南京大学 NANJING UNIVERSITY
电子科学与工程学院
School of Electronic Science and Engineering

基于GPU的非局部均值算法的并行优化设计

Parallel optimization design of GPU-based non-local means algorithms

指导老师：李昀　　教授

答辩人：苗金成

2024/5/29

目录

非局部均值算法（non-local means, **NLM**）属于空间域非局部去噪算法。

空间去噪方法本质就是平均，串行NLM算法对每一个像素做一次滤波，每个像素周围取一个**搜索窗口**。利用搜索窗口内的点做平均，不过每个点的权重取决于这个点与当前待滤波点的邻域相似程度。计算两个邻域（**模板窗口**）的MSE获得权重。因此对于二维图片而言，串行算法核心是6重循环（2重图片，2重搜索窗口，2重模板窗口）

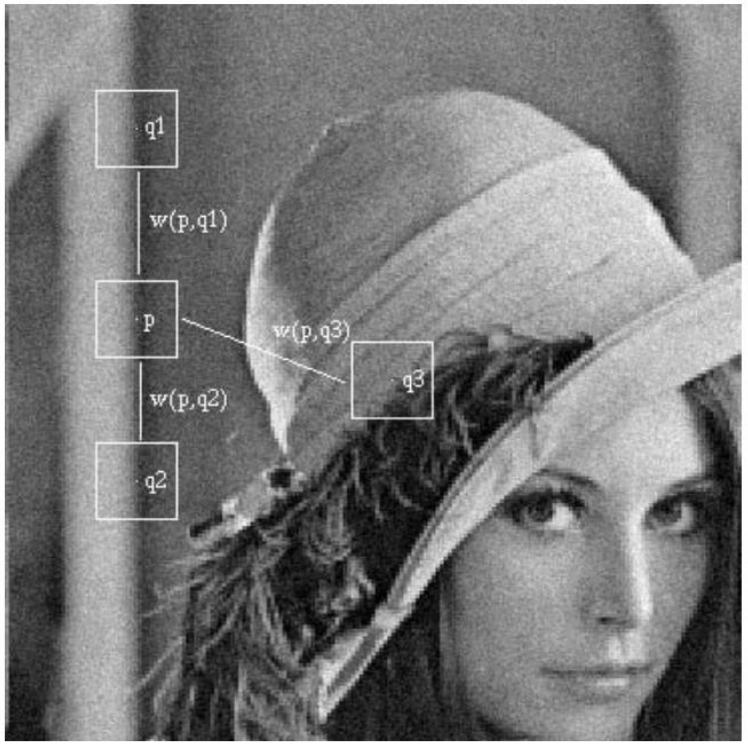


Figure 1. Scheme of NL-means strategy. Similar pixel neighborhoods give a large weight, w(p,q1) and w(p,q2), while much different neighborhoods give a small weight w(p,q3).

给定一个噪声图像$\boldsymbol{v}=\{\boldsymbol{v_i}|\boldsymbol{i}\in\boldsymbol{\Omega}\},\boldsymbol{\Omega}\subset\mathbf{R^2}$，滤波输出图像在i处的像素值$\boldsymbol{u_i}$为：

$$u_i=\sum_{j\in I}w\left(i,j\right)v_j$$

，其中***I***是以点i为中心的搜索窗口

权重表达式如下：

$$w\left(i,j\right)=\frac{1}{Z\left(i\right)}exp\left(-\frac{\left|\left|v_{N_i}-v_{N_j}\right|\right|_{2,a}^2}{h^2}\right)$$

Z(i)起归一化作用，保证权重总和为1:

$$Z\left(i\right)=\sum_j w\left(i,j\right)$$

Algorithm 1 Processing a pixel with NLM

```
for i := 0 To searchSize do
  for j := 0 To searchSize do
    for k := 0 To nbSize do
      for t := 0 To nbSize do
        diff ← pixel1 − pixel2
        weight ← weight + diff ∗ diff
      end for
    end for
    weight ← exp(−weight/(h ∗ h))
    weights ← weights + weight
    sum ← sum + weight ∗ pixel
  end for
end for
```

