A LOCAL TRACKING ALGORITHM FOR THE CENTRAL DRIFT CHAMBER OF BELLE II.

F2F Meeting - Prag

Oliver Frost
Deutsches Elektronen-Synchrotron
13th December 2013

DESY
HELMHOLTZ ASSOCIATION
Belle II

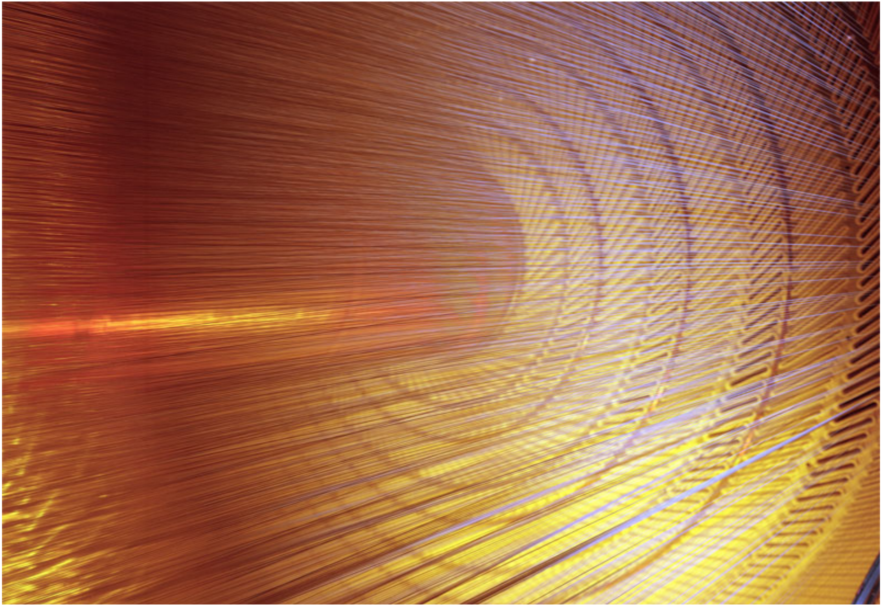
Outline

- Detector
- Tracking finding - problem statement
- Local approach
- Generic algorithms
- Concrete Realization
- Fast fitting
- Further work

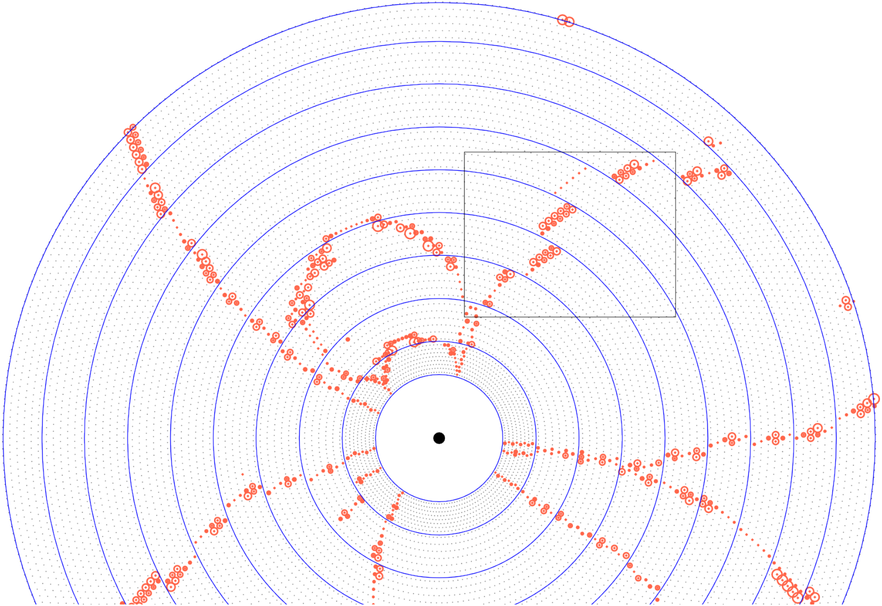
Outline

- Detector
- Tracking finding - problem statement
- Local approach
- Generic algorithms
- Concrete Realization
- Fast fitting
- Further work

Central drift chamber



Typical event xy projection



Typical event xy projection - close up

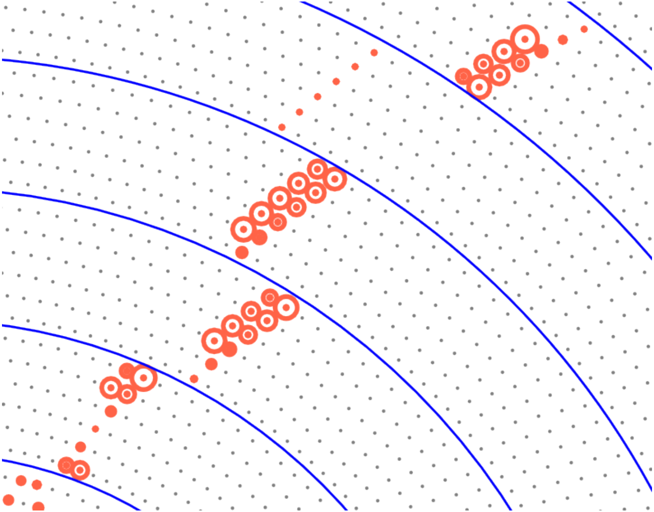


Figure: Typical $B\bar{B}$ event - every circle marks a hit

Geometry of stereo layers

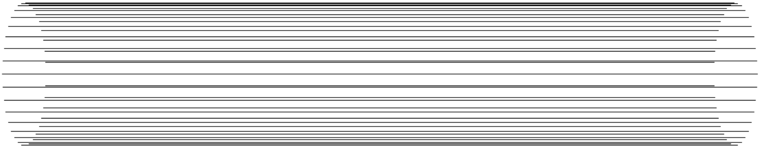


Figure: Axial layer

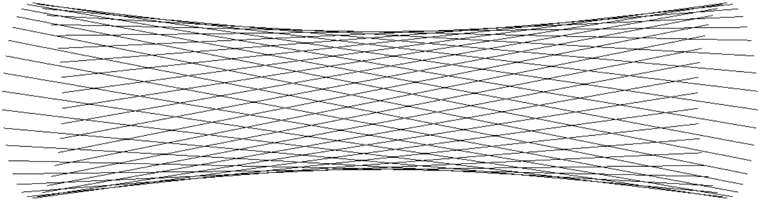


Figure: Stereo layer for z position resolution

Central drift chamber - CDC

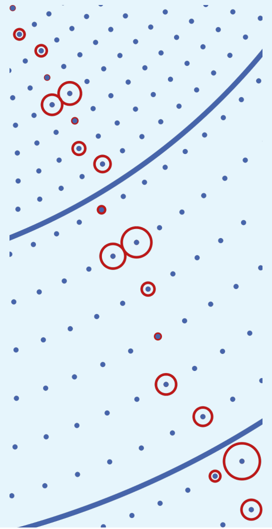
Structure

- 14336 sensitiv wires
- layerwise, hexagonal neighborhoods
- 56 layers in 9 superlayers
- superlayers alternating axial - stereo - axial - ...

