Inner Detector
Simulation and Reconstruction Software:
Status and Plans

presented by
Jo Pater
Manchester University

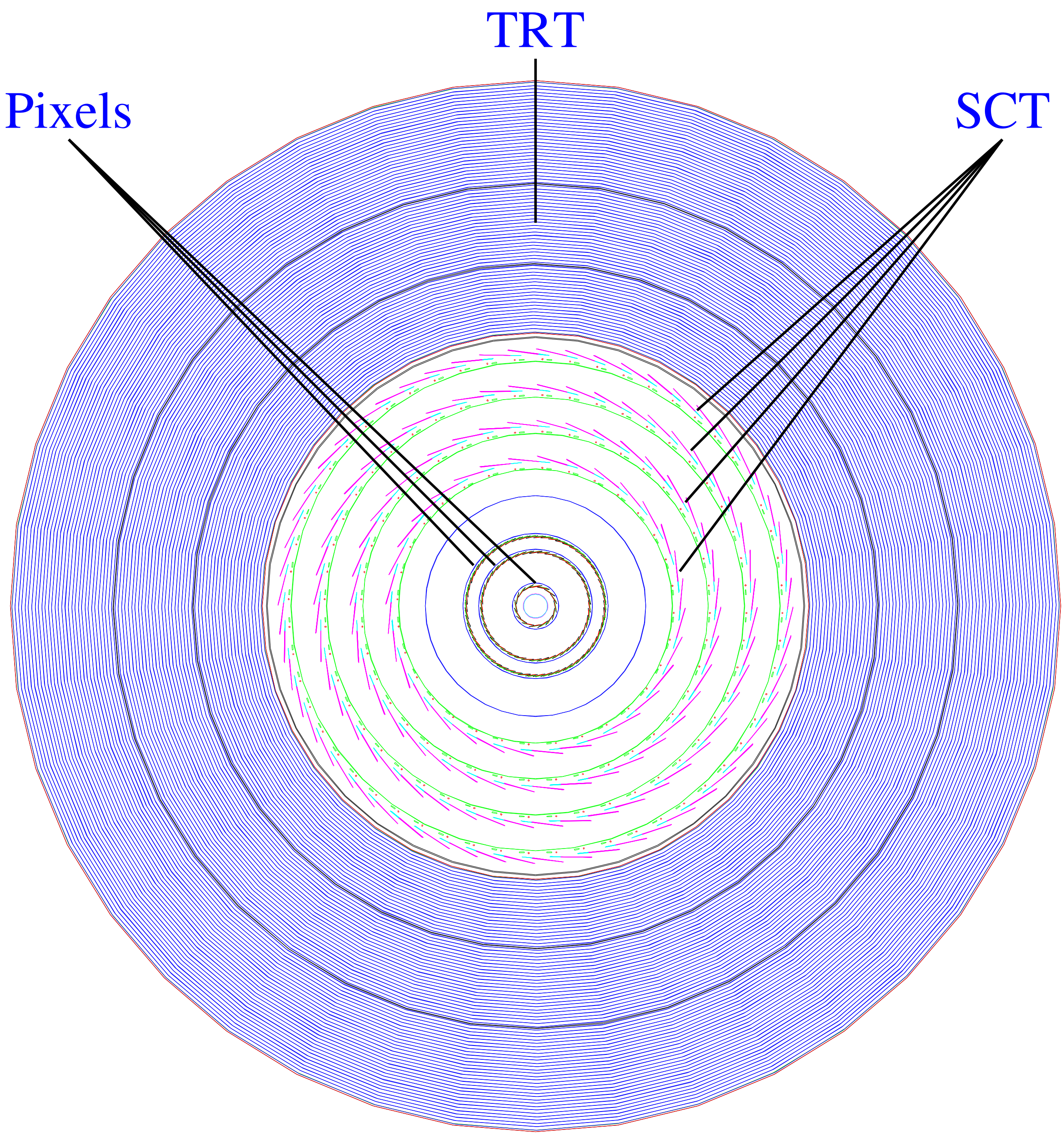
26 February 1998

The Inner Detector Simulation Software

