Image Work Shop:patchmatch 界面

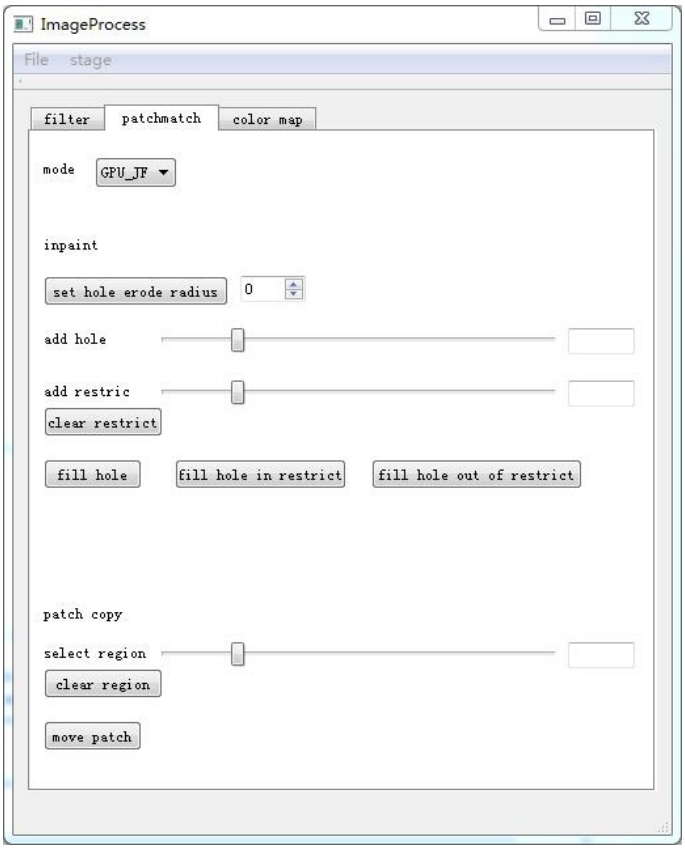
sunsiy10 2015.01

实现时使用 qt5.1.1 和 opencv2.4.9，用 qt 做控制界面，用 opencv 实现与图片相关的操作。

一、 简介和使用方法

CPU 实现对应的代码为 MyPatchMatch.h 中注释为 new inpaint method 和 new copy method 的部分；GPU 实现对应的代码为 MyPatchMatch.h 中注释为 opencl 的部分。

程序界面如下：



Mode 用于选择 CPU 还是 GPU；

inpaint 部分

set hole erode radius：在补洞时腐蚀洞的边界，这里可调节半径；

本程序可直接检测到 RGB 为(0,0,0)和(255,0,0)的洞，若检测不完全可以调节 erode radius 或者手动添加。

Add hole 用于在图像中加洞（黑色），这里可以控制加洞的画笔半径；

add restrict 可以为补洞时为洞指定想要填补的区域；

clear restrict 按钮：用于清除 restrict；

fill hole 按钮：直接填补所有洞；

fill hole in restrict 按钮：只填补 restrict 区域中的洞，填补内容受到 restrict 区域限制；

fill hole out of restrict 按钮：填补所有的洞，填补内容受到 restrict 区域限制；

patch copy 部分

select region：选择想要复制的区域；

clear region 按钮：取消已选择的想要复制的区域；

move patch 按钮：移动已选择的区域并得到结果；

菜单栏

File 菜单栏中的 open 和 save as 用于打开和保存图像。

在遇到不满意的结果时可以点 stage 菜单栏中的 back 即可回到上一次的结果，满意则点 stage 菜单栏中的 save 进行保存。若想要看上一次的结果可点 stage 菜单栏中的 back image，此功能只能看上一次的结果，再往前的结果被舍弃。

二、 实现

2.1 实现过程

利用 PatchMatch 实现这些功能的流程是先用图像金字塔将原图缩小至某一层（inpaint 为直到洞不存在的一层，copy 为洞不存在的下一层），然后对小图片多次执行 patchmatch 得到比较好的 NNF 初始值。

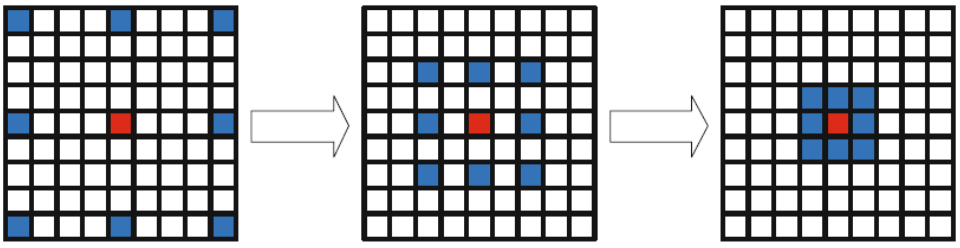
随后根据图像金字塔放大图片，并拓展 NNF 为当前图片 scale，进行多次 patchmatch 操作以得到精确的 NNF，再 vote 得到新的结果图片。

重复上一步直到图片变为原图片大小。

实现时 patch 大小为 5×5。

CPU 实现的 patchmatch 采用的是 propagation 和 random search 结合的方法，每个 patch 只寻找一个最近邻 patch。

GPU 实现的 patchmatch 采用的是 jump flooding 方法，使用的语言是 opencl。每个 patch 寻找 5 个最近邻 patch。jump flooding 便于 GPU 并行实现，每个像素点寻找其周围 8 个像素点所在 patch 的最近邻 patch 进行比较。这 8 个像素点的定位方式如下图所示。



