C++ Analyzer Status Report

Ole Hansen
Hall A Data Analysis Workshop
11 December 2002

- Generalities, Program Design
- Comparison of ESPACE vs. C++ tracking algorithms
- Comparison of single-track results
- Multi-cluster/multi-track analysis in ESPACE
- To Do

The Hall A C++ Analyzer

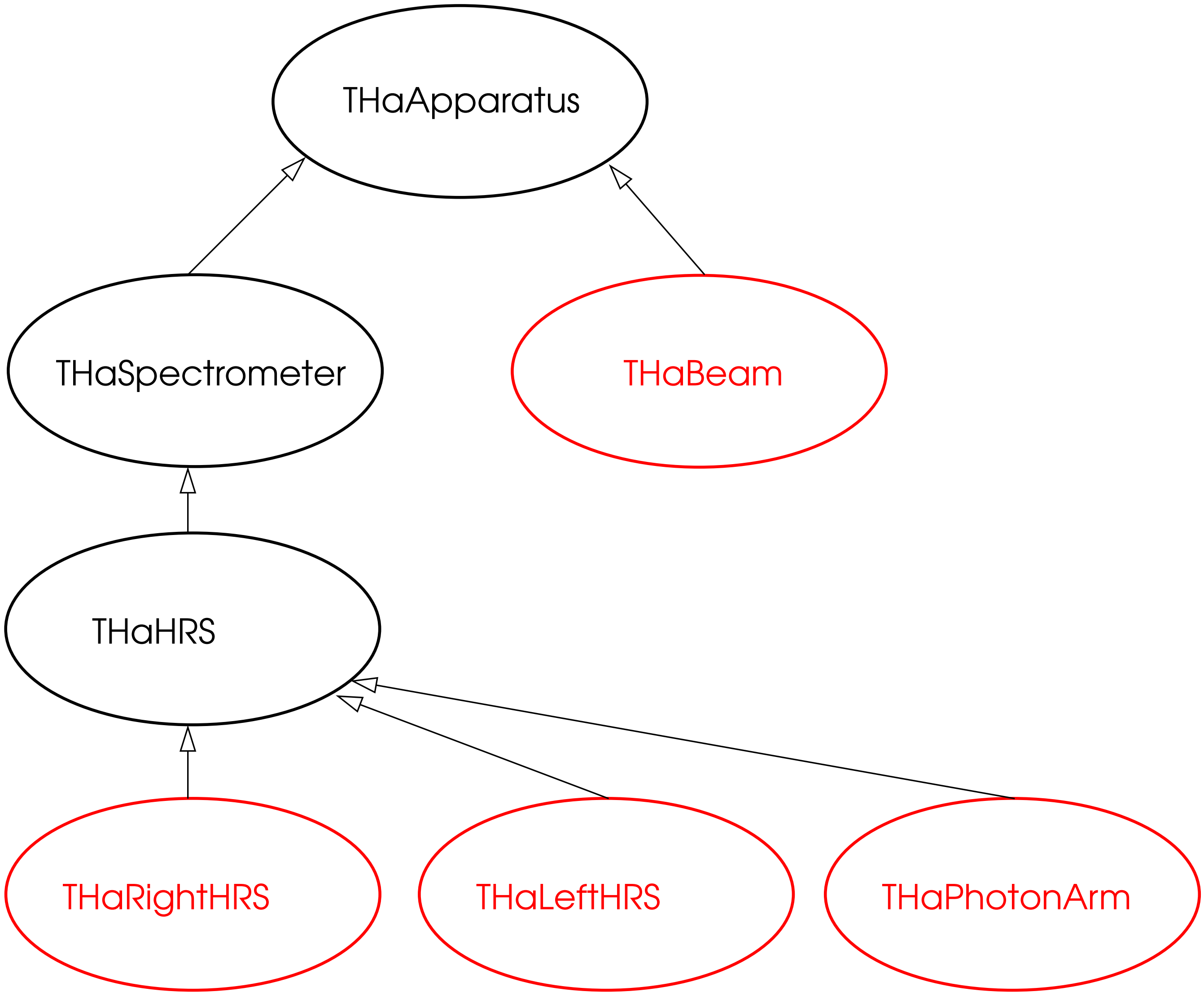
Design Guidelines

- **Flexibility:** Easy to add/remove detectors/spectrometers. Achieved through **object-oriented** programming.
- Critical **re-evaluation of algorithms**. Complete re-design, not re-write of ESPACE.

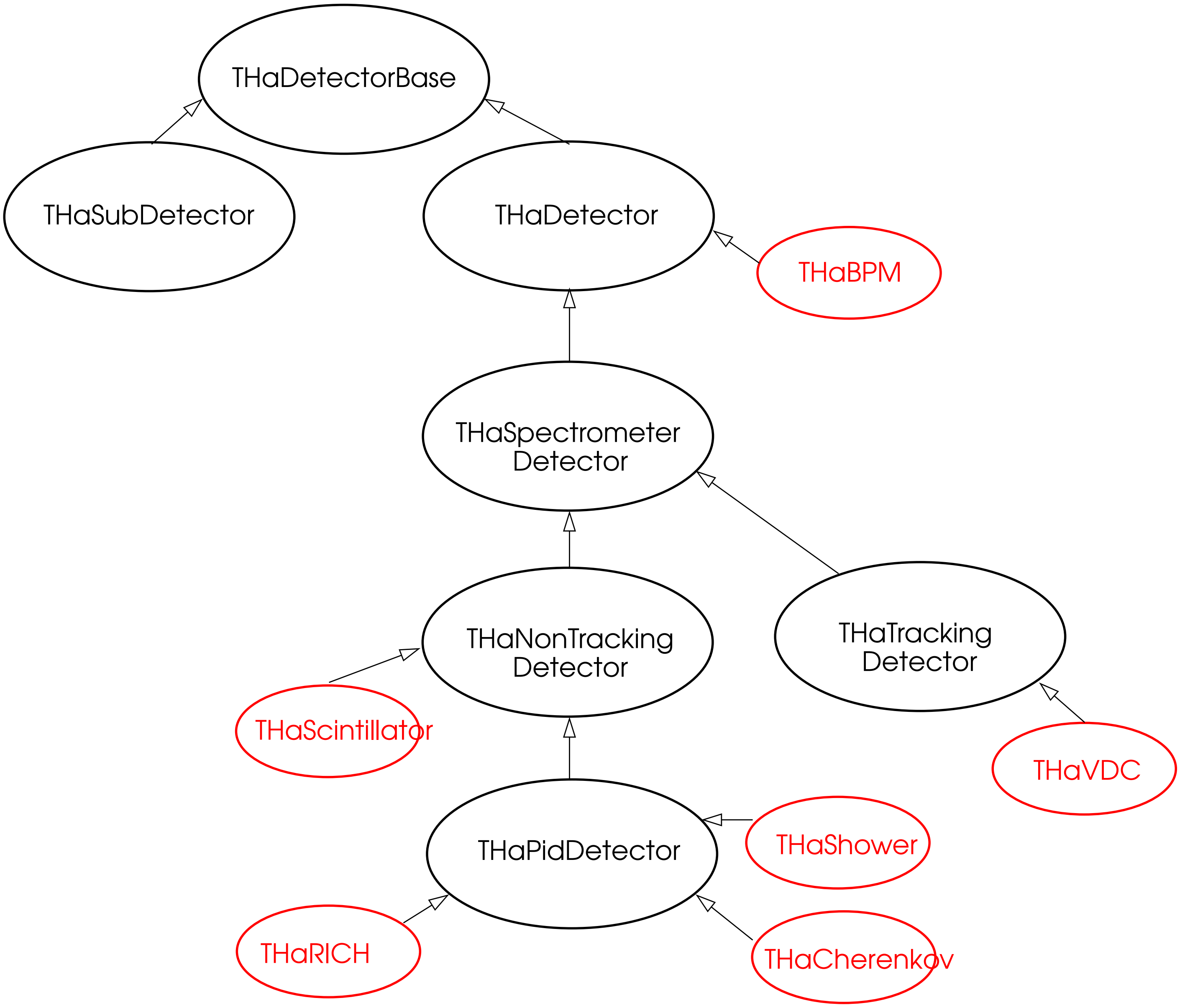
Technical Features

- Based on **ROOT**. All of ROOT available at command prompt.
- **C++ interpreter CINT:** Scripting and programming languages are identical.
- Original, custom-written symbolic variable package allowing **dynamic definition of cuts, histograms, ntuples, etc.**

