CLSIFT:基于OpenCL的SIFT算法实现及优化

王伟俨
中国科学院软件研究所

SIFT应用背景

- Scale Invariance Feature Transform(SIFT)是一种

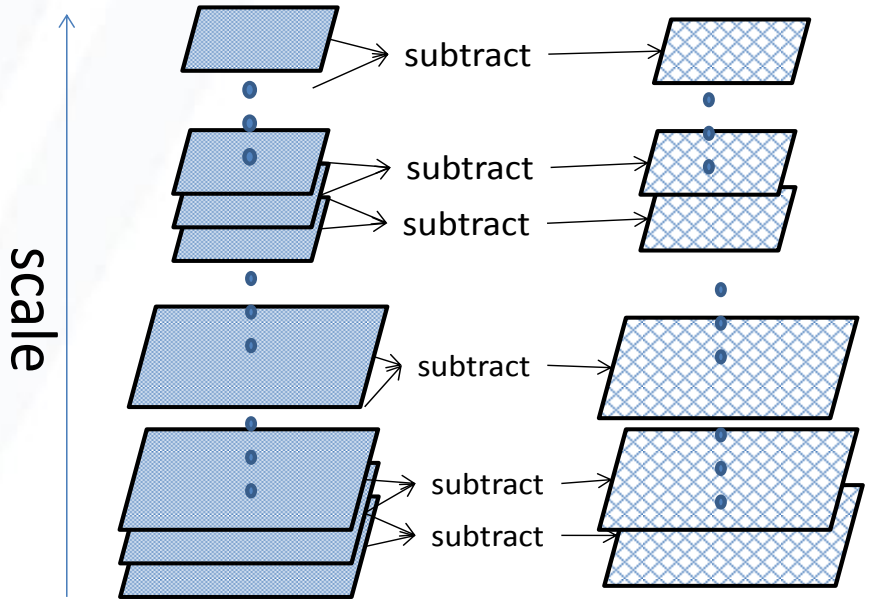
位与导航、图像拼接、三维建模、手势识别、视频跟踪、笔记鉴定、指纹与人脸识别等很多场景有用武之地

由于时间复杂度，难以在实时环境下应用！

SIFT算法：尺度空间生成

- 尺度空间:高斯金字塔及差分高斯金字塔:

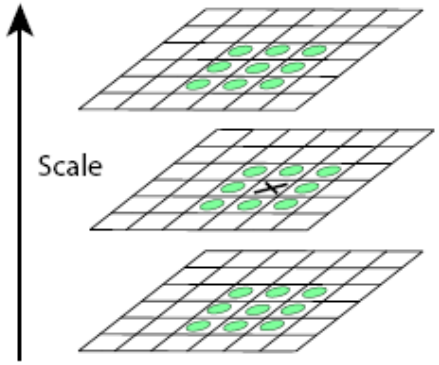
计算量大，读写次数多



输入图像经过高斯卷积以及放缩生成多组多层的高斯金字塔，高斯金子塔相邻两层相减生成差分高斯金字塔。

SIFT算法：特征点定位

- 在整个尺度空间中寻找极值点



- 泰勒插值拟合准确的响应强度，过滤强度低的点
- 借助**Hessian**矩阵计算曲率，过滤主边缘的点

条件分歧 搜索整个尺度空间的访存

SIFT算法：分配方向及特征描述

- 给特征点分配方向，提供旋度不变性
- 特征描述统计周围像素信息，提供部分视角不变性

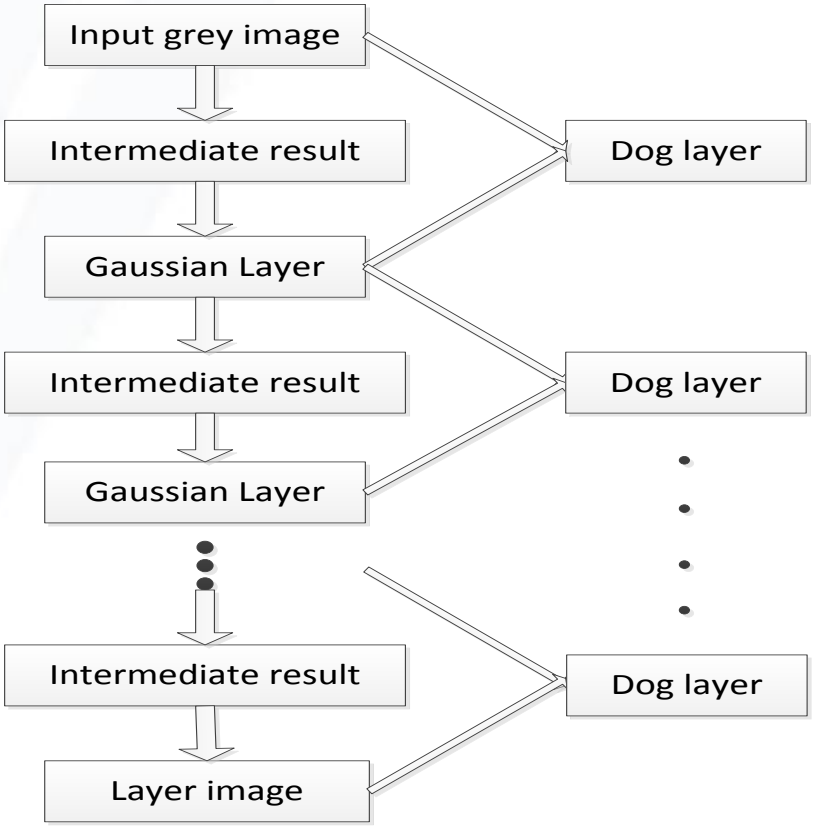
- 分别计算特征点某个半径内的像素的方向及强度，在直方图中统计
- 方向:直方图平滑后，选取极值方向，泰勒拟合出准确方向
- 特征描述:需要将坐标系移动到方向位置再统计直方图，直方图归一及门限化后作为特征描述

- 本质上是规约过程

CLSIFT实现与优化：尺度空间生成

- 通过Profile的time line发现尺度空间生成较耗时
  - 瓶颈在I/O

高斯核在CPU并行计算，异步传输掩盖高斯核计算时间以及数据传输时间



数据流图

CLSIFT实现与优化：合并kernel

- 考虑将两个横向高斯卷积和纵向高斯卷积kernel合并
- 考虑将高斯卷积和差分kernel合并
- 考虑将两个，甚至多个高斯卷积合并

- 合并多个高斯卷积及差分的效果比预想要低

- 合并kernel需要使用local mem和寄存器暂存中间结果，使得GPU发起的work group减少，profile显示设备占有率变低。
- 借助profile统计不同合并方式的资源使用情况
- 平衡减少glaobal mem 访问与资源消耗

CLSIFT：关键点定位优化

- 在整个尺度空间遍历检查每一个像素，访存多

- 判断3个条件:

```
If(extreme(P)==true)
If(respond(P)>bias)
If(curve(P)>bias)
        record(P);
```

显然将出现branch divergence

- 图像中关键点分布不均，造成负载不平衡

CLSIFT:关键点定位中的数据重用

- ✓ 循环队列预取尺度空间，重用数据

  改变遍历尺度空间的顺序:先上下层再左右滑动

- ✓ 暂存中间结果，重用计算结果:

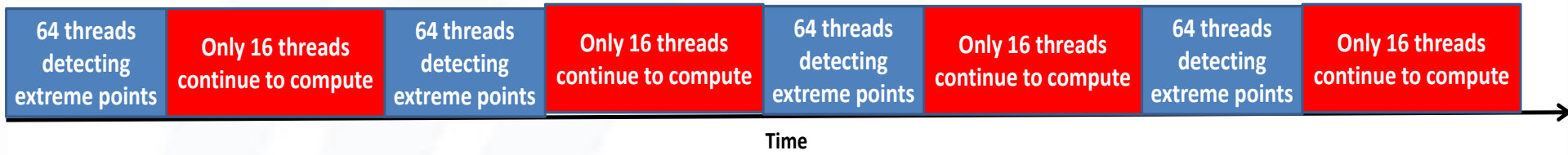
  检查1个点是否为极值需要26次比较，同一层的比较结果可暂存到另一个循环队列中，重用计算结果。

- ✓ **Local mem VS Image Buffer**

  当访存模式高度可预见时，将local mem当作手工cache。可避免Image Buffer的cache使用的条件，确保100%的命中率，避免维护cache的硬件开销。

