

# 中国科学技术大学

# 本科毕业论文



## 基于样例的图像修复

院 系 信息学院 电子工程与信息科学系

姓 名 张博栋 学 号 PB10210189

导 师 张荣 副教授

日 期 2014 年 6 月



# University of Science and Technology of China

## Undergraduate Graduation Thesis



## Exemplar-Based Image Completion

**Department** Department of Electronic Engineering

and Information Science, School of

Information Science and Technology

**Name** Bodong Zhang **Student ID** PB10210189

**Advisor** Associate Professor Rong Zhang

**Date** June, 2014



## 致谢

光阴荏苒，转眼间本科四年时间即将过去，我也将离开科大踏往新的征程。回顾大学四年，我想在此先感谢各位教授过我课的老师，是你们将自己宝贵的知识和良好的做人品德无私地奉献、传授给我们才造就了一代又一代的优秀毕业生，也才成就了今天中国科大的辉煌。

特别要感谢张荣老师，我在上张老师的数字图像处理课时，就被张老师高尚的师德所感染。进了张荣老师的实验室之后，我得到了良好的科研条件，并且受到了耐心的指导，才使得我取得一定的进步。张老师对我新想法的鼓励给了我很大的动力，肯定了我的价值。在申请出国的过程中，张荣老师给予了我很大的帮助，并十分关切，经常询问申请进展。我还要感谢实验室各位研究生的很多帮助，尤其是薛玮玮师兄，我从进实验室起就受到了他耐心的辅导，使我能够一步步学习更多的知识和技能，还有黄俊师兄、薛迪秀师兄、张智瑞师姐等众多师兄师姐，谢谢你们的帮助和指导。

感谢大一信息学院三班班主任谭立湘老师和大二至大四 6 系班主任刘桂英老师，你们在各方面关注我们的成长，如同对待儿女一样地关爱我们。感谢身边的各位同学，我们共同在科学的道路上披荆斩棘，相互帮助，有你们的鼓励和支持才使得我坚持到现在。感谢我的信院三班舍友金鹏飞、孙鹏程、陈扬斌和 6 系舍友游昌盛、李皓、顾宇，我和你们度过了一段难忘又美好的时光。感谢有幸和黄双垒高中和大学 7 年同班，有幸和陈子豪、郭闻、薛鸿飞、王一凡、范以文、齐韦男、陶天训、杜璞、柏雪高中同班，又共同在中国科大度过四年，愿我们的友谊地久天长。

最后，我要感谢我的父母，你们给了我很大的精神上的支持。当我遇到困难时，你们安慰我，鼓励我，最终使我克服各种困难，当我成功时，你们分享我的快乐，谢谢你们！

# 目录

中文内容摘要.....	VIII
ABSTRACT .....	IX
第 1 章 绪论.....	1
1. 1 图像修复研究背景与意义 .....	1
1. 2 图像修复的种类及发展现状 .....	1
1. 3 图像修复的应用 .....	2
1. 3. 1 文物保护 .....	2
1. 3. 2 剔除多余物体 .....	3
1. 3. 3 去除多余文字 .....	3
1. 3. 4 影视特技 .....	4
1. 3. 5 老照片修复 .....	4
第 2 章 图像修复种类简介.....	6
2. 1 利用偏微分方程的图像修复 .....	6
2. 1. 1 整体变分法 .....	6
2. 1. 2 基于曲率的扩散模型 .....	8
2. 2 基于样例的图像修复 .....	8
2. 2. 1 贪心法 .....	8
2. 2. 2 图优化方法 .....	12
第 3 章 基于样例的修复算法的改进.....	13
3. 1 传统的基于样例的修复算法的缺点及改进思路 .....	13
3. 2 基于样例的修复算法的改进 .....	14
3. 2. 1 利用图像统计特性 .....	14
3. 2. 2 交互产生额外向量 .....	18
3. 2. 3 对图像进行分区 .....	22

3.3 算法实现 .....	29
3.3.1 算法描述 .....	29
3.3.2 算法流程图 .....	30
3.4 实验结果 .....	30
 第 4 章 实验界面设计 .....	36
4.1 制作 MASK 图 .....	37
4.2 制作分区图 .....	42
4.3 载入图像 .....	49
4.4 运行 .....	50
 第 5 章 总结 .....	55
 参考文献 .....	56

## 中文内容摘要

图像修复主要解决的问题是利用图像的结构、纹理等特征，来修复图像中缺损的部分，使得最终得到看起来真实的图片。它有着巨大的应用范围，可以修复中世纪画作，还可以对视频中的文字进行去除，剔除照片中不想要的物体，如划痕，附加物等。图像修复的两大难题是图像信息量太大导致的修复速度很慢以及图像纹理、结构的复杂性导致修复后的图像出现看起来不真实的效果。在图像破损区域宽度较小时，则可以利用热传导模型等传统偏微分方程法进行像素级的从源区域到缺损区域的渗透。而在图像破损区域较大时，传统修复方法会导致图像的模糊与失真。

本课题主要研究较为复杂的拥有大块缺损区域的图像修复，主要采用基于样例的图像修复法（Exemplar-Based Image Completion）。基于样例的图像修复是一种基于块操作的修复算法。首先，根据缺失区域边缘的信息在源区域中寻找最相似的块，然后将此块中的像素复制到缺损区域对应的块中。这个块复制过程会一直持续到所有的缺损区域全部被修复完成。

本文在基于样例的图像修复的基础上加入了诸多改进，提出了一种交互产生分区算法，改善修复质量，修复了一些先前图像修复算法做出的失败作品，并且结合图像的稀疏统计特性，实现了快速修复，减少了计算量，大幅缩短修复时间。另外，本文通过加入用户交互，可以额外产生块偏移向量，更加保证了修复的质量。

关键词：数字图像处理，图像修复，基于样例的图像修复

## ABSTRACT

Image completion (Inpainting) solves the problem of filling missing regions in a visually plausible way with consistent textures and continuous structures. It has wide applications, such as filling the gaps in paintings created in Middle Ages, words removal in videos, objects removal in images, like scratches and extra objects. The two main problems in image completion are low processing speed caused by large amount of information in an image and unreal effects caused by complexion of image texture and structure. When a missing part is narrow, Partial Differential Equations method like heat equation penetrates pixels from source region to missing region and has good results. However, when the broken region is large, traditional filling method would lead to blur and distortion.

The thesis mainly focuses on image completion with large missing region, which bases on Exemplar-Based Image Completion. Exemplar-based image completion heals images based on patch level. First, it looks for the most similar patch in source region of an image on the basis of the margin of missing region and then copies pixels from that patch into corresponding patch in missing region. This process continues until all the missing pixels are healed.

In this thesis, many improvements have been created and achieved. Divide-into-parts algorithm by using user interaction is created and improves inpainting quality significantly. Many failed results could be healed well by using this algorithm. Statistical Image Completion is achieved, running time is greatly reduced and computer needs less calculating. Moreover, by adding user interaction, we get extra offsets so image completion quality is guaranteed.

Key words: digital image processing, digital image completion, exemplar-based image completion



## 第1章 绪论

### 1.1 图像修复研究背景与意义

图像修复就是利用图像中未破損区域的信息，来修复图中破損区域，并且使得看起来具有真实的效果，称做 Image Completion，又叫 Inpainting。在实际操作中，需要事先人为指定图像的破損区域和源区域，并只对破損区域进行修复，重新着色。图像修复可以追溯到文艺复兴时期，那时人们为了修复中世纪的画作，常对画作中的缺损、掉色等区域进行修补，尽量恢复原来的面貌，使得看起来栩栩如生。然而当时并没有数字技术，修复需要非常有经验的画家直接在原作上进行绘制，所以具有很大的风险，并且费时费力。随着科技的进步，数字化修复成为可能，我们不必再冒很大的风险，只需要将原作进行数字化采集，在计算机上进行尝试。现今的图像修复技术已经是数字图像处理、计算机视觉中的重要领域，在文物保护、特效制作、去除图片和视频中的文字、多余景物等众多方面有着广泛的应用价值。在视频中，经常会出现标识文字或者由于传输导致的部分图像损坏，这些都可以通过图像修复技术得到良好的解决，总之，图像修复技术是当今的研究热点，正受到越来越多人的关注和重视。

### 1.2 图像修复的种类及发展现状

在当今图像修复研究领域，主要存在两种算法，第一类方法称为基于偏微分方程的修复算法[1][2]，最早是 Bertalmio 等人[1]模拟艺术家进行古文物修复所提出的方法，这种方法基于缺损区域边缘颜色平滑和结构连续的假设，将边缘上的已知像素信息逐渐传递到未知区域中。其中 Photoshop 软件中的很多修复工具就是利用这种方法进行修复，修复速度达到交互级。这种方法对于小尺度缺损区域如划痕，噪点等效果较好，但是对于大面积的缺失区域会产生模糊效应，极大影响修复效果。第二类方法称为基于样例的修复算法，该方法基于图像纹理块进行操作，通过从已知区域拷贝像素块到未知区域完成修复。其中 Criminisi 等人[3]提出了一种考虑结构重要性的算法，使得具有复杂结构线的图像区域被优

先修复。这种方法简单且相当有效，可以应付大面积缺损区域的修复。但是，这种方法运行速度较慢，每次复制一个小块到缺损区域都要遍历全图搜寻最佳源块，极其耗时，一张长宽为几百像素的图像修复时间长达几十分钟，并且对于过于复杂的纹理不能很好地修复。Fang [4]等人又提出一种先训练出纹理的特征再进行修复的方法，但是速度仍然较慢。Sun[5]等人又提出可以在图像上画线体现出结构线的方法提高结构复杂图像的修复质量。最近，He 等人[6]利用大量实验证明了自然图像的块偏移具有稀疏特性并且少量的块偏移就可以反映出图像的整体结构，提高了修复速度。

### 1.3 图像修复的应用

#### 1.3.1 文物保护

在文物保护上，数字图像修复技术提供了另一种安全的修复方法，使得人们不再需要直接在原图上进行修改，保护文物，避免了不再受到二次损害的可能性。



图 1.1 文物保护中的数字图像修复



图 1.2 文物保护中的数字图像修复

### 1.3.2 剔除多余物体

在一张图像中，常常会出现我们不想要的物体，这时可以利用图像修复技术，将不想要的物体设置为破损区域，进行填补。



图 1.3 利用图像修复去除窗台上的南瓜（图片来自[5]）



图 1.4 去除多余人物

### 1.3.3 去除多余文字

有些图像可能经过人为处理，导致图上增加了文字，但是可能其他人并不想要这些文字，如果用图像修复就可以很好地解决这个问题。

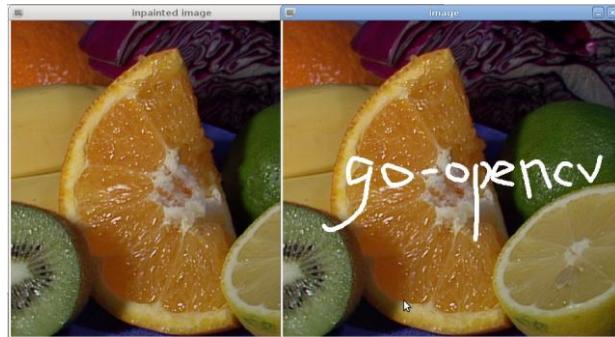


图 1.5 修复多余文字

#### 1.3.4 影视特技

在特技动作中，我们常常需要将道具进行抹除，才能达到真正的效果。我们可以利用图像完美再现腾空等动作。



图 1.6 将蹦极绳索修复掉，产生凌空飞行的效果

#### 1.3.5 老照片修复

老照片修复是图像修复中最贴近人们生活的应用领域，由于照片存放时间过长，常导致不同程度的受损，如油污，划痕，掉色等，但是这些照片大都对拥有者具有重要的意义，所以老照片的修复是很重要的一个应用。



图 1.7 修复老照片



图 1.8 修复老照片



图 1.9 修复老照片

## 第 2 章 图像修复种类简介

图像修复中，主要有基于偏微分方程修复和基于纹理合成修复两类，前者依据偏微分方程模型，通过确定传播信息和传播方向，实现区域边界外围信息向内扩散，后者在可用的区域寻找近似区域，拷贝到受损区域，重建图像细节信息。本章将详细介绍各种修复方法。

### 2.1 利用偏微分方程的图像修复

#### 2.1.1 整体变分法[7]

在众多图像处理的实际应用领域，有很多人利用了数学物理方程的研究成果，利用偏微分方程的图像处理逐步发展起来，现已成为一种实用、有效、严谨的方法，其中一个重要的分支就是整体变分法，并正受到国内外很多研究者的关注。整体变分法起初是由 Rudin[8]等人提出的，一开始的目的是为了去除图像中的噪声以及不必要的微小细节，并且在操作过程中保护像素颜色的突变，防止破坏边缘。直至后来整体变分法才被用来图像修复。整体变分法的最初模型如下：

$$\min_u TV(u), \text{ subject to } \|u - u_0\|^2 = \sigma^2,$$

$$TV(u) = \int_{\Omega} |\nabla u| dx dy = \int_{\Omega} \sqrt{u_x^2 + u_y^2} dx dy \quad (2.1)$$

$u(x, y)$  代表恢复的图像， $u_0(x, y)$  是带噪观测图像， $\sigma^2$  是图像中的噪声分布。TV( $u$ ) 是图像的整体变分公式，通过求解上式即可去除噪声，保留图像的边缘。

Rudin[8]等人注意到，有噪图像的整体变分值比无噪图像的整体变分值大很多，所以用此方法可以有效地去除噪声。

去噪效果如图：

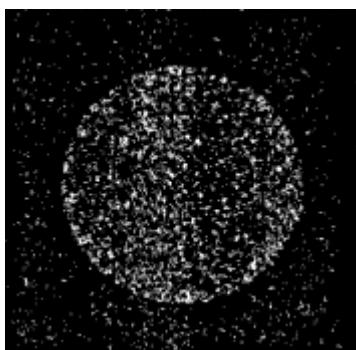


图 2.1 去噪前（图像来自[7]）



图 2.2 去噪后（图像来自[7]）

在进行图像修复的研究中，人们意识到一种常用的修复方法是自然地补全等照度线。于是 Chan[9]等人提出了一种基于整体变分的图像修补方法，基本模型（简记为 ROF TV 模型）如下：

$$\min \int_{\Omega} |\nabla u| + \lambda \int_{\Omega \setminus D} (u - u_0)^2 dx dy \quad (2.2)$$

D 代表缺失区域， $u$  代表出去缺失了区域 D 的观测图像

求解上式需要设定一个空间可变的标准化的参数  $\lambda_e(x)$ ，在区域 D 时， $\lambda_e(x)=0$ ；在区域  $\Omega \setminus D$  时， $\lambda_e(x)=\lambda$ 。如此设定后，在区域 D 是在做图像修复操作，在区域  $\Omega \setminus D$  是在做去噪操作。

利用整体变分法进行图像修复的结果如下图所示

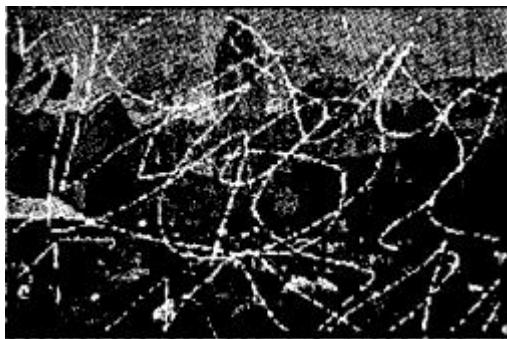


图 2.3 待修补图像（图像来自[7]）



图 2.4 修补后图像（图像来自[7]）

整体变分法的缺陷：

- 1 **图像的对比度会下降。** 当在一个限定的图像区域上定义整体变分模型时，我们会用它的均值对图像进行标准化，最大像素值和最小像素值的差会变小，从而使得对比度下降。
- 2 **块效应。** 使用整体变分法后，去噪后可能出现阶梯式的像素值现象，即在某些段很平缓，段与段之间存在突变。整体变分法尽可能地减小了震荡，导致在某些段上非常平滑，却失去了段之间的连接性。
- 3 **丢失图像的纹理信息。** 整体变分模型对于去噪有良好的效果，也可以有效地保护重要的轮廓。但是在去噪的同时，会把图像的小纹理当成是噪声去除，因为小纹理的波动性与噪声很像，难以分辨，导致图像丢失纹理，变得模糊。
- 4 **只对窄区域有良好修复效果。** 由于归根结底采用的是偏微分方程的方法，在像素向待修复区域扩散过程中主要依赖于梯度值，无法保持图像原有的纹理信息，如果缺损区域很大的话会造成较大的模糊。

