的图像处理图像集，图像为.bmp 格式。

我将网上的去块效应的Matlab代码运行的数据与我的最终版本的VC代码进行比较，得出了如下数据。

其中可以明显看出，我的 VC 代码算法的运行结果好于 Matlab 版本的算法，因为 C++ 本身的运行速度较快，直接进行机器数据的操作，而且我将 OpenCV 原来自带的较慢的 SVD 算法进行改写，并改变 OpenCV 本身较慢的存取数据操作，并加入并行化操作，最终得到较好的结果。经过比较，图像大小对于算法的运行速度有较大影响，图像的细节越多，纹理质地越多，对于图像的处理速度耗时越长。

7.1 图像处理时间比较

图 7-1 Lena

'Lena'　512*512	总时间	相似块查找	SVD 分解时间
Matlab	34.554	11.569	19.261
VC	18.950	2.653	16.676

图 7-2 House

'House' 256*256	总时间	相似块查找	SVD 分解时间
Matlab	9.778	3.944	5.365
VC	4.726	0.593	3.604

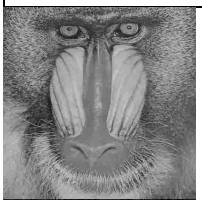

图 7-3 Baboon

'Baboon' 512*512	总时间	相似块查找	SVD 分解时间
Matlab	37.387	12.581	22.081
VC	24.301	2.405	21.355

图 7-4 Barbara

'Barbara' 512*512	总时间	相似块查找	SVD 分解时间
Matlab	36.721	12.152	21.926
VC	19.286	2.013	20.184

图 7-5 Peppers

'Peppers' 512*512	总时间	相似块查找	SVD 分解时间
Matlab	34.284	11.638	17.982
VC	19.761	2.262	15.408