Input of tracking / variables of the hits

- Projected xy wire positions
- Skewness of wires (axial - stereo)
- Drift time / drift circle radii according to known drift velocity function
- (Energy deposition)

Close up



Outline

> Detector

> Tracking finding - problem statement

> Local approach

> Generic algorithms

> Concrete Realization

> Fast fitting

> Further work

Finding tracks - problem statement

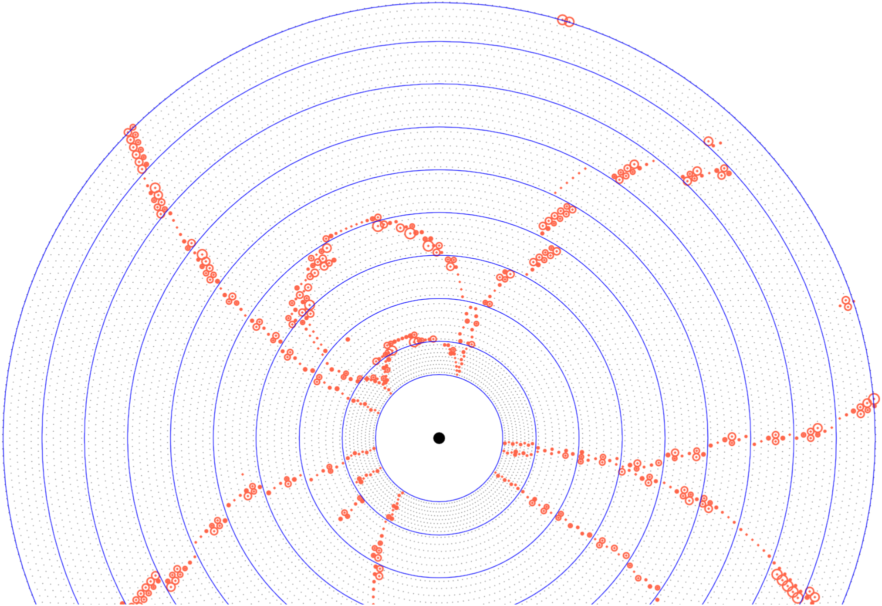
Goals

- Group measurements / hits by the particle that caused them.
- Sort hits in the order of occurance.
- Provide initial parameters for track fit.

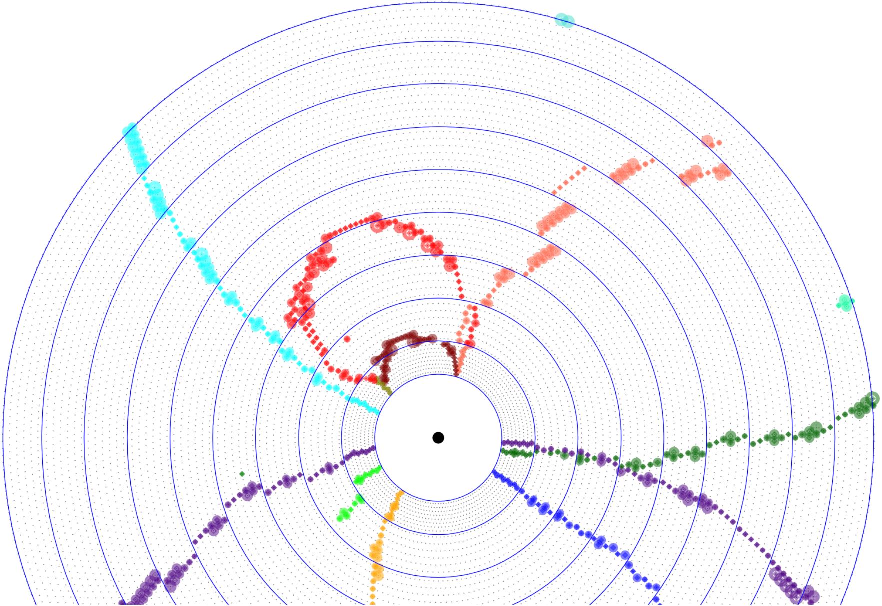
Non-goal

- Accurate track fitting $\rightarrow$ Kalmanfilter of Genfit

Turn this . . .



. . . into that



Conditions and requirements

Conditions

- Tracks are **helices** locally
- Distorted by **multiple scattering**
- Particles might **curl** inside the CDC
- Mind **delay times** (TOF and in wire propagation time)
- Much increased **beam background** over Belle

Requirements

- Introduce as **little direction orientation** as possible
- Provide **intial values** for track fit
- Maximize **efficiency** (find all tracks)
- Maximize **purity** (introduce only few fake tracks)
- Be **fast**

General approaches

Global — top – down

- Recognize hits supporting a trajectory with few parameters
- Assign the hits to that global form

Local — bottom – up

- Combine neighboring hits
- Continuously increase group size
- Judge continuations by quick extrapolations

General approaches

Global — top – down

- Recognize hits supporting a trajectory with few parameters
- Assign the hits to that global form

Local — bottom – up

- Combine neighboring hits
- Continuously increase group size
- Judge continuations by quick extrapolations

Advantage

Fast and simple to implement

Advantage

Detailed modeling of tracks

Disadvantage

Neglects scattering and secondary decays

Disadvantage

Many tunable parameters to be optimized

General approaches

Global — top – down

- Recognize hits supporting a trajectory with few parameters
- Assign the hits to that global form

Advantage

Fast and simple to implement

Disadvantage

Neglects scattering and secondary decays

Local — bottom – up

- Combine neighboring hits
- Continuously increase group size
- Judge continuations by quick extrapolations

Advantage

Detailed modeling of tracks

Disadvantage

Many tunable parameters to be optimized

General approaches

Global — top – down

- Recognize hits supporting a trajectory with few parameters
- Assign the hits to that global form

Local — bottom – up

- Combine neighboring hits
- Continuously increase group size
- Judge continuations by quick extrapolations

Methods

Hough / Legendre transformation

Methods

Networks / Cellular automata + fast fits

General approaches

Global — top – down

- Recognize hits supporting a trajectory with few parameters
- Assign the hits to that global form

Local — bottom – up

- Combine neighboring hits
- Continuously increase group size
- Judge continuations by quick extrapolations

Methods

Networks / Cellular automata + fast fits

Outline

- Detector
- Tracking finding - problem statement
- Local approach
- Generic algorithms
- Concrete Realization
- Fast fitting
- Further work

Local approach - generic building blocks

Positions of particle

Combine closeby hits to form a possible position of the particle

Transitions

Find neighboring positions such that particle could have transitioned from one to the another

Positions and transitions

- Encode possible movements of particles
- Should closely resemble the physical movement
- $\rightarrow$ **Paths** in the graph **represent tracks**

Local approach - generic building blocks

Positions of particle

Combine closeby hits to form a possible position of the particle

Transitions

Find neighboring positions such that particle could have transitioned from one to the another

Positions and transitions

- Encode possible movements of particles
- Should closely resemble the physical movement
- $\rightarrow$ **Paths** in the graph **represent tracks**

Local approach - generic building blocks

Positions of particle

Combine closeby hits to form a possible position of the particle

Transitions

Find neighboring positions such that particle could have transitioned from one to the another

Positions and transitions

- Encode possible movements of particles
- Should closely resemble the physical movement
- $\rightarrow$ **Paths** in the graph **represent tracks**

Local approach - generic building blocks

Positions of particle - **Vertices / Cells**

Combine closeby hits to form a possible position of the particle

Transitions - **Edges / Neighbors**

Find neighboring positions such that particle could have transitioned from one to the another

Positions and transitions - **Graph**

- Encode possible movements of particles
- Should closely resemble the physical movement
- $\rightarrow$ **Paths** in the graph **represent tracks**

Design decisions

Graph properties

- Directional or adirectional?
- Loop free?
- Symmetric or asymmetric?
- Weighted vertices and/or edges?

