2019年10月7日 星期一

Beamtest software

束流分析软件框架是在NumEval的框架上设计的，目的在于可以在束流实验时快速查看所预期的击中光子分布，因此舍弃了一些NumEval的功能。

束流实验数据分为三种：

RICH：4片AGET板（每板4片AGET）读出的数据

Track-AGET：共三块Tracker，各一块AGET板（每板4片AGET）的数据

Track-VMM：共两块Tracker，各一块VMM板（每板2片VMM）的数据

处理的流程如下：

1. 首先将每次数据进行打包，按照/data/DataTemplate下的格式，将原始数据分别拷贝到对应的目录下：

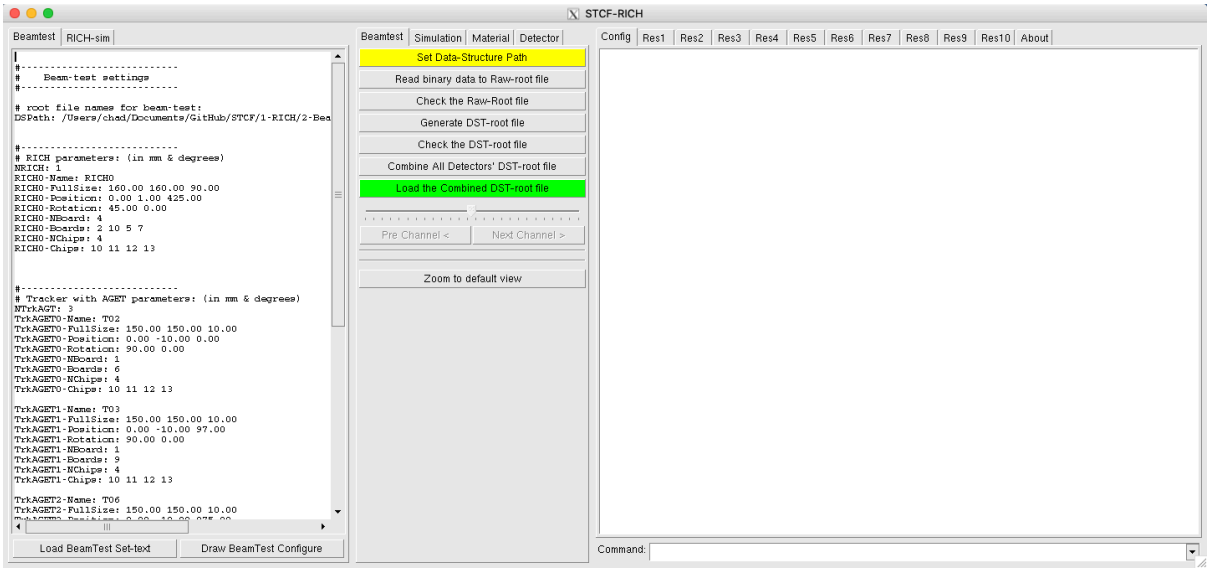
/data/DataTemplate/Combine　(分析后合并到数据，具体格式见下文)

/RICH　(RICH的原始数据放在这里)

/TrackAGET　(TrackAGET的原始数据放在这里)

/TrackVMM　(TrackVMM的原始数据放在这里)

/idle.dat　(用于在GUI界面上设置路径用的空文件)



2019年10月7日 星期一

2. 在GUI界面上，选择 黄色的 “Set Data-Structure Path”， 然后选择上面的 idle.dat文件。

3. 然后选择 “Read binary data to Raw-root File”。

这一步就开始将原始的二进制数据读入，并保存为Raw-root格式的root文件。

Raw-root格式及转换是在 CombineData.h 里定义的。 整个读取过程没有任何的cut条件。只是单纯的将二进制文件读入。对于 RICH 和 Track-AGET 同时生成的还有Pedestal文件 ped.root。

RICH 和 Track-AGET 的 Raw-root 格式为 TTree->Branch:

————Int event: 事例号/trigger号

————Int board: 板号

————Int chip: 芯片号

————Int channel: 通道号

———— vector<double> wave: 波形

Track-VMM 的 Raw-root 格式为 TTree->Branch:

————Int event: 事例号/trigger号

————Int board: 板号

————Int chip: 芯片号

————Int channel: 通道号

———— Short PDO: 电荷值

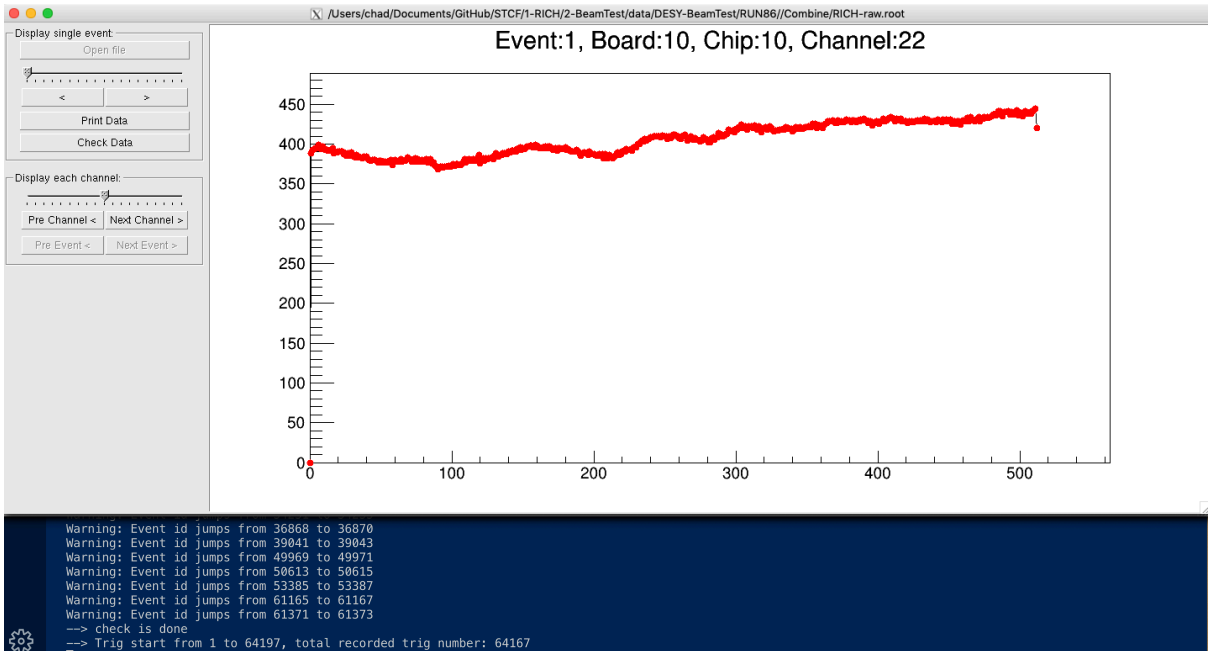
———— Short BCID:

———— Short TDO: 和BCID组成时间信息，暂时不用

生成Raw-root的这一步，可以使用./data/DESY-BeamTest/ConvertRawRoot.C 来进行批量处理。

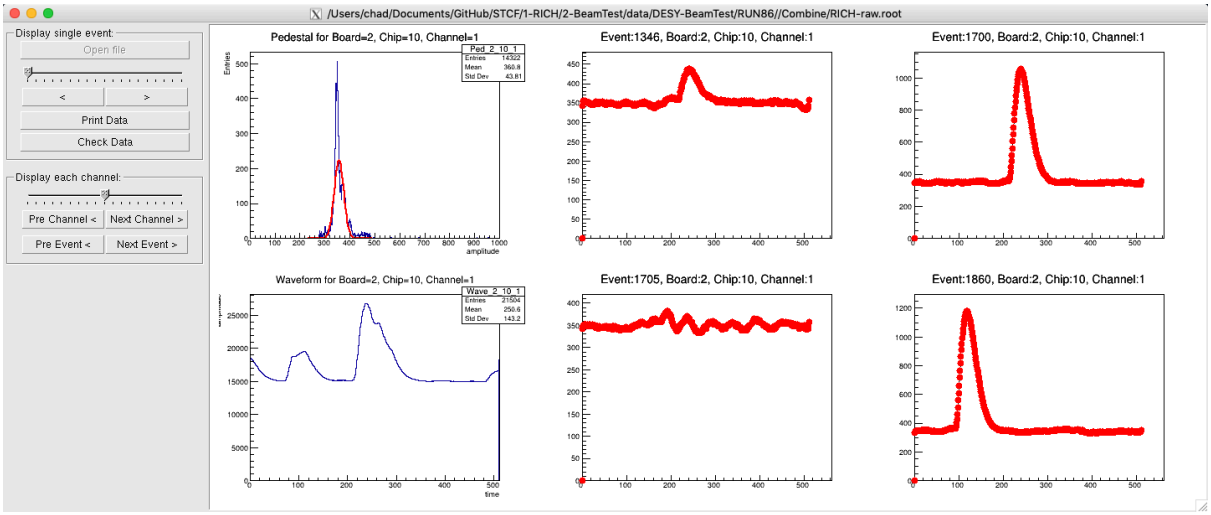
4. 此时可以检查一下这个原始数据，点击 “Check the Raw-root File”，选择想要检查的RICH/Track-AGET/Track-VMM即可。

在新弹出的GUI界面上，左上区域为单个事例显示的控制按钮。点击“check data”还可以显示触发号不连续的事例统计结果。



左下区域为按通道显示的控制按钮。下图中显示的是board=2，chip=10，channel=1的通道，给出了pedstal的分布，所有波形叠加后的分布，以及在不同的event里这个通道的信号波形图。

此功能也可以单独调用：./data/checkRawRoot.C 来查看Raw-Root文件。



5. 选择 “Generate the DST-root File” / destination data

这一步将三个系统的Raw-Root文件读入后，以探测器个体为单位，分别将数据进行处理后，另存为一个新的DST-root格式的root文件。DST-root格式定义为一个 MyBeamTestData 类，定义在 MyBeamTestDetector.h 中。转换在 MyBeamTest.cpp 里实现。

对于RICH/Track-AGET的数据，会调用 MyBeamTestData -> Analysis 函数来分析波形，根据 pedestal 文件 ped.root 来得到 Q 和 T。对于Track-VMM，其数据里直接给出了 Q 和 T，因此不再需要分析。目前是在MyBeamTest.cpp里调用了Analysis函数，用0～150时间计算ped，200～300来求最大值。

此外，对于每个系统，还会调用每个系统的 AnalysisCluster 函数。在三个文件中 MyBeamTestRICH.cpp / MyBeamTestTrackAGET.cpp / MyBeamTestTrackVMM.cpp 均有这个函数的实现。在这个函数里进行了 Map 对应，Cluster 簇的寻找。

完成后，会对每个探测器系统生成一个DST-root文件。例如 Track-AGET 有三个探测器，就会生成 T02-dst.root, T03-dst.root 和 T06-dst.root 三个文件。这里的名字及分类是根据用户在GUI界面上设置的名字及板号来进行区分的。

```
class MyBeamTestData
{
public:
    int id = -1; //探测器的id号
    int event = -1; //触发事例号
    vector<UShort_t> board;
    vector<UShort_t> chip;
    vector<UShort_t> channel;
    vector<vector<double>> wave; //一次event有多个hit，这里保存每个hit的q/t/ped/wave
    vector<double> charge;       //用Analysis计算这个Q/T
    vector<double> time;
    vector<double> pedeMean;
    vector<double> pedeRms;

    vector<pair<double, double>> hit; //这是mapping后的探测器阳极板对应的channel号（不是FEE电子学的channel号）

    //将hit进行分簇branch
    vector<vector<BeamHit>> branch;  //Pad读出：将一次event的hit分簇，branch.size()就是每个的cluster_size
    vector<vector<BeamHit>> Xbranch; //条读出：X/Y的hit分簇，branch.size()就是每个的cluster_size，Xbranch给出X坐标，所以表示Y的second为-999
    vector<vector<BeamHit>> Ybranch; //条读出：X/Y的hit分簇，branch.size()就是每个的cluster_size，Ybranch给出Y坐标，所以表示X的first为-999

    //将branch按重心得到击中信息
    vector<RealBeamHit> cluster; //Pad读出，分簇后按照重心得到的三维真实坐标的击中信息
    vector<RealBeamHit> Xcluster; //条读出，分簇后按照重心得到的三维真实坐标的击中信息
    vector<RealBeamHit> Ycluster; //条读出，分簇后按照重心得到的三维真实坐标的击中信息
```