7.2 图像去块效应平台测试

为了使错误得到反馈处理，使得界面更加人性化和友好化，我将用户输入数据和选取数据界面进行了优化，设置了提示窗口及误操作提示，可避免系统平台崩溃和算法运行失败。

以下便是几种操作提示：

在点击按钮选择图片时并未选择图片时，提示如下：

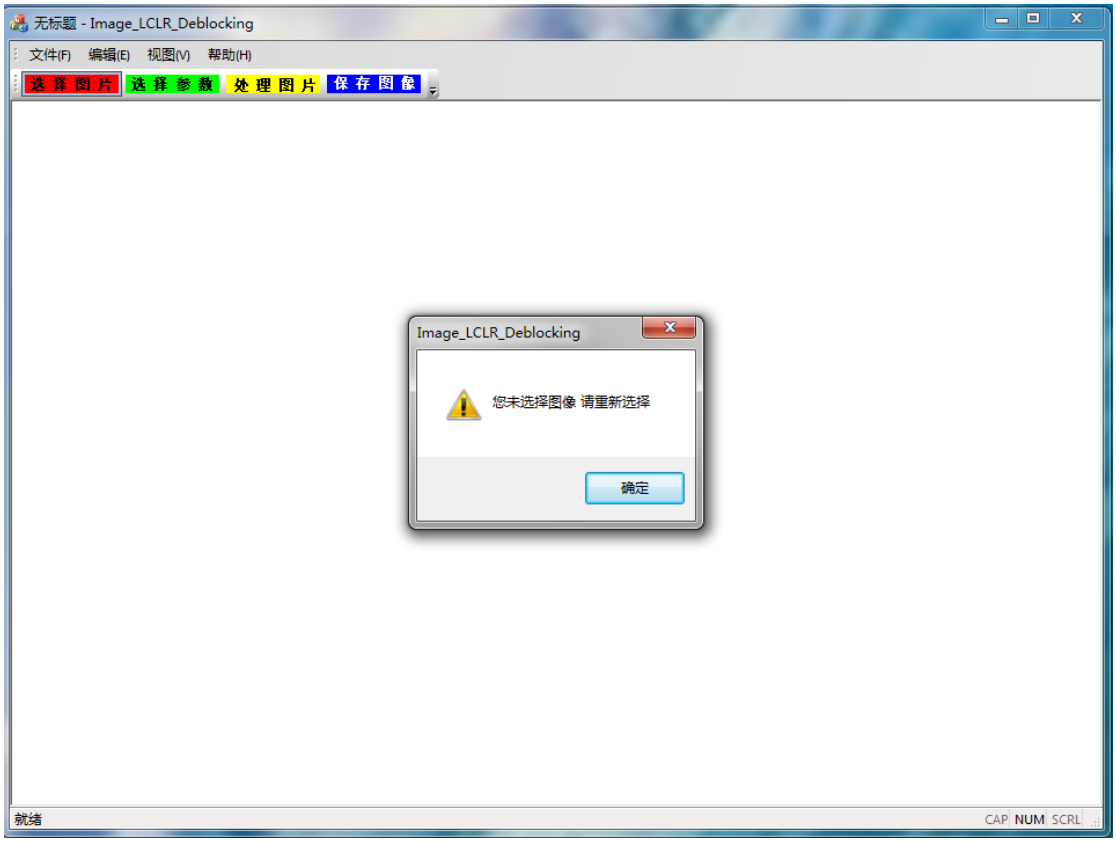

图 7-6 未选择图像

在点击按钮选择参数时并选择的参数不在范围内时，提示如下：

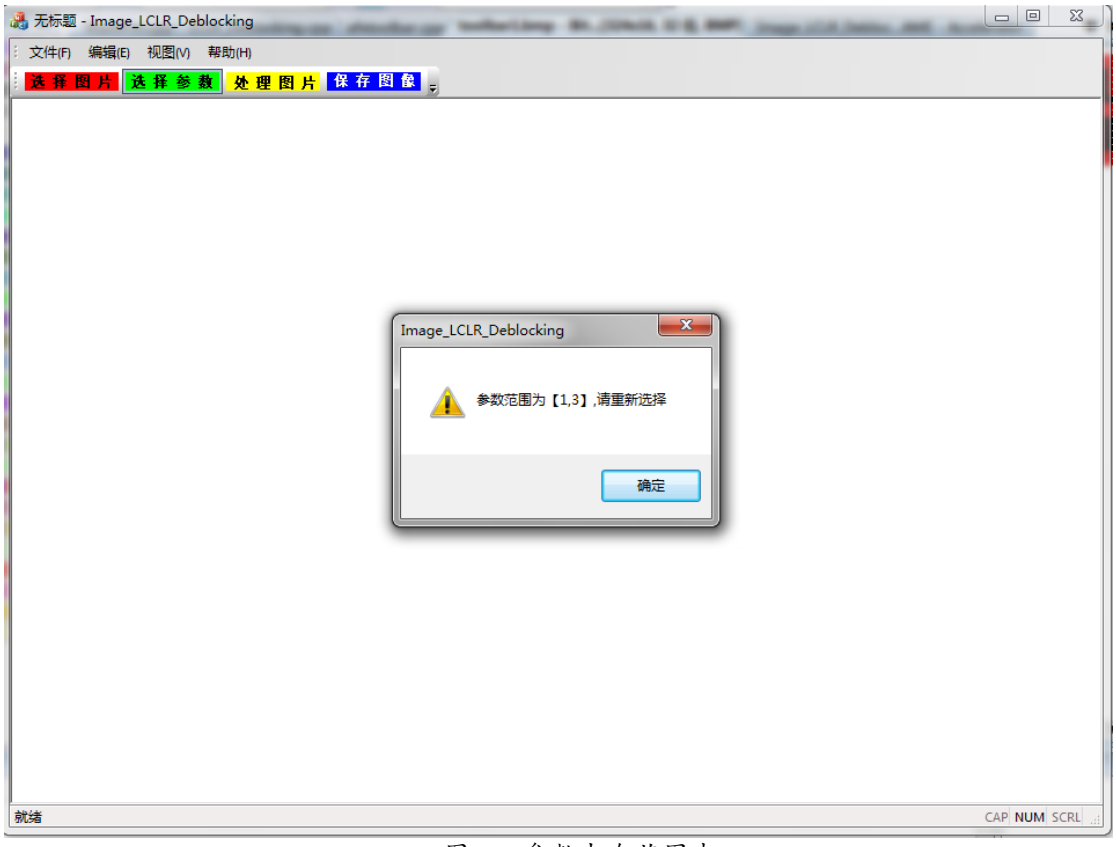

图 7-7 参数未在范围内

在点击按钮选择参数时并未选择参数时，提示如下：

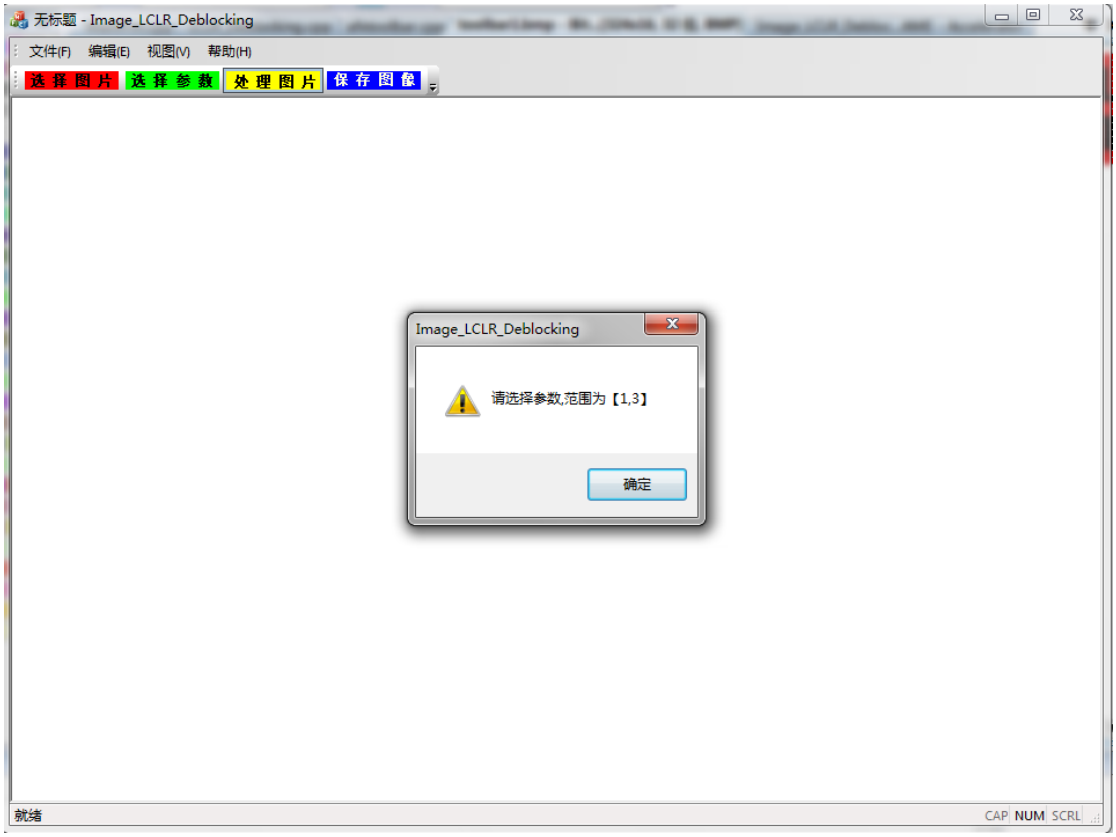

图 7-8 未选择参数

第八章　总结

图像处理去块效应技术对现在网络的发展和人们的生活质量非常重要，它的开发过程并不简单。必须严格按照软件开发流程，结合各种图像处理算法。才能开发出适合实际的有价值的和有计算效率的图像去块效应平台系统。通过本系统的开发，使我有了很大的收获，以下是对系统开发进行的总结以及一些心得。

系统的开发流程一般为：

1) 对需求进行分析，对系统的可行性和用例加以论证，并对系统进行了建模。
2) 对系统的架构划分了平台架构，并详细设计了 MFC 系统工作流程。
3) 对部分功能模块加以实现，并对图像处理和 OpenCV 等关键技术进行了研究。
4) 依据分块测试的方法对去块效应算法部分内容进行了测试，分部输出算法结果，得出了结论。
5) 把算法时间数据与已有的算法进行比较，并进行个人的改写，最后将耗时长的算法部分改写为并行算法和耗时耗资源少的算法。

个人心得：

1.在开始做这个毕设项目之前我对于 Matlab 以及 OpenCV 的掌握并不是特别的好，

通过几个月项目的锻炼，我会对于项目中的问题有了很多想法，并且更加理解了 OpenCV、Matlab 思想的不同

2.通过几个月论文的阅读和对于网上博客的学习，我对于这个方向有了一些深入的了解，我很喜欢在学习中进步的方式，我可以清楚的知道自己想要什么，需要学到什么，让我学到很多图像处理的办法