### 2.1.2 基于曲率的扩散模型[10]

基于曲率的扩散模型[10]是整体变分模型的演化，整体变分模型的欧拉-拉格朗日方程为：

$$\operatorname{div}(\nabla u / |\nabla u|) + \lambda_e(u - u_0) = 0 \quad (2.3)$$

其中梯度下降

$$\partial_t u_i = \operatorname{div}(\nabla u / |\nabla u|) + \lambda_e(u - u_0)$$

令  $g(|\nabla u|) = 1/|\nabla u|$ ，与热扩散方程相比，整体变分图像修复模型相当于热传导系数为  $\frac{1}{|\nabla u|}$  的扩散，即扩散的强度只取决于梯度值而不取决于等照度线的曲率信息。根据以上分析，曲率推动扩散修复模型在整体变分图像修复方法的基础上将传导系数  $\frac{1}{|\nabla u|}$  改为  $\frac{g(k)}{|\nabla u|}$ ，从而引入曲率项，得到  $\partial_t u_i = \operatorname{div}\left(g(k)\frac{\nabla u}{|\nabla u|}\right)$ 。其中， $k = \operatorname{div}\frac{\nabla u}{|\nabla u|}$ ，是水平线曲率， $g(k)$  是关于曲率的单调递增函数，所以叫曲率驱动扩散。其中

$$g(k) = \begin{cases} 0 & s=0 \\ \infty & s=\infty \end{cases}, \text{ 通常取 } g(k) = k^p (1 \leq p) \quad (2.4)$$

加入惩罚后，扩散强度不但取决于梯度值，还取决于等照度线的曲率。这样就使得等照度线逐渐拉伸，变得更加平缓而不会出现曲率很大的情况。

## 2.2 基于样例的图像修复（纹理合成法）

### 2.2.1 贪心法

Criminisi 等人[3]提出了一种新的贪心法图像修复技术，在破损区域很大的情况下依然能够避免模糊现象。其基本原理如下。给定一个输入图像  $I$ ，其中缺损区域为  $\Omega$ ，完好区域为  $\Phi$ ，即  $\Phi + \Omega = I$ ，缺损区域边界为  $\delta \Omega$ 。贪心法就是要在  $\Phi$  区域内寻找最相似的块，将其中的信息复制到缺损区域  $\Omega$  内，从而进行修复。

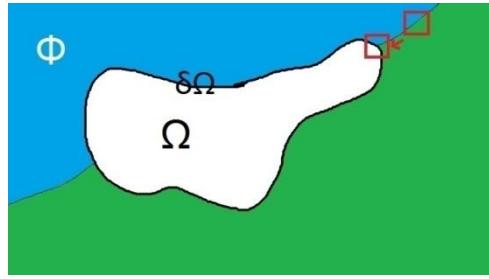


图 2.5 修复算法原理图

通常块的大小可以选取  $7 \times 7$ , 这样可以既使得块中保留了纹理信息, 又使得修复过程更加细腻, 提高修复质量。将某一块修复完成后, 将会改变缺损区域的大小和边界信息, 进行新的迭代 (如下图 2.6 所示), 缺损区域会逐渐从外向内缩小, 直到将所有缺损区域填充完成。

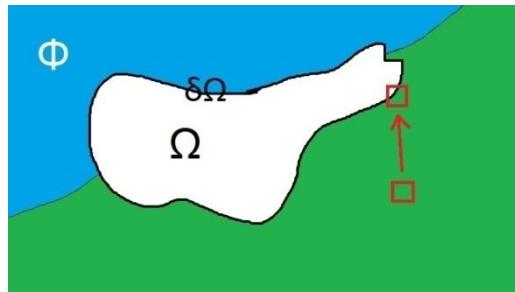


图 2.6 进行新的修复

每次迭代都按照三步进行:

### 1. 计算块优先权

首先我们需要定义块修复的优先顺序, 决定哪一个块要被最先修复。计算优先权时, 主要考虑一下两点: 1) 块是否包含了重要的结构或边缘 2) 块是否含有足够的置信度。基于这种考虑, 我们这样定义优先权的大小:

给定一个块  $\Psi_p$ , 中心点  $p$  在缺损区域的边缘, 即  $p \in \delta \Omega$ , 则  $P(p)$  是两项的积形式。

$$\begin{aligned} P(p) &= C(p) D(p). \\ \text{其中 } C(p) &= \frac{\sum_{q \in \Psi_p \cap (\Omega - \Omega)} C(q)}{|\Psi_p|} \\ D(p) &= \frac{|\nabla I_p^\perp \cdot n_p|}{\alpha} \end{aligned} \quad (2.5)$$

$|\Psi_p|$  是  $\Psi_p$  的面积,  $\alpha$  是标准化因子,  $n_p$  是与边界切向垂直的单位向量,  $\nabla I_p^\perp$  是梯度, 在初始时,  $C(p)=0$ ,  $\forall p \in \Omega$ ,  $C(p)=1$ ,  $\forall p \in \Phi$ 。

$C(p)$ 的作用是衡量块的置信度，在初始时就是块中包含源区域信息的像素点的个数占总像素点的个数之比。 $C(p)$ 越大，则能提供的可靠的信息就越多，越不易受到噪声的影响。可以发现，在缺损区域的拐角处  $C(p)$  较大，我们在修复时趋向于先修复这样的块。显然，越靠近缺损区域中心， $C(p)$  越小，越被后修复，这样，修复时首先从边缘修复，逐渐向里延伸，直至修复完成。

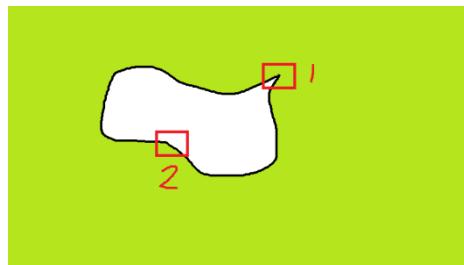


图 2.7 优先修复块 1

如图 2.7 所示，贪心算法趋向于优先修复块 1 而不是块 2。块 1 中源区域像素点个数占总像素点个数比重较大，约 80%，所以  $C(p) \approx 0.8$ 。对于块 2，源区域像素个数约占 30%，所以  $C(p) \approx 0.3$ 。

$D(p)$  函数的作用是考虑到图像的结构信息，优先修复含有重要结构线的块，以期望把残缺的结构线连接起来。因为结构线是图像的重要信息，也决定了图像修复的成败，所以先将重要结构修补完成有利于提高修复质量。

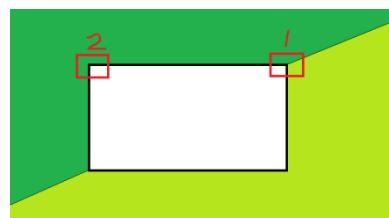


图 2.8 优先修复块 1

如图 2.8，贪心算法优先修复块 1，因为块 1 中包含了结构纹理，优先修复有利于还原出结构线。则可能的修复中间结果为图 2.9，但是如果没有  $D(p)$  这一项，就很有可能出现错误的复制，可能导致的中间结果为图 2.10。（下面两幅图是为了说明情况的假想图，真实修复中间结果可能与之不同）

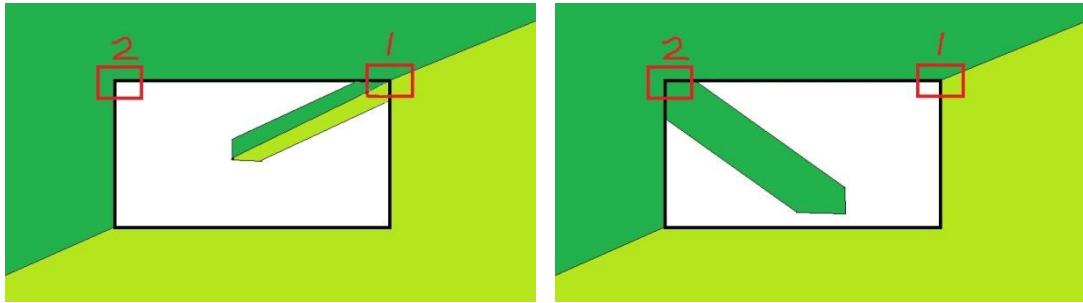


图 2.9 正确的修复顺序

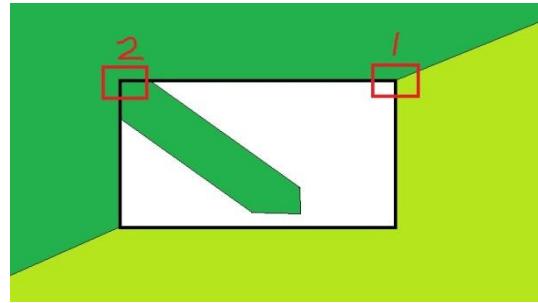


图 2.10 错误的修复顺序

综合考虑，将  $C(p)$  和  $D(p)$  的乘积作为决定优先权的标准，算法将计算所有中心点在缺损区域边缘的块的优先权大小，并找出最大的一个准备进行修复。

## 2. 寻找最佳源块

当我们找出最先要修复的块  $\hat{\Psi}_p$  后，就需要在源区域  $\Phi$  中寻找最相似的块，然后把源块的信息复制到目标块内。我们通过计算源块与目标块的距离来衡量相似度，得到最相似的源块为  $\hat{\Psi}_q$

$$\hat{\Psi}_q = \arg \min_{\Psi_q \in \Phi} d(\hat{\Psi}_p, \Psi_q) \quad (2.6)$$

两个块  $\Psi_a$  和  $\Psi_b$  的距离  $d(\Psi_a, \Psi_b)$  定义为对应像素点的差的平方和。找到最相似的源块  $\hat{\Psi}_q$  后，就将块  $\hat{\Psi}_p$  中缺损部分  $\hat{\Psi}_p \cap \Omega$  的像素点用源块对应点填上，将像素值复制过去。

## 3. 更新置信度信息

每进行一次块复制，都会有一部分缺损区域被填充上，从而改变了缺损区域的形状、面积，被填充上的区域的置信度  $C(p)$  更新为

$$C(p) = C(\hat{\Psi}_p) \quad \forall p \in \hat{\Psi}_p \cap \Omega \quad (2.7)$$

此值介于 0 和 1 之间。

按照这三个步骤不停重复迭代，最后则将所有缺损区域修复完毕为止。

贪心法图像修复结果如下图 2.11、图 2.12 所示：



图 2.11 原图 (图像数据来自[3])



图 2.12 修复后的图(图像数据来自[3])

## 2.2.2 图优化方法

基于图优化的方法是将整个图像看成是一个结构，在寻找相似块匹配上，不光考虑源块与目标块的相似性，还考虑被修复的块与块之间的平滑性。图像修复的算法可以转换成基于图结构的马尔科夫随机场问题，也就是对于某一位置的块或者像素点看做图结构中的节点，每个节点有不同的候选向量，利用候选集中的向量可以寻找到合适的源块进行修复，候选集中每个成员可以定义一个能量，被修复的相邻块之间也有交互能量。这样，就可以转化成能量最小化问题[11]。

能量优化公式为

$$E(L) = \sum_{x \in \Omega} E_d(L(x)) + \sum_{(x, x') | x \in \Omega, x' \in \Omega} E_s(L(x), L(x')) \quad (2.8)$$

$x, x'$  分别为某一像素点的坐标， $L(x)=i$  表示我们将  $x+\tau_i$  上的像素复制到  $x$  处，其中  $i$  表示候选集中某一向量的标签号。 $E_d$  是数据系数，用来表示编号是否有效，如果  $x+\tau_i$  在源区域，则  $E_d$  为 0，否则为无穷大。 $E_s$  为交互项，用来衡量领域平滑度，设领域某两个像素标签号为  $L(x)=i, L(x')=j, i \neq j$ ，则  $E_s$  为

$$E_s(i, j) = \|I(x + \tau_i) - I(x + \tau_j)\|^2 + \|I(x' + \tau_i) - I(x' + \tau_j)\|^2 \quad (2.9)$$

$I(x)$  表示  $x$  处的像素值，定义能量之后，就可以通过求解最小能量来完成图像修复。

通过全局最优求解得到的修复结果具有一定优势，但是由于每次对块的修复都会影响自己和周围的能量值，牵一发而动全身，所以计算量非常巨大，算法效率较低，且算法没有考虑结构信息，对于结构特别复杂的图像修复结果也较差。

## 第3章 基于样例的修复算法的改进

### 3.1 传统的基于样例的修复算法的缺点及改进思路

本文主要针对拥有大块缺损区域的图像修复，而偏微分方程修复法只适合小区域的修复，所以采用基于样例的图像修复作为基本方法。鉴于本人的前期实验室研究工作，采用基于样例的图像修复的一个重要分支——贪心法为基本框架，并对算法进行了很多改进。

贪心法是在目前图像修复中常用的一种方法，由于是基于块的操作，所以完美地避免了传统的基于偏微分方程的修复方法中的易造成模糊的缺点，保留了重要的纹理特征，使得更加接近真实效果。

同时，贪心法也存在很多缺点。首先在算法效率上有很大的提升空间，由贪心法的算法流程可知，每找到一个需要修复的块后，都需要遍历整张图中的源块进行寻找匹配计算相似度，假设图像尺寸为 $500 \times 500$ ，块的大小为 $7 \times 7$ ，那么每迭代一次需要寻找的源块数量级为 $500 \times 500$ ，每次源块与目标块的相似度需要约 $3 \times 7 \times 7$ 次计算，需要迭代的次数也就是需要修复的块数量数量级约为 $500 \times 500 \div (7 \times 7)$ ，那么不算上计算块的优先级也要约 $500 \times 500 \times 3 \times 7 \times 7 \times 500 \times 500 \div (7 \times 7) = 187500000000$ 次运算。如果采用随机抽取源块的方法，虽然速度会提高，但是可有可能会错过最佳源块，导致修复结果质量差。

另外，由于贪心法在修复图像时基于块操作，在每次把源块信息复制到目标块前，都只是基于缺损区域边缘几十个像素的信息来决定最佳源块，而没有从整张图和图像实际内容及意义上考虑应该如何进行修复，所以随着修复逐步深入到缺损区域中心，逐次累积导致的误差就会加大。虽然贪心法考虑到结构线问题，可以在一定程度上合理修复出较为明显的结构线，但是如果结构线断裂较长，会逐渐累积误差导致错误的结构线走向，另外对于复杂的结构仍然无法有效修复。

基于以上缺点，我们有以下改进的思路。首先是算法效率问题，如果查找全部源块，消耗的时间会非常长，如果纯随机查找，质量无法保证。所以一种思路是有倾向性地查找成为最佳源块概率较大的地方，这样既可以使运算时间缩短，又可以很好地保护修复质量。然后是修复质量问题，创新的思路是可以在修复前增加简单的人为交互，给修复操作进行暗示。因为计算机目前还没有达到高度智能化，不可能基于图像意义对所有破损图像进行完美的修复，而对修复图像进行

评价的却是用人眼进行几十年图像采集、处理和分析的人类，这种矛盾造成了对于复杂图像修复效果差强人意，所以有必要加入交互，让计算机按照使人类满意的方向进行运算和处理。

## 3.2 基于样例的修复算法的改进

根据对贪心法图像修复缺点的分析，我们首先利用图像的统计特性大幅缩短运行时间，并通过额外提供向量保证修复质量，最后，使用本文提出的用户交互产生分区的方法，保证在待修复图像极其复杂时也可以有良好的修复效果，经实验结果显示，使用分区方法可以成功修复很多现有修复工具无法良好修复的图像。