数据结构的定义很直观，这里只简要说明：

1. hit 包含的是这次事例的击中位置信息，此位置信息为 mapping 之后的阳极条编号，不再是电子学的channle号。因此可以在此基础上换算得到真实坐标信息。

2019年10月7日 星期一

2. branch 是pad读出后分cluster的结果，是将hit按击中位置分组得到的。系统里只有RICH是pad读出，所以RICH的结果保存在这里。

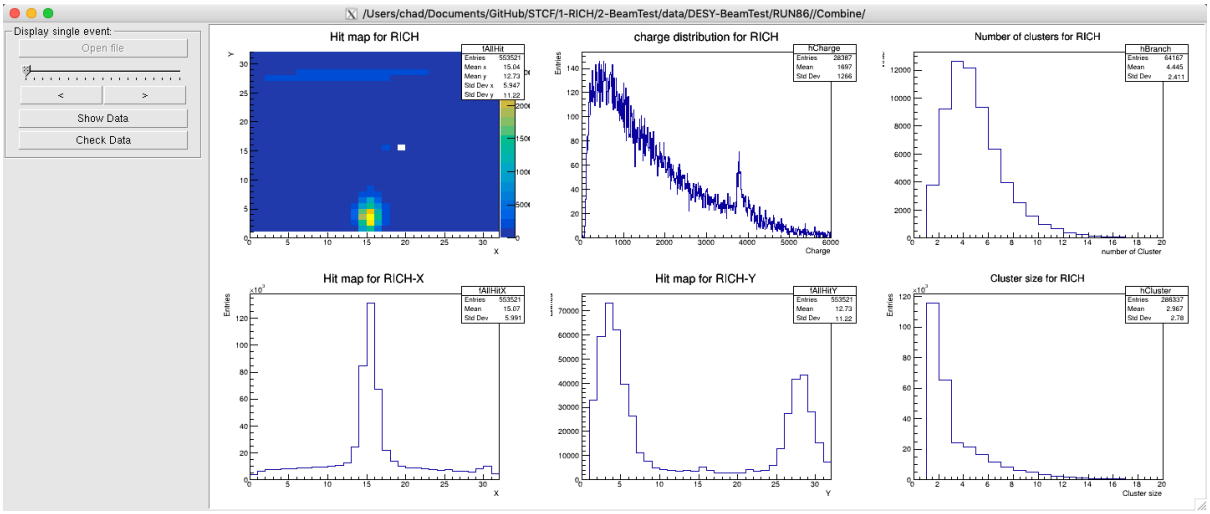
3. Xbranch/Ybranch 是条读出后分cluster的结果，也是按hit击中位置分组得到的。Track-AGET / VMM 是条读出，因此分簇后的结果保存在这里，类型为BeamHit。

