

数图小作业二

2017011552 陈昭熹

题目——改进均值与标准差滤波算法

原理与实现

本次作业实现了基于两种不同原理的改进均值（标准差）算法，分别适用于不同的图像处理情况。

方法——使用积分图

均值

对图像进行积分操作得到积分图 I ，在复杂度与图像尺寸相当 $O(size)$ 的情况下，对于每一个像素 (i, j) 和 $[a, b]$ 的邻域尺寸，可以通过

$$\frac{I(x + a/2, y + b/2) + I(x - a/2, y - b/2) - I(x, y - b/2) - I(x - a/2, y)}{a \times b}$$

来得到该点在平均滤波意义下的取值。由于提前构造了积分图因此每一个点的均值计算只需 $O(1) \times 4 = O(1)$ 的时间，总体来看整个矩阵求解完毕也只有 $O(4size)$ 的时间，因此是个线性的算法，其速度将非常快。

标准差

由于标准差是建立在均值的基础上推导得来的，因此也可以用上面的方法，只不过需要额外维护一个原图每点平方的积分图。在统计学中标准差定义为：

$$Var(x) = \sqrt{\frac{1}{n-1} \sum (i(x) - u)^2}$$

其中 u 就是上面一节求出来的均值， $i(x)$ 即每一点的像素值。对上面的式子展开并整理化简可以得到下式：

$$Var(x) = \sqrt{\frac{1}{n-1} (\sum i^2(x) - \frac{1}{n} (\sum i(x))^2)}$$

利用上式结合我们维护的关于 $I(x), I^2(x)$ 的积分图，对于这种求和式可以快速的得到结果。

方法二——空间换时间

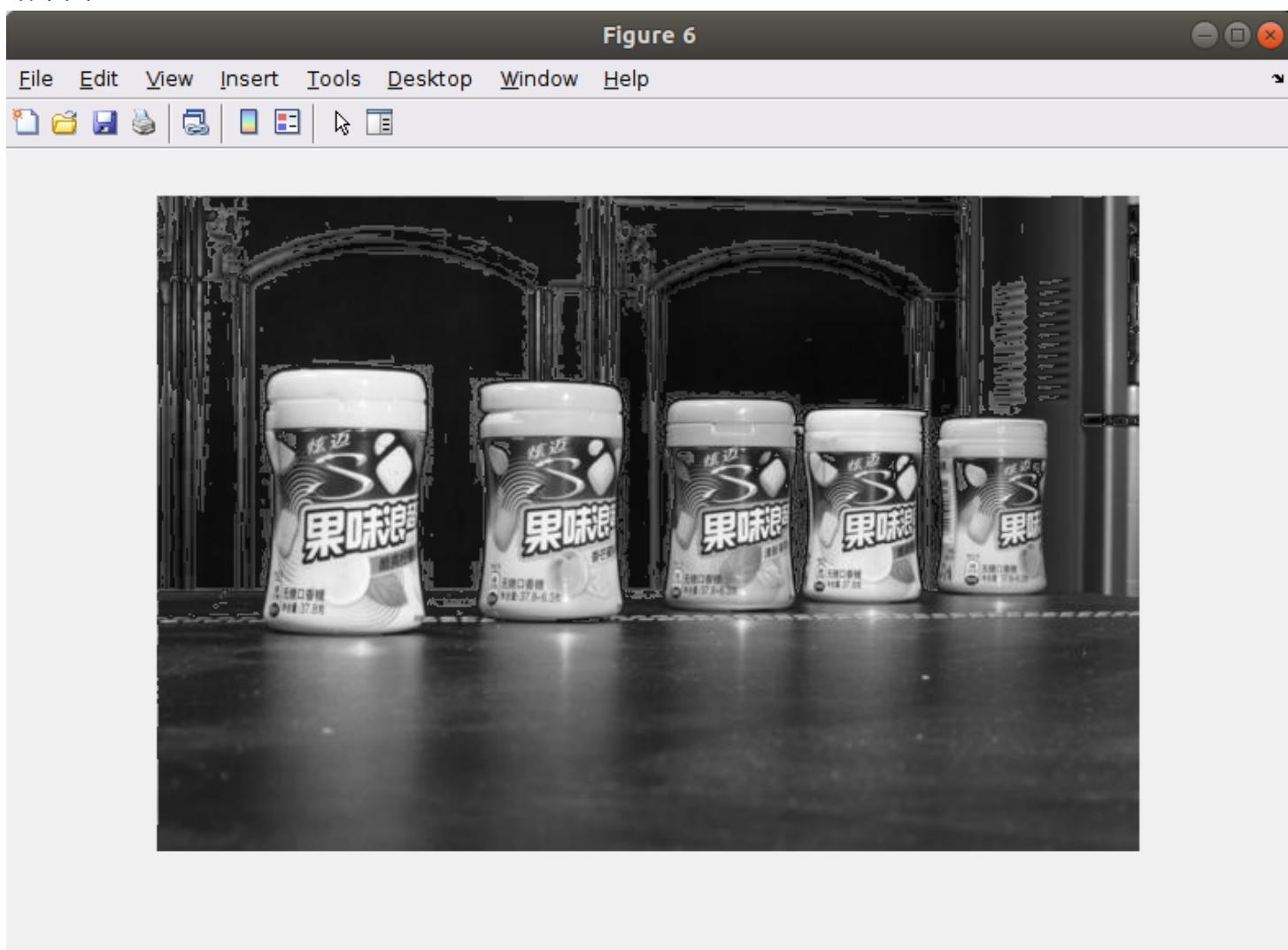
这里更倾向于实现一个**通用的卷积计算法**，而非对于均值和方差进行特定的优化。事实上，在大图像或者是大卷积核的情况下，使用卷积，仅在空域下来实现的滤波器都将会比较慢，因此这一个方法的性能是不能与方法一对比的，但是它还是会优于 `nlfilter`。我采取空间换时间的思路，经过逐步跟踪，我发现 `nlfilter` 的运算慢在卷积核滑动过程中频繁的二维访存操作。我们不一定肯定相应的图像存储在内存中是连续的地址，因此这会导致这种频繁访存让算法速度大大下降。

