# MC-GPU说明书

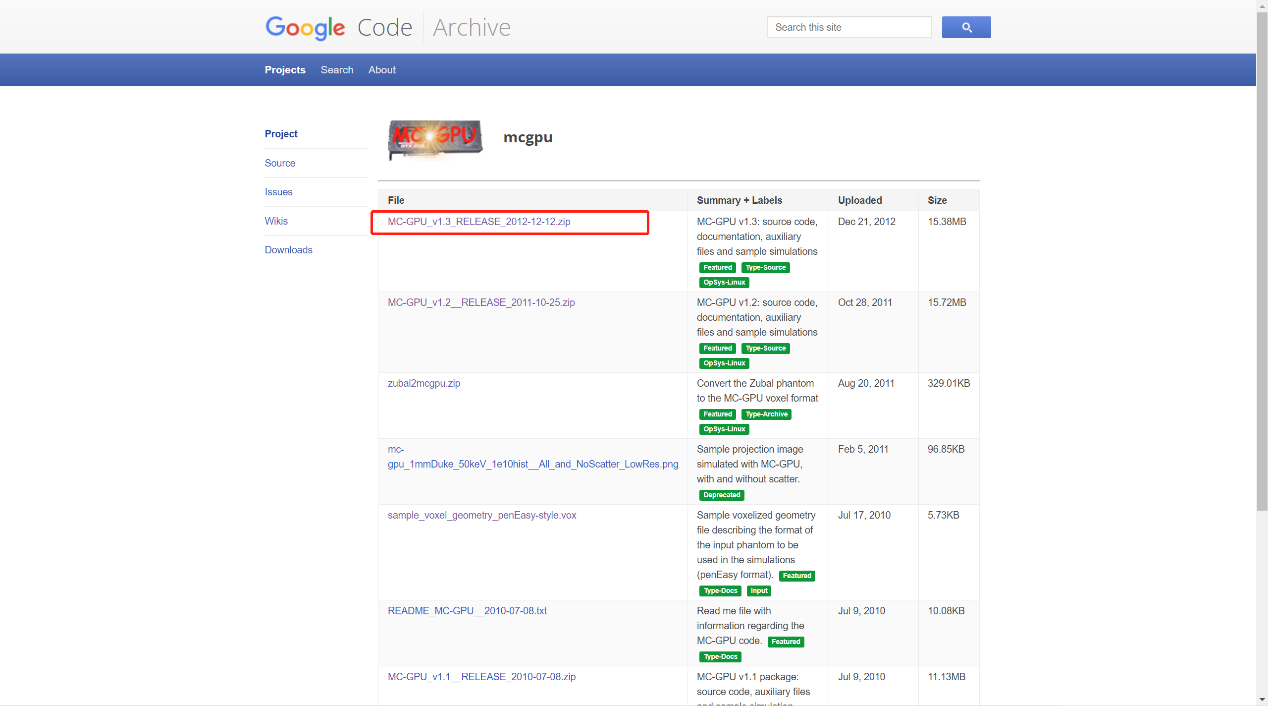
MC-GPU代码 [1] 是基于蒙特卡洛模拟的代码。它可以利用GPU生成人体模型的CT扫描信号。该代码实现了大规模多线程蒙特卡洛模拟算法，成功地模拟了X射线在体素化几何体中传输的过程。

## 下载流程：

### 1.1下载网址：

https://code.google.com/archive/p/mcgpu/downloads

下载最新版（用红框标出） ，MC-GPU仅在Linux操作系统中开发和测试,所以大家需要在Linux操作系统下解压文件夹。



**图一** MC-GPU的下载网页

### 1.2解压后的文件及其用途：

**DOCUMENTATION:** MC-GPU说明书

**GNUPLOT\_SCRIPTS\_VISUALIZATION：**官方提供的画图文件。不建议使用。

**MC-GPU\_material\_files：物质材料的压缩文件。**模体包括以下材料的各种组合：空气、肺、脂肪、软组织、大脑、皮肤、肌肉等物质。根据从PENELOPE数据库 [2] 获得每种物质与X射线能谱相互作用时的衰减情况和散射情况。

**SAMPLE\_SIMULATION\_simple\_geometry:** 简单模体。文件夹里存在模体文件，模拟输入文件。

**SAMPLE\_SIMULATION\_Zubal\_phantom：**Zubal模体 [3]。

后缀名为**spc**的文件：光谱文件。

[**.sh**文件，**Makefile**文件]：与代码编译和程序运行有关。我推荐直接用命令行来使程序运行，所以不会使用这些脚本文件。

[包含“**create\_material**”的文件]：与生成材料有关。但是在**MC-GPU\_material\_files**里已经有物质材料的压缩包了，我们可以忽略这两个文件。