NLM 算法去噪效果显著，但是缺陷明显。计算量复杂度太大，运行的时间明显慢于其他算法，如下表[1]所示，

Method	2048*1361	800*600	256*256
TF	0.930	0.171	0.011
NLM	56.73	7.36	1.07
INLM	11.426	1.959	0.20
BM3D	0.937	0.125	0.017
K-SVD	1.168	1.729	1.568
LPG-PCA	160.35	21.74	3.12

代表性去噪算法的相对计算时间（单位:分钟）[1]

[1] Ling Shao, Ruomei Yan, Xuelong Li, and Yan Liu. From Heuristic Optimization to Dictionary Learning: A Review and Comprehensive Comparison of Image Denoising Algorithms. IEEE TRANSACTIONS ON CYBERNETICS, VOL. 44, NO. 7, JULY 2014

NVIDIA GPU (Graphics Processing Unit)架构是围绕一个可扩展的多线程流多处理器(SM，Streaming Multiprocessor)阵列构建的。

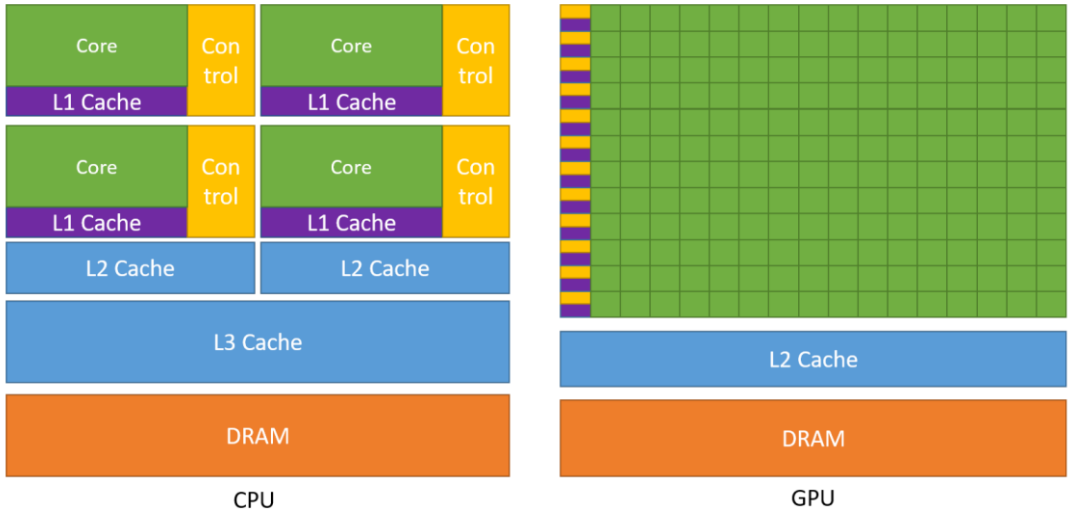


Figure 2： *The GPU Devotes More Transistors to Data Processing*

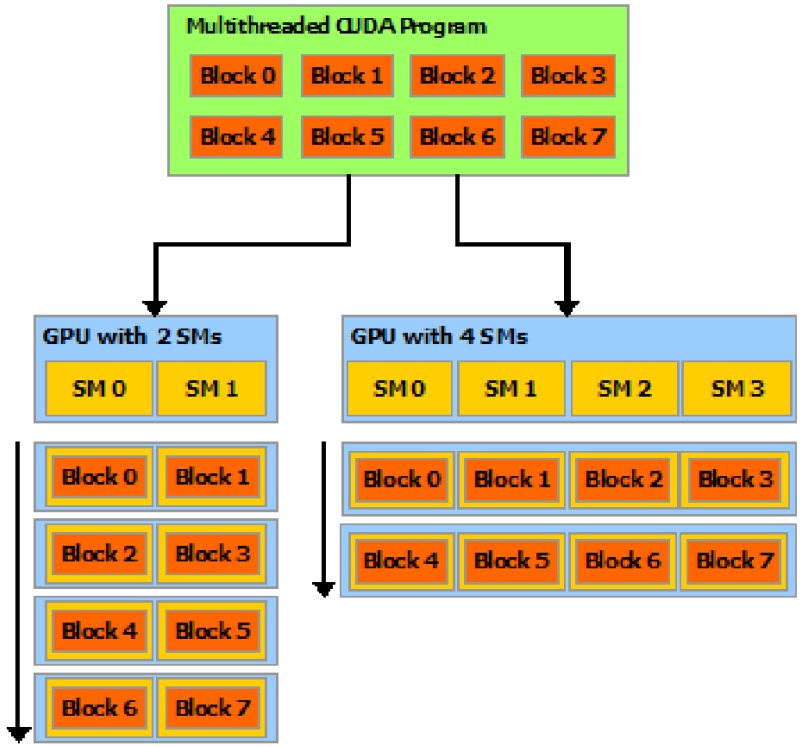


Figure 3： *Automatic Scalability*

一个SM中包含:

- 流处理器(Streaming Processor, SP)：SP是最基本的数据处理单元，支持整数，浮点，逻辑等运算操作，可以做多线程并行处理，也能进行单线程处理。
- 特殊函数单元（SFU，special function units）：用来执行超越函数和插值函数计算。
- LD/ST单元：执行指令存储、加载操作。
- 寄存器文件：管理线程束调度

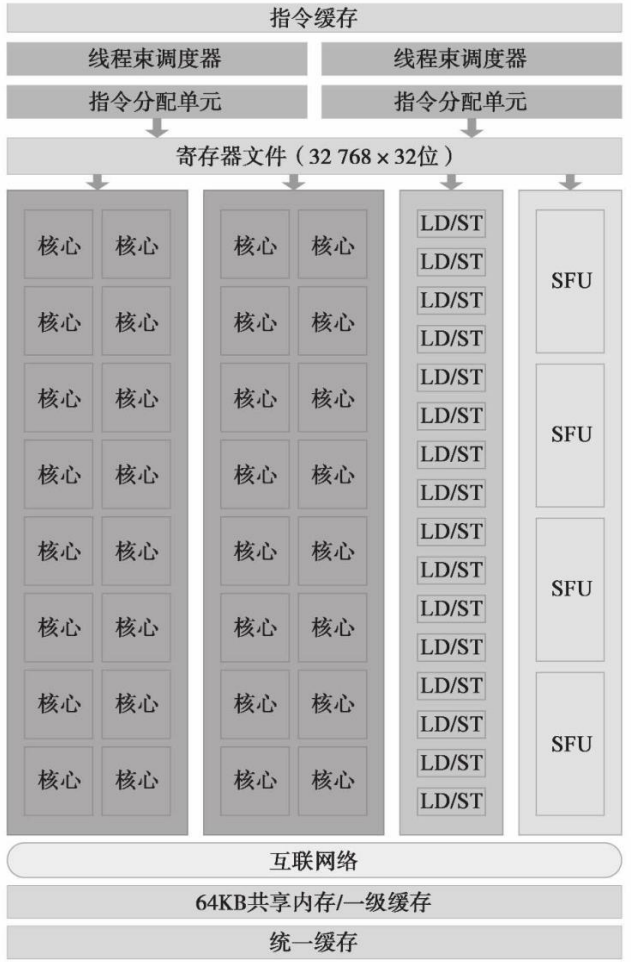
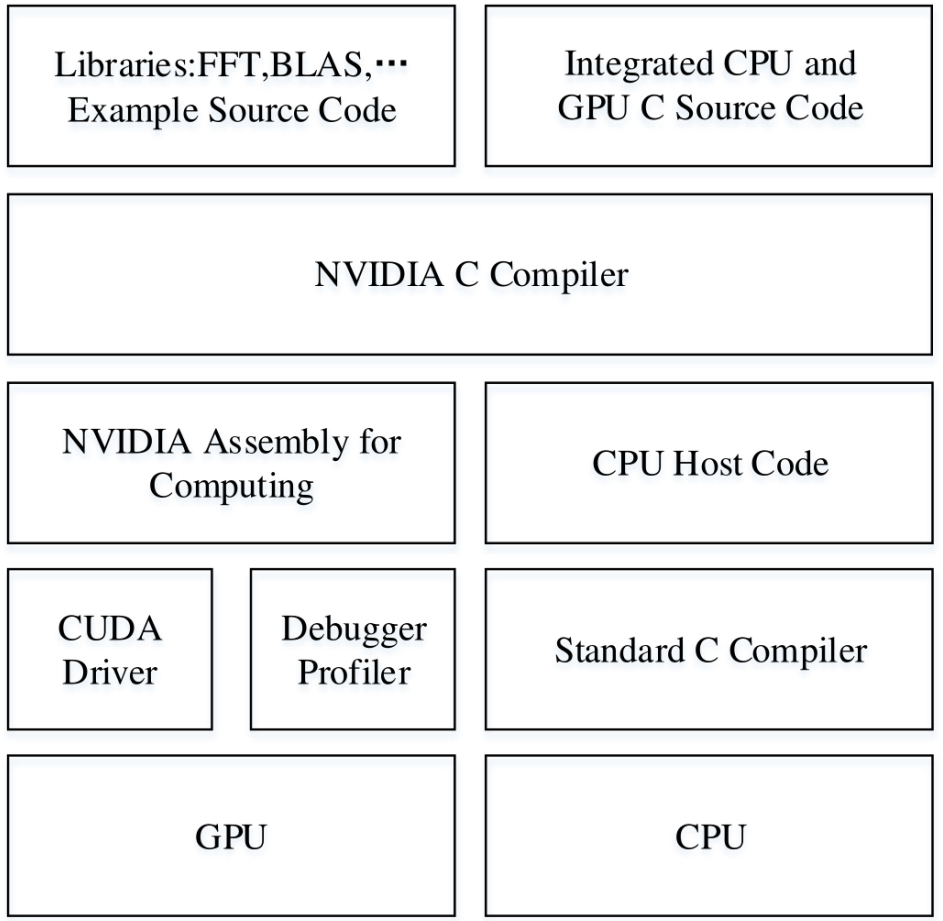


