A new software for physics-agnostic reconstruction in the T2K near-detector TPCs

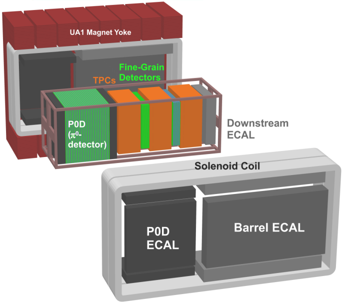
L. Koch

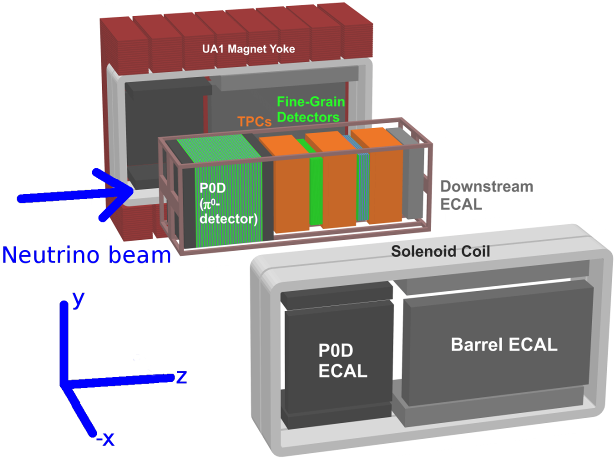
III. Physikalisches Institut B
RWTH Aachen University

Workshop on Software for Time Projection Chambers for Nuclear Physics Experiments, FRIB, 2016-08-09

T2K and ND280

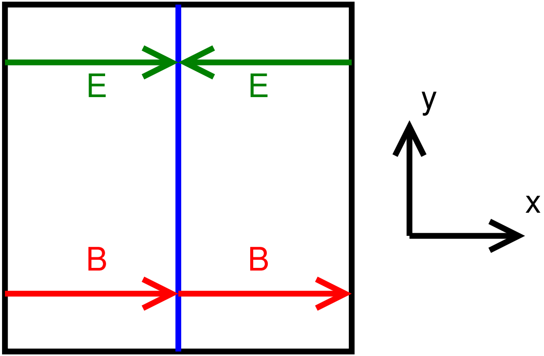
- Tokai To Kamioka
  - Long baseline, neutrino-beam experiment in Japan
- Near Detector 280
  - Multi purpose, magnetised detector
  - 280 m downstream the graphite target
  - Scintillators and 3 large TPCs
  - Un-oscillated beam characterisation
  - Cross-section measurements





- 3 large TPCs $\sim 3\,\mathrm{m}^3$ each
- Gas mixture, "T2K-gas", by volume
  - $95\,\%$ Argon, Ar
  - $3\,\%$ Tetra-fluoro-methane, CF_4
  - $2\,\%$ Isobutane, iC_4H_{10}

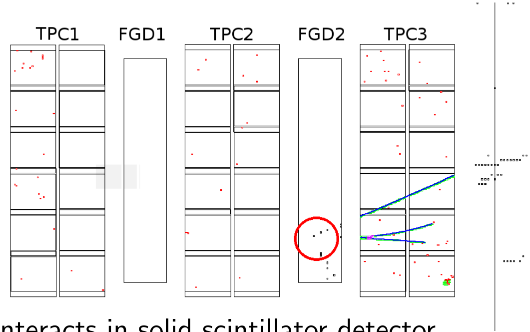
The TPCs



- Central cathode
- Drift along x-axis, $v_d \sim 80\,\mu\mathrm{m/ns}$
- Magnetic field ($\sim 0.2\,\mathrm{T}$) parallel to electric field ($\sim 300\,\mathrm{V/cm}$)
- Pad-based ($\sim 10 \times 7\,\mathrm{mm}^2$) MicroMeGaS readout at anodes

Why TREx (TPC Reconstruction Extension)?

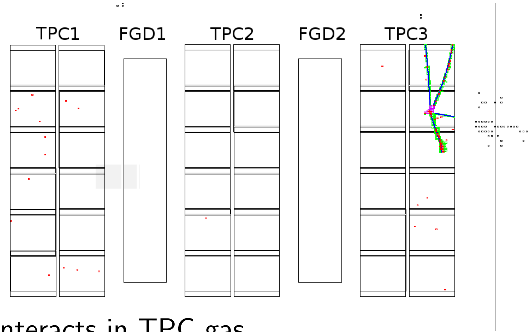
Main measurements



- Neutrino interacts in solid scintillator detector
- Products are identified in the TPCs ($\mathrm{d}E/\mathrm{d}x$ vs. p)
- High density target material
  - ⊞ High statistics
  - ⊟ High energy detection threshold
- TPC reco software optimized for through-going particles

Why TREx (TPC Reconstruction Extension)?

