Track Models in Reconstruction

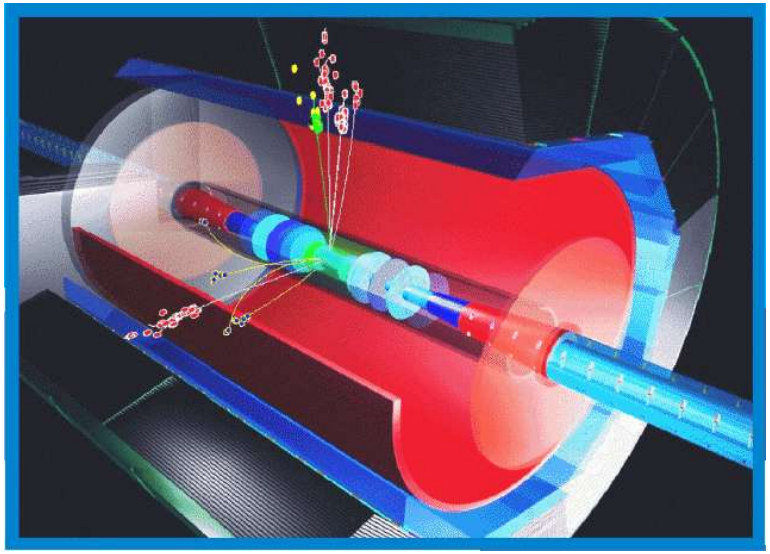
Benno List

ILC Workshop Valencia 2006
8.11.2006

- **Introduction**
- **First Theme: Iterative Track Reconstruction**
- **Second Theme: Separating Hits, Tracks, Algorithms**

Introduction

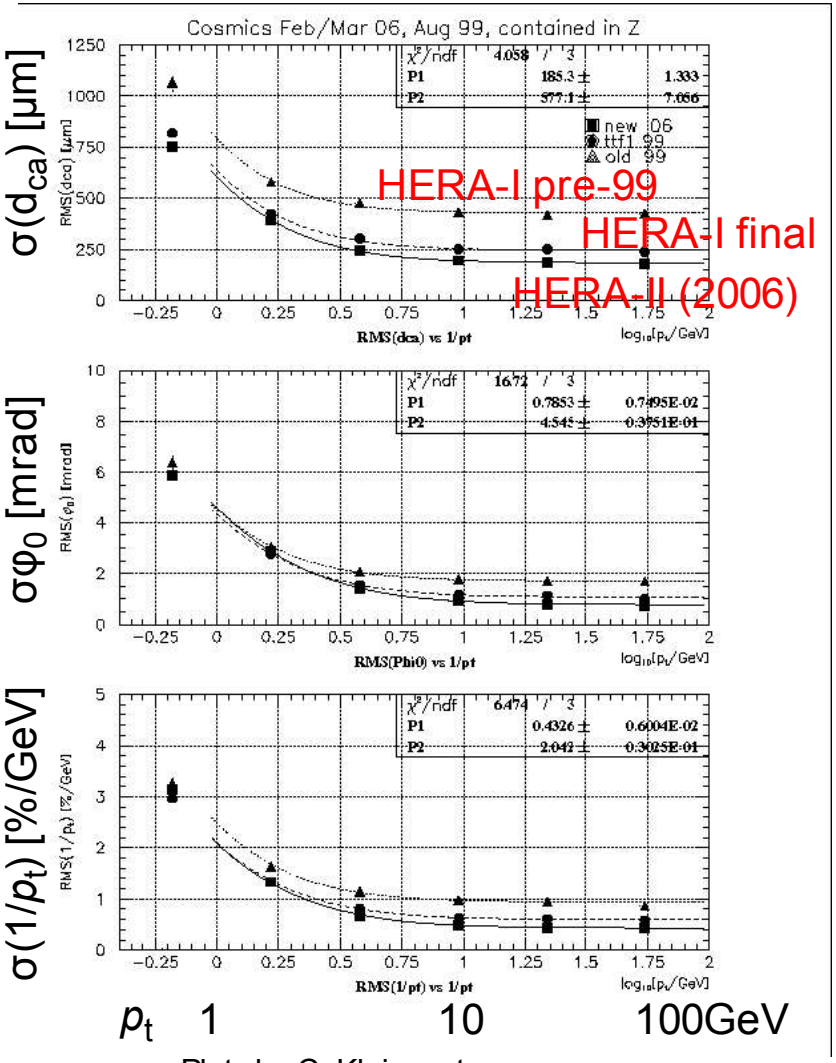
- ILC Detector Concept: One big tracker
  - vertex detector
  - central tracker
  - tracking calorimeter
  - muon system
- Emphasis is on **Precision** tracking



