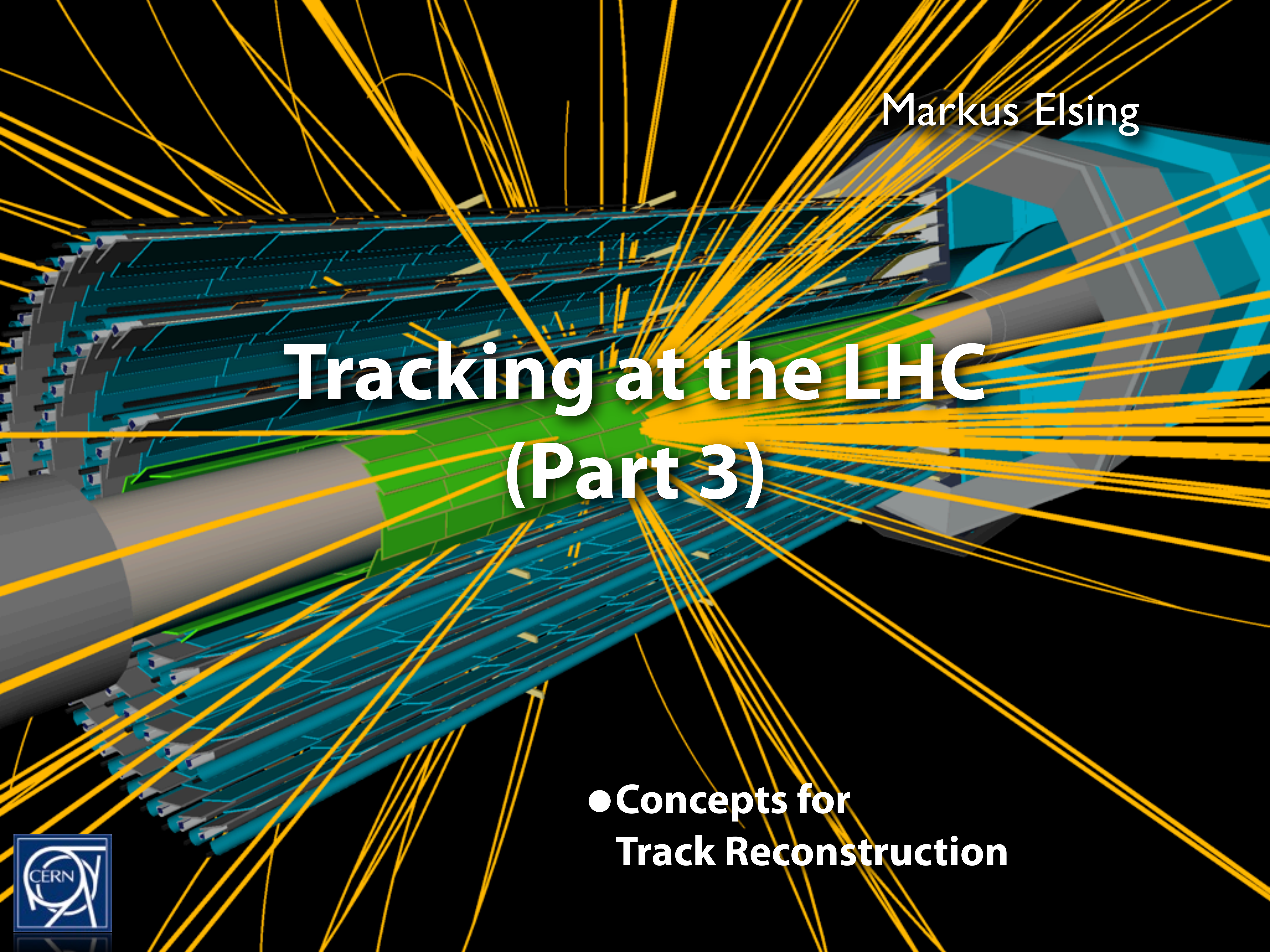


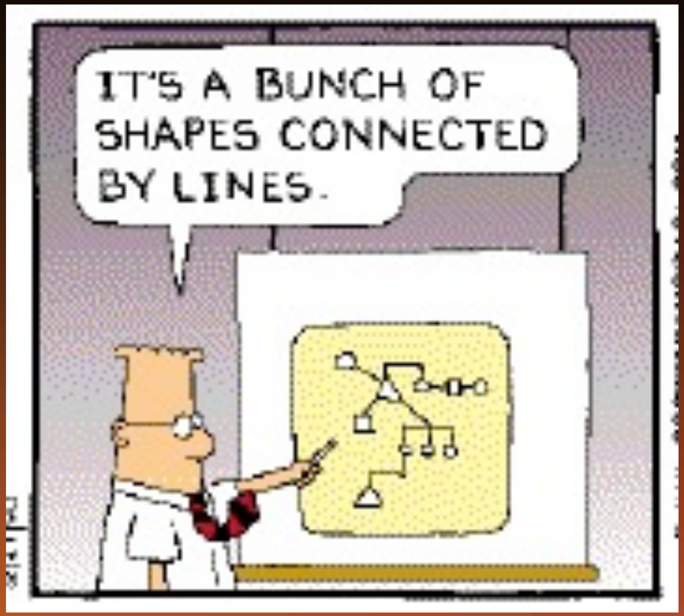
Markus Elsing

Tracking at the LHC
(Part 3)

- **Concepts for Track Reconstruction**

Introduction:

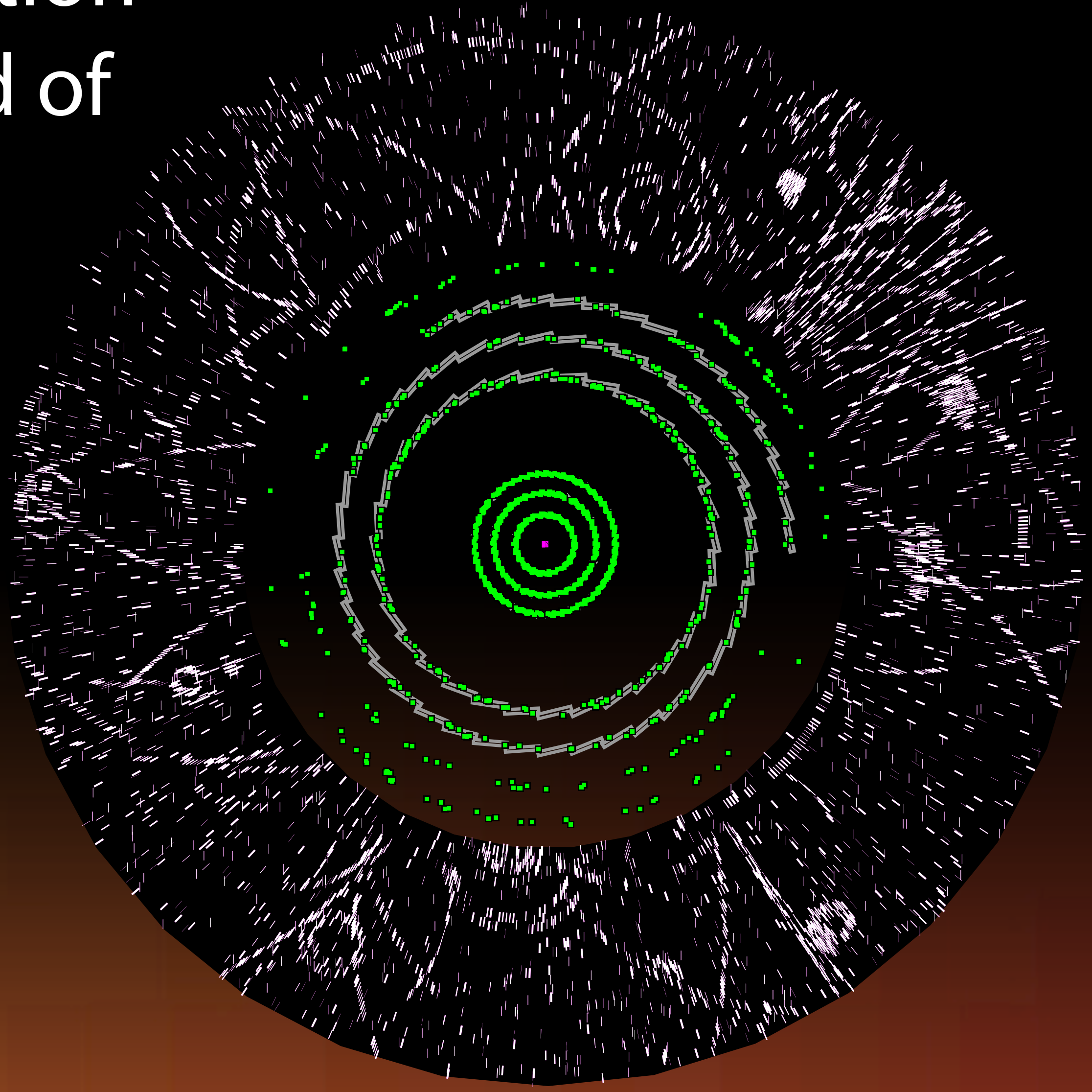
- in this lecture I will discuss the concepts of track reconstruction
- will have to introduce various techniques for
  - ➡ pattern recognition, detector geometry, track fitting, extrapolation ...
  - ➡ including mathematical concepts and aspects of software design



... so why does it matter ?

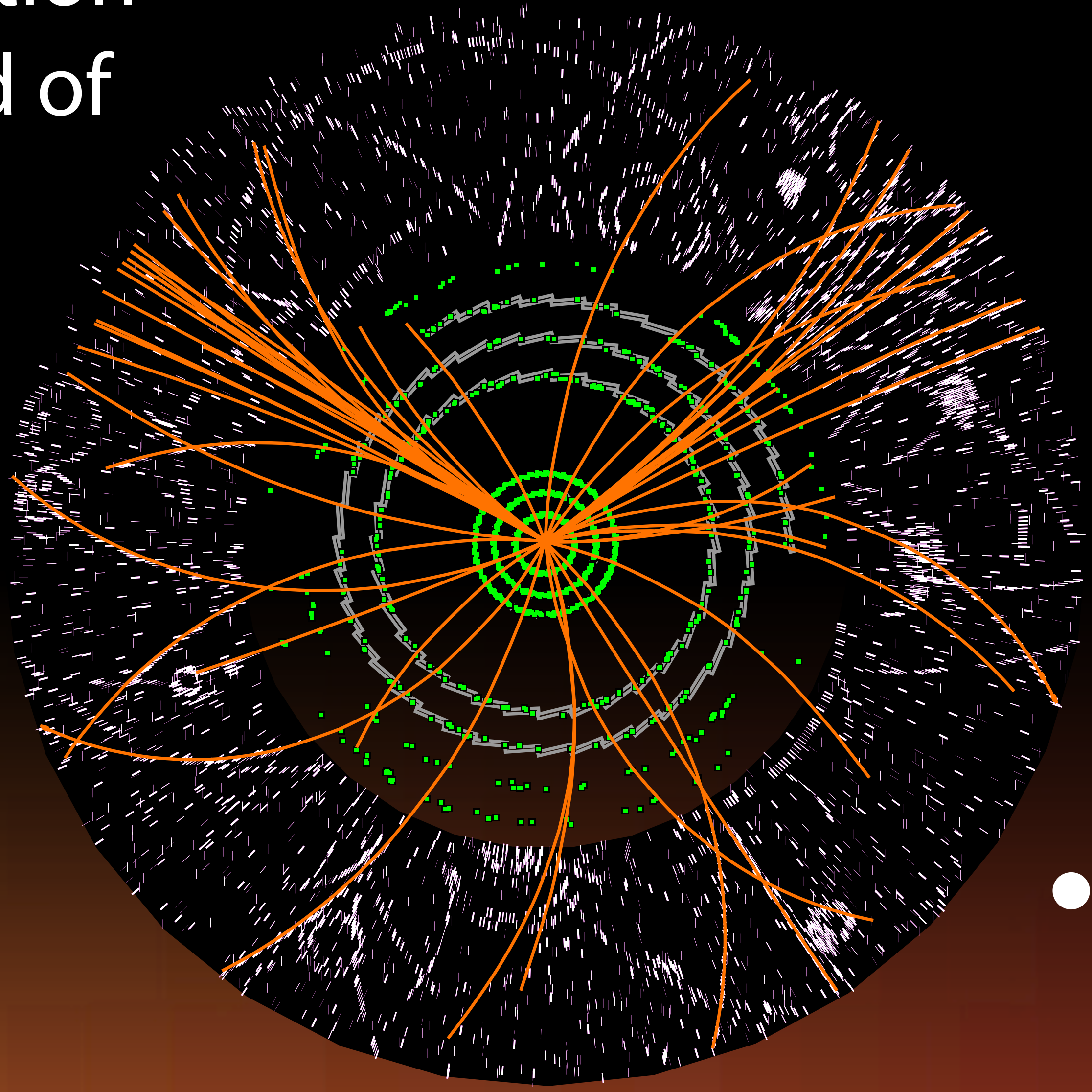
The Tracking Problem

- particles produce in an interaction leave a cloud of hits in the detector



The Tracking Problem

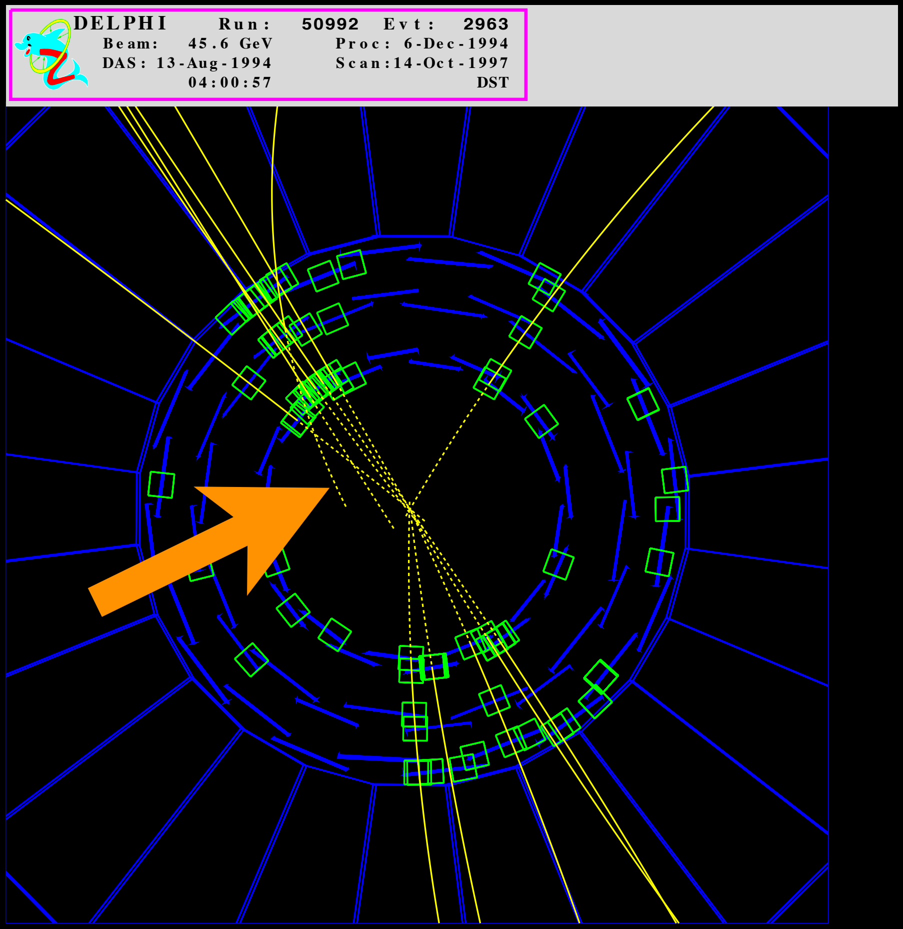
- particles produce in an interaction leave a cloud of hits in the detector

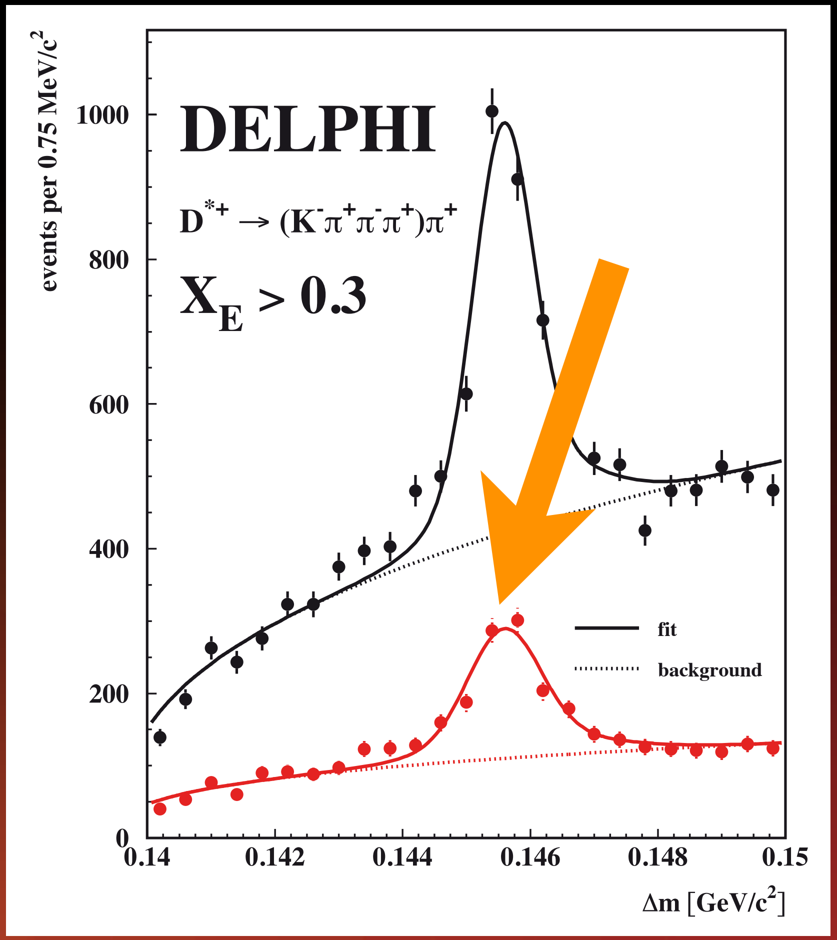


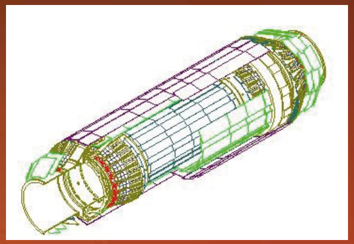
- tracking software is used to reconstruct their trajectories

Role of Tracking Software

- optimal tracking software
  - ➡ required to fully explore performance of detector
- example: DELPHI Experiment at LEP
  - ➡ silicon vertex detector upgrade
  - ➡ initially not used in tracking to resolve dense jets
    - pattern mistakes in jet-chamber limit performance



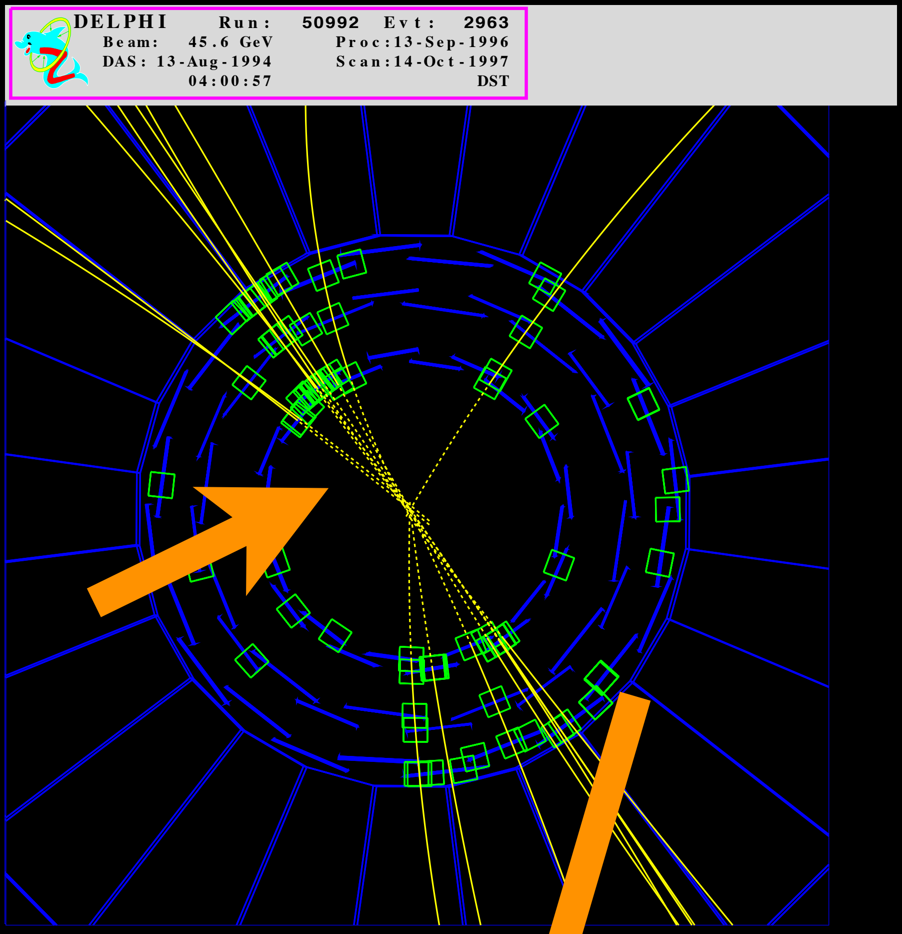


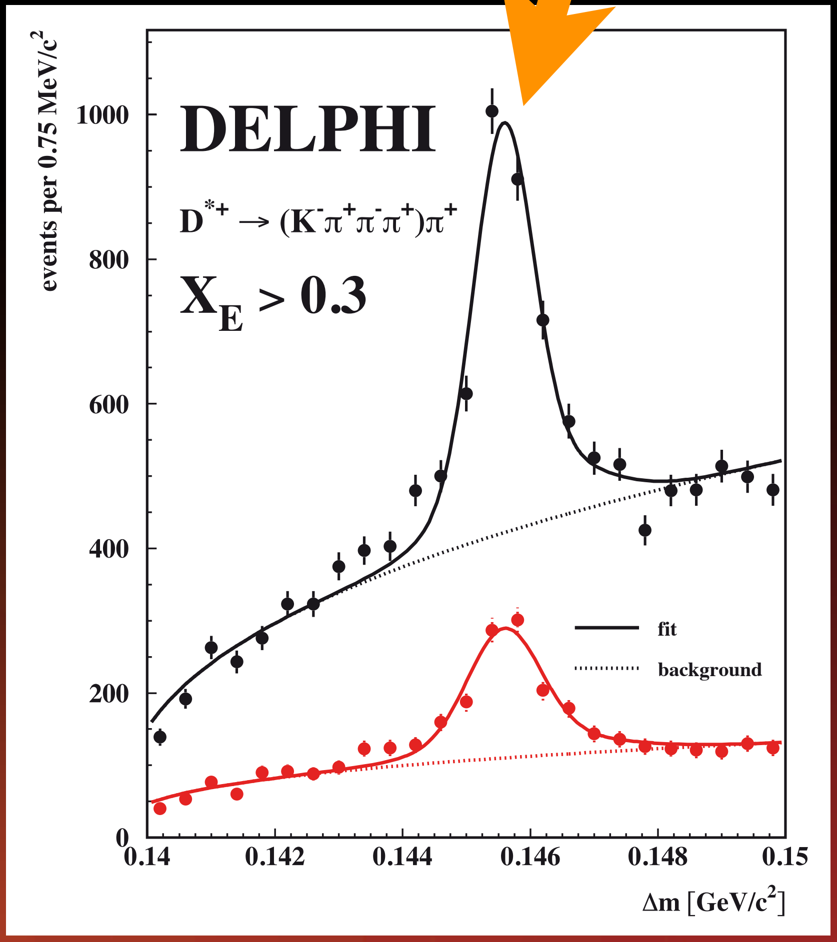


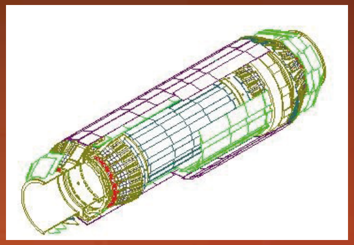
Role of Tracking Software

- optimal tracking software
  - ➡ required to fully explore performance of detector
- example: DELPHI Experiment at LEP
  - ➡ silicon vertex detector upgrade
  - ➡ initially not used in tracking to resolve dense jets
    - pattern mistakes in jet-chamber limit performance
  - ➡ 1994: redesign of tracking software
    - start track finding in vertex detector
    - correct jet-chamber information
  - ➡ **factor ~ 2.5 in D* acceptance** after reprocessing

(M.Feindt, M.E. et al)







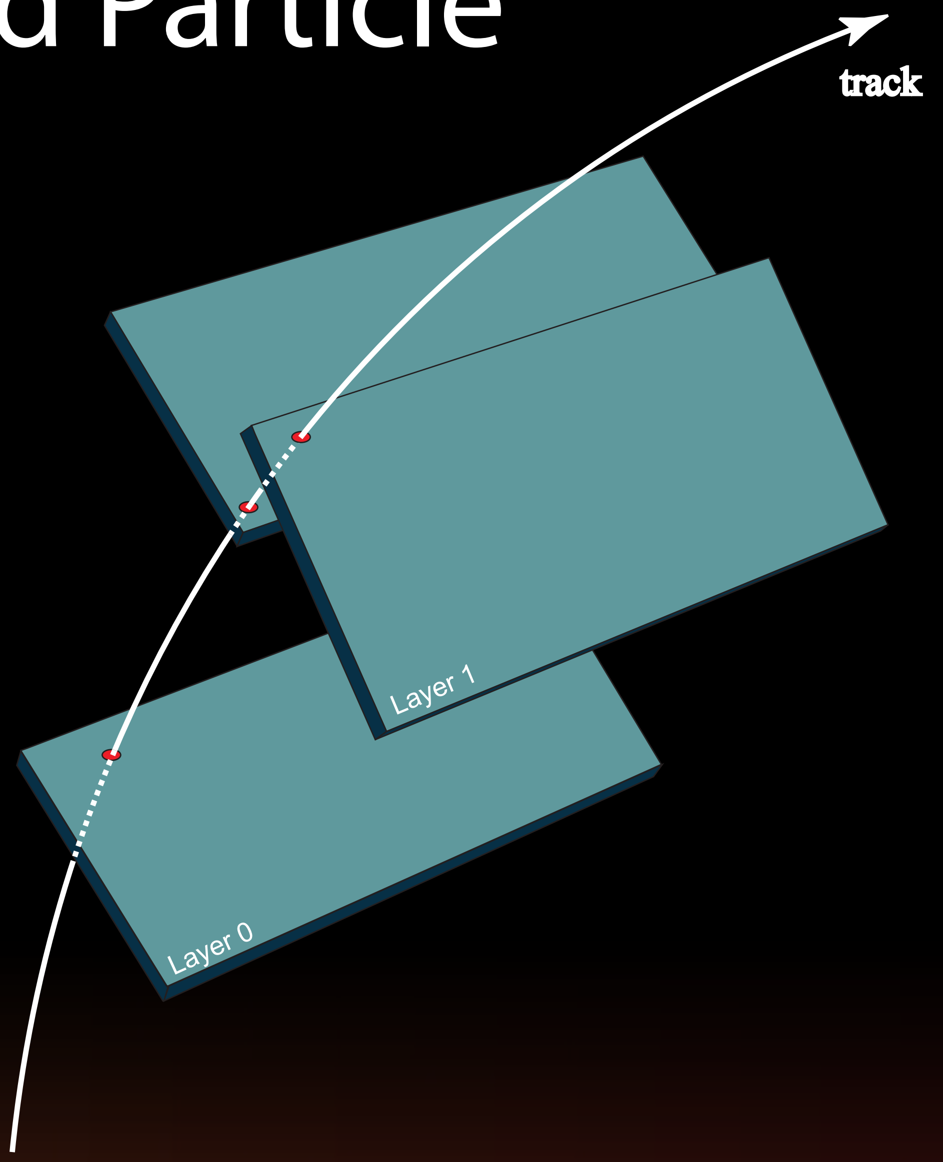
Outline of Part 3

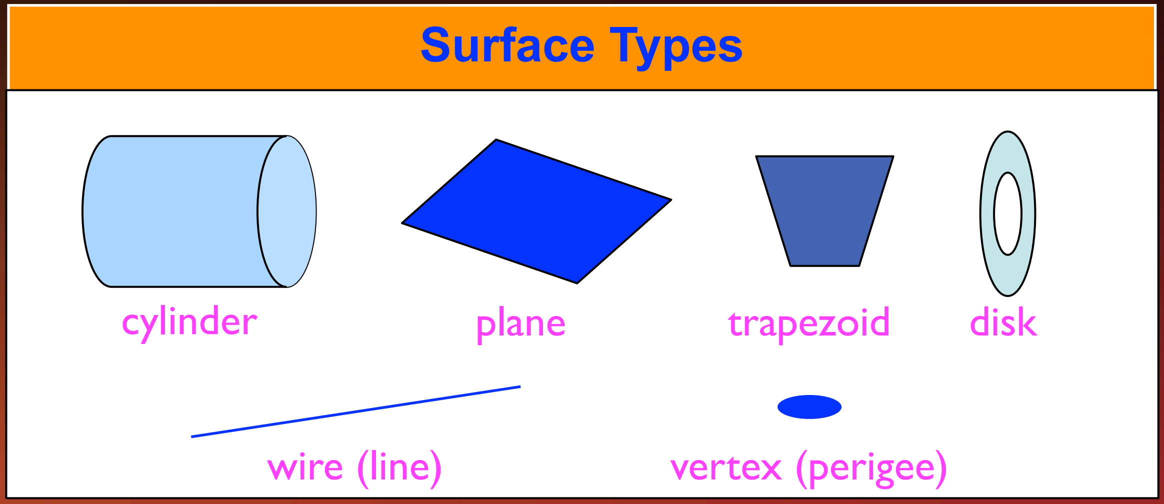
- charged particle trajectories and extrapolation
  - ➡ trajectory representations and trajectory following in a realistic detector
  - ➡ detector description, navigation and simulation toolkits
- track fitting
  - ➡ classical least square track fit and a Kalman filter track fit
  - ➡ examples for advanced techniques
- track finding
  - ➡ search strategies, Hough transforms, progressive track finding, ambiguity solution
  - ➡ as an example, the ATLAS track reconstruction

