Project1 实验报告

计 32 黄世宇
2013011304

【项目概述】

该项目做的是基于内容敏感的图像缩放实验。在很多时候，我们希望改变图像的尺寸，但不希望影响图片中的一些重要元素。通过选取图片中的像素，计算像素的重要性，把不重要的像素进行删除或者重复，完成对图片的尺寸变换。

通过一定的算法计算图片中像素的重要性和选择合适的一组像素进行删除，力求做到删除后的图像对重要的区域影响最小。

该项目完成了图像缩放功能，还进一步实现了预处理，加强版的像素删除代价，以及对预处理的加速。

【开发环境】

- 实验语言：C++
- 实验平台：Visual Studio 12
- 实验使用工具：OpenCV-2.4.9
- 版本控制：github
- 项目地址：https://github.com/huangshiyu13/Media-Computing

【运行说明】

使用 Visual Studio 12 打开 resize.sln（项目的工程配置文件），按 F7 进行编译（需要预先配置 OpenCV-2.4.9），编译成功后，按 F5 运行程序。

【算法细节】

1.算法基础

每个像素的重要性可以定义为

$$e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|$$

每条竖直细缝定义为

$$\mathbf{s^x} = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \text{ s.t. } \forall i, |x(i) - x(i-1)| \leq 1$$

同样，每条水平细缝定义为

$$\mathbf{s^y} = \{s_j^y\}_{j=1}^m = \{(j, y(j))\}_{j=1}^m, \text{ s.t. } \forall j |y(j) - y(j-1)| \leq 1$$

一条细缝的能量定义为

$$E(\mathbf{s}) = E(\mathbf{I_s}) = \sum_{i=1}^n e(\mathbf{I}(s_i))$$

算法每次会选取能量最小的细缝进行删除，直至宽度或高度符合要求。

该算法通过动态规划来选择每条需要删除的细缝，以水平说缩放为例，我们需要在图中找一条最有的竖直细缝。假设一张图为矩阵 M^{n×m}，那么用以下转移函数进行动态规划

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

这样便解决了删除一条细缝的问题，但是图像需要删除多条竖直细缝和多条水平细缝时应该怎么办呢？难道需要人为规定需要删除行列顺序。显然人为指定删除的顺序不一定是最优的，我在算法中又加入了对删除顺序的动态规划。如果原图大小为 n x m，需要变换成 (n-r)x(m-c) 大的尺寸。就可以用动态规划进行求解。

首先令

$$\mathbf{T}(0,0) = 0$$

状态转移方程为

$$\mathbf{T}(r,c) = min(\mathbf{T}(r-1,c) + E(\mathbf{s^x}(\mathbf{I_{n-r-1\times m-c}})), \mathbf{T}(r,c-1) + E(\mathbf{s^y}(\mathbf{I_{n-r\times m-c-1}})))$$