Gas interaction measurements



- Neutrino interacts in TPC gas
- Products are identified in the TPC ($\mathrm{d}E/\mathrm{d}x$ vs. p)
- Low density target material
  - ⊟ Low statistics
  - ⊞ Low energy detection threshold
- Vertexing in TPCs needed new software

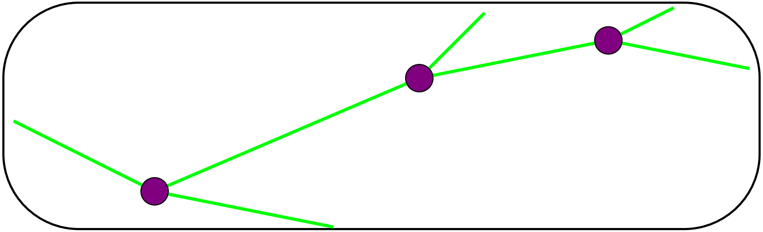
Design goals

- Isotropy
  - Full 3D reconstruction
  - No assumptions about particle directions
- Homogeneity
  - Interactions can happen anywhere in the TPC
  - No assumptions about vertex positions
- Physics-agnosticism
  - Reconstruct objects, but do not try to interpret them

Disclaimer

TREx is quite complex and explaining everything in detail would take multiple talks. I will concentrate on the general principles rather than implementation details.

Output objects

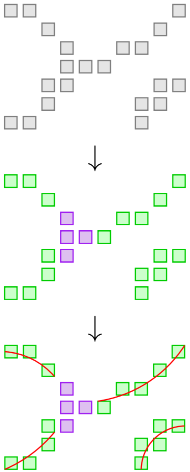


- Patterns
  - Collection of connected paths and junctions
- Paths
  - A series of connected hits that form a particle track
- Junctions
  - Hits where multiple paths meet or branch off
- No vertices!
  - TREx makes no distinction between vertices and secondary interactions
  - Analyser must decide whether junction is a vertex or a delta-ray, etc.

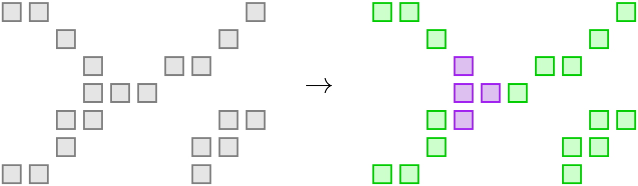
How does it work?

TREx works in two phases

1. Pattern recognition
   - Grouping hits into paths and junctions
   - Based on A*-algorithm
     - Well-known path finding algorithm
2. Track fitting
   - Fit helices to paths
   - Likelihood based
   - Merge broken-up tracks of the same particle

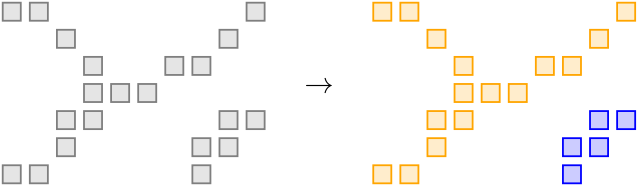


Pattern recognition



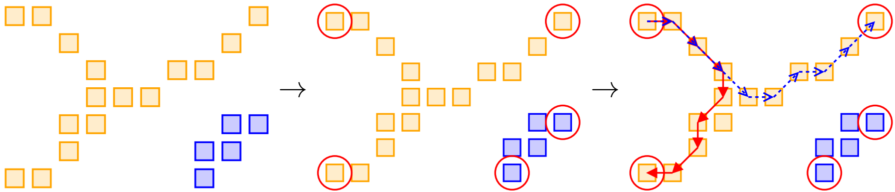
1. Group hits into patterns
2. Look for edges, i.e. track ends
3. Build paths and look for junctions
4. Assign hits to paths/junctions
5. Clustering

Grouping



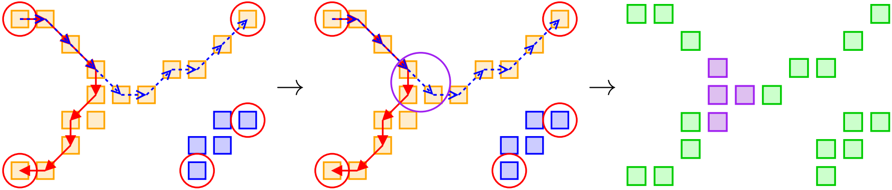
- Neighbouring hits are grouped into patterns
- Equivalent statements:
  - Two hits are in the same pattern
  - There exists a path between the two hits