4. Cluster 和 Xluster / Ycluster 是按照重心法分析上面的branch后得到的击中位置，类型为 RealBeamHit。

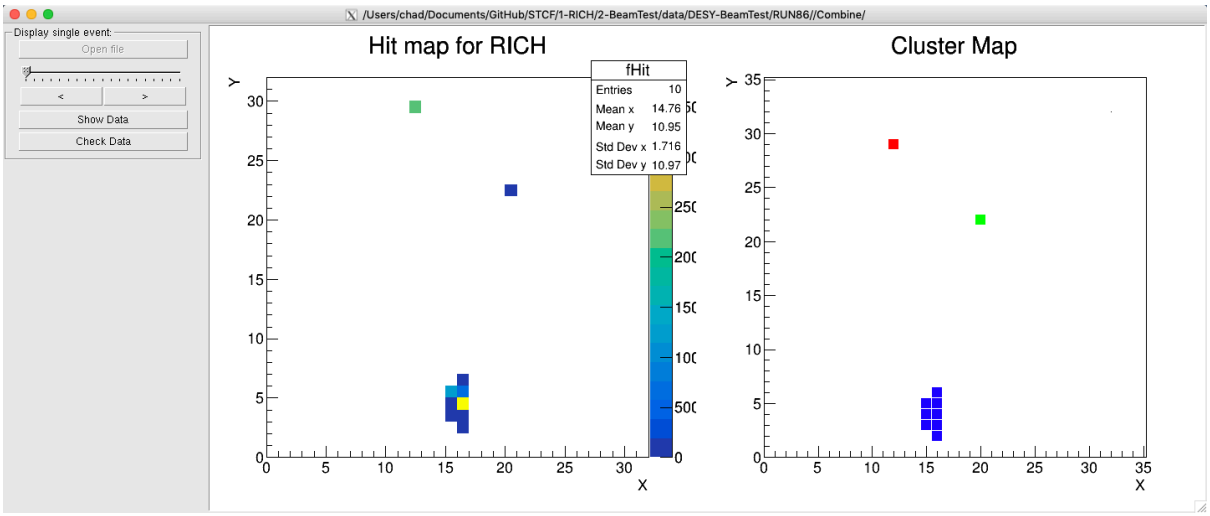
```
//----------------------------
// hit击中信息合并处理后的击中数据结构
struct BeamHit //hit的击中channel坐标及q值
{
    int id;
    double q;
    double t;
    pair<double, double> hit;
};

struct RealBeamHit //cluster的中心值及q值
{
    int id;
    int nhit;
    double q;
    double t;
    double hit[3];
    double hiterr[3];
};
```

6. 此时可以检查一下这个 dst-root 数据，点击 “Check the DST-root File”，选择想要检查的 dst-root 文件即可。

首先输出的是几种位置，cluster大小，电荷分布等总体信息。这个信息也可以通过选择 “Show Data” 重新显示。



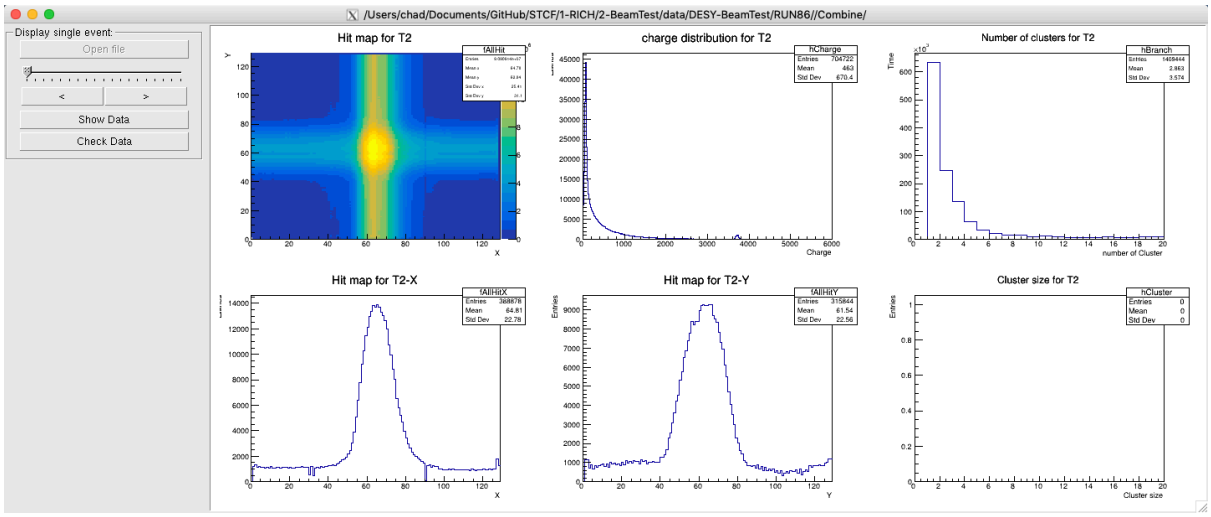
点击左右箭头，就会显示单个事例的结果。这里左边为按 Q 填图的结果，不同颜色代表电荷 Q 的大小；右边为分cluster的结果，不同颜色代表不同的cluster。