其他文件：与代码编译和程序运行有关，我们接下来会简单说明。

## 二、MC-GPU简介

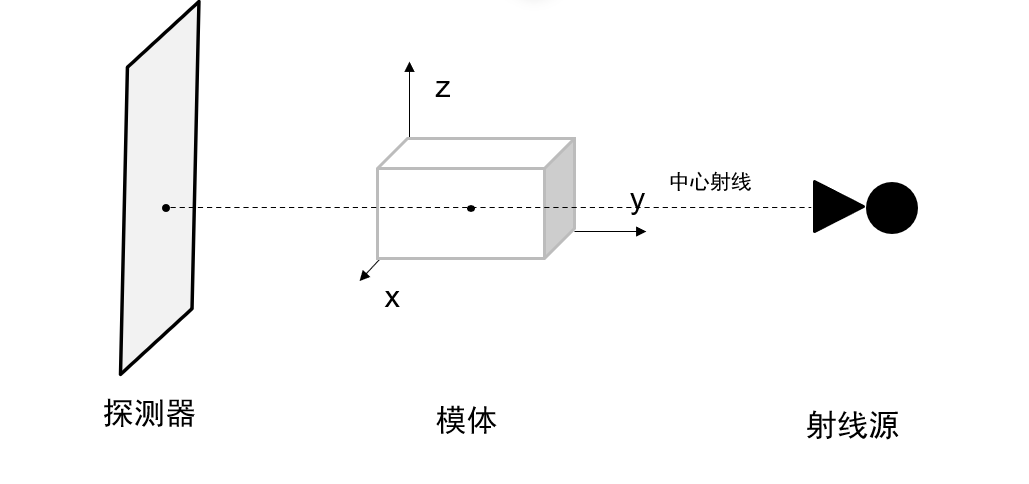
**我们在这里提供MC-GPU的简要描述。**MC-GPU代码中各个函数和变量的解释在DOCUMENTATION里的文件里。我们这里大体解释MC-GPU代码实现模拟的过程：

1. 指定用作模拟的体素化模体，将数据写入后缀名为vox的文件（后缀名其实无所谓，都是文本文件）
2. 调整模拟输入的文件，将射线源和探测器的属性写入后缀名为.in的文件
3. 编译并运行模拟程序。

**模拟结束后，我们会得到平板探测器上收集到的信号。**这些信号分为四种：没有发生散射的X射线信号，康普顿散射信号，瑞利散射信号，多散射X射线信号(multi-scattered x-rays)。我们将四种信号相加即可得到受到散射污染的信号。

**同时，我们需要写python语言处理探测器采集的信号，得到投影并进行重建。**首先，我们在探测器和射线源中不放置物体，模拟结束后，可在探测器上得到射线源的空扫信号。我们将空扫信号和放置模体时的采集信号均转换成numpy文件，通过对数转换后，可得到投影。最后，我们可利用FDK算法将不同角度的投影重建为CT图像。重建工具是astra-toolbox [4]。

我们接下来会具体解释模拟过程。

****

**图二** MC-GPU里的坐标系

### 2.1指定用作模拟的体素化模体。

我们以文件夹**SAMPLE\_SIMULATION\_simple\_geometry**中的简单模体为例。该模体只有六个体素组成。六个体素具有两种材质以及三种不同的密度，具体形状可见png文件。如图2所示。在**[SECTION VOXELS HEADER]**下，各行依次代表：

1. 沿X，Y，Z轴的像素数。这里我们需要解释一下MC-GPU里的坐标系如何确定。模体在MC-GPU里的形状都为长方体。如图二所示，坐标系原点位于模体的后下角位置。X，Y，Z轴在长方体的棱上。整个模体均在第一象限内。射线源和探测器位于模体的两边。
2. 每个体素沿X,Y,Z轴的长度。
3. 第一列数字代表何种物质。
4. 第二列数字代表何种密度。
5. 官方文件的意思是空白行代表一行体素（沿X轴）的结束。实际上如果观察实际代码，这个值为0或1对结果无影响。我也做过实验验证过这个推断的正确性。

在**[END OF VXH SECTION]**下，两列数值分别代表的每个体素的材料和密度。每个体素按照X，Y，Z的顺序依次写入。第一个体素位于原点上，并且后续体素沿X、Y和Z轴（第一象限）排列。