通过 `im2col` 函数，我针对卷积核的大小来将原始图像转化为列向量，其行数为卷积核的 `numel`，列数为适应卷积运算扩展后的原图 `numel`。换句话讲，就是把原来的二维卷积操作中在原图需要计算卷积的区域的快照转为列向量并存储下来。这其中肯定会有极大地重叠存在。

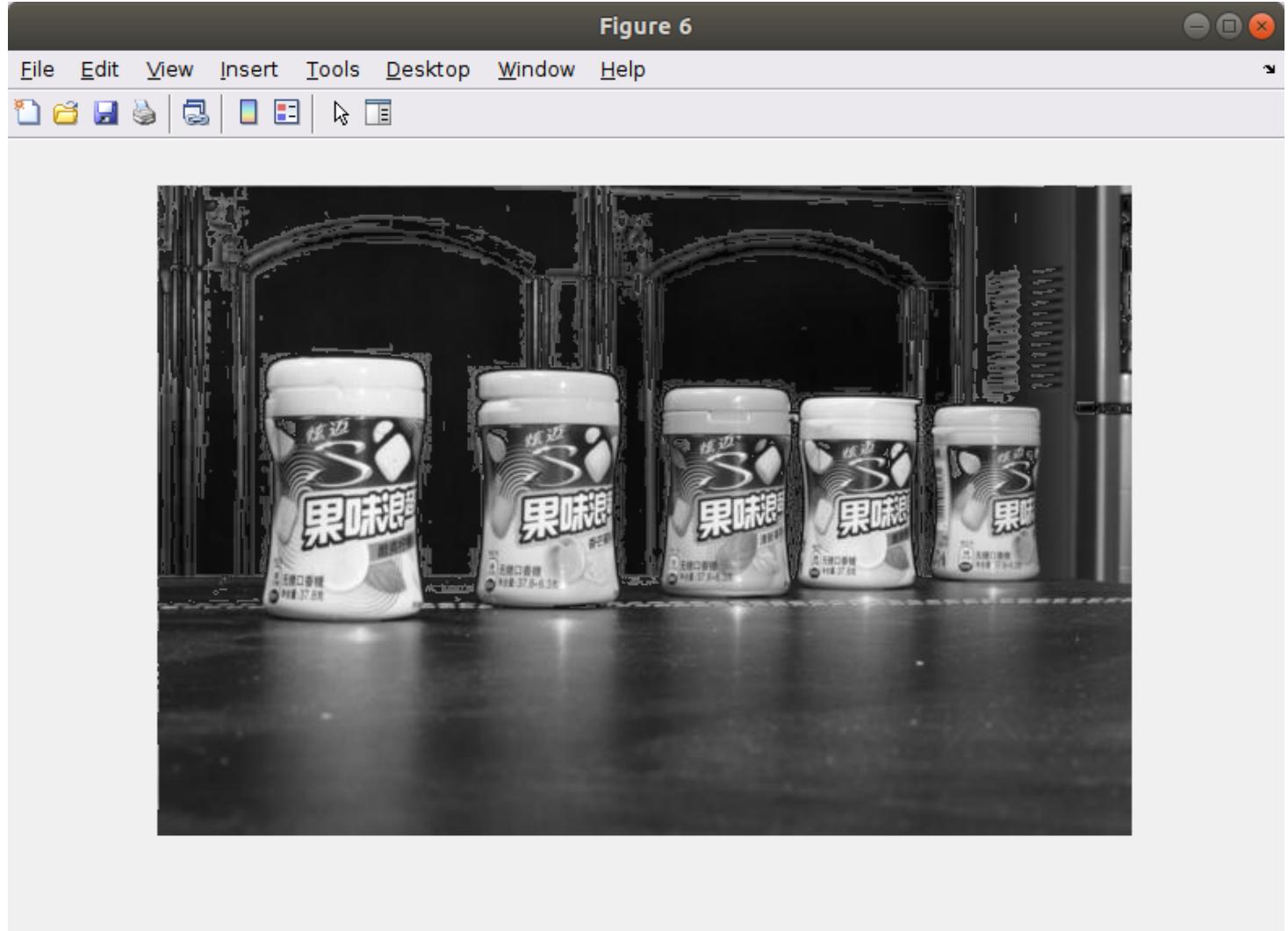
最后通过传入的滤波函数（在本题中就是 `mean2` 和 `std2`），对新建的矩阵每一列进行运算，并放到原图中对应位置。由于 `matlab` 的一些函数是专门为向量化运算设计的，因此这样的计算方式也会比较快。

