

利用HOG特征和SVM实现目标检测

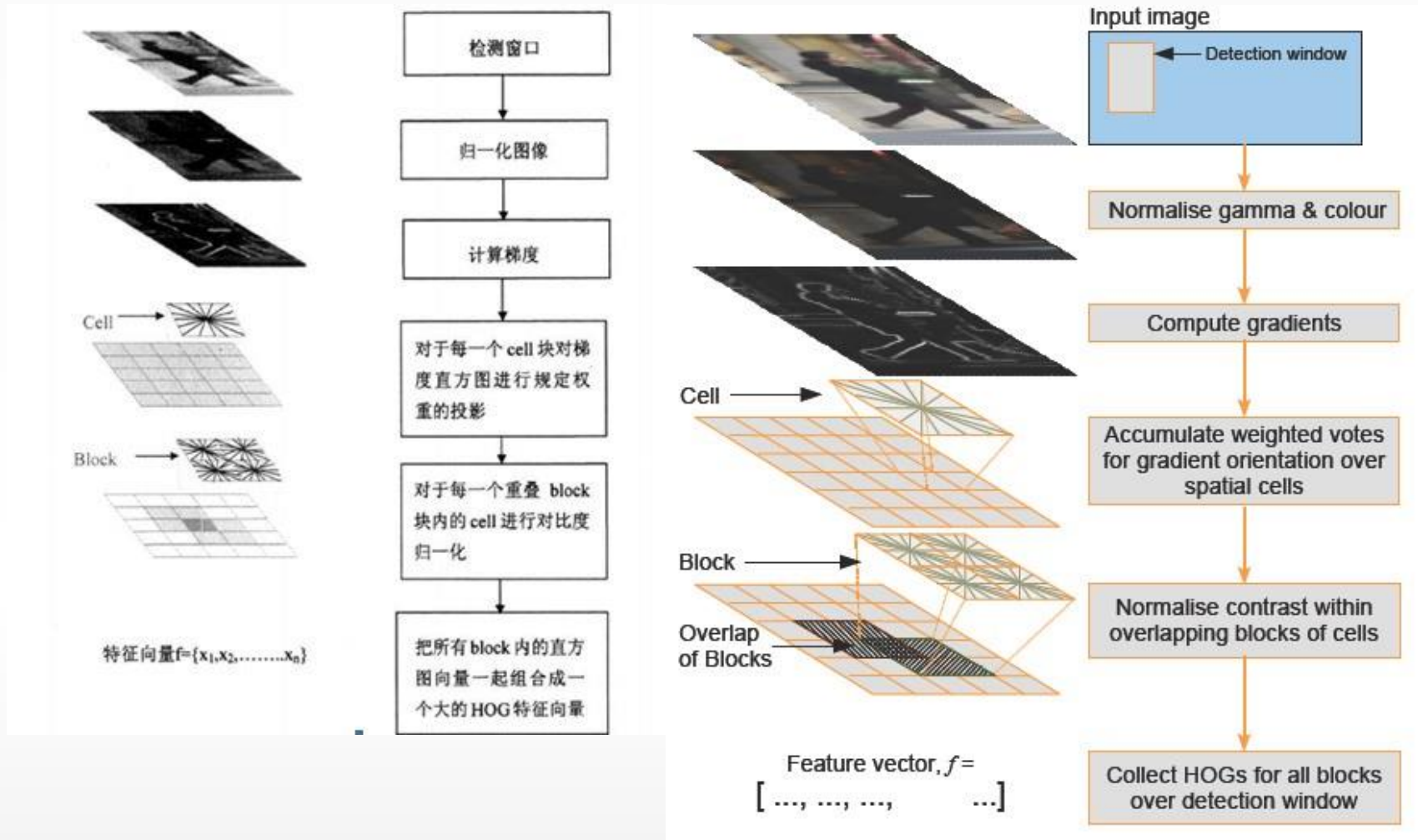
——CUDA结合OpenCV编程

方向梯度直方图 (HOG)