- Aim for a **Concept** of tracks that is able to encompass “high-order effects”, even when the initial implementations are based on “tree level relations”
- Would be nice to have a common concept for all trackers

Lessons from H1

- In 1999: Reprocessing of H1 data
- Improved tracking resolution of central jet chamber by 40%
- At HERA-II: Another 10%
- => Resolution improved by a factor of 2 over the last 7 years
- This was not due to any single problem source, but is the result of hunting down many percent effects
- Some effects were difficult or impossible to incorporate into the tracking model that is used

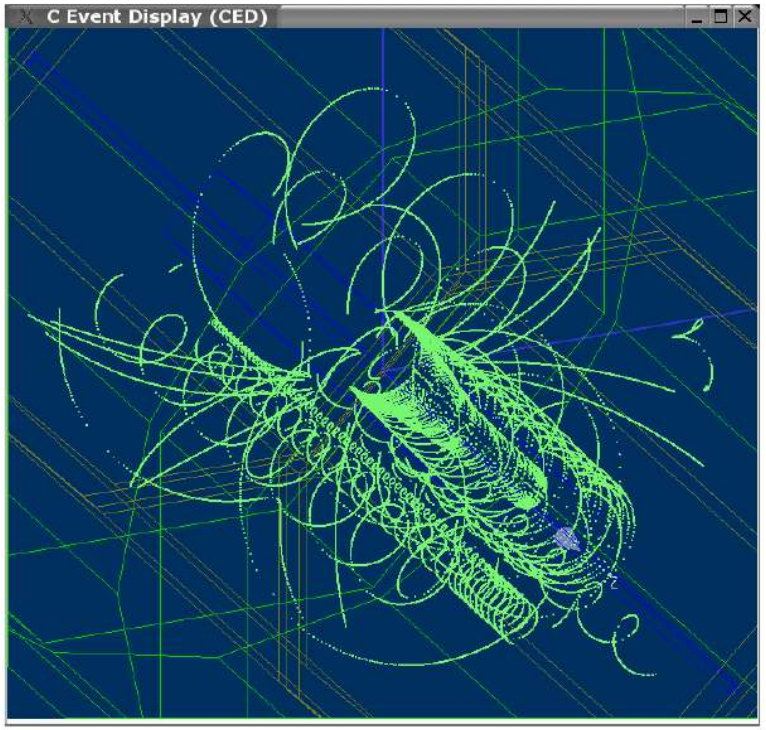


Plots by C. Kleinwort

ILC Tracking Demands

- High energy leptons: 250GeV electrons, muons:
  - extremely good pt resolution
  - radiative losses significant, even for muons!?
- High density jets
  - Extremely good pattern recognition
- Tracking down to low momenta (Giga-Z):
  - energy loss corrections depend on particle type
  - multiple scattering depends on particle type
  - Curlers

