Topological reconstruction

speedup plans

David Meyhöfer

September 18, 2017

Universität Hamburg

Introduction

Goal

Use topological reconstruction method to gain knowledge on:

- distinctive features of (muon) tracks
- differential energy loss $\frac{\mathrm{d}E}{\mathrm{d}x}$

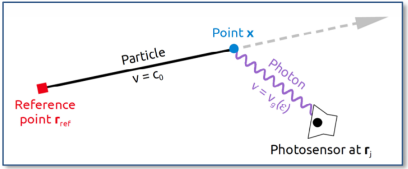
Gain:

- GeV: spatially more confined vetoes
- MeV: help to discriminate background (e.g. e^+/e^-)

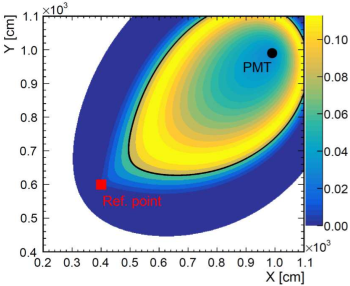
How to achieve this

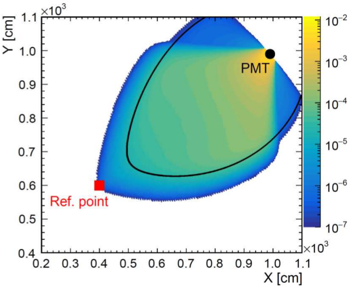
Assume simple model:

- Clear defined **reference point** and **time**
- p.d.f. for timing uncertainty and scintillation process



$$t(\mathbf{x}) = t_{\text{ref}} \pm \underbrace{\frac{|\mathbf{x}-\mathbf{r}_{\text{ref}}|}{c_0}}_{\text{particle}} + \underbrace{\frac{|\mathbf{r}_j-\mathbf{x}|}{v_g(\epsilon)}}_{\text{photon}}$$





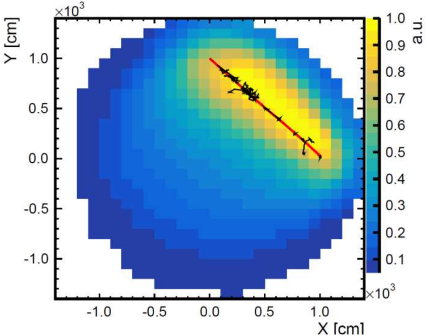
Reconstruction process

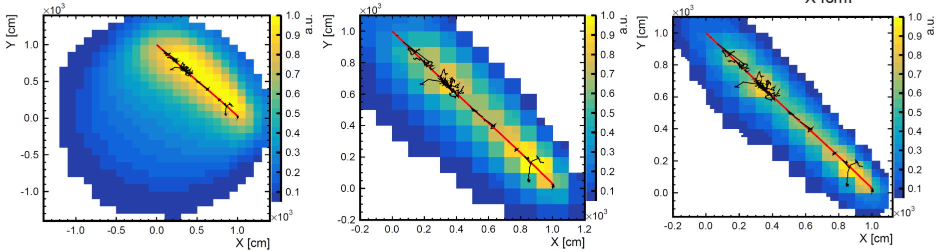
Putting **all hits** from **all PMTs** together:

- determine number density distribution of photon emission

Photon emission is **correlated**:

- use result as prior information for iterative process





Enhancements

Reconstruction expansions and plans:

- JUNO adaptation
- Borexino adaptation
- Vertex reconstruction
- μ,*e* separation
- π^0 discrimination
- Scattered light determination
- **speed up**

Time that was needed

MeV range

- Around 2 to 8 hours

GeV range

- Around 1 to 5 days

- Expected μ rate is **3 Hz**

$\Rightarrow$ Reconstruction time needs to come down!

Current status

What has been done:

- Analysing the code and performing preliminary changes
- Implementation of raw reconstruction
- Implementation of resolution based PMT consideration
- Implementation of scattered light identification
- CPU parallelization

Future plans:

- GPU parallelization
- Unification of expansions

Analysing the software

gprof

- An easy to use tool most Linux systems have
- Add -pg while compiling
- Run the program
- Use gprof with the program and the extra generated .out file