Figure 4： architecture of SM

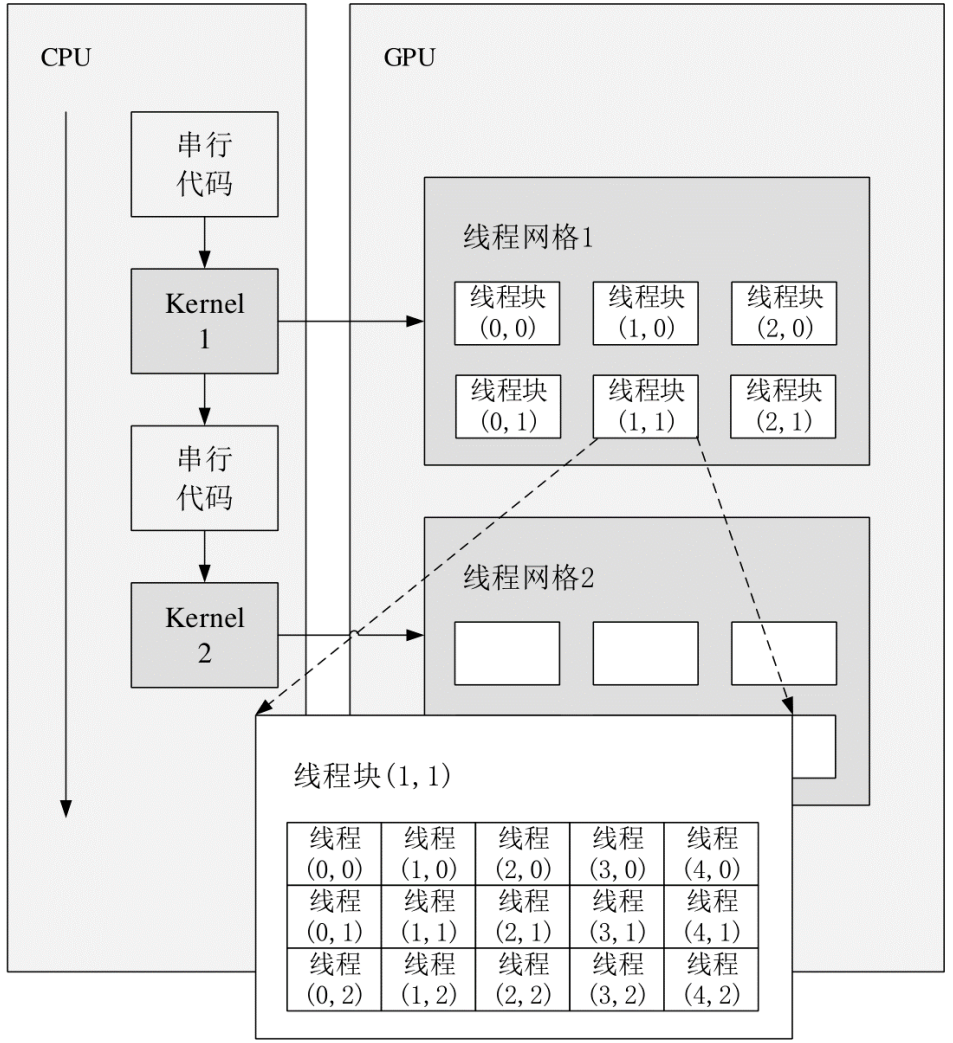
2006年11月，英伟达推出了CUDA(Compute Unified Device Architecture，通用并行计算架构)，专门用来在旗下的GPU上解决复杂的并行计算问题。