Graph vertices

- How do vertices relate to hits?

Graph edges

- How can we find neighboring vertices?
- Can we exploit **geometrical constraints**?
- Which **extrapolation** methods can refine our judgement?

Path / track extraction

Which **algorithms** generate tracks from the graph best?

Design decisions

Graph properties

- Directional or adirectional?
- Loop free?
- Symmetric or asymmetric?
- Weighted vertices and/or edges?

Graph vertices

- How do vertices relate to hits?

Graph edges

- How can we find neighboring vertices?
- Can we exploit **geometrical constraints**?
- Which **extrapolation** methods can refine our judgement?

Path / track extraction

Which **algorithms** generate tracks from the graph best?

Outline

- Detector
- Tracking finding - problem statement
- Local approach
- Generic algorithms
- Concrete Realization
- Fast fitting
- Further work

Algorithms to extract paths from graphs

Brute force

> Complete backtracking

Networks and simplifications

> Hopfield-Network
> Denby-Peterson-Network
> ?
> Cellular automaton

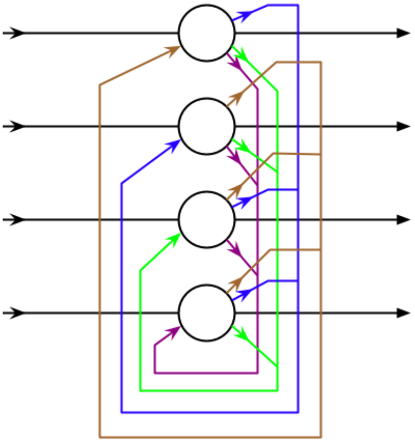


Figure: Scheme of Hopfield-Network

Algorithms to extract paths from graphs

Brute force

- Complete backtracking

Networks and simplifications

- Hopfield-Network
- Denby-Peterson-Network
- **?**
- Cellular automaton

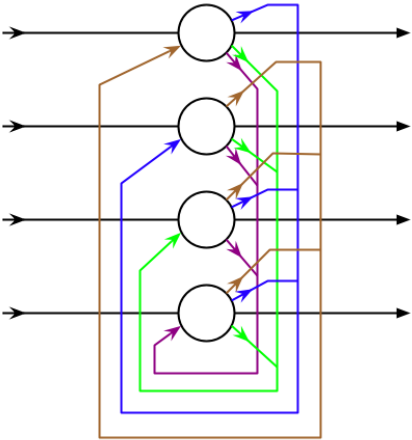


Figure: Scheme of Hopfield-Network

Hopfield network

Characteristics

- Mimics a network of neurons
- Model for assoziative memory

Other applications

- Traveling sales man problem
- Ising spin model (actually equivalent)
- PXD/SVD tracking (Jakob)

Benefits

- No backtracking
- Desired global patter emerges from the local connections by itself

Hopfield-Networks

Input graph

- Cells in inactive state ($s_i = 0$)
- Weighted adirectional edges w_{ij} among cells
- $w_{ij} > 0$ for support / $w_{ij} < 0$ for mutual exclusion of cells
- Weight θ_i for each cell encoding an external excitation

Process

1. Update s_i according the sign of a weighted sum of excitations
2. Repeat until activity states s_i are stationary.

Variations

- Mean field approximation (sign $\rightarrow$ sigmoid function)
- Simultanious update / asynchronous update

Hopfield-Network - schematic

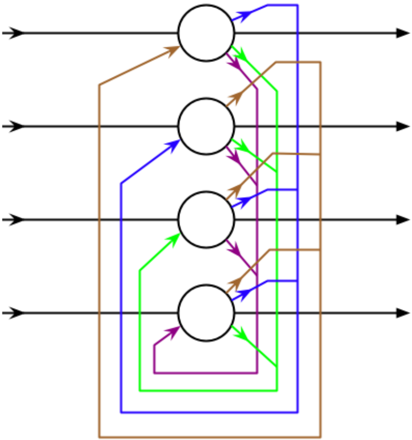


Figure: Scheme of Hopfield network

Update rule

$$s_i = \begin{cases} 1 & \text{if } \sum_j w_{ij} \cdot s_j + \theta_i > 0 \\ 0 & \text{if } \sum_j w_{ij} \cdot s_j + \theta_i < 0 \end{cases}$$

Minimized energy function

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} \cdot s_i \cdot s_j - \sum_i \theta_i \cdot s_i$$

(compare Ising spin model)

Hopfield network - properties

Final output

- Stationary activity state $s_i = 1$ indicates cell belongs to sought track

$$E_{\text{final}} = -\frac{1}{2} \sum_{s_i, s_j = 1} w_{ij} - \sum_{s_i = 1} \theta_i$$

Denby-Peterson-Network

- Concrete realization using **two hits** to form a **straight line cell**
- Edge weights set in terms of angular deviation

$$w_{ij} = \frac{\cos^m \alpha_{ij}}{l_i \cdot l_j}$$

Difficulty

Too slow to converge for tracking application

Cellular automaton - 1

Introduced at DESY

Discrete form of Hopfield-Network in CATS (Cellular Automaton for tracking in Silicon) by Kisel for HERA-B

Simplifications over Denby-Peterson-Network

- Make edges **directed** in **forward** particle movement
- $\longrightarrow$ directed loop free graph or **feed forward network**
- Uniform edge weights $w_{ij} = 1$ for allowed edges
- No external excitation $\theta_i = 0$
- Change update scheme
- Cells carry **energy** state E_i (not activation state s_i)

Process

- Update the energy state to highest neighbor energy plus 1

Cellular automaton - schematic

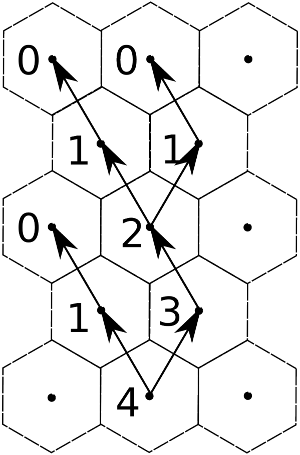


Figure: Cellular automaton in final state

Update rule

$$E_i = \max_{\text{neighbor } j} (E_j + w_{ij}) = \max_{\text{neighbor } j} E_j + 1$$

Output

- **Cells update only once**
- Hight cell marks end of track

Maximzed energy function

(negative of Hopfield-Network)

$$E_{i,\text{final}} = \sum_{\text{max. path to } i} w_{ij} = \#\text{Cells in path} - 1$$

Cellular automaton - 2

Output

- Energy state indicates number of cells in the longest path from this cell
- Create path / track by following the highest states
- Always figures out the longest path without backtracking