A Trajectory of a Charged Particle

- ➡ in a solenoid B field a charged particle trajectory is describing a **helix**
  - a circle in the plane perpendicular to the field (Rϕ)
  - a path (not a line) at constant polar angle (θ) in the Rz plane
- ➡ a trajectory in space is defined by **5 parameters**
  - the **local position** (l_1,l_2) on a plane, a cylinder, ..., on the surface or reference system
  - the **direction** in θ and ϕ plus the **curvature** Q/P_T
- ➡ ATLAS choice:

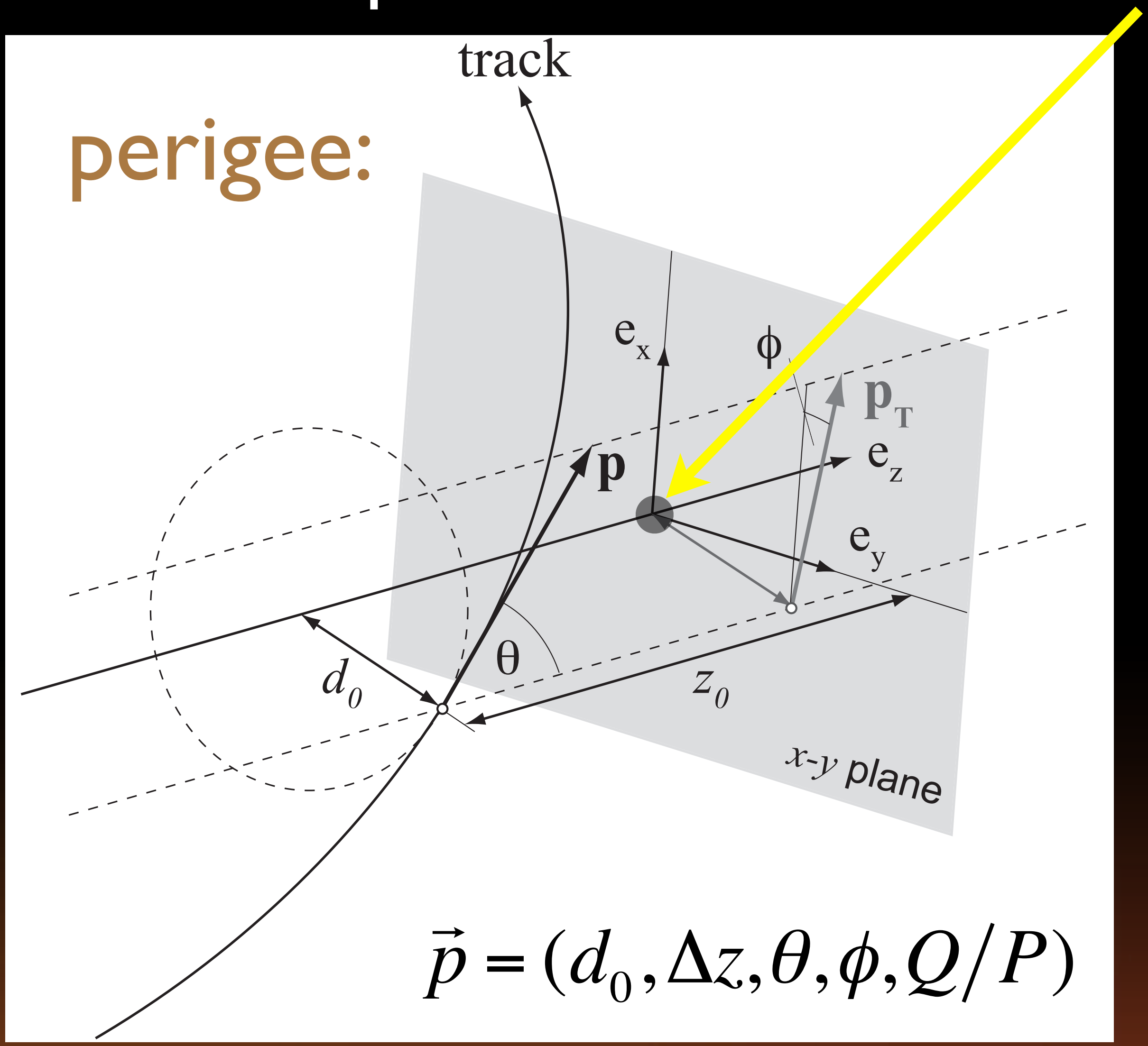
$$\vec{p} = (l_1, l_2, \theta, \phi, Q/P)$$





The Perigee Parameterization

- helix representation w.r.t. a vertex

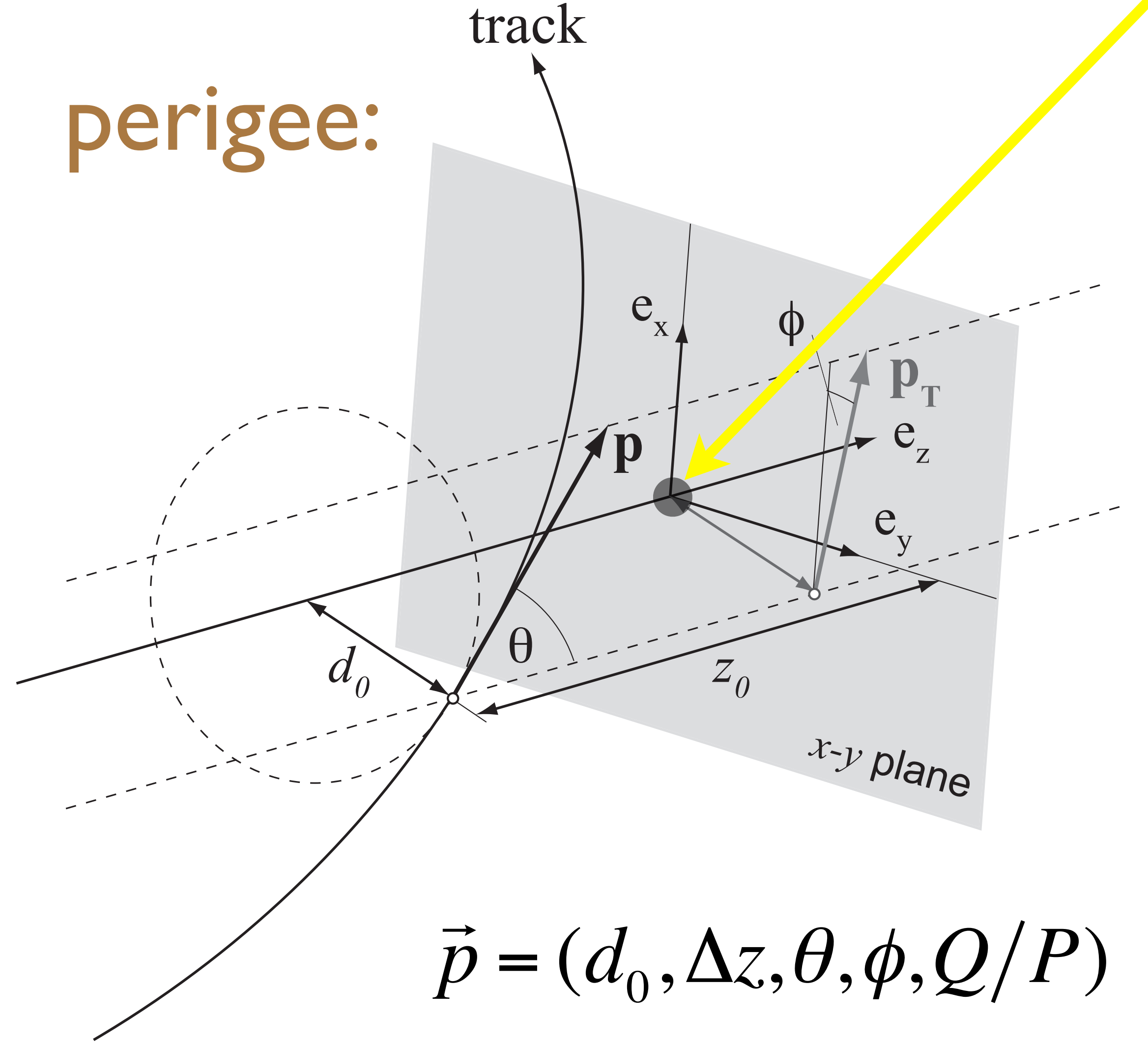


$$\vec{p} = (d_0, \Delta z, \theta, \phi, Q/P)$$

- commonly used
  - ➡ to express track parameters near the production vertex
  - ➡ in implementations of vertex finding algorithms
  - ➡ as well in b-tagging codes

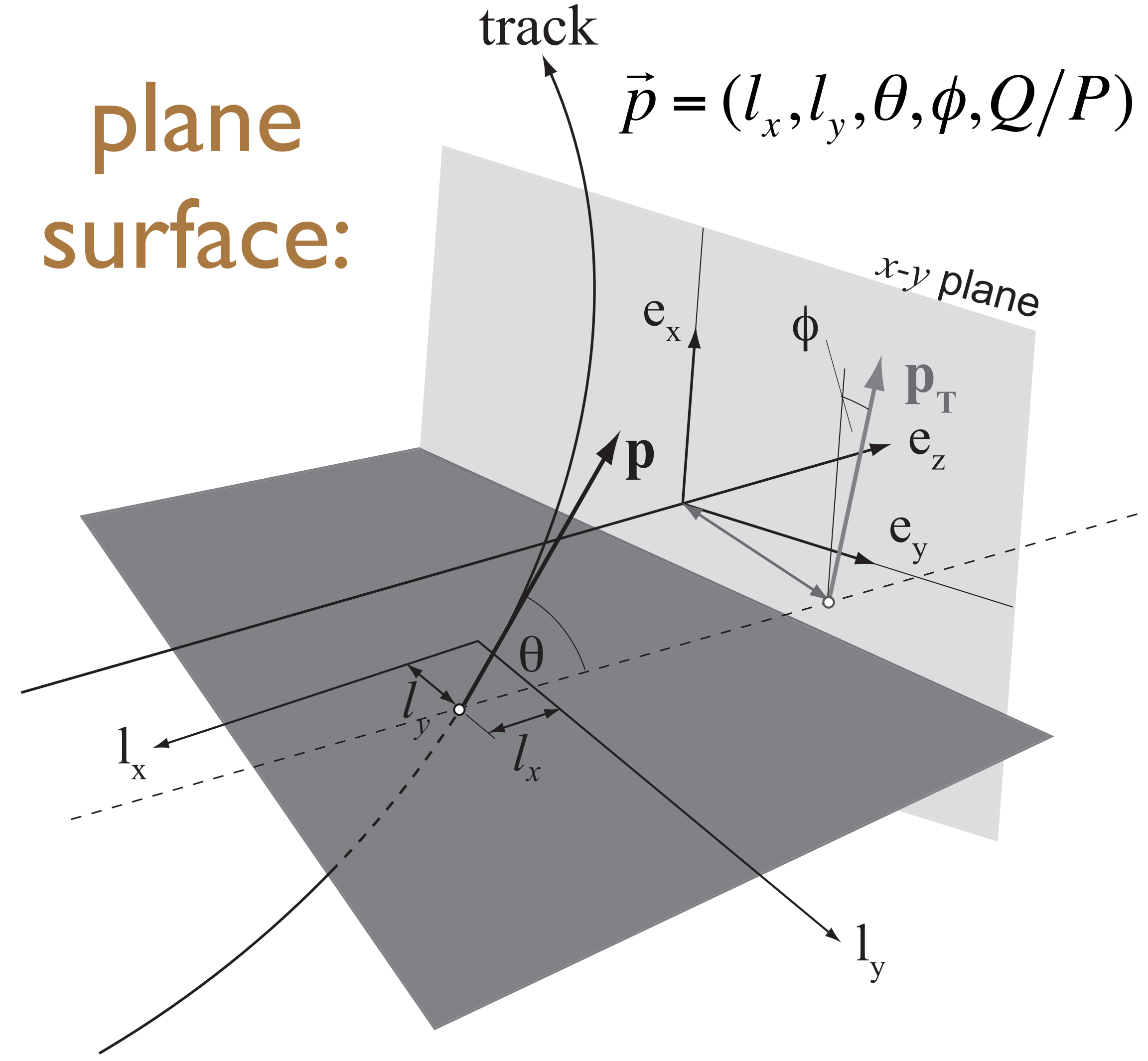
The Perigee Parameterization

- helix representation w.r.t. a vertex



perigee:

$$\vec{p} = (d_0, \Delta z, \theta, \phi, Q/P)$$



plane surface:

$$\vec{p} = (l_x, l_y, \theta, \phi, Q/P)$$

- commonly used
  - ➡ to express track parameters near the production vertex
  - ➡ in implementations of vertex finding algorithms
  - ➡ as well in b-tagging codes

Following the Particle Trajectory

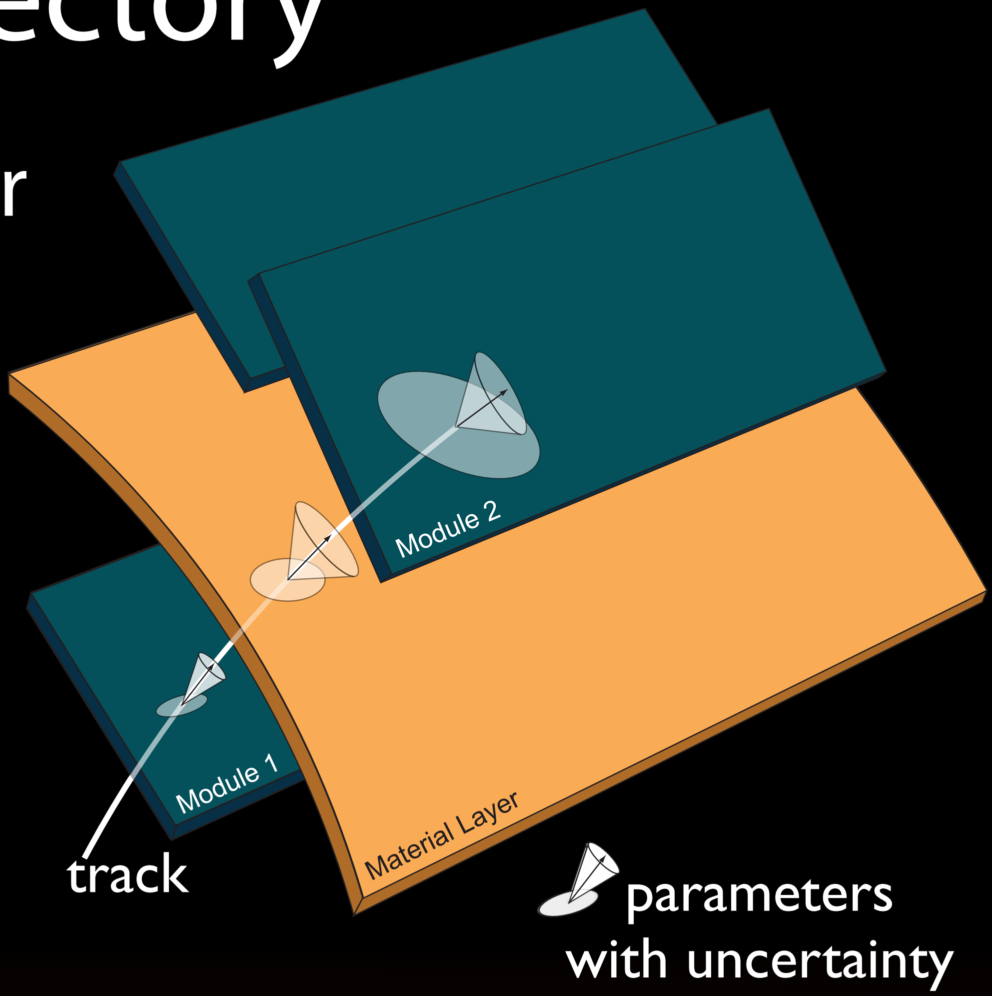
- basic problems to be solved in order to follow a track:
  - ➡ next detector module that it intersects ?
  - ➡ what are its parameters on this surface ?
  - ➡ what is the uncertainty of those parameters ?
  - ➡ for how much material do I have to correct ?

- requires:
  - ➡ a **detector geometry**
    - surfaces for active detectors
    - passive material layers
  - ➡ a method to discover which is the next surface (**navigation**)
  - ➡ a **propagator** to calculate the new parameters and its errors
    - often referred to as "track model"

- for a constant B-field (or no field)
  - ➡ an analytical formula can be calculated for an intersection of a helix (or a straight line) on simple surfaces (plane, cylinder, vertex,...)

Following the Particle Trajectory

- basic problems to be solved in order to follow a track:
  - ➡ next detector module that it intersects ?
  - ➡ what are its parameters on this surface ?
  - ➡ what is the uncertainty of those parameters ?
  - ➡ for how much material do I have to correct ?

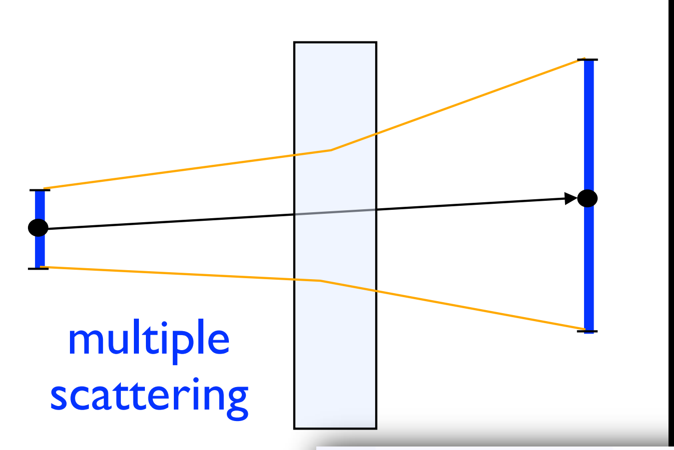


- requires:
  - ➡ a **detector geometry**
    - surfaces for active detectors
    - passive material layers
  - ➡ a method to discover which is the next surface (**navigation**)
  - ➡ a **propagator** to calculate the new parameters and its errors
    - often referred to as “track model”

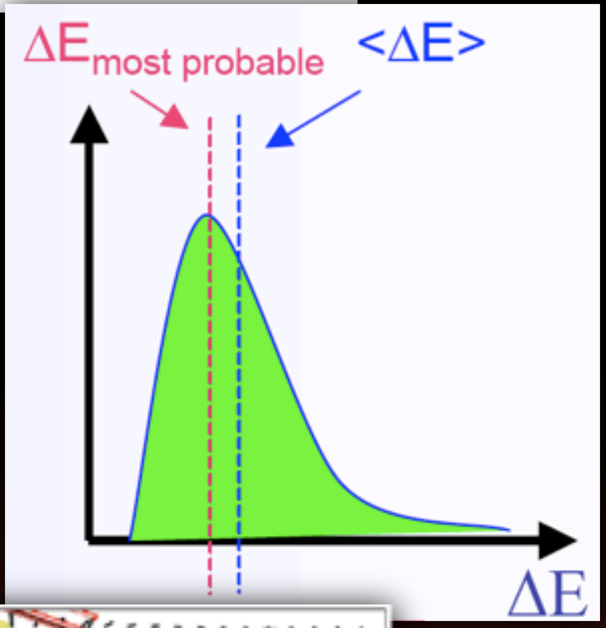
- for a constant B-field (or no field)
  - ➡ an analytical formula can be calculated for an intersection of a helix (or a straight line) on simple surfaces (plane, cylinder, vertex,...)

Material Effects and Realistic B-Field

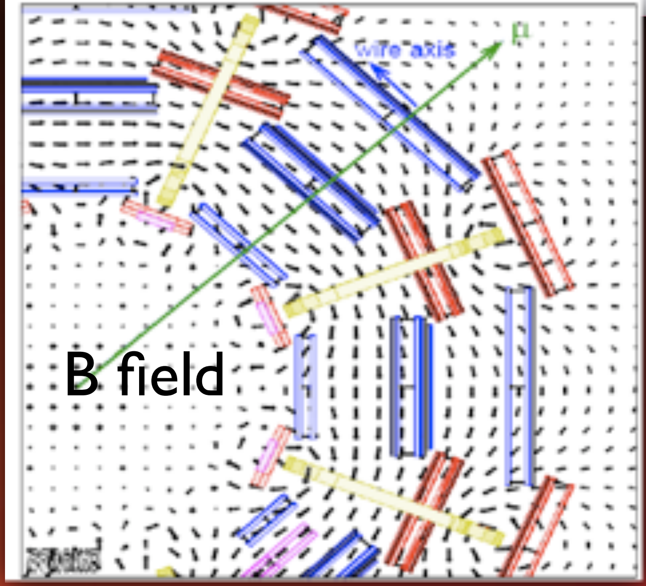
- multiple scattering
  - ➡ increases **uncertainty on direction** of track
  - ➡ for given x/X_0 traversed add term to covariances of θ and ϕ on a material "layer"



- energy loss
  - ➡ use most **probably energy loss** for x/X_0
  - ➡ correct momentum (curvature) and its covariance

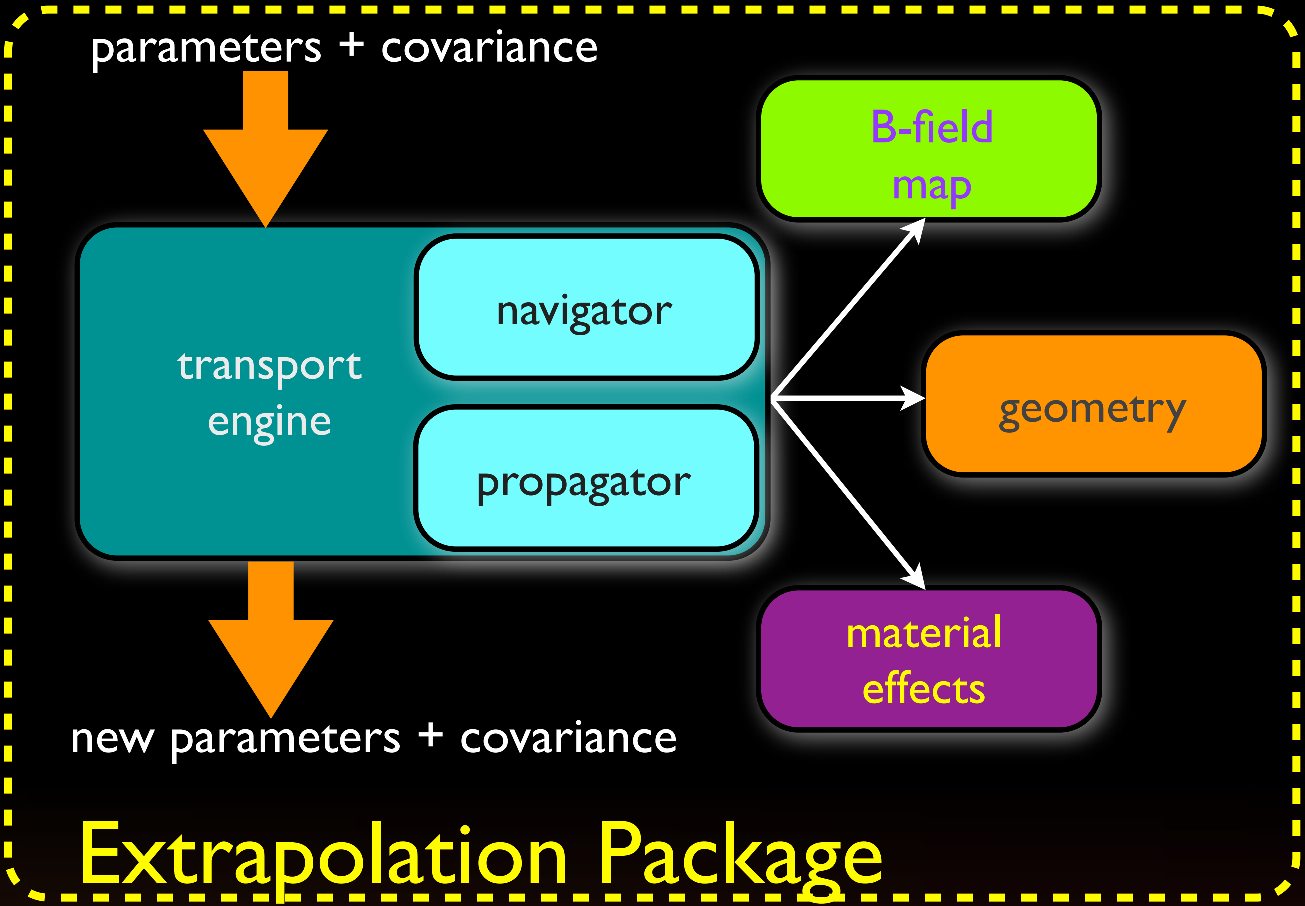


- realistic non-homogeneous B-field
  - ➡ analytical helix propagation has to be replaced by numerical B-field integration along the path of the trajectory
  - ➡ in ATLAS and CMS a 4th order adaptive **Runge-Kutta-Nystrom** approach is used
  - ➡ propagates covariance matrix in parallel *(Bugge, Myrheim, 1981, NIM 179, p.365)*
    - ‣ for experts: muon reconstruction in ATLAS+CMS uses the STEP track model with continuous energy loss and multiple scattering



The Track Extrapolation Package

- a transport engine used in tracking software
  - ➡ central tool for pattern recognition, track fitting, etc.
  - ➡ parameter transport from **surface to surface**, including covariance
  - ➡ encapsulates the track model, geometry and material corrections



parameters + covariance

transport engine

navigator

propagator

B-field map

geometry

material effects

new parameters + covariance

Extrapolation Package

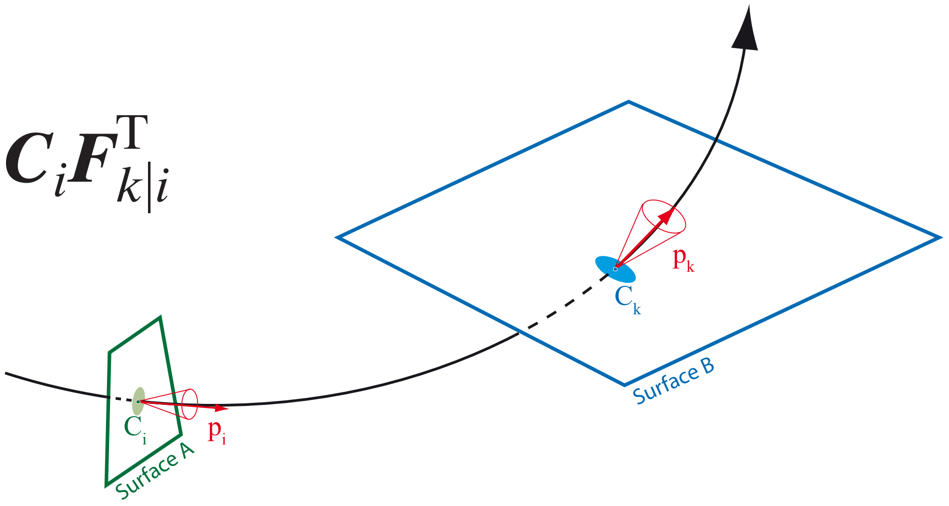
track following in mathematical terms:

$$\boldsymbol{q}_k = \boldsymbol{f}_{k|i}(\boldsymbol{q}_i)$$

convariance: $\boldsymbol{C}_k = \boldsymbol{F}_{k|i} \boldsymbol{C}_i \boldsymbol{F}_{k|i}^{\mathrm{T}}$

