Hall A Analysis Software Status

Ole Hansen

Jefferson Lab

Hall A Analysis Workshop
December 12, 2007

Outline

Improvements in Podd 1.4

- Workaround for “timezone bug”
- `THaOutput` performance enhancement & bug fixes
- Energy loss module `Init()` bug fix
- Support for ROOT 5.16

New Features in Podd 1.5

- Expanded database capabilities
- Extended detector maps
- Restructured helicity classes
- Support for ROOT 5.18

Available March 2008

Example Database File

Printed by Ole Hansen

Dec 11, 07 12:44 **db_B.mwdc.dat** Page 1/1

```
B.mwdc.planeconfig = u1 u1p x1 x1p v1 v1p \
                     u2 x2 v2 \
                     u3 u3p x3 x3p v3 v3p

B.mwdc.cratemap =  3     6       21      1877   500     96  \
                   4     4       11      1877   500     96  \
                   4     17      24      1877   500     96

B.mwdc.nwires       = 200
B.mwdc.u1.nwires    = 141

B.mwdc.size         = 2.0  0.5  0.0
B.mwdc.x1.size      = 1.4  0.35 0.0
```

Tuesday December 11, 2007 db_B.mwdc.dat 1/1

Code Example: Database Request

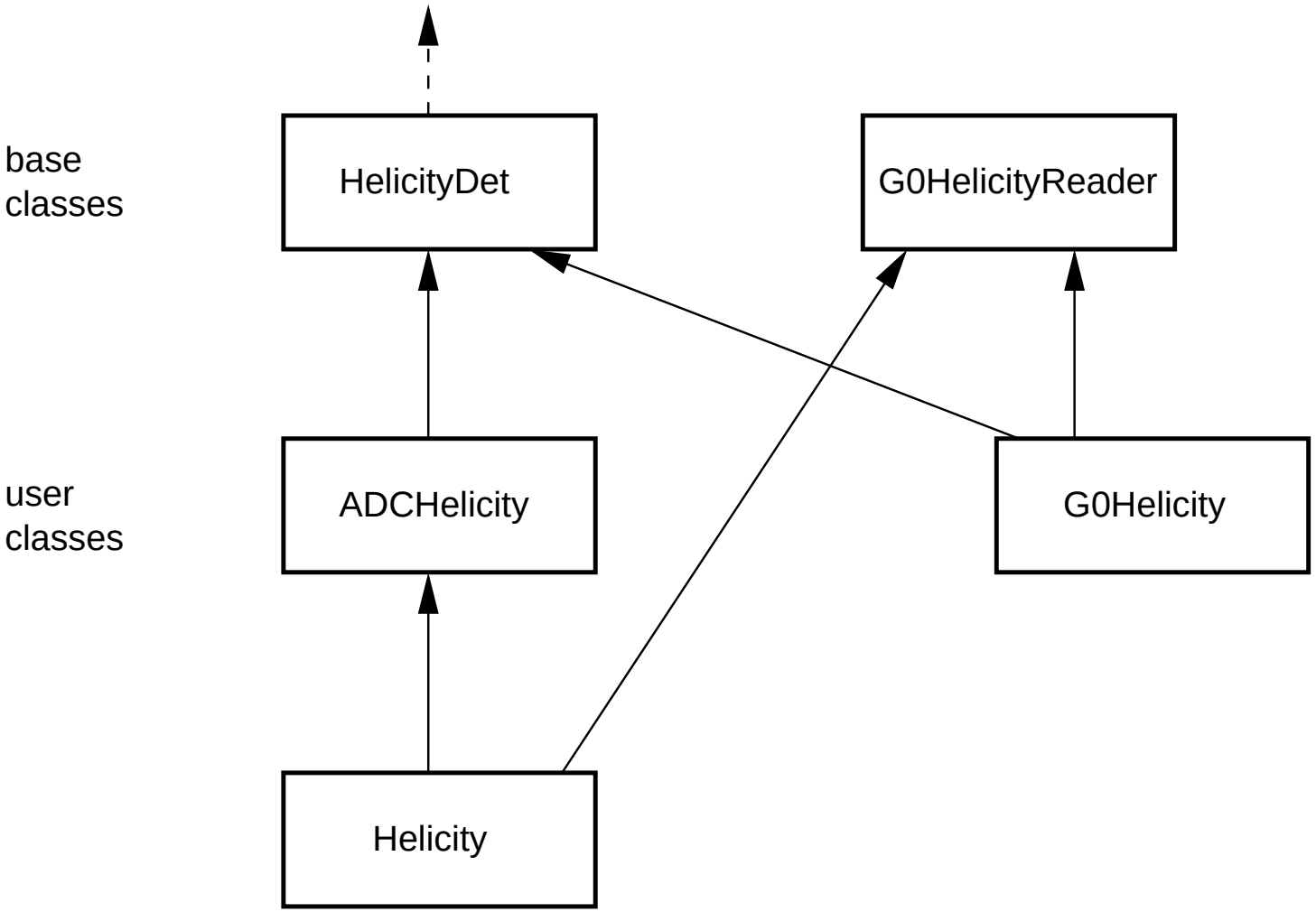
Printed by Ole Hansen

Dec 11, 07 12:46 **MWDC.cxx** Page 1/1

```
FILE* file = OpenFile( date );

vector<vector<Int_t> > *cmap = new vector<vector<Int_t> >;
string planeconfig;
Int_t time_cut = 1, pairs_only = 0, mc_data = 0;

DBRequest request[] = {
  { "planeconfig",  &planeconfig, kString },
  { "cratemap",     cmap,         kIntM,    6 },
  { "timecut",      &time_cut,    kInt,     0, 1 },
  { "pairsonly",    &pairs_only,  kInt,     0, 1 },
  { "MCdata",       &mc_data,     kInt,     0, 1 },
  { 0 }
};
Int_t err = LoadDB( file, date, request, fPrefix );
fclose(file);
```

Tuesday December 11, 2007 MWDC.cxx 1/1

New Helicity Class Hierarchy



User-driven Development

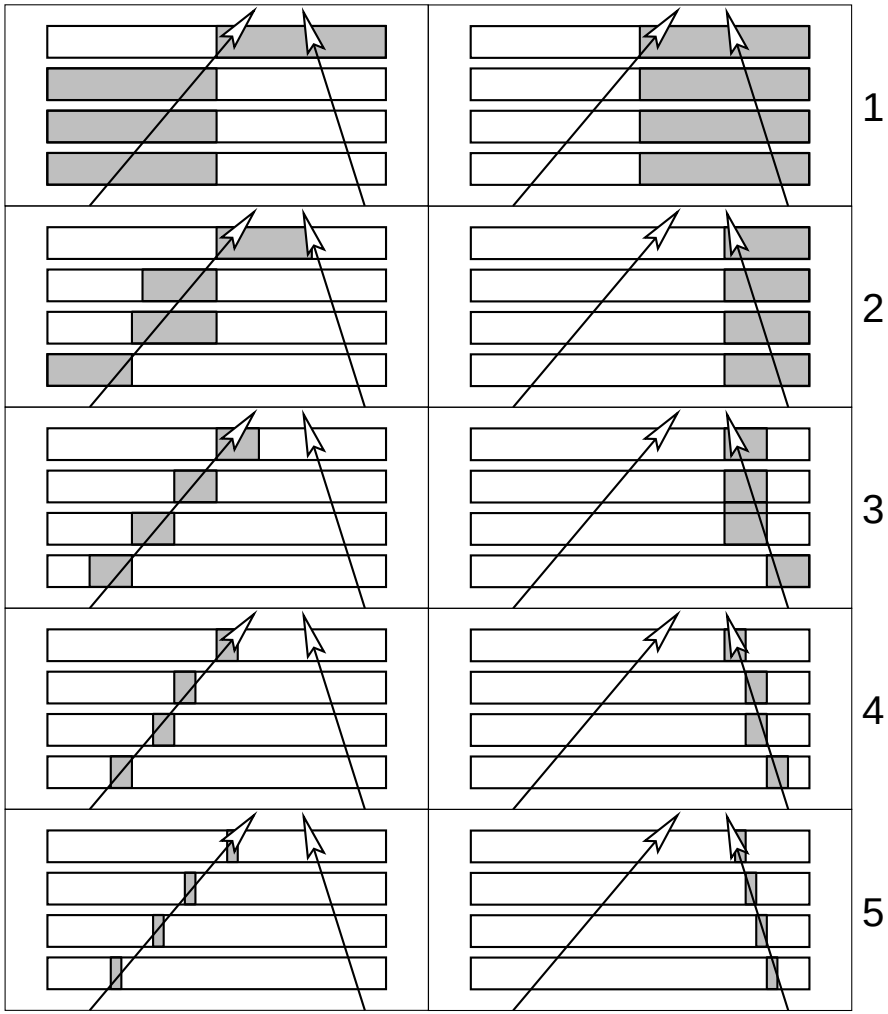
- Core analyzer reasonably mature
- Hall A staff will maintain current core analyzer
- New features expected to be user-contributed
  - Detectors, apparatuses, etc.
  - Improved algorithms