其中 L 的初始值为图片的宽和高中的较大值，没执行一次 L 减半。L 变为 1 后再多执行两次，这两次 L 分别为 2 和 1。

2.2 实现技巧

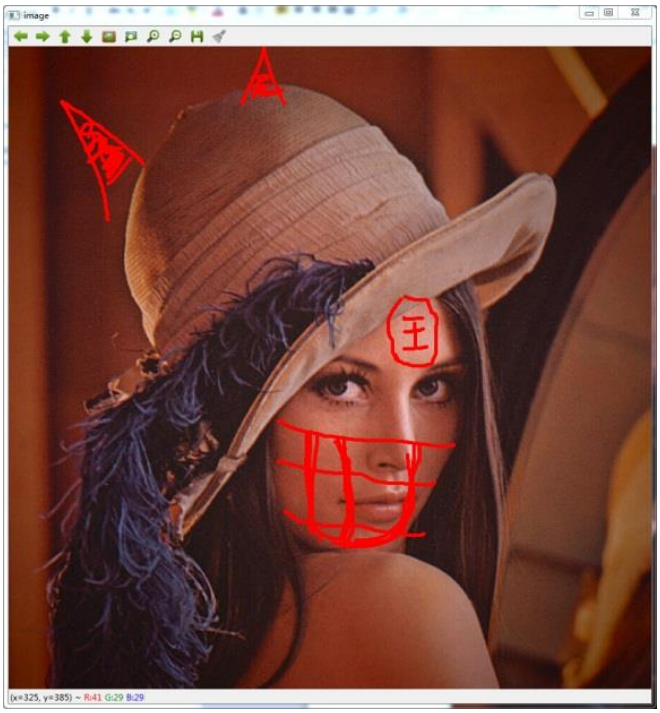
我经过多次尝试，虽然都是用了图像金字塔，但效果仍有不小的差距。我总结了 PatchMatch 实现过程中的一些有利于提升效果的技巧，这里简单介绍。

A. 建立图像金字塔时应采用滤波的方式进行，这样比直接调用 opencv 的 prydown 函数更能保证原图特征的位置。
B. Vote 时最好根据不同 patch 与目标 patch 的距离对这些 patch 的贡献加权；而且不同的贡献值可能会差异很大，应做进一步处理，可以利用直方图并只取分布在中间的部分来降低“坏贡献值”的干扰。
C. 在金字塔高层时应进行较多的迭代以得到更为精确的初始值。
D. 要利用距离函数中对洞的惩罚这个思想，并把这个思想用到交互上去。