### 3.2.1 利用图像统计特性

首先要解决的问题是图像修复算法效率问题，如前所述如果每次都遍历整张图像寻找源块的话计算次数过多，修复时间使人无法忍耐，如果只是盲目随机寻找源块则有极大可能遗漏最佳源块。有很多学者对此进行了研究，Barnes[12]等人提出了一种新的块匹配搜索策略，可以在很短的时间内搜索到近似最佳的源块。He[13]等人发现图像具体稀疏的块匹配统计特性，即由整张图统计出的最佳源块和目标块之间的块偏移（向量差）集中分布在少数向量上。结合这些方法和特性，我们采用如下方法进行对源块的寻找：

1. 利用 Barnes 等人的块匹配搜索策略在源区域内对所有源块进行寻找最佳匹配源块。在这一步中，我们遍历图像中的所有在源区域（未损毁区域）的块，对每一个块，寻找同在源区域中的最相似的块，寻找完之后便存入最相似块的坐标。这一步骤的目的是便于下一步根据统计块和其最相似的块的向量差得出每幅图像特有的得到高频的向量差，即图像的统计特性，从而利用高频向量差快速修复图像。

寻找最佳匹配块结束之后，每一个像素点中都会存有一个向量，假设像素点  $x, y$  坐标为  $(34, 54)$ ，存的向量为  $(40, 80)$ ，那么就表明以  $(40, 80)$  为中心的块是以  $(34, 54)$  为中心的块的最佳相似块，向量差即为  $(6, 26)$ 。

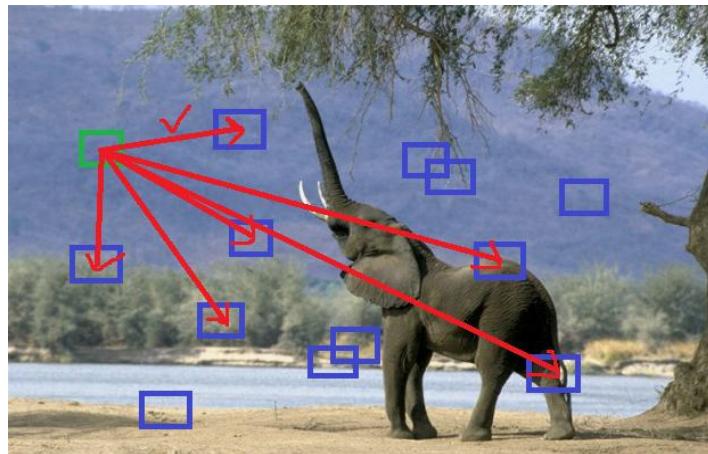


图 3.1 寻找最佳匹配的块

为源区域的所有块寻找最佳块匹配的搜索算法如下：

- ① 遍历图像中所有源区域中的块，为每个块随机分配一个向量，先假设这个向量指向了最佳相似块。
- ② 再次从下至上，从右至左遍历整个源区域中的所有块，假设遍历到以  $(x, y)$  为中心的块，其中目前储存的向量为  $(u, v)$ ，即表明目前最佳匹配块为以  $(u, v)$  为中心的块，并假设以  $(x+1, y)$  为中心的块储存的向量为  $(e, t)$ ，则比较以  $(u, v)$  为中心的块与以  $(e-1, t)$  为中心的块哪个与以  $(x, y)$  为中心的块更相近，存入更相近的向量替代原有向量。  
再假设以  $(x, y+1)$  为中心的块储存的向量为  $(e2, t2)$ ，则比较以  $(u, v)$  为中心的块与以  $(e2, t2-1)$  为中心的块哪个与以  $(x, y)$  为中心的块更相近，存入更相近的向量替代原有向量。
- ③ 再次遍历整张图，假设遍历到以  $(x, y)$  为中心的块，其中目前储存的向量为  $(u, v)$ ，则在以  $(u, v)$  为中心的矩形内再次随机选取一个坐标，假设选到了坐标  $(u2, v2)$ ，则比较以  $(u, v)$  为中心的块与以  $(u2, v2)$  为中心的块哪一个与以  $(x, y)$  为中心的块更相近，将更相近的块的坐标  $(u3, v3)$  保存下来 ( $(u3, v3) = (u, v)$  或  $(u2, v2)$ )。然后再以  $(u3, v3)$  为中心的矩形内再次随机选取一个坐标，重复步骤，每次矩形会比上一次缩小一定比例，直至矩形缩小为一个像素点为止。
- ④ 到此一次迭代完成，重复进行步骤②③，直至达到程序设定迭代次数为止，每次奇数遍历时从下至上，从右至左遍历；偶数遍历时，从上至下，从左至右遍历，并且将步骤②中的“+1”改为“-1”，“-1”改为“+1”。

搜索算法结束后，就会得到统计图，每一个像素点内都记录了一个坐标。在实际实验中，使用图像对坐标进行存储，因为图像中每个像素点都有 RGB 三个数值，便利用这个数值表示横纵坐标，由于每个 RGB 能表示 0 至 255，所以，充分利用便能表示尺寸在  $4352 \times 3840$  内的图像。具体算法如下：假设 RGB 内存入的数据为  $r, g1 \times 17 + g2, b$ ，其中  $g2 < 17$ ，则表示的坐标为  $(r+g1 \times 256, b+g2 \times 256)$ ，所以表示的最大尺寸为  $(256 \times 17) \times (256 \times 15) = 4352 \times 3840$ ，其中 15 是因为  $15 \times 17 = 256$ 。这个尺寸已经明显超出一般图像的尺寸大小，足够使用。

根据实验结果，这种块匹配搜索策略效果非常好，找到的块很相似。图 3.2 是为了测试实际效果而做的直观实验。



图 3.2(a) 电影中片段截图



图 3.2(b) 电影中另一片段截图



图 3.2(c) 根据块匹配结果合成的图

图 3.2 (a) (b) 为原图, 运行块匹配搜索时, 遍历 a 图中的所有块, 对于每一个块, 都要在 b 图中寻找最相似的块, 便于理解, 假设 a 图中有一个像素点坐标为  $(ax, ay)$ , 在 b 图中寻找的最相似的块为  $(bx, by)$ , 则将块  $(bx, by)$  中的像素点的 RGB 像素值放入  $(ax, ay)$  中, 最后得到图 c。用于存储向量值信息的图如下:

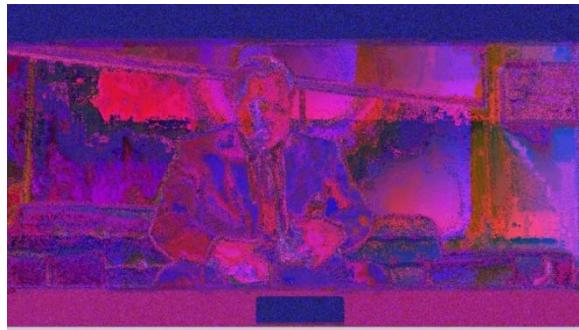


图 3.2(d) 记录块匹配结果的图

(下方蓝色是盖住字幕导致的效果, 原先 a 图有字幕, b 图无字幕, 人为盖住 a 图字幕)

也就是说, c 图完全是由 b 图和 d 图得到的, c 图中的所有像素点全部来自 b 图, c 图与 a 图的相似性表明块匹配效果非常好。

**2. 利用图像的稀疏统计特性得到高频块偏移向量。**He[13]等人发现图像的块偏移具有稀疏特性, 更简单的说, 每个块与寻找到的最相似的块的坐标差, 也就是块偏移向量集中在特定向量上。一整张图中可能会有某些块偏移向量出现几百次, 而大部分向量一次都没有出现。鉴于图像的这种特征, 我们可以预先统计出频次较高的向量, 舍弃频次很低或者为零的向量, 在修复图像时, 我们只根据频次较高的向量寻找最佳源块。具体来说, 我们已经得到高频向量  $(u_i, v_i)$ ,  $i=1, 2, 3\cdots$ , 则在寻找以  $(x, y)$  为中心的块的最佳匹配块时, 我们只寻找以  $(x+u_i, y+v_i)$  为中心的源块, 并将源块中的信息复制到目标块中缺损部位。由于频次很高的向量个数很少, 所以使用这种方法极大地节省了时间, 同时也保证修复质量不会下降。假设图像大小为  $500 \times 500$ , 如果我们只利用频次前 1000 的向量进行搜索, 则寻找匹配所用时间为原先的  $1000/(500 \times 500)=0.4\%$ .

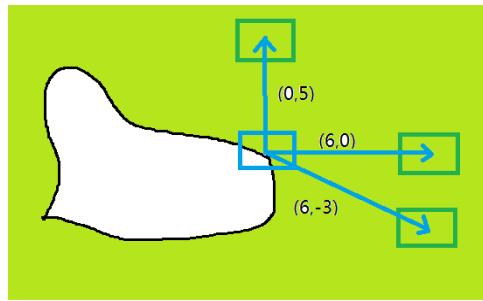


图 3.3 假设已统计出优势块偏移是  $(0, 5)$ ,  $(6, 0)$ ,  $(6, -3)$ , 则寻找如图所示的块

另外要注意的是实际修复过程中是在同一张图中预先寻找最佳匹配块，显然如果需要寻找以  $(x, y)$  为中心的块的最相似块，往往最像的则是将块上下左右移动一个像素得到的新的块，然而这种模很低（为 1）的块偏移在实际修复过程中是没有用的，会导致纹理周期性重复以及自己复制自己的问题，所以在实验中人为设置了向量的最短长度避免这种情况发生。

### 3.2.2 交互产生额外向量

利用图像的统计特性和高向量进行图像修复极大地提高了修复速度，在大部分情况下具有良好的修复质量。但是，为了防止高向量并不是能够有效修复图像的向量这种意外情况发生，本文算法提供了产生额外向量的功能。

Sun[5]等人提出了在图像修复中利用交互提供结构线的方法，受到启发，本文采用在修复前用户画线的方式提供额外的块偏移向量，并入之前由统计得到的块偏移向量中，形成新的向量集合，在修复时利用新的向量集合寻找最相似源块。

通过这种方法，用户可以按照自己的想法提供信息，保证了修复时寻找的源块包含了用户希望寻找的块，用户对程序有更大的控制性，也使得修复质量得到提高。

在本文算法中，提供了两种方式进行交互画线。

#### 1. 画直线方式

用户可以直接在图像上画直线，最多可以画 100 条，画完之后，便会将直线采样，从最短距离开始，根据设定的采样率依次取更长的向量。假设用户在图像上画了水平线，系统预先设置的最短向量长度为 20，采样率预先设为 5，那么就会产生的向量为  $(20, 0)$ ,  $(-20, 0)$ ,  $(25, 0)$ ,  $(-25, 0)$ ,  $(30, 0)$ ,  $(-30, 0)$ ,  $(35, 0)$ ,  $(-35, 0)$ ,  $(40, 0)$ ,  $(-40, 0)$  …直到向量长度达到图像对角线长度为止。

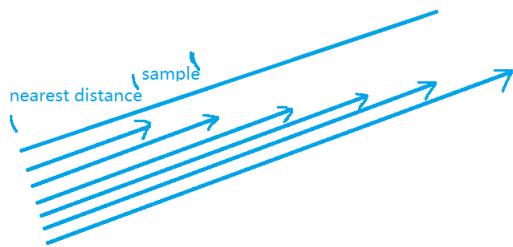


图 3.4 用户画直线后产生的向量

如图 3.4, 如果画出上方的蓝线, 则会自动产生下方的向量以及与下方向量方向完全相反模相同的向量。

实际操作图如下:

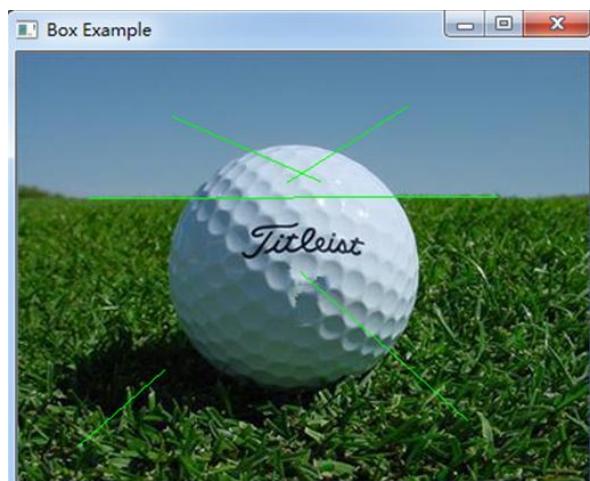


图 3.5 用户画线实际效果图

由于是在图上直接画, 用户对于直线的角度把握会更加准确。

## 2. 画曲线方式

另一种交互画线的方式就是通过画曲线来提供, 用户还可以在图像上拖动鼠标画各种形状的线, 来产生更多的块偏移向量。

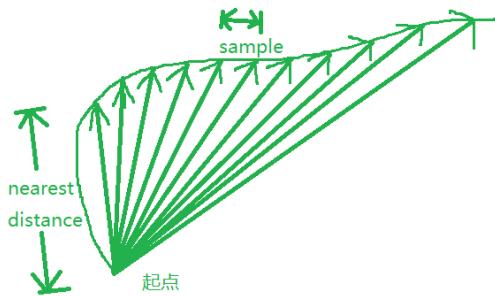


图 3.6 用户画曲线后产生的向量

如图 3.6，在图上画出曲线后，便会自动产生向量。

实际操作图如下：

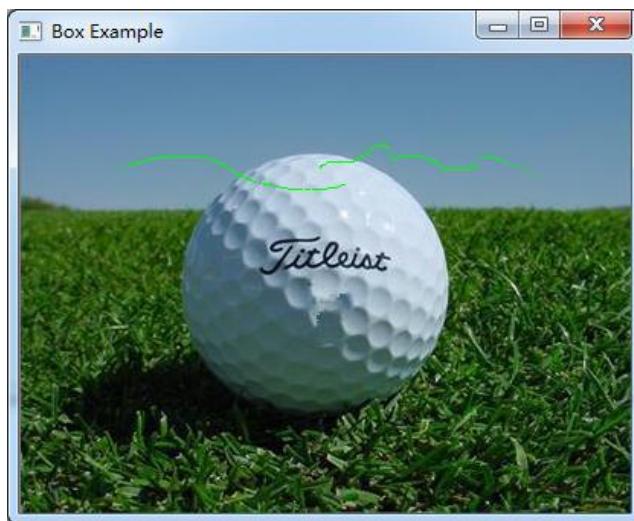


图 3.7 用户画曲线实际效果图

值得注意的是，利用交互产生额外的块偏移向量是一种辅助手段，在用户不提供交互的情况下依然有可能较完美的修复图像。当修复效果不佳时，交互向量提供了一种优化的方式。

然而在某些情况下，交互向量是很有作用的。虽然在利用图像的统计特性得到高频块偏移向量中期望得到模较长的向量，并且为了防止得到过短的向量设置了最短长度限制，但是仍然有可能导致大部分统计得到的向量模长刚好大于最短长度限制，这就会导致修复到图像缺损区域中心时缺乏有效的长的向量。例如图 3.7，我们想要把球当成破损区域，将球移除掉。假设设置的最短向量长度限制为 20，而球的半径为 50，则利用统计得到的向量很有可能最长长度小于 50，则在修复球心部位时，无法寻找到球外区域，只能寻找到原先是破损区域，现已修好的区域的块，如果发生这种情况则易导致周期性的纹理。