性能对比

结果图：



原始方法结果图：

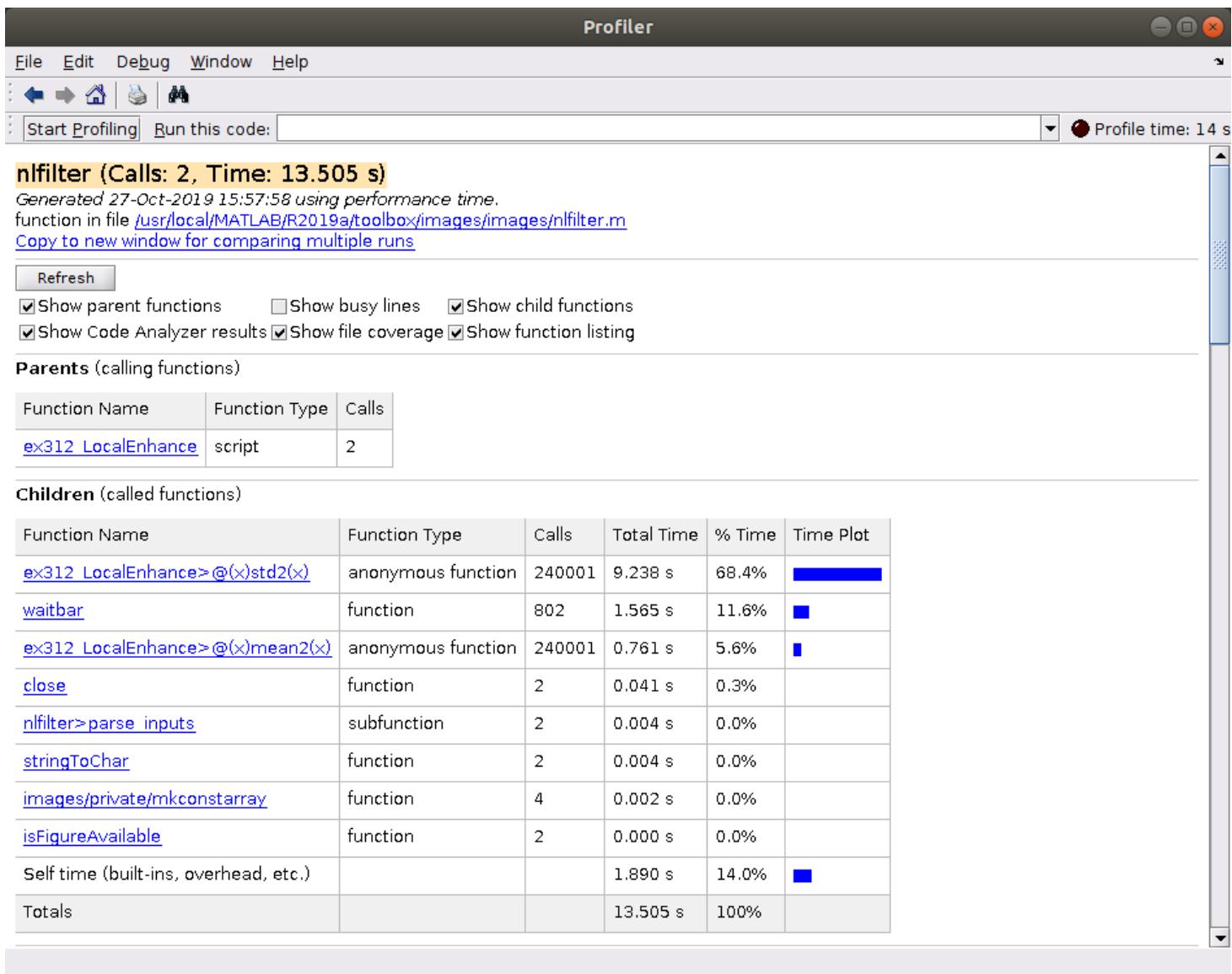


可以看出所实现的方法效果与nlfilter相同。

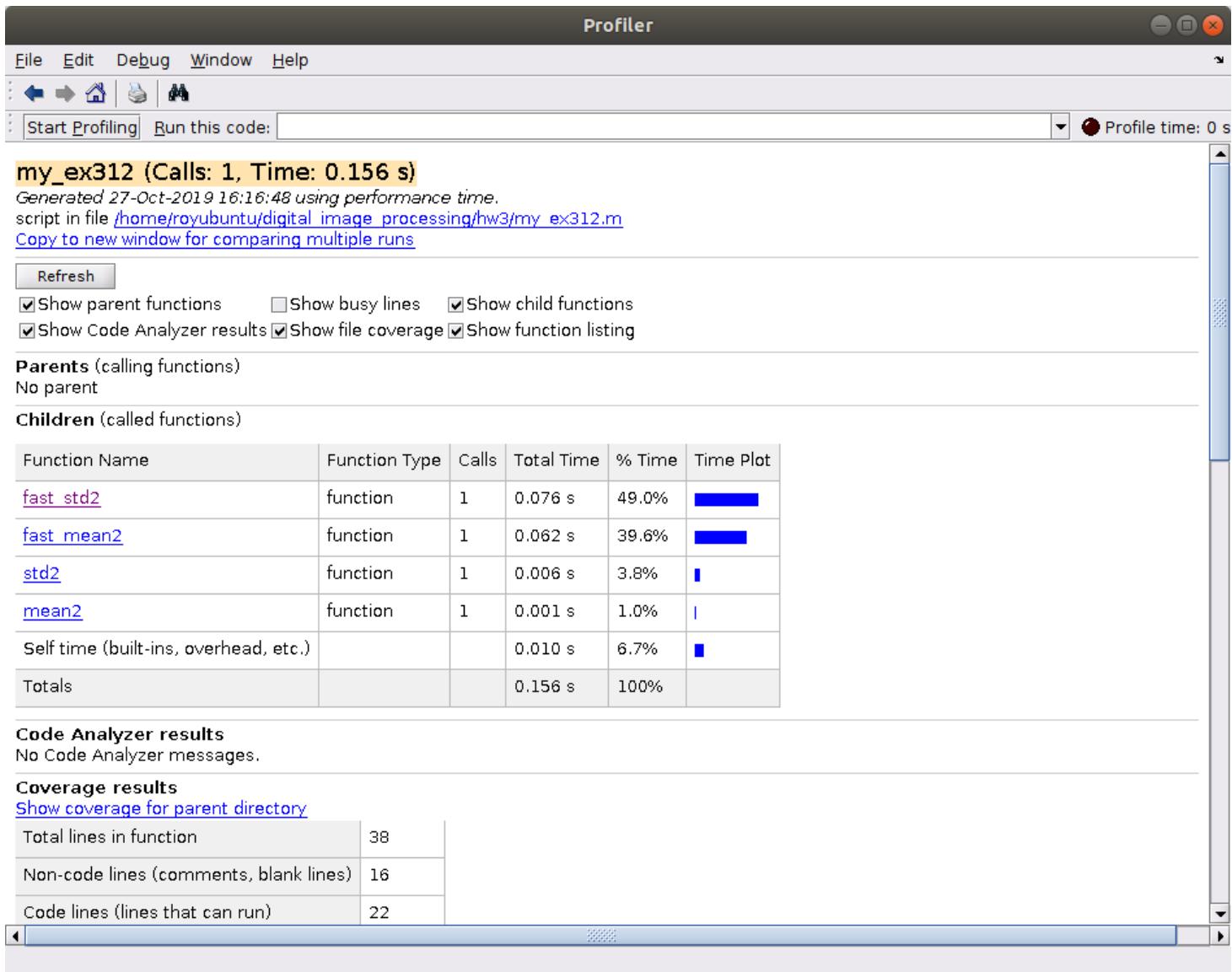
将两种方案与老师在示例代码中使用的非线性滤波器 nlfilter 进行时间效率对比，结果如下：

