时间：2012.10.24

Cuda 项目组第六次例会

一）.关于代码的两点调整

1.关于 gridsize 和 blocksize

```
dim3 blocksize, gridsize;
blocksize.x = DEF_BLOCK_X;
blocksize.y = DEF_BLOCK_Y;
gridsize.x = (outsubimgCud.imgMeta.width + DEF_BLOCK_X - 1) / DEF_BLOCK_X;
gridsize.y = (outsubimgCud.imgMeta.height + DEF_BLOCK_Y * 4 - 1) /
             (DEF_BLOCK_Y * 4);
```

因为 gridsize 是与 blocksize 逻辑相关的，所以为了可能出现的逻辑错误，**应该改为：**

```
dim3 blocksize, gridsize;
blocksize.x = DEF_BLOCK_X;
blocksize.y = DEF_BLOCK_Y;
gridsize.x = (outsubimgCud.imgMeta.width + blocksize.x - 1) / blocksize.x;
gridsize.y = (outsubimgCud.imgMeta.height + blocksize.y * 4 - 1) /
             (blocksize.y * 4);
```

2. 关于__device__成员函数的隐藏的错误

__device__是以 inline 的形式执行的，所以对于 CLASS 中的__device__方法，在方法声明时需要将定义写出来。

```
public:

    // 构造函数：HistogramSpec
    // 无参数版本的构造函数，所有的成员变量皆初始化为默认
    __host__ __device__
    HistogramSpec()
    {
        // 使用默认值为类的各个成员变量赋值。
        refimg = NULL;      // 参考图像默认为空。
        refHisto = NULL;    // 参考直方图数组默认为空。
    }
```

二.会议记实：

王媛媛:

GetObjectContour 算法

1）代码进行编写时，模板中心点选择出现了错误，需要改正。

1）实现了 GetObjectContour 算法第一步，即圆形模板的膨胀过程

2）计划是实现该算法的第二步。

侯怡婷：Hough 变换检测直线

1）使用直线的极坐标表示： $\rho = x * \cos\theta + y * \sin\theta$

这里想了两种方法：

方法一：想采用对于矩阵的每一行、每一列取一个最大值，然后将这些最大值合并，以求得最终的结果

方法一劣势：

（1）行最大值有 1280 个，列最大值只有 360 个，行的最大值会包括列的最大值。

（2）图像中若有一条直线，则在 BufHough 数组中会统计得出一个峰值，但是该峰值周围的邻域的值也会比较大，如果只是按行、列来寻找最值，最终的最值可能都在一个小的区域内，代表一条直线。

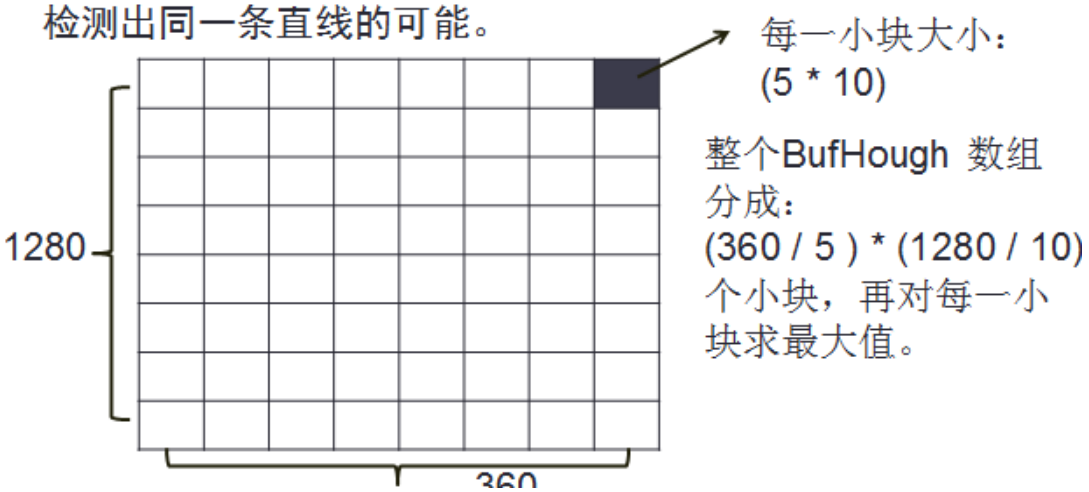
5	7	387	3
7	5	515	3
37	235	605	2
134	626	501	2
237	626	370	1
340	625	243	1
443	625	112	
508	385	11	
489	6	5	