### **2.2调整模拟输入文件**

输入文件MC-GPU\_v1.3\_6voxels.in可以分为以下部分。

#### 2.2.1 GPU参数和蒙卡模拟参数配置 [SECTION SIMULATION CONFIG]

**第一行：模拟的X射线总数**。数目越大，辐射剂量越大，模拟所需的时间越长。

**第二行：随机数种子**。蒙卡模拟需要设计随机数。RANECU是PENELOPE中使用的随机数生成器。该代码可以生成伪随机数，是根据递推公式实现的。这里的值相当于提供了初始值，实际上该值大小对实际结果影响不大。我们采取默认的值不变。

**第三行：GPU编号。**当我们不使用MPI时，又有多个显卡可以同时用来模拟时，我们可以指定模拟程序运行的编号。如有4张卡，可在0-3里选取数字以指定显卡，以使模拟程序高速运行。

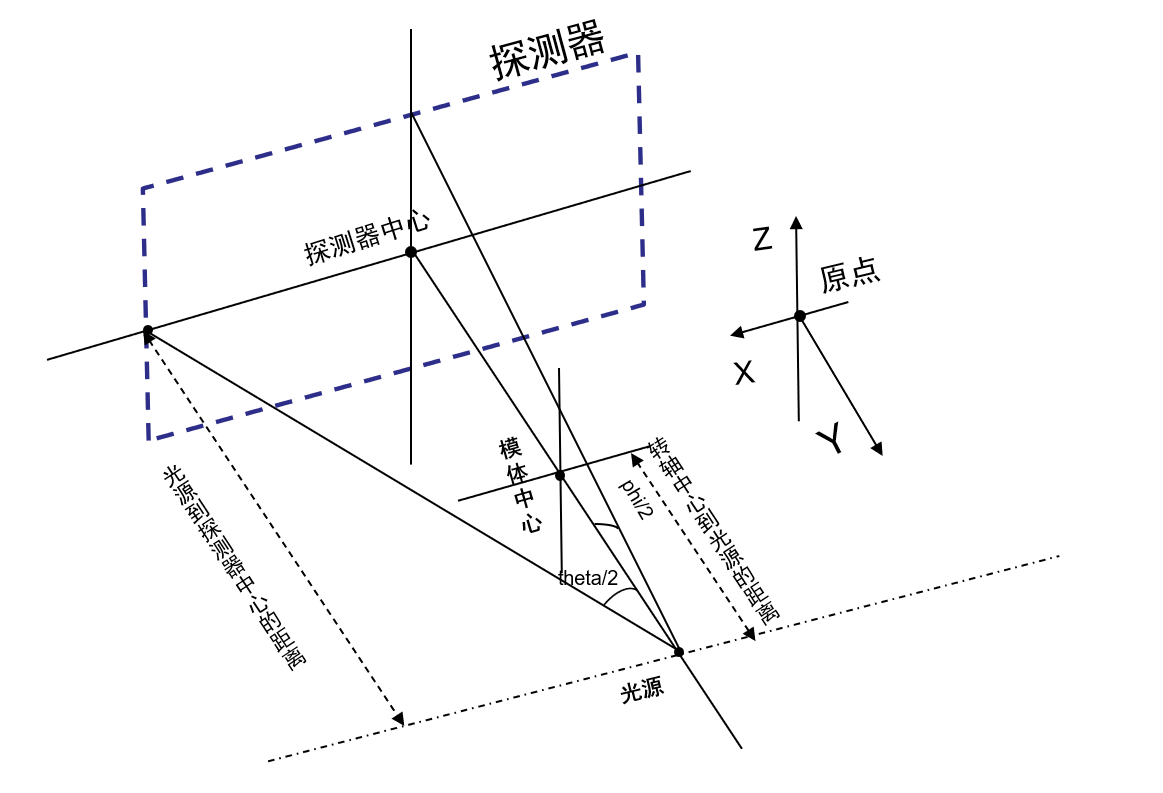
**第四行：显卡里每个BLOCK里运行线程数。**

**第五行：每个线程模拟的X射线数。**这几个变量存在数学关系关系：block数 \* 每个BLOCK的线程数 \* 线程模拟的X射线数 模拟的X射线总数。当对一些复杂的模体进行模拟时，GPU的利用率是100%。改变第四行和第五行的值不会提高代码的运行效率。