图像600x400 卷积核尺寸3x3

nlfilter

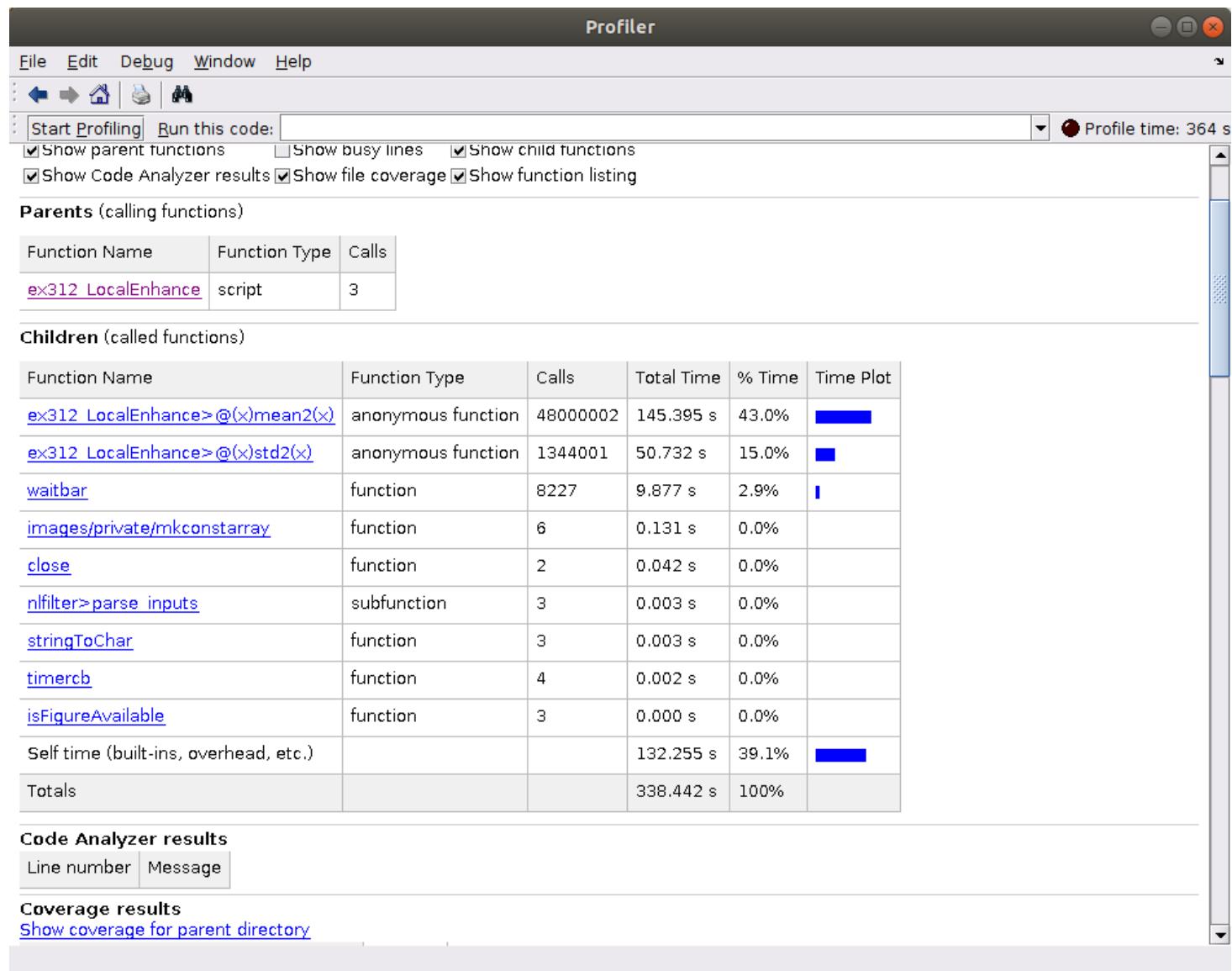


本文方法

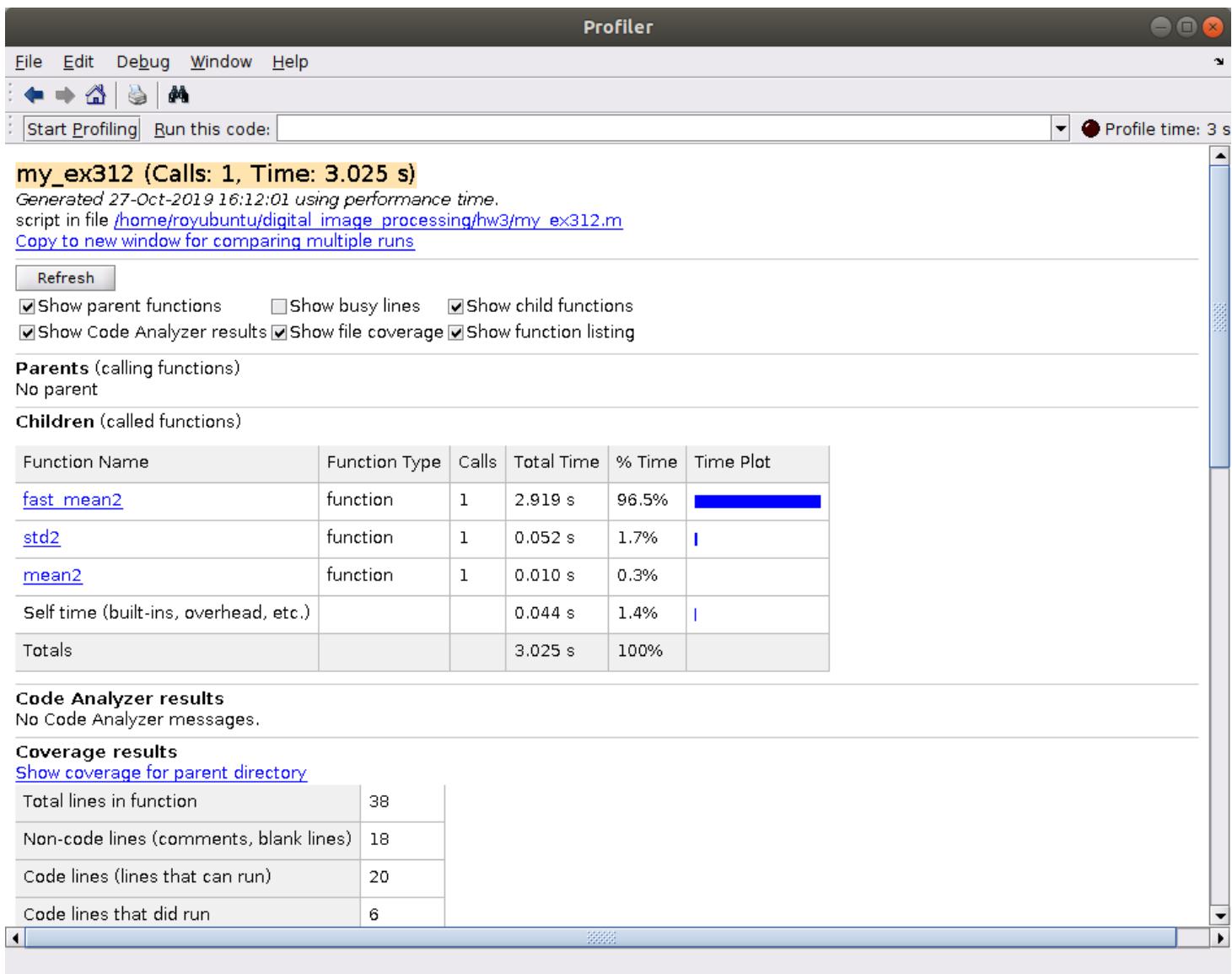


图像6000x4000 卷积核尺寸3x3

nfilter

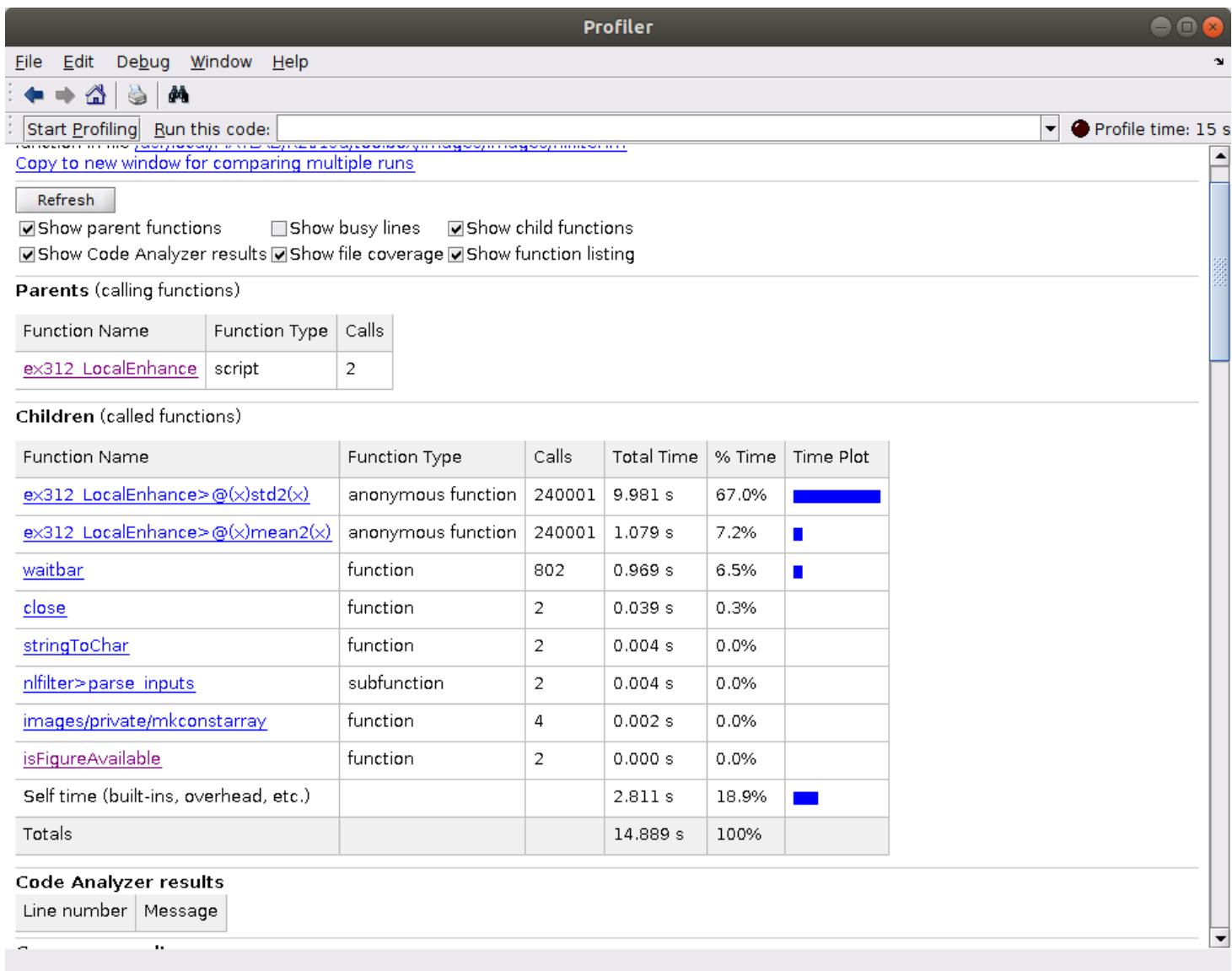


本文方法

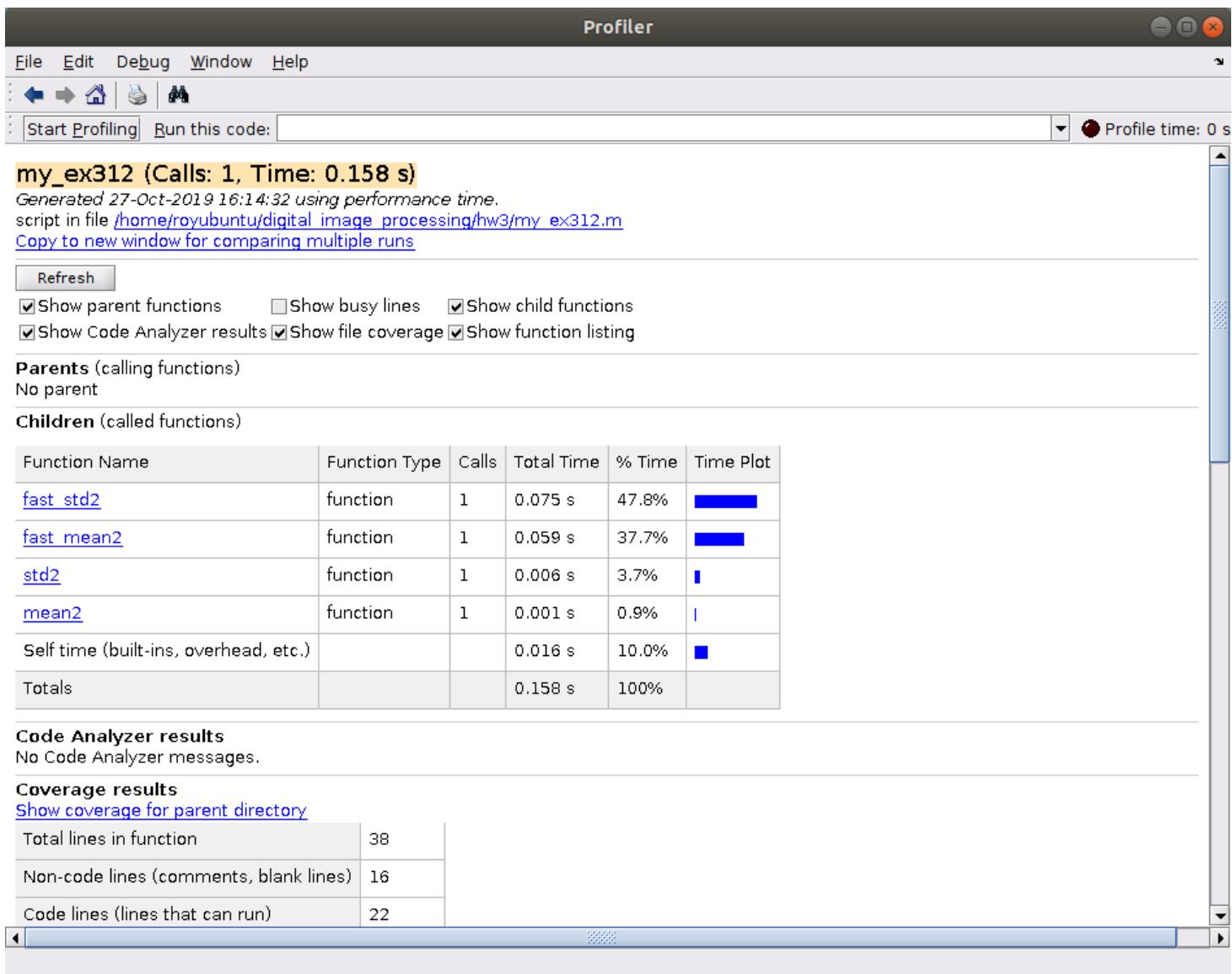


图像600x400 卷积核尺寸32x32

nfilter



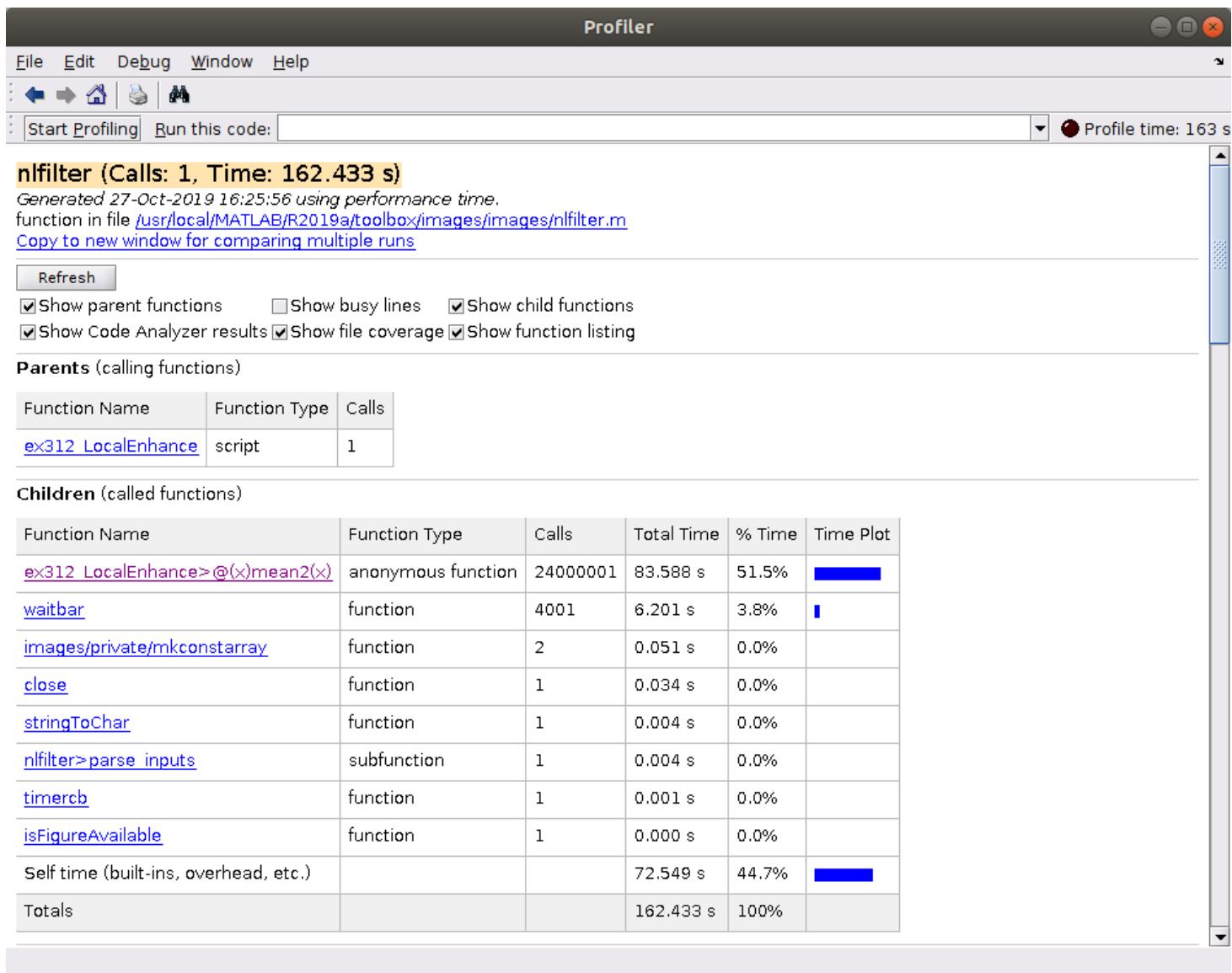
本文方法



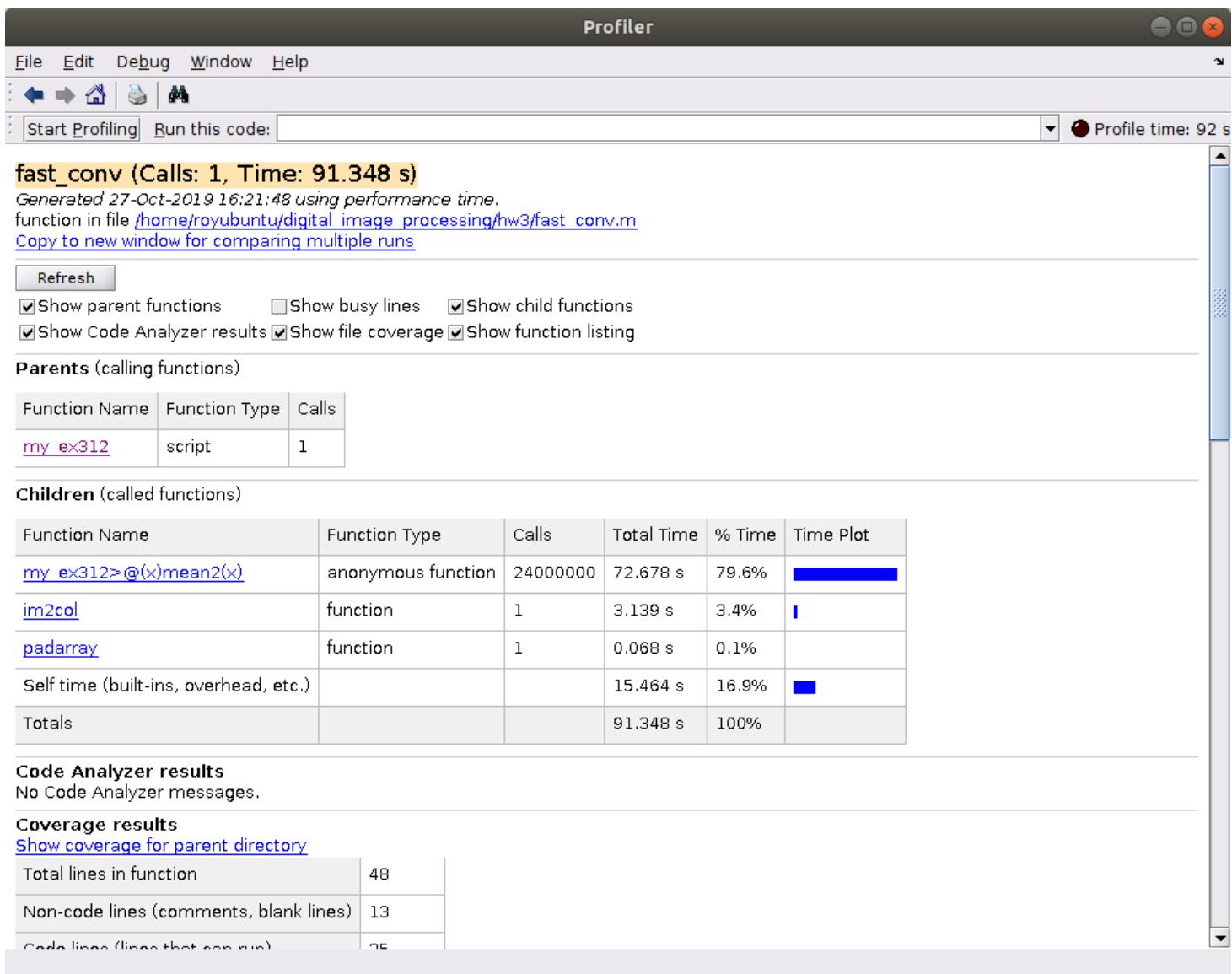
方法二性能 图片6000x4000 卷积核3x3

方法二在大图片 大卷积核场景下表现比nlfilter好，在于其空间换时间的策略需要让时间复杂度达到一定的阈值才能让这部分加速的空间代价得到代偿。

原始方法：



空间换时间后：



题目二——模拟景深

原理