Short comings

Weighting scheme quite **unnecessary rigid**

Algorithms to extract paths from graphs

Brute force

- Complete backtracking

Networks and simplifications

- Hopfield-Network
- Denby-Peterson-Network
- **?**
- Cellular automaton

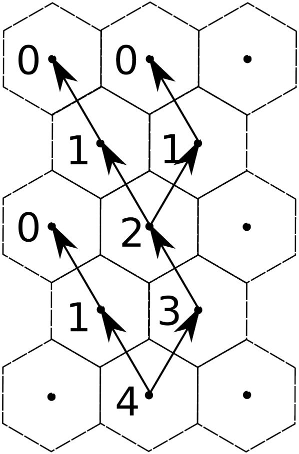


Figure: Cellular automaton in final state

Algorithms to extract paths from graphs

Brute force

- Complete backtracking

Networks and simplifications

- Hopfield-Network
- Denby-Peterson-Network
- **Weighted cellular automaton**
- Cellular automaton

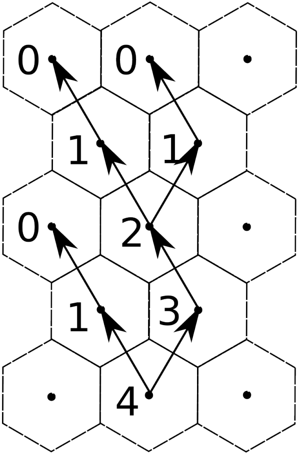


Figure: Cellular automaton in final state

Weighted cellular automaton

Allow weights in graph

- Arbitrary edge weights w_{ij} refines connection quality
- Arbitrary vertex weight θ_i measures quality of cell (useful for complex combound cells)

Update rule

$$E_i = \max_{\text{neighbor } j} (E_j + w_{ij}) + \theta_i$$

Maximized energy function

$$E_{i,\text{final}} = \sum_{\text{best path to } i} w_{ij} + \sum_{\text{best path } i} \theta_i$$

same as Hopfield network!

Weighted cellular automaton - summary

Advantages

- Fast - assignment time only linear in hits $\mathcal{O}(n)$
- Weights allow detailed modeling
- Resembles the Hopfield-Network closely

Small obstacle

- Must enforce loop free condition on the graph

Design decisions

Graph properties

- Directional or adirectional?
- Loop free?
- Symmetric or asymmetric?
- Weighted vertices and/or edges?

Graph vertices

- How do vertices relate to hits?

Graph edges

- How can we find neighboring vertices?
- Can we exploit **geometrical constraints**?
- Which **extrapolation** methods can refine our judgement?

Path / track extraction

Which **algorithms** generate tracks from the graph best?

Design decisions

Graph properties

- Directional
- Loop free!
- Weighted vertices and edges allowed

Graph vertices

- How do vertices relate to hits?

Graph edges

- How can we find neighboring vertices?
- Can we exploit **geometrical constraints**?
- Which **extrapolation** methods can refine our judgement?

Path / track extraction

Weighted cellular automaton

Design decisions

Graph properties

- Directional
- Loop free!
- Weighted vertices and edges allowed

Graph vertices

- How do vertices relate to hits?

Graph edges

- How can we find neighboring vertices?
- Can we exploit **geometrical constraints**?
- Which **extrapolation** methods can refine our judgement?

Path / track extraction

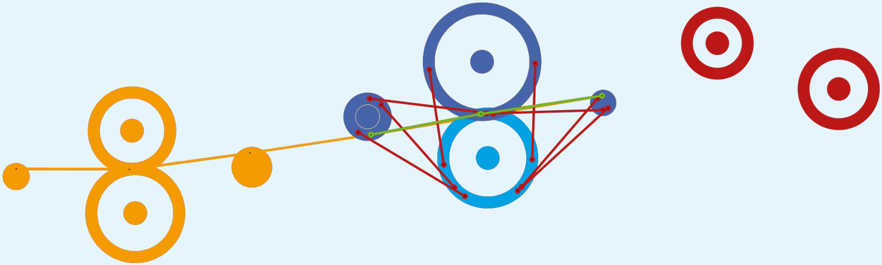
Weighted cellular automaton

Outline

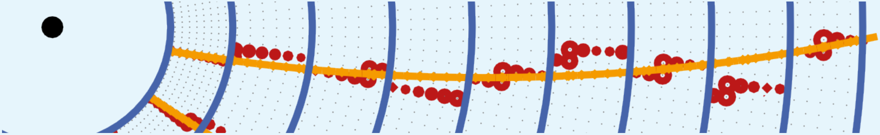
- Detector
- Tracking finding - problem statement
- Local approach
- Generic algorithms
- Concrete Realization
- Fast fitting
- Further work

Bottom-up - A two stage process

Combine hits to segments limited by the superlayer bounds

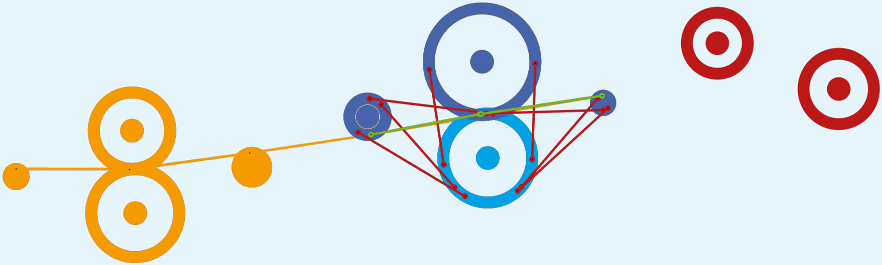


Combine segments to tracks

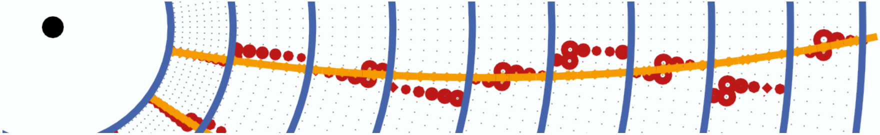


Bottom-up - A two stage process

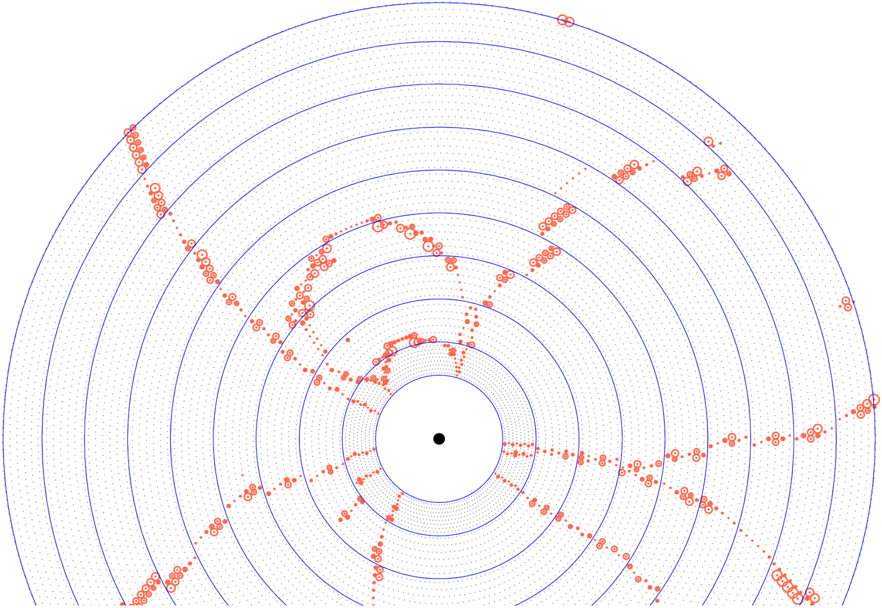
Combine hits to segments limited by the superlayer bounds



Combine segments to tracks



Typical event xy projection



Preparation

Clustering

- Many seperate groups = many smaller graphs
- Generate by expanding minimal hexagonal neighborhood of wires.
- Analyze each cluster (in parallel?)