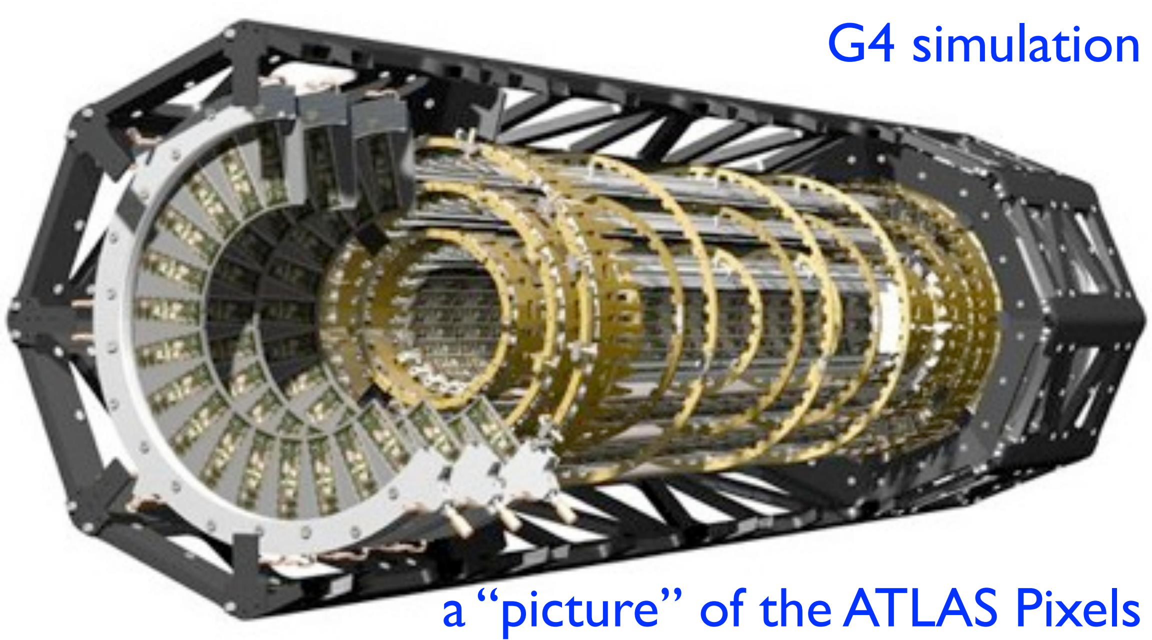
with: $\boldsymbol{f}_{k|i}$ ~ track model

$\boldsymbol{F}_{k|i} = \dfrac{\partial \boldsymbol{q}_k}{\partial \boldsymbol{q}_i}$ ~ Jacobi matrix



Detector Geometry

- interactions in detector material limiting tracking performance
  - ➡ ATLAS/CMS significantly more material in trackers than e.g. LEP experiments or CDF and D0
- LHC detectors are complex
  - ➡ experiments developed geometry models, translation into G4 simulation
  - ➡ huge number of volumes
- physics requirement to reach LHC goals (e.g. W mass)
  - ➡ control material close to beam pipe at % level


G4 simulation

a "picture" of the ATLAS Pixels

	model	placed volumes
ALICE	Root	4.3 M
ATLAS	GeoModel	4.8 M
CMS	DDD	2.7 M
LHCb	LHCb Det.Des.	18.5 M

Weighing Detectors during Construction

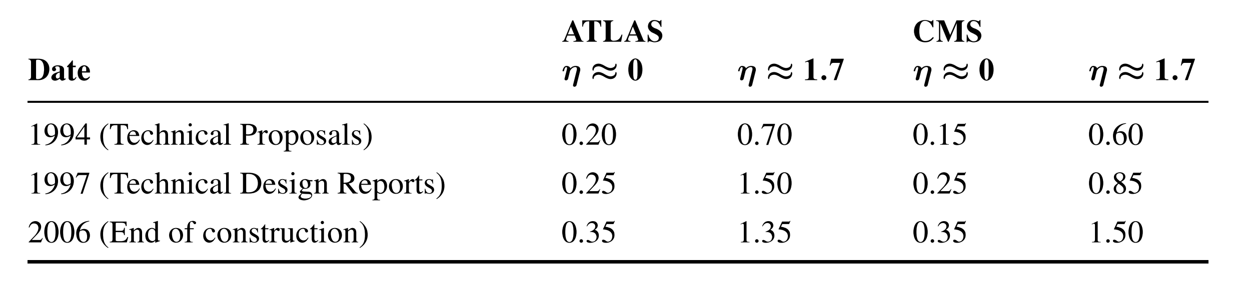
- huge effort in experiments
  - ➡ put each individual detector part on balance and compare with model
  - ➡ CMS and ATLAS measured weight of their tracker and its components
  - ➡ correct the geometry implementation in simulation and reconstruction

CMS	estimated from measurements	simulation
active Pixels	2598 g	2455 g
full detector	6350 kg	6173 kg

ATLAS	estimated from measurements	simulation
Pixel package	201 kg	197 kg
SCT detector	672 ±15 kg	672 kg
TRT detector	2961 ±14 kg	2962 kg

Preliminary

example: ATLAS TRT measured before and after insertion of the SCT



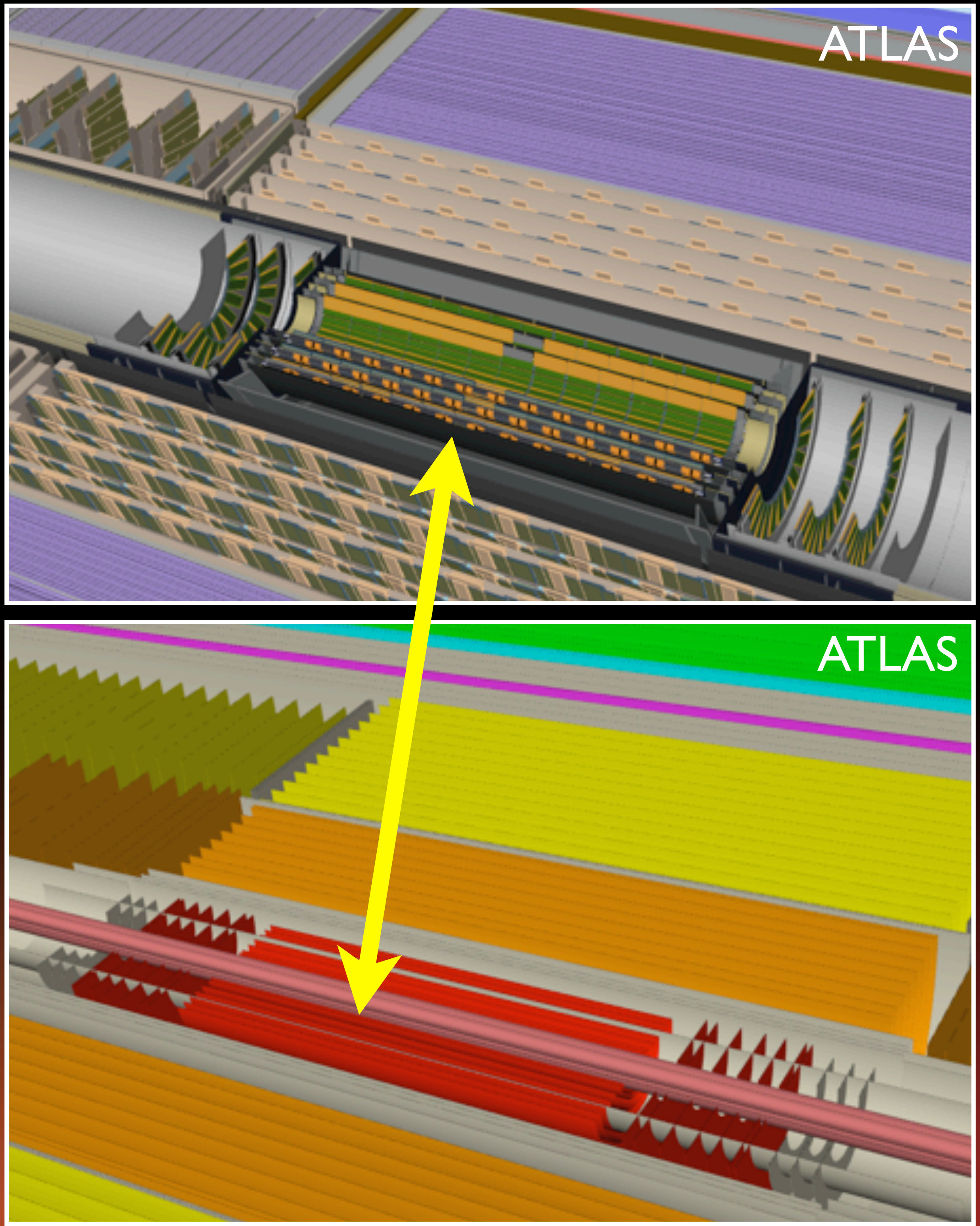
Date	ATLAS		CMS	
	$\eta \approx 0$	$\eta \approx 1.7$	$\eta \approx 0$	$\eta \approx 1.7$
1994 (Technical Proposals)	0.20	0.70	0.15	0.60
1997 (Technical Design Reports)	0.25	1.50	0.25	0.85
2006 (End of construction)	0.35	1.35	0.35	1.50

Full and Fast (Tracking) Geometries

- complex G4 geometries not optimal for reconstruction
  - ➡ simplified **tracking geometries**
  - ➡ material surfaces, field volumes
- reduced number of volumes
  - ➡ blending details of material onto simple surfaces/volumes
  - ➡ surfaces with 2D material density maps, templates per Si sensor...

	G4	tracking
ALICE	4.3 M	same *[1]
ATLAS	4.8 M	10.2K *[2]
CMS	2.7 M	3.8K *[2]
LHCb	18.5 M	30

*[1] ALICE uses full geometry (TGeo)
*[2] plus a surface per Si sensor



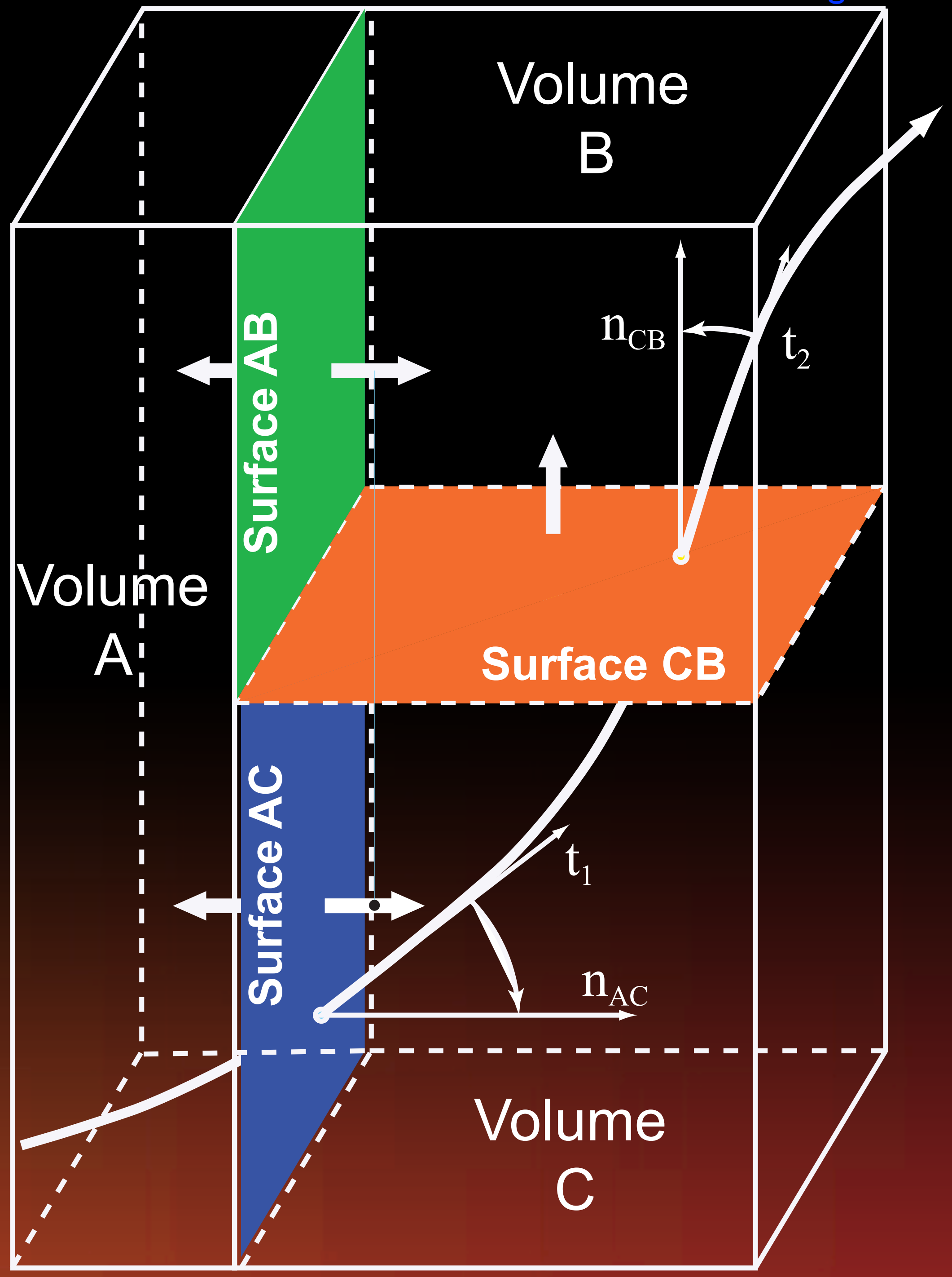
Embedded Navigation Schemes

- embedded navigation scheme in tracking geometries
  - ➡ G4 navigation uses voxelisation as generic navigation mechanism
  - ➡ **embedded navigation** for simplified models
  - ➡ used in pattern recognition, extrapolation, track fitting and fast simulation
- example: ATLAS
  - ➡ developed geometry of connected volumes
  - ➡ boundary surfaces connect neighboring volumes to predict next step

ATLAS	G4	tracking	ratio
crossed volumes in tracker	474	95	5
time in SI2K sec	19.1	2.3	8.4

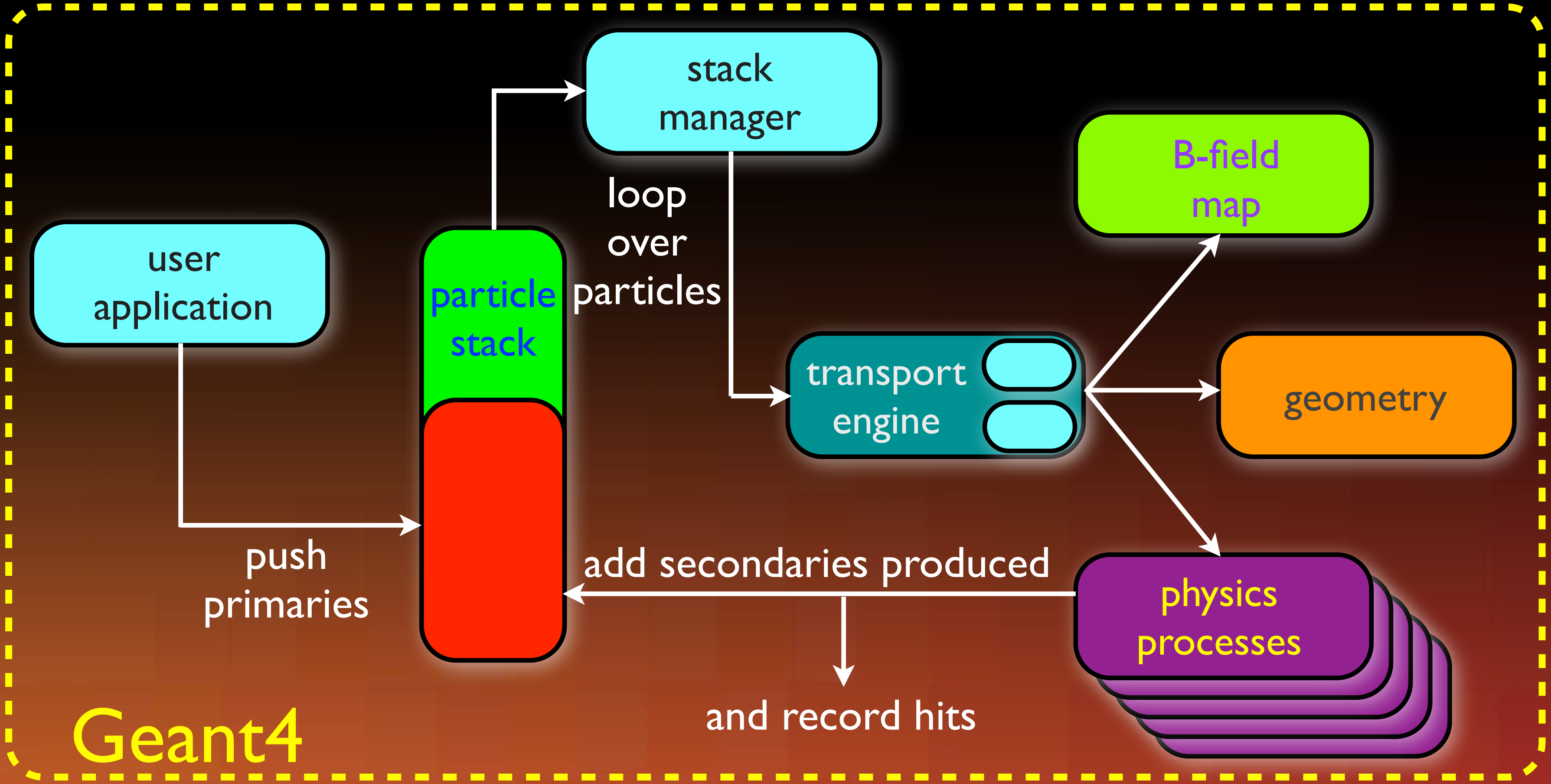
(neutral geantinos, no field lookups)

A.Salzburger



Some Remarks on Simulation: Geant4

- Geant4 is based upon
  - ➡ **stack** to keep track of all particles produced and stack manager
  - ➡ **extrapolation system** to propagate each particle:
    - transport engine with navigatoin
    - geometry model
    - B-field
  - ➡ set of **physics processes** describing interaction of particles with matter
  - ➡ a user application interface, ...



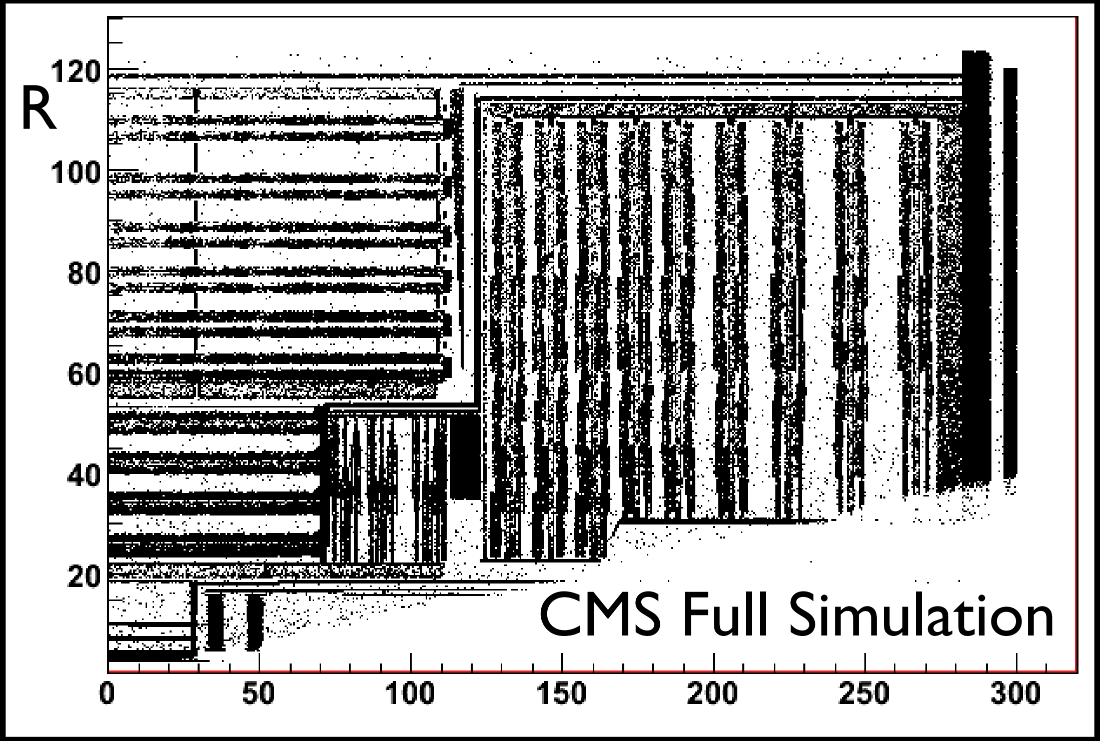
Fast Simulation

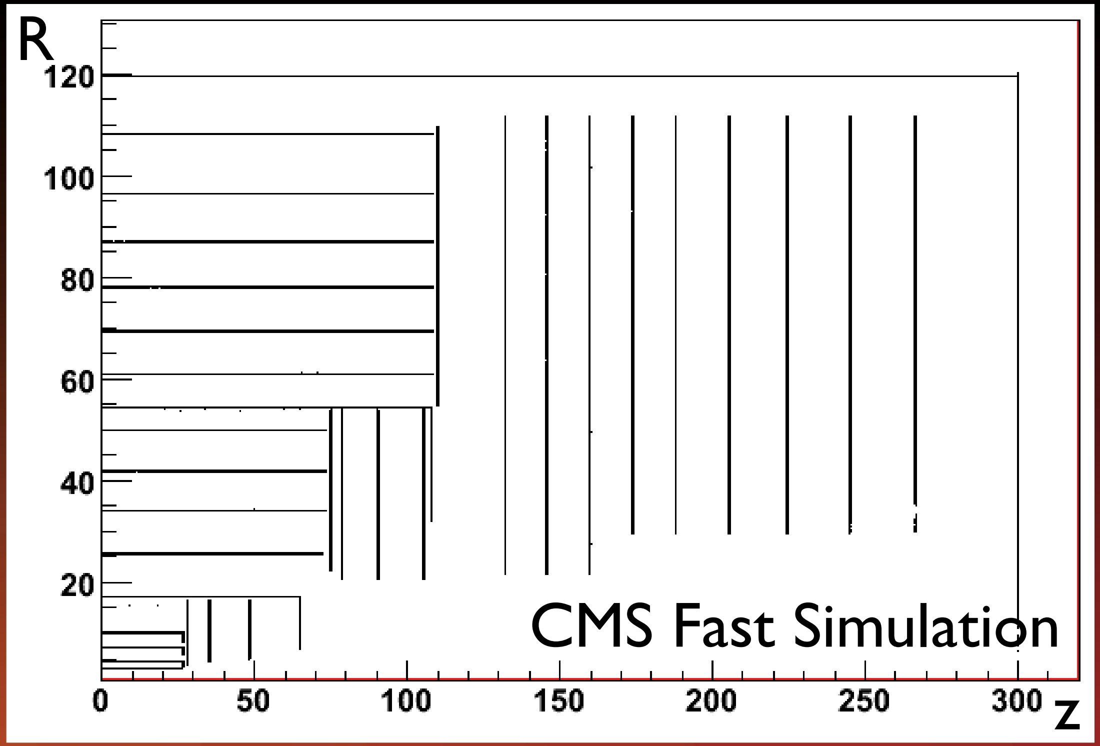
- CPU needs for full G4 exceeds computing models
    - ➡ simulation strategies of experiments mix full G4 and fast simulation

	G4	fast sim.
CMS	360	0.8
ATLAS	1990	7.4

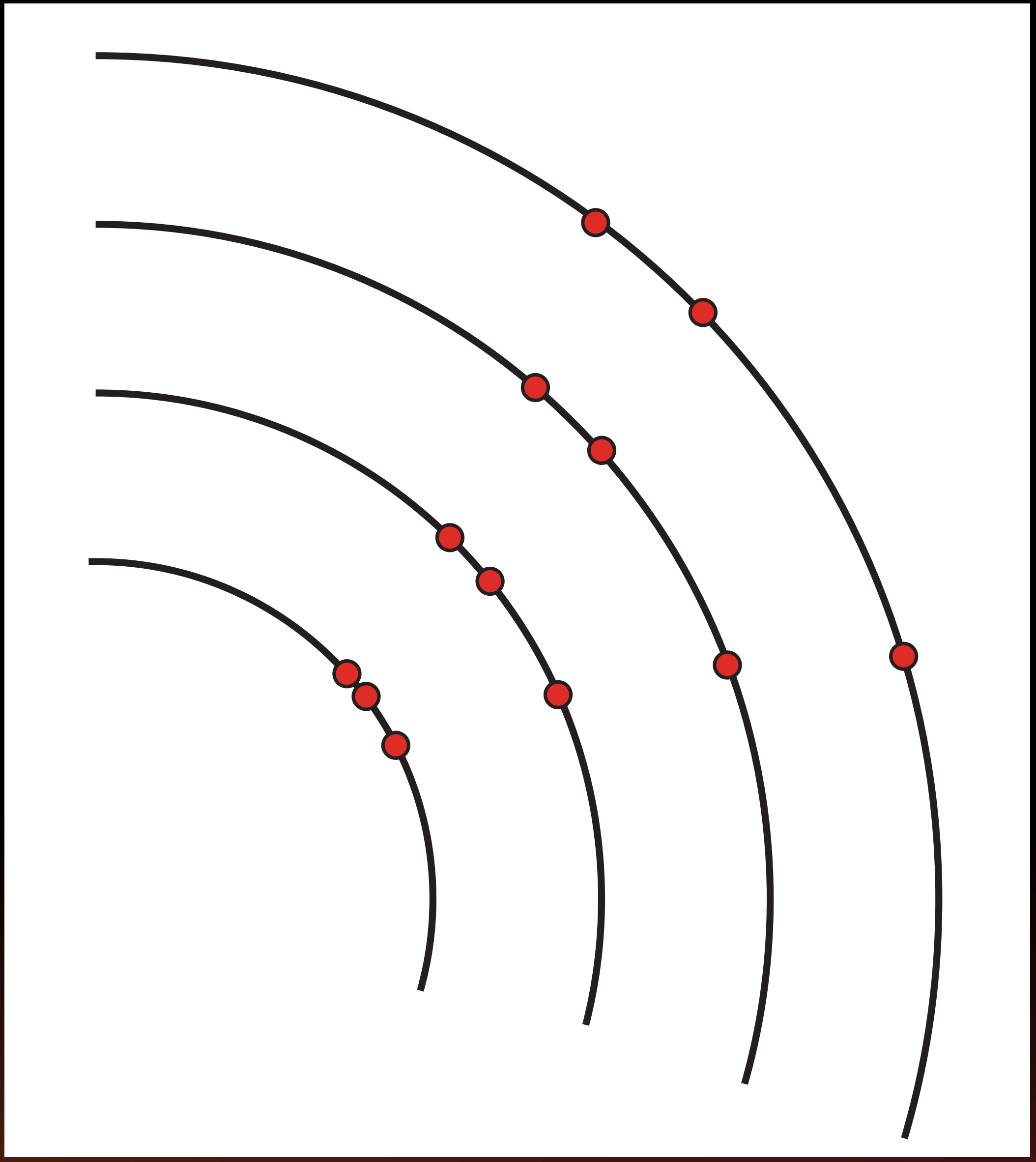
- ttbar events, in kSI2K sec
- G4 differences: calo.modeling , phys.list, η cuts, b-field

- fast simulation engines
    - ➡ fast calo. simulation (parameterization, showers libraries, ...)
    - ➡ simplified (tracking) geometries
    - ➡ simplify physics processes w.r.t. G4
    - ➡ output in same data model as full sim.
    - ➡ able to run full reconstruction (+trigger)





Back to Tracking: Track Fitting



- task of a track fit:
  - ➡ estimate the track parameters from a set of measurements
- measurement model
  - ➡ in mathematical terms:

$$\boldsymbol{m}_k = \boldsymbol{h}_k(\boldsymbol{q}_k) + \boldsymbol{\gamma}_k$$

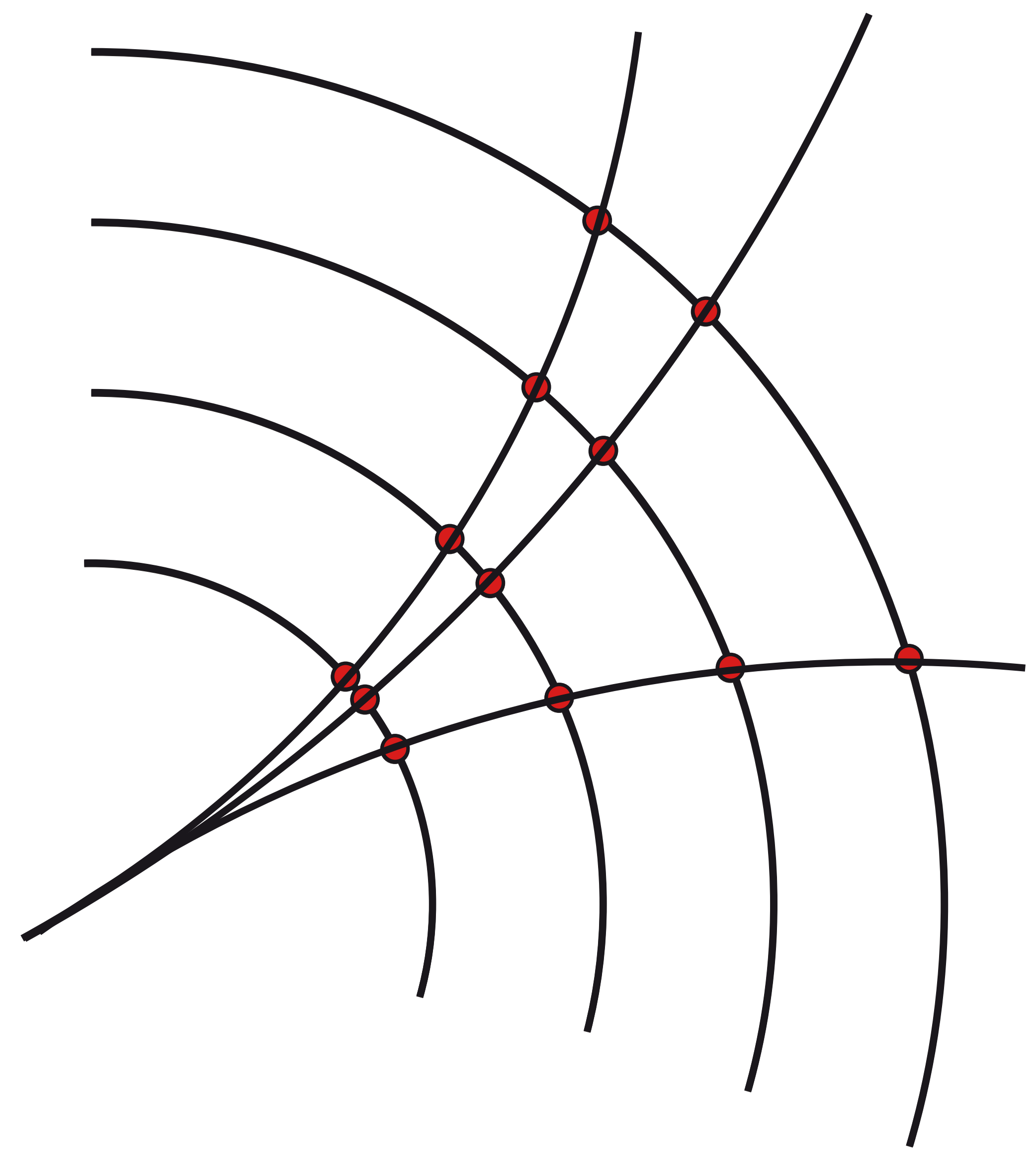
with: $\boldsymbol{h}_k$ ~ functional dependency of measurement on e.g. track angle

$\boldsymbol{\gamma}_k$ ~ error (noise term)

$\boldsymbol{H}_k = \frac{\partial \boldsymbol{m}_k}{\partial \boldsymbol{q}_k}$ ~ Jacobian, often contains only rotations and projections

  - ➡ in practice those are clusters, drift circles, ...
- examples for fitting techniques
  - ➡ Least Square track fit or Kalman Filter track fit
  - ➡ more specialized versions: Gaussian Sum Filter or Deterministic Annealing Filters

Back to Tracking: Track Fitting



- task of a track fit:
  - ➡ estimate the track parameters from a set of measurements
- measurement model
  - ➡ in mathematical terms:

$$m_k = h_k(q_k) + \gamma_k$$

with: h_k ~ functional dependency of measurement on e.g. track angle

γ_k ~ error (noise term)

$H_k = \frac{\partial m_k}{\partial q_k}$ ~ Jacobian, often contains only rotations and projections

  - ➡ in practice those are clusters, drift circles, ...
- examples for fitting techniques
  - ➡ Least Square track fit or Kalman Filter track fit
  - ➡ more specialized versions: Gaussian Sum Filter or Deterministic Annealing Filters

Classical Least Square Track Fit

Carl Friedrich Gauss is credited with developing the fundamentals of the basis for least-squares analysis in 1795 at the age of eighteen. **Legendre** was the first to publish the method, however.