Consider “phase space corners” early



Steve Aplin, ILC software tools workshop, Cambridge, 4.4.2006

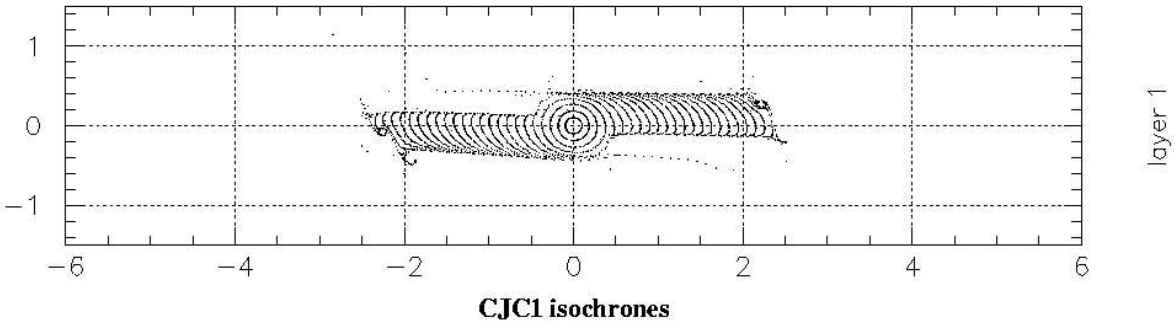
First Theme

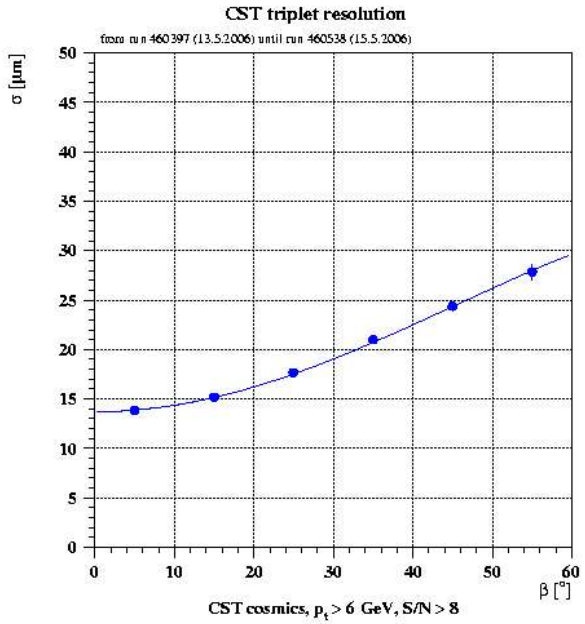
My First Point:

- For optimal tracking, you'll always have corrections that depend on the result of the next reconstruction step
- As a consequence, tracking is an iterative process

Hits and Tracks

- Simple example: Hits on a drift chamber:
  - Corrections depend on angle β within drift cell and side on which track passes
    => hit position known only in conjunction with a track!
  - Wire tilt, twist, sagging: corrections to hit position that depend on z of track
    => needs track, again
  - Silicon sensors: Alignment corrections, silicon bending lead to track-dependent corrections. Also resoloution depends on β of track
    => not even covariance matrix is independent of external track

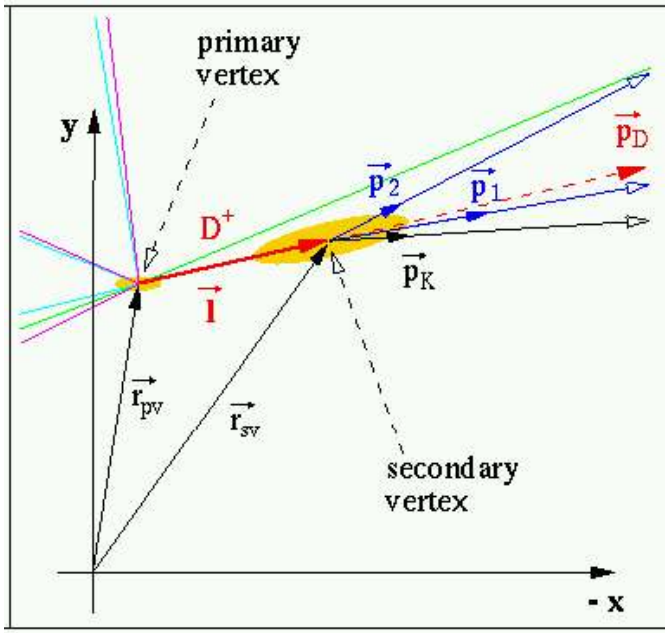




Track Finding&Fitting Depends on Particle Type

- Going through the detector, a particle
  - loses energy through ionisation
  - loses energy through bremsstrahlung
  - undergoes multiple scattering
- These processes depend on the particle's mass (and charge)
- Mass is generally unknown during initial track reconstruction
- Particle ID is based on result of tracking, plus other detector parts:
  - dE/dx measurement
  - energy deposition in calorimeter, track in muon system
- If a particle ID has been made, tracking can be improved:
  - different parameters (energy loss, multiple scattering) for Kalman filter
    => may even lead to changes in hit assignment to tracks!
  - May join two tracks at a kink, or separate them