为了通过滤波算法来实现模拟景深的效果，首先应当对相机产生景深的原理有所了解。通过光圈调整使得焦平面缩小，从而让特定景物有清晰地聚焦效果而背景模糊。因此我们也需要通过算法来模拟这些元素的效果。

首先，应当确定图像中焦平面的位置和大小，这样就能将前景和背景区分开。而后将背景通过滤波虚化，将前景图像增强，即可得到初步的效果。

仔细观察给出的示例照片，不难发现，图像的模糊程度与其深度是成正相关的，而在二维平面上就直观的体现为与焦平面中心的距离成正相关，距离越远越模糊，越近越清晰。可以注意到，在聚焦的瓶子旁侧的瓶子一般是仍旧能看清字和花样的。为了实现这样的效果，还需要将前景到背景的过渡部分通过渐变的模糊滤波来实现平滑的效果。

实现

全程无手工编辑，不使用示例图片做辅助图片。

焦平面选取

使用需要聚焦的瓶子的中心坐标来作为焦平面中心，相应的坐标是通过模糊定位来得到的。焦平面半径在实现过程中使用了scale过后的（为了与非线性函数来匹配，后文会提到），因此其具体数值在此无参考价值，选取原则是能够使得焦平面覆盖目标物体，同时也能覆盖到一些相邻的两个物体的紧邻边。

滤波器实现

全过程使用均值滤波和高斯滤波。

- **均值滤波** 使用第一题所实现的快速均值算法，能够在很短的时间内完成大量的均值操作。