- construct and minimize the χ^2 function:

$$\chi^2 = \sum_k \Delta m_k^T G_K^{-1} \Delta m_k \quad \text{with:} \quad \Delta m_k = m_k - d_k(p)$$

d_k contains measurement model and propagation of the parameters p :

$$\boldsymbol{d}_k = \boldsymbol{h}_k \circ \boldsymbol{f}_{k|k-1} \circ \cdots \circ \boldsymbol{f}_{2|1} \circ \boldsymbol{f}_{1|0}$$

G_k is the covariance matrix of m_k . Linearize the problem:

$$d_k(p_0 + \delta p) \cong d_k(p_0) + D_k \cdot \delta p \ + \text{higher terms}$$

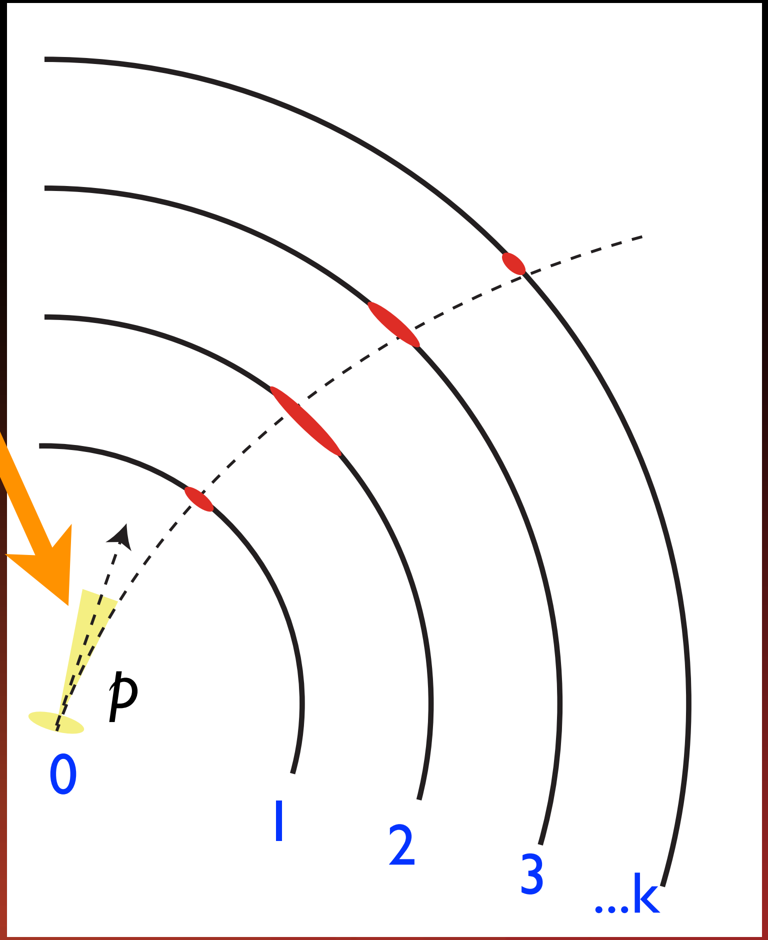
with Jacobian:

$$\boldsymbol{D}_k = \boldsymbol{H}_k \boldsymbol{F}_{k|k-1} \cdots \boldsymbol{F}_{2|1} \boldsymbol{F}_{1|0}$$

minimizing the linearized χ^2 yields:

$$\frac{\partial \chi^2}{\partial p} = 0 \ \Rightarrow \ \delta p = \left(\sum_k D_k^T G_k^{-1} D_k \right)^{-1} \sum_k D_k^T G_k^{-1} \left(m_k - d_k(p_0) \right)$$

and covariance of δp is: $C = \left(\sum_k D_k^T G_k^{-1} D_k \right)^{-1}$

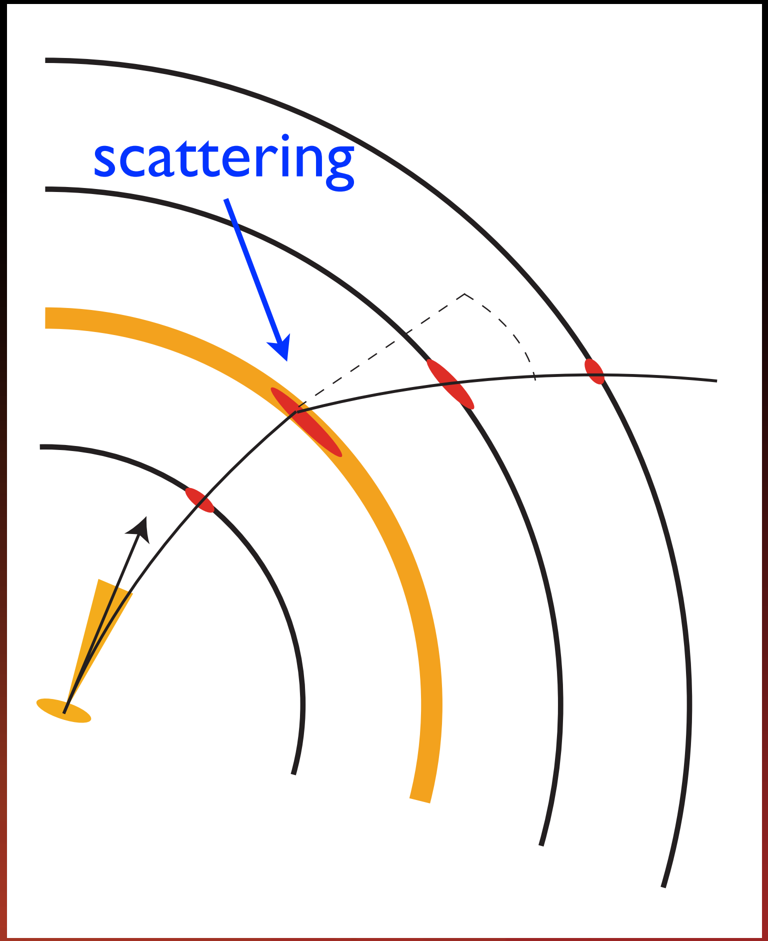


Classical Least Square Track Fit

- material effects
  - ➡ can be absorbed in track model $\mathbf{f}_{k|i}$, provided effects are small
  - ➡ for substantial multiple scatting, allows for **scattering angles** in the fit
- scattering angles
  - ➡ on each material surface, add 2 angles $\boldsymbol{\delta\theta_i}$ as fee parameters to the fit
  - ➡ expected mean of those angles is 0 (!), their covariance $\mathbf{Q_i}$ is given by multiple scattering in x/X_0
- changes to χ^2 formula on previous slide

$$\chi^2 = \sum_k \Delta m_k^T G_K^{-1} \Delta m_k + \sum_i \delta\theta_i^T Q_i^{-1} \delta\theta_i$$

with: $\Delta m_k = m_k - d_k\left(p, \delta\theta_i\right)$

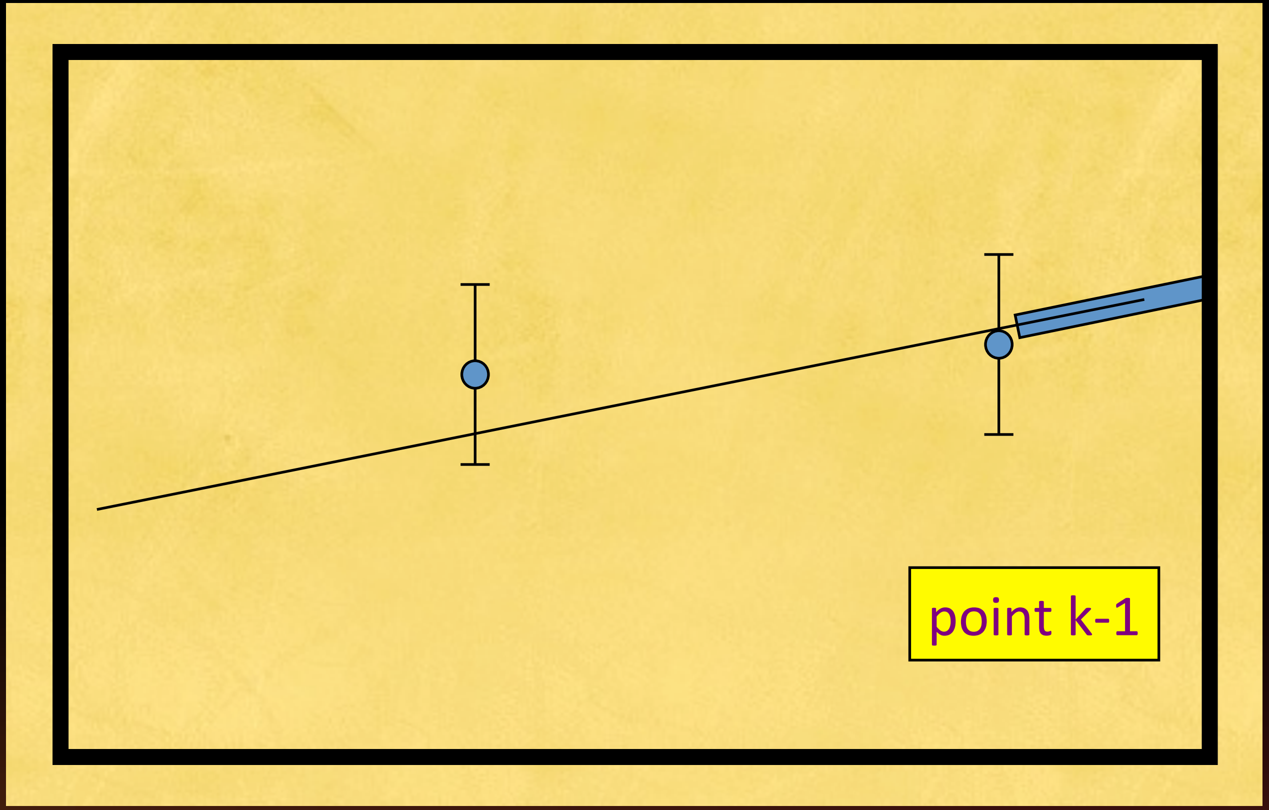


  - ➡ computationally expensive: *need to invert a (5+2*n) matrix*
  - ➡ advantage is that the fitted track precisely follows the particle trajectory: (e.g. for ATLAS muon reconstruction)

The Kalman Filter Track Fit

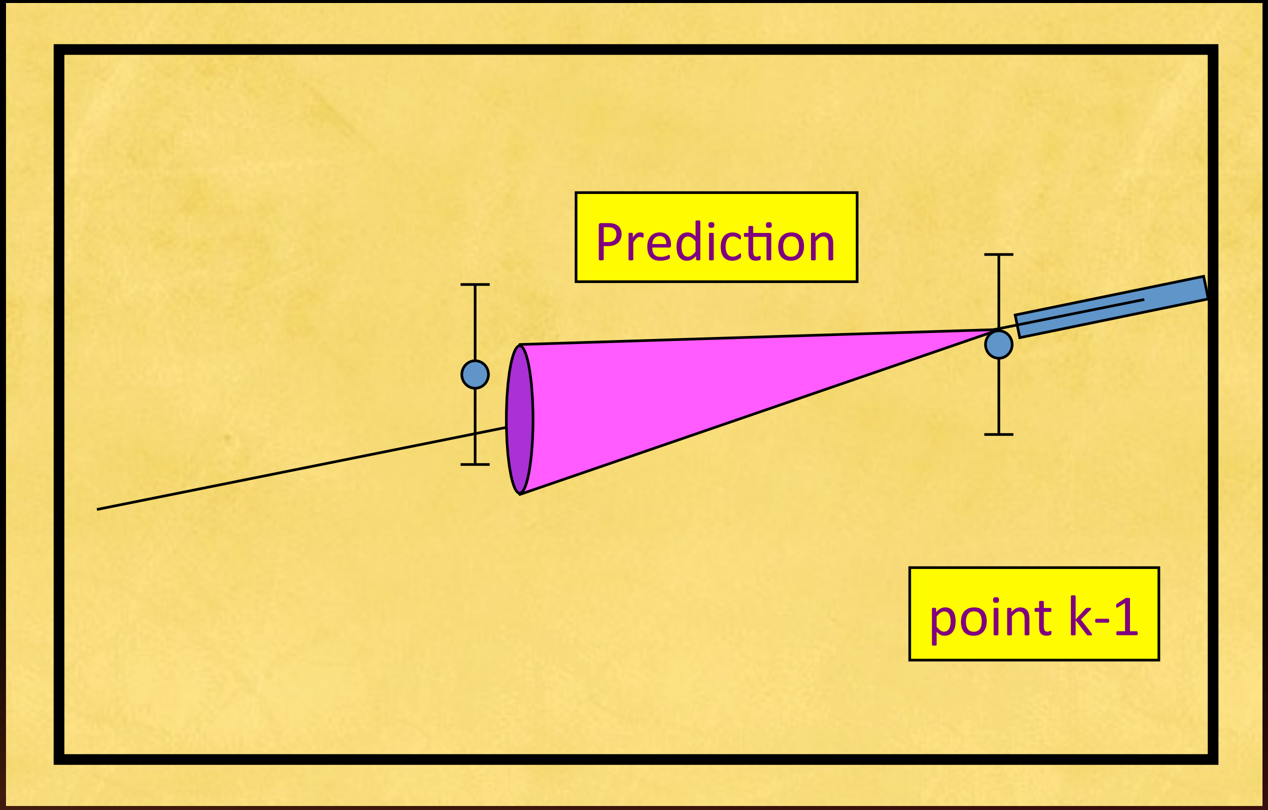
- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ mathematically equivalent

- how does the filter work ?
  1. trajectory parameters at point **k-1**



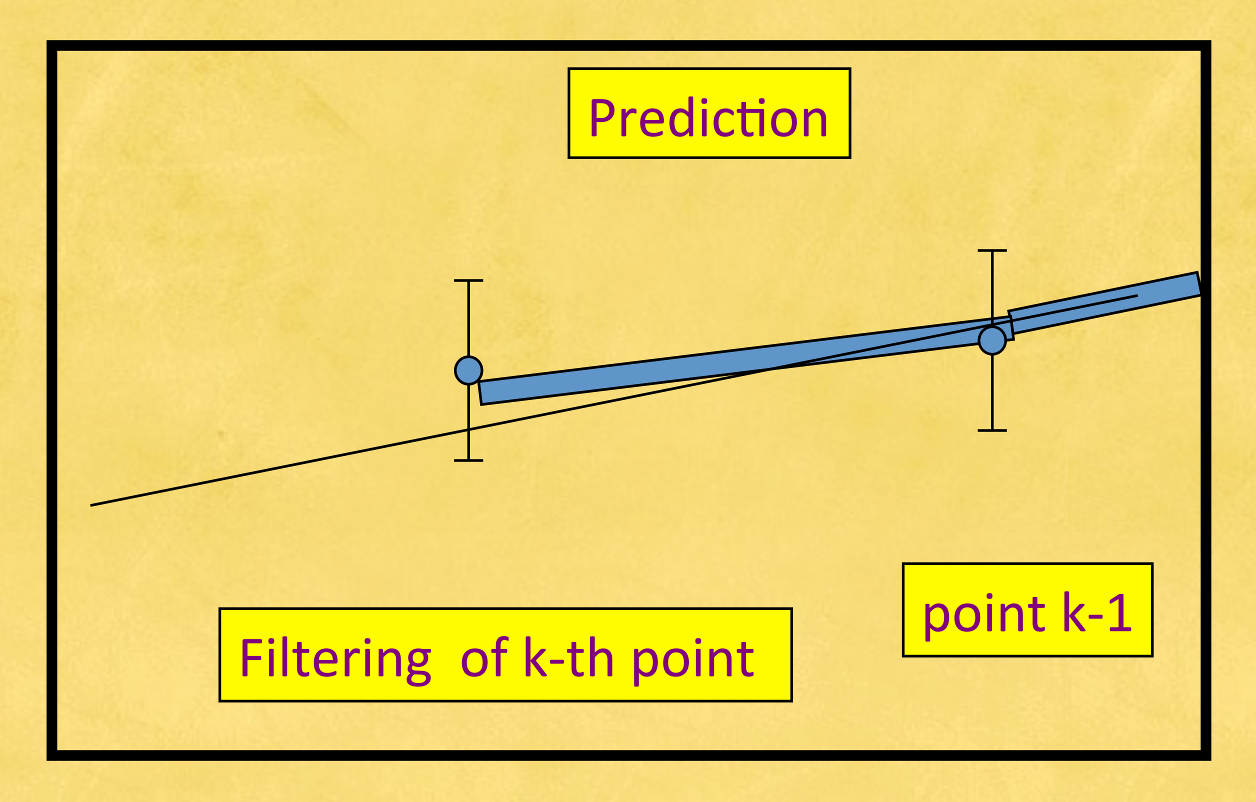
The Kalman Filter Track Fit

- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ mathematically equivalent
- how does the filter work ?
  1. trajectory parameters at point **k-1**
  2. propagate to point **k** to get predicted parameters
     (let's ignore material effects)



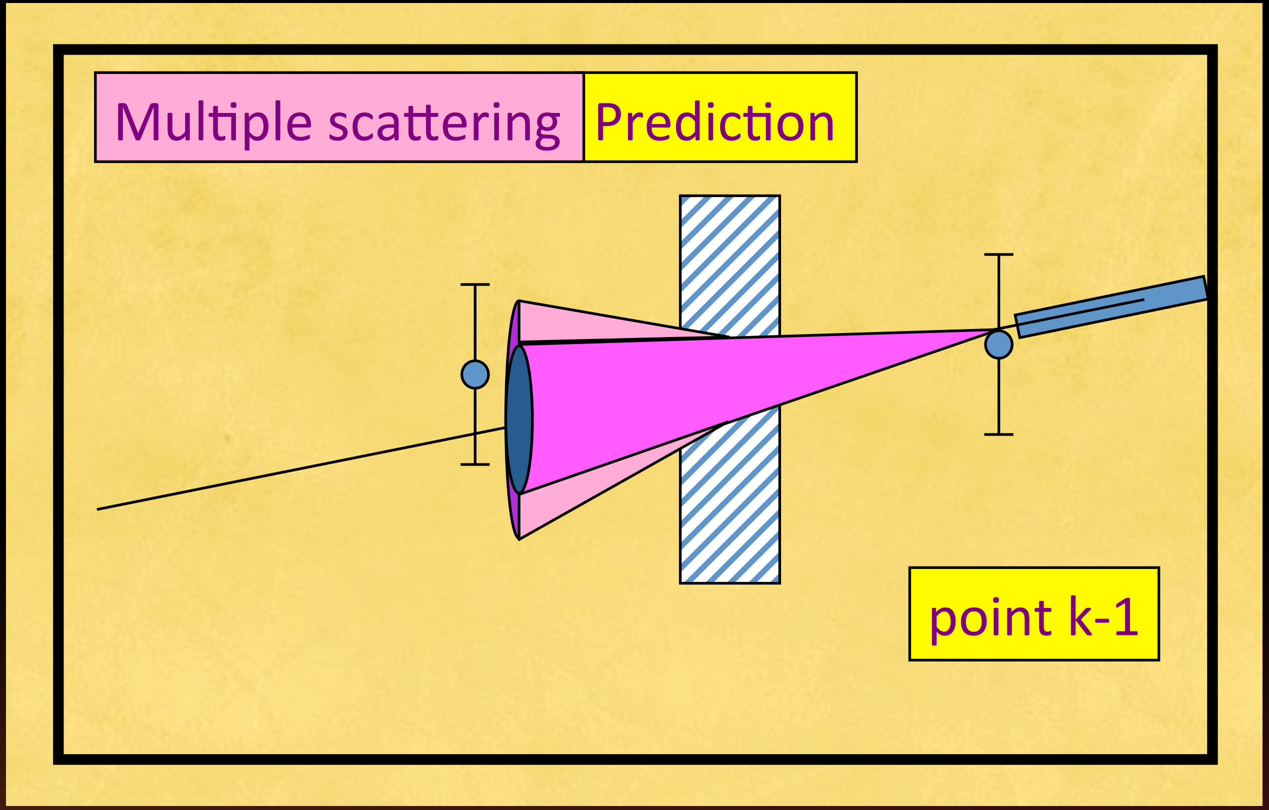
The Kalman Filter Track Fit

- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ mathematically equivalent
- how does the filter work ?
  1. trajectory parameters at point **k-1**
  2. propagate to point **k** to get predicted parameters
     (let's ignore material effects)
  3. update predicted parameters with measurement **k**
     (simple weighted mean or gain matrix update)
  4. and start over with 1.



The Kalman Filter Track Fit

- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ mathematically equivalent
- how does the filter work ?
  1. trajectory parameters at point **k-1**
  2. propagate to point **k** to get predicted parameters
     (let's ignore material effects)
  3. update predicted parameters with measurement **k**
     (simple weighted mean or gain matrix update)
  4. and start over with 1.

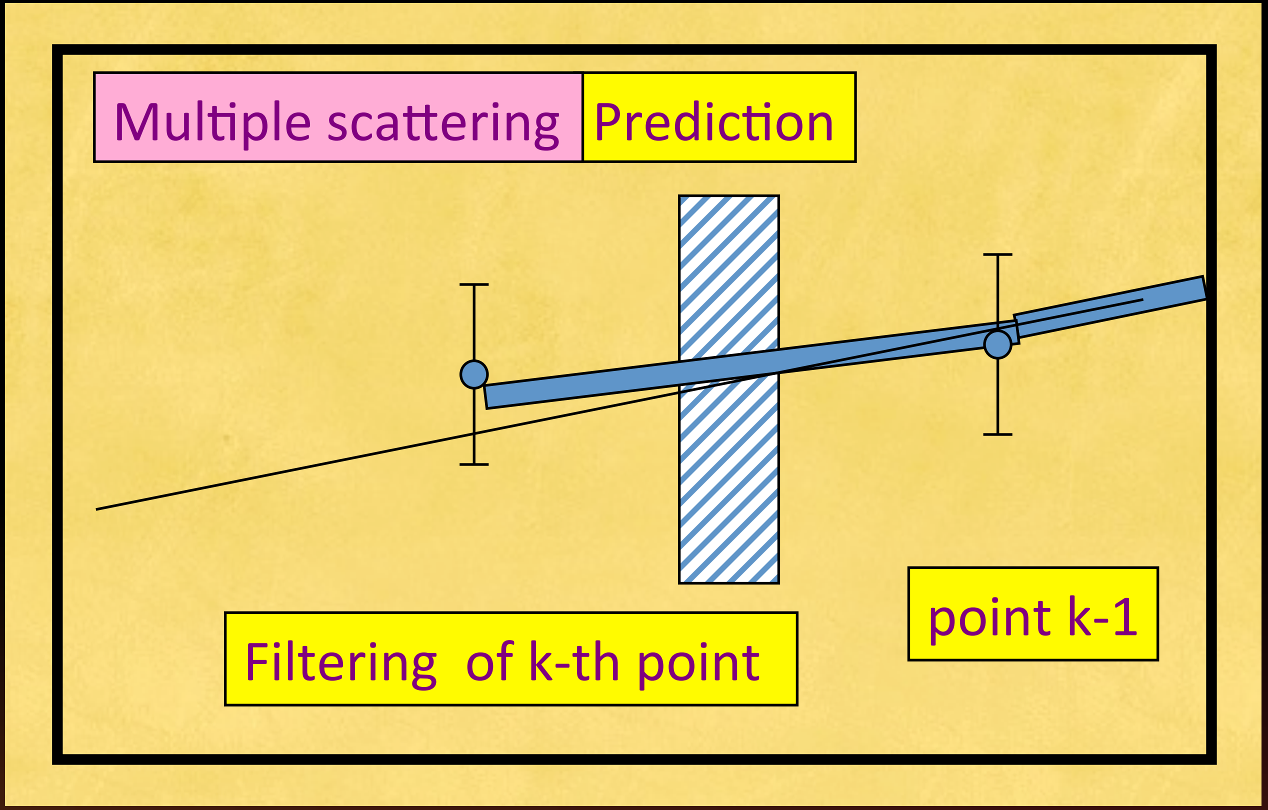


- material effects (multiple scattering and energy loss)
  - ➡ incorporated in the propagated parameters (prediction)

The Kalman Filter Track Fit

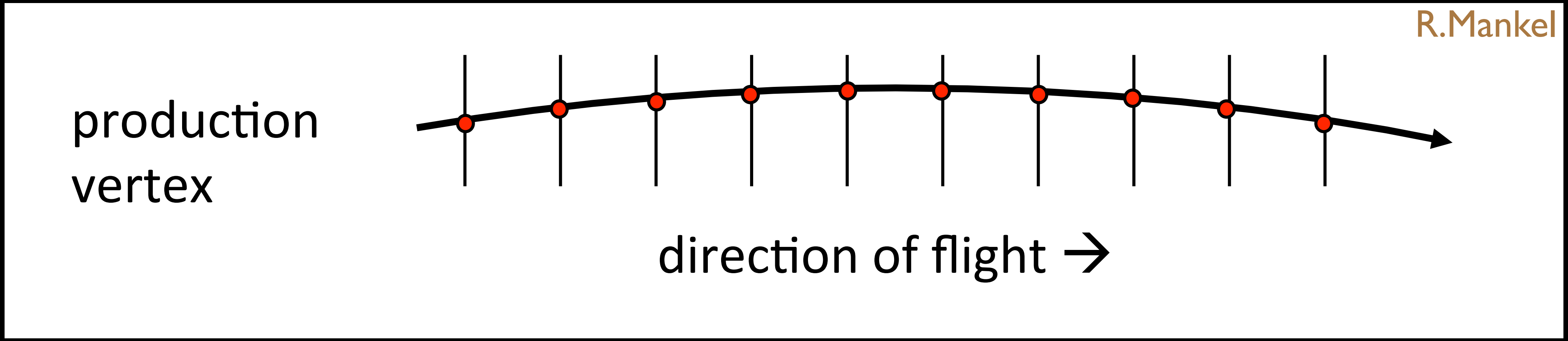
- a Kalman Filter is a progressive way of performing a least square fit
  - ➡ mathematically equivalent

- how does the filter work ?
  1. trajectory parameters at point **k-1**
  2. propagate to point **k** to get predicted parameters
     (let's ignore material effects)
  3. update predicted parameters with measurement **k**
     (simple weighted mean or gain matrix update)
  4. and start over with 1.