这便得到了最优的删除顺序。

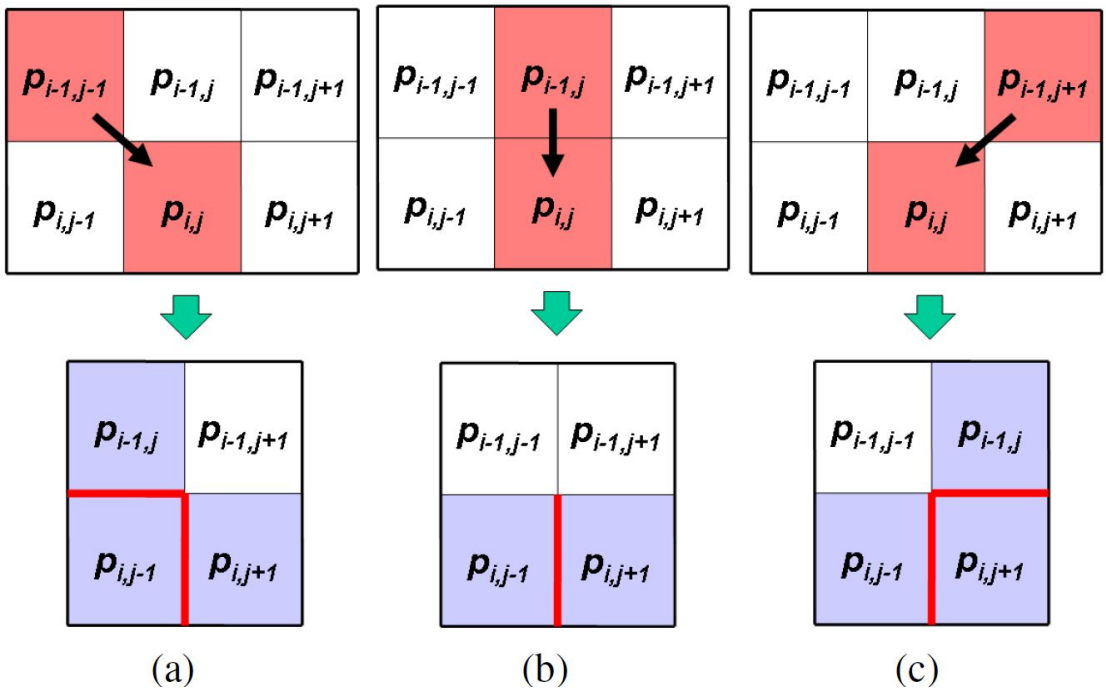
2. 预处理加速优化

由于上述方法使用了两层的动态规划，算法复杂度高，运行时间很长，所以我通过预处理来进行加速。

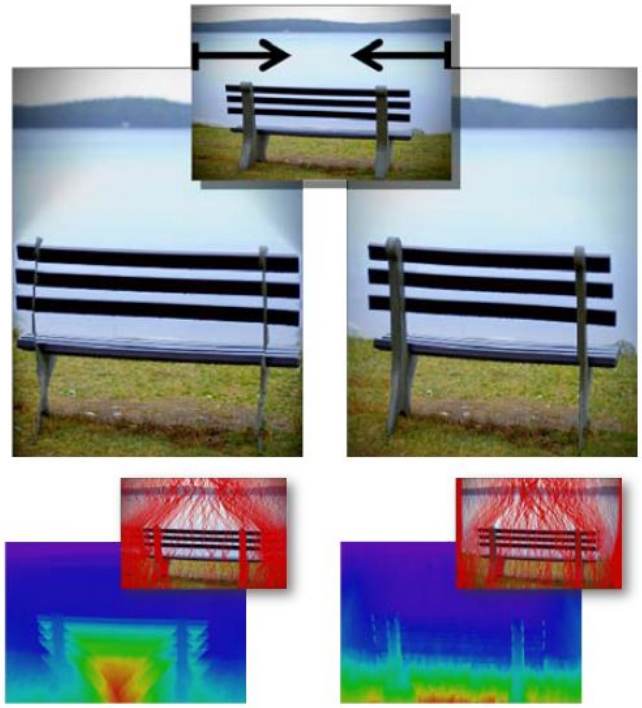
在选定图片后，立即对图片进行预处理。即记录每种尺寸下需要删除的细缝和删除顺序。然后在实际进行缩放的时候，就可以比较快速地得到想要的尺寸。比如下图，需要记录下红色细缝像素所在的位置，为以后删除做准备。这样大大地加速了实际操作的速度。

3. 评价方式优化

原来的细缝增删算法考虑的是删除某个像素的代价，但是删除某个像素后会导致原来不相邻的像素变成相邻，这便插入了新的代价。如下图所示



所以删除某个像素的代价还需要加上删除该像素后邻边的代价。这种方法对于下面这种图片有着更好的效果。



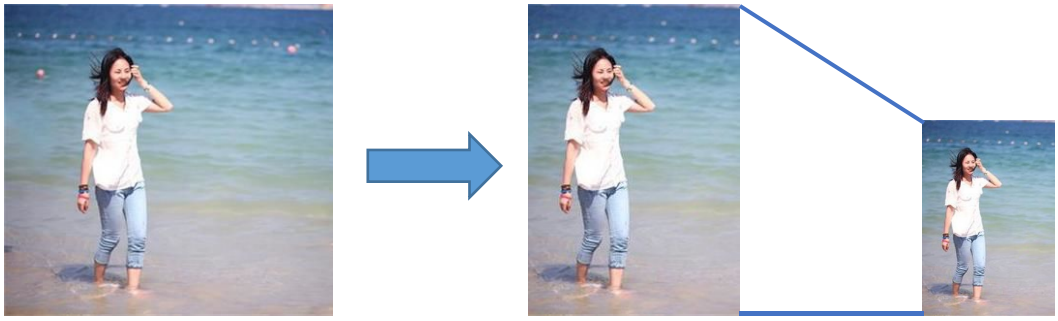
4. 对预处理进行加速

有了预处理后，虽然实际操作等待的时间变短了，但是预处理的时间又变得很长。所以，我对预处理提出了一种加速方案。

首先，在预处理时，我们不需要考虑对竖直和水平的同时缩放。只需要考虑对仅对水平或仅对竖直进行缩放。这样时间复杂度便从 n^4 降到了 n^3。预处理速度得到了很大的提高。