#### 2.2.2射线源属性配置 [SECTION SOURCE]

**第一行：X射线能量光谱文件的位置。**我们以90keV.spc和120kVp\_4.3mmAl.spc为例。90keV.spc文件代表单能射线。将后缀改为txt即可查看光谱文件的具体信息。120kVp文件代表连续能量分布射线，4.3mmAl代表在X射线管出口放置4.3mm的Al,以吸收X射线的低能成分。射线源可以采用MC-GPU提供的光谱文件，也可以采用一些光谱模拟软件。见[I-STAR/SPEKTR: A computational tool for x‐ray spectrum modeling and analysis. (github.com)](https://github.com/I-STAR/SPEKTR)

**第二行：光源的位置坐标。X射线源被定义为发射X射线的点源。**我们在上文解释过MC-GPU代码坐标系的设置。为了更好理解，我们在图三里更为细致的解释。光源的位置是（15，-45，15），显然光源的y坐标为负。图三里的光源的y坐标为正，目的是为了便于画图解释。



**图三** 某个角度投影下的几何视图

**第三行：光源射线的射出方向。**这里是用方向向量表示的。很好理解，光源位置y坐标为负，而模体放置在第一象限，所以光源射线的方向向量的y方向为正。

**第四行：方位角和极角.**在MC-GPU里，光源发射出的是准直后的锥形光束。射线穿过模体后，在探测器平面上产生矩形场。矩形场在指定的方位角(phi)和极角(theta)范围内。当方位角和极角为正时，锥形光束范围由这两个值确定。当方位角和极角为负时，锥形光束范围由探测器的信号采集范围确定。这个值会在探测器属性配置里确定。

#### 2.2.3探测器属性配置[SECTION IMAGE DETECTOR]

**第一行：探测器采集到的信号的文件名**。如果投影次数为1，文件名就是mc-gpu\_image.dat。如果次数大于n，文件名为mc-gpu\_image.dat0000，mc-gpu\_image.dat0001…mc-gpu\_image.dat000n。

**第二行: 探测器的信号采集范围的像素数。**

**第三行：探测器的信号采集范围的大小。**当方位角和极角为正时，信号采集范围往往比探测器采集到的锥形光束范围大。当方位角和极角为负时，锥形光束范围就是信号采集范围。

**第四行：光源到探测器中心的距离。**

#### 2.2.4 CT扫描轨迹设置 [SECTION CT SCAN TRAJECTORY]

**第一行：投影数。**

**第二行：每个投影间的角度。**

**第三行：感兴趣的角度范围。**不在这个角度范围的将不计数。所以这个范围较大点好。

**第四行：光源到转轴间的距离。**

**第五行：投影间的垂直平移。**如果该值不为0，则CT扫描可能与螺旋CT有关。我们不考虑螺旋CT，所以该值设置为0即可。

#### 2.2.5剂量沉积属性配置 [SECTION DOSE DEPOSITION]

MC-GPU会生成两种计数文件：X射线图像的投影文件，模体内沉积的辐射剂量计数文件。每个计数值单位为：。这个不在我们的研究范围内，我也没做过多了解。

#### 2.2.6 扫描模体的指定

**输入模体的文件名**。

#### 2.2.7体素对应的材料的指定

**每一行代表不同物质的压缩文件。**每种物质的行号与vox文件中的物质相对应。

### 2.3 编译和运行代码程序

MC-GPU需在linux里开发和测试。我们可以用makefile脚本编译代码，也可以用命令行编译代码。我们需要安装gcc编译器和CUDA库。因此，我们若用makefile脚本,需要修改相应的库路径。我们选择更好理解的命令行编译代码。

若电脑未安装gcc和CUDA，我们需要进行安装。除此，我们也可以openmpi进行加速。我通过实验，在不改变代码的情况下，模拟速度影响不大。我们推荐两个教程辅助安装：

[(205条消息) linux 安装cuda教程（图文）\_软件测试李同学的博客-CSDN博客\_cuda linux](https://blog.csdn.net/qq_29720657/article/details/109076087)

[(205条消息) Linux下gcc编译器的安装与使用\_prest0r的博客-CSDN博客\_gcc编译器安装linux](https://blog.csdn.net/weixin_39947884/article/details/108566437?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522167239999616800184174264%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=167239999616800184174264&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-108566437-null-null.142%5ev68%5epc_rank_34_queryrelevant25,201%5ev4%5eadd_ask,213%5ev2%5et3_esquery_v3&utm_term=%E5%AE%89%E8%A3%85gcc&spm=1018.2226.3001.4187)