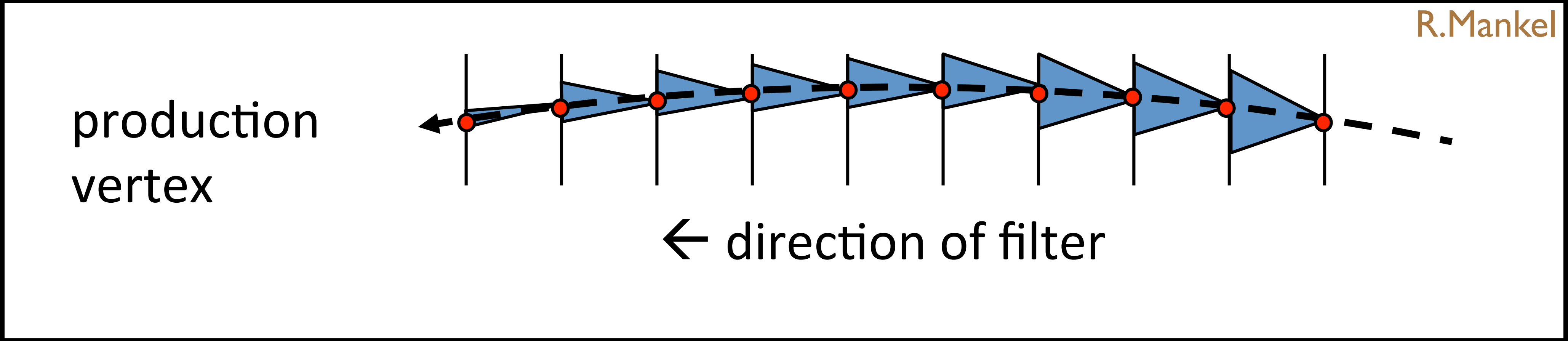


- material effects (multiple scattering and energy loss)
  - ➡ incorporated in the propagated parameters (prediction)
  - ➡ and therefore enters into the updated parameters at point **k**

The Kalman Filter Track Fit

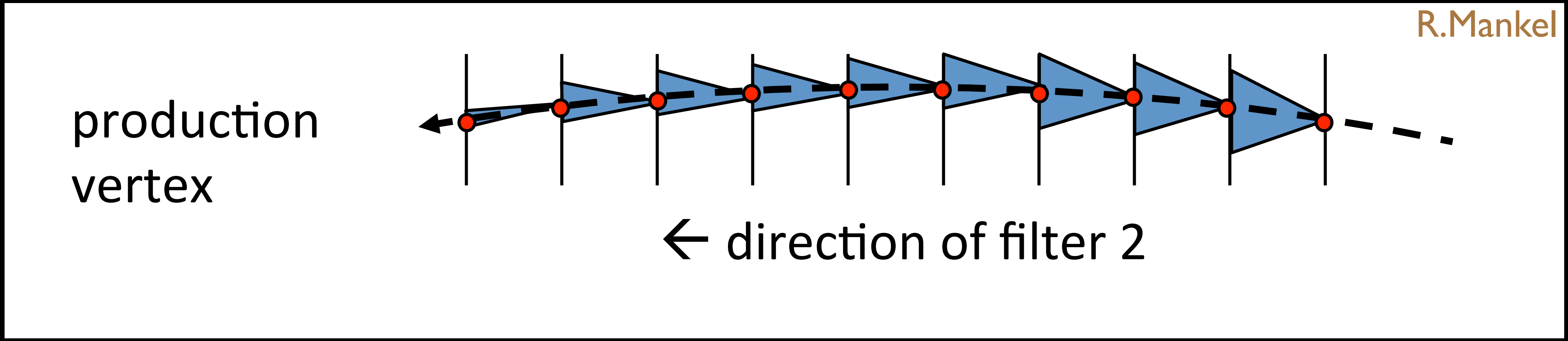


The Kalman Filter Track Fit



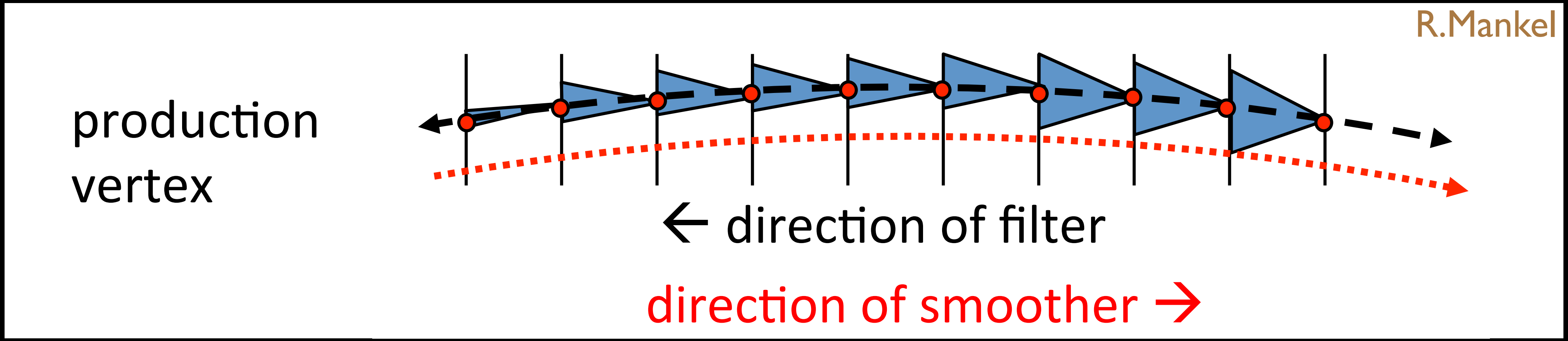
- in its minimal form
  - ➡ Kalman filter track fit proceeds in the direction opposite to the particle's flight (**backward filter**)
  - ➡ parameter **estimate near production vertex** contains information of all hits and therefore is most prices
  - ➡ fastest version of a Kalman filter track fit

The Kalman Filter Track Fit



- combining **forward** with **backward filter**
  - ➡ precise parameter estimates at end of track (e.g. near calorimeter entry point) and **near production vertex**
  - ➡ forward filter parameter can be used to start backward filter

The Kalman Filter Track Fit



- **Kalman smoother** can be run to obtain precise parameters everywhere along the trajectory
  - ➡ run after backward filter, gives best estimates along the track
  - ➡ computationally expensive, need to invert matrix of rank 5 for each point

The Kalman Filter Track Fit

- in mathematical terms:

1. propagate p_{k-1} and its covariance C_{k-1} :

$$\boldsymbol{q}_{k|k-1} = \boldsymbol{f}_{k|k-1}(\boldsymbol{q}_{k-1|k-1})$$

$$\boldsymbol{C}_{k|k-1} = \boldsymbol{F}_{k|k-1}\boldsymbol{C}_{k-1|k-1}\boldsymbol{F}^{\mathrm{T}}_{k|k-1} + \boldsymbol{Q}_k$$

with Q_k ~ noise term (M.S.)

2. update prediction to get $q_{k|k}$ and $C_{k|k}$:

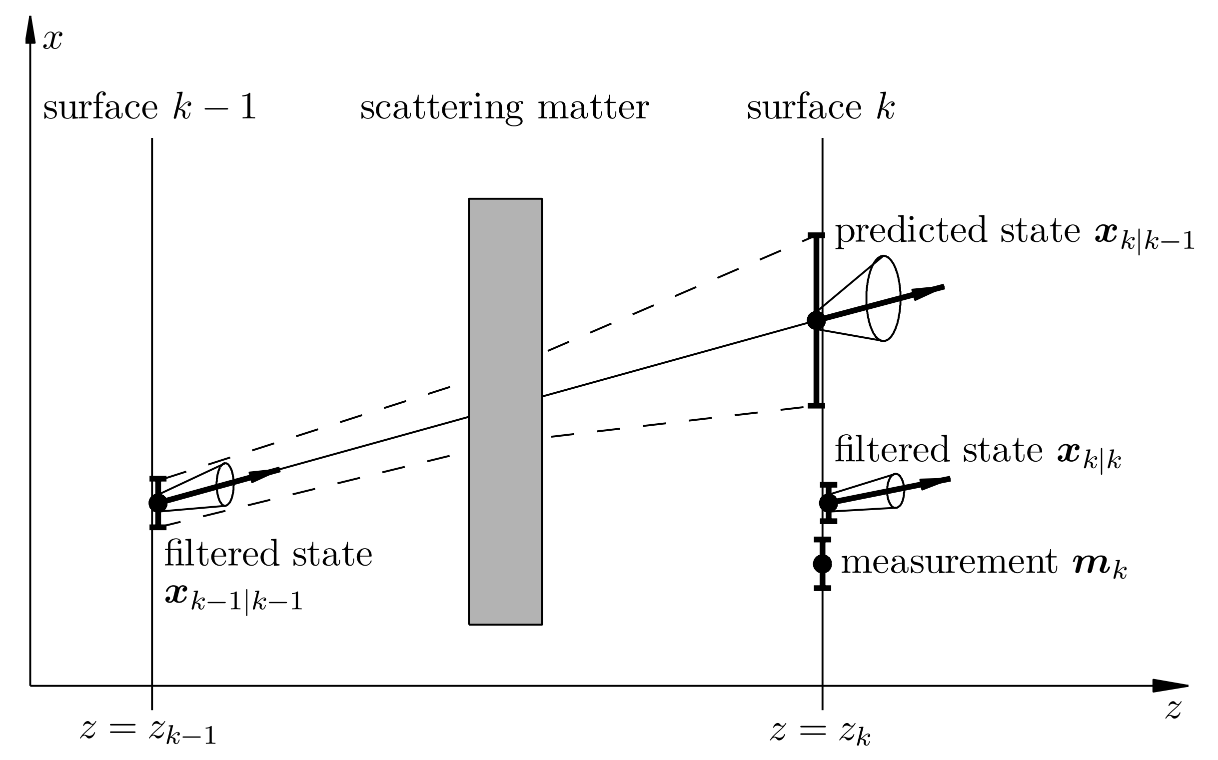
$$\boldsymbol{q}_{k|k} = \boldsymbol{q}_{k|k-1} + \boldsymbol{K}_k[\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{q}_{k|k-1})]$$

$$\boldsymbol{C}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_{k|k-1}$$

with K_k ~ gain matrix :

$$\boldsymbol{K}_k = \boldsymbol{C}_{k|k-1}\boldsymbol{H}^{\mathrm{T}}_k(\boldsymbol{G}_k + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}^{\mathrm{T}}_k)^{-1}$$

- ➡ alternative to gain matrix approach is a weighted mean to obtian $p_{k|k}$
  - but requires to invert 5x5 matrix instead of a matrix of *rank(G_k)*



- Kalman Smoother:
  - ➡ provides full information along track

proceeds from layer *k+1* to layer *k* :

$$\boldsymbol{q}_{k|n} = \boldsymbol{q}_{k|k} + \boldsymbol{A}_k(\boldsymbol{q}_{k+1|n} - \boldsymbol{q}_{k+1|k})$$

$$\boldsymbol{C}_{k|n} = \boldsymbol{C}_{k|k} - \boldsymbol{A}_k(\boldsymbol{C}_{k+1|k} - \boldsymbol{C}_{k+1|n})\boldsymbol{A}^{\mathrm{T}}_k$$

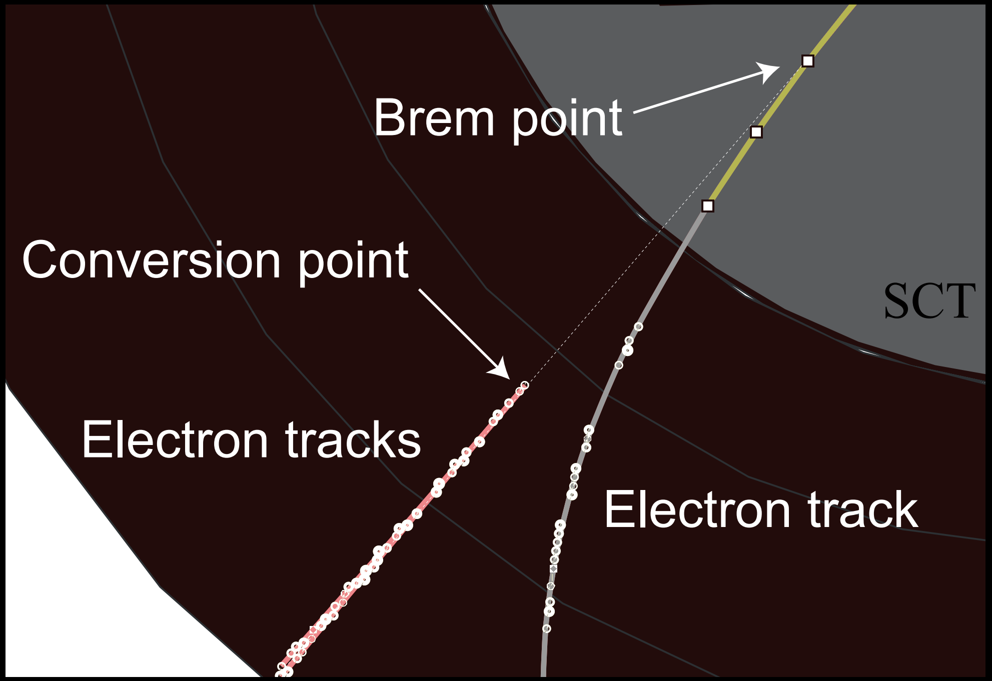
with A_k ~ smoother gain matrix :

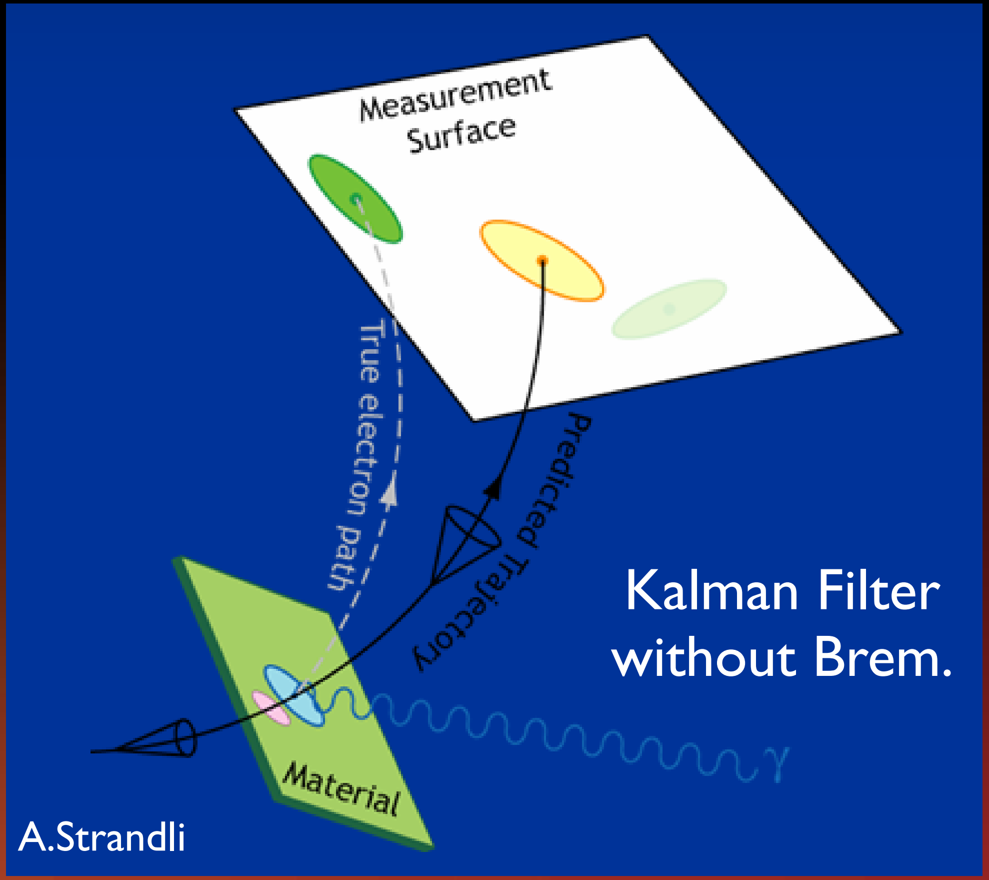
$$\boldsymbol{A}_k = \boldsymbol{C}_{k|k}\boldsymbol{F}^{\mathrm{T}}_{k+1|k}(\boldsymbol{C}_{k+1|k})^{-1}$$

- ➡ equivalent: combine forw./back. filter

Brem. Fitting for Electrons

- material in tracker
  - ➡ e-bremsstrahlung and γ-conversions
- electron efficiency limited
  - ➡ momentum loss due to bremsstrahlung leads to large changes in track curvature
  - ➡ fit is biased towards small momenta or fails completely
- techniques to allow for bremsstrahlung in track fitting
  - ➡ brem. point in Least Square track fit
  - ➡ Kalman Filter with dynamic noise adjustment
  - ➡ Gaussian Sum Filter



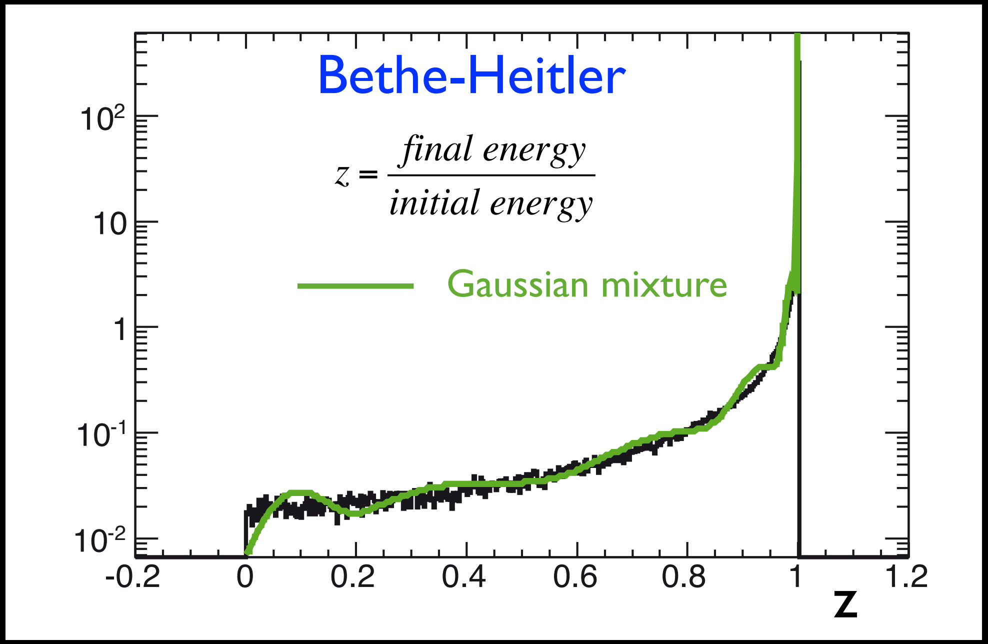


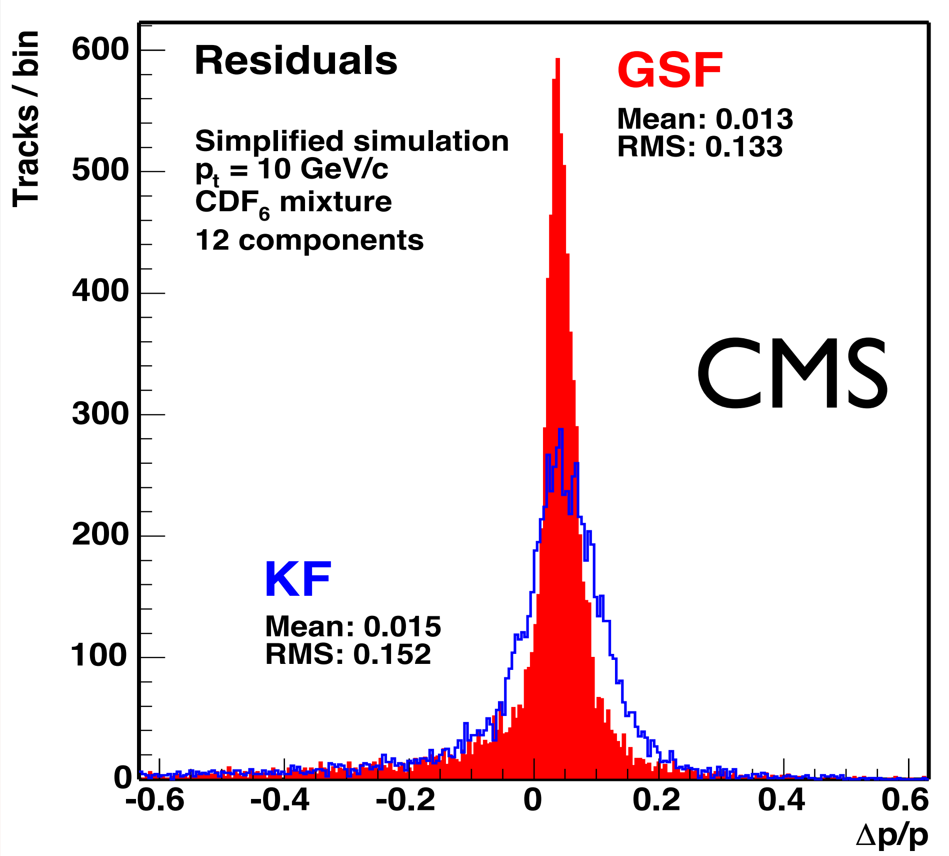
Kalman Filter without Brem.

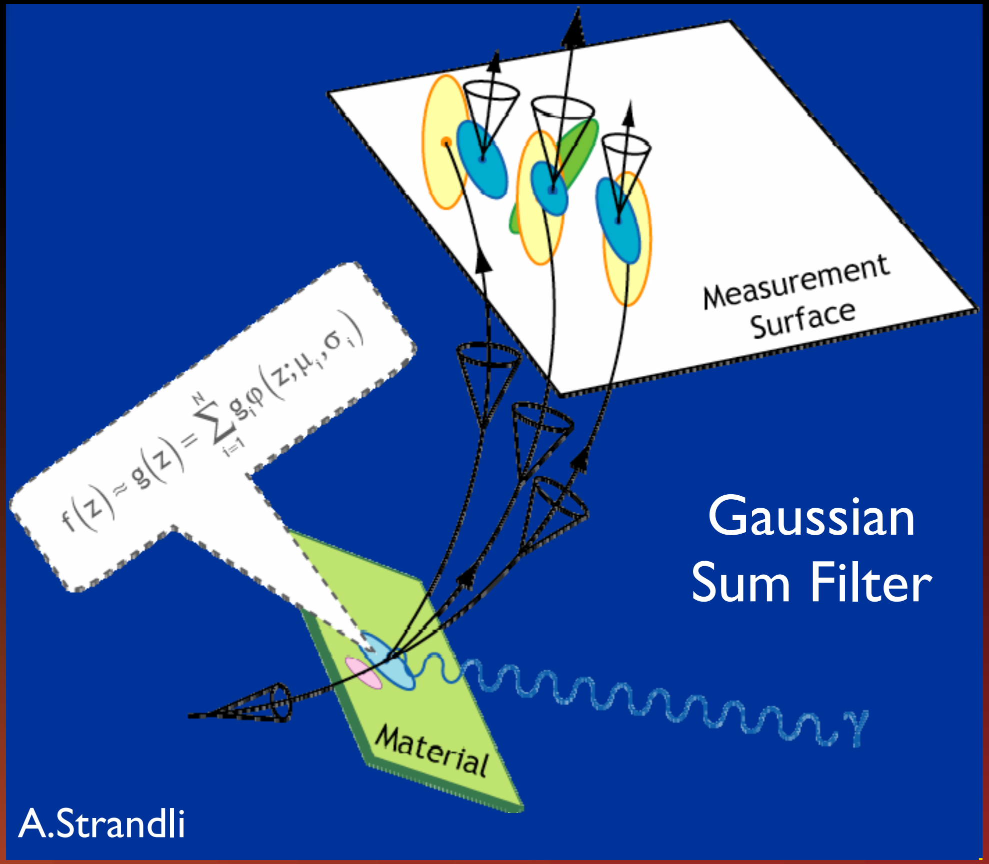
A.Strandli

Gaussian Sum Filter

- approximate Bethe-Heitler distribution as Gaussian mixture
- state vector after material correction becomes sum of Gaussian components
- GSF resembles set of parallel Kalman Filters for N components
- computationally expensive !
- default electron fitter in CMS and ATLAS



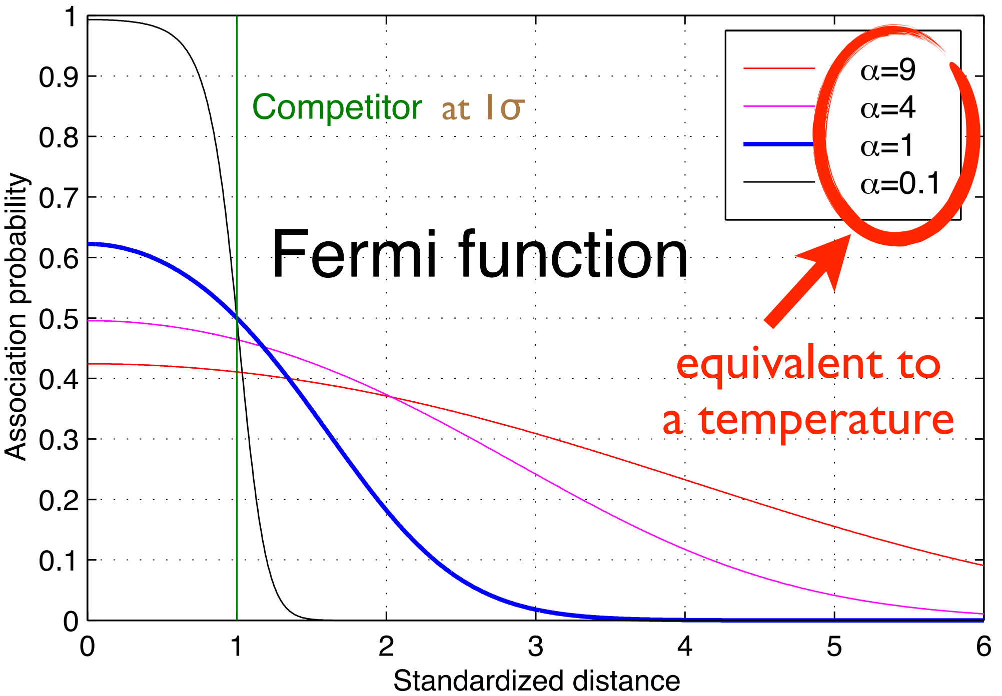




A.Strandli

Deterministic Annealing Filters

- robust technique
  - ➡ developed for fitting with high occupancies
    - e.g. ATLAS TRT with high event pileup
    - reconstruction of 3-prong τ decays
  - ➡ can deal with several close by hits on a layer



Fermi function

equivalent to a temperature

- adaptive fit
  - ➡ multiply weight of each hit in layer with assignment probability:

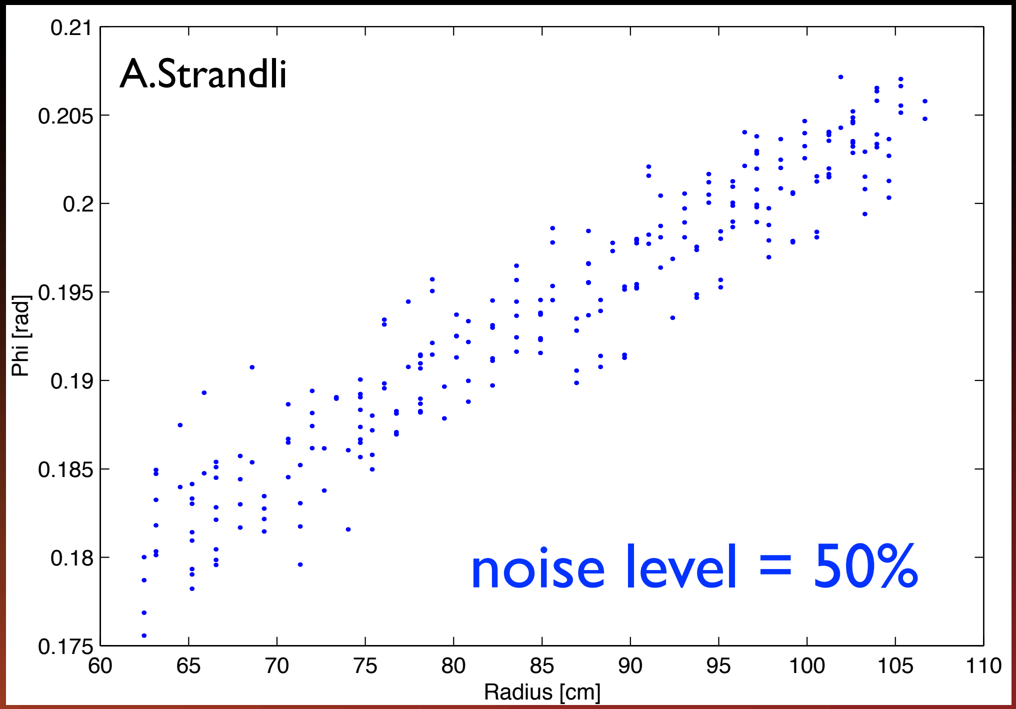
$$p_{ik} = \frac{\exp\left(-\hat{d}^2_{ik}/T\right)}{\sum_{j=1}^{n_k} \exp\left(-\hat{d}^2_{jk}/T\right)}$$