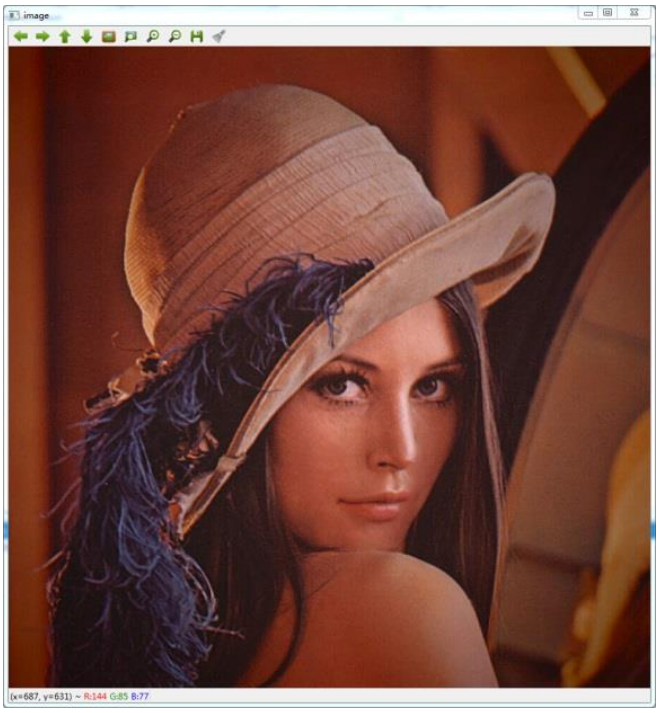
三、 inpaint 结果

对于有些图像，一些区域无法直接修补成合理的结果，如块状区域，这时需要进行交互，令其在用户指定的 restrict 的区域内寻找 patch。

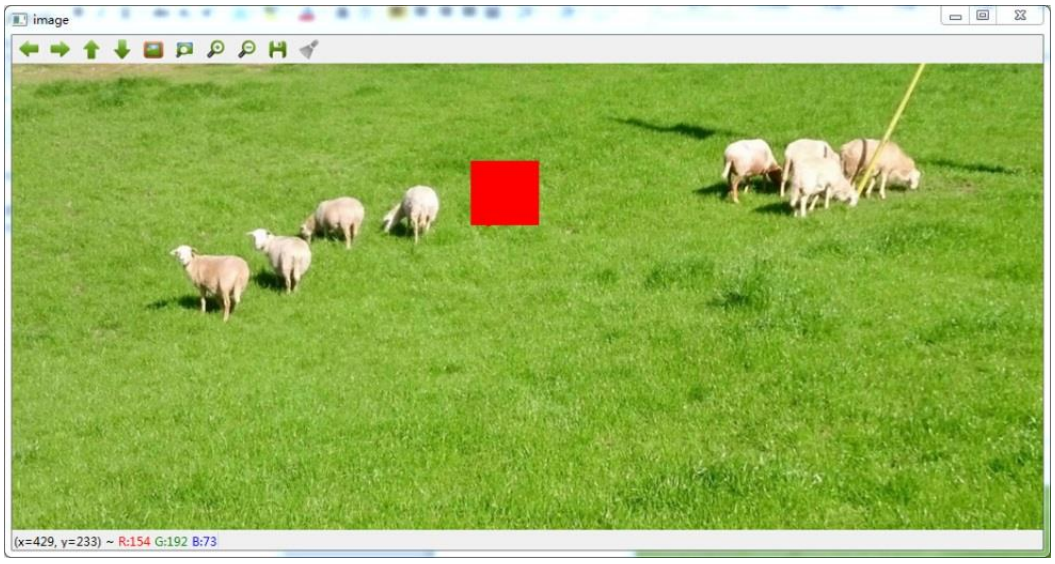
3.1 不使用 restrict 交互的结果



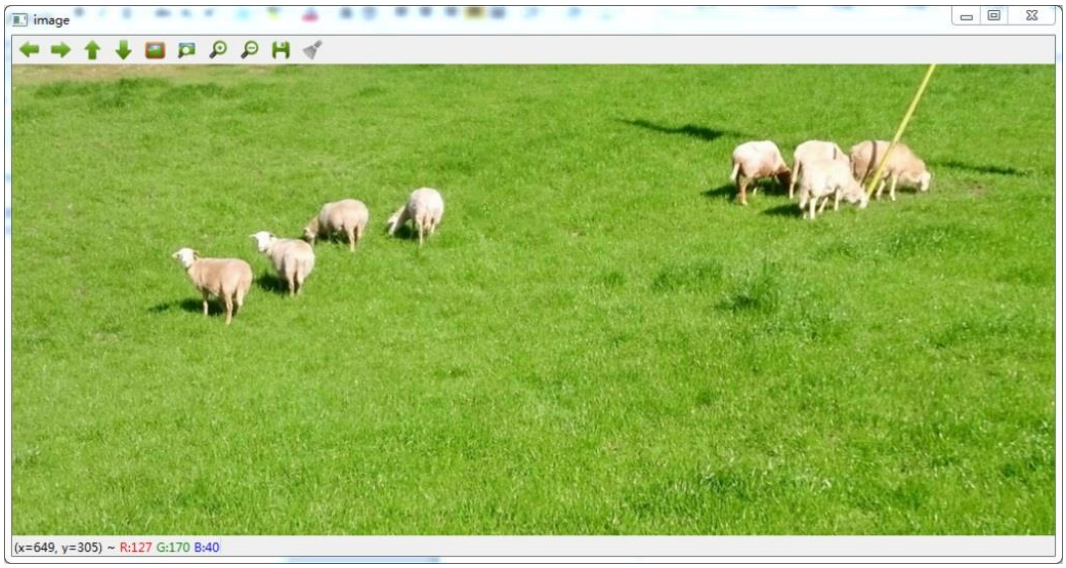
