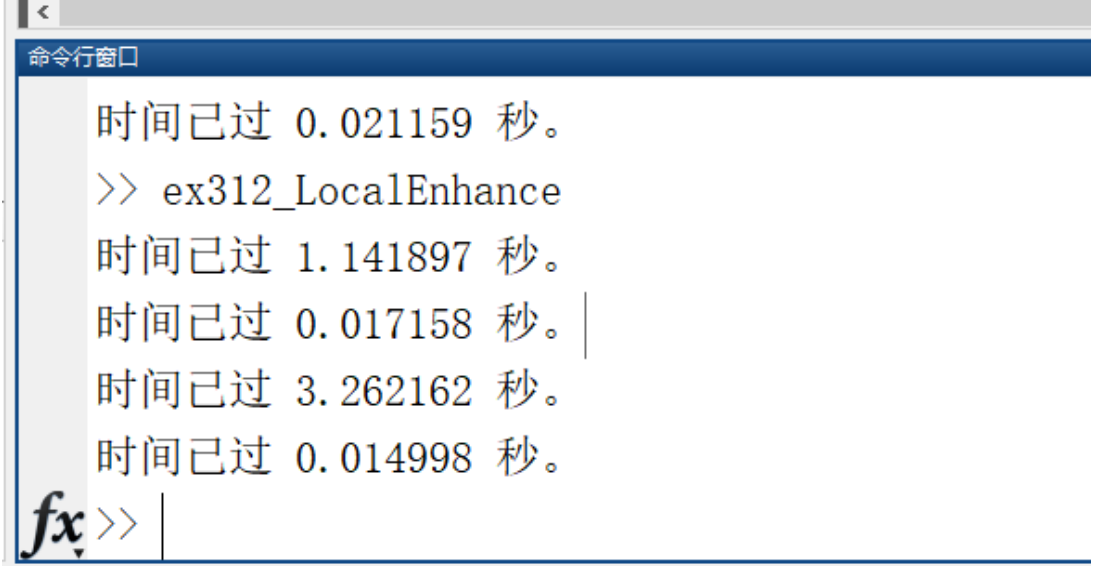


题目一：

为比较好效果，将原代码里 fun2 由标准差改为方差了。（因为题目要求局部方差）

以下是两组计时和对应结果：

```
tic;mean_local = nfilter(I,[31 31],fun1);toc;  
tic;mean_local2 = fast_local_mean(I,15);toc;  
tic;std_local = nfilter(I,[15 15],fun2);toc;  
tic;var_local = fast_local_var(I,7);toc;
```