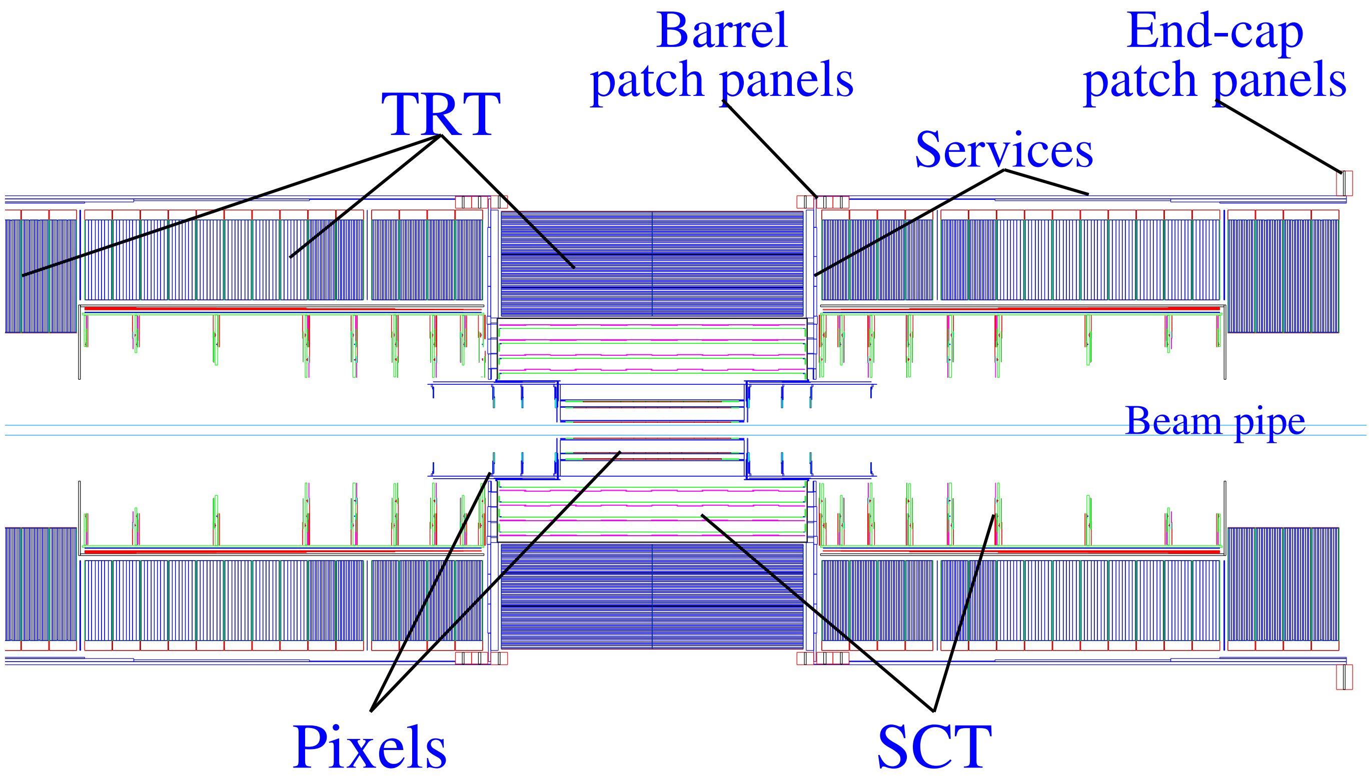
Very few changes since 97_3 release (used for Performance Studies for Inner Detector TDR)

However, some changes are in progress, others are being planned.