Apparatus Class Hierarchy



Detector Class Hierarchy



Standard Analyzer — Concept

THaAnalyzer:

- **"Decode"** — Analyze detectors individually
  - Apply calibrations (e.g. pedestals, offsets, gains)
- **"Reconstruct"** — Analyze apparatuses individually
  - Track fitting & target reconstruction
  - Combine detectors, e.g. use scintillator data to assist tracking
- **"Physics"** — Combine apparatuses
  - Computation of kinematical quantities (Q^2, x, etc.)
  - Vertex corrections using BPM information.
- **"Output"** — Write selected data to file or shared memory (`THaOutput` class)

Standard Analyzer II – Spectrometer::Reconstruct()

The main work is done in the spectrometer reconstruction. The standard algorithm works as follows:

- For each defined THaTrackingDetector, call **CoarseTrack()**
- For each defined THaNonTrackingDetector, call **CoarseProcess()**
- For each defined THaTrackingDetector, call **FineTrack()**
- For each defined THaNonTrackingDetector, call **FineProcess()**
- Call **FindVertices().**
  Reconstructs tracks to target.
- Call **TrackCalc().**
  Calculates global track properties such as momentum and beta. Identifies "Golden Track".
- Call **CalcPID().**
  Combines individual PID results from PID detectors to overall PID probabilities for each particle type.

“Global Variables”

- Provide read-only access to data via **symbolic names.**
- Data can be scalars, fixed- and variable-size arrays.
- Data can be part of object member variables or arrays (e.g. `TClonesArray`).
- All basic data types supported.
- Essential to allow dynamic definition of histograms, ntuples, cuts.
- Accessible via global list **gHaVars**

Extending the Core Analyzer

Adding a new detector involves, in simplest case, two lines of code in the analysis script:

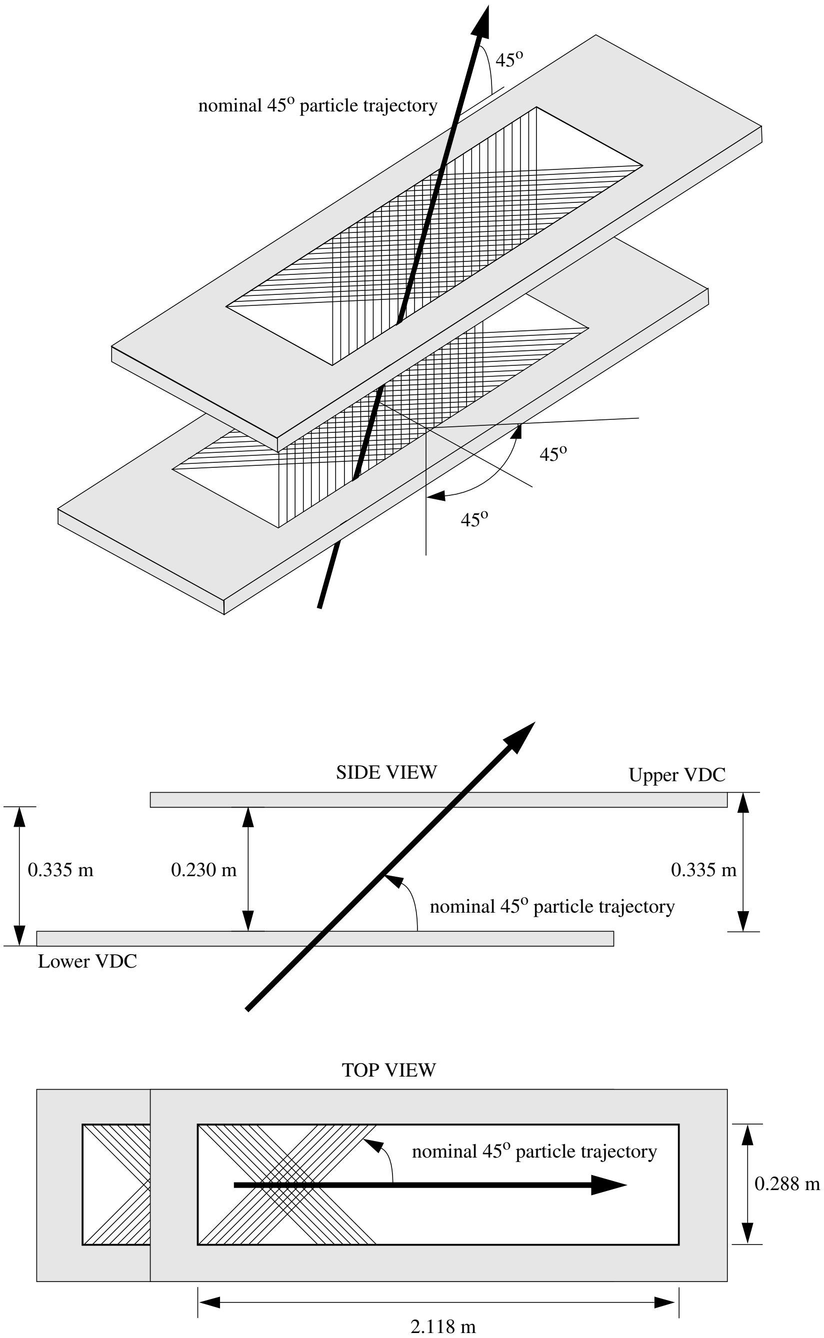
```
gSystem->Load("libKaon.so");
HRSL->AddDetector( new THaRICH("rich", "The Hall A RICH" ));
```

Of course, this is a bit deceptive . . . One still needs to:

- Write the code for the detector and compile it into a shared library.
  - Largely independent of core analyzer.
  - Might be able to inherit from existing detector.
- Create proper database entries for the detector.

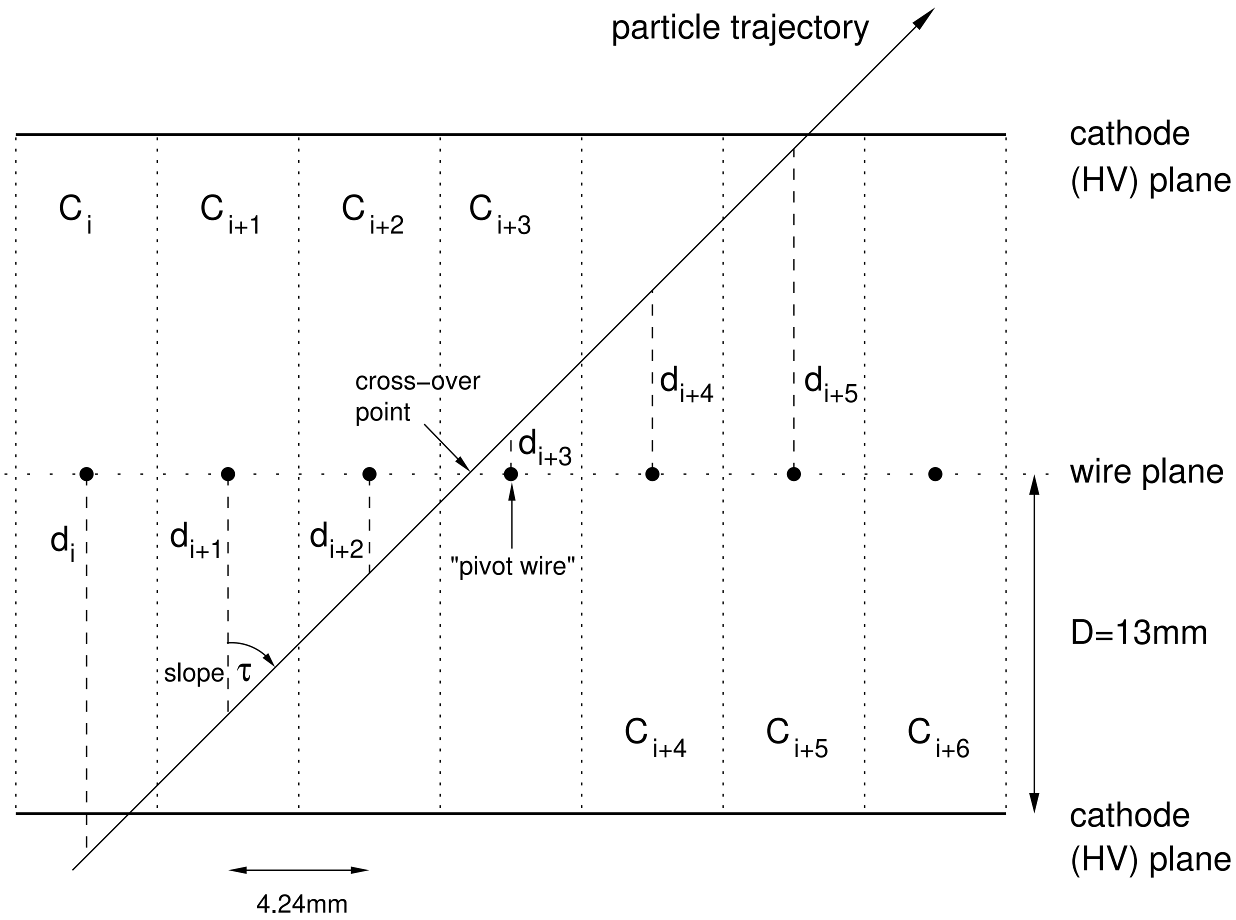
Removing detectors not as easy because of dependencies.