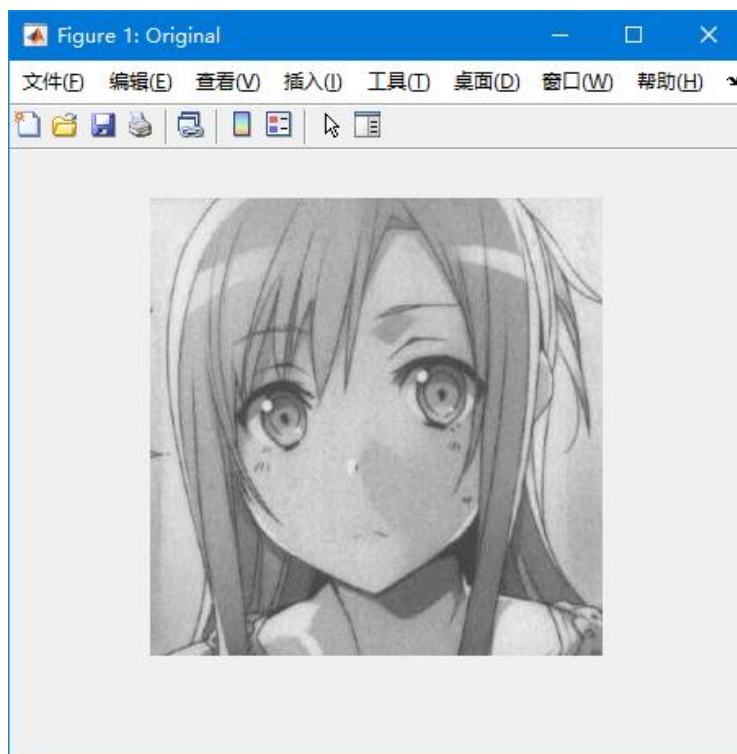


命令窗口

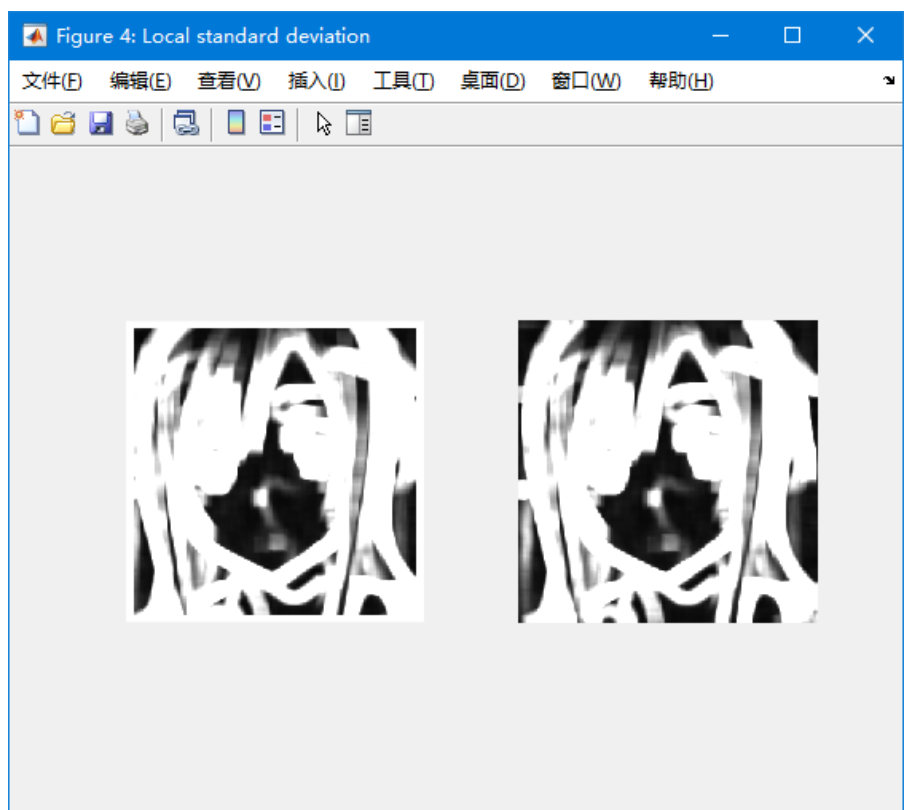
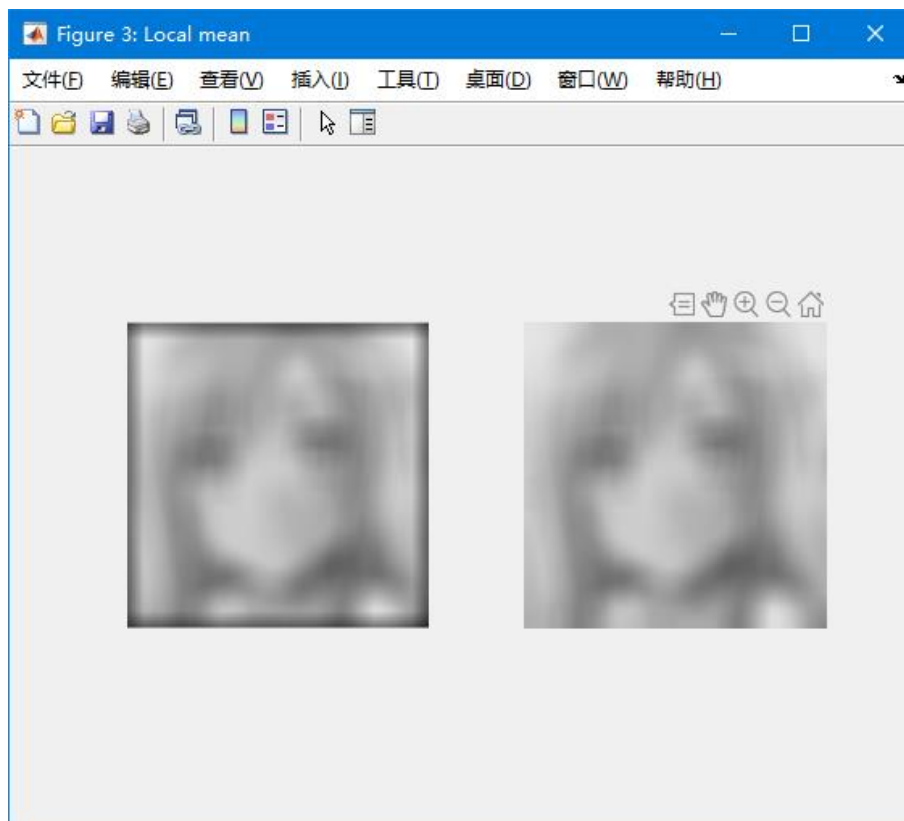
```
时间已过 0.021159 秒。  
>> ex312_LocalEnhance  
时间已过 1.141897 秒。  
时间已过 0.017158 秒。  
时间已过 3.262162 秒。  
时间已过 0.014998 秒。  
fx>>
```