BigBite Tracking Software

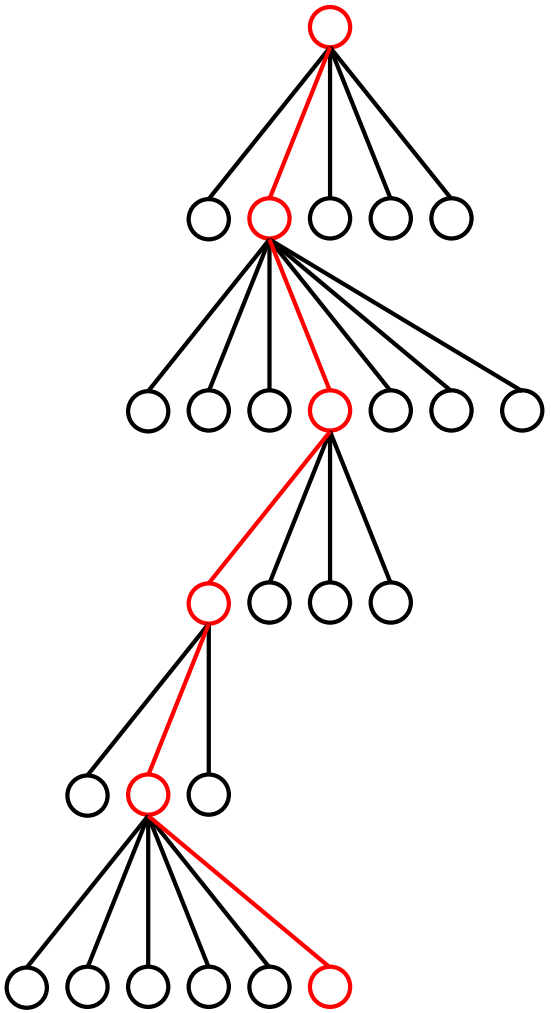
General

- Recursive template matching
- Proven at HERMES with chambers similar to BigBite's
- Fast and efficient (speed and memory)
- Best with simple geometry $\longrightarrow$ given in BigBite

Successive Approximation Method



Tree Structure



Reconstruction Flow

- Create hitpattern from raw data
- Find straight-line patterns in hitpattern by searching pattern database.
  - Per road: $\mathcal{O}(100)$ pattern comparisons.
  - Recursion depth of $\mathcal{O}(10)$
- Fit each road to find exact track.
- De-ghost (χ^2 cut, minimum spanning tree, track parameter clustering etc.)
- Combine 2D track projections to 3D tracks.
- Optional: Further track selection (e.g. shower cut)
- u, v, x projections independent $\longrightarrow$ parallelize

Pattern Recognition Routine

TreeSearch

```
Int_t Projection::TreeSearch()
{
   // Set up tree iterator and compare-to-hitpattern function object

   TreeWalk walk( fNlevels );
   ComparePattern compare( fHitpattern );

   // Visit every tree node recursively and apply compare()

   walk( fPatternTree->GetRoot(), compare );

   return 0;
}
```

Tree Iterator

Printed by Ole Hansen

Oct 09, 07 0:18 **TreeWalk.h** Page 1/1

```
template<typename Operation>
inline TreeWalk::ETreeOp
TreeWalk::operator()( Link* link, Operation& action, Pattern* parent,
                      UInt_t depth, UInt_t shift, Bool_t mirrored ) const
{
  // Traverse the tree and call function object "action" for each link.
  // The return value from action determines the behavior:
  //  kRecurse: process child nodes until reaching maxdepth
  //  kRecurseUncond: process child nodes (regardless of depth)
  //  fSkipChildNodes: ignore child nodes
  //  kError: error, return immediately

  if( !link ) return TreeWalk::kError;
  ETreeOp ret = action(NodeDescriptor(link, parent, shift, mirrored, depth));
  if( (ret == TreeWalk::kRecurse and depth+1 < fNlevels) or
      ret == TreeWalk::kRecurseUncond) {
    Pattern* pat = link->GetPattern();
    Link* ln = pat->GetChild();
    while( ln ) {
      Bool_t new_mir = mirrored xor ln->Mirrored();
      ret = (*this)( ln, action, pat, depth+1,
                     (shift << 1) + (new_mir xor ln->Shift()), new_mir );
      if( ret == TreeWalk::kError ) return ret;
      // Continue along the linked list of child nodes
      ln = ln->Next();
    }
  }
  return ret;
}
```