然后，在实际操作的时候，先把图像按竖直方向或按水平方向进行删除，直到达到和目标图像一样的长宽比例，然后再把图像按等长宽比例缩放到目标尺寸。

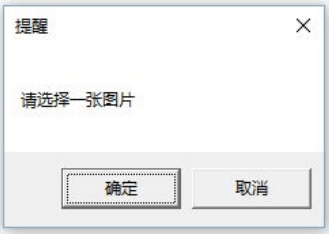
由于预处理加速改变了算法的流程，所以我提交的程序版本有两个，V1 版本的不含预处理加速，用的是动态规划套动态规划的算法，V2 版本使用了预处理加速，V2 版本的功能更完善。



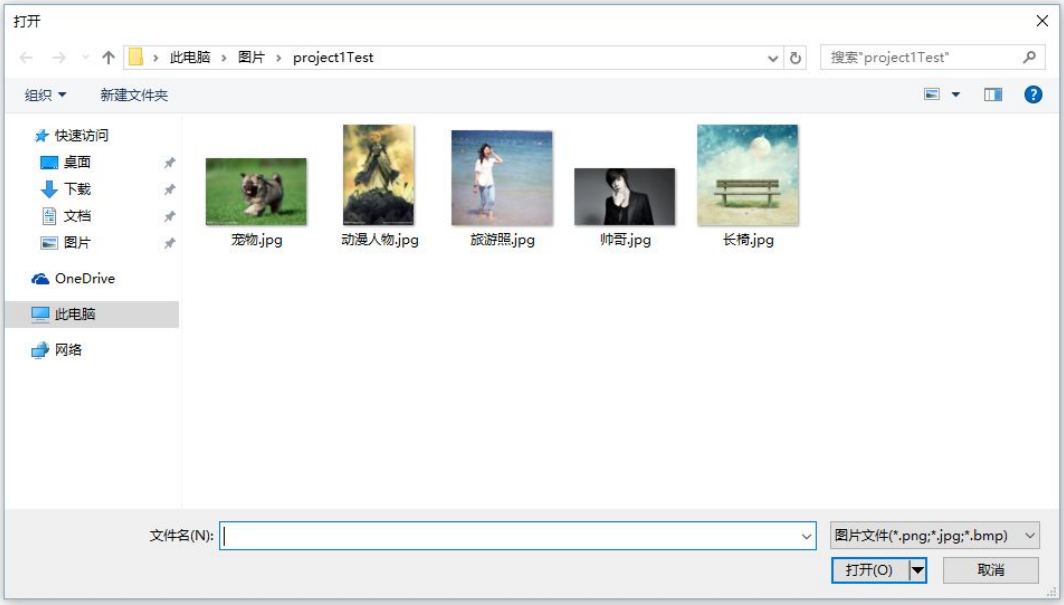
原始图像　　删像素后的图像　　目标图像

【结果说明】