在实验早期，确实因为没有考虑到这种情况而导致周期性的纹理，所以在后续实验中尽量不寻找那些原来是破损区域而现在已修复好的区域提供的块。早期产生的周期纹理如下：



图 3.8 修复产生的周期性纹理



图 3.9 修复产生的周期性纹理

所以加了交互向量之后，这种问题就得到避免，在实验中后期未发生过这种情况。



图 3.10(a) 原图



图 3.10(b) 标明要修复区域的 mask 图

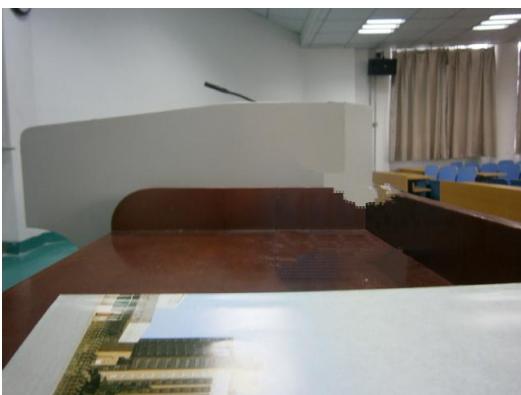


图 3.10(c) 不使用交互产生额外向量保留其他改进算法时的结果

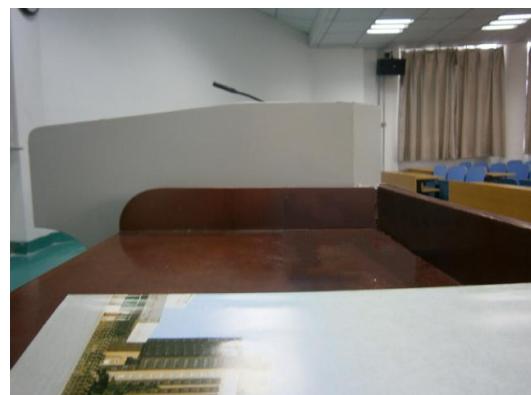


图 3.10(d) 使用交互产生额外向量和其他改进算法时的结果

### 3.2.3 对图像进行分区

对图像进行分区是本文创新的一种修复优化方法。通过观察，可以发现大部分图像的待修复区域都可以根据结构划分为几个部分，每个部分之间有很大的差异性。在进行修复算法思考时，发现很多前人失败的例子都是由于一个区域的块被填入了其他块中，具体实例如下：

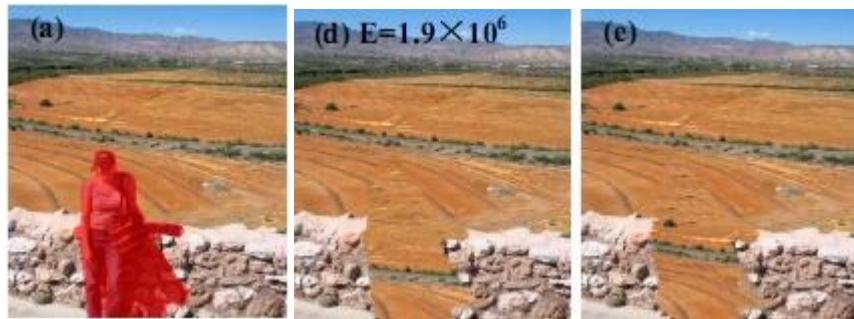


图 3.11(图像数据来自[6])把人修复掉时将较远的土地区域填入了本应是石墙区域，修复失败。

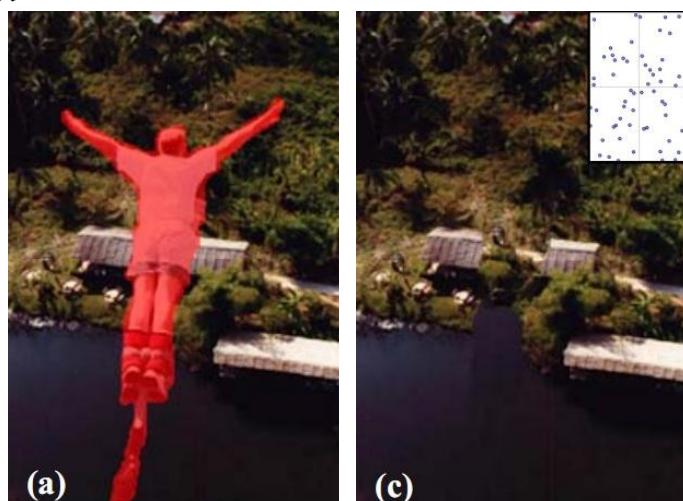


图 3.12(图像数据来自[6])将人修复掉时将水修复到了岸边，导致失败的结果。

通过观察，大部分图像的待修复区域通常都包含了许多完全不同的区域，这些区域有明显的不同，所以本文的想法是可以通过将图像人为分成几个不同的部分，在修复时使计算机趋向于在同一区域内寻找源块，提高修复质量。具体如何划分不同部分的例子如下



图 3.13 主要包含石块、浪、海



图 3.14 主要包含石墙和土地两个部分



图 3.15 主要包含瀑布、草地、巨石



图 3.16 主要包含河岸沙滩、河流、树丛、远处背景



图 3.17 主要包含窗框、窗台、玻璃

经过实验，通过分区得到的修复效果明显好于不使用分区的效果。由于分区加入了人类的智慧，有效地指引了计算机修复的过程，其效果远超过任何智能的修复算法，并且分区并不需要花费很多时间，只需要人为勾勒出几个区域即可，其复杂度与人为勾勒出缺损区域相当，而画出缺损区域是修复必须的步骤，所以分区方法简单有效。

具体分区算法步骤是，首先用户通过交互指定某个区域为区域 1，然后再依次指定区域 2，区域 3，区域 4…最多可以指定到区域 9，通常 9 个区域已经足够使用，况且再多的话也会显得操作复杂。区域数越低优先级越高，修复时会首先修复区域数低的，而且一旦区域  $m$  指定某个像素为区域  $m$ ，则即使区域  $n$  ( $0 < m < n$ ) 再次指定那个坐标为区域  $n$ ，也是无效的。这样的好处是，我们只需要认真的沿着区域边缘画一次即可，节省了精力。最后没有被任何区域指定的地方作为区域 0，最后修复。

接下来举一个具体例子展示分区的算法。

首先是一张待修复的原图以及表明待修复部位的 mask 图，我们想要把球从图像上去除。



图 3.18(a) 原图



图 3.18(b) mask 图

在进行修复前用户可以将原图分为几个区域。



图 3.18(c) 红线内即为区域 1，然后再依次指定区域 2 和区域 3

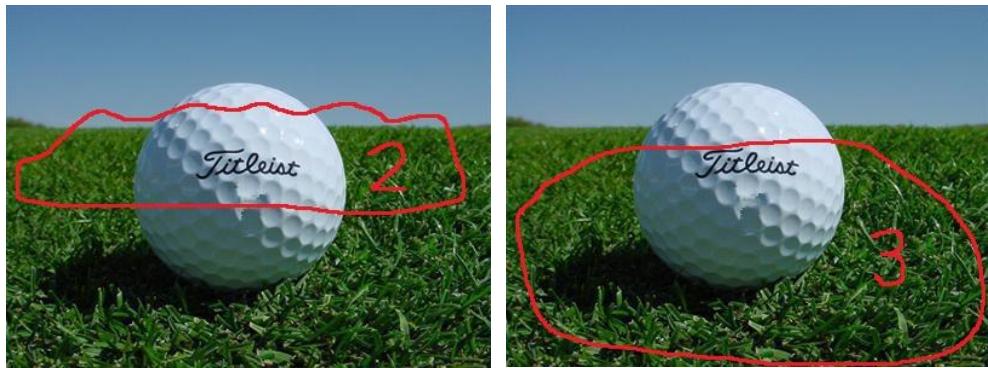


图 3.18(d) 画出区域 2

图 3.18(e) 画出区域 3

可以看到再画区域 2 的上方时可以不太精细甚至可以画入一部分天空而不影响最后的修复，这是因为区域 1 已经将天空包含进去了，即使区域 2 再次画入也没有任何坏的影响。最后的区域分割效果大致如下：

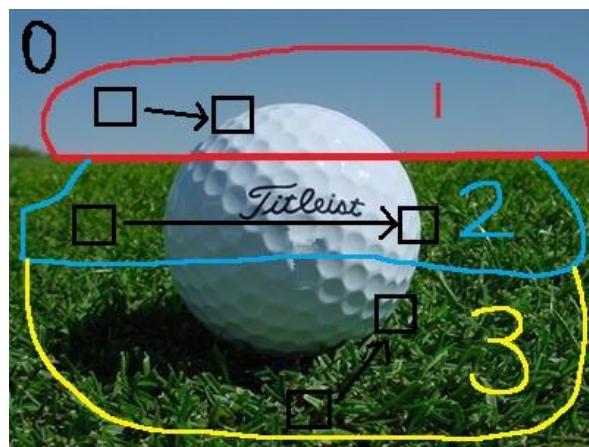


图 3.18(f) 区域分割效果示意图

图中的数字表示被分割的不同区域，修复时首先修复区域 1，将区域 1 中的缺损区域全部修复完毕后再修复区域 2，区域 3，区域 0 中不含缺损区域，所以不需要修复。修复每个区域时，优先寻找在同一区域的块，具体的方法是，如果寻找到不在同一区域的源块，在计算块的相似度的时候会增加惩罚，直接增大计算出来的块的距离，以保证尽可能寻找同区域的源块。

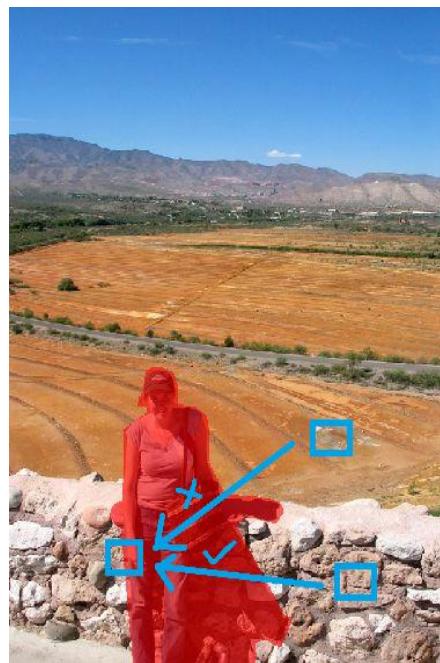


图 3.19 分区有助于找到更好的块

图 3.19 是一个分区优点的展示，在修复左边的块时（块内左侧是源区域，右侧是待修复区域），如果不进行分区，则可能会寻找到右上方的块（右上方的块的左侧与之很相似），那么就会导致土的颜色被错误地复制到石墙上。但是如果采用分区方法，将石墙圈成一个区域，则不会寻找到右上方区域而很有可能寻找到右下方的块。

分区的另一个优势是即使缺乏最佳源块的图也可以很好地修复。如图 3.20 所示，



图 3.20(a) 原图



图 3.20(b) 修复后的图

图 3.20(a)是待修复的图像，图 3.20(b)是利用分区算法修复出的图像（又额外加上了块的标记）

可以看到，当我们修复中间的块时，缺乏有效的相似源块。中间的块左侧应是横放的浅色木头，右侧应是竖立的木头，而在图中是没有这样的源块的。例如上方的源块左侧是远处背景，下方的源块左侧颜色过深。而如果使用分区方法，可以将竖立的树木当成一个区域，横着的树木当成另一个区域，则得到的修复效果如上右图。

现加入分区功能，重新运行由其他程序得到的失败的结果，最终结果如下：



图 3.21(a) 原图



图 3.21(b) 使用本文算法的修复结果



图 3.22 (a) 原图



图 3.22 (b) 使用本文算法的修复结果

对于如图 3.23 所示的结构线较为复杂的图像来说，缺乏智能的计算机基本无法达到满意的修复效果，然而结合了人类智慧（通过分区来加入）和计算机高速性能后，就可以较为完美地修复图像了。



图 3.23(a) 原图

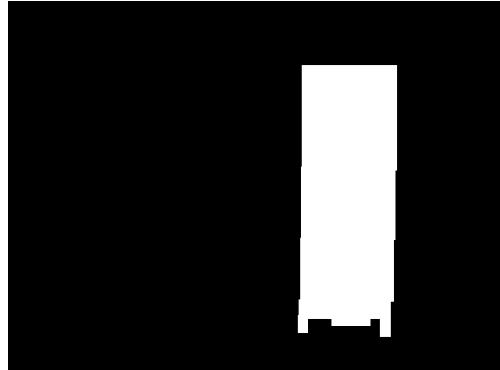


图 3.23(b) 标明待修复区域的 Mask 图



图 3.23(c) Photoshop 自动修复结果



图 3.23(d) 不使用分区功能时程序得到的结果



图 3.23(e) 使用分区后程序得到的结果



图 3.24(a) 原图



图 3.24(b) mask 图



图 3.24(c) 不使用分区的结果图

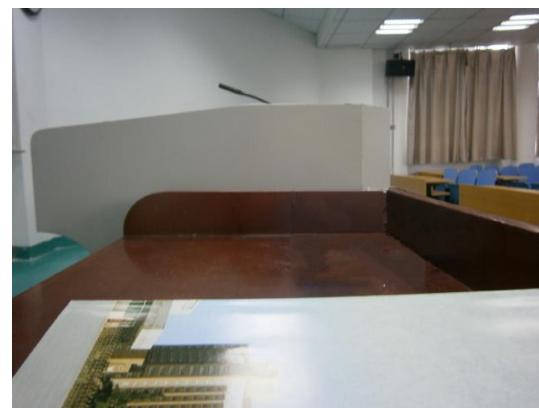


图 3.24(d) 使用分区后的结果图

### 3.3 算法实现

#### 3.3.1 算法描述

在本小节中,将对算法进行整体性的描述。输入一张原图后,首先是画出 mask 图像标出哪些地方需要被修复,然后是用户画出不同的区域将图像分成若干部分,产生一张表示分区的图。将这三张图像读入后,用户可以在原图上通过画线方式提供块匹配向量,之后程序自动根据原图得到优势的块偏移向量,然后根据优化后的基于样例的图像修复算法依次修复区域 1, 区域 2, 区域 3……直至修复完成。