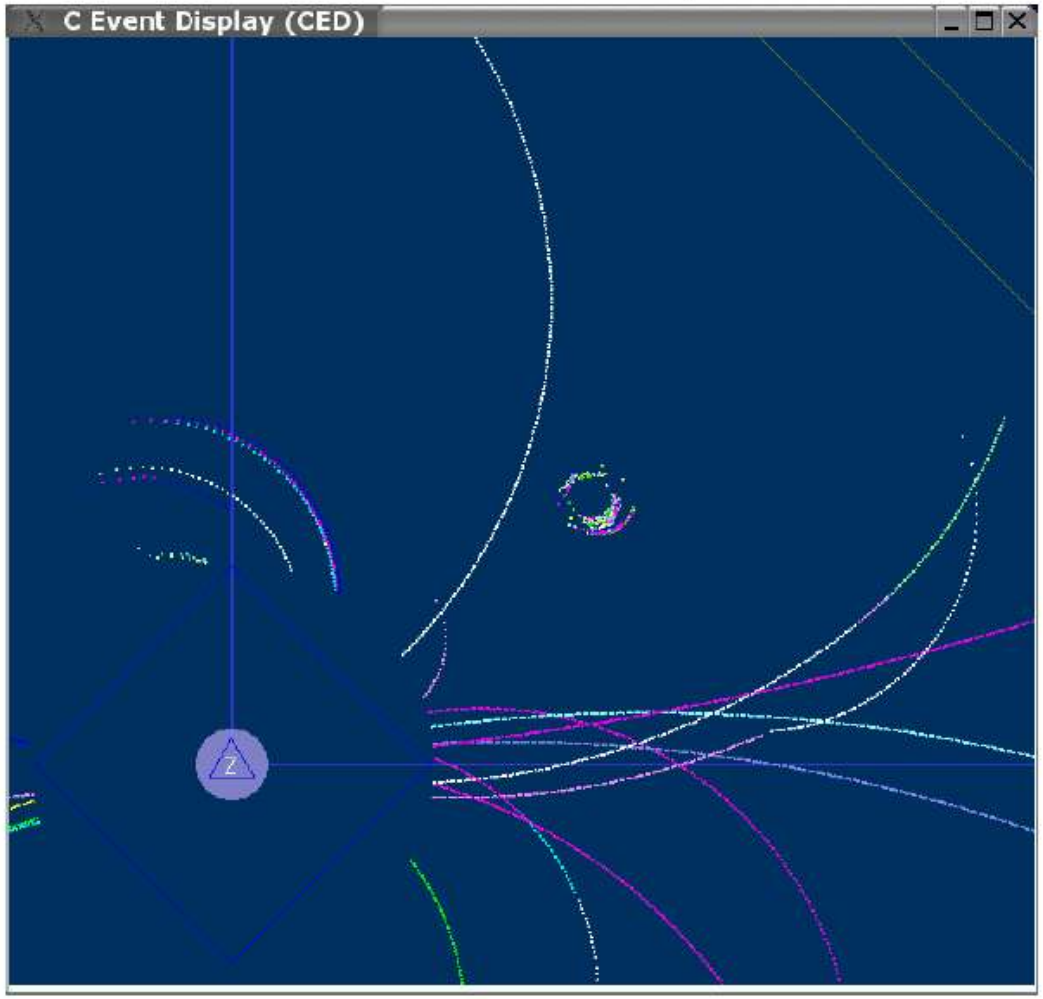
Vertexing and Particle ID



- Golden decay D* -> Kππ,
  with D0 -> Kπ at secondary vertex
- Assignment Kaon/Pion often ambiguous,
  dE/dx is not always good enough
- Precise location of secondary vertex may
  be improved when track is corrected with
  kaon hypothesis.
  It may be necessary to try both
  assignments (pion or kaon) in the same event!
- => Particle ID hypothesis can come very late in the game
- => Want to store enough info with each track so that track parameters for a given particle hypothesis can be derived