Tracking: VDCs



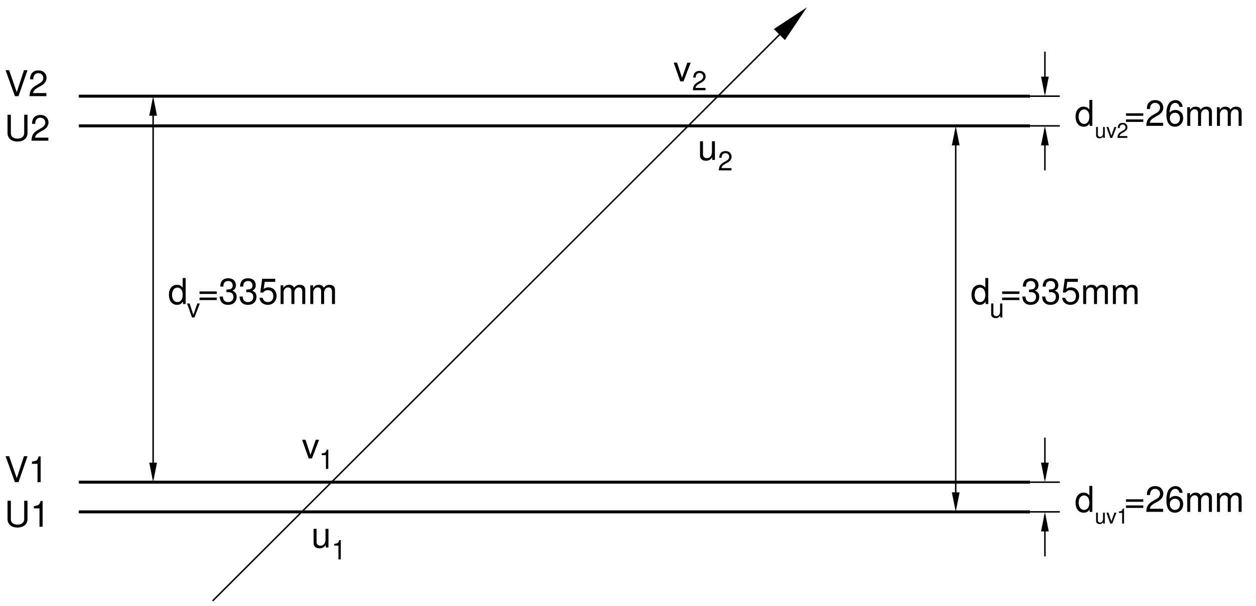
VDC Cluster

View along wires



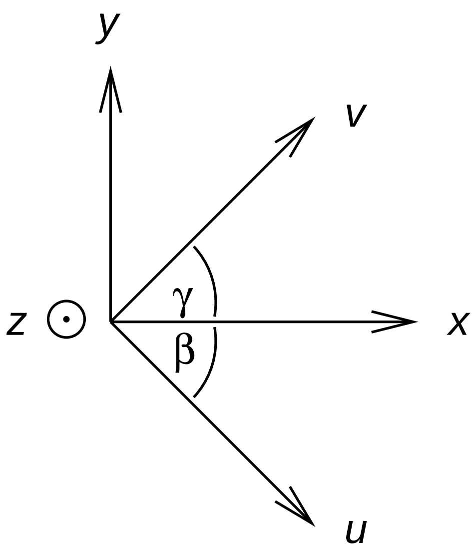
VDC Wire Planes

Side View



VDC Coordinate Transformations

Top View



Transformation to "Detector Coordinates"

$$
\begin{aligned}
x &= \frac{u \sin\gamma - v \sin\beta}{\sin(\gamma - \beta)} \\
y &= \frac{u \cos\gamma - v \cos\beta}{\sin(\gamma - \beta)} \\
\tan\theta &= \frac{\sin\gamma \tan\alpha_u - \sin\beta \tan\alpha_v}{\sin(\gamma - \beta)} \\
\tan\phi &= \frac{\cos\gamma \tan\alpha_u - \cos\beta \tan\alpha_v}{\sin(\gamma - \beta)}
\end{aligned}
$$

Tracking Algorithm

Common procedure

- Decode
- Find clusters
- Apply timing corrections
- Fit clusters
- Combine clusters from different planes
- Transform coordinates
- Reconstruct to target
- x_0 and extended target corrections

Tracking Algorithm Differences

Handling of Decoding, Cluster Finding, Timing Corrections, and Cluster Fitting for **single-cluster analysis**:

ESPACE

- Only largest TDC value (first hit) per wire
- Optional TDC cut: $-0.5D \leq t * \eta \leq 1.5D$ (not used here)
- Minimum cluster size: 2
- Maximum cluster gap: 1
- Timing corrections: ToF to S1, timewalk
- Analytic time-to-distance conversion using fitted coefficients. (Optional: lookup table; polynomial with coefficients from GARFIELD simulation.)
- Complex 2-parameter `polfit` routine to fit straight line. (Recently replaced by simpler `linefit` (PDB).)

C++

- Only largest TDC value (first hit) per wire
- Ignore negative times (after applying TDC offset)
- Hard TDC cut: $800 \leq TDC \leq 2200$
- Maximum cluster gap: 1
- No ToF & timewalk corrections (yet)
- Analytic time-to-distance conversion using fitted coefficients.
- Textbook 2-parameter linear fit.

Cluster combination and coordinate transformations for **single-cluster analysis**:

ESPACE

$$
\begin{aligned}
\tan\alpha_u &= \frac{u_2 - u_1}{d_u} && \text{global } u \text{ angle} \\
\tan\alpha_v &= \frac{v_2 - v_1}{d_v} && \text{global } v \text{ angle} \\
u &= u_1 \\
v &= v_1 - d_{uv1} \tan\alpha_v
\end{aligned}
$$

C++

$$
\begin{aligned}
\tan\alpha_u &= \tau_u && \text{local } u \text{ angle} \\
\tan\alpha_v &= \tau_v && \text{local } v \text{ angle} \\
u &= u_{1,2} \\
v &= v_{1,2} - d_{uv1,2}\tau_v
\end{aligned}
$$

Now convert to local (x, y, θ, ϕ) ("UV track"). Then,

$$
\begin{aligned}
x &= x_1 \\
y &= y_1 \\
\tan\theta &= \frac{x_2 - x_1}{d_u} \\
\tan\phi &= \frac{y_2 - y_1}{d_u}
\end{aligned}
$$