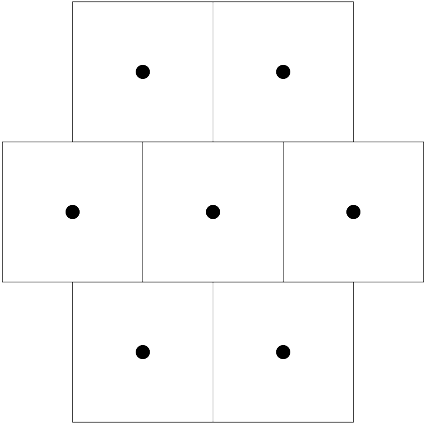
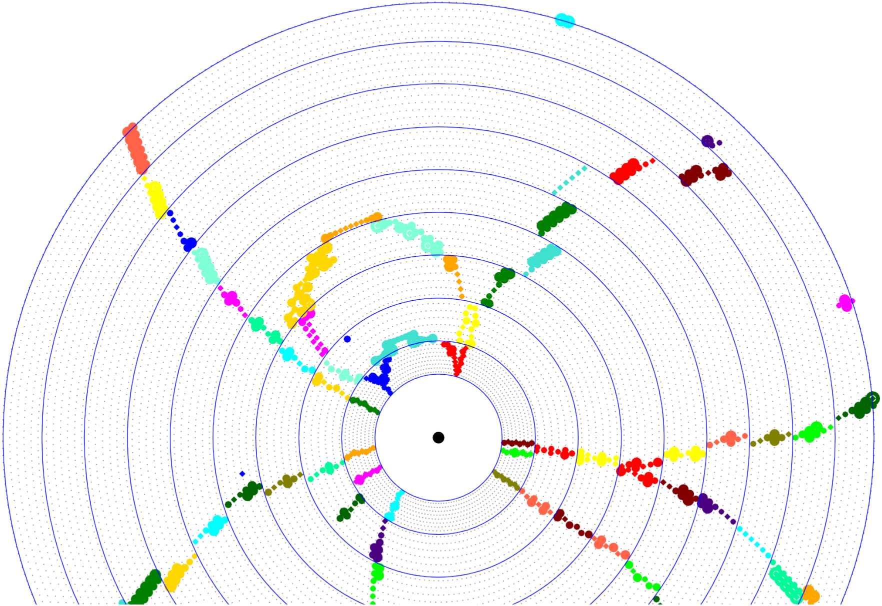


Figure: Nearest six neighbors of a sense wire

Clusters of the typical event



In each cluster - What is a suitable cell?