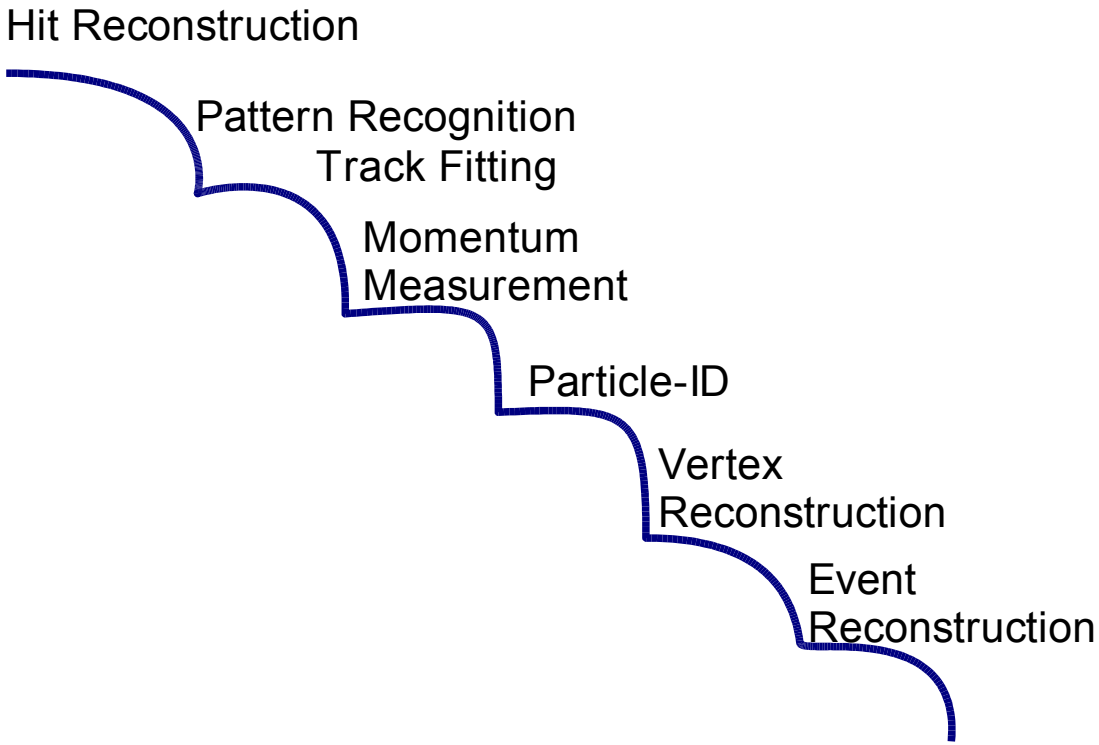
Kinks

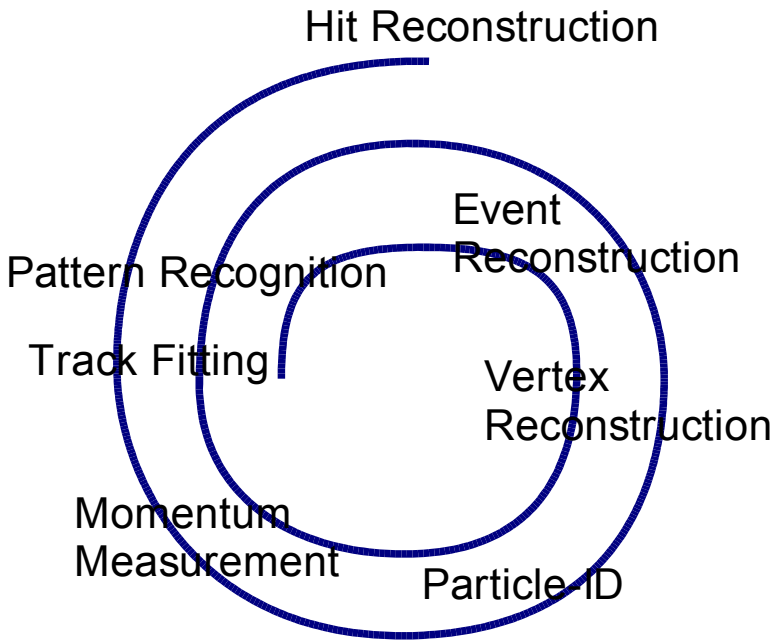
- Whether a kink is “significant” or not depends on particle type [hypothesis]
- Example of a case where the form of a trajectory between the ends is of interest:
  - Want to combine as many kinked tracks into single tracks as possible to avoid double counting
  - But: Kinks may indicate in-flight decays, also from long-lived new particles (e.g. from certain SUSY breaking scenarios)
    => want to keep the “kink” info! => Can that be included in the data model?