Cluster Combination & Coordinate Transformation

ESPACE

$$U1, U2 \longrightarrow (u_1, \alpha_u)$$

$$\longrightarrow (x, y, \theta, \phi)_{det}$$

$$V1, V2 \longrightarrow (v_1, \alpha_v)$$

C++

$$U1, V1 \longrightarrow (x, y, \theta, \phi)_{lower}$$

$$\longrightarrow (x, y, \theta, \phi)_{det}$$

$$U2, V2 \longrightarrow (x, y, \theta, \phi)_{upper}$$

Target Reconstruction & Final Corrections

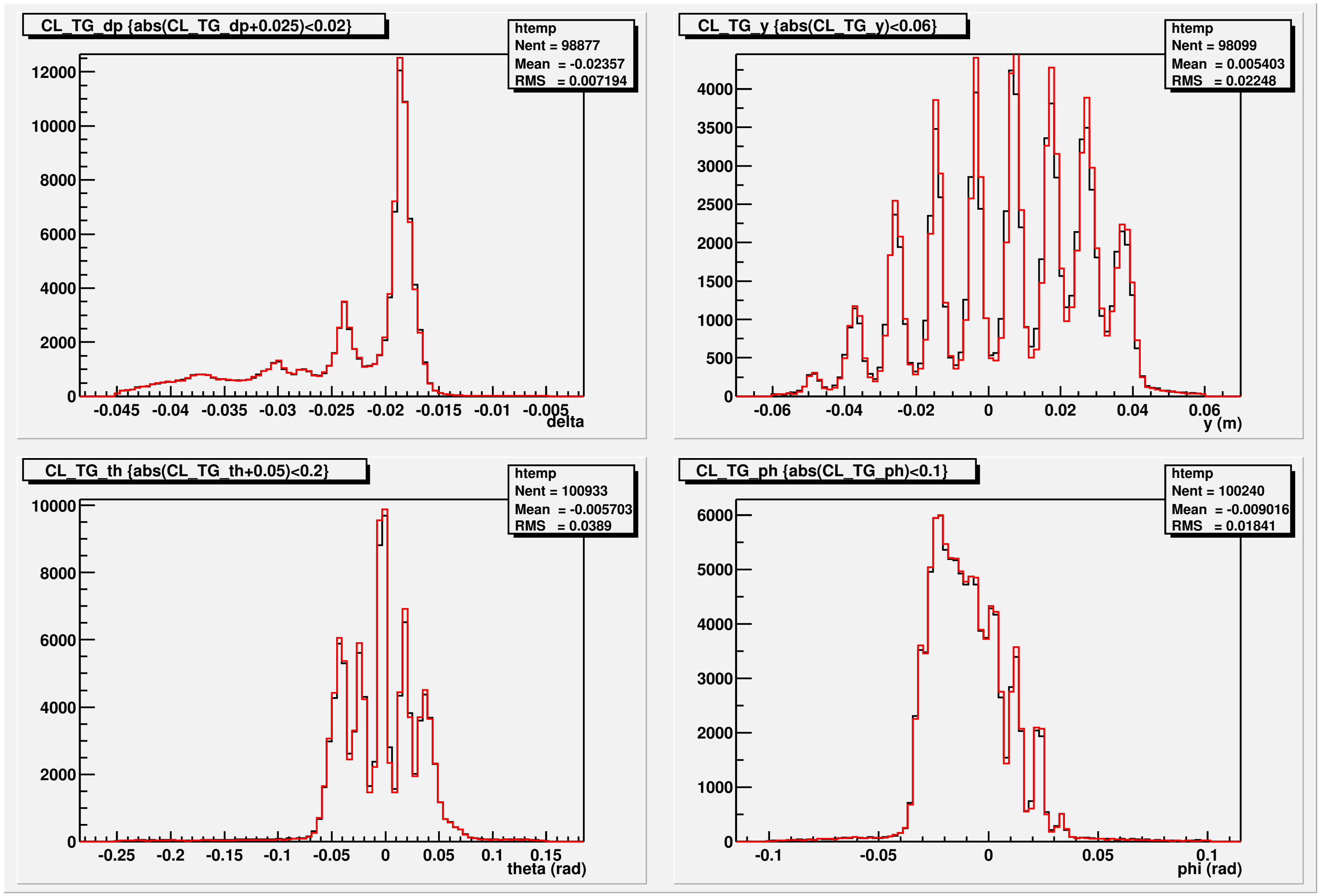
Common to both programs (exactly the same algorithm)

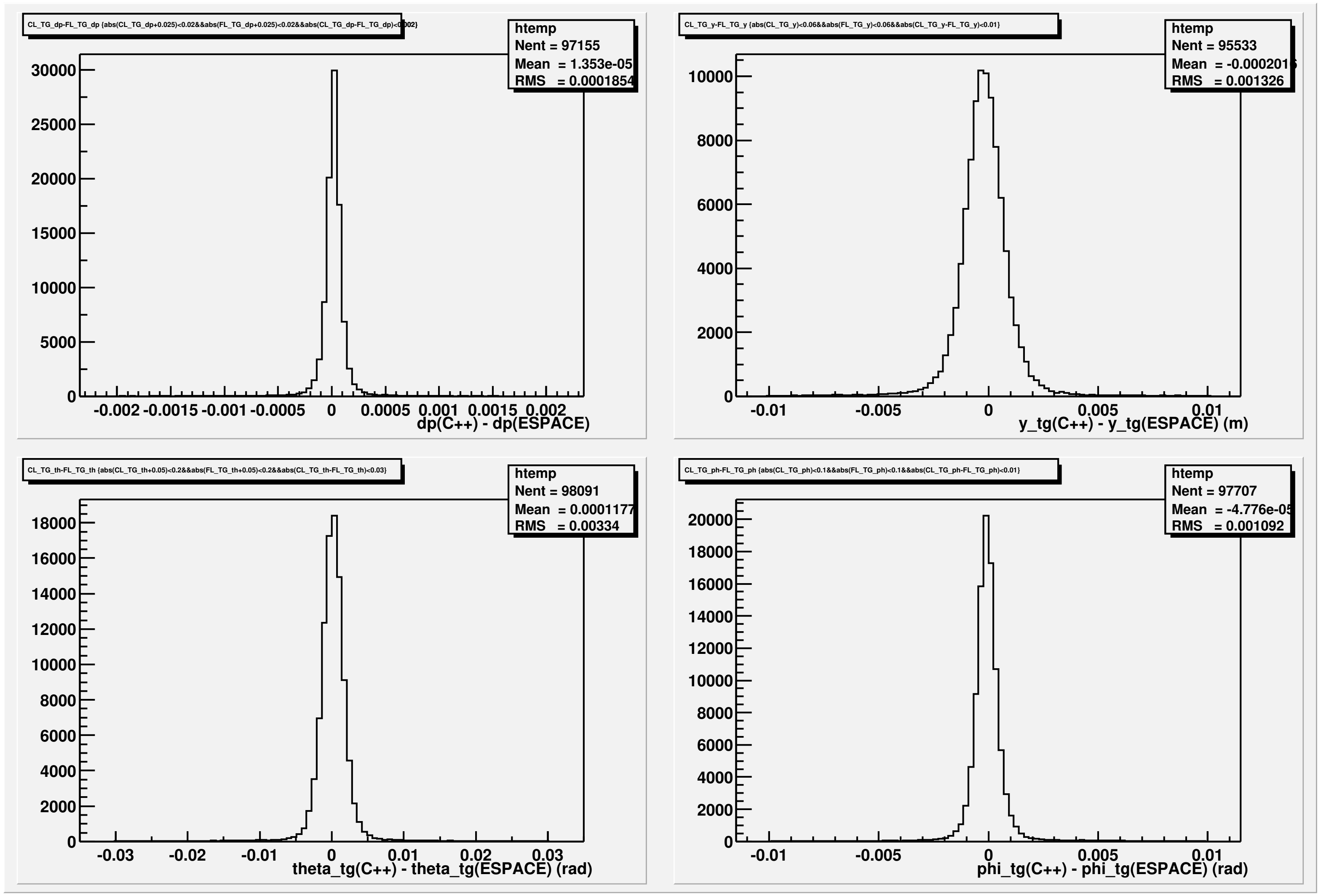
- Transform (x, y, θ, ϕ) to “rotating TRANSPORT frame”
- Multiply resulting vector with inverse TRANSPORT matrix (up to 5-th order).
- Details: Nilanga’s optimization writeup.