### 3.3.2 算法流程图

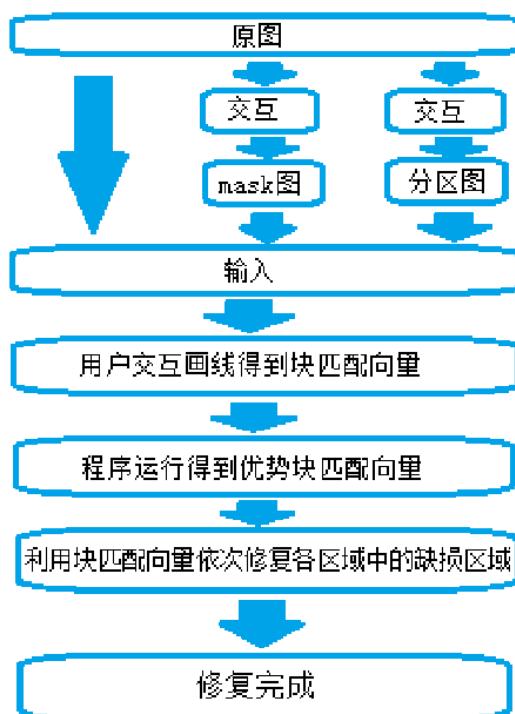


图 3.25 流程图

### 3.4 实验结果

在本小节将展示通过本文算法实现的程序所得到的部分修复结果。



图 3.26 (a) 原图



图 3.26 (b) 修复后图像



图 3.27(a) 原图



图 3.27(b) 修复后图像

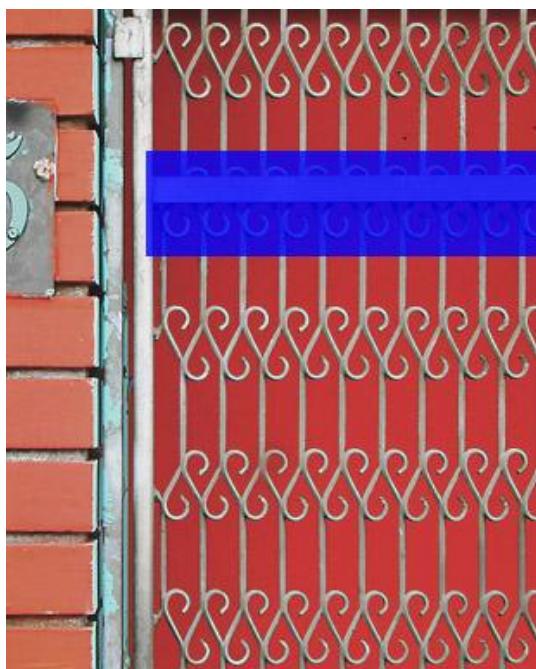


图 3.28(a) 原图

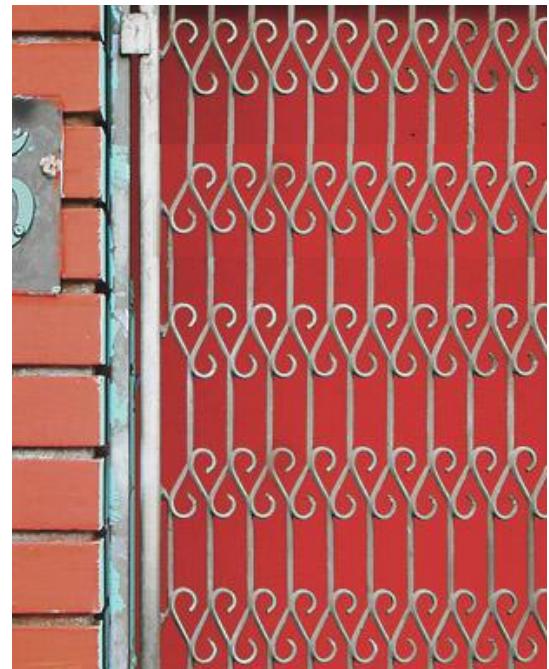


图 3.28(b) 修复后图像



图 3.29(a) 原图

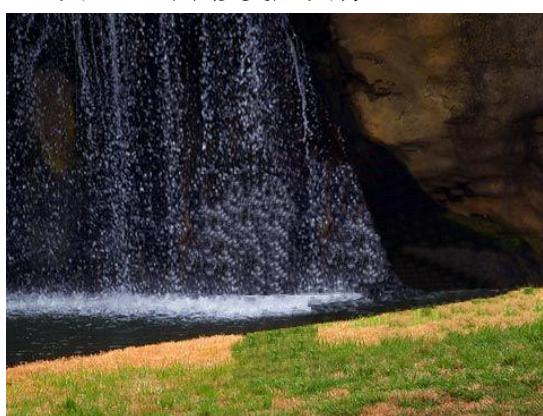


图 3.29(b) 修复后图像



图 3.30(a) 原图

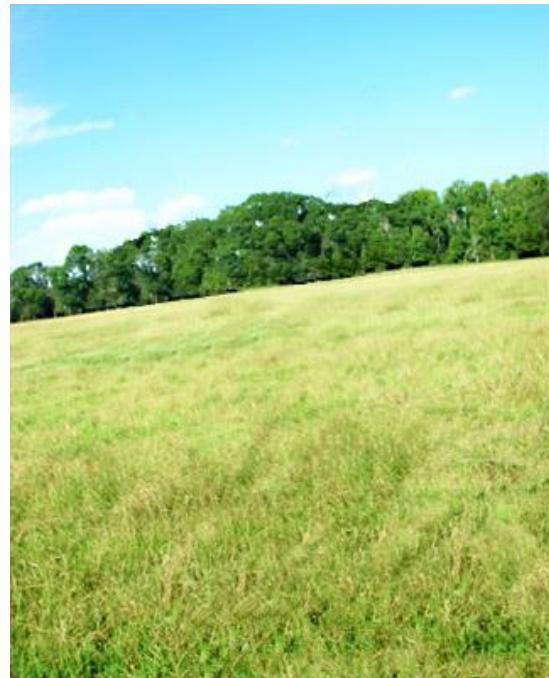


图 3.30(b) 修复后图像

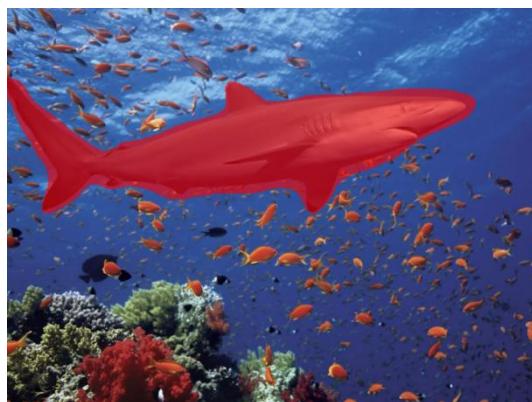


图 3.31(a) 原图



图 3.31(b) 修复后图像



图 3.32(a) 原图



图 3.32(b) mask 图



图 3.32(c) 修复结果再次作为原图



图 3.32(d) mask 图



图 3.32(e) 最终修复后的图像



图 3.33(a) 原图



图 3.33(b) mask 图



图 3.33(c) 不使用分区得到的结果图



图 3.33(d) 使用分区算法得到的结果图



图 3.34(a) 原图



图 3.34(b) 修复后图像



图 3.35(a) 原图



图 3.35(b) 修复后图像



图 3.36(a) 原图



图 3.36(b) 修复后图像



图 3.37(a) 原图



图 3.37(b) 修复后图像



图 3.38(a) 原图



图 3.38(b) 修复后图像



图 3.39(a) 原图



图 3.39(b) 修复后图像



图 3.40(a) 原图



图 3.40(b) 修复后图像

## 第 4 章 实验界面设计

为了对算法效果进行验证，并通过实际结果比较各种算法的优劣，特搭建实验平台进行实验。平台可以实现本文中提到的各种算法，还可以分别关闭交互画线以及分区算法，来比较算法的优势。同时，本平台还可以在修复前帮助制作 mask 图和分区图。

打开时程序界面如下：

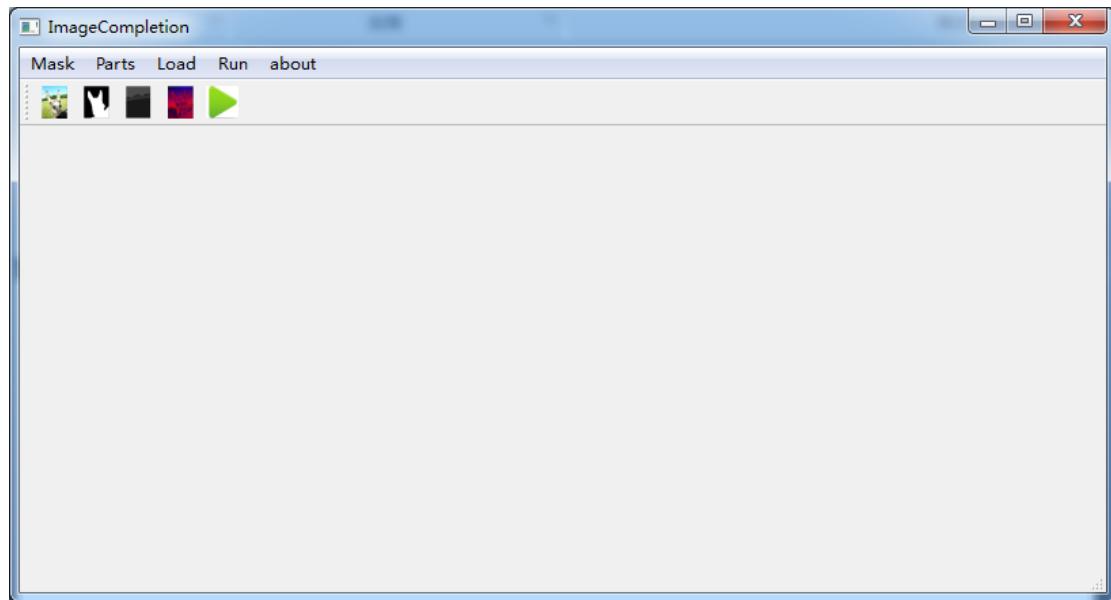


图 4.1 程序界面

接下来将以修复下方照片为例，根据顺序依次介绍各个功能。假设我们的目的是将照片中的球修复掉。



图 4.2 将要修复的图像

正式修复时需要读入原图、mask 图、分区图即可进行修复，在此前可以利用原图和 mask、parts 菜单得到 mask 图和分区图。

首先我们用 mask 菜单得到 mask 图，用来标识出待修复的部分，接着通过 parts 菜单将原图分成不同区域，并用一张图表示出来，以提高修复质量。可以看出，原图可以分为三个不同的纹理部分，上方是天空，中间是远处纹理较密的草地景色，下方是近处景色，运行完 parts 菜单后会得到一张表明各像素属于哪个区域的标识图。前期工作完成后，通过 Load 菜单将原图、mask 图、分区图读入，即可利用 Run 菜单运行修复，得到修复好的图像。

#### 4.1 制作 mask 图

首先可以看到界面中有个 mask 菜单，这个菜单的作用是用来由原图制作出表明待修复区域的 mask 图。

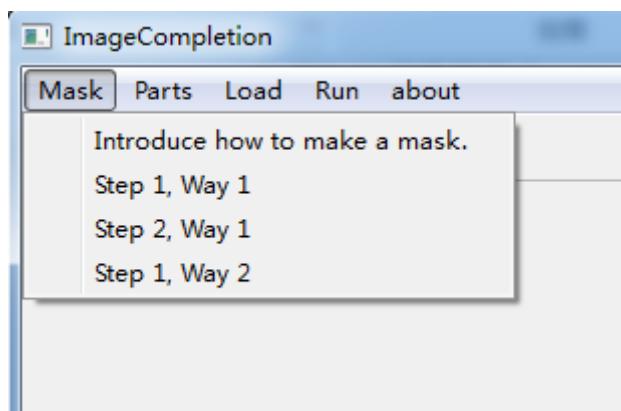


图 4.3 Mask 菜单，用来产生 mask 图

第一个“Introduce how to make a mask”是介绍 mask 的功能以及如何使用 mask 菜单下各个操作。在程序中一共提供了两种得到 mask 的方式，分为方式 1 和方式 2，方式 1 需要两步完成，方式 2 只需一步即可。方式 1 虽然较为繁琐并且需要其他画图工具的辅助，但是可以非常精细地画出 mask，方式 2 则是直接在本程序中勾勒出 mask，较为方便，但画出的 mask 比较粗糙，无法做精细修改。两种方式最终都可以得到一张 mask 图。

首先使用方式 1 画出 mask 图，点击“Step1, Way1”，就会出现对话框

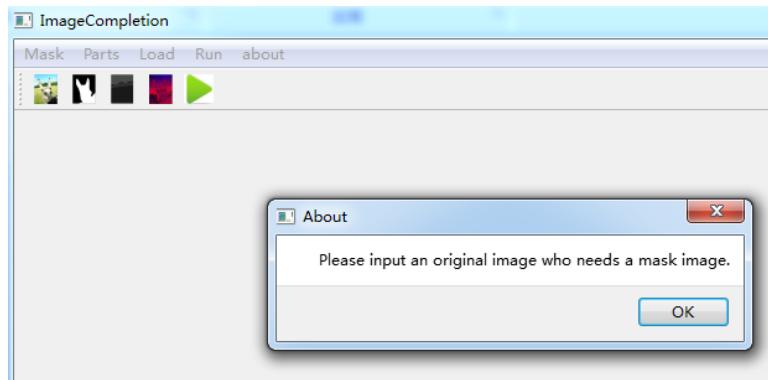


图 4.4 读入原图以制作 mask 图

点击“OK”后，便可以选择想要打开的原图

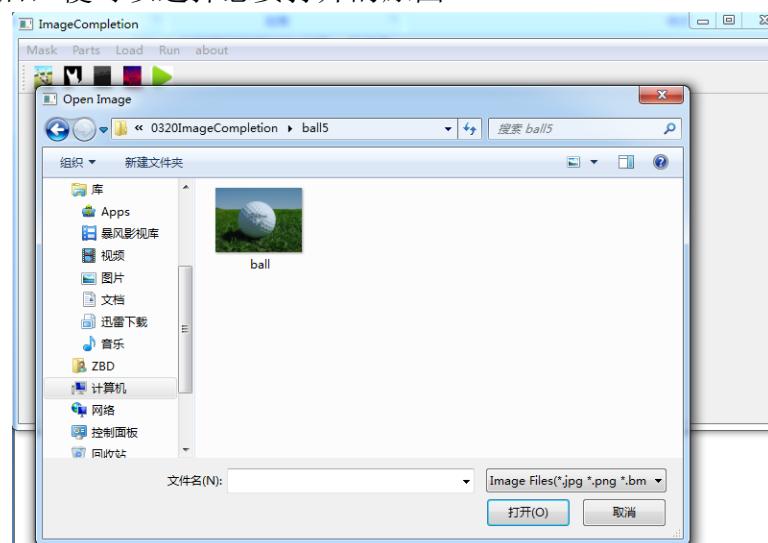


图 4.5 读入图像

成功打开后就会出现如下对话框

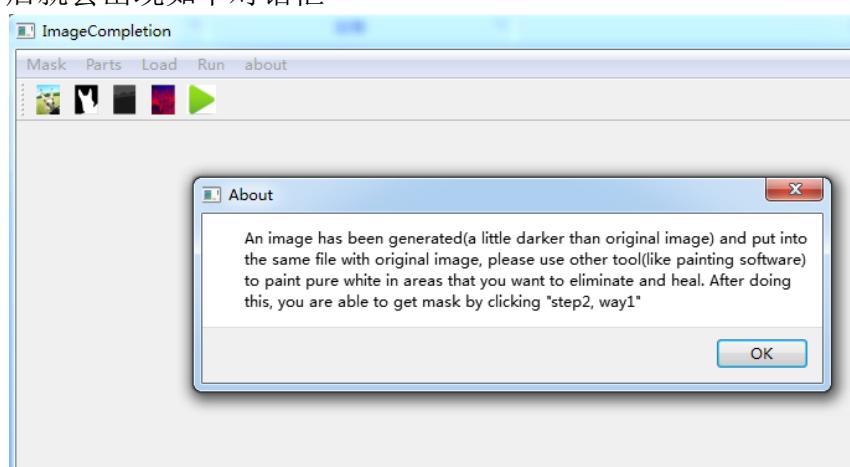


图 4.6 制作 mask 图的第一步完成

这时就会在原图同一文件夹下出现一张较暗的图像，图像内容与原图一致。  
产生的图如下图 4.7



图 4.7 制作 mask 图第一步完成后得到的图

然后用户就可以用普通画图工具在此图上填涂纯白色，被涂上纯白色的区域将会作为被修复区域。把原图变暗的原因就是为了防止原图中含有纯白色，与用户涂上的纯白色混淆。填涂后的效果如下