Tuesday October 09, 2007 TreeWalk.h 1/1

Hitpattern Comparison Function Object

Printed by Ole Hansen

Oct 09, 07 0:14 **ComparePattern.cxx** Page 1/1

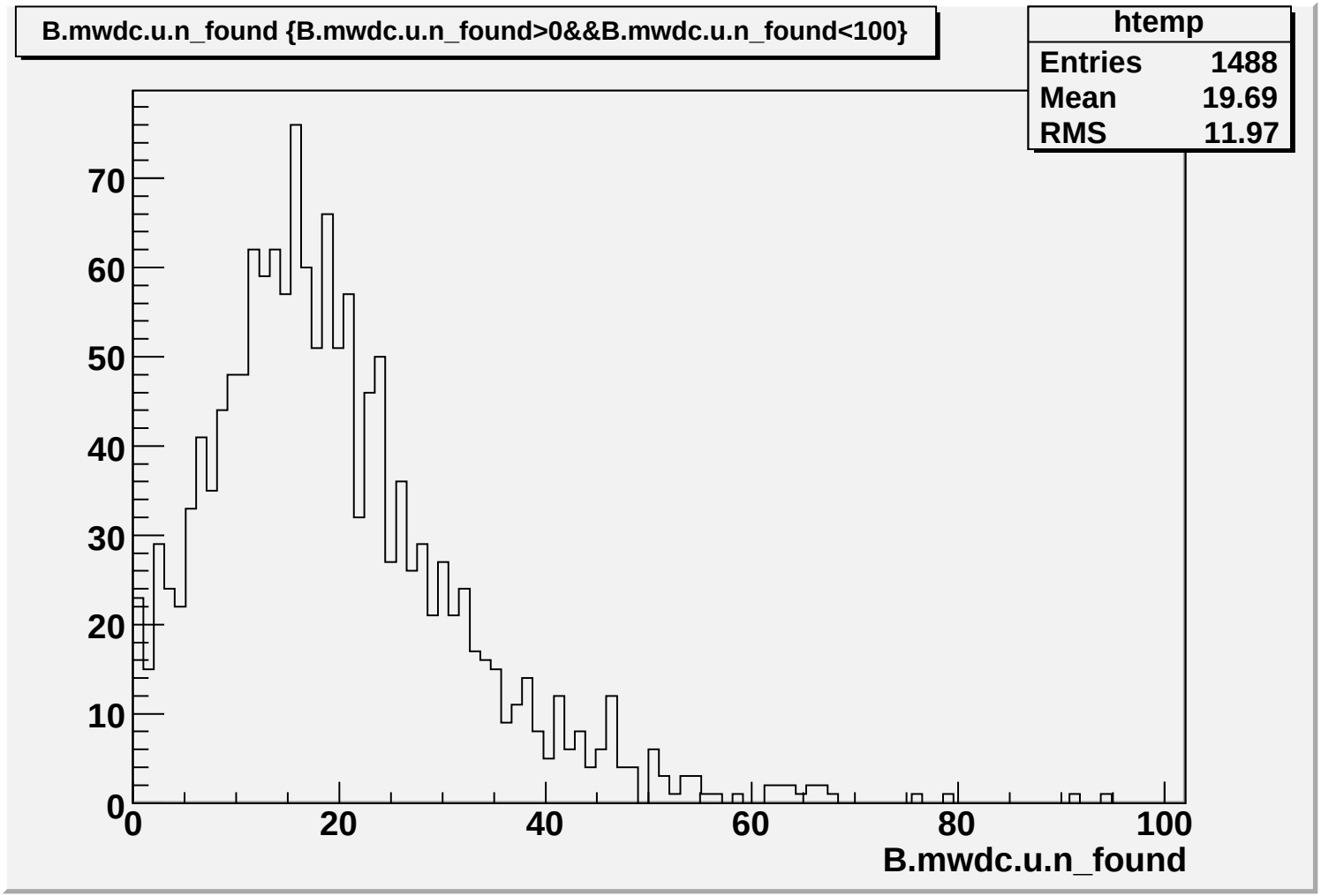
```
TreeWalk::ETreeOp
Projection::ComparePattern::operator() ( const NodeDescriptor& nd )
{
  // Test if the pattern from the database that is given by NodeDescriptor
  // is present in the current event's hitpattern

  // Match?
  fProj->n_test++;
  if( fHitpattern->ContainsPattern(nd) == fHitpattern->GetNplanes() ) {
    if( nd.depth < fHitpattern->GetNlevels()-1 )
      return TreeWalk::kRecurse;

    // Found a match at the maximum resolution: add it to the list of roads
    //TODO...
    fProj->n_found++;
  }
  return TreeWalk::kSkipChildNodes;
}
```