CLSIFT:维护队列避免条件分歧

⌘ 条件分歧实例



- ✓ 在local memory维护一个队列，当满足条件时将任务加入队列，当队列长度>=64时计算下一个条件

经实际测试，在维护队列但直接计算而不避免条件分歧时，运行速度和没有队列时基本一致，维护队列开销小

CLSIFT:负载平衡

- 关键点在图像中不同部分有多有少
- 图像中同一区域中的关键点，在尺度空间的不同层次应该服从均匀分布
- **Workgroup**当完成尺度空间中一组层次的关键点定位后，移动另一个区域继续搜索。使得不同**work group**负载趋于平衡
- 不使用队列的原因在于，**GPU**全局队列以及全局同步的开销极大，所有的线程要竞争**global** 变量。开销与收益差距不大

CLSIFT:直方图统计

- 方向分配和特征描述生成本质上都是累加出直方图，是一个规约过程。

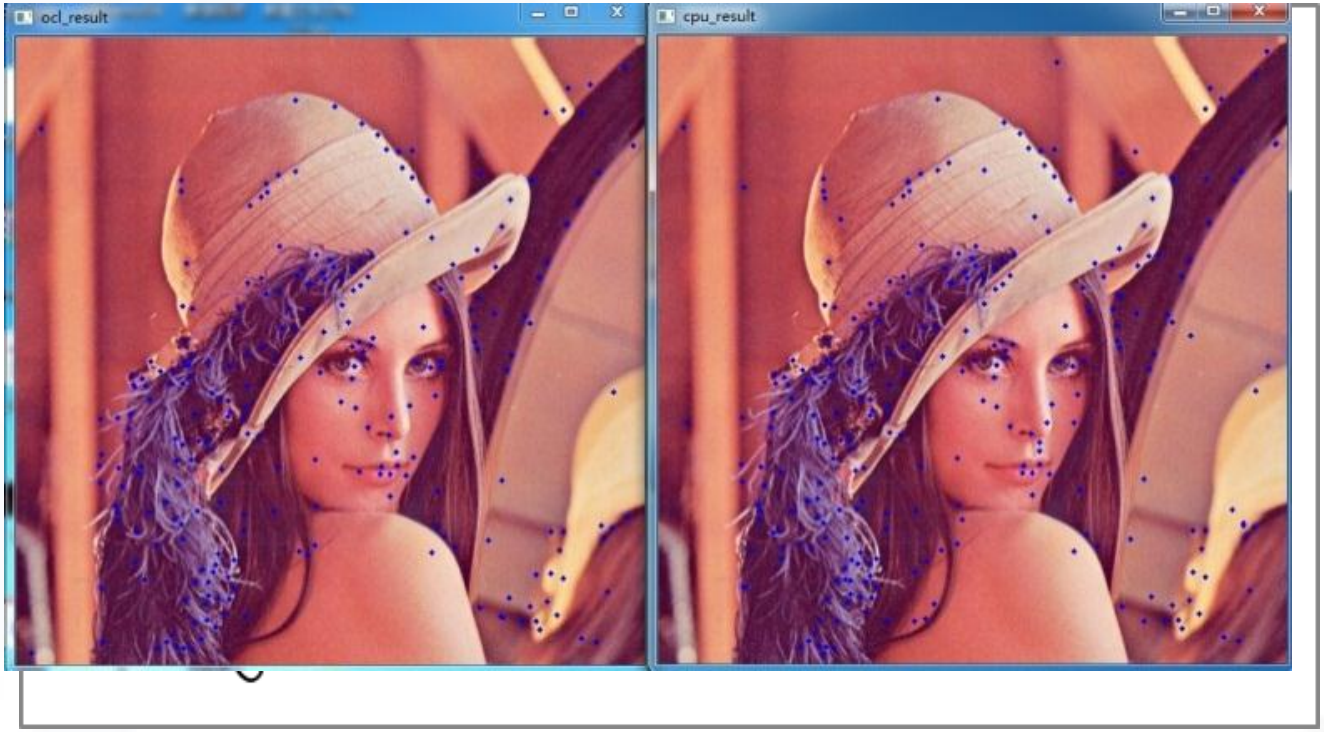
- 如何发掘并行度?