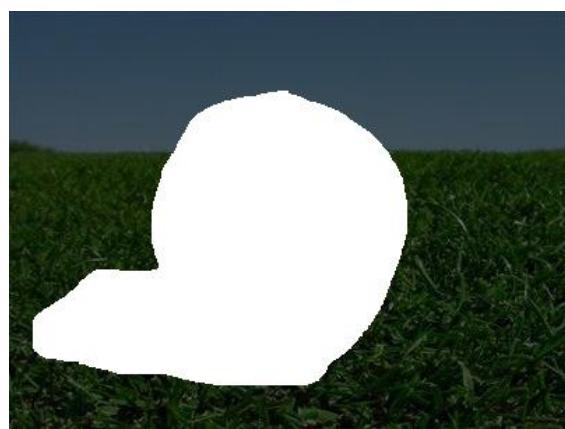


图 4.8 进行第二步前将第一步产生的图欲修复区域涂白

填涂结束后将图像保存，之后便进行第二步，点击“Step2, Way1”，运行这一步是将纯白色区域识别出来，得到 mask 图。

点击“Step2, Way1”后会要求读入一张图，便将填涂后的图读入。

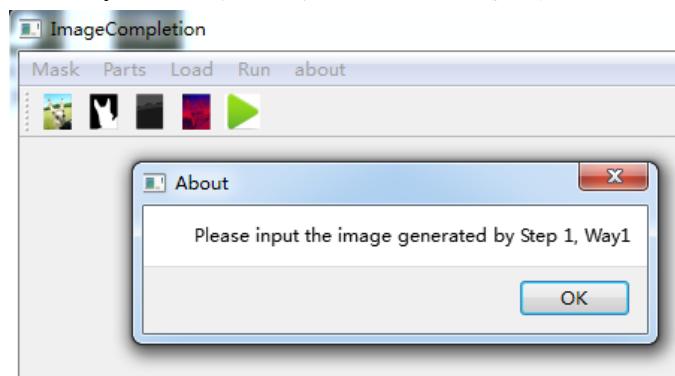


图 4.9 读入图像绘制的图像产生 mask 图

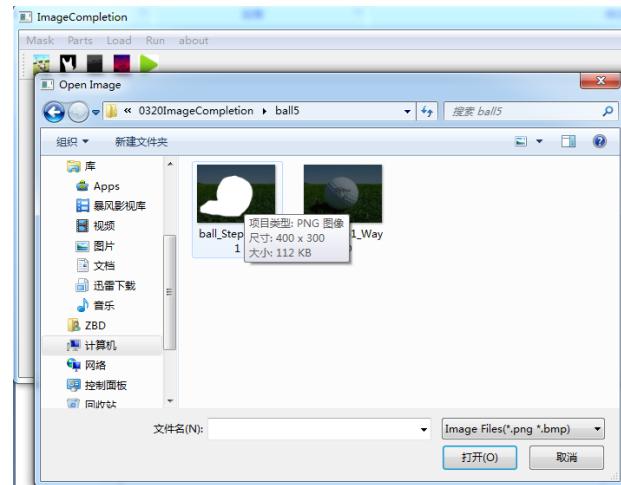


图 4.10 读入图像

读入之后便会提示 mask 图已经产生，并且被保存在同一文件夹下。

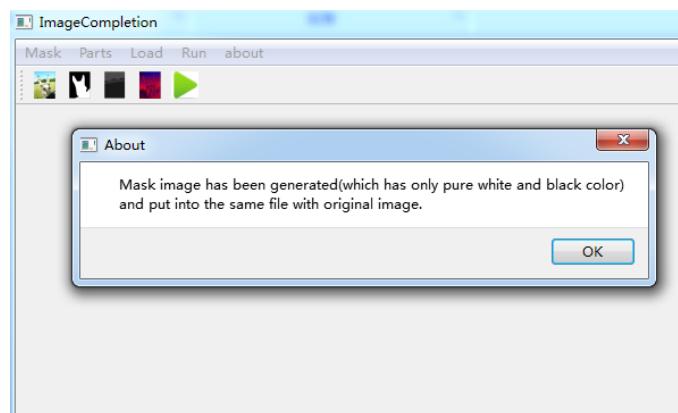


图 4.11 mask 图已经生成

打开文件夹，便可以发现 mask 图（纯黑白图）已经被保存。



图 4.12 mask 图

这样便成功得到了 mask 图。

也可以使用简单的方式 2 得到 mask 图。点击 mask 菜单下的“Step1, Way2”，得到

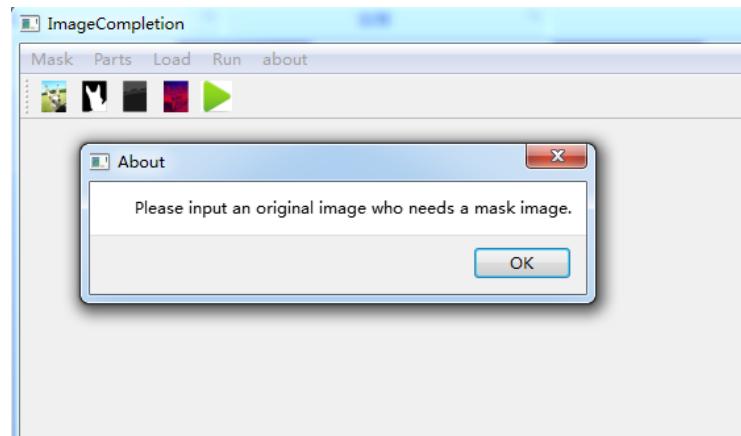


图 4.13 利用简单的方法一步得到 mask

点击“OK”后输入原图，就会弹出图像，可以在图像上圈出 mask 区域，完毕后按“Esc”键即可观察画出的 mask 图像，再次按“Esc”即可使图像消失，mask 已存入。

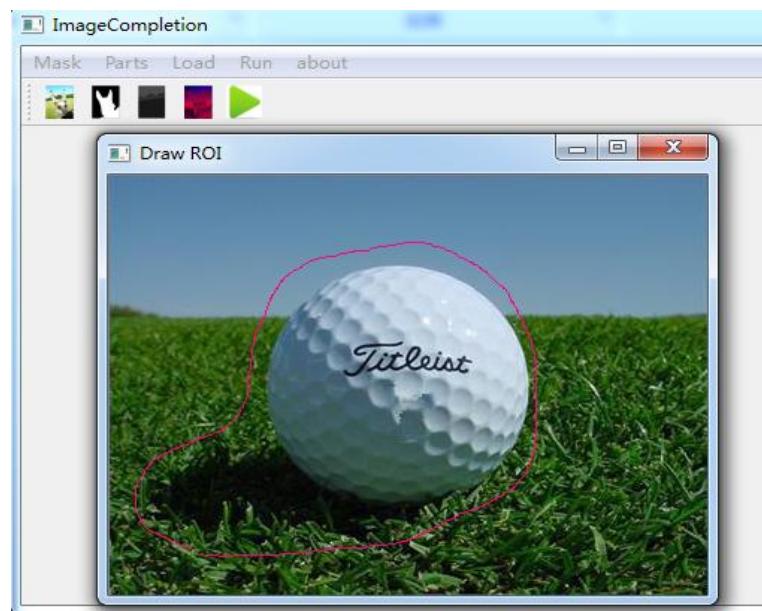


图 4.14 通过画线来圈出要修复区域

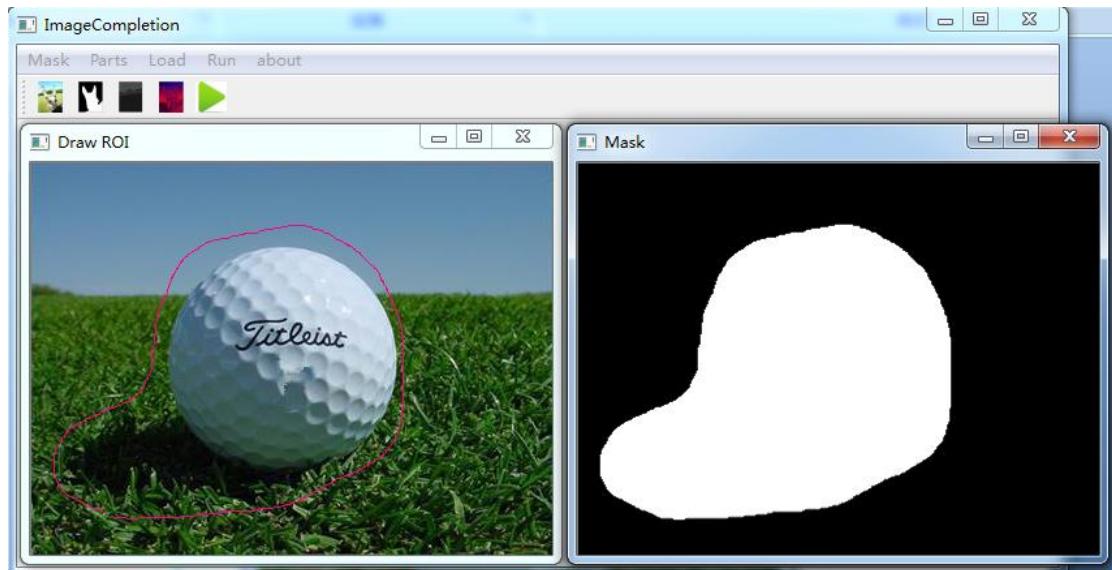


图 4.15 得到 mask 图

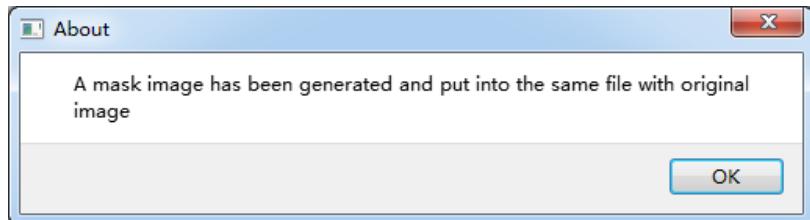


图 4.16 mask 图已经得到并保存

## 4.2 制作分区图

在菜单栏中有“Parts”菜单，它的作用就是制作分区图，用来标注个像素点分别属于第几个区域。

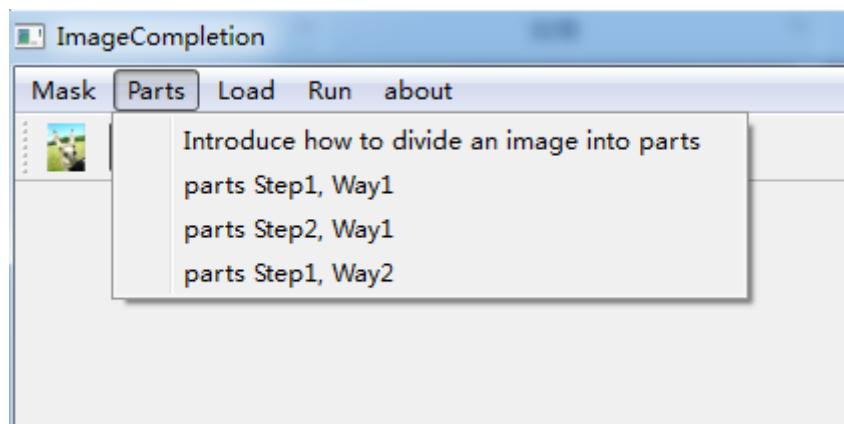


图 4.17 用来产生分区图的 parts 菜单

第一个“Introduce how to divide an image into parts”是介绍“parts”的功能以及如何使用 parts 菜单下各个操作。和制作 mask 图一样，在程序中一

共提供了两种得到分区图的方式，分为方式 1 和方式 2，方式 1 需要两步完成，方式 2 只需一步即可。方式 1 较为繁琐并且需要其他画图工具的辅助，但是可以非常精细地画出每一区域，方式 2 则是直接在本程序中勾勒出各个区域，较为方便，但画出的轮廓比较粗糙，无法做精细修改。两种方式最终都可以得到一张分区图。

首先使用方式 1 画出分区图，点击“parts Step1， Way1”，就会出现对话框

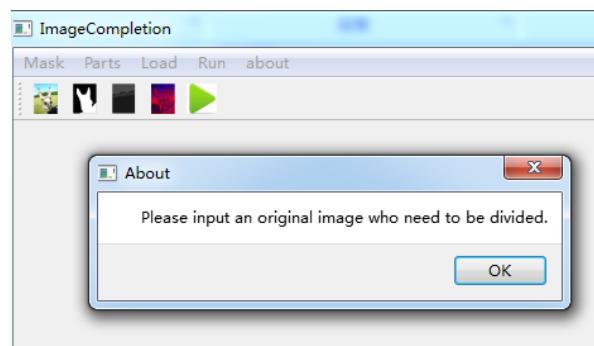


图 4.18 输入原图

这一步是要输入原图，点击“OK”。

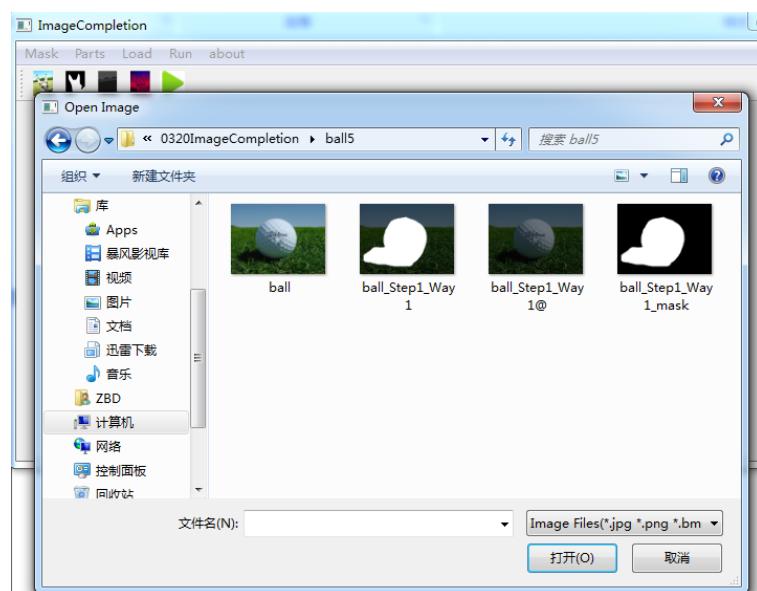


图 4.19 打开原图

打开原图后，会再次让用户打开 mask 图

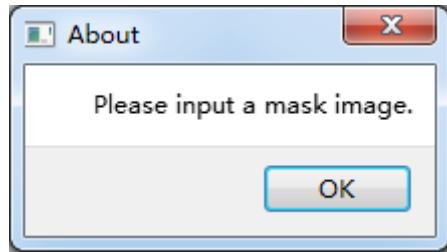


图 4.20 打开 mask 图

再次输入 mask 图之后，就会在同文件夹下得到一张新图，如图 4.21。



图 4.21 读入原图和 mask 图之后得到的图像

这张图同样由于处理过的原因没有纯白色，并且用红色区域标注出待修复区域，用户可以在此图上涂纯白色，用来表示不同区域。根据此图特点，可以将图像从上到下分为天空，远处草地，近处草地三个主要部分。由此分别填涂三个区域，得到三张图。



图 4.22 填涂出区域 1



图 4.23 填涂出区域 2



图 4.24 填涂出区域 3



图 4.25 得到的分区图

这里特意在填涂区域 3 时将区域 2 中的部分区域覆盖，就是为了展示即使覆盖优先级更高的区域也没有影响，这样在勾绘重要线条时(如草地与天空分界线)只需细心地画一次，节省精力。

将此三张图保存后，就可以点击“parts”菜单下的“parts Step2, Way1”，就可以依次读入这三张图，最后在同文件夹下会产生表明分区结构的一张图，如图 4.25。

也可以使用简单的方式 2 得到 mask 图。点击 Parts 菜单下的“Step1, Way2”，依次输入原图和 mask 图之后就可以依次圈出各个区域。第一个圈出的区域是区域 1，圈完后按“Esc”即可，然后再圈出区域 2，依次类推。每次程序都会询问是否想要再画出一个区域，如果已经完成则随时可以退出。

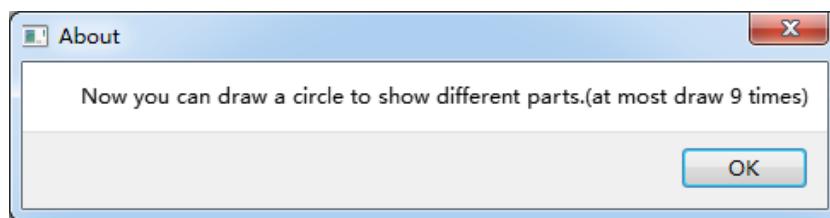


图 4.26 使用较为简单的方式 2 得到分区图

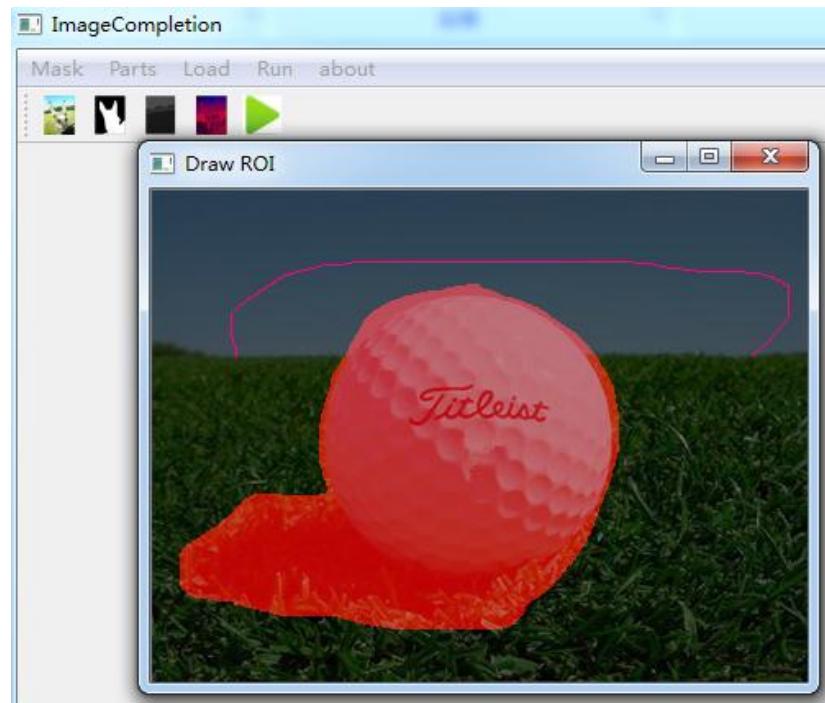


图 4.27 通过画线划分不同区域

画好区域 1 后按“Esc”即可。(线条可以不闭合,会自动连接起来)