Edge detection and path finding



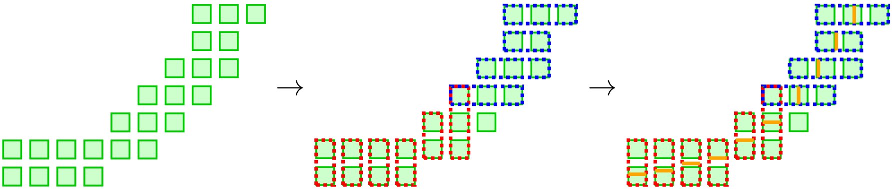
- Patterns are scanned for edges, i.e. track ends
  - Look for maximum coordinates
- Use A* algorithm to find shortest connections between edges
- To find stopping track ends
  - Remove found paths
  - Repeat edge search

Junction detection and hit association



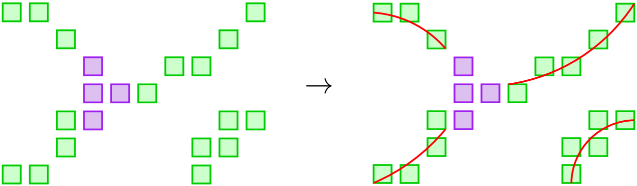
- Add junctions where paths diverge
- Add all unused hits to found paths and junctions

Clustering



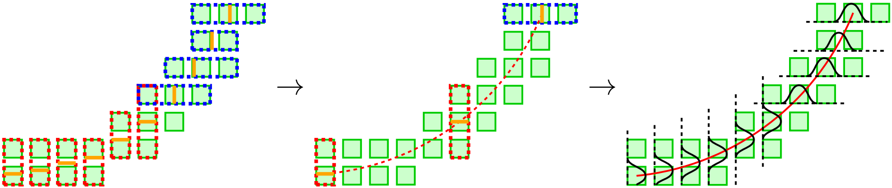
- A cluster is a collection of hits in horizontal or vertical direction
  - Has nothing to do with ionization clusters
- Horizontal or vertical clustering depends on local angle
- Used to calculate precise y or z positions

Track fitting



1. Seeding
2. Likelihood fit
3. Track matching and merging

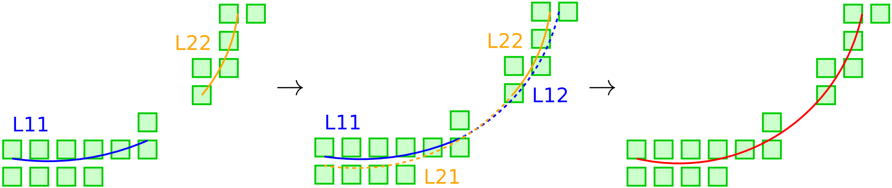
Seeding and likelihood fit



- Seed parameters for fit (i.e. the first guess) taken from start, end and mid-point of paths
- Likelihood calculated for each cluster separately
- Propagate helix to cluster plane (xy or xz)
- Get expected charge distributions from track position and angle
- Calculate likelihoods from expectation for all hits in the cluster
- Maximize total likelihood of all clusters for best fit track

D. Karlen, P. Poffenberger, and G. Rosenbaum.
Nuclear Instruments and Methods in Physics Research, A555:80-92, 2005.

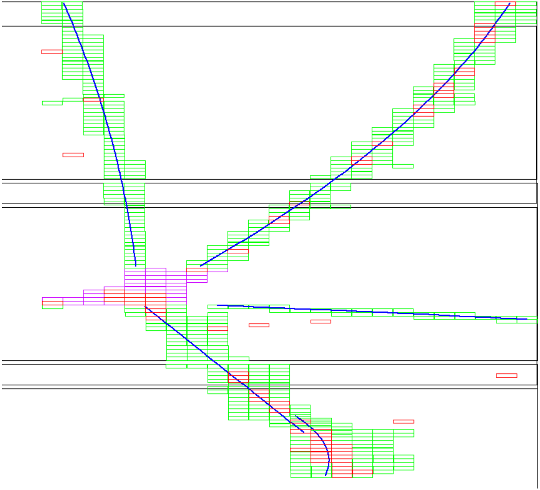
Likelihood match and merging



- Sometimes tracks “break”: one particle is split into multiple paths
  - Due to missing hits or delta-ray junctions
- Each path has its own fitted helix with its maximum likelihood
  - $L11$ and $L22$
- We can propagate those to the other paths and calculate their likelihoods
  - Helix 1 propagated to path 2: $L12$
  - Helix 2 propagated to path 1: $L21$
- $(L11 \cdot L12) \ll (L11 \cdot L22) \gg (L21 \cdot L22)$