原图



修复后

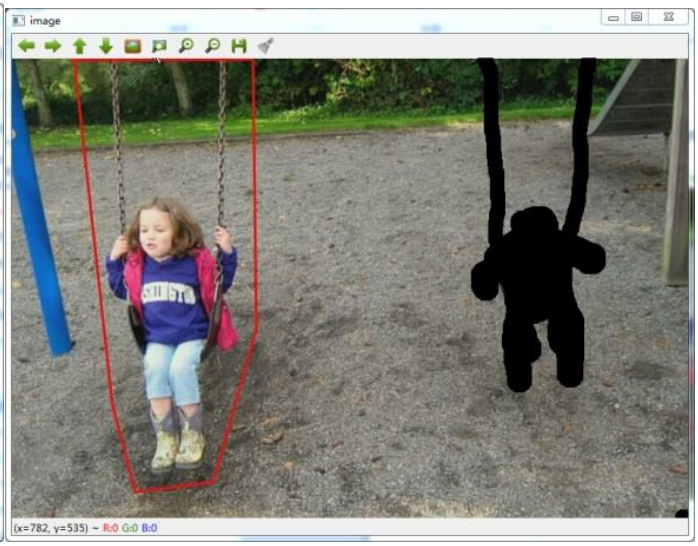


原图

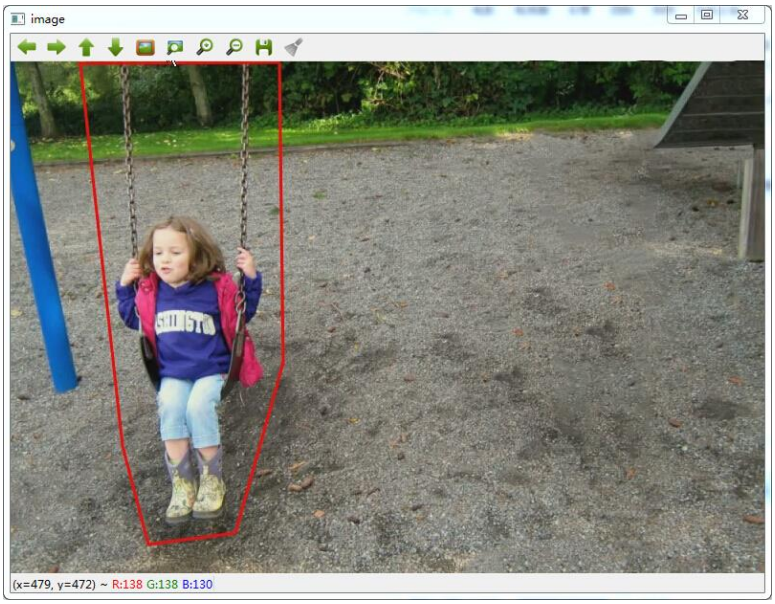


修复后

原图



加洞

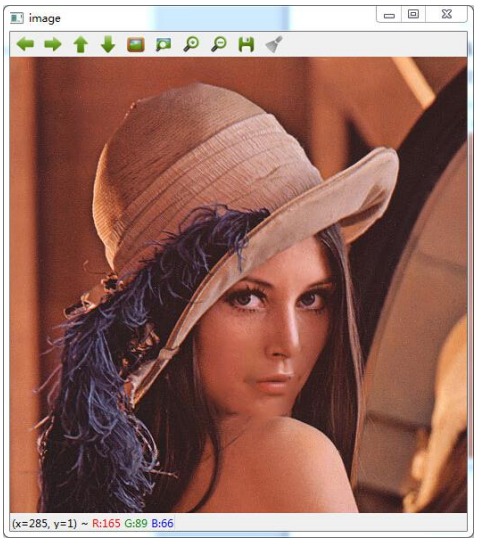


结果

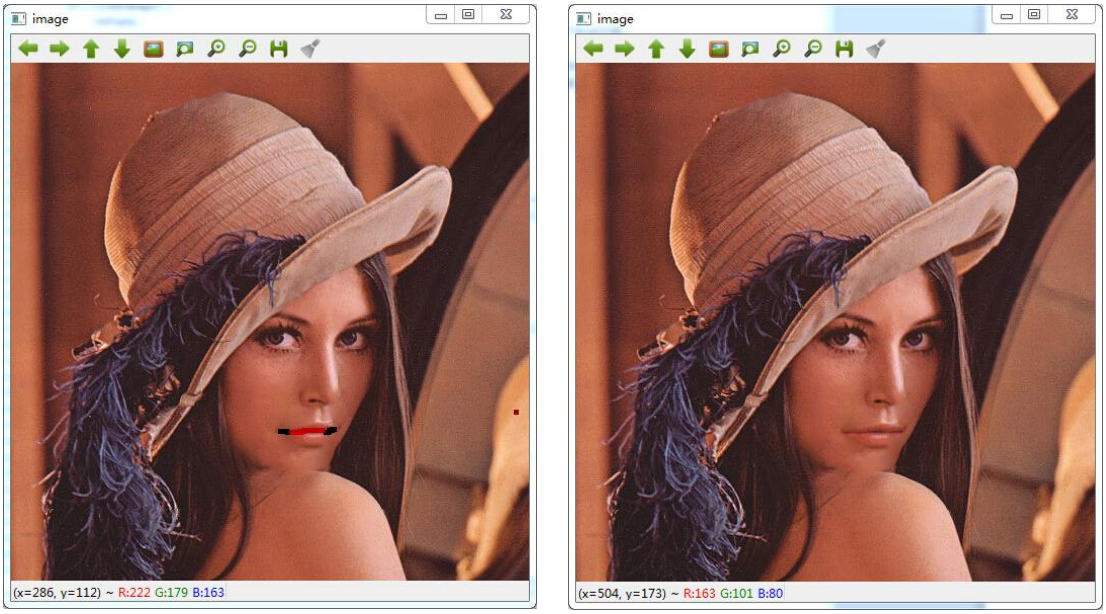