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  us/call  us/call  name
 66.67      0.02     0.02    54790     0.37     0.37  cv::Mat::release()
 33.33      0.03     0.01      332    30.12    69.49  detectAndDraw(cv::M
  0.00      0.03     0.00     4325     0.00     0.00  cv::_InputArray::in
  0.00      0.03     0.00     3323     0.00     0.00  cv::Size_<int>::Siz
  0.00      0.03     0.00     2992     0.00     0.00  cv::_InputArray::~_
  0.00      0.03     0.00     1992     0.00     0.00  cv::MatStep::operat
  0.00      0.03     0.00     1992     0.00     0.00  cv::MatStep::operat
  0.00      0.03     0.00     1660     0.00     0.00  cv::_InputArray::_I
  0.00      0.03     0.00     1332     0.00     0.00  cv::_OutputArray::~
  0.00      0.03     0.00     1331     0.00     0.00  cv::_InputArray::_I
  0.00      0.03     0.00     1328     0.00     0.00  cv::_OutputArray::_
  0.00      0.03     0.00     1002     0.00    19.60  cv::Mat::~Mat()
  0.00      0.03     0.00      999     0.00     0.00  __gnu_cxx::__normal
  0.00      0.03     0.00      916     0.00     0.00  std::_Vector_base<c
  0.00      0.03     0.00      750     0.00     0.00  __gnu_cxx::__normal
  0.00      0.03     0.00      750     0.00     0.00  void std::_Destroy_
  0.00      0.03     0.00      750     0.00     0.00  std::_Vector_base<c
  0.00      0.03     0.00      750     0.00     0.00  void std::_Destroy<
  0.00      0.03     0.00      750     0.00     0.00  void std::_Destroy<
  0.00      0.03     0.00      673     0.00     0.00  cvRound(double)
  0.00      0.03     0.00      670     0.00     0.00  cv::MatSize::MatSiz
  0.00      0.03     0.00      670     0.00     0.00  cv::MatStep::MatSte
  0.00      0.03     0.00      667     0.00     0.00  __gnu_cxx::new_allo
  0.00      0.03     0.00      667     0.00     0.00  __gnu_cxx::new_allo
  0.00      0.03     0.00      667     0.00     0.00  std::allocator<cv::
  0.00      0.03     0.00      667     0.00     0.00  std::allocator<cv::
  0.00      0.03     0.00      667     0.00     0.00  std::_Vector_base<c
  0.00      0.03     0.00      667     0.00     0.00  std::_Vector_base<c
  0.00      0.03     0.00      667     0.00     0.00  std::_Vector_base<c
```

After code change

⇒ Result: The program runs about 20% faster

Raw reconstruction iteration

- Not directly for speed up
- Stabilize the reconstruction at the beginning
- Only uses time information

Because only the time information is used:

$\Rightarrow$ less computation

$\sim$ 10% speed up

- Win win situation by expanding software

Information selection

- Directly connected to speedup
- Only use event information that is needed
- Similar results in less time
- Applicable for same detector architectures

How it works:

$\Rightarrow$ Determine minimum information needed for resolution wanted

$\Rightarrow$ Find set of PMTs representing the event and minimal information

$\Rightarrow$ Do reconstruction faster

Preliminary result:

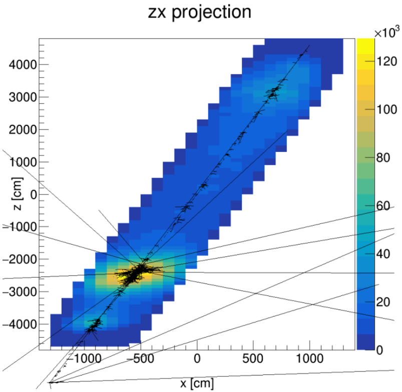
- A reconstruction that took a 1 day now takes $\sim$20 minutes
- Reconstruction about **72 times faster** than before

Scattered light identification

Adjusting the reconstruction for a specific task:

- 40 GeV muon
- Predefined corridor for event

- Identification of scattered light by travel time
- Result achieved with first iteration
- Speedup for this case $\sim$10 time

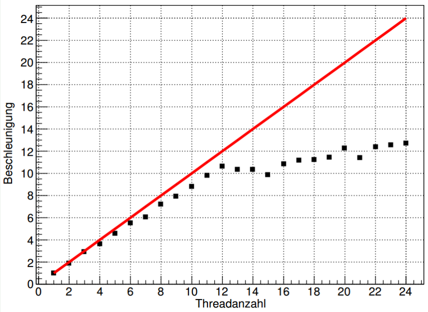


General application scattered light:

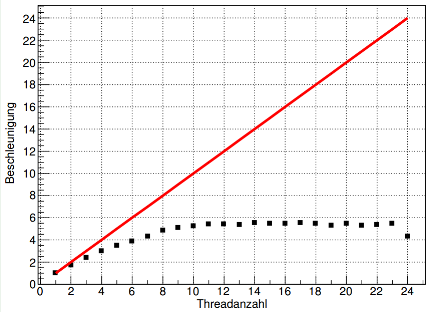
Performing a cut on scattered light:
$\Rightarrow$ Increase information selection speedup by $\sim$20%

CPU parallelization

MeV range



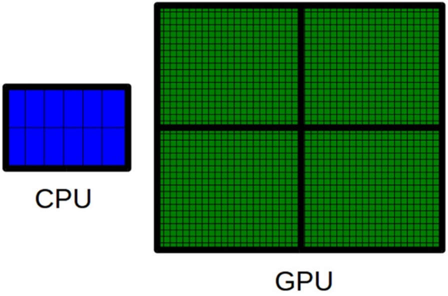
GeV range



- 12 physical cores (extra 12 by hyper threading)
- MeV range: speedup per physical CPU about 90% faster
- GeV range: speedup per physical CPU about 50% faster

GPU parallelization

- Not yet done
- Directly related to speedup
- Needs a lot of recoding work
- Cores on GPU 3000+



Expected speedup?

- Compare to CPU parallelization of +50% faster per core

$\Rightarrow$ with $\sim$3000 cores in a GPU a speedup factor 1500 maybe possible

What is possible?

Rough estimate:

$$\overbrace{\sim 1500}^{\text{GPUs}} \times \overbrace{\sim 72}^{\text{Sel. info.}} \times \overbrace{\sim 1.5}^{\text{other}} = \overbrace{\sim 162000}^{\text{total speedup factor}}$$

$\Rightarrow$ Reconstruction that took 1 day would then take $\sim$0.5 seconds

This is **close to 3 Hz** but not quite there.

Finalizing thoughts:

- Currently focus on getting the best theoretical result possible
- Optimization for specific task is still an option
- Rough estimate: rise and fall by slight adjustments of numbers!