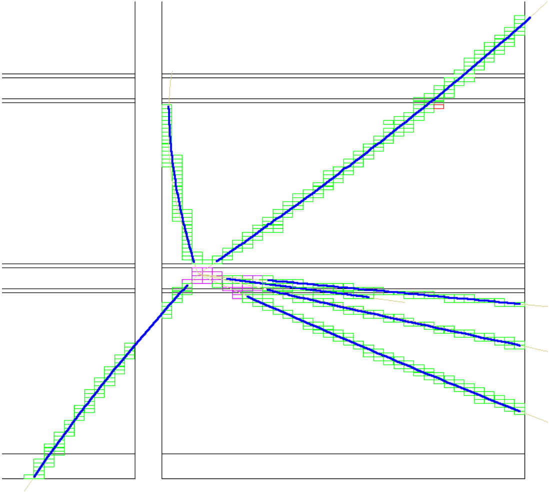
  $\Rightarrow$ Likely two separate particles
- Otherwise merge and refit or save information for analyser to decide

Real data 4-track gas-interaction-like event



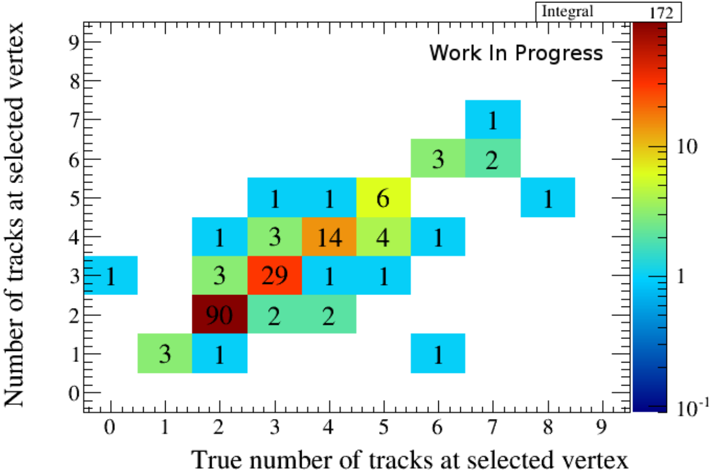
- All visible tracks are reconstructed, except for (possible) stub on the left

7-track MC gas interaction event



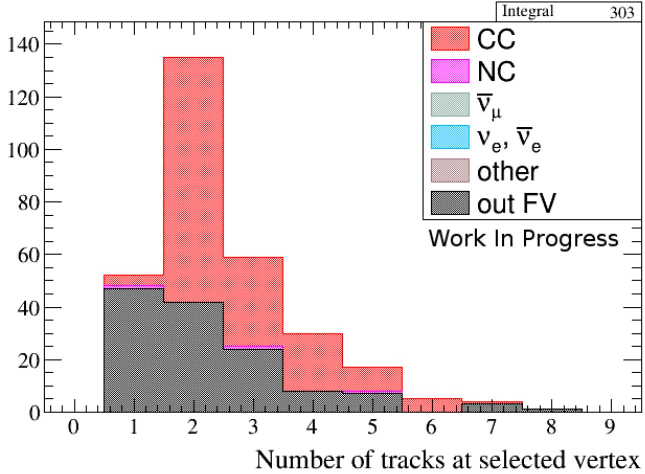
- Difficult to reconstruct close to vertex, but actually just one junction!

Multiplicity migration matrix



- Reco: paths connected to vertex junction
- Truth: charged particles coming from a gas interaction vertex

CC_{inc} gas interaction selection performance



- Purity: $\sim 60\,\%$
- Efficiency: $\sim 45\,\%$

Conclusion

- TREx is a versatile tool for TPC reconstruction
- Already performing very well both for through-going particles and gas interactions
- Improvements for handling some fringe cases still possible (and planned)
  - Rare cases, but relevant for high-BG gas interaction analysis
- First neutrino gas interaction analysis paper is coming up soon
- Stay tuned!

Thank you!

Backup

A*-algorithm

- Find shortest connection between two nodes of a graph
- Cost for connection = actual cost (i.e. length) of connection + heuristic cost of chosen node
- Heuristic cost = distance of chosen node from destination
- Essentially: Evaluate connections that get you closer to the destination node first
- Depending on heuristic cost function, guaranteed to find shortest connection