我们可在CPU，自己电脑上的GPU，实验室工作站上的GPU（推荐NVIDIA RTX 3090 GPU）,学院集群上的GPU上进行模拟。

若用CPU编译，可在命令行中输出以下代码：

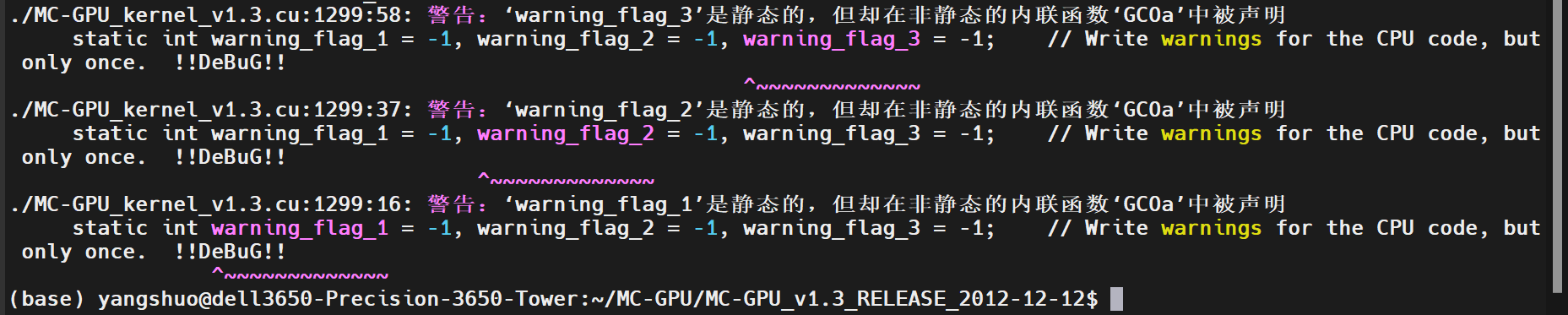
*gcc -x c -O3 MC-GPU\_v1.3.cu -o MC-GPU\_v1.3\_CPU.x -I./ -lm -lz*

**-O3**: 提供最高级别的代码优化。

**-I.** 用来指定程序要链接的库。-I参数紧接着的就是库名。-lm代表数学库。-lz代表压缩库。

**MC-GPU\_v1.3\_CPU.x：**链接阶段结束后生成的可执行文件。

输入以下代码后，会出现以下警告。当我们用GPU编译时，不会出现以下结果。我们比较后，发现采集到的信号是一样的，所以该警告可以忽略。



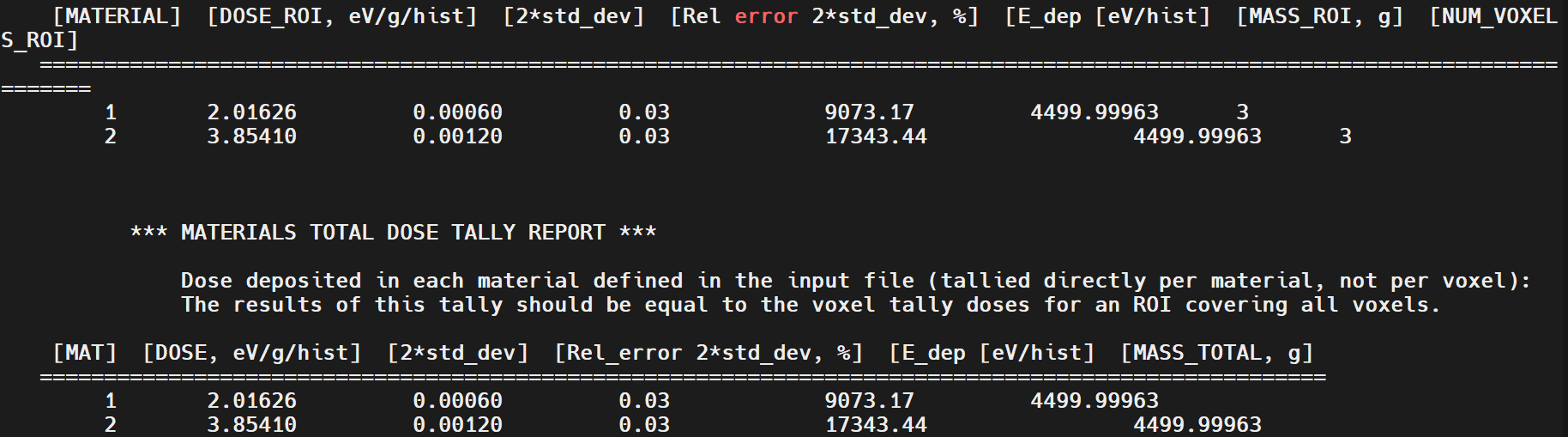
**图四** 利用CPU编译代码的命令行输出

我们用以下代码进行模拟

*cd SAMPLE\_SIMULATION\_simple\_geometry*

*../MC-GPU\_v1.3\_CPU.x MC-GPU\_v1.3\_6voxels.in | tee MC-GPU\_v1.3\_6voxels.out*