- **高斯滤波** 采用快速傅里叶变换进行计算。由于原图尺寸极大(4000x6000)，同时我还需要使用窗尺寸较大的卷积核(30~150)，因此空域卷积的代价是极大的， $O(n^2)$ 。通过FFT，能够将时间复杂度降到 $O(n \log n)$ ，极大地提高了处理速度。（虽然不要求是实时算法，但是直接空域卷积真的处理的太慢了，一张图一次高斯滤波我就没跑完过）

值得注意的是，我也尝试了在空域中实现快速卷积算法。二维高斯分布定义为：

$$Gauss(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

注意到这样的表达形式实际上说明了，通过它实现的滤波器具有可分离的特性，利用其**可分离性**可以将空域下二维卷积操作转化为两个一维卷积操作，从而提高计算速度。这部分功能实现
在 `fast_gauss_filter` 中，但无奈还是没有FFT来得快，所以最后没有使用。

背景模糊

背景模糊主要分为三步：

- 通过边缘检测和灰度增强来获得5个瓶子所在的区域。
- 通过一定阈值的二值化来生成背景区域的mask。
- 在生成的mask上做大窗的高斯模糊，从而实现背景模糊。

过渡模糊

过渡模糊即为了实现不同深度的不同模糊效果。通过计算每一个像素点与焦平面中心的距离 $dist$ ，来作为滤波器的窗口大小参数来实现不同模糊效果。上文提出过，该距离与模糊程度是正相关，但非线性。而直观上我们也能看到，距离聚焦点附近的瓶子一般都还能看清字和花样，但是再远一下清晰度就急剧下降，最后在较远的地方模糊程度也就近似差不多了。从这样的定性分析上，我们可以总结出距离与模糊

程度存在一个非线性关系，而这样的关系需要中距离部分梯度极大。基于这样的结论，我想到了 sigmoid 函数来拟合这样的关系。Sigmoid 函数定义如下：

$$\text{sigmoid}(x, \omega) = \frac{1}{1 + e^{-\omega x}}$$

其中 ω 用来控制其梯度，而 x 就是由 $dist$ 生成的，具体关系如下：

$$x = dist/5 - 2$$

最终实现模糊效果的不同是通过控制均值邻域窗口大小来实现的，窗口大小为：

$$window = 100 \times [\text{floor}(\text{sigmoid}(x, 2)), \text{floor}(\text{sigmoid}(x, 2))]$$