如左图所示，该区域中红色为该行的最大值 626，而下面三行的最大值与该值很接近，并且代表同一条直线。

- 方法二：

将 BufHough （360 * 1280）矩阵分块，分块计算最大值，这样能保证在一个邻域中只有一个最大值。

这里分块的大小就会影响最终的直线检测结果。如果分块太大，则可能会漏掉某条直线，若分块太小，还是会有检测出同一条直线的可能。



在分块之后，计算量仍然很大，因此需要用到双调排序。

龙哥建议：1）对于 cos θ 和 sin θ 可以直接调用库函数；

3） 分块的大小可以设计为宏，在使用的时候可以根据实际情况对其调整。

刘宇：

1）初步完成几何矩（空间矩、中心矩、Hu 矩）

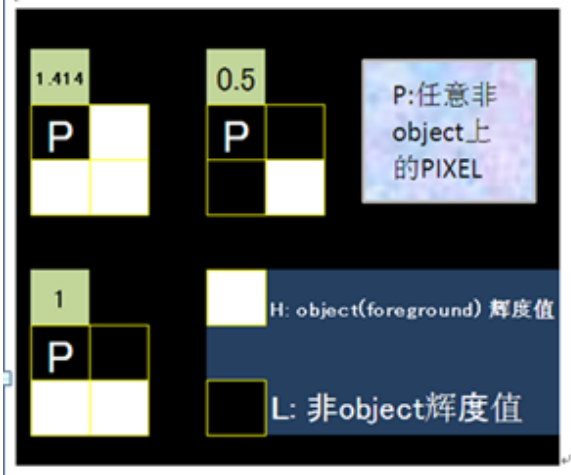
Moments.cu Moments.h

参照了《基于差分矩因子的灰度图像矩快速算法》，王冰，计算机学报，2005.

2）完成轮廓长并行计算

GeometryProperties.cu GeometryProperties.h

轮廓长的并行计算



分别统计上述 PATTERN 在指定图像范围内的个数----P 的位置为任意,不限于左上角.
PATTERN 的种类有 3 种.

A. P 的 3-近邻内有 1 个 OBJECT 上的 PIXEL;
B. P 的 3-近邻内有 2 个 OBJECT 上的 PIXEL;
C. P 的 3-近邻内有 3 个 OBJECT 上的 PIXEL;
记上述 COUNTER 分别为 A,B,C, 则 OBJECT 的轮廓长度.

$L = 0.5A + B + 1.414C$ when $C > 0$
$L = A + B$ when $C = 0$

罗劼：

图像匹配算法

1） 目前已经搭好了程序的框架，完成了部分通过坐标映射来进行图像匹配算法的代码。
其中，在实现时采用了三维，第三维表示旋转角度。

2） 按照上周龙哥的建议，新定义了一个 struct，减少了参数的个数。

计划：

因为涉及到图像的标准化，所以下周对其进行研究。

邓建平：

1）根据河边的要求，重复双边滤波，使用了纹理内存。
2）完成了一个线程处理四个点，性能提高了 0.2 至 0.3ms。

龙哥：

删除 texture 重绑定。

欧阳翔：

1） 讲解了 Freckle Filter 问题需求

龙哥建议：不一定要按照需求文档上的要求一步一步来，可以寻求等价的方法来获得想要得结果。

张丽洁：

1）按照自己的理解写出了过程算法示意，反馈给了河边老师。

杨伟光

1）涉及到两种参数 Image 和坐标点集
2）在原来的函数基础上重载形成一个可执行版本。

仲思惠

1）完成了优势法，准备实现第二种方法。

李冬

多阈值二值化算法正在调试中。

龙哥建议：

可以通过一些手段使得自己的代码尽量处于可运行状态，这样可以快速定位自己编码的 bug 在哪儿。

勤发布已有版本，只要有可以运行的版本都可以发布，防止因为优化导致以前的版本被无意删除。