Vertex / Cell property

Reflect the xy position of the particle

Edges / Neighbor property

Reflect the possible transition from one position to another

Single hits are not the answer

- Position too ambiguous
- No direction of flight information

Static cell neighborhood for hits

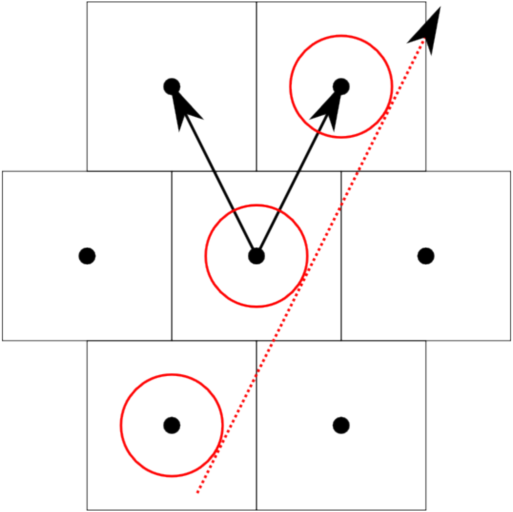


Figure: Static cell neighborhood

Static cell neighborhood for hits

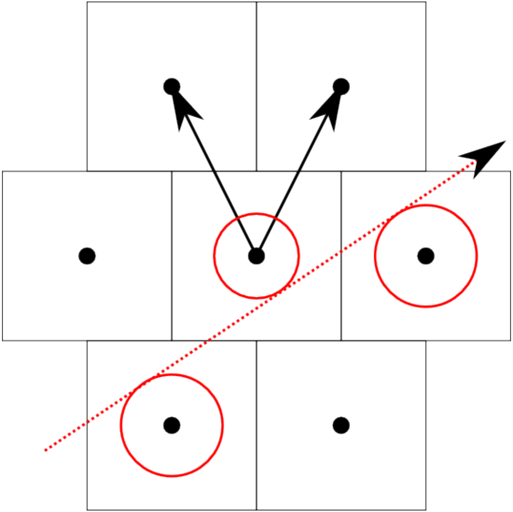


Figure: Static cell neighborhood - cannot follow bend particle trajectories

Introduction of facets

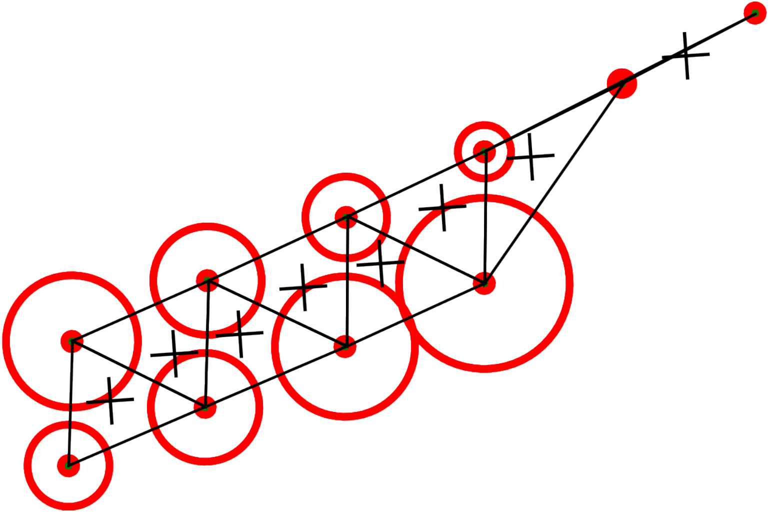


Figure: Idea: Use three hits to triangulate the position of the particle

Types of facets - Ortho

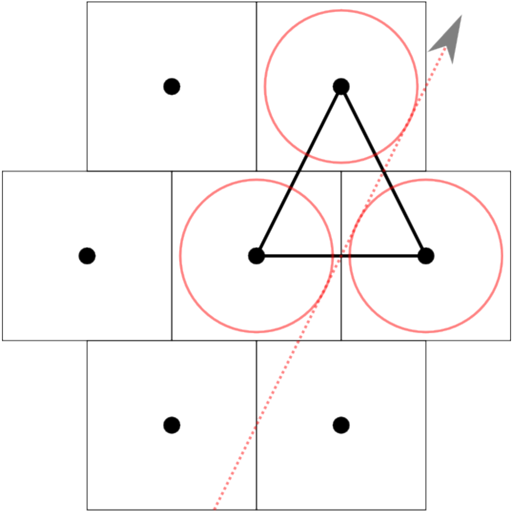


Figure: Naming lend from Benzol derivate Xylol

Types of facets - Meta

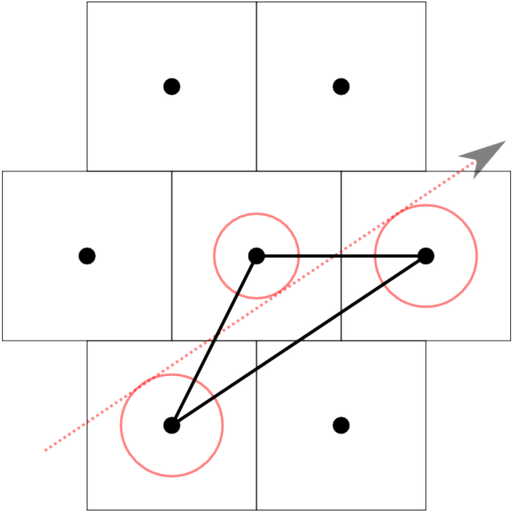


Figure: Naming lend from Benzol derivate Xylol

Types of facets - Para

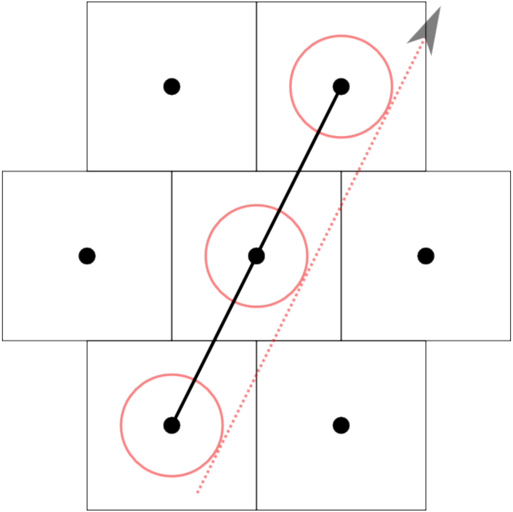


Figure: Naming lend from Benzol derivate Xylol

Facets as cells

Properties of facets

- Ordered triple of neighboring hits
- Each hit has a right-left-passage information assigned to disambiguate trajectory
- **Linear trajectory** by least square fit
- **Residual curvature** over trajectory line

Neighbors of facet

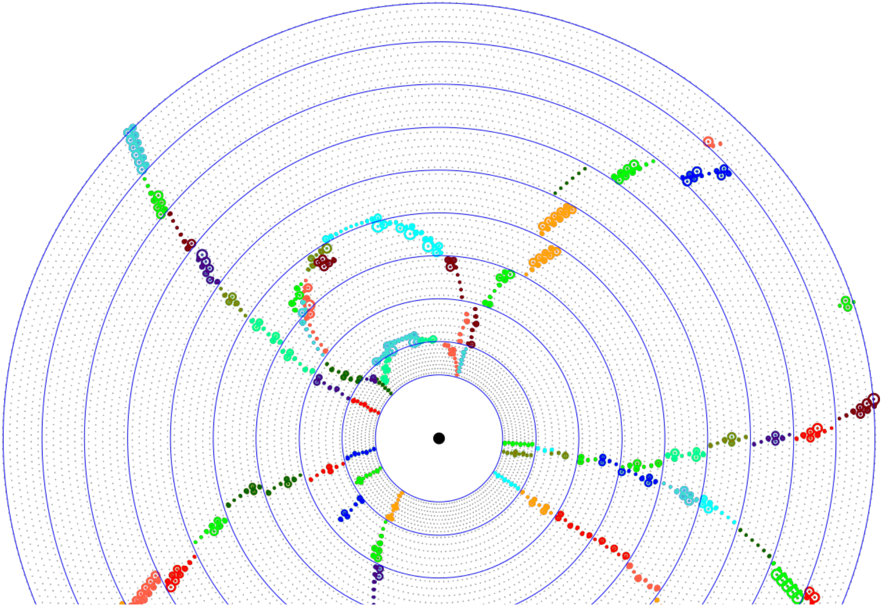
- Neighboring facet have to common hits
- Weight refinement possible in
  - Flight direction
  - Loss cut on the curling direction
  - (Maybe even favouring energy loss in the forward direction)

First stage overview

From hits to segments

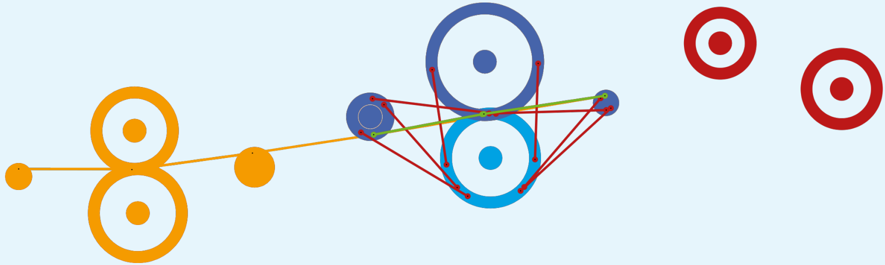
1. Translate the raw data and combined it with the detector geometry.
2. Group the hits into **clusters**.
3. For each cluster:
   3.1 Build triples of wire hits, called **facets, as cells** to be given to the cellular automaton.
   3.2 Construct the weighted graph edges by searching **connections** of each **facet**.
   3.3 Retrieve the **paths** from the **cellular automaton** in a **multi-pass** manner.
   3.4 Reduce the **paths** of facets **to segments**.

Typical event after the first stage

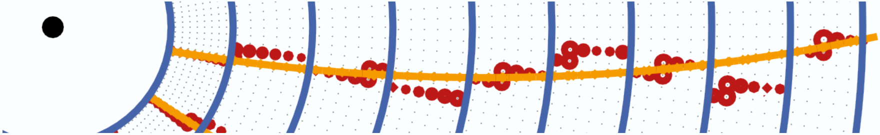


Bottom-up - A two stage process

Combine hits to segments limited by the superlayer bounds

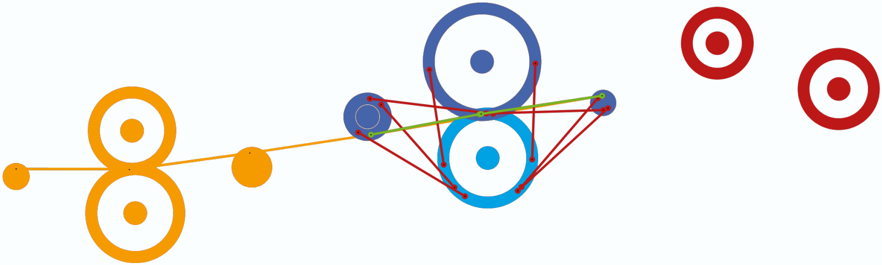


Combine segments to tracks

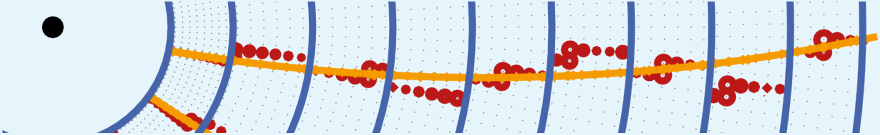


Bottom-up - A two stage process

Combine hits to segments limited by the superlayer bounds



Combine segments to tracks



What is a suitable cell?