Tuesday October 09, 2007 ComparePattern.cxx 1/1

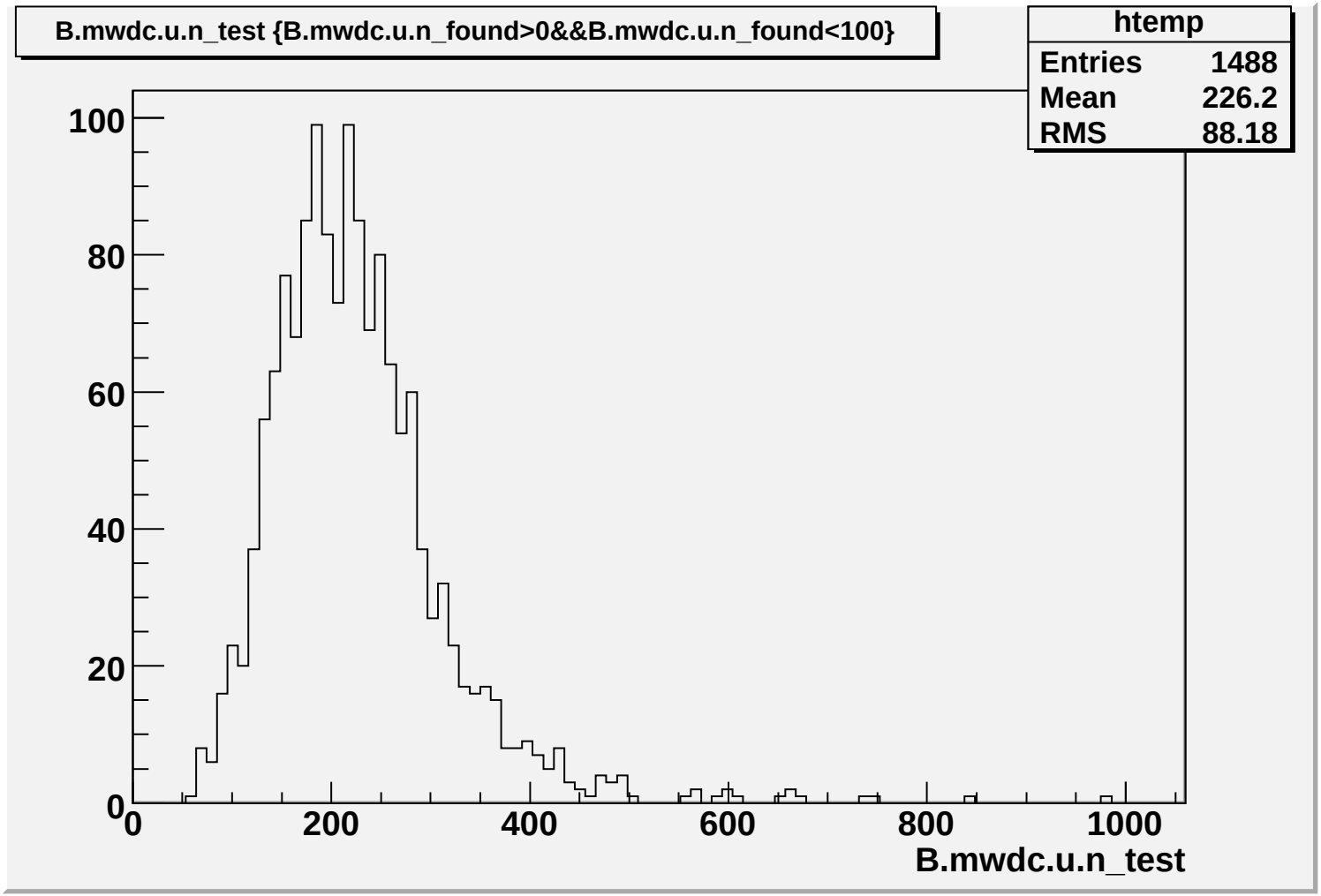
Replay Parameters

- G_E^n run 4427 (low-luminosity hydrogen)
- maxdepth = 12, maxslope = 2.5
- TDC time cut 0–150 ns; hit pairs only
- 10000 events.
- Output: 9 variables
- 32-bit 2.4 GHz Xeon, single-threaded code
- Replay rate 1600 Hz