Steve Aplin, ILC software tools workshop, Cambridge, 4.4.2006

Waterfall Model vs. Spiral Model





Each step done independently
Information available at later stages used to fudge/correct info from previous stage

At each stage, information from later stages can be used, if present.
As new information becomes available, earlier stages may be repeated to get optimal result

Second Theme

My Second Point

- Try to Separate:
  - Hits
  - Trajectories
  - Algorithms that find & fit Trajectories from Hits

- The Goal: The tracking software should be
  - Performant: find all tracks, have best possible parameter resolution
    => requires well-structured code that can be understood, maintained and verified
  - Fast enough

Tracking Across Subdetectors

Want to follow a particle throughout the detector

- This means: Tracking across subdetector boundaries
- May be facilitated by a common track model suitable to describe tracks under all circumstances:
  - Within (rapidly) changing magnetic field (coil!)
  - Within a dense medium (calo)
- A track stub from any subdetector should be usable as seed for any other (neighboring) tracking detector:
  - Follow backscattered particles from calo back into TPC?
  - Make sure a photon is not an electron with a track missing
  - Use calo seeds to extend tracking range to largest $|\cos\theta|$
  - Interface between trajectories:
    intersection with a given surface (2 par's)+momentum (3 par's)

Isolating Tracks from Hits

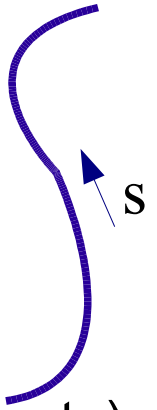
- The ideal situation:
  Have a track finder/fitter that works for all detector geometries
- The reality: Track finders mostly tailored to a specific situation:
  - 3d hits (TPC, pixel tracker): look for full helix
    2d hits (jet chamber, silicon strips, muon chambers): look for circles in r/phi
  - circular structure (TPC, jet chamber, barrel silicon tracker): progress in r
    or planar structure (forward trackers): proceed in z
- A fully detector-independent tracking code would probably be prohibitively slow and/or have suboptimal performance
- But: We can sacrifice a bit of speed for generality, if we get:
  - Cleaner software design => better maintainability, better reliability
  - More versatile software => faster integration of new detector parts or algorithms
  - Easier transfer of tracking code across detector concepts

Hits and 3D Space Points

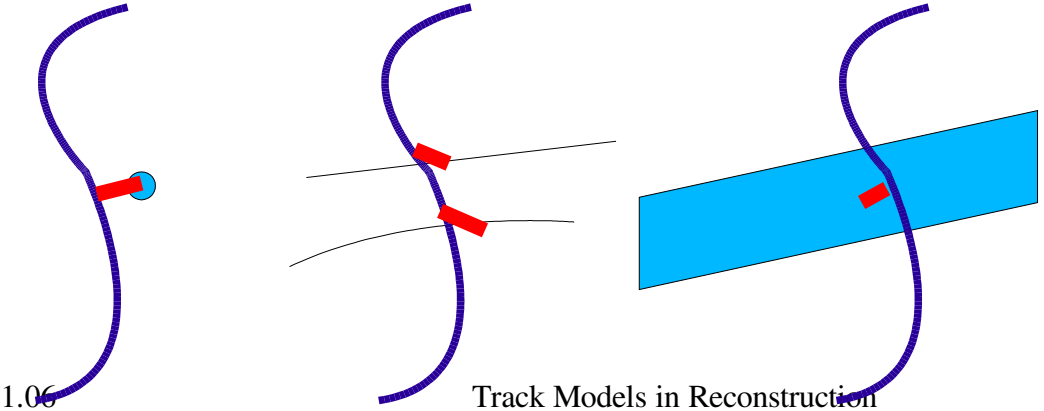
- Idealizing detector response signals (“hits”) as 3D space points with a covariance matrix may work much better in theory (MC) than practice (data)
  - Alignment issues: alignment corrections in r/phi often depend on z coordinate, often in a non-linear way. Often these effects are missing in MC.
  - Corrections may depend on track parameters (incidence angle)
    => track fit has to be iterated, hit has to be corrected
  - Ambiguity resolution (mirror hits, multiple hit hypotheses from ganged pixels)
- Not all hits are well described by 3D space points:
  - hits on silicon strip sensors: one coordinate is not described by a Gaussian, but by a box.
  - Hits in single silicon pixels or single calo cells: also non-Gaussian
  - Silicon sensors may be bent, wires sag: hits are “banana-shaped”