3.2 使用了 restrict 交互的结果

例子 1：lena

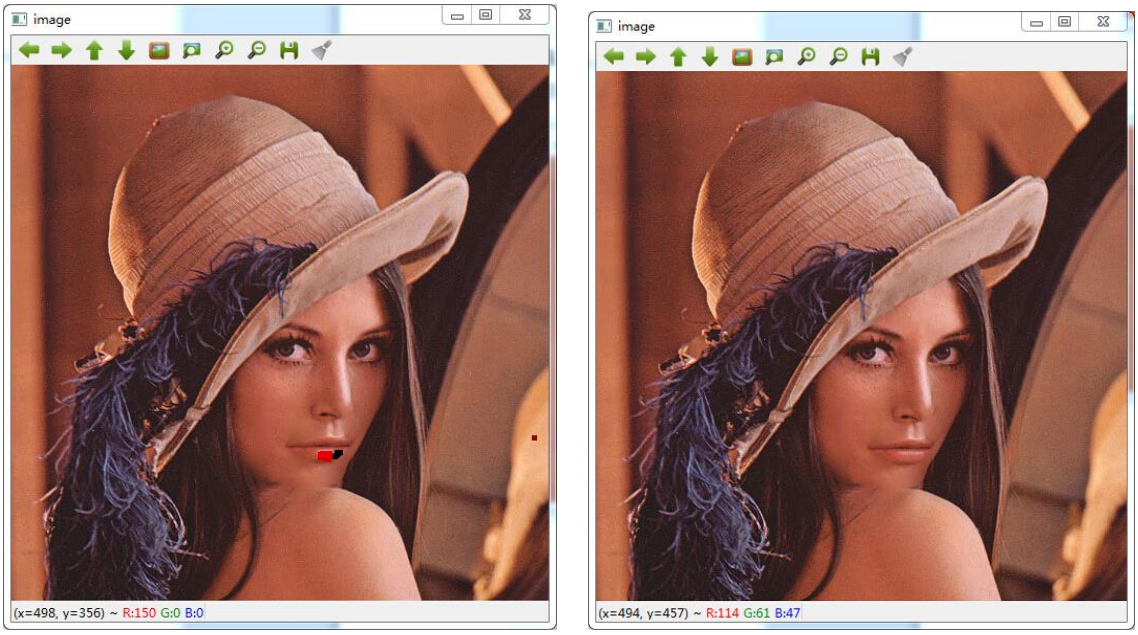
对于 lena 图中嘴的部分，有时候不交互将会出现缺失，所以需要交互进行弥补。



补洞后发现嘴有缺陷

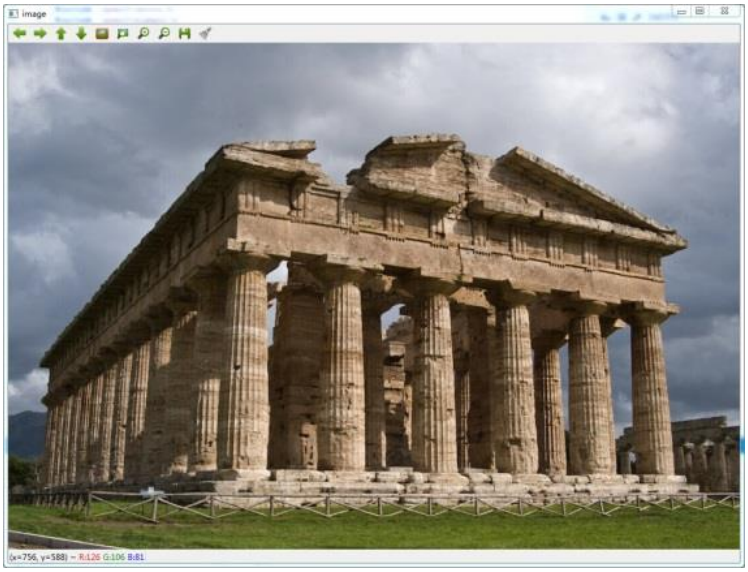


补全嘴线

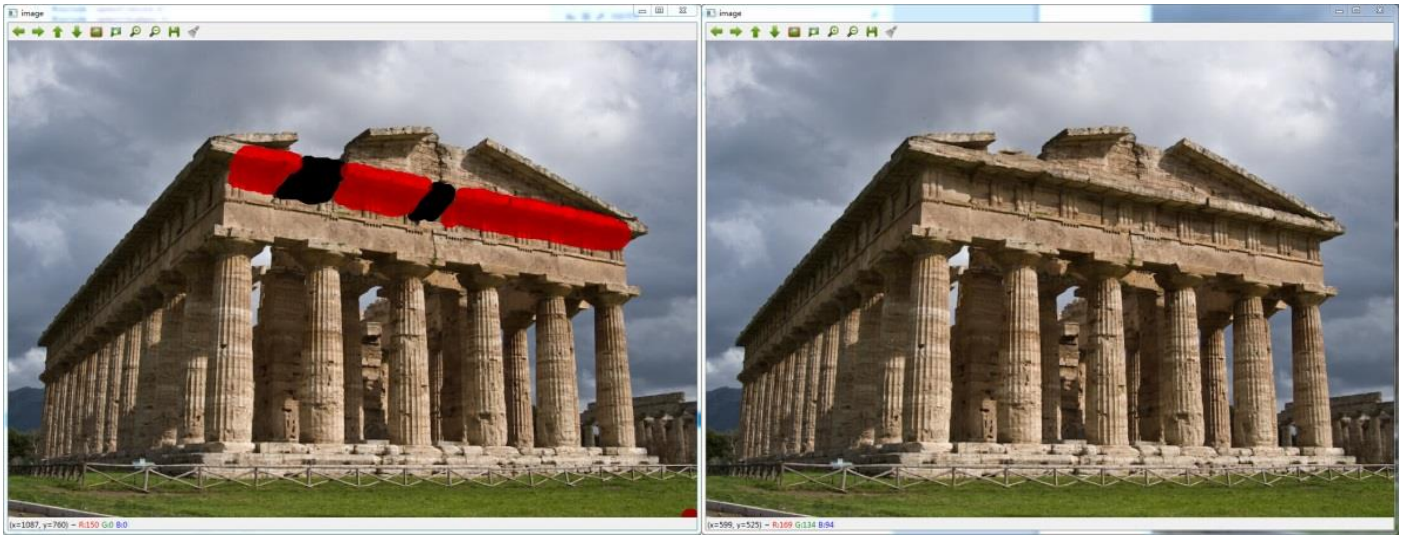


补全嘴唇