在 CUDA 中，将 GPU 上执行的函数成为内核函数（Kernel），一个内核函数对应一个并行化任务，内核函数的定义如下：

KernalName<<<grid, block>>>(A, B…)



网格和块的维度由gridDim和blockDim两个变量指定，它们是dim3类型的变量，基于uint3定义，可以通过block.x、block.y、block.z分别获得三个维度的大小。右图中的网格组织成线程块的二维数组形式，线程块也组织成线程的二维形式。blockIdx是线程块在网格内的索引，threadIdx是块内的线程索引，这两个变量也是基于uint3定义的，可以通过x, y, z三个字段获得坐标。

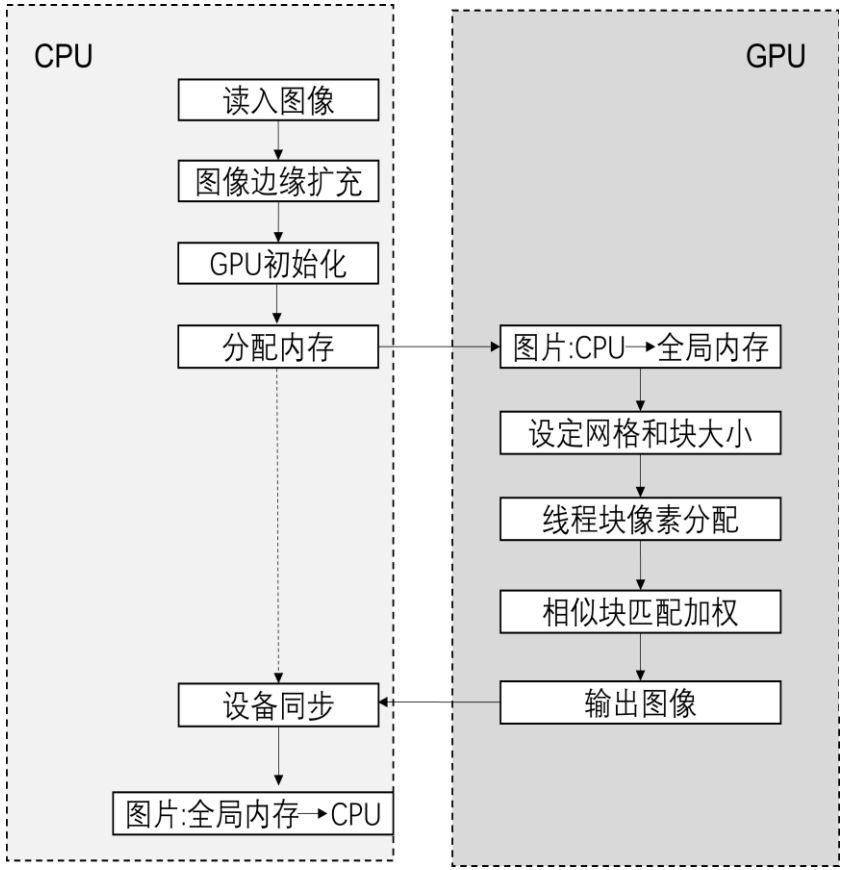


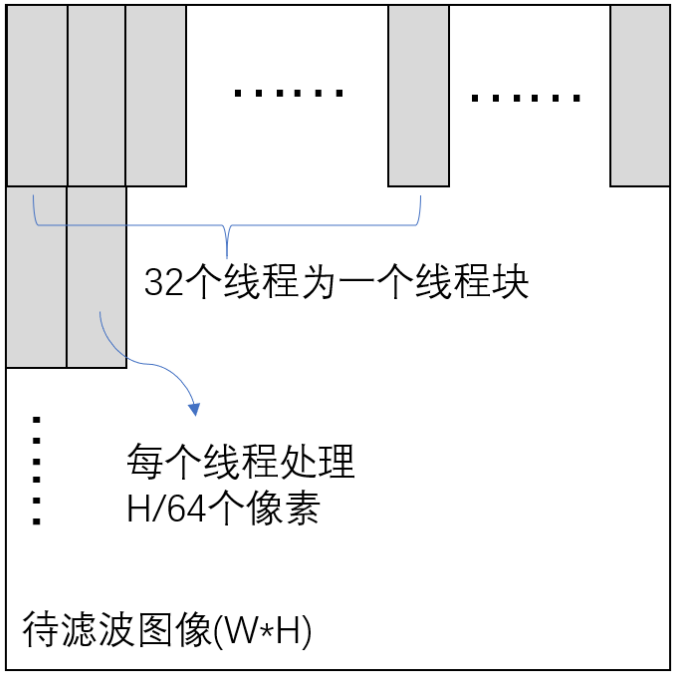
02 算法设计

全局内存实现

共享内存实现