Vertex / Cell property

Reflect the 3D position of the particle

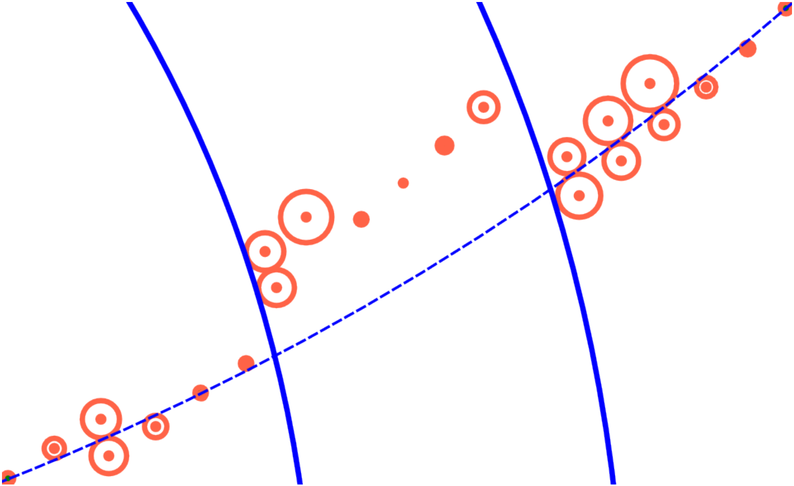
Edges / Neighbor property

Reflect the possible transition from one position to another (requires extrapolation across superlayer bounds)

Its not single segments

- No z information
- Only available by comparing axial with stereo segments

Introduction of segment triples



Segment triples of cells

Properties of segment triples

- Ordered triple of segments in arrangment axial - stereo - axial
- **Riemann circle fit** to the two axial segments
- Reconstructed z information of the stereo segment
- Linear z over travel distance fit

Neighbors of segment triples

- Neighboring segment triples have one axial segment in common
- Weight refinement possible in
  - extrapolated xy position
  - extrapolated z displacement
  - momentum.

Second stage overview

From segments to tracks

1. Build **triples of segments as cells** to be given to the cellular automaton.
2. Construct the graph edges by searching **neighbors** of each **segment triple**.
3. Retrieve the **paths** from the **cellular automaton** in a **multi-pass** manner.
4. Reduce the **paths** of segment triples to three dimensional **tracks.**
5. Decide, whether the tracks should interpreted as **reversed**.
6. **Export to** track candidates, which can be fitted by **Genfit** algorithm.

Outline

- Detector
- Tracking finding - problem statement
- Local approach
- Generic algorithms
- Concrete Realization
- Fast fitting
- Further work

Fast track fitting and extrapolation

Iterative methods

- Non-linear optimization
- Newton algorithm ...
- Kalman filter

Noniterative methods

- Least square fitting
- + Transformation to appropriate space

properties

- slower (many steps)
- accurate
- unbiased
- needs initial parameters

properties

- faster
- approximate
- may be biased
- yields initial parameters

Conformal projection

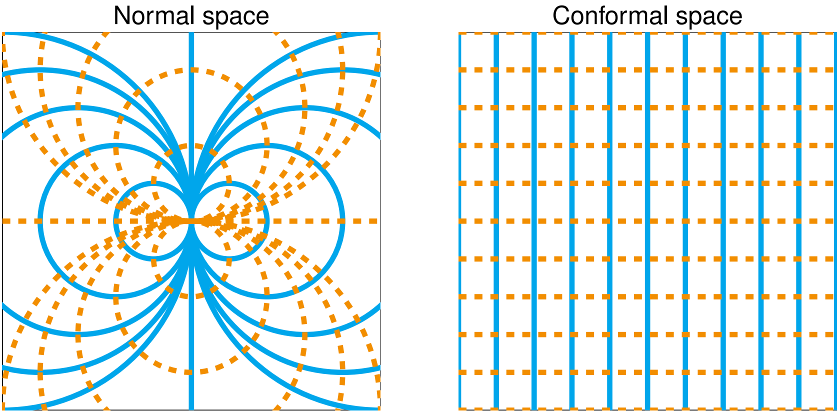
Formula

$$X = \frac{x}{x^2 + y^2}$$
$$Y = \frac{x}{x^2 + y^2}$$

Properties

- Maps generalized circles to generalized circles
- Maps circles through the origin to lines
- $\rightarrow$ Only suitable for tracks coming from the origin

The conformal map revealed



Conformal projection - distortion

Distortion

$$D \approx \frac{d}{r^2}$$

Fix

Reweighting with

$$w = \frac{1}{r^4}$$

Big Brother - Stereographic projection

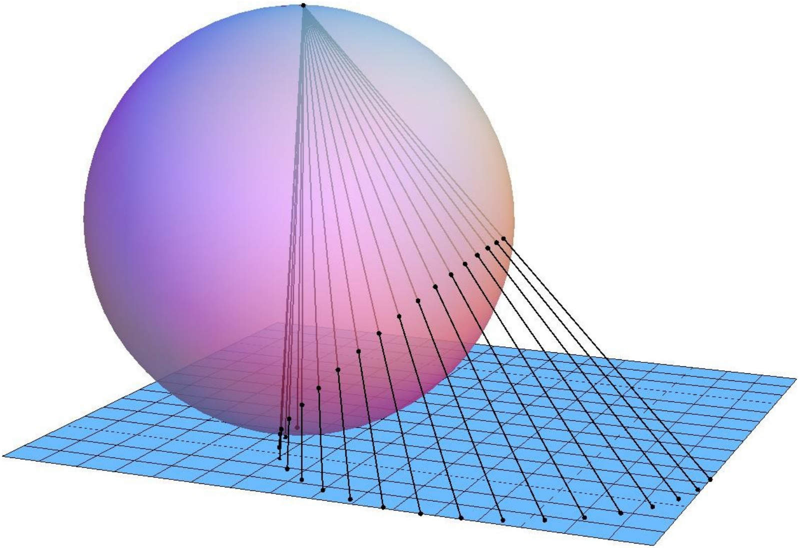
Formula

$$U = \frac{x}{r^2 + 1}$$
$$V = \frac{y}{r^2 + 1}$$
$$W = \frac{r^2}{r^2 + 1}$$

Properties

- Maps the 2D plane to a 3D unit sphere surface.
- Maps generalized circles to circles on this sphere.
- $\rightarrow$ All points of circle are in one plane after the projection.
- Suitable for all kinds of tracks