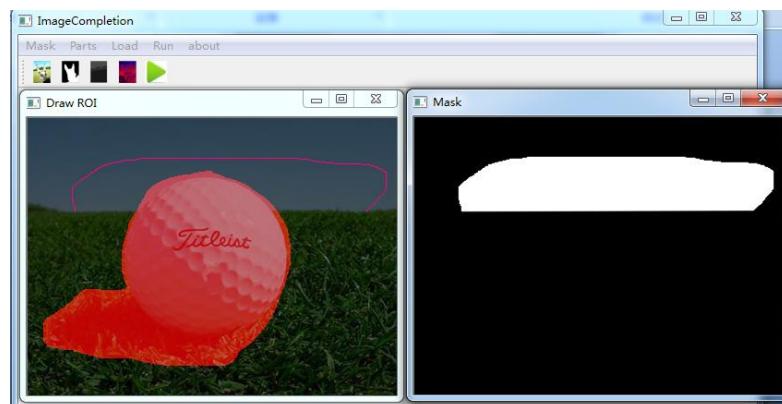


图 4.28 已成功画出区域 1

再按一次“Esc”取消显示。

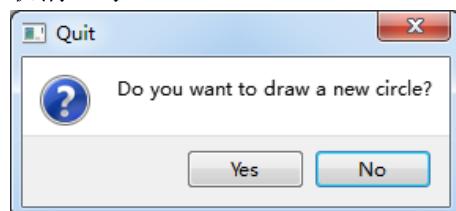


图 4.29 是否开辟另一个区域

选择“Yes”画区域 2

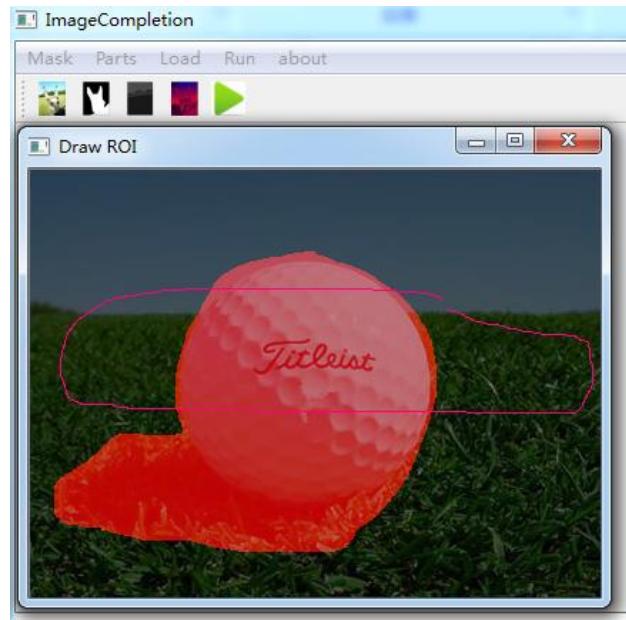


图 4.30 圈出区域 2

按“Esc”表示画完

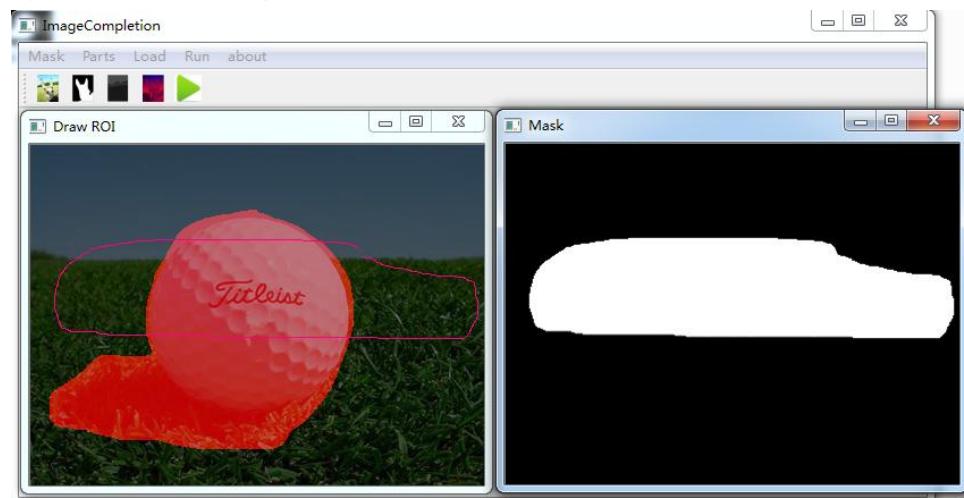


图 4.31 圈出区域 2

再按一次“Esc”取消显示。

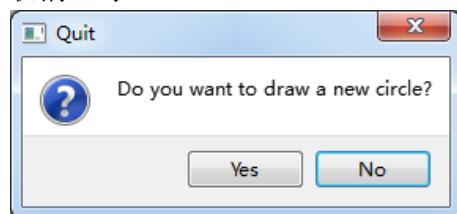


图 4.32 询问是否再次开辟一个区域

选择“Yes”

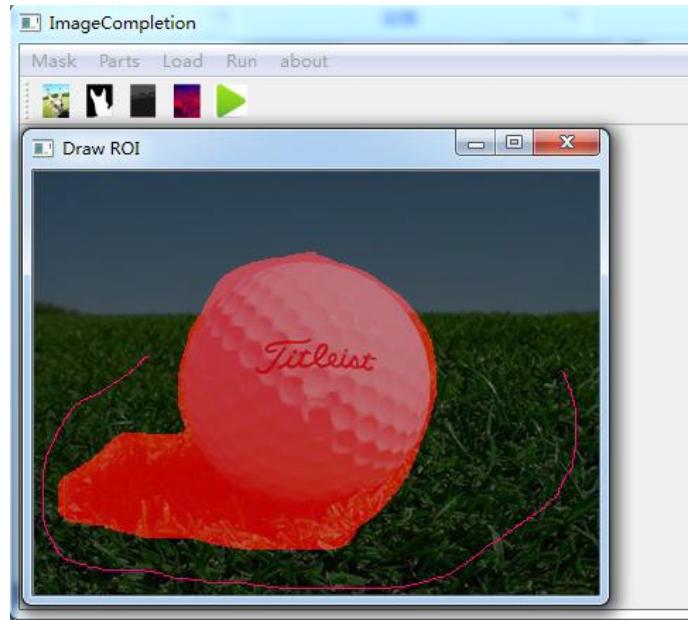


图 4.33 圈出区域 3

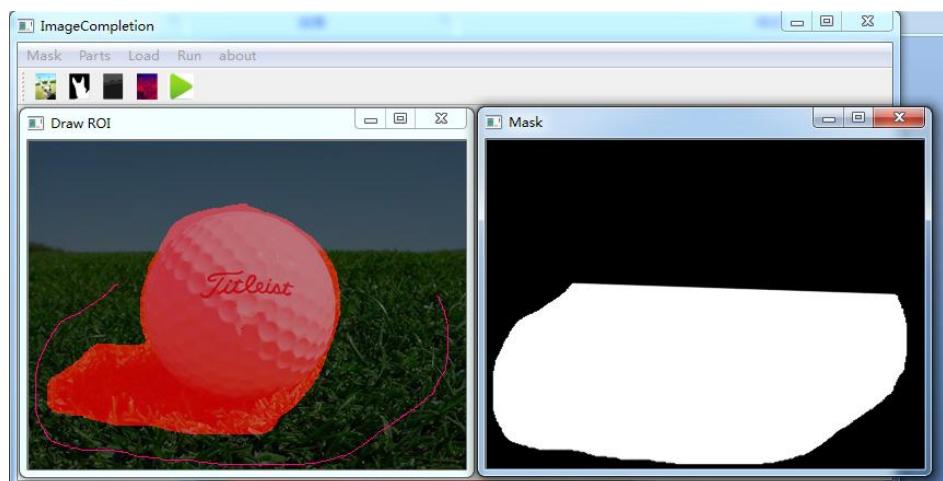


图 4.34 得到区域 3

画完后会显示出分区图的结果，并会将此图自动保存。这样分区操作便结束了。

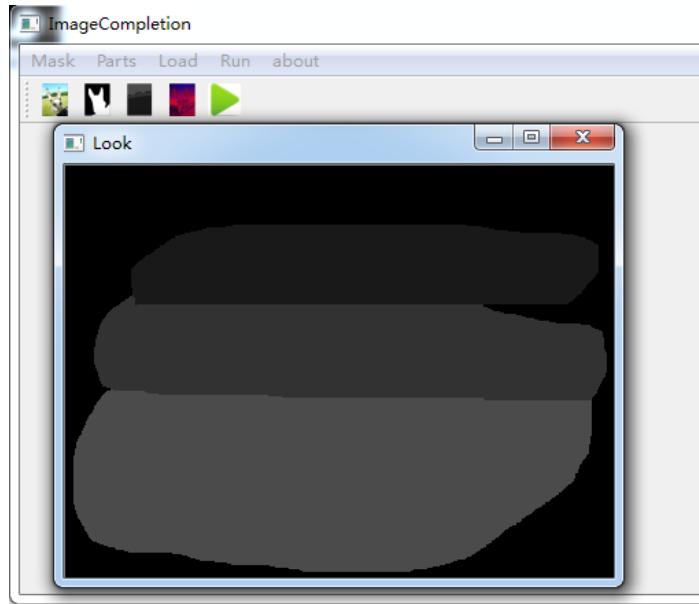


图 4.35 最终得到分区图

### 4.3 载入图像

运行前两个菜单后，便有了原图，mask 图，分区图，至此便可以进行正式的运行了。在运行前，需要将图全部读入，于是第三个菜单“Load”便是可以载入这几张图。

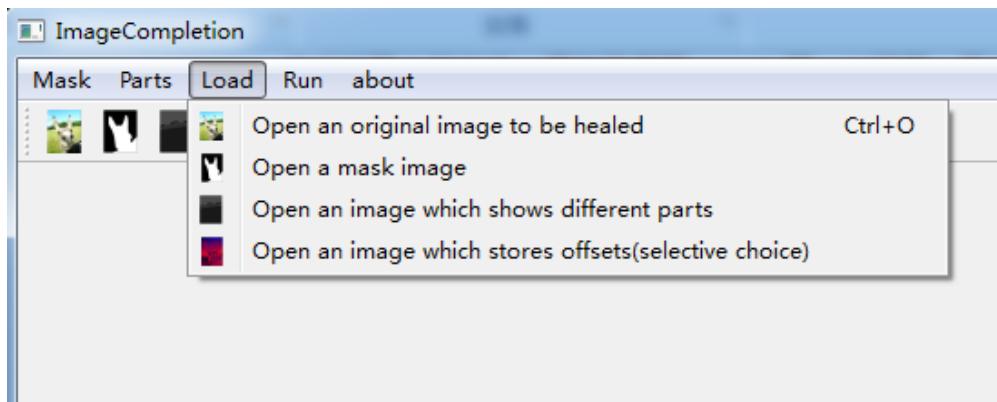


图 4.36 载入图像菜单，可以读入原图之前产生的 mask 图和分区图

“Load”菜单下第四项还可以读入用图像表示的图像统计特性的图，在第一次运行时程序没有产生这张图，所以可以不读入，只读入前三个即可。如果对第一次的修复结果不满意，在第二次运行同一张图时，由于第一次运行时会自动产生显示图像统计特性的图，就可以读入第四项，这样就不需要再次重复得到图像统计特性的过程，节省运行时间。

在菜单栏下有几个图标，按键功能与图像对应的“Load”下的操作功能一致。

#### 4.4 运行

当读入图像完成后，便可以运行程序了，可以点击“Run”菜单启动修复程序，也可以点击工具栏中的三角形来启动程序。

第一步是用户交互提供块偏移向量。(如果不提供可以直接按“Esc”)