3.这一次我使用了 openmp 多线程处理手段，让我更加对于多线程有了充分的理解，以后遇到并行处理以及时间优化问题会采用这个办法

这是我第一次编写大型系统平台程序，需要将多种技术结合在一起。以前都是为达到某种效果的简单编程并不用考虑到用户的需求，无论用什么方法只要达到最后的目的就可以了。而本系统编写的过程中必须同时考虑到用户体验和算法效果。这对我的编程思路产生了很大的影响。编程思路的转变这对我来说是最大的收获。另外，在系统编写的过程中系统的学习了 MFC 知识。从安装部署到建立框架，再到编写界面过程，编程界面中的接口等等。对于 OpenCV 的使用更加熟练，对图像处理及算法迭代开发也有了一个全新的认识。而且对于 C++及 Visual Studio 的一些高级的功能都有了更加深入的了解。整个过程下来，我发现对于知识的了解不能只停留在书本上，要结合实际才能深入透彻的了解一门知识。在遇到问题时，要主动查阅资料和积极同学交流，向学长和学姐请教。随着开发的深入，我渐渐的掌握了一些编程技巧，了解了一些编程方面的规律。

总之，本次图像去块效应系统的开发过程对我来说不仅仅是一次毕设，这也是一次提高、学习的过程。通过本系统的开发也让我看到了自己知识结构、知识储备、学习能力上的一些不足。在今后的学习中我会不断努力提高自己各方面素质。为了自己的梦想而奋斗。

参考文献

[1]姚庆栋, 毕厚杰, 王兆华, 徐孟侠. 图像编码基础. 清华大学出版社. 2006: 17-18.

[2] 谢胜利, 石敏. 基于 DCT 域的高压缩图像去块效应算法. 控制理论与应用, 2006, 23(1):139-142.

[3] 胡栋. 静止图像编码的基本方法与国际标准. 北京邮电大学出版社, 2003: 21.

[4] 周炯槃. 信息理论基础. 邮电出版社, 1983: 15-16.

[5] 张春田, 苏育挺, 张静. 数字图像压缩编码. 清华大学出版社, 2006: 30.

[6] 张鑫. 数字图像的去块效应的研究. 重庆大学, 2009: 5-8

[7] 周长发. 精通 Visual C++图像处理编程. 电子工业出版社, 2004: 17-40.

[8] 黎松, 平西建, 丁益洪. 开放源代码的计算机视觉类库 OpenCV 的应用. 计算机应用与软件, 2005, 22(8):134-136.

[9] 胡伟, 王弘. 如何在 VC++中用 MFC 进行 OpenGL 编程. 计算机应用, 2001, 21(8):87-89.

[10] 胡乡峰, 卫金茂. 基于奇异值分解(SVD)的图像压缩. 东北师大学报：自然科学版, 2006, 38(3):36-39.

[11] 张学全, 顾晓东, 孙辉先. 基于自适应空域滤波的图像去块效应算法. 计算机工程, 2009, 35(4):218-220.

[12] 许志良, 谢胜利. 一种基于人类视觉系统的去块效应算法. 电子与信息学报, 2005, 27(11):1717-1721.

[13] Dagum, L, Menon, R. OpenMP: an industry standard API for shared-memory programming. 1998: 46-55.

致　谢

光阴荏苒，白驹过隙。转眼间四年的本科生活就要结束了。四年的大学生活，说不上精彩，但总归没有什么遗憾。四年来有苦有乐，哭过笑过。收获很多友谊、知识。这一路走来要感谢的人很多很多。

首先要感谢的是我的父母。四年来无论遇到什么样的困难，父母都是我最坚强的后盾，他们无微不至的关心我，即使不在身边，也都无时无刻不牵挂着我。正是由于他们的鼓励支持，教育关爱我才能走到今天。真世间也唯有父母的爱是从来不求回报，毫无怨言。感谢老爸老妈一直以来对我的支持，我会继续努力奋斗，不让他们失望。

其次要感谢是我的导师李巍海老师，在本次毕业设计中，我从李巍海老师身上学到了很多东西。本次毕业设计是在老师的亲切关怀和细心指导下完成的。这个设计和论文的完成，凝结着李老师的心血和汗水。李老师严谨的治学态度，开拓性的工作作风和科学的思维方法都使我受益非浅。李老师严于律己，忘我高尚的品德为我树立了很好的典范。李老师对我的设计和论文给予了莫大的关心和帮助，在此，我表示衷心的感谢和诚挚的谢意。

最后要感谢电子院的每一位教导过我的老师，感谢他们在四年生活中对我的谆谆教诲，对我世界观价值观的影响。这些都是受益一生的财富。感谢电子 8 班级光信息 8 班的全体同学。感谢与你们的相识、相知。让我收获很多。感谢一路上有你们，愿我们友谊长存。