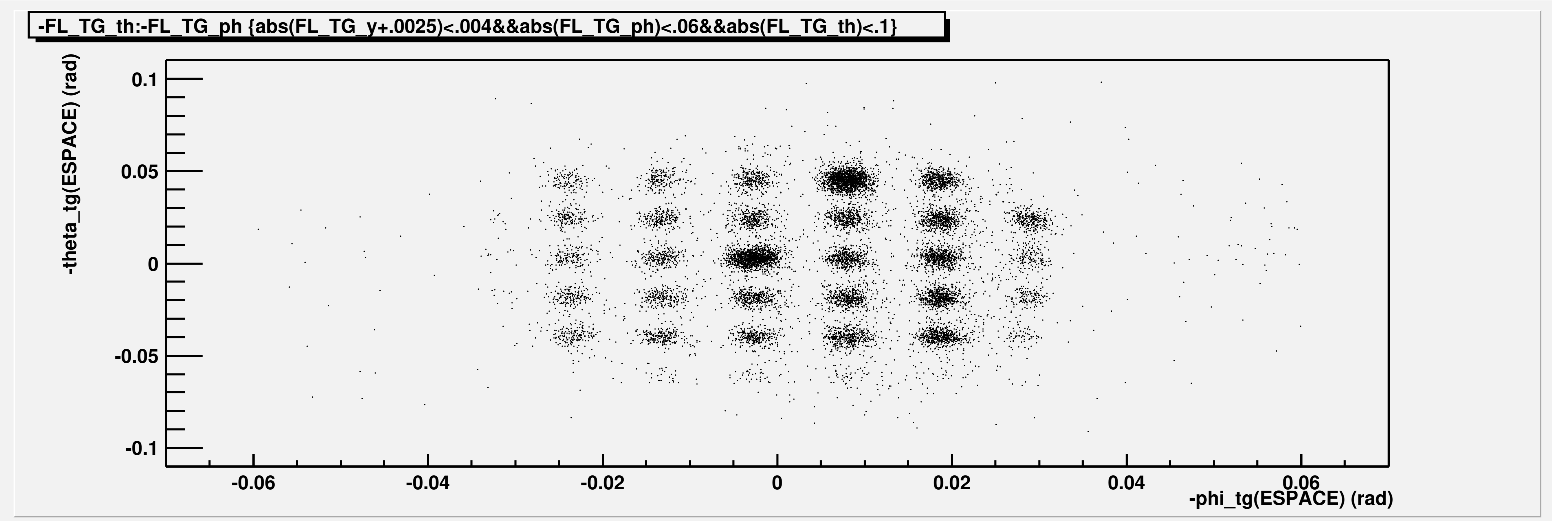
ESPACE also has provisions for x_0 and, related to it, extended target corrections, if requested.
Not yet implemented in C++ Analyzer.

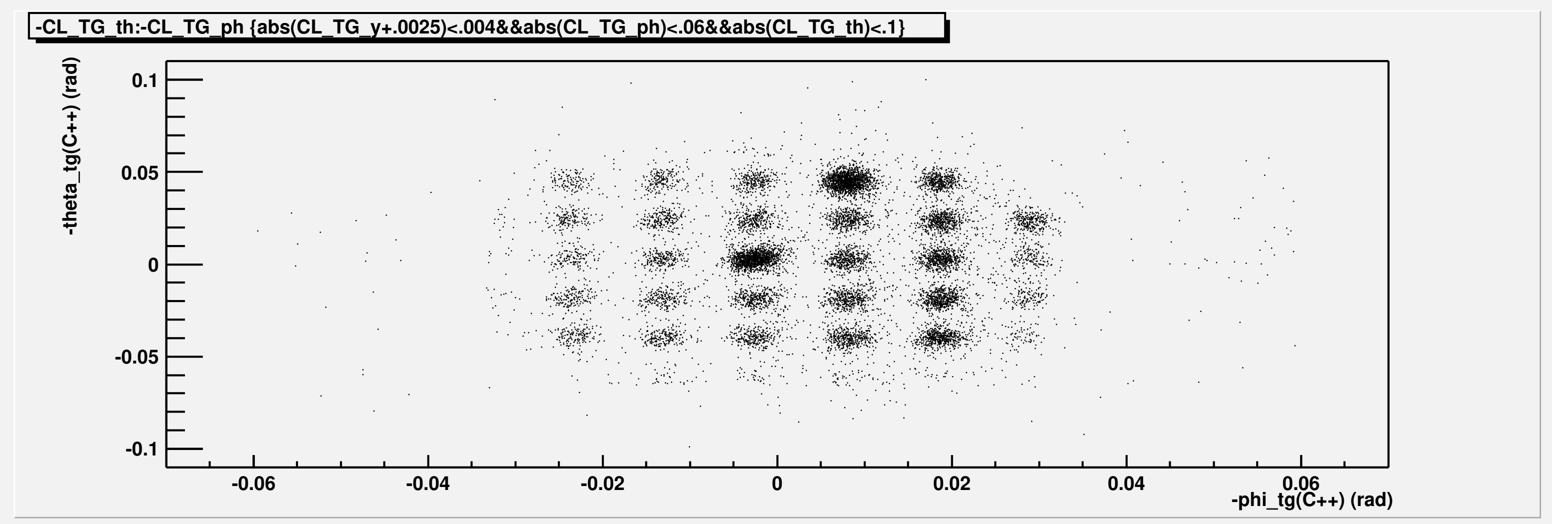
Comparison of Results

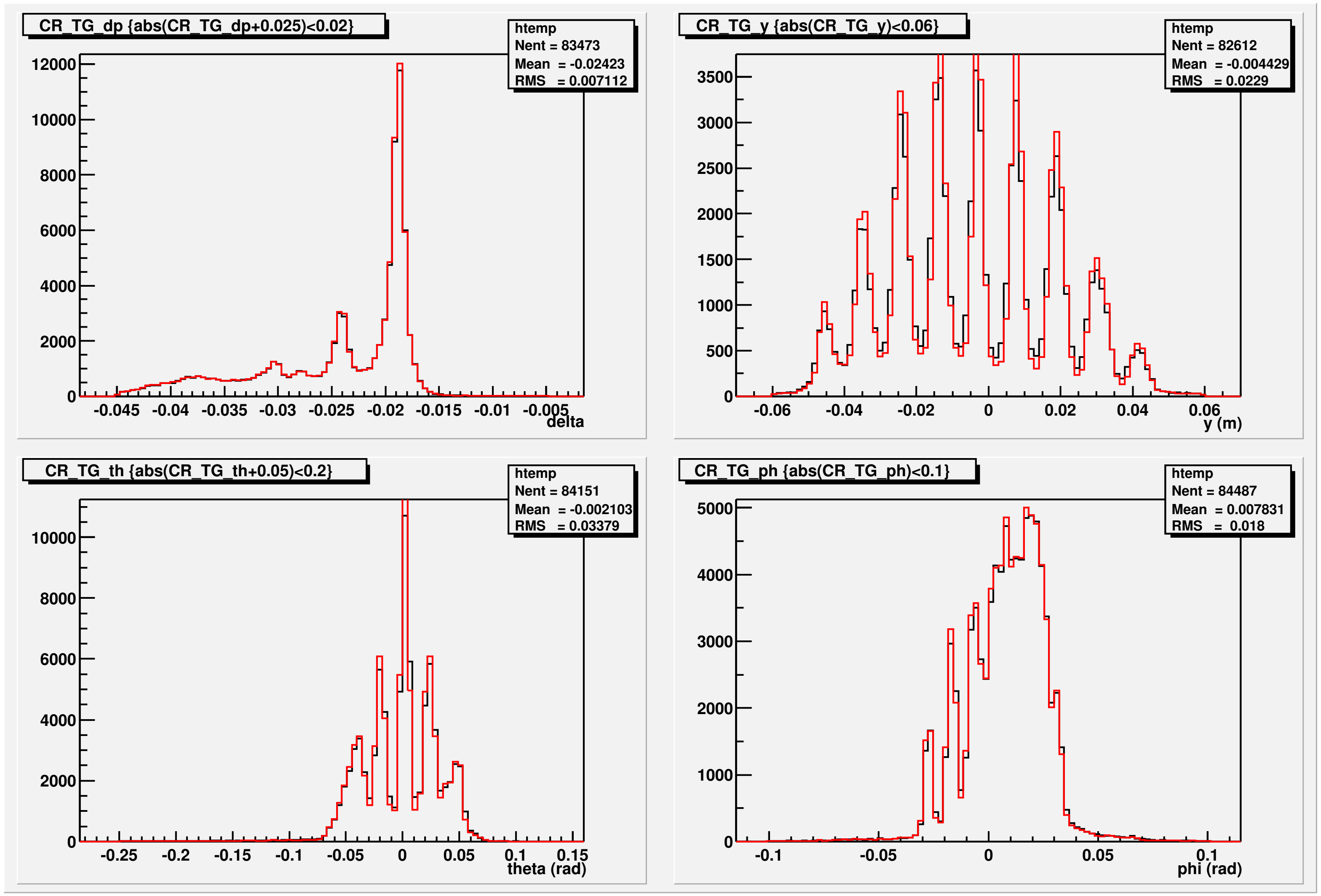
- ^{12}C(e,e) optics data, run 1146, 30 Sep 2000
- $E = 840$ MeV, $\theta = 16^\circ$
- 9-foil carbon target, sieve slit for both HRS
- Identical databases
- Extended target correction disabled in ESPACE