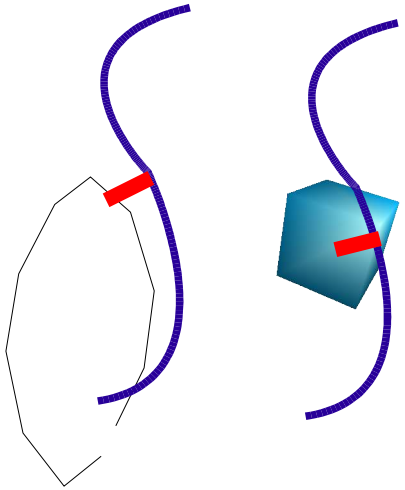
Basic Concepts (Classes)

- A Trajectory:
  - A curve in 3d-space,
  - Parametrized by some arc length s
  - Need not be a helix or a straight line!
  - Knows its own momentum, mass, and charge (may just be an estimate)



- A Hit:
  - Can be a space point, several space points (mirror hits!), a straight or cuved line, a piece of a cylinder, of a plane
  - Knows how to get its own distance from a trajectory





Fitting a Trajectory to Hits

- To fit a trajectory to hits: Minimize some sort of penalty function
- Encapsulate the trajectory's precise form in the trajectory class
- Trajectory can “answer questions” like:
  - arc length or intersection point with a plane, a cylinder, a sphere
  - point of closest approach or distance to a point, a line, a circle
  - For each arc length, we may also get the momentum vector of the track
- Hit can answer question:
  - What is the hits distance to a trajectory
  - Hit calculates this using appropriate info from track
  - Hit may “correct itself” using (possibly preliminary) information from track

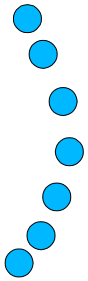
Fitting without Knowing the Parametrization

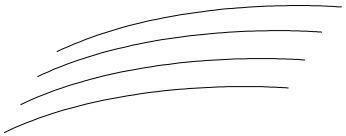
Fitting a Trajectory is possible without knowing the form of the parametrization:

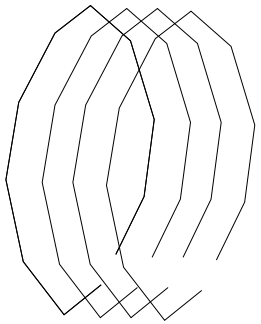
- Fitting means minimizing a penalty function (χ^2)
  this needs the derivatives w.r.t. the parameters $\partial\chi^2/\partial p_i$
- Penalty can be gotten from the hits, along with vector of derivatives
- If trajectory provides number of parameters and derivatives of penalty term w.r.t. the parameters, an external fitter can minimize the penalty term without knowledge of the parametrization at all
- A kinematic fitter based on this principle has been developed and works quite nicely. Adding new particles with new parametrizations is quite easy.

Helping Pattern Recognition

- Track Fitting comes after Track Finding
- Important question: Where's the next hit
- Hits belong to a “Detector”; the detector knows how to order hits: radially, along z, whatever
- During pattern recognition, a track finder may use a detector object to sort the hits, or provide the next hit candidate .g. for a Kalman filter







Conclusions

- Structure tracking software such that it can use external imput to improve the tracking result
  - particle ID information
  - Track seeds
- Be careful with abstractions like “hits are space points”: consider early how higher-order corrections can be incorporated
- How far can we separate Tracking algorithms from specific trajectory parametrizatiuons and specific assumptions about detector geometries without compromizing performance?

Reserve

Interfacing between Trajectories

- Problem:
  - Two tracks from 2 tracking devices shall be combined
  - Each tracker uses its own tracking model
- Ansatz 1: Use a commmon 5-parameter helix parametrization
  - Problem: does not work for neutral particles or zero B field
- Ansatz 2: Use a 3d-point and momentum vector (plus particle mass, charge):
  - Problem: covariance matrix of 6 parameters is singular
- Ansatz 3: Use point of intersection with a plane or a cylinder or a sphere (2 parameters) and momentum vector
  - Works for all particles in all fields
  - Covariance matrix well-behaved
  - It is exactly the information the particle carries when it crosses the plane/cylinder/sphere

Remarks

- I'm not an ILC detector expert
- I'm not an ILC software expert
- I'll present a few ideas that come from my personal experience with tracking software at H1
- Many things may be old news to you
- The question (I'm asking) is not
  “What should the software be able to do in the next release?”
  but
  “Does the existing software and data model make certain things easy/hard/impossible?”