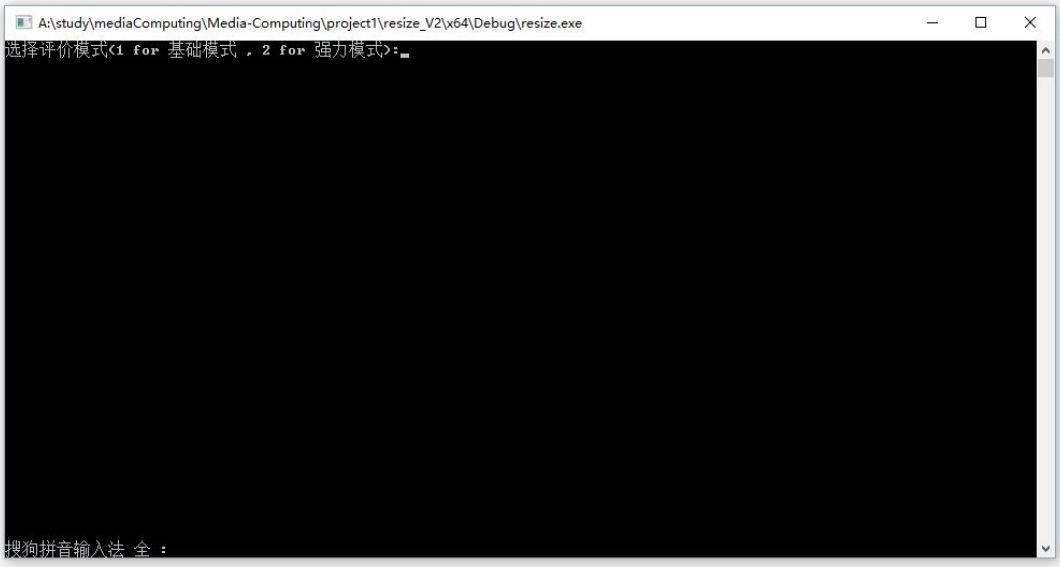
运行程序后会提示选择一张图片。



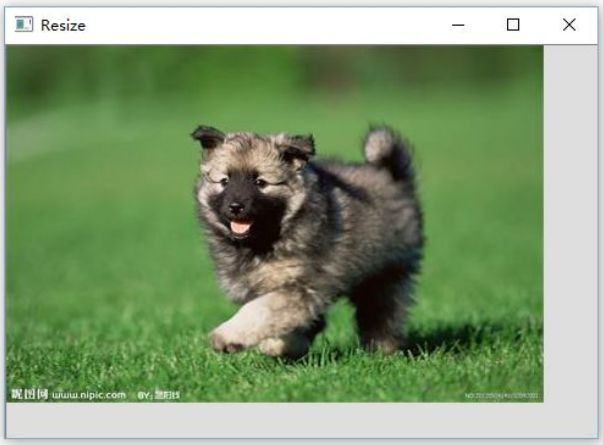
点确定选择一张图片。



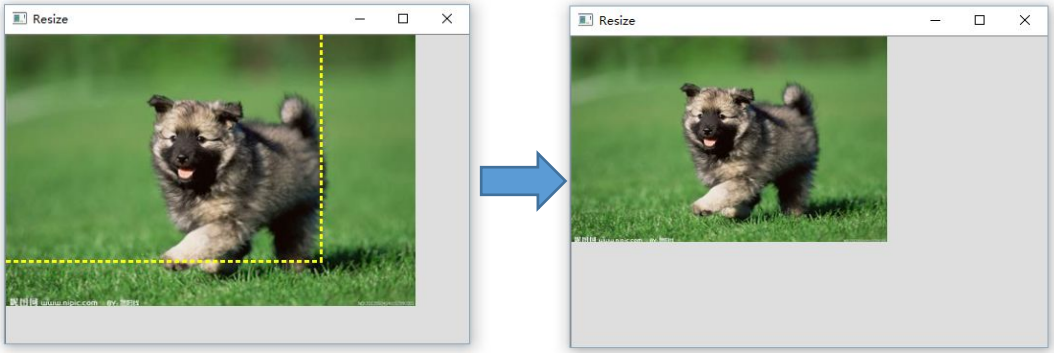
选择图片后会提示选择评价函数的模式。输入 1 选择基础模式，2 选择加强模式（即考虑删除像素后的相邻元素的代价）。



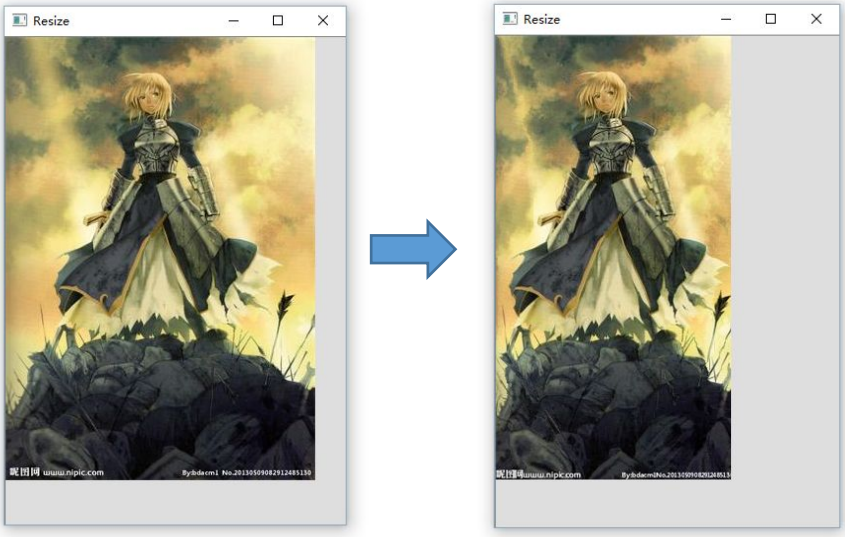
选择完后，程序会对图片进行预处理，这需要等待一段时间。然后会出现一个带有图片的窗口。