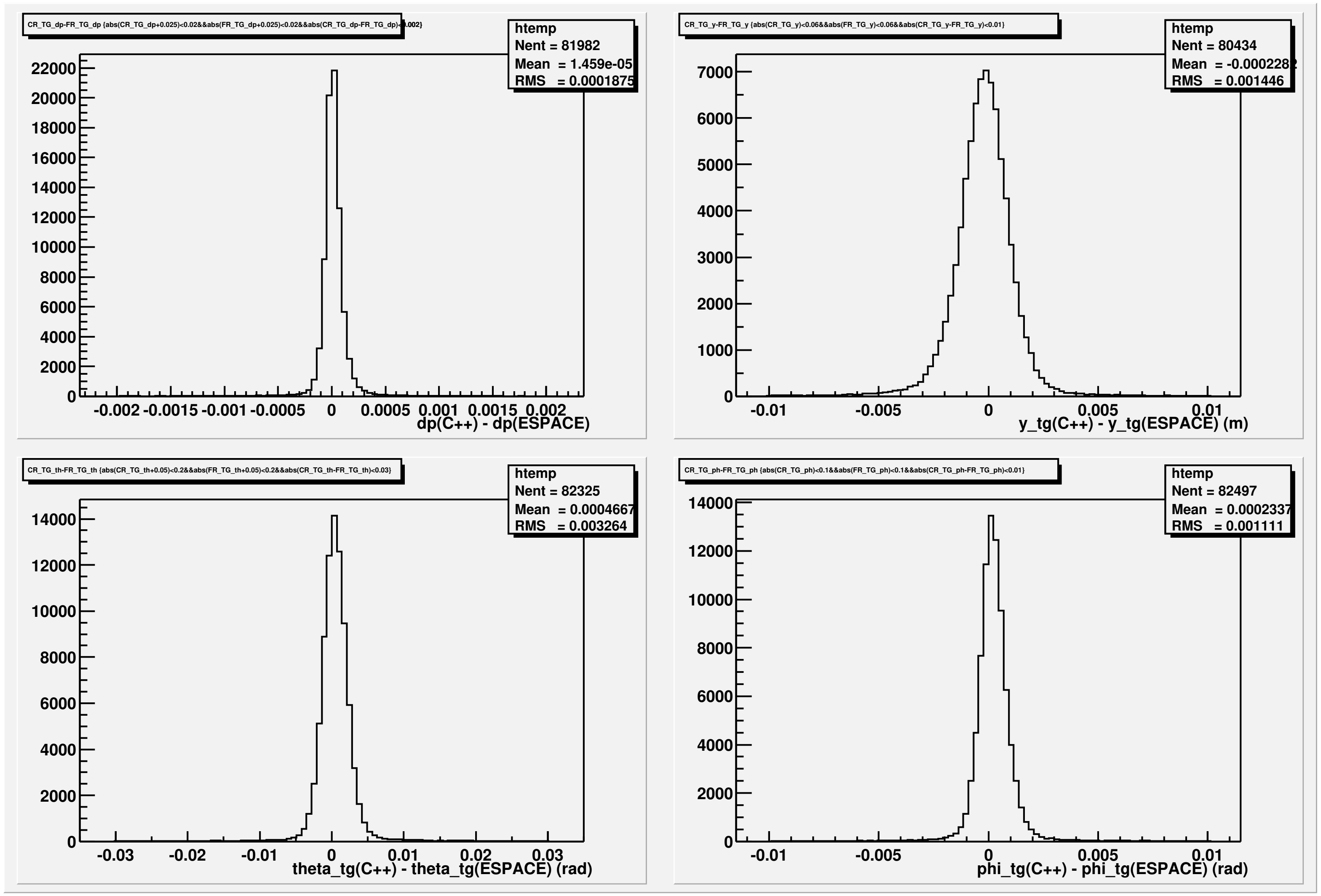

CL_TG_dp {abs(CL_TG_dp+0.025)<0.02}
htemp
Nent = 98877
Mean = -0.02357
RMS = 0.007194
delta
CL_TG_y {abs(CL_TG_y)<0.06}
htemp
Nent = 98099
Mean = 0.005403
RMS = 0.02248
y (m)
CL_TG_th {abs(CL_TG_th+0.05)<0.2}
htemp
Nent = 100933
Mean = -0.005703
RMS = 0.0389
theta (rad)
CL_TG_ph {abs(CL_TG_ph)<0.1}
htemp
Nent = 100240
Mean = -0.009016
RMS = 0.01841
phi (rad)


CL_TG_dp-FL_TG_dp {abs(CL_TG_dp+0.025)<0.02&&abs(FL_TG_dp+0.025)<0.02&&abs(CL_TG_dp-FL_TG_dp)<0.002}
htemp
Nent = 97155
Mean = 1.353e-05
RMS = 0.0001854
30000
25000
20000
15000
10000
5000
0
-0.002 -0.0015 -0.001 -0.0005 0 0.0005 0.001 0.0015 0.002
dp(C++) - dp(ESPACE)
CL_TG_y-FL_TG_y {abs(CL_TG_y)<0.06&&abs(FL_TG_y)<0.06&&abs(CL_TG_y-FL_TG_y)<0.01}
htemp
Nent = 95533
Mean = -0.000201
RMS = 0.001326
10000
8000
6000
4000
2000
0
-0.01 -0.005 0 0.005 0.01
y_tg(C++) - y_tg(ESPACE) (m)
CL_TG_th-FL_TG_th {abs(CL_TG_th+0.05)<0.2&&abs(FL_TG_th+0.05)<0.2&&abs(CL_TG_th-FL_TG_th)<0.03}
htemp
Nent = 98091
Mean = 0.0001177
RMS = 0.00334
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
-0.03 -0.02 -0.01 0 0.01 0.02 0.03
theta_tg(C++) - theta_tg(ESPACE) (rad)
CL_TG_ph-FL_TG_ph {abs(CL_TG_ph)<0.1&&abs(FL_TG_ph)<0.1&&abs(CL_TG_ph-FL_TG_ph)<0.01}
htemp
Nent = 97707
Mean = -4.776e-05
RMS = 0.001092
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
-0.01 -0.005 0 0.005 0.01
phi_tg(C++) - phi_tg(ESPACE) (rad)