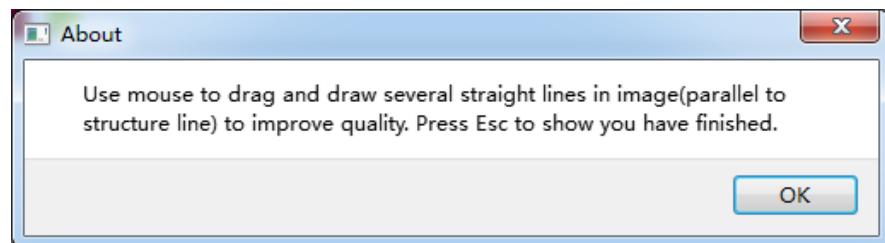


图 4.37 用户交互提供块偏移

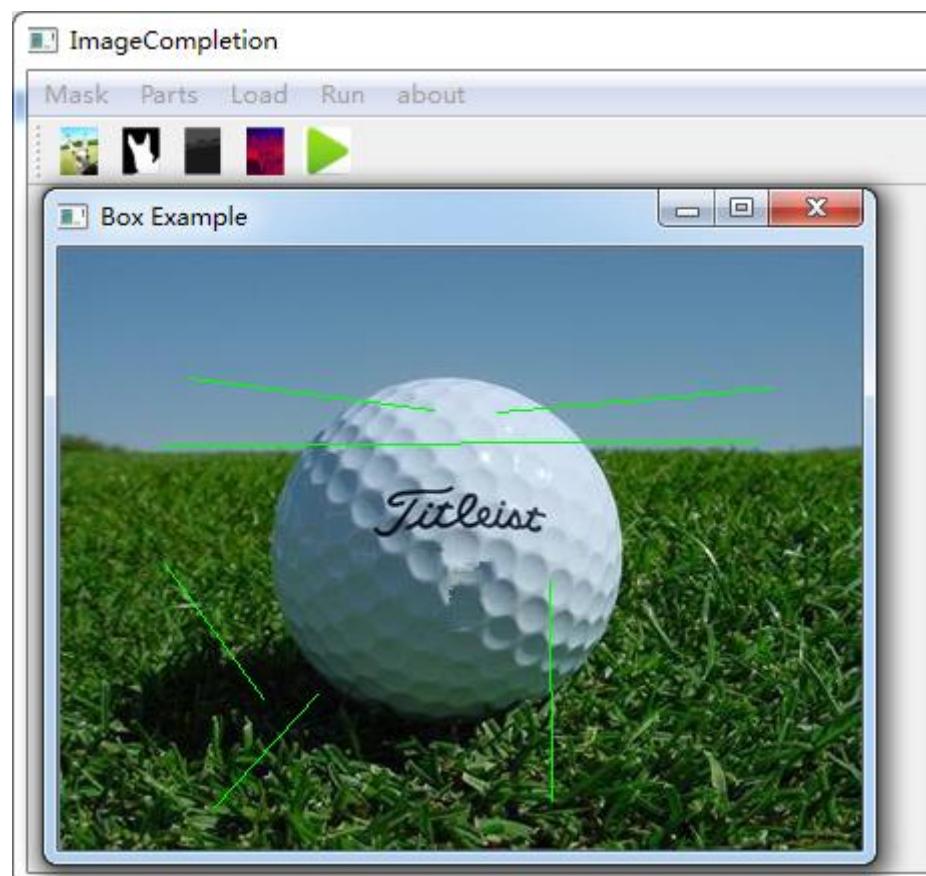


图 4.38 画直线方式得到块偏移

然后就会计算图像的统计特性，会有进度条显示运行进度。

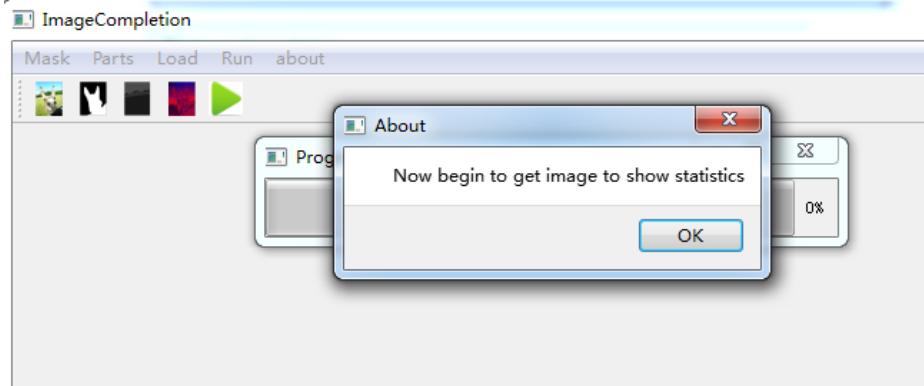


图 4.39 计算图像统计特性

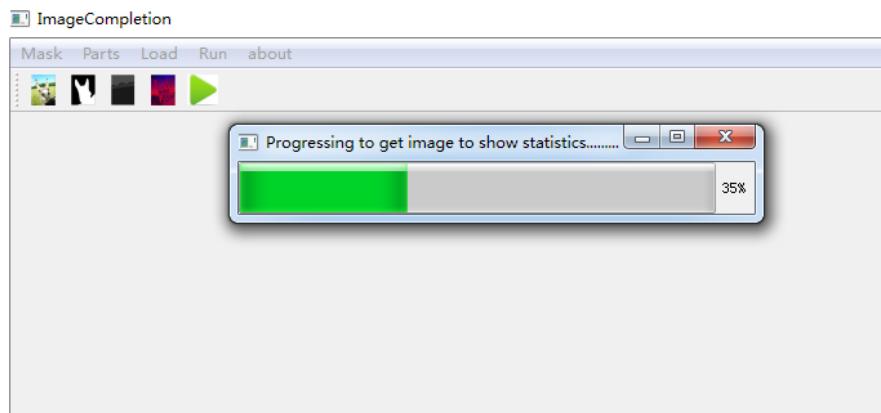


图 4.40 正在计算图像统计特性

运行完之后便产生了统计特性的图，第二次运行同一张图时便可以读入，这样就不必再次重复计算统计特性了。

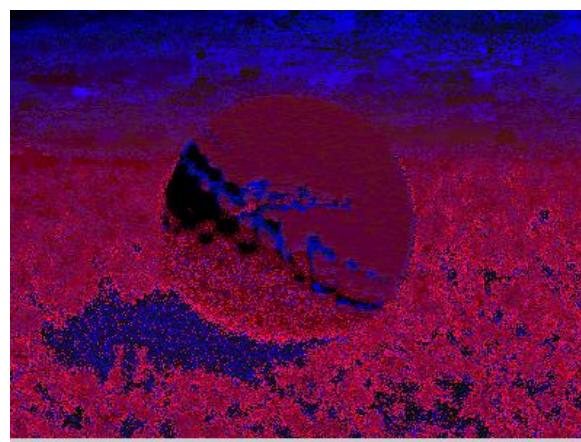


图 4.41 统计特性图

程序会询问是否观察整个修复过程，选择“Yes”会出现窗口显示图像一步一步被修复的过程，速度较慢，选择“No”则会出现进度条快速修复图像。

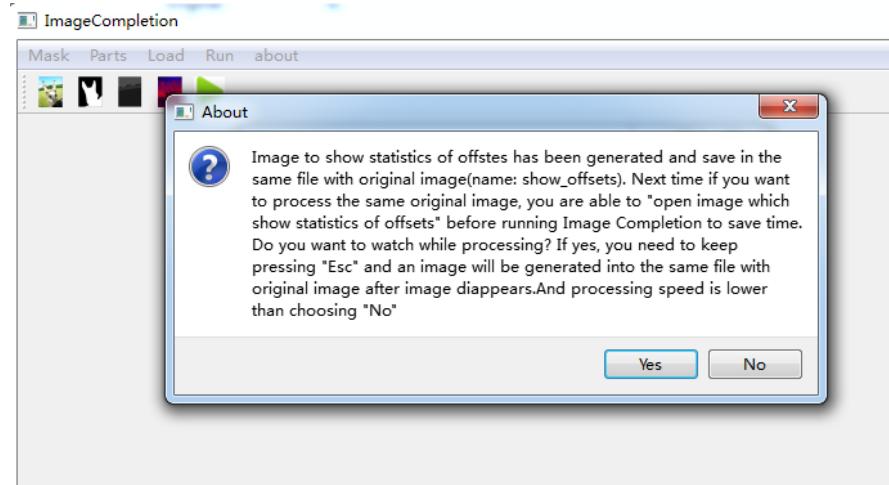


图 4.42 询问运行时是否观察修复过程

选择“Yes”后窗口如下：

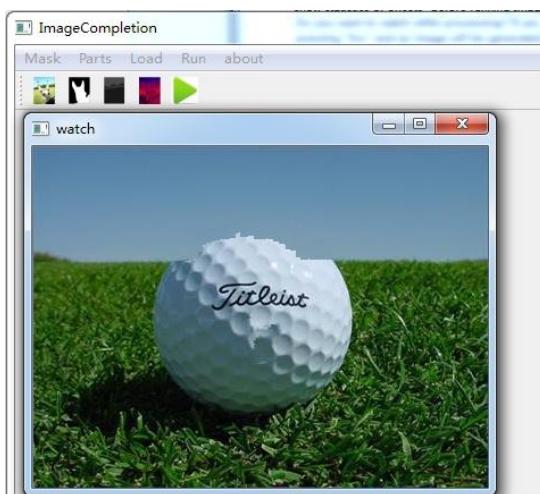


图 4.43 逐渐修复区域 1

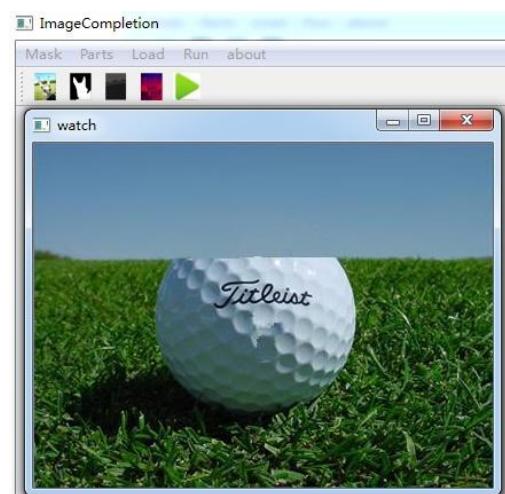


图 4.44 区域 1 修复完毕，修复区域 2

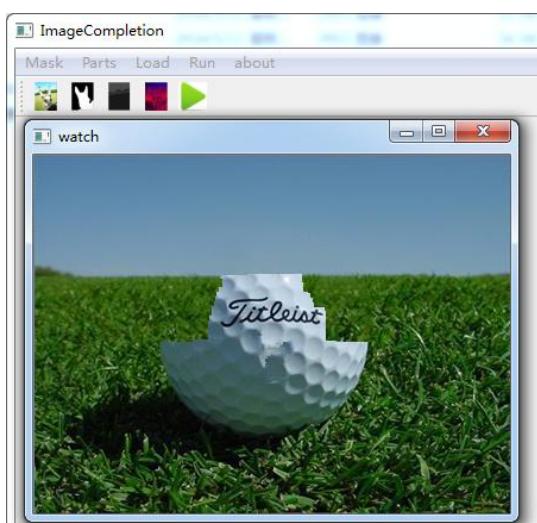


图 4.45 修复区域 2

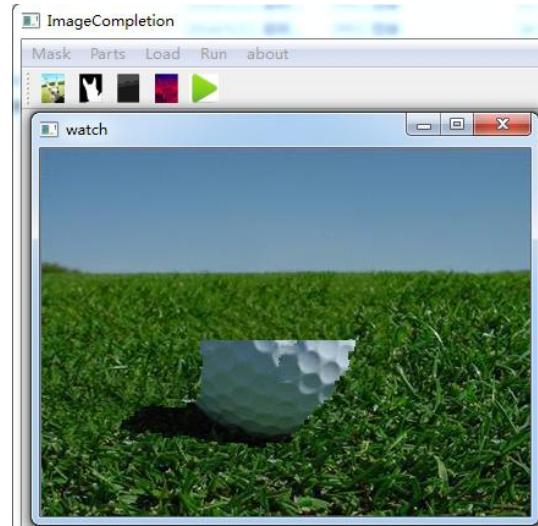


图 4.46 修复区域 3

修复完毕后，窗口自动关闭，显示修复成功。

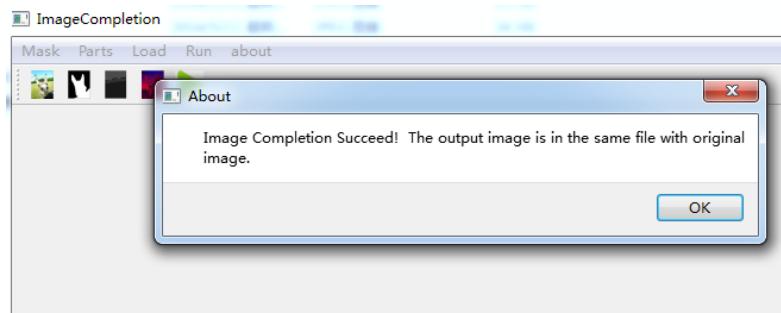


图 4.47 修复成功，图像已保存

选择“No”窗口如下，会显示修复进度条。

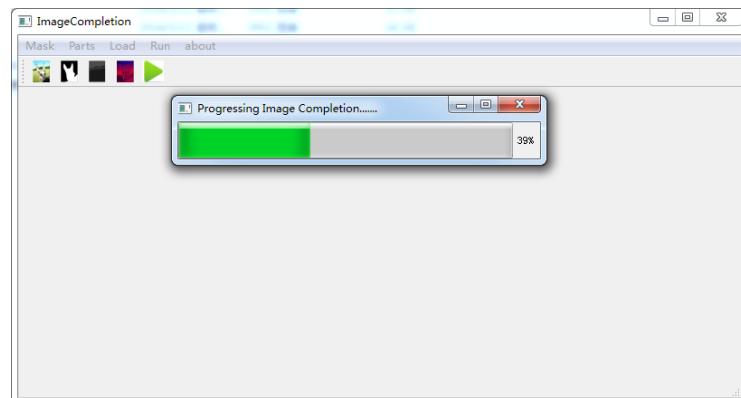


图 4.48 不观察修复过程直接修复

修复完成后也会得到提示。

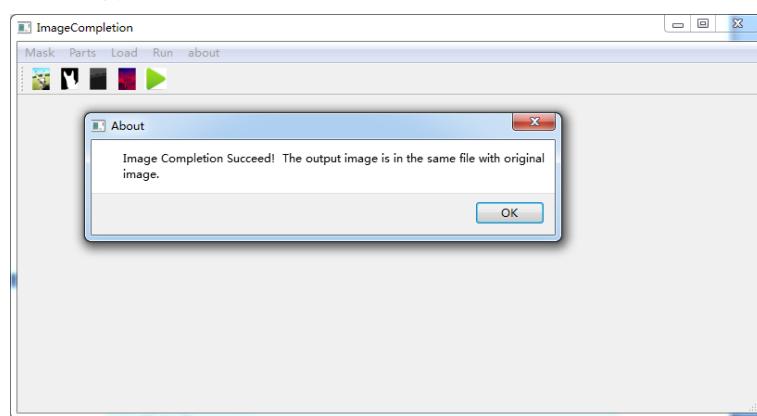


图 4.49 修复完成

最终修复结果图被保存在同一文件夹下。



图 4.50 修复结果图

至此，修复过程成功完成，得到了一张修复好的图，并被存入文件夹下。

## 第 5 章 总结

随着数字图像处理技术的发展，图像修复作为其中一个重要的分支在各个领域发挥着举足轻重的作用，很多研究者都在致力于创造出新的高效算法。

本文在传统的基于样例的图像修复的基础上，仔细分析了基于样例的图像修复的缺点和原因，并提出相应的解决方案。首先是由于图像本身数据量庞大，块匹配查找耗费了大量的时间，而通过利用图像的块偏移统计特性，则可以有重点性的进行查找，有效缩短运行时间。而用户交互产生额外块匹配向量则更加确保块匹配查找结果不会因为缩短了运行时间而变差，并且给用户提供了更多尝试机会。最后，本文提出了交互分区的方法，使得自动精确修复结构复杂的图像成为可能，用户对修复过程有了更大的控制性，计算机会按照人的意愿进行图像修复，可以说本文很多成功的修复结果完全无法离开分区算法的功劳。

为了验证算法的有效性，本文对各种图像进行修复，并在实验过程中根据结果反思不足，改进参数设置和算法，不断提高修复速度和质量，根据众多的实验，本文所提算法具有较高的修复效果。

## 参考文献

- [1] M. Bertalmio, G. Sapiro, V. Caselles, C. Ballester. Image Inpainting [J]. *Proceedings of SIGGRAPH* 2000, 2000
- [2] 叶学义, 王靖, 赵知劲, 陈华华. 鲁棒的梯度驱动图像修复算法 [J]. 中国图象图形学报, 2012, 17, 630-635. Ye Xueyi, Wang Jing, Zhan Zhijin, Chen Huahua. Robust gradient driving iamge inpainting method [J]. Jornal of Image and Graphics, 2012, 17, 630-635
- [3] A. Criminisi, P. P érez, K. Toyama. Region filling and object removal by exemplar-based image inpainting [J]. *Image Processing, IEEE Transactions on*, 2004, 13, 1200-1212.
- [4] C.-W. Fang and J.-J. Lien, “Rapid image completion system using multiresolution patch-based directional and nondirectional approaches,” *IEEE Trans. Image Process.*, vol. 18, no. 12, pp. 2769–2779, Dec. 2009
- [5] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, “Image completion with structure propagation,” *ACM Trans. Graph.*, vol. 24, pp. 861–868, 2005
- [6] K. He, J. Sun, Statistics of Patch Offsets for Image Completion [C]. *ECCV*, 2012.
- [7] Zhuang Honglin, Wang Wenbing, Duan Peng, “Image Processing Methods Based on Total Variation,” Journal of Yunnan Nationalities University(Natural Sciences Edition)., vol.17 No.2 Apr. 2008
- [8] RUDIN L,OSHER S,FATEMI E. Nonlinear Total Variation Based Noise Removal Algorithms[J]. *Number Math*, 1997, 76:167-188
- [9] CHAN T, WONG C. Total Variation Blind Deconvolution[J]. *IEEE Trans Image Process*, 1998, 7 :370-375
- [10] ZHANG Fu-mei, WU Jin-xue, MENG Xian-chao, LI Ying-ming. Numerical analysis and experiment based on inpainting model by curvature-driven diffusions. Journal of Shandong University of Technology(Natural Science Edition)., vol. 22 No.3 May 2008
- [11] Y. Boykov, O. Veksler, R. Zabih. Fast approximate energy minimization via graph cuts [J]. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2001, 23, 1222-1239.

- [12] Barnes C, Shechtman E, Finkelstein A, et al. PatchMatch: a randomized correspondence algorithm for structural image editing[J]. *ACM Transactions on Graphics-TOG*, 2009, 28(3): 24.
- [13] K. He, J. Sun, Statistics of Patch Offsets for Image Completion [C]. *ECCV*, 2012.