Boltzman factor

with: $\hat{d}_{ik} = d_{ik}/\sigma_k$

normalized distance

  - ➡ process decreasing temperature T is called annealing (iterative)
    - start at high T ~ all hits contribute same
    - at low T ~ close by hits remain
  - ➡ can be written as a Multi Track Filter



noise level = 50%

Deterministic Annealing Filters

- robust technique
  - ➡ developed for fitting with high occupancies
    - e.g. ATLAS TRT with high event pileup
    - reconstruction of 3-prong τ decays
  - ➡ can deal with several close by hits on a layer

- adaptive fit
  - ➡ multiply weight of each hit in layer with assignment probability:

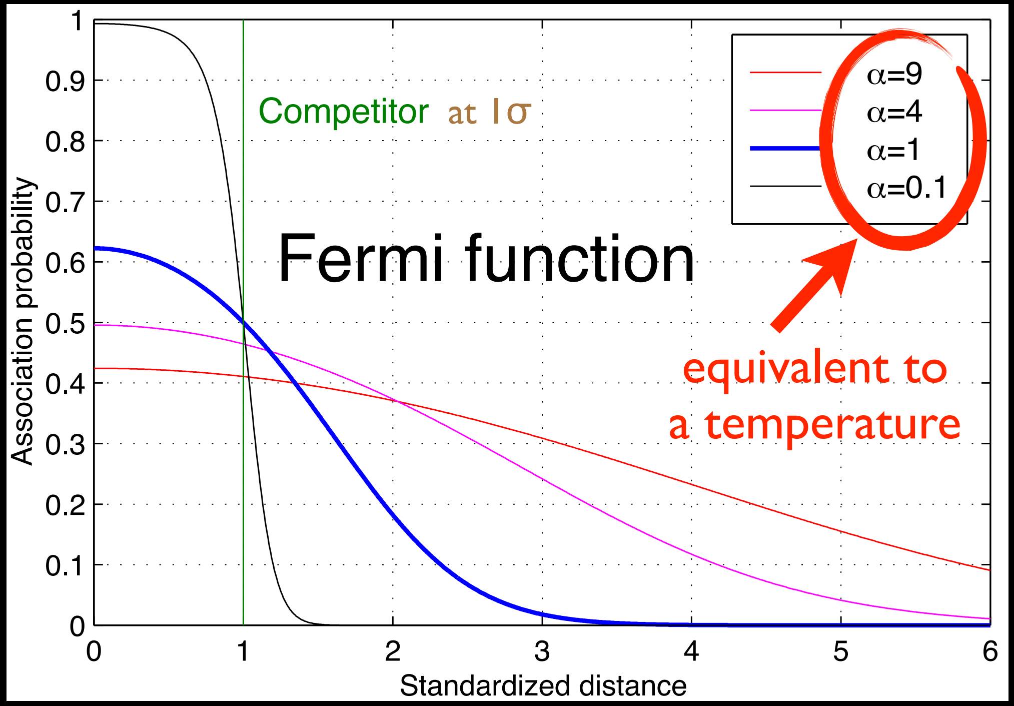
$$p_{ik} = \frac{\exp\left(-\hat{d}_{ik}^2/T\right)}{\sum_{j=1}^{n_k} \exp\left(-\hat{d}_{jk}^2/T\right)}$$

Boltzman factor

with: $\hat{d}_{ik} = d_{ik}/\sigma_k$

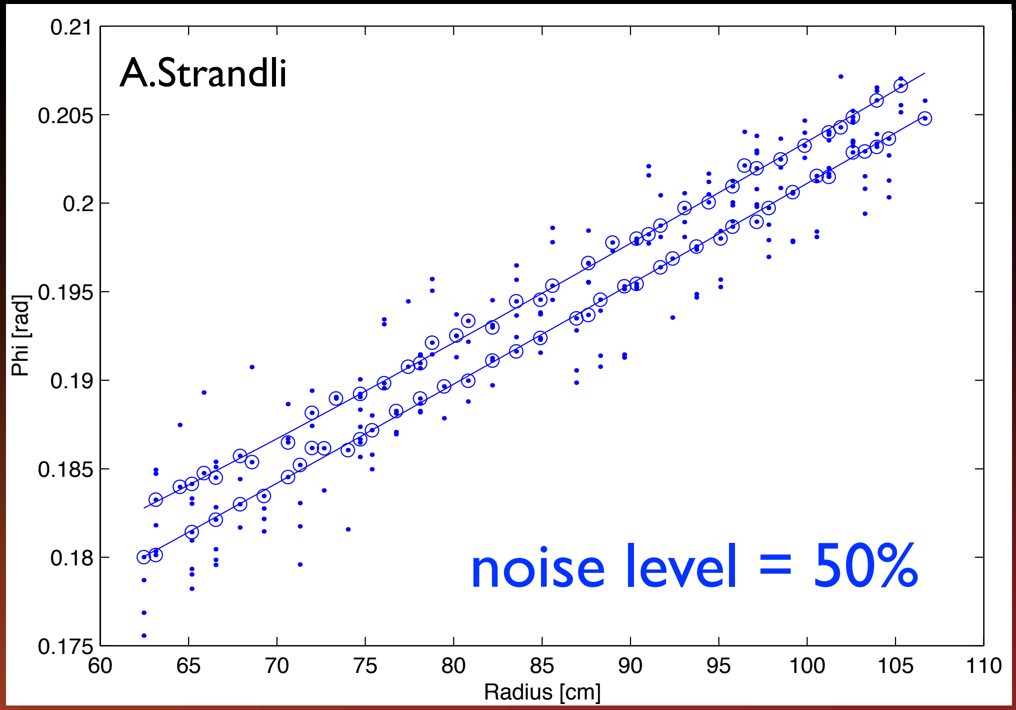
normalized distance

  - ➡ process decreasing temperature T is called annealing (iterative)
    - start at high T ~ all hits contribute same
    - at low T ~ close by hits remain
  - ➡ can be written as a Multi Track Filter



Fermi function

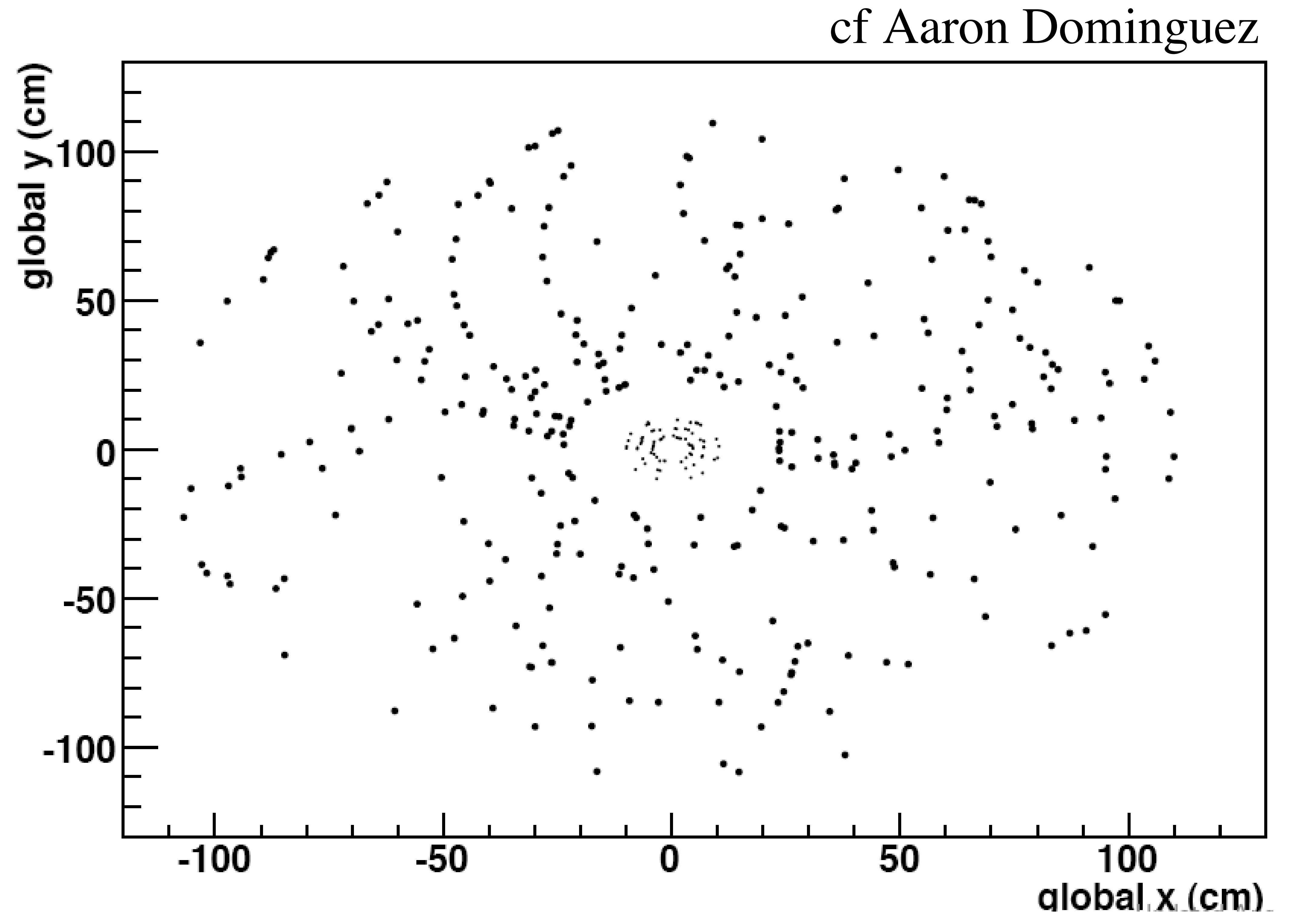
equivalent to a temperature



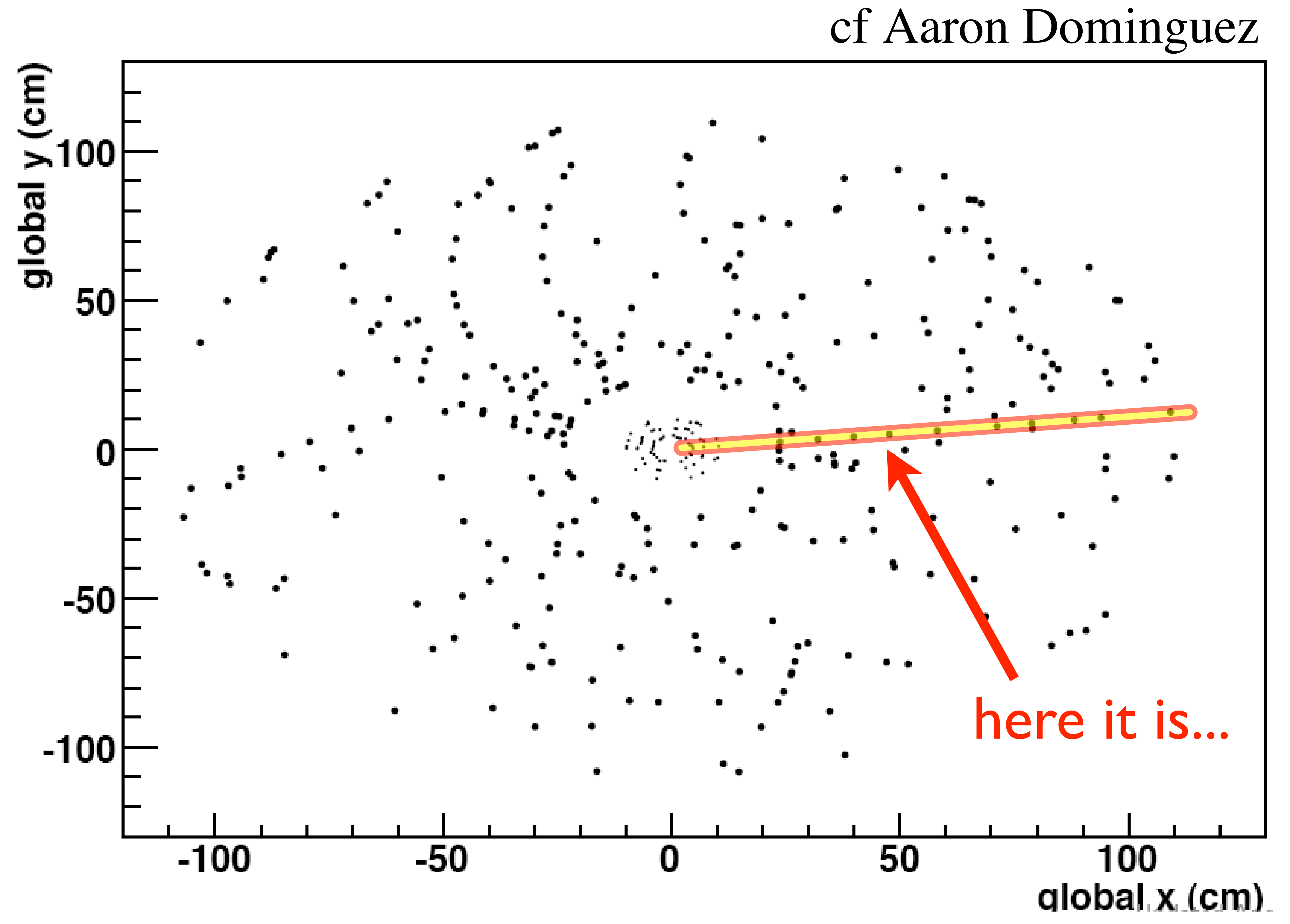
A.Strandli

noise level = 50%

Track Finding: Can you find the 50 GeV track?

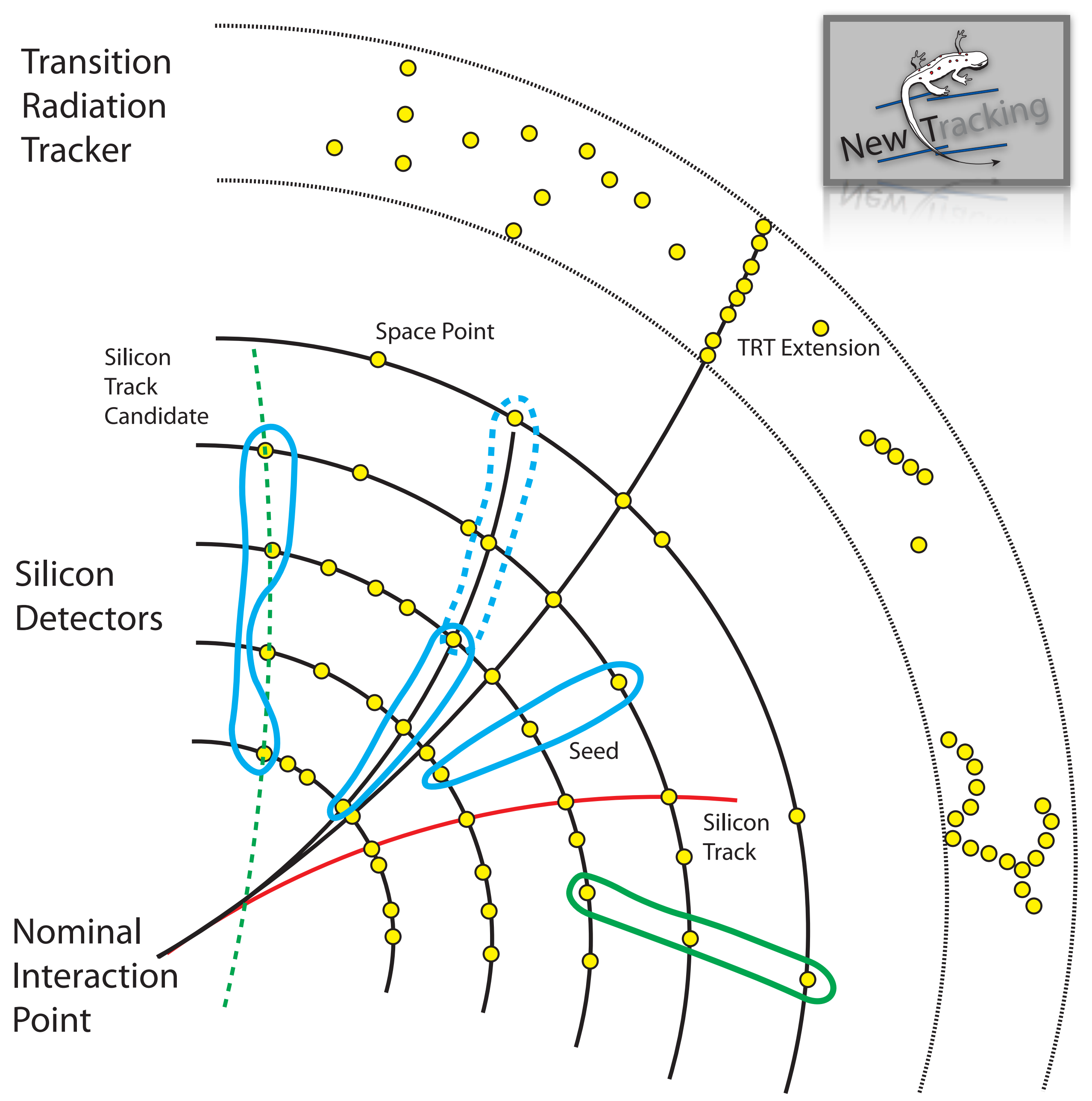


Track Finding: Can you find the 50 GeV track?



Track Finding

- the task of the track finding
  - ➡ identify **track candidates** in event
  - ➡ cope with the combinatorial explosion of possible **hit combinations**
- different techniques
  - ➡ rough distinction: **local/sequential** and **global/parallel** methods
  - ➡ local method: generate **seeds and complete** them to track candidates
  - ➡ global method: **simultaneous clustering** of detector hits into track candidates
- some local methods
  - ➡ track road
  - ➡ track following
  - ➡ progressive track finding



- some global methods
  - ➡ conformal mapping
    - Hough and Legendre transform
  - ➡ adaptive methods
    - Hopfield network, Elastic net, Cellular automaton ...

    (will not discuss the latter)

Conformal Mapping

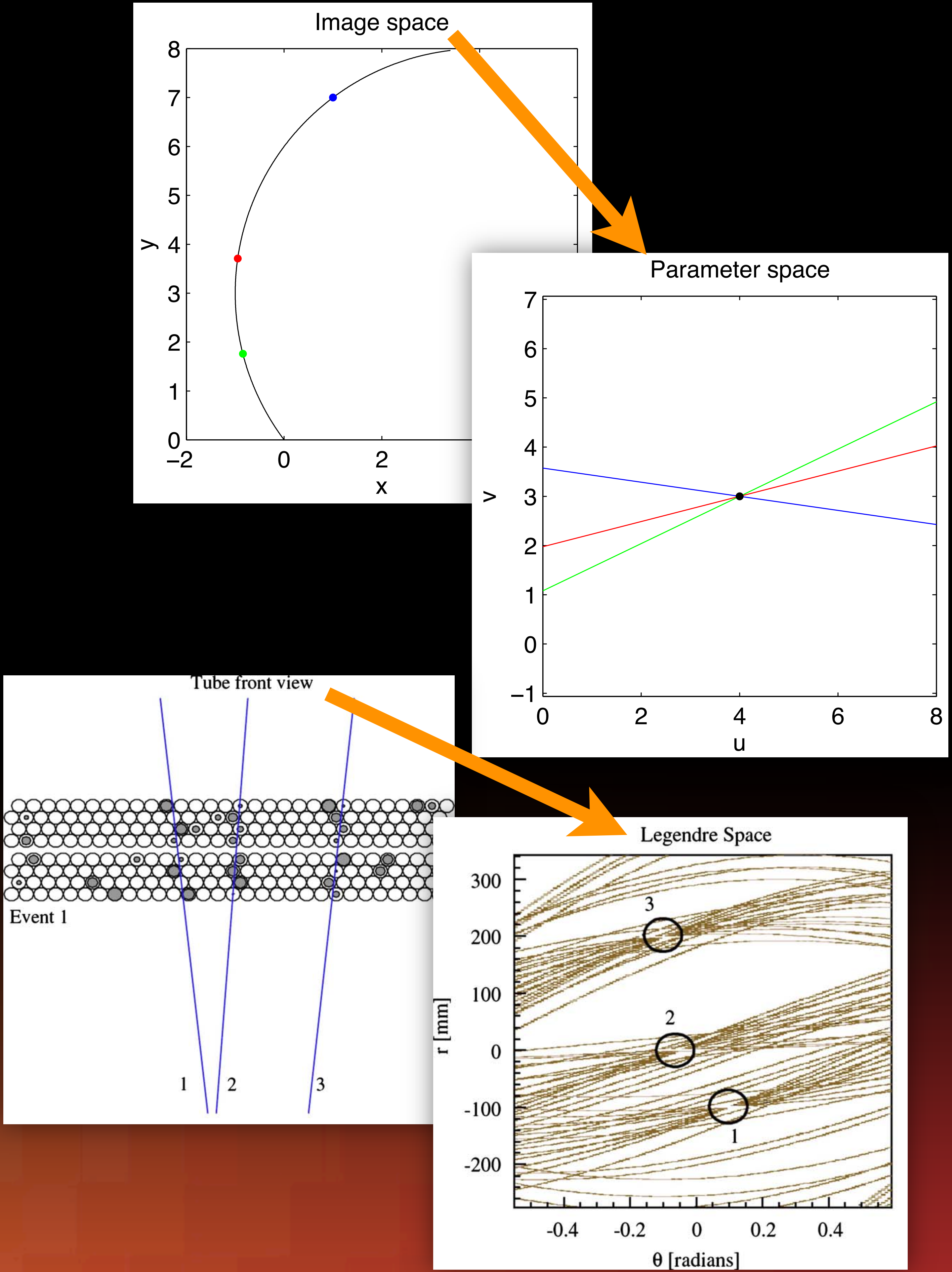
- Hough transform
  - ➡ cycles through the origin in x-y transform into straight lines in u-v

$$u = \frac{x}{x^2+y^2}, \qquad v = \frac{y}{x^2+y^2}$$

$$\Rightarrow \boxed{v = -\frac{x}{y}u + \frac{x^2+y^2}{2y}}$$

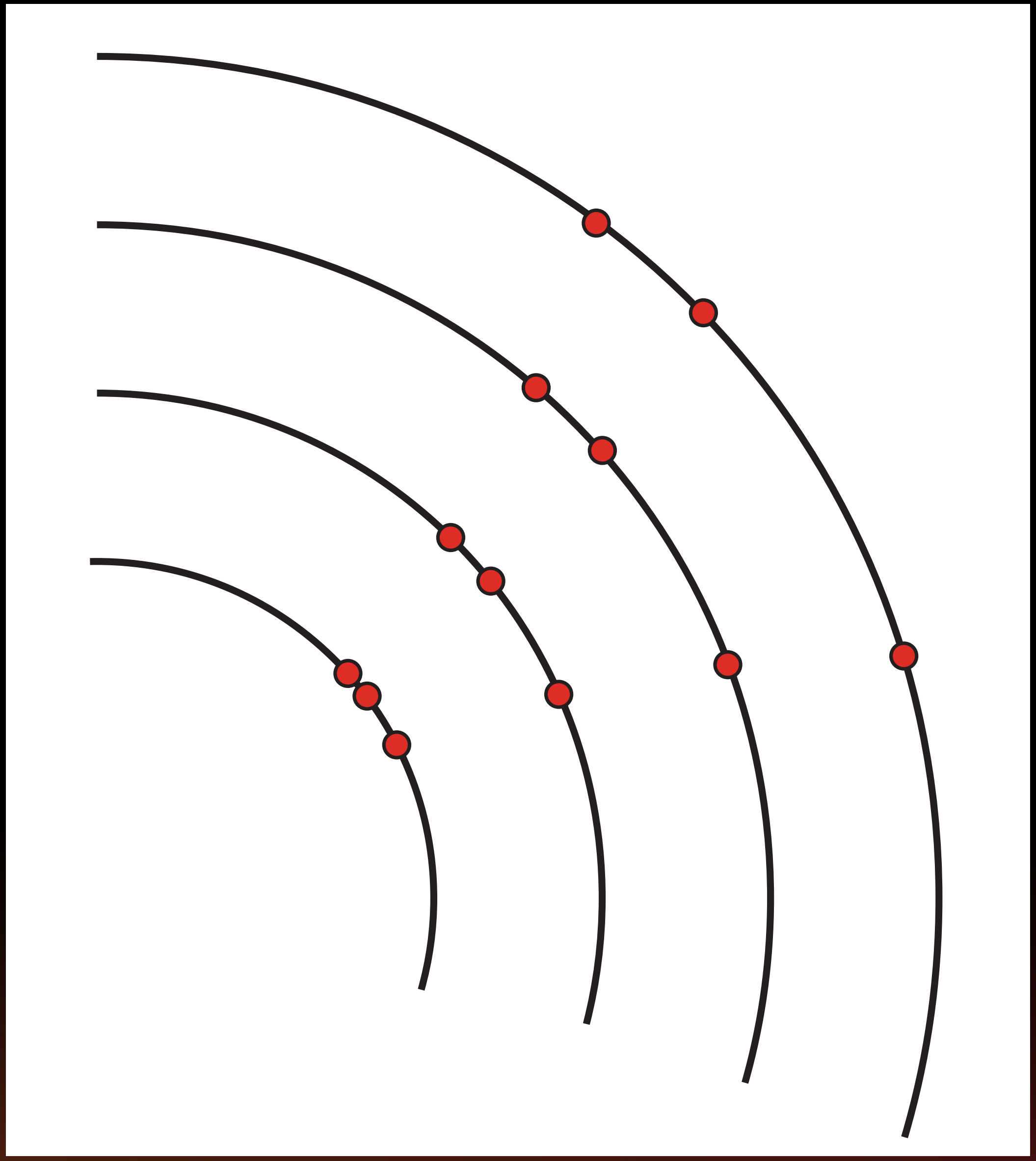
  - ➡ search for maxima (histogram) in **parameter space** to find track candidates

- Legendre transform
  - ➡ used for track finding in drift tubes
  - ➡ drift radius is transformed into sine-curves in **Legendre space**
  - ➡ solves as well L-R ambiguity