结合CUDA的线程模型，可以把图像分为多个小块，分别进行滤波，也就是取二维的网格和块。这些图像块之间天然地不存在计算依赖，而且每个像素的去噪处理过程是类似的。



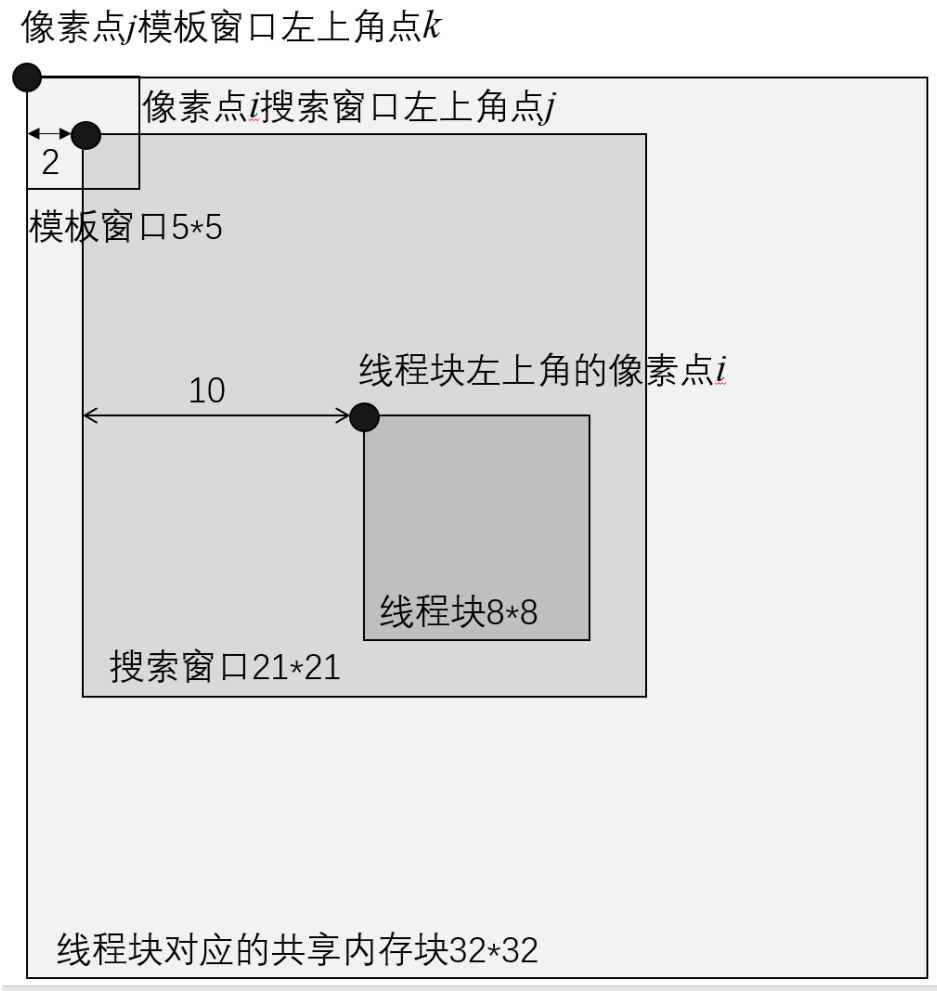


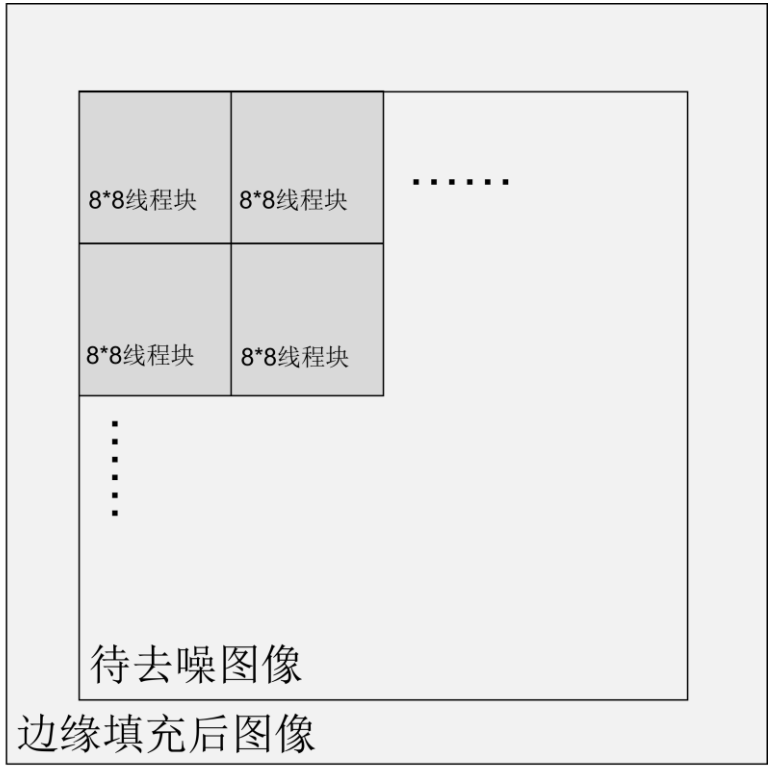
全局内存的并行处理相当于图片遍历的**循环展开**

全局内存实现

共享内存实现

在核函数启动时，每个线程都先复制4*4的信息，从共享内存左上角(点k)开始。线程块划分后，整张图片就被分为了8*8的小块，每个线程块处理其中的64个像素，直至所有像素都处理完毕。





- 使用__syncthreads()函数进行同步。__syncthreads()是一种显式同步，发挥一个“障碍点”的作用。
- 全局内存实现和共享内存实现之前都要先“扩充”图片，保证边缘像素的滤波。

```
//copy global mem to shared mem
for (int i = 4 * threadIdx.y; i < 4 * threadIdx.y + itra; i++) {
    for (int j = 4 * threadIdx.x; j < 4 * threadIdx.x + itra; j++) {
        smem[j + i * share_size] = data[lx + j + (i + ly) * W];
    }
}

__syncthreads();

float sum = 0.0f;
float weights = 0.0f;
float weight = 0.0f;

for (int i = 0; i < sr_size; i++) {
    for (int j = 0; j < sr_size; j++) {
        for(int k = 0; k < nb_size; k++) {
            for(int t = 0;t < nb_size; t++) {
```