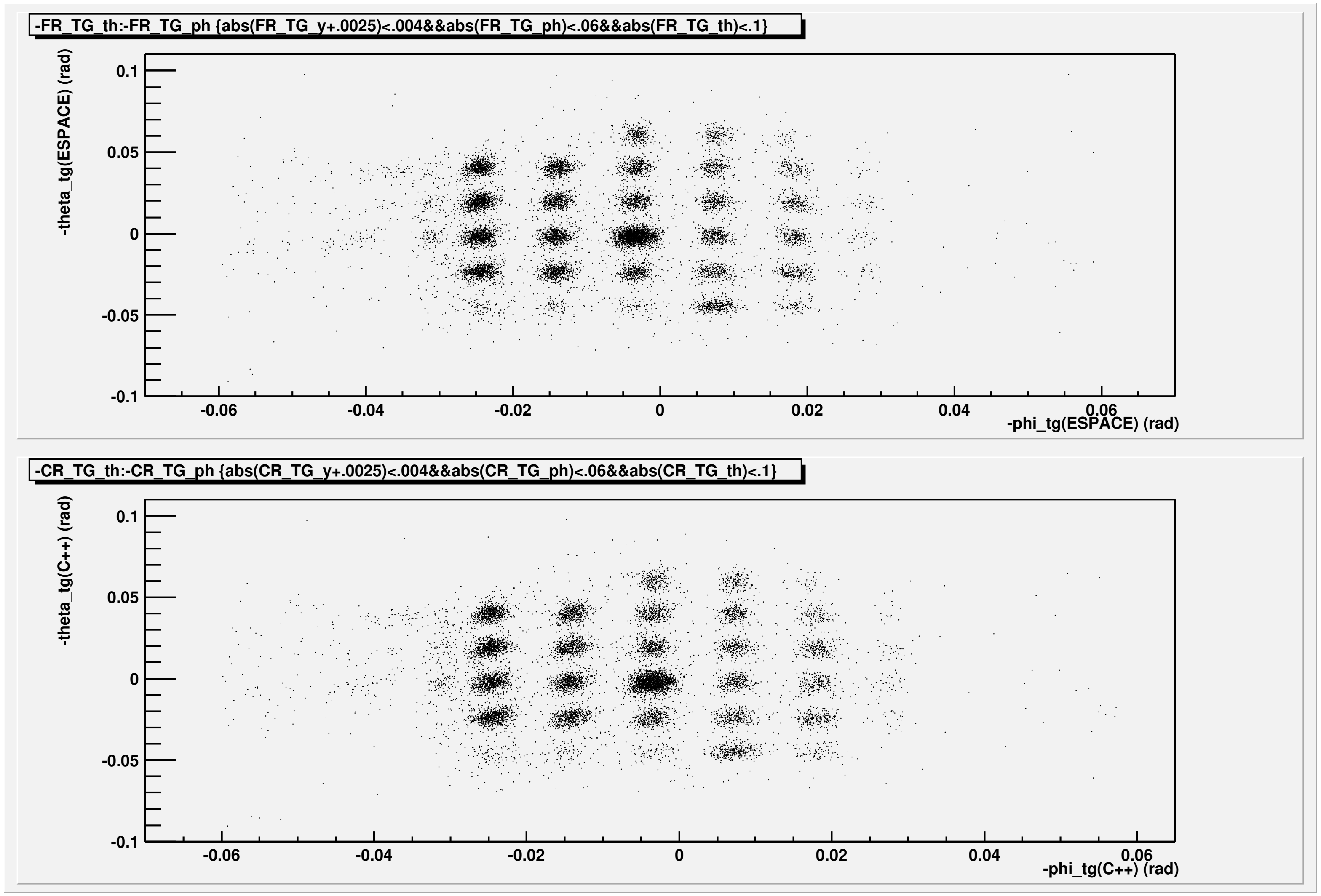

-FL_TG_th:-FL_TG_ph {abs(FL_TG_y+.0025)<.004&&abs(FL_TG_ph)<.06&&abs(FL_TG_th)<.1}
-theta_tg(ESPACE) (rad)
0.1
0.05
0
-0.05
-0.1
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
-phi_tg(ESPACE) (rad)


-CL_TG_th:-CL_TG_ph {abs(CL_TG_y+.0025)<.004&&abs(CL_TG_ph)<.06&&abs(CL_TG_th)<.1}
-theta_tg(C++) (rad)
0.1
0.05
0
-0.05
-0.1
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
-phi_tg(C++) (rad)


CR_TG_dp {abs(CR_TG_dp+0.025)<0.02}
htemp
Nent = 83473
Mean = -0.02423
RMS = 0.007112
delta
CR_TG_y {abs(CR_TG_y)<0.06}
htemp
Nent = 82612
Mean = -0.004429
RMS = 0.0229
y (m)
CR_TG_th {abs(CR_TG_th+0.05)<0.2}
htemp
Nent = 84151
Mean = -0.002103
RMS = 0.03379
theta (rad)
CR_TG_ph {abs(CR_TG_ph)<0.1}
htemp
Nent = 84487
Mean = 0.007831
RMS = 0.018
phi (rad)

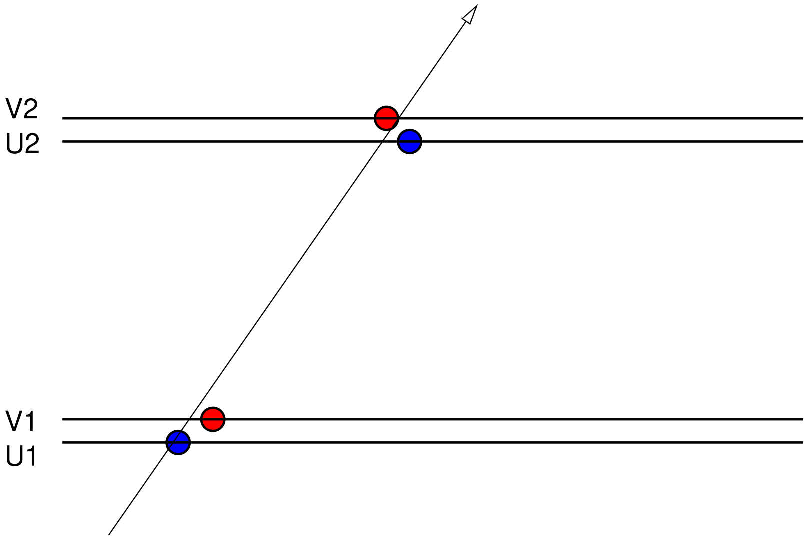


CR_TG_dp-FR_TG_dp (abs(CR_TG_dp+0.025)<0.02&&abs(FR_TG_dp+0.025)<0.02&&abs(CR_TG_dp-FR_TG_dp)<0.002)
htemp
Nent = 81982
Mean = 1.459e-05
RMS = 0.0001875
dp(C++) - dp(ESPACE)
CR_TG_y-FR_TG_y (abs(CR_TG_y)<0.06&&abs(FR_TG_y)<0.06&&abs(CR_TG_y-FR_TG_y)<0.01)
htemp
Nent = 80434
Mean = -0.0002282
RMS = 0.001446
y_tg(C++) - y_tg(ESPACE) (m)
CR_TG_th-FR_TG_th (abs(CR_TG_th+0.05)<0.2&&abs(FR_TG_th+0.05)<0.2&&abs(CR_TG_th-FR_TG_th)<0.03)
htemp
Nent = 82325
Mean = 0.0004667
RMS = 0.003264
theta_tg(C++) - theta_tg(ESPACE) (rad)
CR_TG_ph-FR_TG_ph (abs(CR_TG_ph)<0.1&&abs(FR_TG_ph)<0.1&&abs(CR_TG_ph-FR_TG_ph)<0.01)
htemp
Nent = 82497
Mean = 0.0002337
RMS = 0.001111
phi_tg(C++) - phi_tg(ESPACE) (rad)

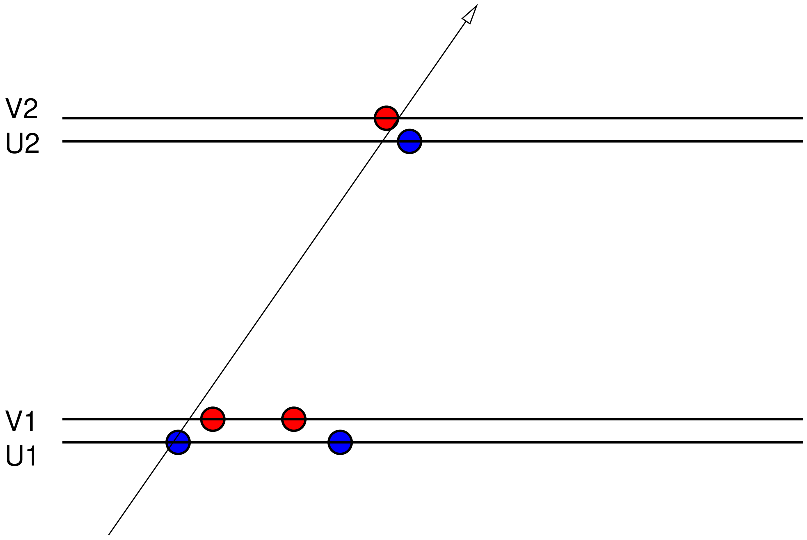

-FR_TG_th:-FR_TG_ph {abs(FR_TG_y+.0025)<.004&&abs(FR_TG_ph)<.06&&abs(FR_TG_th)<.1}
-theta_tg(ESPACE) (rad)
0.1
0.05
0
-0.05
-0.1
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
-phi_tg(ESPACE) (rad)
-CR_TG_th:-CR_TG_ph {abs(CR_TG_y+.0025)<.004&&abs(CR_TG_ph)<.06&&abs(CR_TG_th)<.1}
-theta_tg(C++) (rad)
-phi_tg(C++) (rad)