.out文件里包含了模拟过程中的信息。我们可在命令行中得到以下图五的结果。同时，在SAMPLE\_SIMULATION\_simple\_geometry里得到四个新文件。其中，后缀名为raw的文件为2进制文件，带“dose”的文件与模体沉积剂量有关。因此，我们只处理mc-gpu\_image.dat文件。文件里的四列数值代表的是探测器采集到的四种信号。



**图五** 模拟结束后的命令行输出

**我们用GPU进行模拟可使程序运行速率大大提升。**用GPU编译的代码如下：

*nvcc -DUSING\_CUDA -DUSING\_MPI MC-GPU\_v1.3.cu -o MC-GPU\_v1.3.x -I. -I/usr/local/cuda-10.0/include -I/usr/local/cuda-10.0/samples/common/inc -I/usr/local/cuda-10.0/samples/shared/inc/ -I/usr/include/openmpi* *-lmpi -lz* *-**-ptxas-options=-v*

**nvcc**：编译cuda程序的编译器。

**-DUSING-CUDA -DUSING\_MPI:** 预处理器变量。后面的四个地址根据cuda和openmpi安装的位置做修改。

**-lmpi -lz:** 链接的库。

**--ptxas-options=-v:** 查看kernel在编译阶段确定的所有静态资源。

**命令行输入以上代码后，会出现很多警告。**这是因为有些代码在cuda10.0以后被弃用了。我们将MC-GPU\_v1.3.cu里的**cudaThreadSynchronize()函数改为****cudaDeviceSynchronize()，****将**cudaThreadExit()函数改为cudaDeviceReset()函数，就不会出现这些警告。

我们使用以下代码运行：

*cd SAMPLE\_SIMULATION\_simple\_geometry*

*../MC-GPU\_v1.3.x MC-GPU\_v1.3\_6voxels.in | tee MC-GPU\_v1.3\_6voxels.out*

显示，用GPU和CPU模拟得到的结果一样。而且速度提高很多。

**当我们利用学院集群进行蒙卡模拟时，卡越多，程序运行越快。**我们先使用salloc指令申请GPU显卡，然后可使用module load指令下载CUDA和openmpi(可选择)。编译指令和运行模拟的指令与上面的指令一样。

## 三、总结

在我们了解如何设置模体，配置相关参数，编译和运行程序和对信号处理后，我们就可得到未受散射污染的图像和受到散射污染的图像。事实上，调整X射线模拟总数我们可得到低剂量CT图像。调整采样视图数目，我们可以得到稀疏视图CT图像。所以，我们也许或许可以用MC-GPU进行其他CT成像研究方向的工作。

目前我只是解释了MC-GPU里各种文件的意义，如何编译和运行程序以及我对使用这个程序的经验之谈。在实际操作过程中，可能会遇到一些其他的问题，我无法一一做出说明。更多的细节还是需要查询官方提供的参考书。除此，我在整理信号处理代码和CT重建代码等代码，未来也会给出具体的代码。希望大家能不断指正错误，我也会不断完善这个说明书。

## 四、引用

[1] A. Badal and A. Badano, "Accelerating Monte Carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit," *Medical Physics,* vol. 36, no. 11, pp. 4878-4880, Nov 2009.

[2] J. Sempau, A. Badal, and L. Brualla, "A PENELOPE-based system for the automated Monte Carlo simulation of clinacs and voxelized geometries-application to far-from-axis fields," *Medical Physics,* vol. 38, no. 11, pp. 5887-5895, Nov 2011.

[3] I. G. Zubal, C. R. Harrell, E. O. Smith, Z. Rattner, G. Gindi, and P. B. J. M. p. Hoffer, "Computerized three‐dimensional segmented human anatomy," vol. 21, no. 2, pp. 299-302, 1994.

[4] W. Van Aarle *et al.*, "Fast and flexible X-ray tomography using the ASTRA toolbox," vol. 24, no. 22, pp. 25129-25147, 2016.