HOG DESCRIPTOR

HOG特征描述子的定义：

局部归一化的梯度方向直方图，是一种对图像局部重叠区域的密集型描述符，它通过计算局部区域的梯度方向直方图来构成特征



HOG描述子

HOG DESCRIPTOR

- 检测窗口Win、块Block、单元格Cell的基本信息：

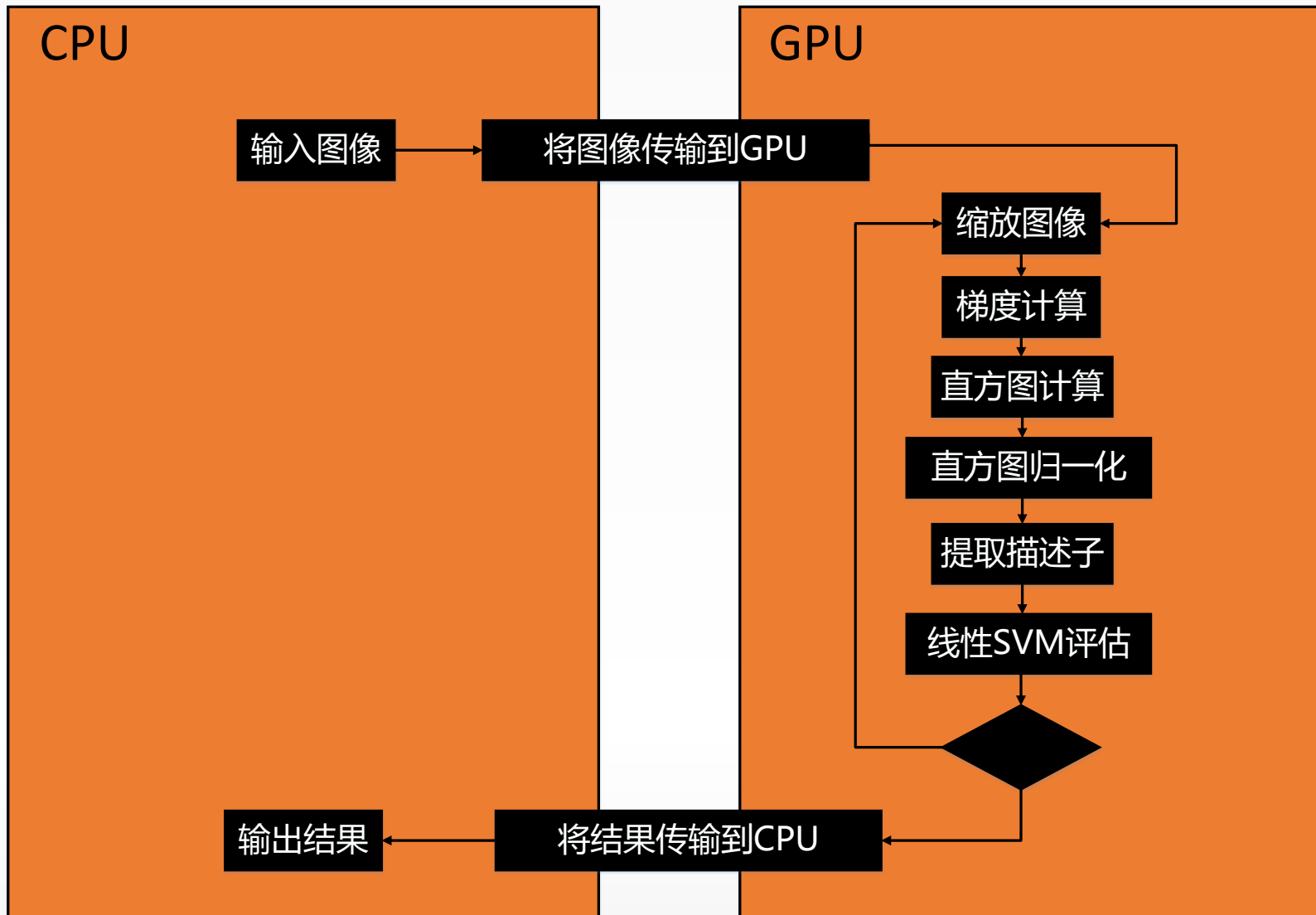
- 大小

- 检测窗口：WinSize=128*64像素，在图像中滑动的步长是8像素
- 块：BlockSize=16*16像素，在检测窗口中滑动的步长是8像素
- 单元格：CellSize=8*8像素
- 梯度方向：一个Cell的梯度方向分为9个方向，在一个单元格内统计9个方向的梯度直方图

- 常量内存

算法流程


PARALLEL ALGORITHMS



尺度缩放（RESIZE方法）

DOWNSCALING AND GAMMA NORMALIZATION

● 纹理内存的使用

- 利用卷积对图像进行滤波处理，这种方法时间复杂度高，处理复杂 
- 利用tex2D函数来获得每个像素的双线性插值
 - 纹理内存本身对传入的图像数组有线性滤波作用
 - 不需要对源图像读取进行越界检查
- 为了最大限度地减少存储器操作，可以在这个阶段应用gamma标准化

块数和线程数选择

NUMBER OF BLOCK AND THREAD IMPLEMENT ON CUDA

步骤	Number of GPU Blocks	Number of Threads
梯度计算	Width* Height / 256	256
直方图计算	图像HOG块的数量	32
直方图的归一化	图像HOG块的数量	64
线性SVM评估	检测窗口的数量	256
提取描述子	图像检测窗口的数量	256

梯度计算

GRADIENTS COMPUTATION

- 在这步中有两个内核函数，一个用于灰度图像，另一个用于彩色图像。每个线程块都包含256个线程，以计算256个像素的区域。这里的实现是逐行完成的
- 当输入图像是彩色时，它有三个通道，alpha通道被忽略。每个线程需要计算三个颜色梯度，三个大小和三个方向（每个通道一个）。对每个像素，选择最大幅度和相应的方向写入host端内存