Single Cluster vs. Multi-Cluster Analysis

Single clusters



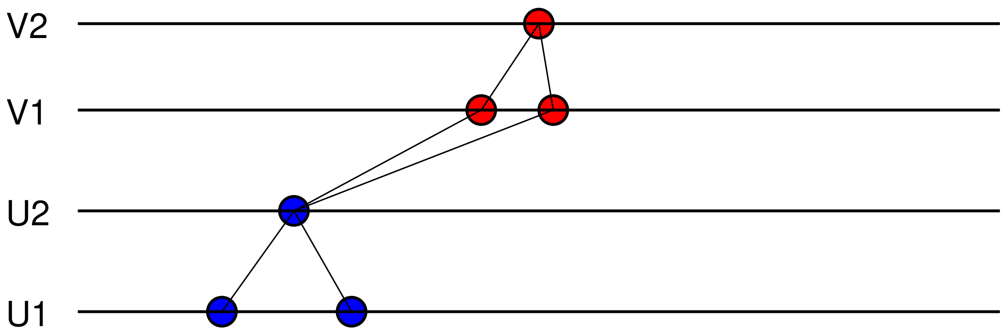
Multiple clusters



Issues with Multiple Clusters

- Must allow for timing offset ("t_0") in cluster fitting
  $\longrightarrow$ 3-parameter linear fit ("`curfit`")
- Cluster matching
- Overlapping clusters $\longrightarrow$ ignored in ESPACE

Cluster Matching & Path Analysis in ESPACE



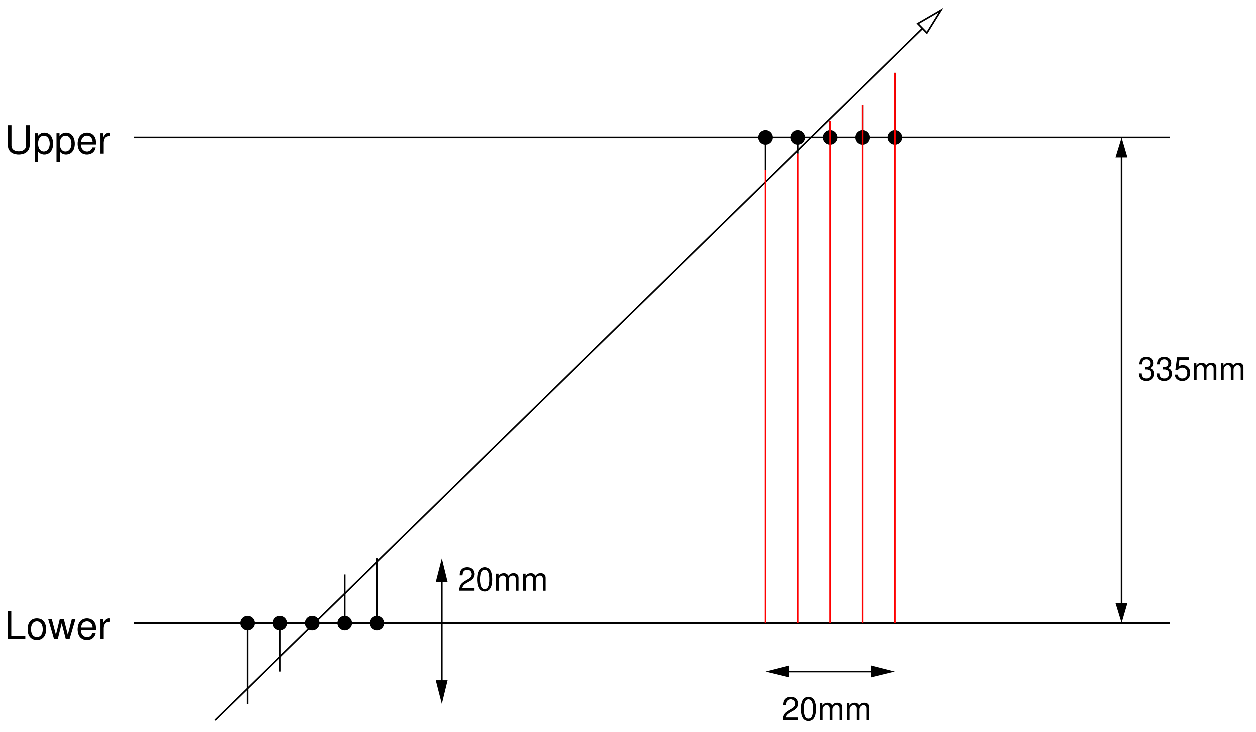
Goodness of fit criterion:

$$\chi^2_{tot} = \chi^2_{u1u2} + \chi^2_{v1v2} + \left(\frac{t_{0,v1v2} - t_{0,u1u2}}{\min(\sigma(t_{0,v1v2}), \sigma(t_{0,u1u2}))}\right)^2$$

Golden Track selection:

$$t_{0,av} = \frac{\frac{t_{0,u1u2}}{\sigma(t_{0,u1u2})^2} + \frac{t_{0,v1v2}}{\sigma(t_{0,v1v2})^2}}{\frac{1}{\sigma(t_{0,u1u2})^2} + \frac{1}{\sigma(t_{0,u1u2})^2}}$$

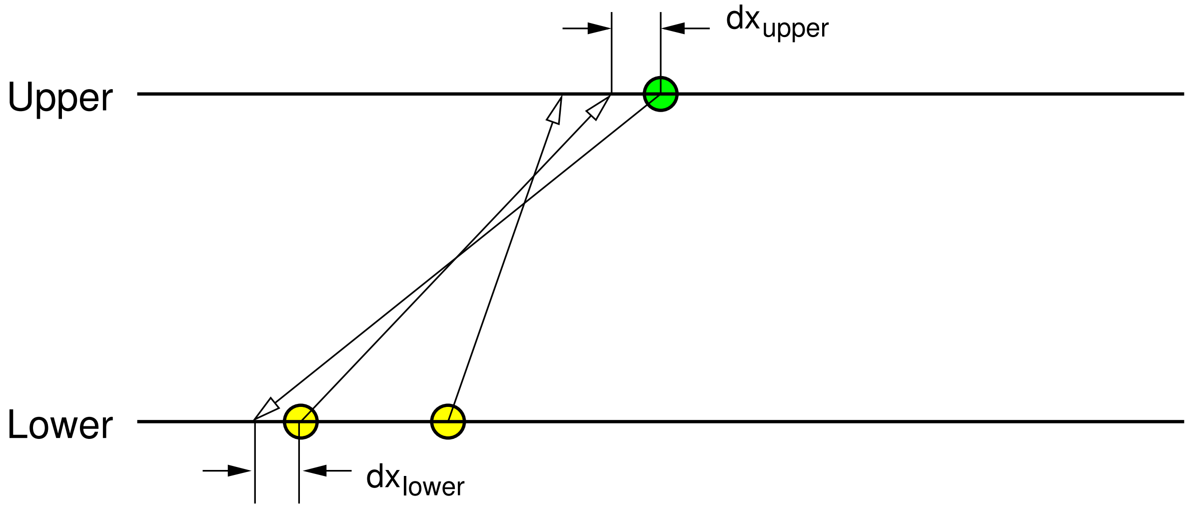
Two-plane "global" fit of ESPACE



NB: Not to scale

Multiple Clusters in the C++ Analyzer

- Match U and V clusters *within the same plane pair* based on their absolute timing (ad hoc solution).
  $\longrightarrow$ "UV tracks"
- Match UV tracks. Minimize distance of projected cross-over points:



Goodness of match criterion:

$$E^2 = dx^2_{upper} + dy^2_{upper} + dx^2_{lower} + dy^2_{lower}$$

To Do

Tracking

- Optimize single-track performance:
  - Global angles for projections
  - ToF and timewalk timing corrections
  - Extended target correction (using BPM info)
  - TDC offset and reconstruction optimization
  - Time-to-distance conversion
- Multi-cluster analysis:
  - Algorithm — neither seems optimal
  - t_0 fit in C++
  - Overlapping clusters, multiple hits per wire
- Tests
- Performance

To Do (cont.)

Analyzer

- Improve histogramming & output:
  - Output expressions of variables
  - Cuts on histograms
  - Formulas of multi-valued quantities
  - Global variable object nesting depth
- Implement cuts in analysis flow
- Databases!
- Support FPP

Resources

Project homepage:

`http://hallaweb.jlab.org/root/`

Mailing List:

`halla_root@jlab.org`

Interested people encouraged to subscribe.
Archive accessible from Hall A homepage.

The ROOT Web Page:

`http://root.cern.ch/`