每个像素的方向及强度计算互相独立，但累加入直方图时会有冲突

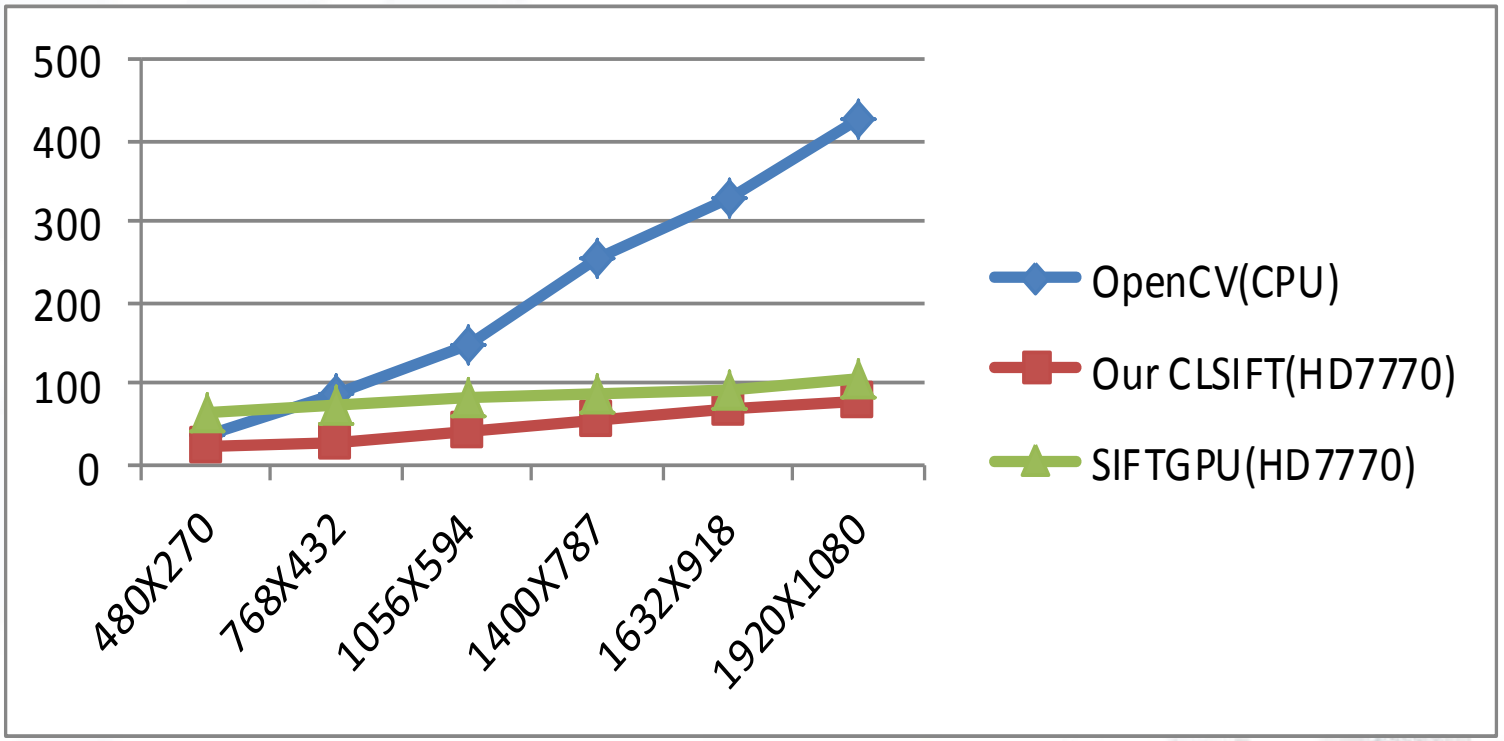
- ✓ 像素计算时并行，并设立多个子直方图并行累加。最终直方图由子直方图累加生成。
- ✓ 子直方图需要消耗local mem，具体数量需要与设备占有率进行平衡

CLSIFT：实验结果

- 以lena.jpg为测试用例，将CLSIFT与OpenCV和SIFTGPU进行了比较，如图所示。



- 使用一组内容相同但大小不一的图像作为测试用例，测试CLSIFT，SIFTGPU以及OpenCV的性能变化趋势



CLSIFT:总结

- ✓ 数据重用与计算中间结果重用
- ✓ 合理的并行粒度
- ✓ 避免条件分歧与负载平衡
- ✓ 选择合适的优化方式，引入的开销因比收益小

- 合理使用profile，codeXL工具
- Timeline 快速分析应用的瓶颈
- Kernel analyzer分析资源使用情况
- Profile分析影响占用率因素，检查内存等资源泄漏
- codeXL可对kernel代码进行Debug

Q&A

感谢聆听！