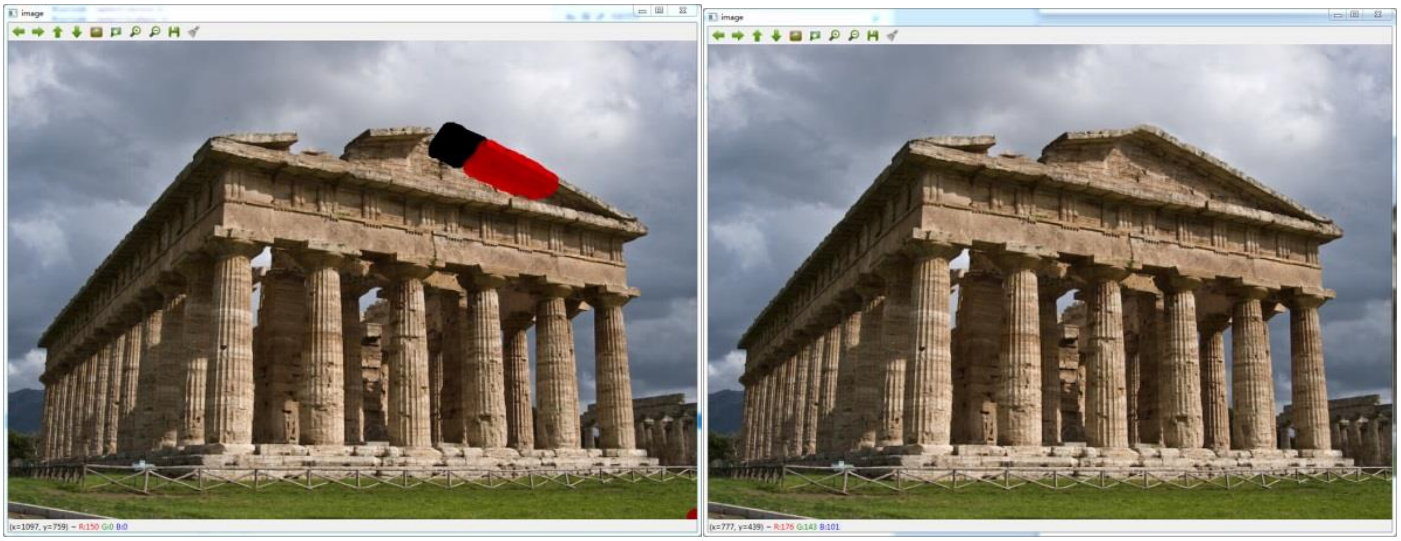
例子 2：



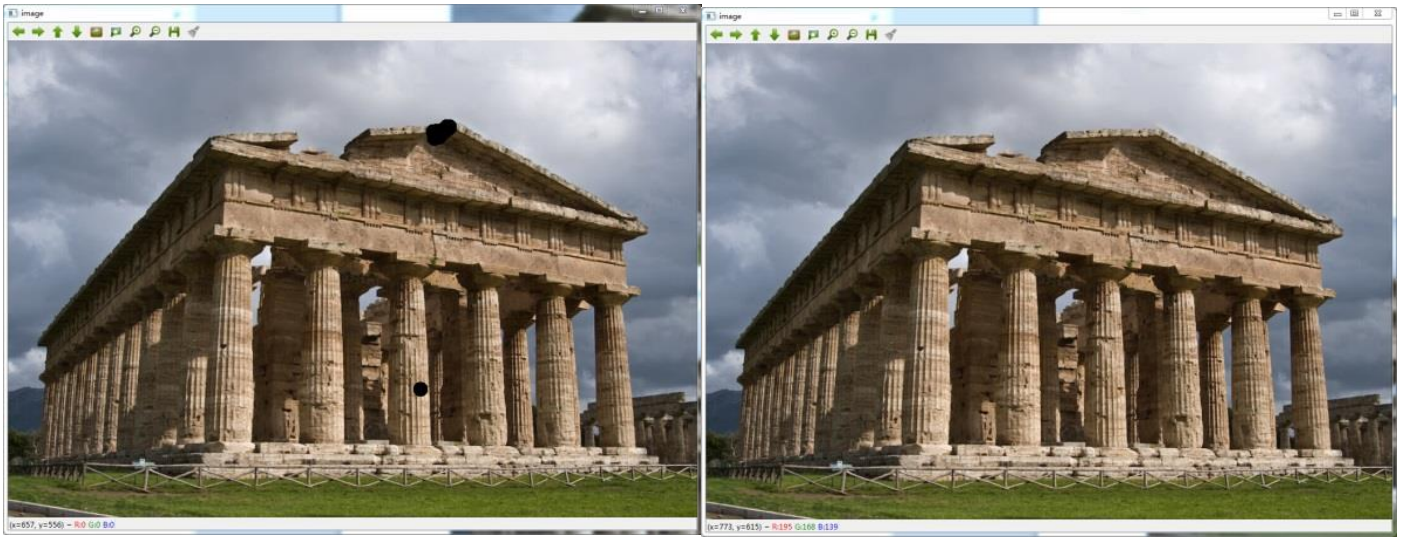
原图



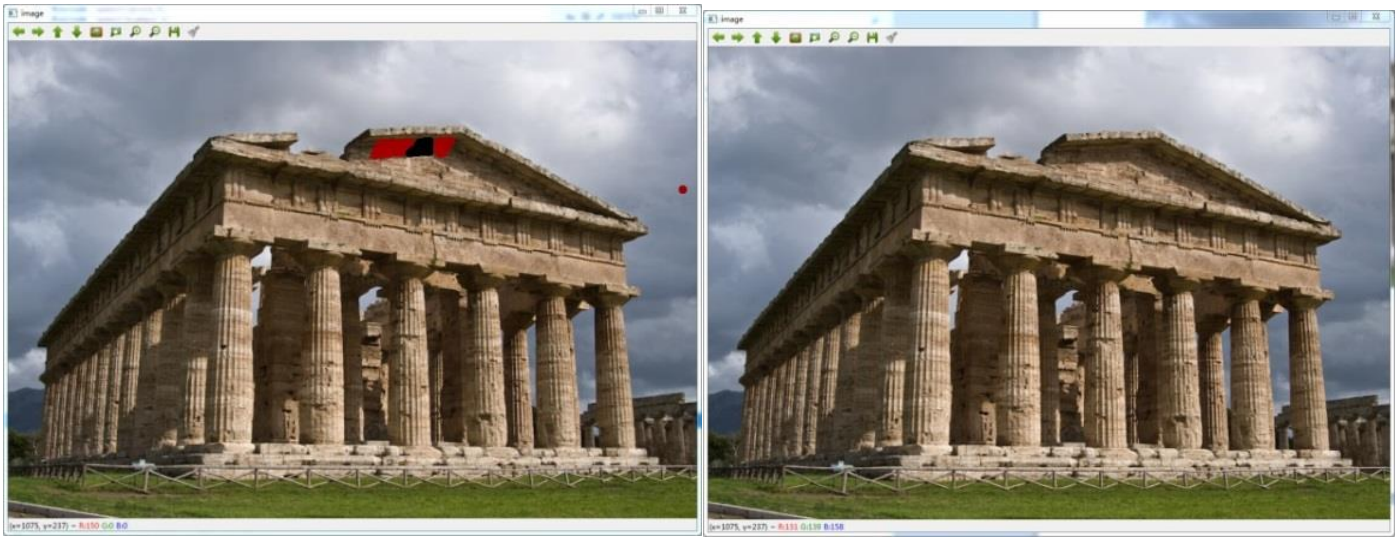
第一步



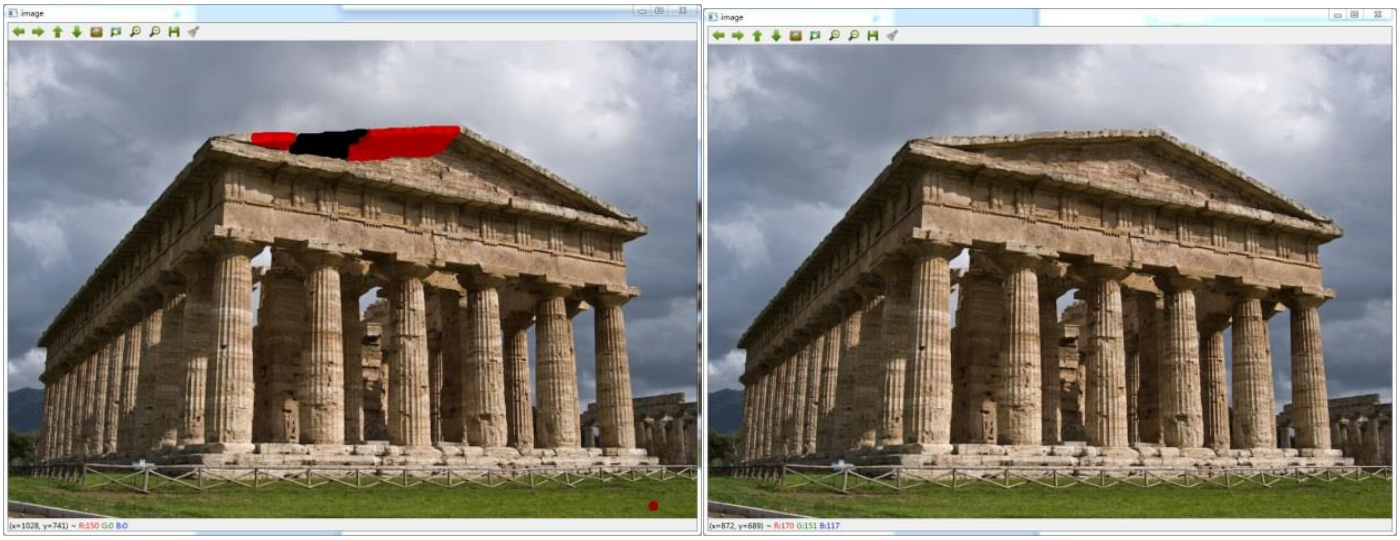
第二步



第三步



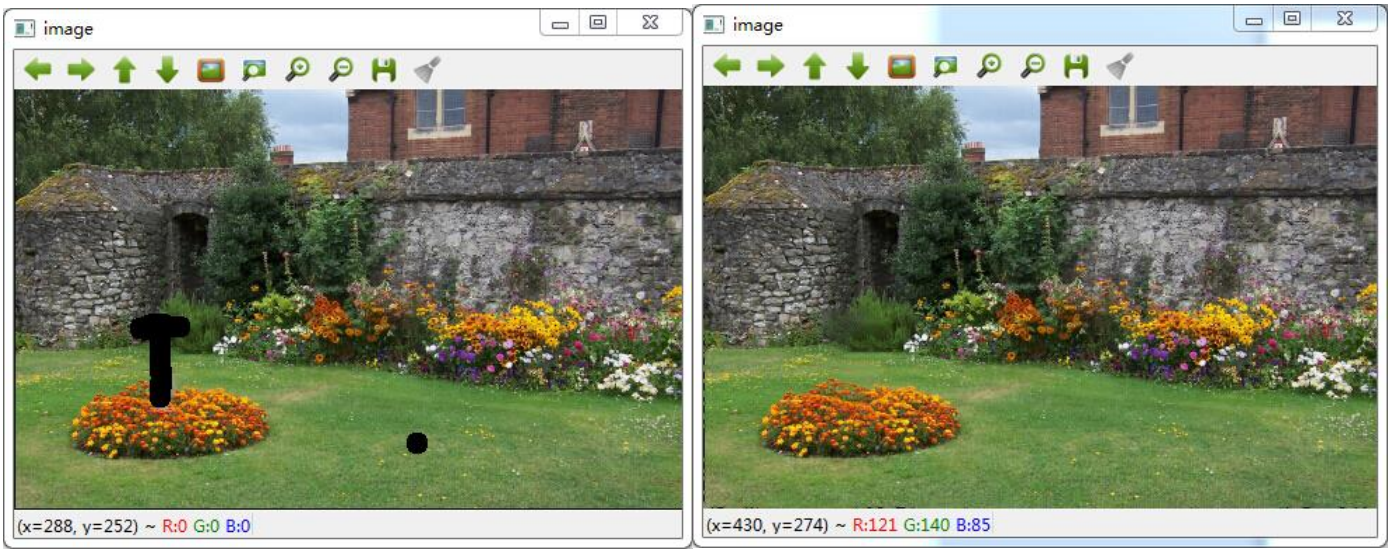