可明显看出我写的快速算法远远快于 nfilter。

原图：



下面是用 nfilter 和我的快速算法处理后出来的图：



可以看到除边界处以外两张图效果相同。

且我的算法边缘处比用 `nlfilter` 更好，我在填充图像时方法选的 `symmetric`。

算法设计与函数讲解：

imgcui 是计算矩阵的积分图，在行和列方向上各 cumsum 一次就可实现。

fast\_local\_sum 是计算  $2r+1$  的局部图像的灰度和。返回矩阵 I 大小和 img 一样，I(x,y) 为以 (x,y) 为中心的矩形的灰度和。一个矩形区域的灰度和可由 imgcui 四个角的加减实现。

fast\_local\_mean 计算局部均值：直接 fast\_local\_sum(img,r) 后除以像素数 N 就可以了。

fast\_local\_var 计算局部方差：

$$Var = \frac{1}{n} \sum x_i^2 - \left( \frac{\sum x_i}{n} \right)^2$$

即，可用 img 每个像素灰度平方的局部均值和 img 的局部均值可计算出。

四个函数都写在 ex312\_LocalEnhance.m 中。

题目二：

整体思路：首先将整个大图分成很多小块，标出 5 个瓶子的中心。将原图复制一个名为 I，对其进行  $r=25$  的均值滤波，然后对选择的中心点 (m,n) 计算其它小块和它的距离，距离小于 6 的，则将 r 取为 3 倍的距离对该小块进行局部均值，便实现了从中心点开始向外模糊程度逐渐上升。

遇到的问题和解决：一开始就是直接分块然后根据距离进行模糊，但是发现远处的小方块 r 太大了，虽然内部确实模糊，但是块与块间边界处看着很明显，效果不好，所以想着到开始出现边界线的距离以内的就渐变得模糊，以外的就靠最初整体的那个模糊来实现。

不足：远离焦点处的模糊程度不够。

为了不出现小方块边界，渐变模糊就有一个范围，而范围内和外的交界处不能模糊程度差别较为明显，就不能把范围外的模糊程度调太高，但是真实的照片离聚焦点很远的地方的模糊效果会比我外围设的  $r=25$  均值滤波的模糊效果更加明显。

可能的改进方法：重写 fast\_local\_mean(img,r) 为 fast\_local\_mean(img,x1,x2,y1,y2,r) 即对  $\text{img}(x1:x2,y1:y1)$  进行局部均值滤波，这样在滤波前填充时就可得到 img 中  $\text{img}(x1:x2,y1:y1)$  外的像素信息，减弱小方块的边界线效果。

下面是生成的 5 张图的效果，可以清楚的看出焦点的变化，模糊度渐变的效果也不错，但远离焦点处的模糊度不够。