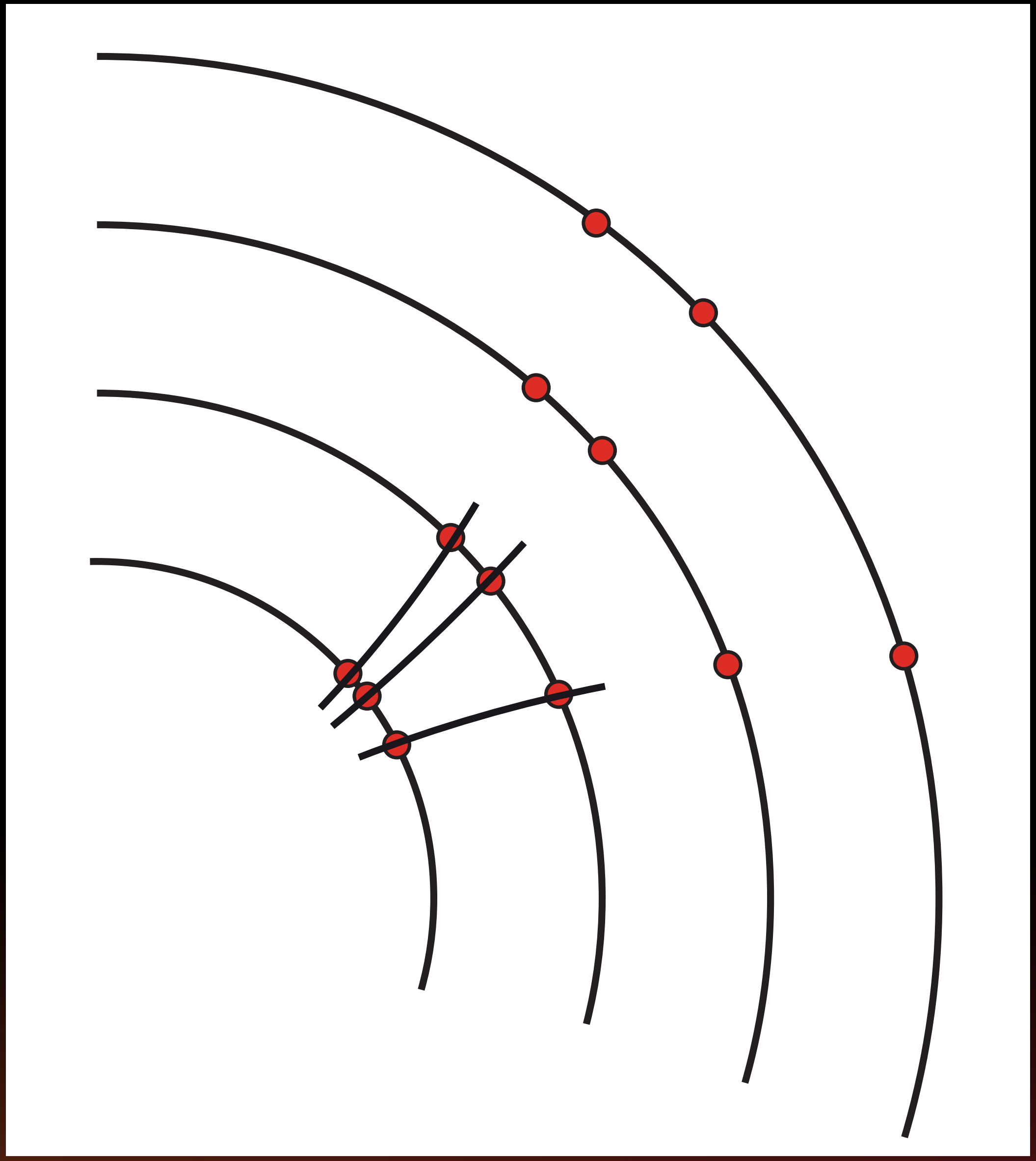
Local Track Finding

- Track Road algorithm



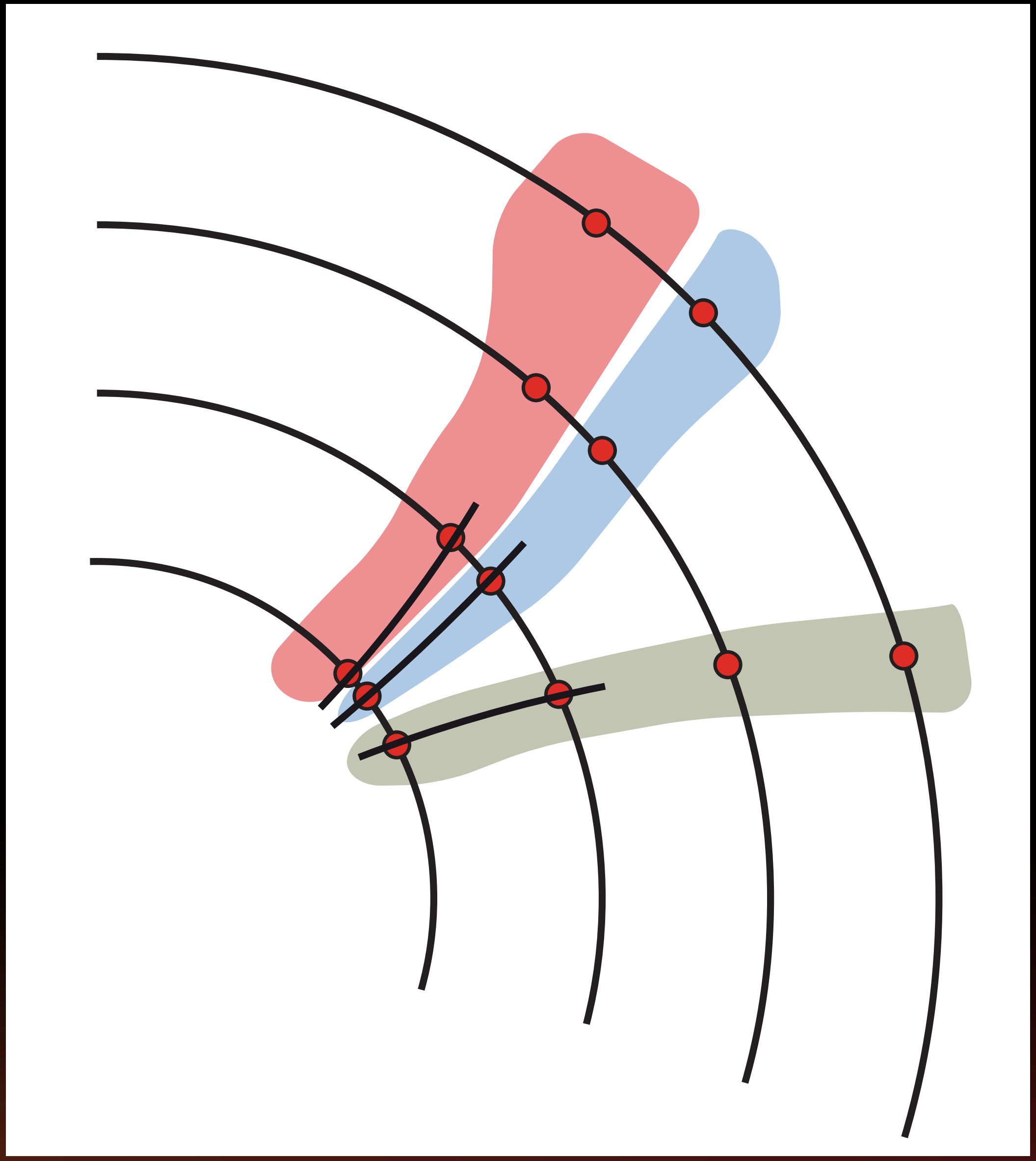
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits



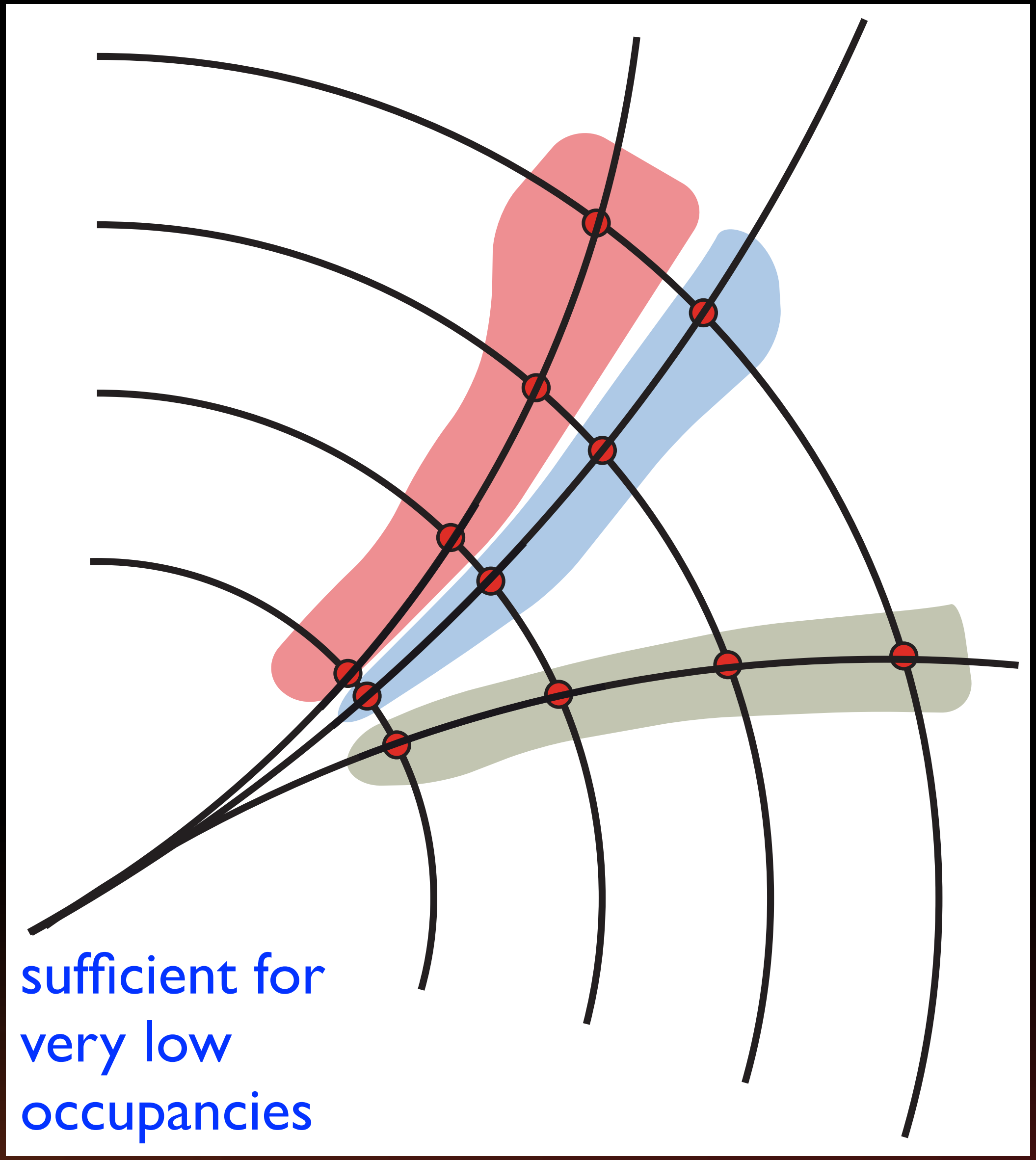
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory



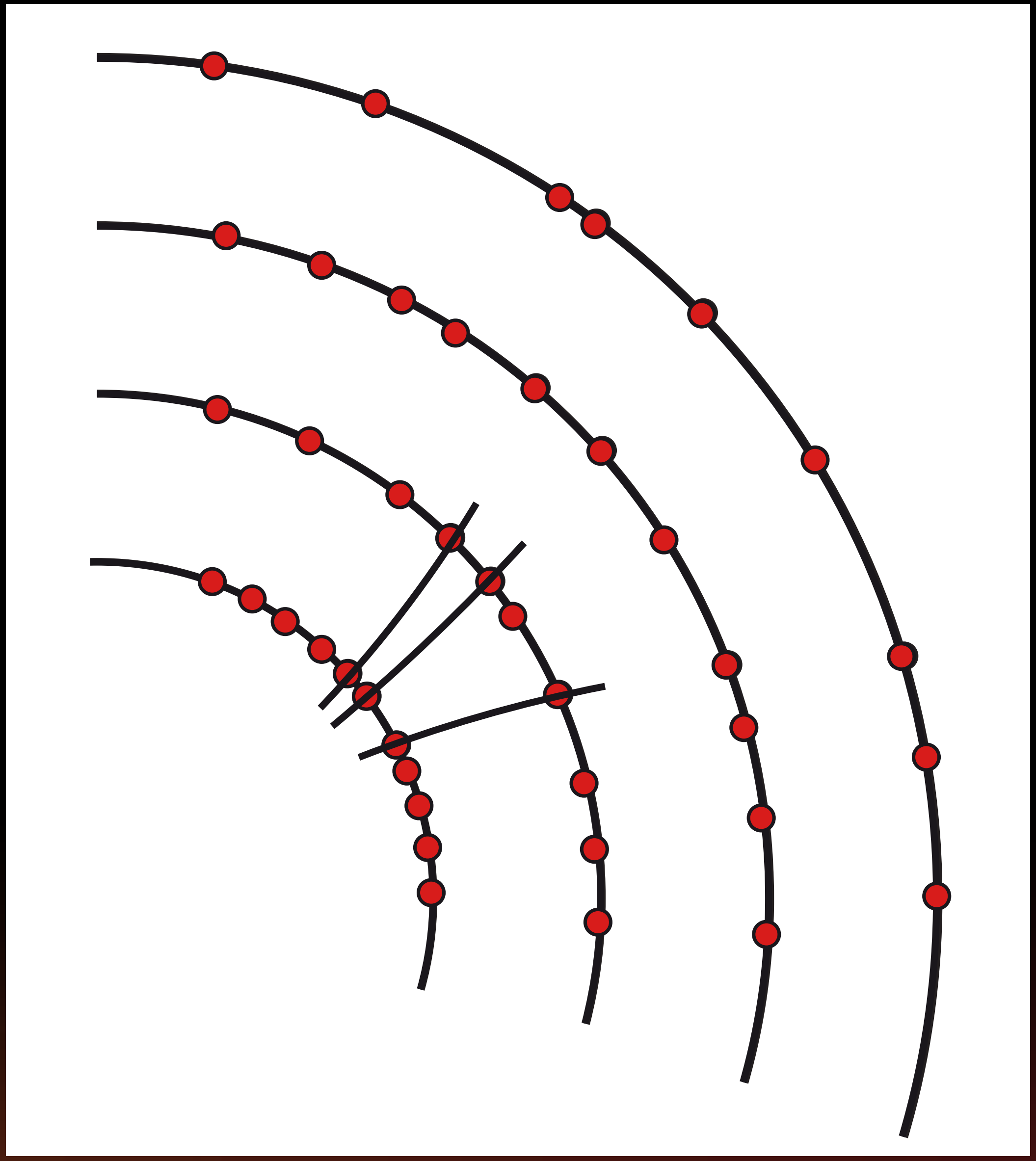
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**


sufficient for very low occupancies

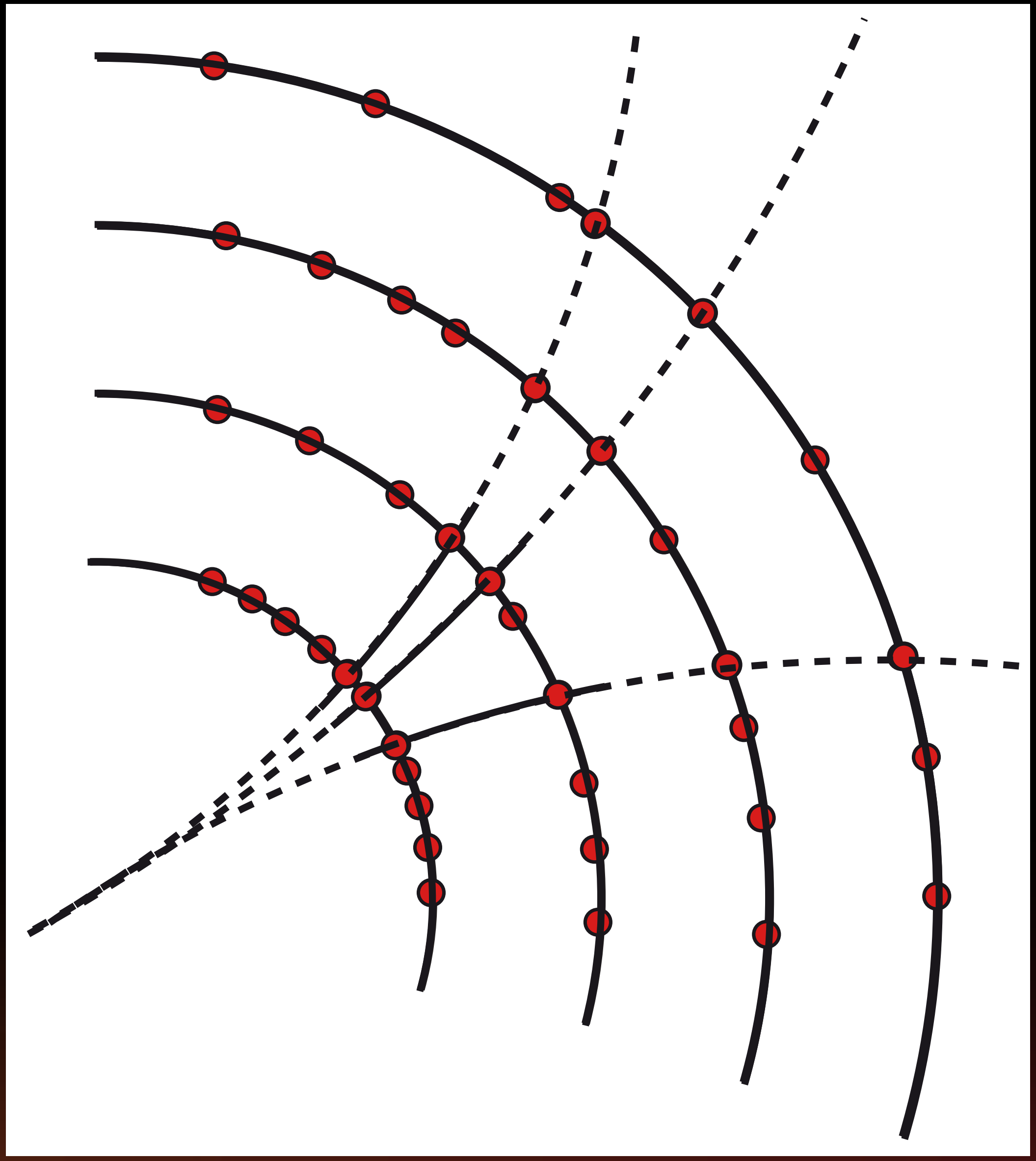
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits



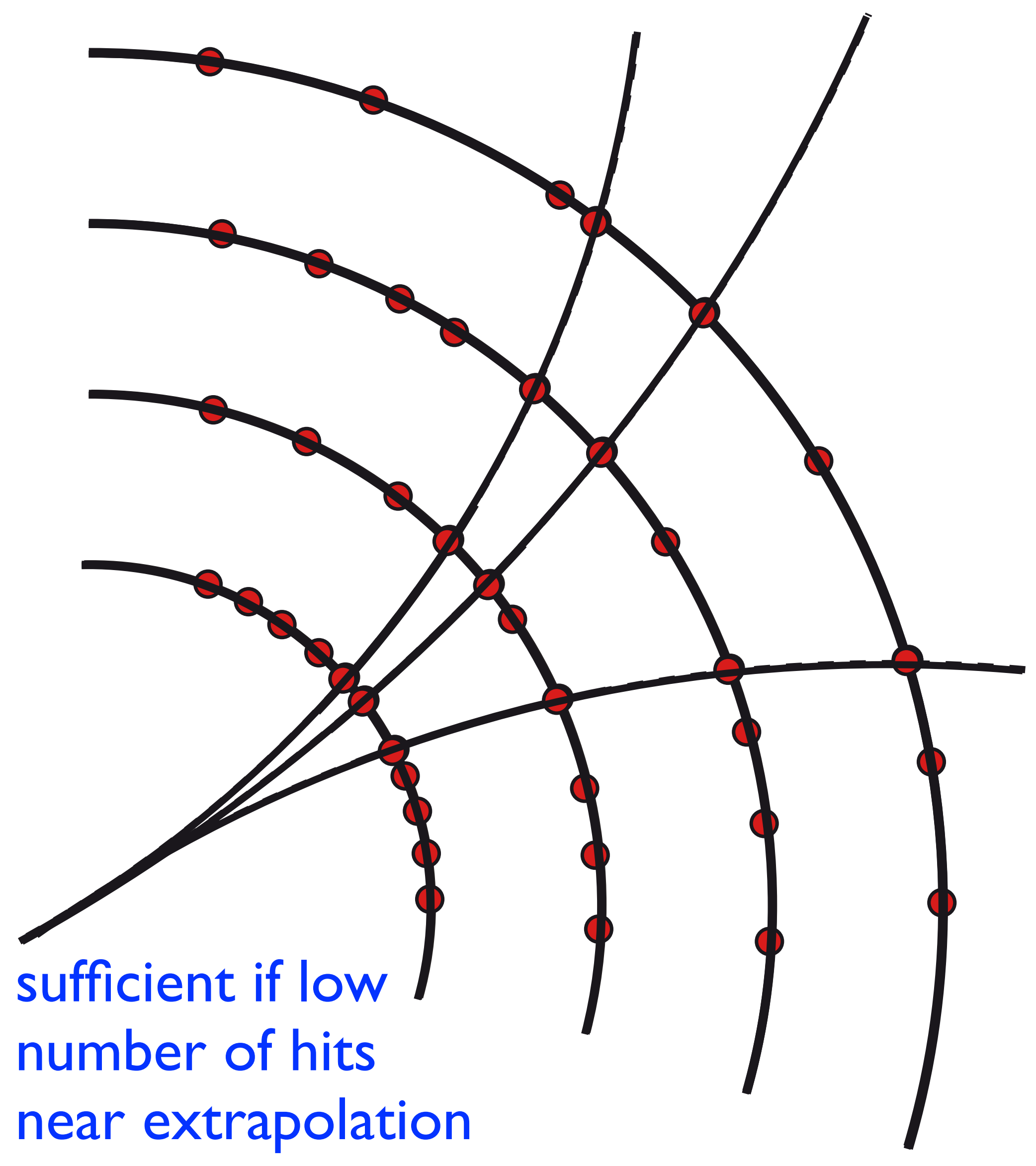
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory



Local Track Finding

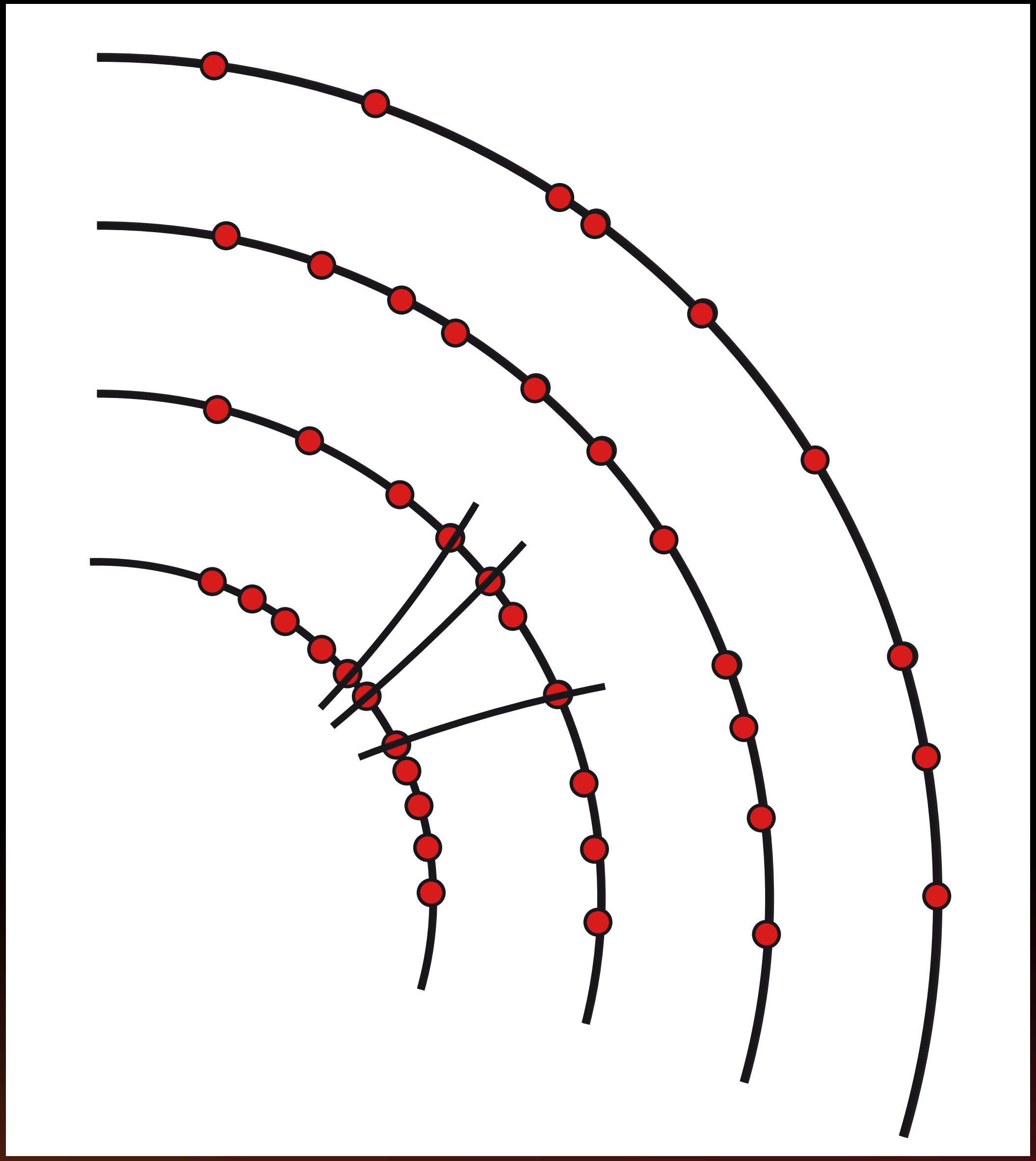
- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**



sufficient if low number of hits near extrapolation

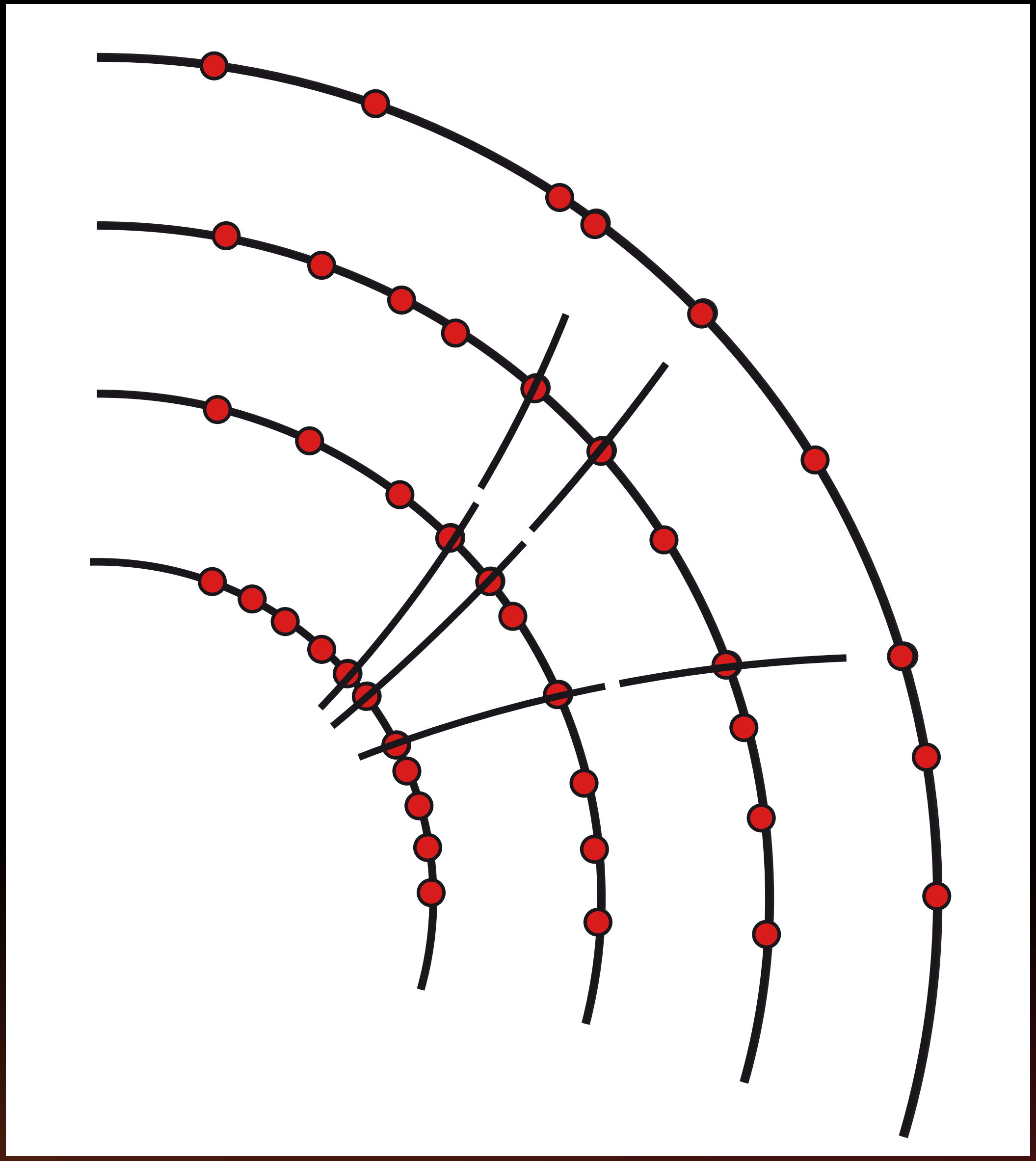
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits