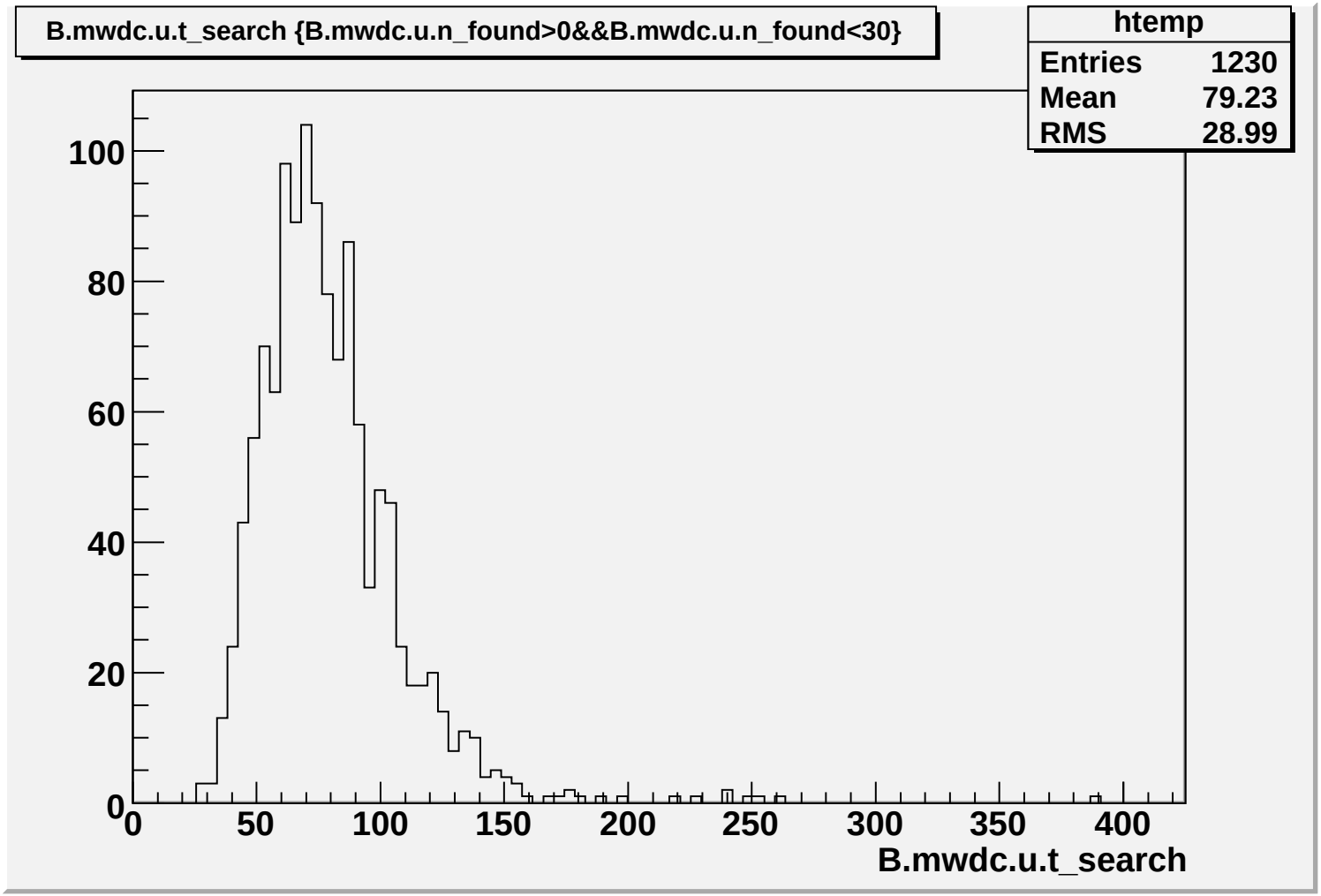
2D Tracks per Event per Projection



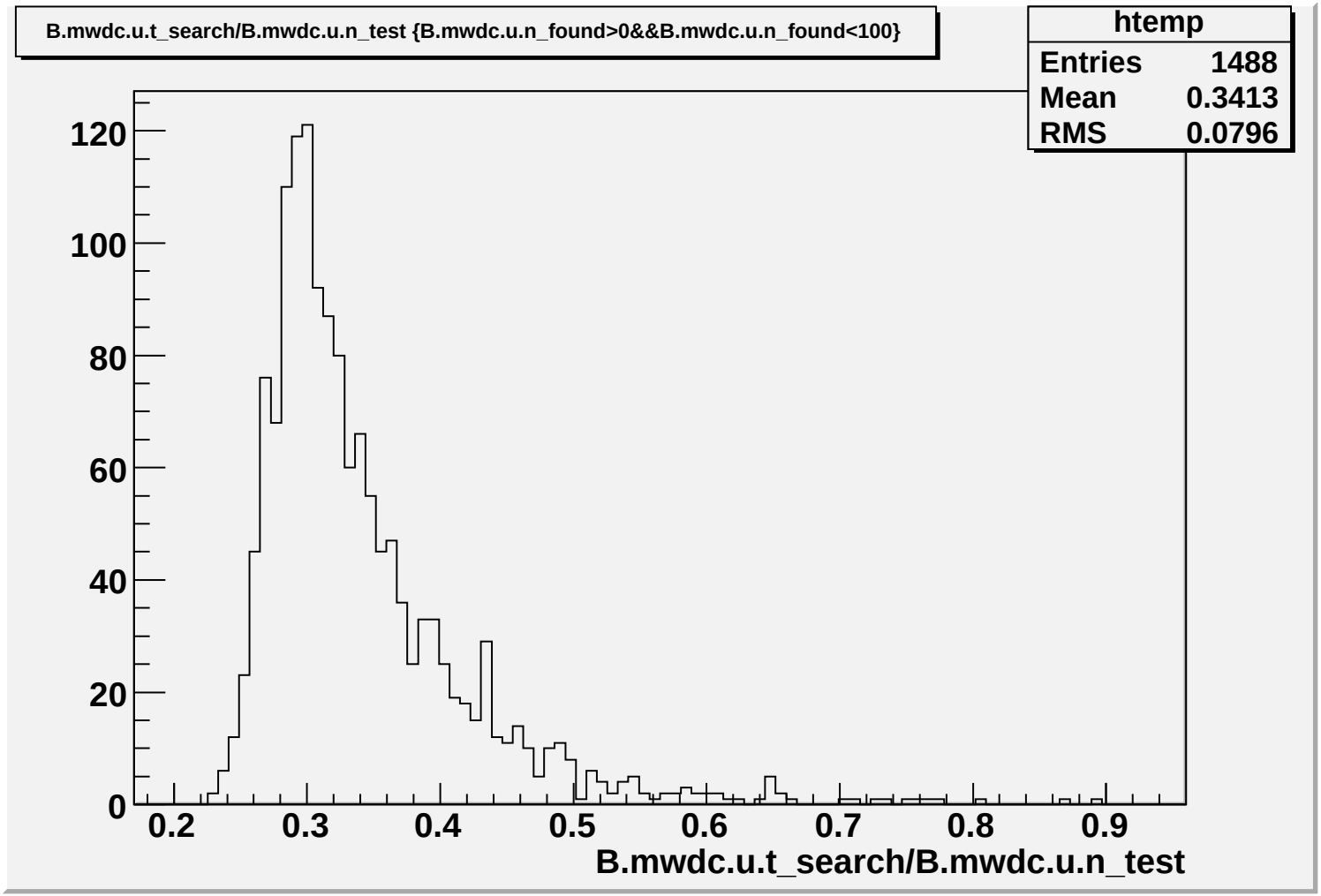
Pattern Comparisons per Event per Projection



Real Time in TreeSearch per Event (μs)



Real Time per Comparison (μs)



Summary

- C++ analysis software is working well.
- New BigBite tracking software promises much improved performance.

- Outlook
  - Analyzer could benefit from optimizations (speed, documentation, etc.)
  - BigBite software to be completed