第四步



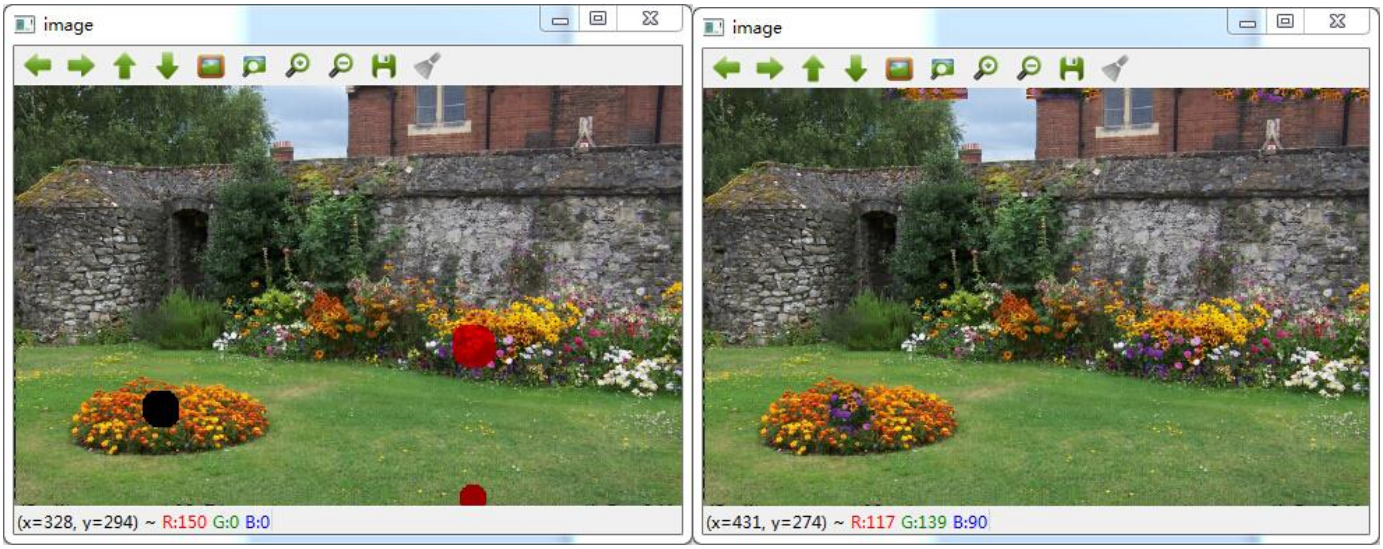
第五步

例子 3：

原图

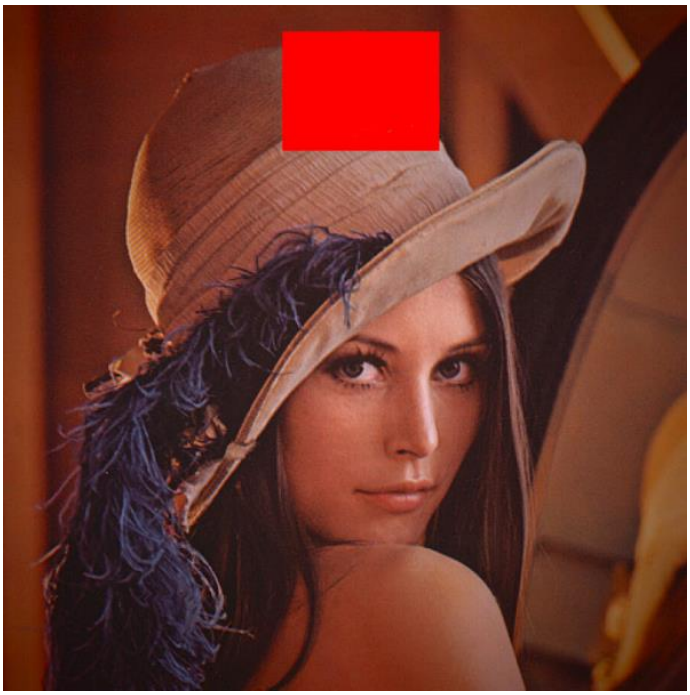


去除石台



将红色区域的花补到洞中

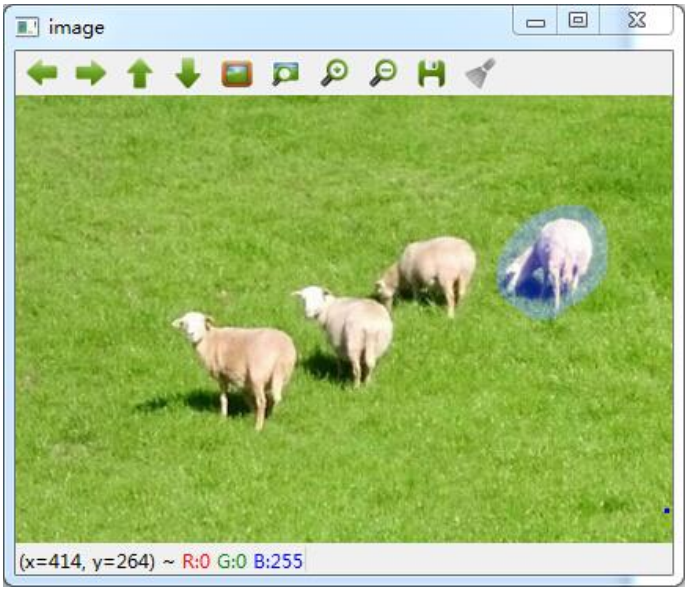
其他结果：利用到了 fill hole in restrict