直方图计算

HISTOGRAM COMPUTATION

● 计算HOG Block

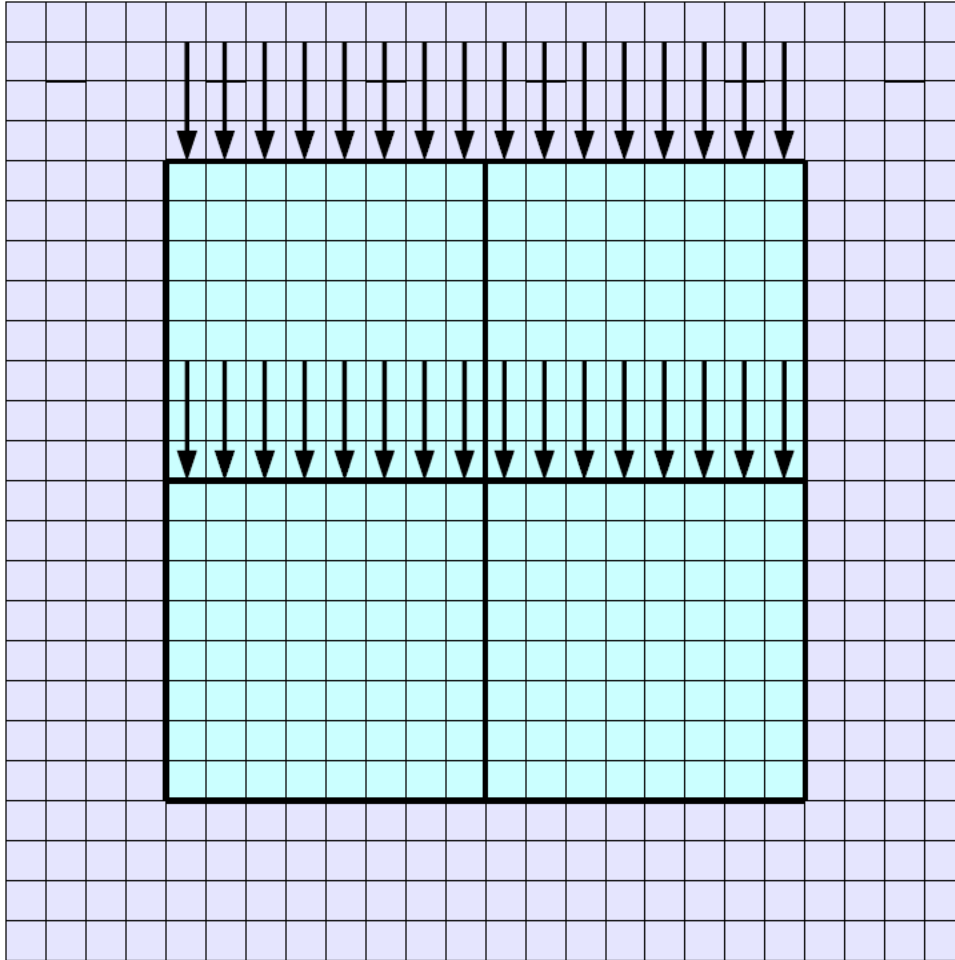
- 每个CUDA线程块对应计算HOG Block
- HOG块有4个单元，每个单元 8×8 像素，所以我们将使用 $8 \times 4 = 32$ 个线程
- 每个线程块分成4个子块，每个子块处理一个cell
- 每个子块由8个线程组成，每个线程处理8个像素（一个单元列）

● 共享内存的使用

- 每个线程计算自己的直方图并将其存储在共享内存中
- 需要每个线程 $9 \times 4 = 36$ 个字节的共享内存
- 对于三线性插值，每个像素将不仅对其cell直方图作出贡献，而且还对其中一个或多个邻近cell作出贡献（*）