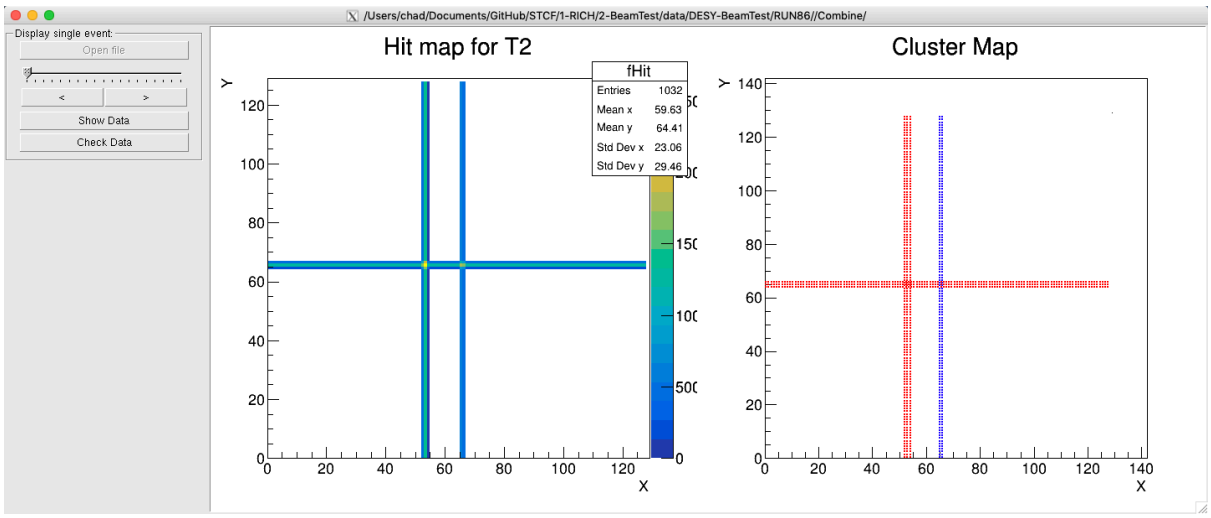


此功能也可以单独调用：./data/checkDSTRoot.C 来查看DST-Root文件。

2019年10月7日 星期一

以下是Track-AGET条读出时候的结果。





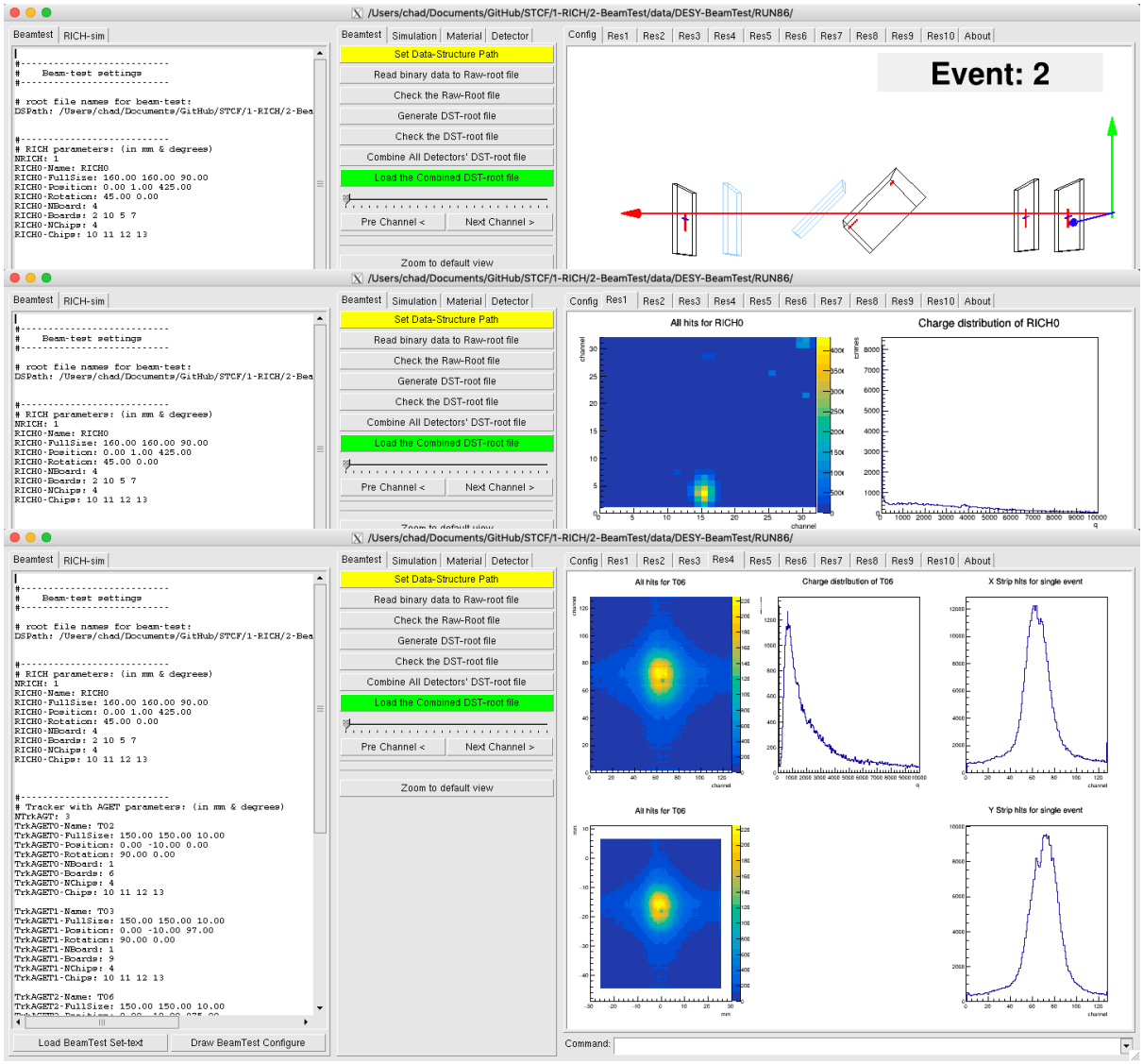
7. 选择 “Combine All Detectors’ DST-root File” 来合并所有事例

读入各系统的DST-root文件，按照trigger号来进行合并。由于在束流实验期间，所有的数据都是触发号reset后才开始取数，因此触发号一定是从接近0的数开始（不排除有些事例信号没有过阈，导致某些触发号没有数据）。另外，触发号到65535后会从0再次开始。

合并后生成的只有一个 Combine-DST.root 文件。Combine-DST.root 格式定义为一个 MyBeamTestHitData 类，定义在 MyBeamTestDetector.h 中。转换在 MyBeamTest.cpp 里实现。

这里除了合并事例外，其他的任何处理都没有做。从现在开始，数据需要进行多次处理，会单独用一个程序再来分析。程序为：./data/checkCMBRoot.C

这里同时也会给出3D的击中分布图，以及二维的击中分布等。



2019年10月7日 星期一

8. 通过运行./data/checkCMBRoot.C 来进行进一步分析

后续的分析较为繁琐且特殊，因此进一步开发可以在这里完成。