原图

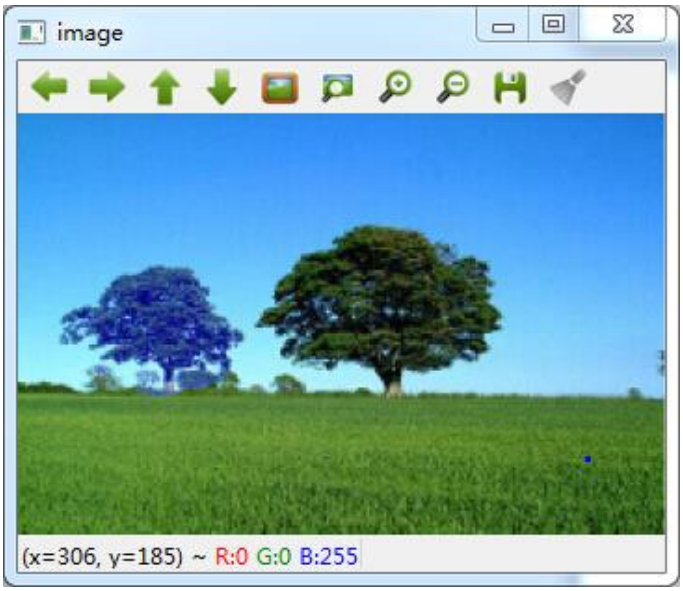
修复后

四、 patch copy 结果

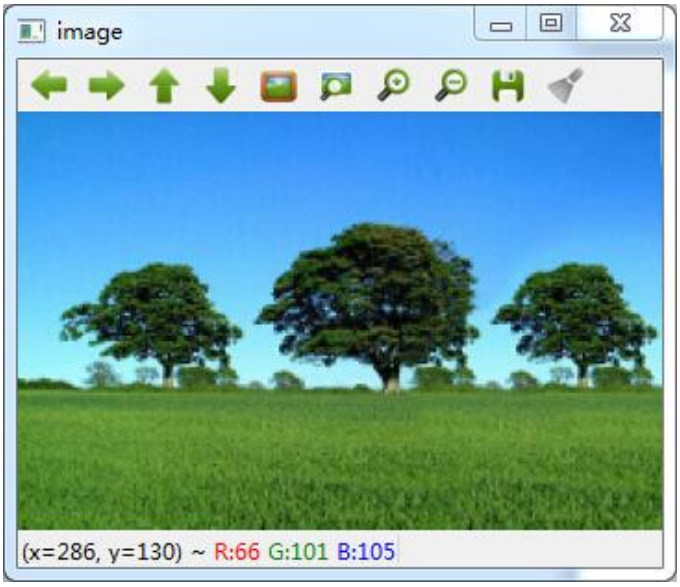


原图

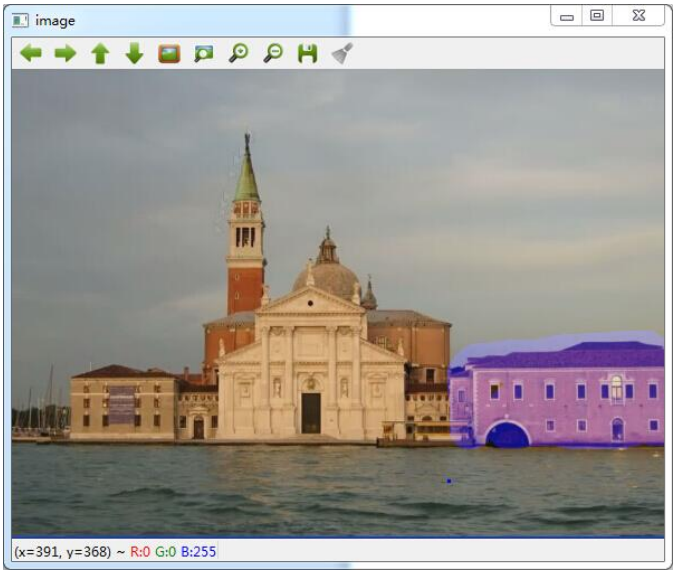
结果



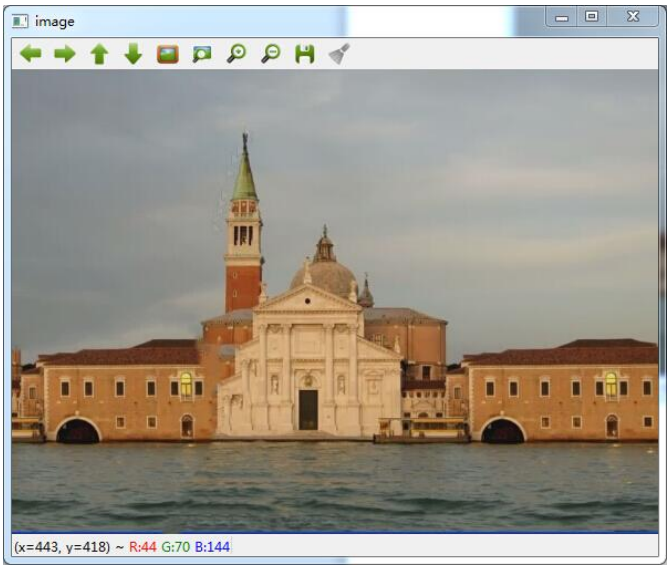
原图



结果



原图



结果