直方图计算

HISTOGRAM COMPUTATION



块直方图步骤的线程配置

垂直箭头表示线程和它们
处理像素的方向

每个线程处理一个块中的
一个单元格中的一列

通过L2HYS-NORM对直方图进行归一化

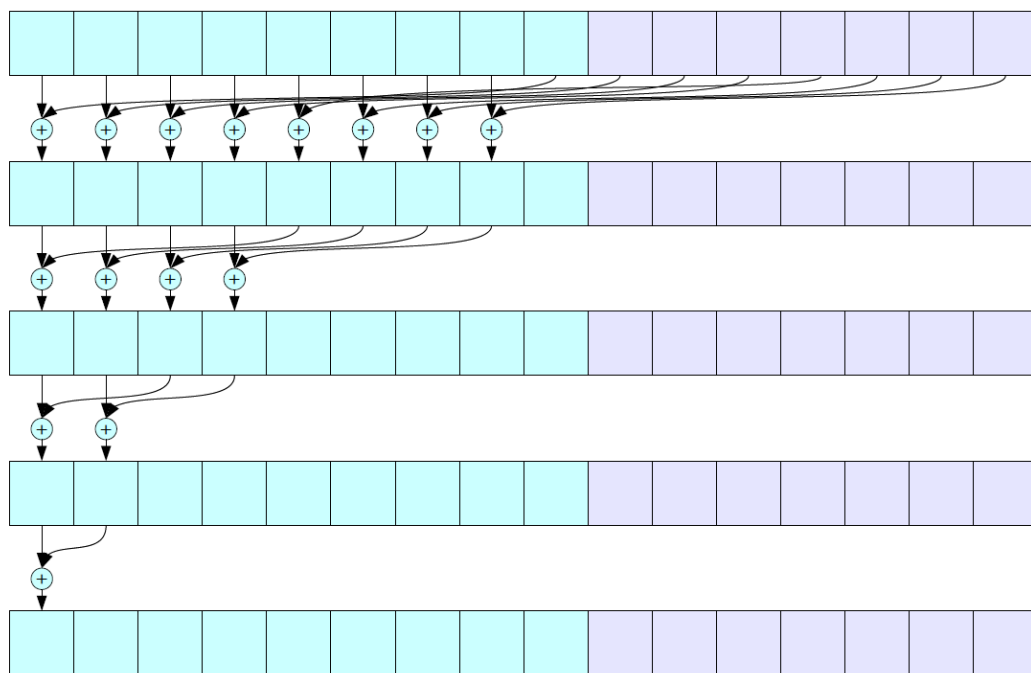
HISTOGRAM NORMALIZATION

- 每个HOG块用L2-Hys归一化
- 每个HOG像素块被映射到CUDA线程块，所以线程块的数量为图像的HOG块的数量
- 每个块由4个直方图组成，每个直方图有9个bin，所以需要 $9 \times 4 = 36$ 个线程来归一化。为了便于计算，我们将线程数设置为64
- 为了避免全局内存访问的延迟，使用共享内存。每块 $64 \times 4 = 256$ 个字节的空间

并行规约算法

PARALLEL REDUCTION

●注意：避免共享内存冲突



使用并行规约一次计算每个块的范数，将每个像素值归一化，将值限制为0.2，再次使用并行规约算法，最后重新归一化

时间复杂度为 $O(\log_2 N)$ ，与串行时时间复杂度 $O(N)$ 相比，减小了计算时间

线性SVM评估

LINEAR SVM EVALUATION

- 每个检测窗口都映射到一个CUDA线程块
- 使用了一个64x128像素的检测窗口，它由7x15块组成。上一步的网格布局时，每个检测窗口的直方图值已经存储在全局内存中
- 每个block detection窗口选择了256个线程，每个处理 $(3780/256 + 1)$ 个值。使用共享内存来存储点积的值，需要每Block $256 \times 4 = 1024$ 个字节的空间
- 将每个直方图bin的值与其线性SVM权重相乘，并行规约得到加权值，然后减去超平面的偏差。被评估的标签最后写回全局内存

提取输入图像的描述子

DESCRIPTORS EXTRICATION

- 在这一步中，我们计算了一个包含 7×15 块的 64×128 检测窗口的3780维特征。每个CUDA线程块都会计算一个HOG功能。所以我们需要105个线程来计算一个HOG特性。为了计算方便，我们在这里设置256个线程
- 由重叠区域内的两个扫描窗口计算的直方图单元产生几乎相同的信息。为了减少这种重复的计算，我们得到独立于前一个内核的全局内存中扫描窗口的直方图单元的值
- 在这一步有两个核函数。第一个内核是按行提取整个图像的HOG特征，而另一个按列

实验结果 (GTX1060 / i7 7700HQ)

EXPERIMENT RESULTS

- 用于1920×1080图像的平均处理时间

图像大小	CPU处理时间	GPU处理时间	加速倍数
1920×1080	313ms	26ms	12

- 1080P摄像头的平均处理帧率

摄像头分辨率	CPU总帧率	CPU HOG 帧率	GPU总帧率	GPU HOG 帧率
1920×1080	3FPS	3FPS	18FPS	39FPS

结论与展望

CONCLUSION & FUTURE WORK

● 结论

- 在GPU上执行的HOG的计算速度远远快于在CPU上运行的HOG
- GPU总帧率 > 10FPS，因此可以实时进行1920x1080的HOG跟踪

● 展望

- 零拷贝方法，以尽量减少相机传感器和GPU之间的数据延迟
- NVIDIA CUDA架构不允许直接使用多个GPU，多GPU系统上的内核实实现和测试