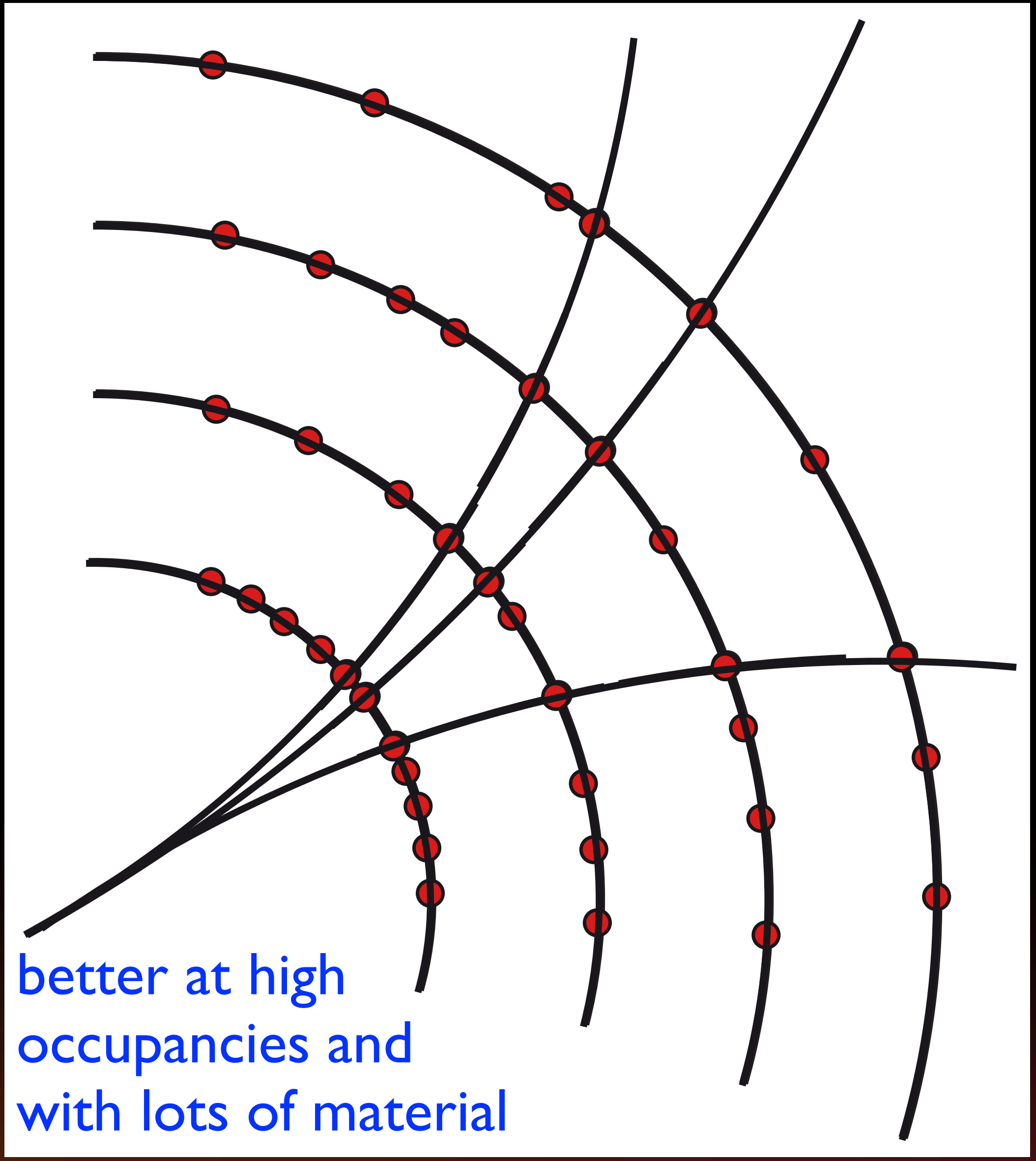
Local Track Finding

- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** to next layer, find **hit** and **update** trajectory



Local Track Finding

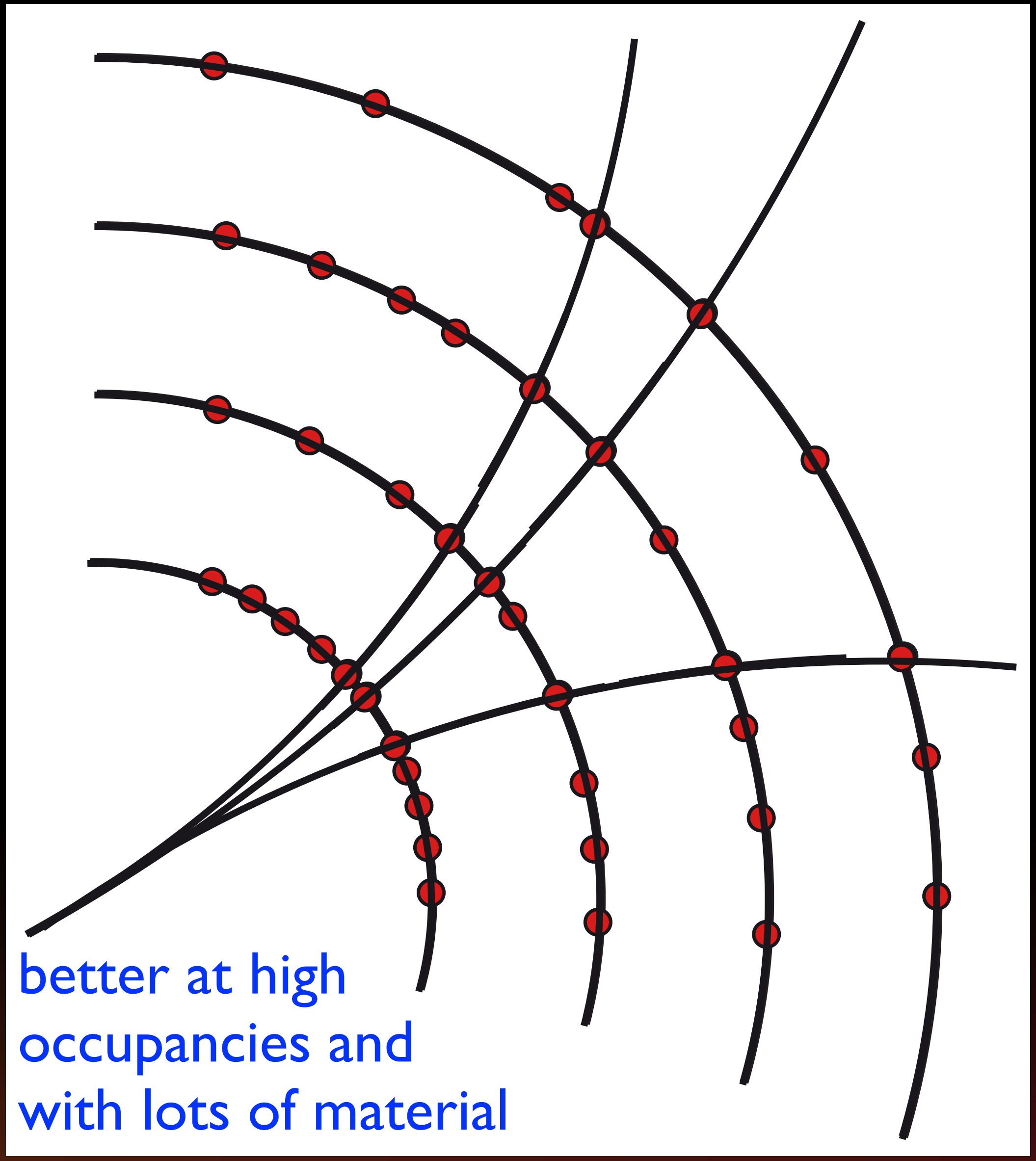
- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** to next layer, find **hit** and **update** trajectory
  - ➡ repeat until last layers to obtain **candidates**



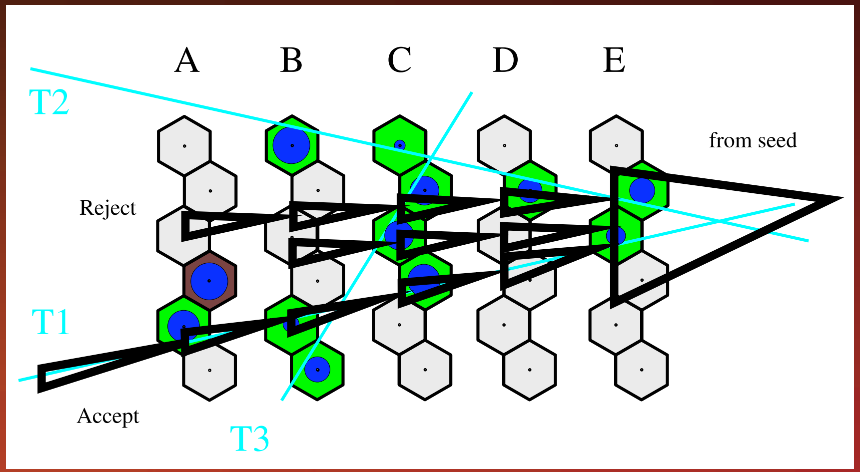
better at high occupancies and with lots of material

Local Track Finding

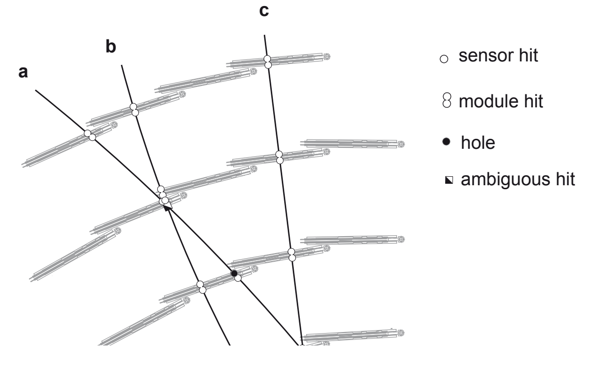
- Track Road algorithm
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ build **road** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Track Following
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** along the likely trajectory
  - ➡ select **hits** on layers to obtain **candidates**
- Progressive Track Finder
  - ➡ find **seeds** ~ combinations of 2-3 hits
  - ➡ extrapolate **seed** to next layer, find **hit** and **update** trajectory
  - ➡ repeat until last layers to obtain **candidates**
- Combinatorial Kalman Filter
  - ➡ extension of a Progressive Track Finder
  - ➡ full **combinatorial exploration**



better at high occupancies and with lots of material

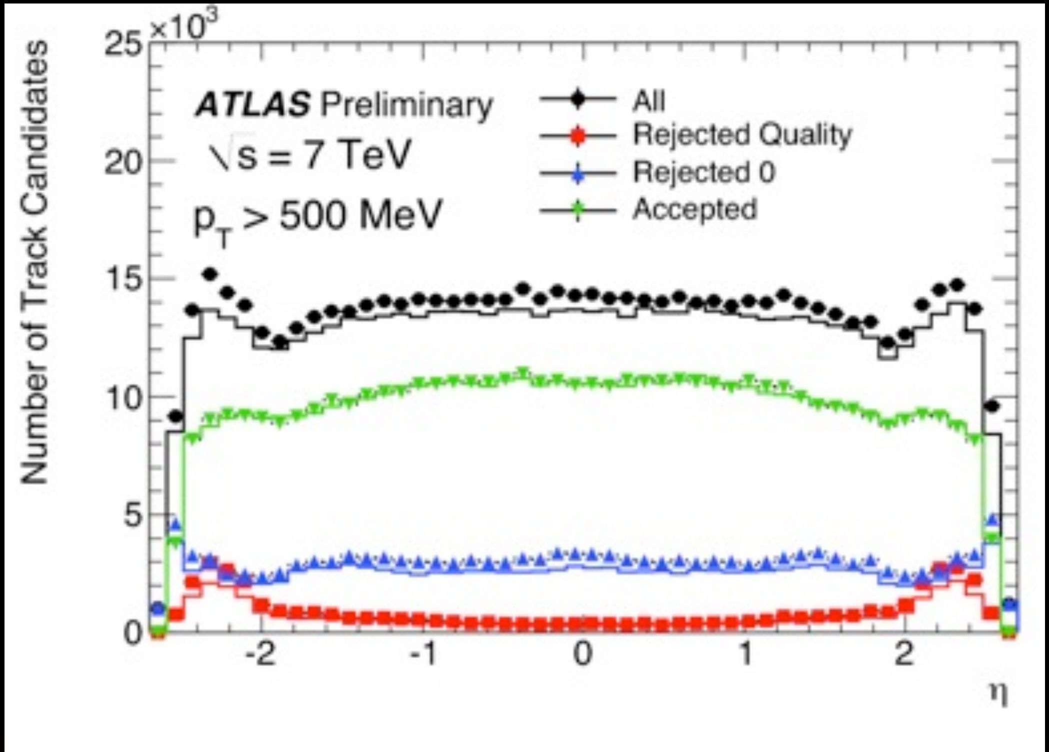


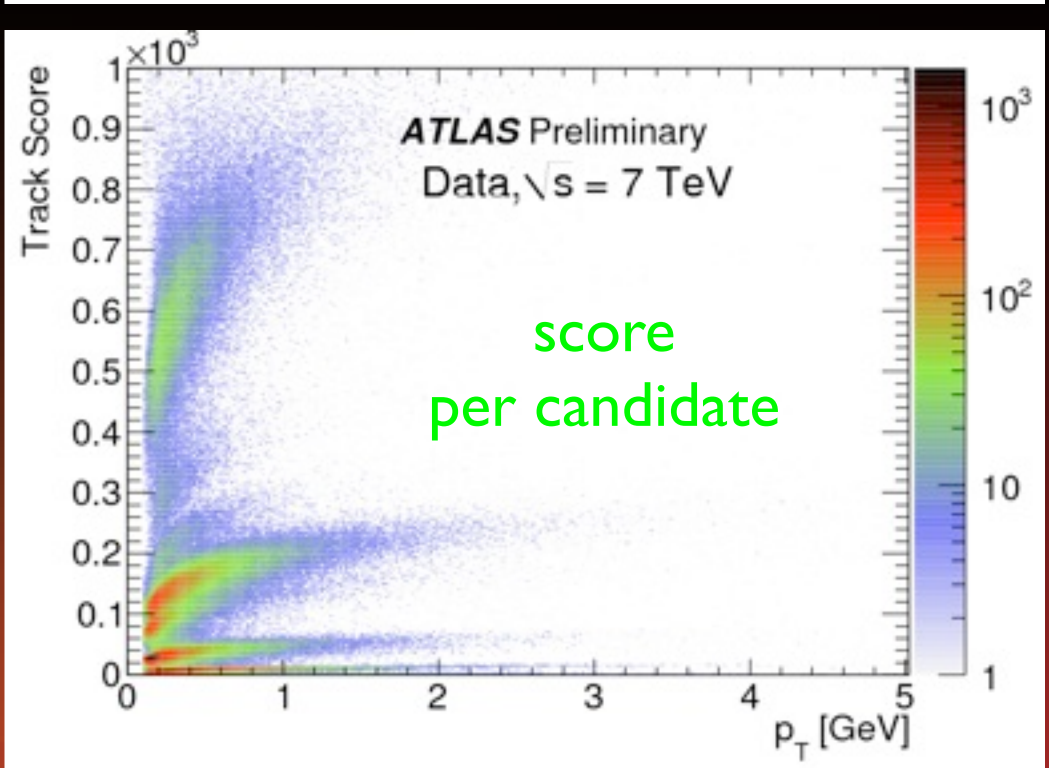
Ambiguity Solution



- track selection cuts
  - ➡ applied at every stage in reconstruction
  - ➡ still more candidates than final tracks

- task of ambiguity solution:
  - ➡ select good tracks and reject fakes
  - ➡ construct quality function (“score”) for each candidate:
    1. hit content, holes
    2. number of shared hits
    3. fit quality...
  - ➡ candidates with best score win
  - ➡ if too many shared hits, create sub-tracks if if possible
  - ➡ in case of ATLAS: as well precise fit



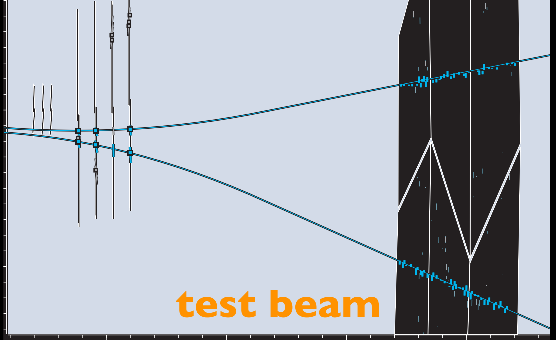


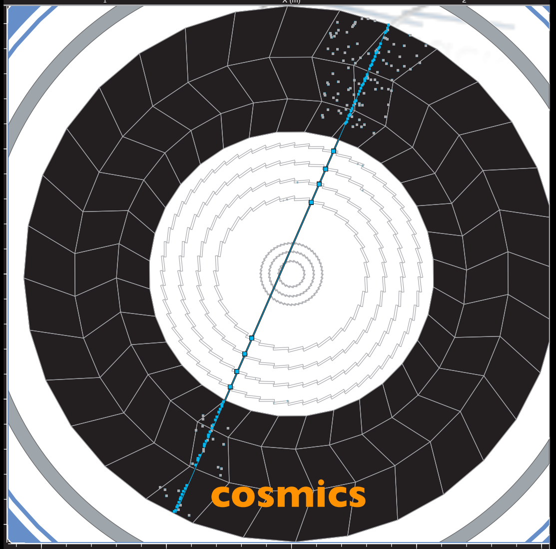
score per candidate

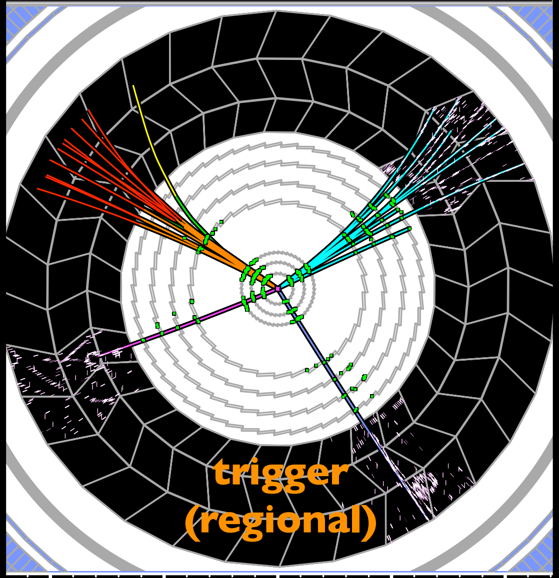
- DELPHI (LEP), LC-Detector:
  - ➡ full recursive ambiguity processor
  - ➡ D.Wicke, M.E.

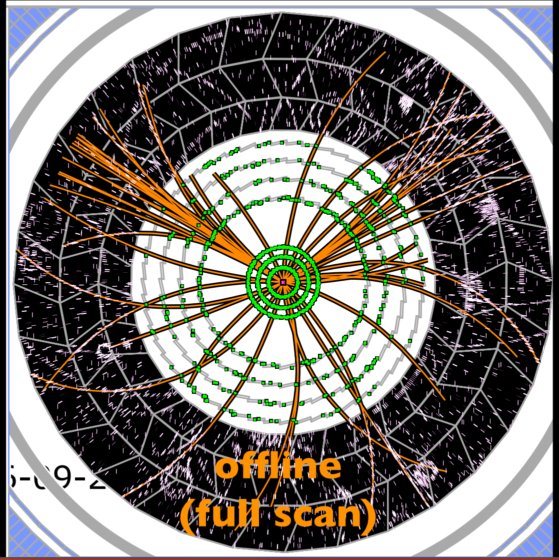
… and in Practice ?

- choice of reconstruction strategy depends on:
  - ➡ detector technologies
  - ➡ physics/performance requirements
  - ➡ occupancy and backgrounds
  - ➡ technical constraints (CPU, memory)
- even for same detector setup one looks at different types of events:
  - ➡ test beam
  - ➡ cosmics
  - ➡ trigger (regional)
  - ➡ offline (full scan)
- track reconstruction used by experiments
  - ➡ usually apply a **combination of different techniques**
  - ➡ often **iterative** ~ different strategies run one after the other to obtain best possible performance within resource constraints





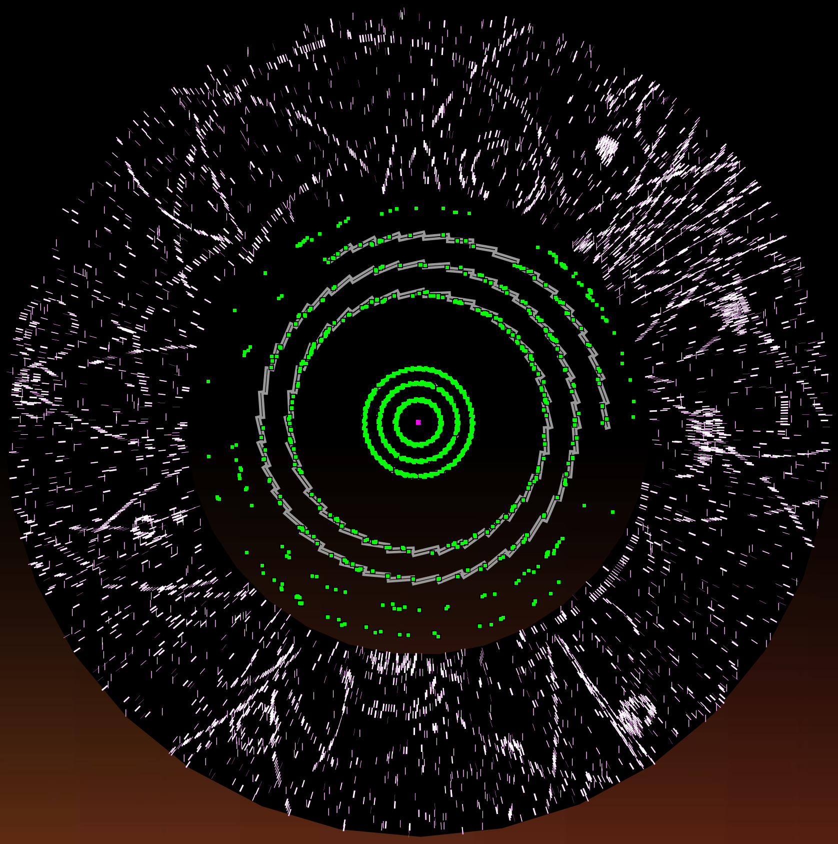




Example: ATLAS NewTracking

pre-precessing

- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation



Example: ATLAS NewTracking

pre-precessing

- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder

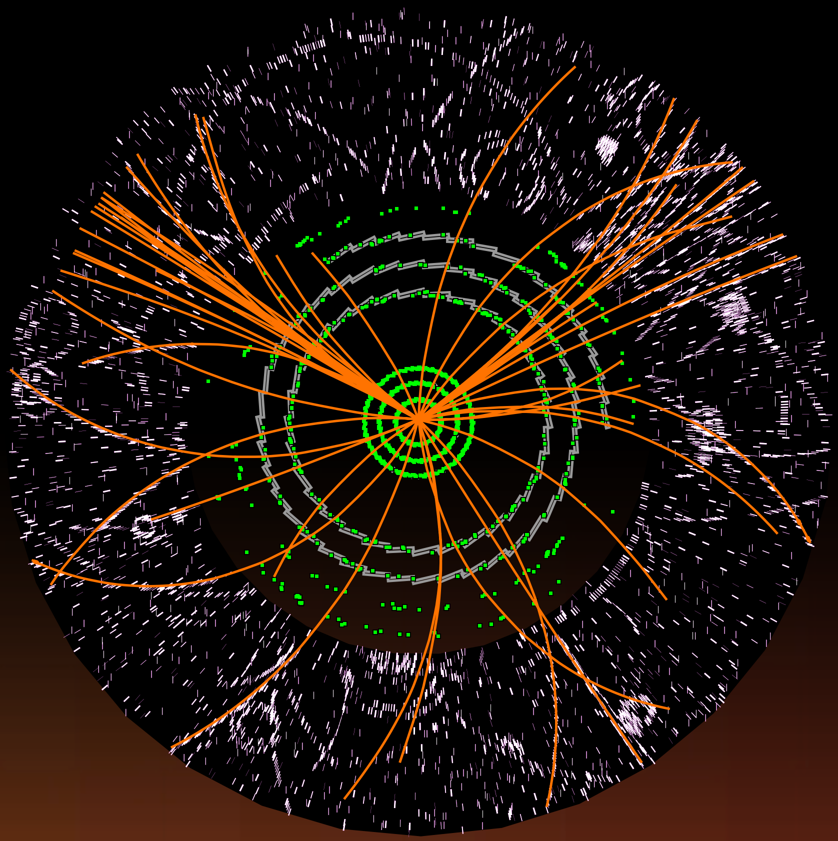
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates

ambiguity solution

- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

extension into TRT

- ➡ progressive finder
- ➡ refit of track and selection



Example: ATLAS NewTracking

pre-precessing

- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder

- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates

ambiguity solution

- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

extension into TRT

- ➡ progressive finder
- ➡ refit of track and selection

TRT segment finder

- ➡ on remaining drift circles
- ➡ uses Hough transform

TRT seeded finder

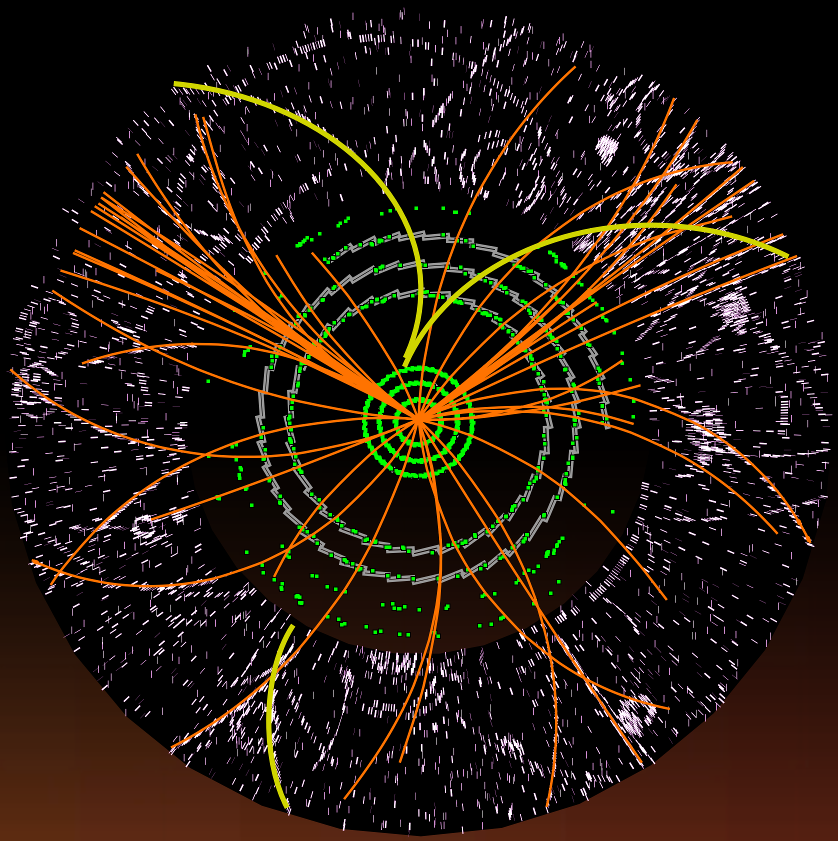
- ➡ from TRT into SCT+Pixels
- ➡ combinatorial finder

ambiguity solution

- ➡ precise fit and selection
- ➡ TRT seeded tracks

standalone TRT

- ➡ unused TRT segments



Example: ATLAS NewTracking

pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates

ambiguity solution
- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

extension into TRT
- ➡ progressive finder
- ➡ refit of track and selection

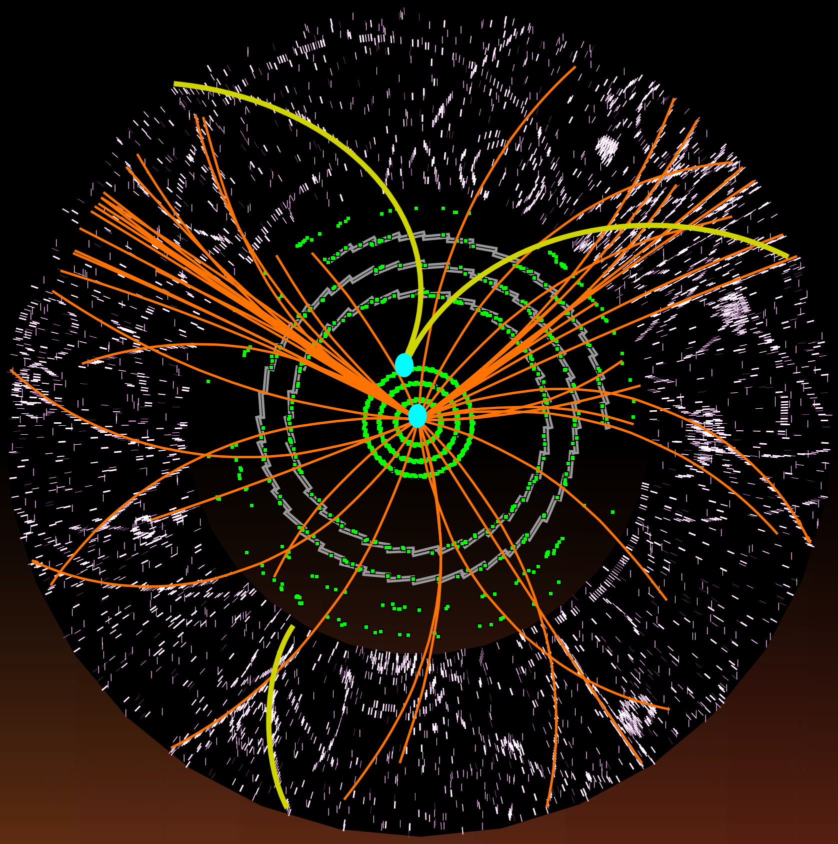
TRT segment finder
- ➡ on remaining drift circles
- ➡ uses Hough transform

TRT seeded finder
- ➡ from TRT into SCT+Pixels
- ➡ combinatorial finder

ambiguity solution
- ➡ precise fit and selection
- ➡ TRT seeded tracks

standalone TRT
- ➡ unused TRT segments

vertexing
- ➡ primary vertexing
- ➡ conversion and V0 search



Let’s Summarize...

- discussed concepts for track reconstruction
- have overview of strategies and mathematical tools
- discussed an example of a track reconstrucion package (ATLAS NewTracking)
- next is to talk about vertexing and its applications