The stereographic projection revealed



Stereographic projection - distortion

Distortion

$$D \approx \frac{d}{(r+1)^2}$$

Fix

Reweighting with

$$w = \frac{1}{(r+1)^4}$$

Even simpler - parabolic projection

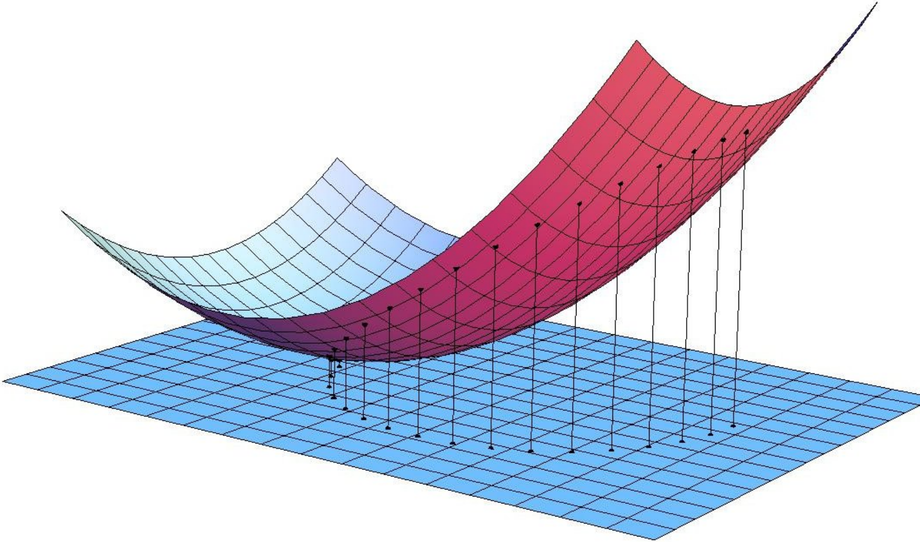
Formula

$$U = x$$
$$V = y$$
$$W = r^2$$

Properties

- Maps the 2D plane to a 3D parabolic surface
- Still maps circles and lines into plane in the 3D world
- Suitable for all kind of tracks
- Easily constrainable

The parabolic projection revealed



Parabolic projection - distortion

No distortion (in first order)

$$D \approx d + \mathcal{O}\left(\frac{d^2}{\text{circle radius}}\right)$$

Properties of fit

- Distances from plane $=$ distance to fit circle in 2D in 1st order $\frac{1}{R}$
- Accurate, since big circles and small distances are expected.
- Use a least square fit to minimze

$$S = \sum_i (n_0 + U \cdot n_1 + V \cdot n_2 + W \cdot n_3)^2$$

- Computable by single matrix inversions or SVD decomposition
- Fast
- Enables extrapolation
- Easily constrainable to tracks from interaction point

Are you left or right ?

Orientation matters

- One can enhance the fit with the drift circle radii ρ

$$S = \sum_i (d_i \pm \rho_i)^2 = \sum_i (n_0 + x_i \cdot n_1 + y_i \cdot n_2 + r_i^2 \cdot n_3 \pm \rho_i)^2$$

- Use plus or minus, if you want the point left or right of the circle.
- Right left information from the tangents used to build the segment

Outline

- Detector
- Tracking finding - problem statement
- Local approach
- Generic algorithms
- Concrete Realization
- Fast fitting
- Further work

Further work

Evaluation

- Compare found tracks and segments to Monte Carlo information
- Optimize the weights in the two stages.
- Check the unbiasedness of fitting procedure (I suspected a numerical instability biasing to low curvature)

Profiling

- Improve the time performance of each step.

Outline

- Reference
- Comparing to the reference

What is the benchmark for tracking?

- What are reconstructable particles?
- What hit contents does the ideal track have?
- What momentum and vertex should be reconstructed most accurate?
- Are we convering secondary particles correctly?

MCTrackFinder

Make sure it yields our definition of ideal!
(If there is more than one kind of ideal, make all available!)

What is the benchmark for tracking?

- What are reconstructable particles?
- What hit contents does the ideal track have?
- What momentum and vertex should be reconstructed most accurate?
- Are we convering secondary particles correctly?

MCTrackFinder

Make sure it yields our definition of ideal!
(If there is more than one kind of ideal, make all available!)

Additions to MCTrackFinder

- Default: Inclusion of secondary particles with enough hits.
- Optional: Momentum at first measurement.
- Optional: Selection of a tag side.
- ...

Outline

- Reference
- Comparing to the reference

Matching to the reference

Matching challenge

Assume a 1 $\leftrightarrow$ 1 relation from
#n Monte Carlo tracks

$\leftrightarrow$

#m Pattern recognition tracks

Matching by hit content

Choose best items of the confusion matrix.

Confusion matrix

	MC tracks			Background
PR tracks	...	...	...	...
	...	Hit / NDF content	...	...
	...		...	...
	...	...	...	...
Unassigned	...	...	...	...

Row-wise matching - purity matching

- Search highest purity Monte Carlo track for each pattern recognition track.
- Look for highest entry in each row.
- Relation *hp*: 1 $\leftrightarrow$ n

Column-wise matching - efficiency matching

- Search highest efficiency pattern recognition track for each Monte Carlo track.
- Look for highest entry in each column.
- Relation *he*: 1 $\leftrightarrow$ m

Confusion matrix

	MC tracks			Background
	…	…	…	…
PR tracks	…	Hit / NDF content	…	…
	…		…	…
	…	…	…	…
Unassigned	…	…	…	…

Row-wise matching - purity matching

- Search highest purity Monte Carlo track for each pattern recognition track.
- Look for highest entry in each row.
- Relation *hp*: $1 \leftrightarrow n$

Column-wise matching - efficiency matching

- Search highest efficiency pattern recognition track for each Monte Carlo track.
- Look for highest entry in each column.
- Relation *he*: $1 \leftrightarrow m$

Confusion matrix

	MC tracks			Background
	…	…	…	…
PR	…	Hit / NDF	…	…
tracks	…	content	…	…
	…	…	…	…
Unassigned	…	…	…	…

Row-wise matching - purity matching

- Search highest purity Monte Carlo track for each pattern recognition track.
- Look for highest entry in each row.
- Relation *hp*: 1 $\leftrightarrow$ n

Column-wise matching - efficiency matching

- Search highest efficiency pattern recognition track for each Monte Carlo track.
- Look for highest entry in each column.
- Relation *he*: 1 $\leftrightarrow$ m

Two sided matching

Two sided matching - concatination of the former

- Highest purity and highest efficiency relation agree!
- The highest purity Monte Carlo track mc_2 of the highest efficiency pattern recognition track of Monte Carlo track mc_1 is the same as the Monte Carlo track mc_1.

$$mc_2 := hp(he(mc_1)) = mc_1$$

- Or equivalent: The highest efficiency pattern recognition track pr_2 of the highest purity Monte Carlo track of pattern recognition track pr_1 is the same as the pattern recognition track pr_1.

$$pr_2 := he(hp(pr_1)) = pr_1$$

- Relation: $1 \leftrightarrow 1$

Classification

Classification of pattern recognition tracks pr_i

Ghost Highest purity is smaller than acceptable contamination threshold 0.66.

Background $he(mc_i)) =$ background column

Clone $hp(he(mc_i)) \neq mc_i$

Matched $hp(he(mc_i)) == mc_i$

Classification of Monte Carlo tracks mc_i / MCParticles

Missing $he(mc_i)) =$ bad purity pattern recognition track / unassigned column

Merged $hp(he(mc_i)) \neq mc_i$

Matched $hp(he(mc_i)) == mc_i$

New module : MCTrackMatcherModule

Input

1. Ideal Monte Carlo tracks
2. Pattern recognition tracks

Output for subsequent evalutation

1. Highest purity relation (negative for clone PR tracks)
2. Highest efficiency relation (negative for merged MC tracks)
3. Pattern recognition tracks to MCParticle relation
4. The McTrackId property of the pattern recognition tracks

Options

1. Usage of detectors
2. Switch for ghost assignement to MCParticles

Short term goal - Common combinatorial evaluation

- Ghost rate
- Tracked background rate
- Clone rate
- Missing rate
- Merged rate
- Matched rate

by

- Multiplizity
- p_t
- *PDG* code

in

- Gun events with muons, pions,... ,
- specific decay $B \rightarrow K\pi\pi\pi$ and
- generic events