部分代码

实验分为两部分，第一部分对去噪效果进行评估，第二部分对算法性能进行评估。

去噪效果测试使用了四张单通道灰度图像，分辨率均为512*512，在去噪前均添加了标准差为10的零均值高斯白噪声。

PSNR是最常用的图像评价指标之一，是信号可能达到的最大值与染噪图像的噪声能量之比，单位是db

$$MSE(U,V)=\frac{1}{M*N}\sum_{i\in X,X\subset Z^2}\left(u(i)-v(i)\right)^2$$

$$PSNR=10log_{10}\left(\frac{MAX^2}{MSE}\right)$$

SSIM（结构相似性）也是一种衡量两幅图片相似度的指标

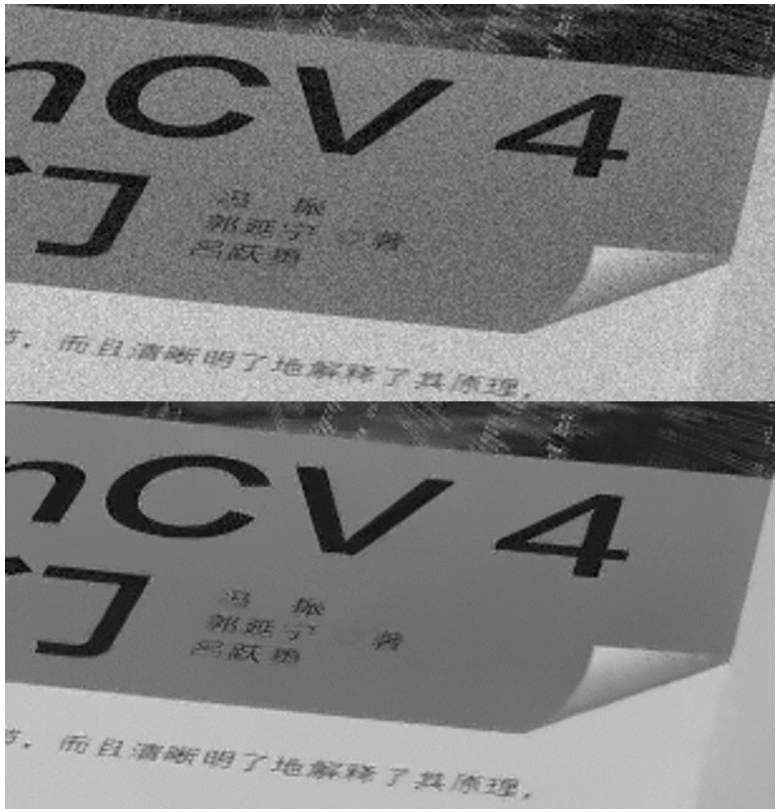
$$SSIM(x,y)=\frac{(2\mu_x\mu_y+c_1)(\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$$

methods	metrics	(a)lena	(b)cameraman	(c)book	(d)sky
中值滤波	PSNR	32.18	31.32	31.27	32.14
	SSIM	0.820	0.838	0.811	0.800
高斯滤波	PSNR	29.41	27.12	28.95	30.86
	SSIM	0.823	0.838	0.846	0.865
双边滤波	PSNR	30.95	31.35	31.43	31.74
	SSIM	0.752	0.749	0.726	0.691
NLM(GM)	PSNR	33.15	33.39	32.56	35.45
	SSIM	0.881	0.910	0.921	0.914
NLM(SM)	PSNR	34.37	34.63	33.52	36.36
	SSIM	0.898	0.924	0.936	0.924

去噪效果测试

算法性能测试

(a) 噪声图像　　(b)NLM(GM)输出图像　　(c)NLM(SM)输出图像



噪声图像（上）NLM(SM)去噪图像（下）

以下各表时间单位均为ms

	256*256	512*512	1024*1024	2048*2048
串行 NLM	2795	11322	44233	173560
NLM(GM)	8.3	31.7	125.5	492.5
NLM(SM)	5.6	21.9	85.4	336.2

图像大小对算法速度的影响

- 共享内存的 NLM 算法相较串行算法，加速了 516 倍。极大加快了 NLM 算法的运行速度
- 共享内存版本相较全局内存版本，速度进一步加快，平均加快了 30%左右。

模板 / 搜索	3	5	7	9	11
21	125.3	125.6	126.3	128.2	148.1
35	349.5	350.2	351.4	355.6	413.7

窗口设置对 NLM(GM)滤波时间的影响

模板 / 搜索	3	5	7	9	11
21	85.5	85.3	93.5	87.0	114.9
35	243.4	243.8	239.7	262.4	330.3

窗口设置对 NLM(SM)滤波时间的影响

网格 \ 块	32	64	128	256
8	126.1	137.5	137.5	183.2
16	131.7	131.7	139.8	137.4
32	126.7	126.7	126.7	137.4
64	125.5	126.3	126.3	126.0
128	123.3	123.3	123.3	126.0

网格和块设置对 NLM(GM)滤波时间的影响

block.x	**2**	**3**	**4**	**8**
运行时间	688.7	302.8	178.3	85.3

块设置对 NLM(SM)滤波时间的影响

南京大学
NANJING UNIVERSITY

感谢各位老师评审与垂听

基于GPU的非局部均值算法的并行优化设计

答辩人：苗金成　　指导老师：李 昀　　教授
学号：201850005　　日期：2024年5月29日