在实际使用中，上面的实现过程会在参数控制不佳的情况下，图片中出现明显的环状条纹。为了避免这一情况，在距离上增加了一个高斯噪声

$$dist = dist + (1 + \text{randn}(1, 1)) \times \frac{dist}{20}$$

从而模糊取整函数的边缘，将每一段平滑的连接了起来。

效果图











算法的不足

尽管花费了很长时间尝试完美算法，但无奈当前算法仍存在一些问题。

- 无法实现焦外二直线性效果。示例图片中可以发现背景中的模糊会在一些亮点出产生圆形光晕或者光斑（即焦外二直线性），这是由于镜头的结构原因造成的。但是在现在算法中无法实现这样的效果。
- 背景模糊不够充分。示例图片中背景上的亮斑和光是完全被模糊掉了，完全看不到背景的小反光。但是在当前算法中这样的局部大梯度特征是无法被这两个滤波器所抹掉的。也许需要在背景上做一些其他的滤波处理，来使得背景模糊更加彻底和逼真。
- 细小局部有一些噪声。由于添加了噪声来避免明显的分层现象，但是在有些地方，特别是均值滤波窗口还不是很大的地方会有一点点噪声而非模糊滤波的痕迹。也许对于模糊的实现上应当使用连续的参数控制，而不应当使用窗口大小这一离散的参数控制。