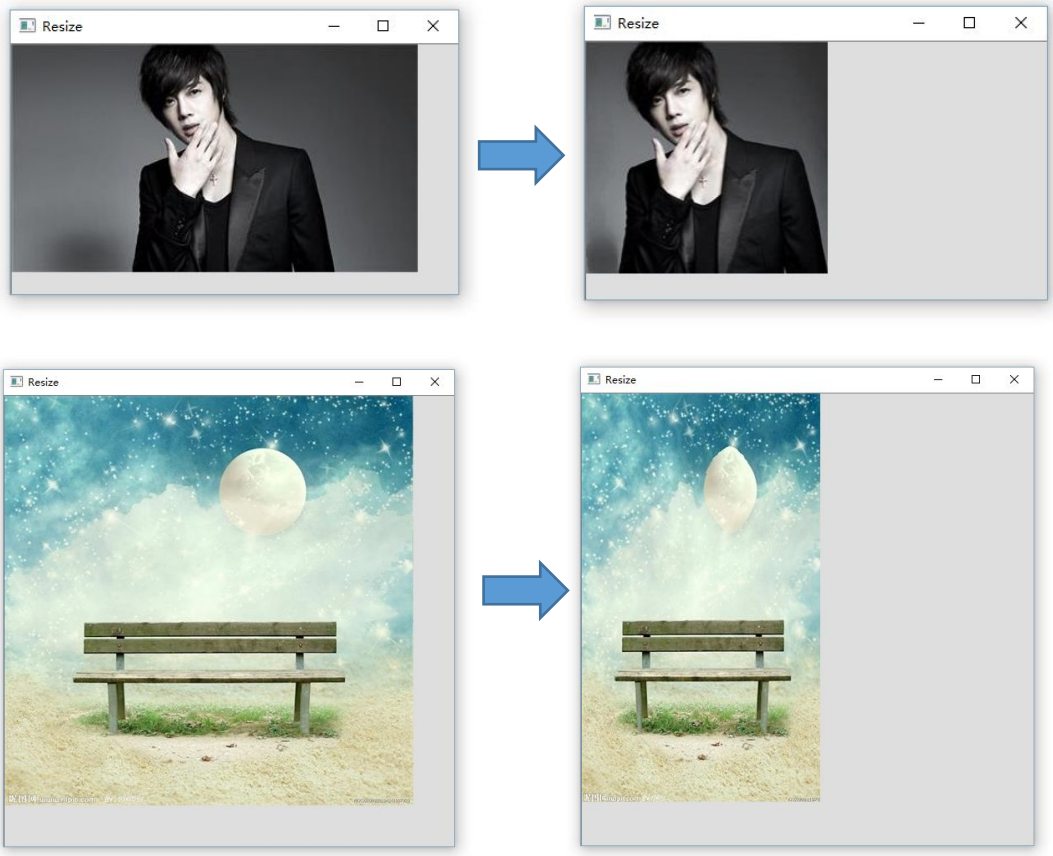


把指针移到图片下边缘或者图片右边缘或者图片右下角，按住鼠标左键进行拖动，便可以改变图像的尺寸，松开鼠标左键显示尺寸变换后的结果。



更多结果：





【项目总结】

1. 通过这个项目，我学到了更多关于 OpenCV 的用法。
2. 加深了对论文的理解。
3. 取得了比较好的结果。
4. 预处理时被内存泄露所困扰，但还是顺利解决了，很开心。
5. 尝试了新的想法--预处理加速，并且实现了，很开心。

【参考文献】

[1]Seam Carving for Content-Aware Image Resizing. Shai Avidan et. al. SIGGRAPH 2007.
[2]Improved Seam Carving for Video Retargeting. Michael Rubinstein et. al. SIGGRAPH 2008.