ATLAS Inner Detector Simulation



ATLAS Inner Detector Simulation



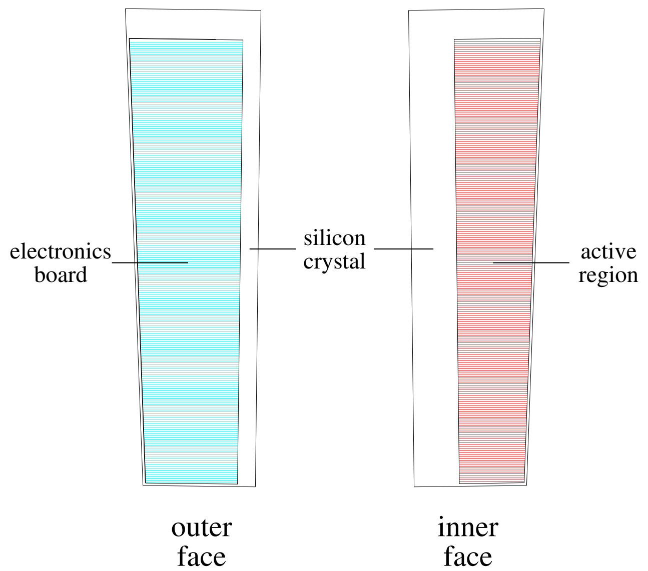
Pixel Geometry Simulation

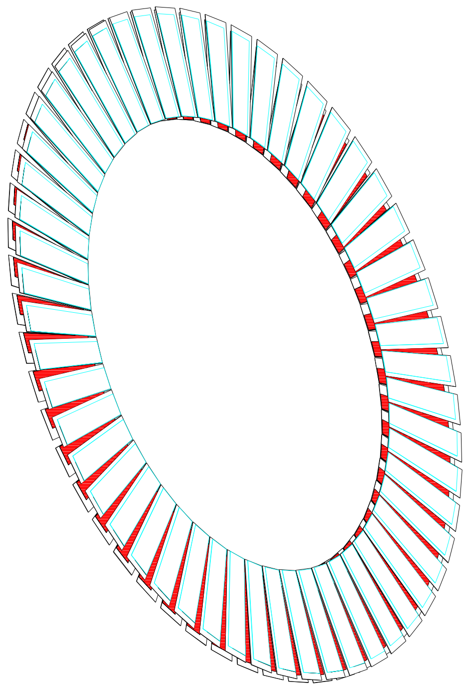
Barrel:

- rectangular crystals arranged on 3 barrel layers
- modules tilted in ϕ:
  - provide active overlap in ϕ
  - use Lorentz effect to maximise cluster width
- adjacent modules overlap in z by stagger in r

Endcaps:

- trapezoid-shaped crystals arranged in 4 wheels
- active overlap in r and ϕ

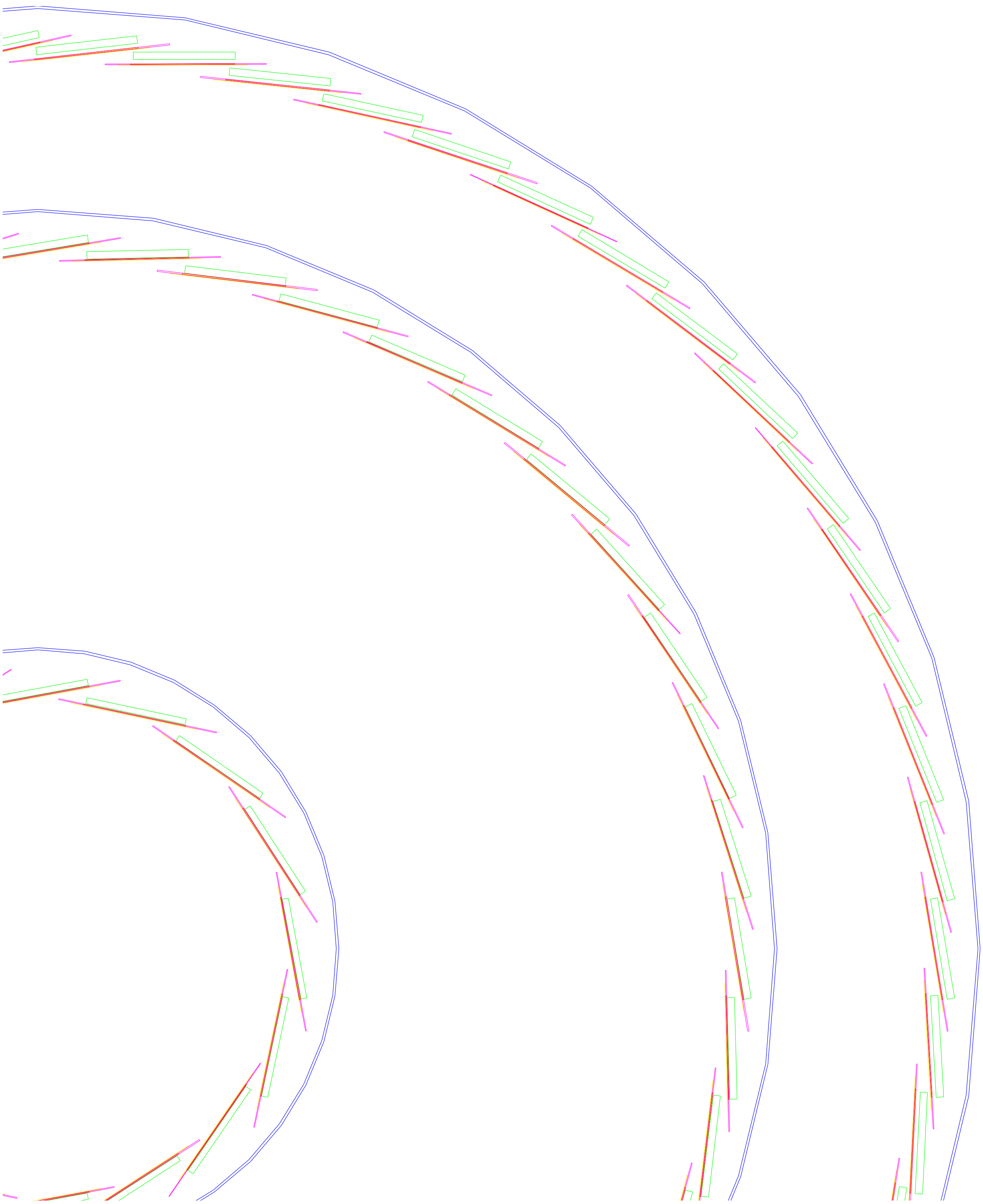




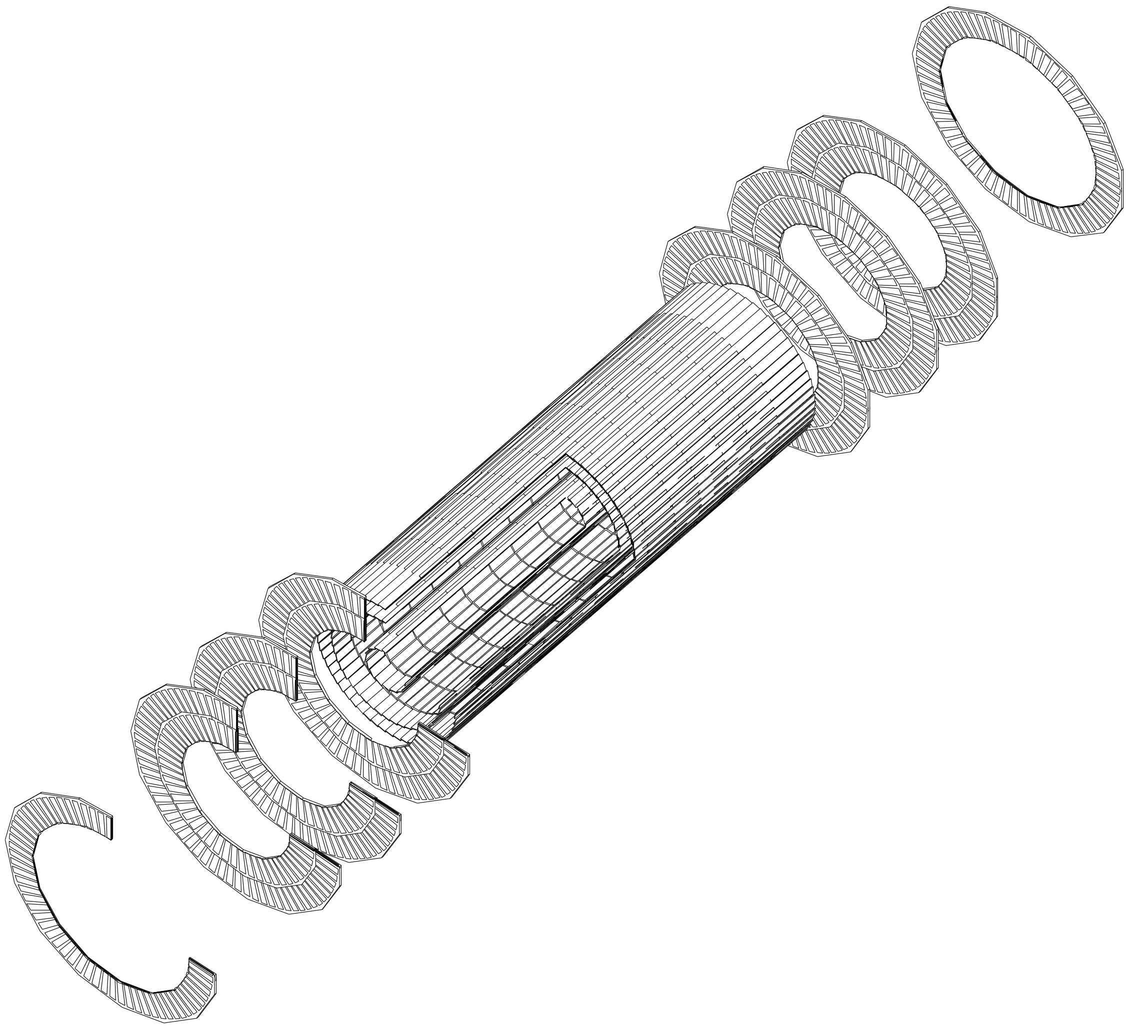
Services: support structures, cooling tubes, cables are simulated

- correct amounts (radiation lengths) of material
- geometrical approximations

Pixel Barrel Simulation



Pixel Detector Simulation



Pixel Digitisation Simulation

Simulation Includes:

- Lorentz angle effect
- Diffusion
- Threshhold fluctuation
- Dead channels
- Electronics noise

Changes in pixel simulation since ID TDR

Current production version:

- tilt angle changed sign
- services of b-layer routed along beampipe
- material increased from minimum-range to mid-range of engineer's estimate (1.56%X0 $\rightarrow$ 1.63%X0)
- cables moved to be in front of sensors (more realistic)

Test version (not yet released): corresponds to description of pixel system in ID TDR (volume 2)

- detailed note available on web: http://marpix1.in2p3.fr/Pixel/dice/intro_dice.html
- needs thorough testing with reconstruction
- further changes probable as pixel design evolves

SCT Geometry Simulation

Barrel:

- rectangular modules arranged in 4 layers from R=30cm to R=52cm
- active half-length = 74.5 cm (12 modules in z)
- modules tilted 10° in ϕ:
  - provide active overlap in ϕ
  - use Lorentz effect to minimise cluster width
- adjacent modules overlap in z by stagger in r

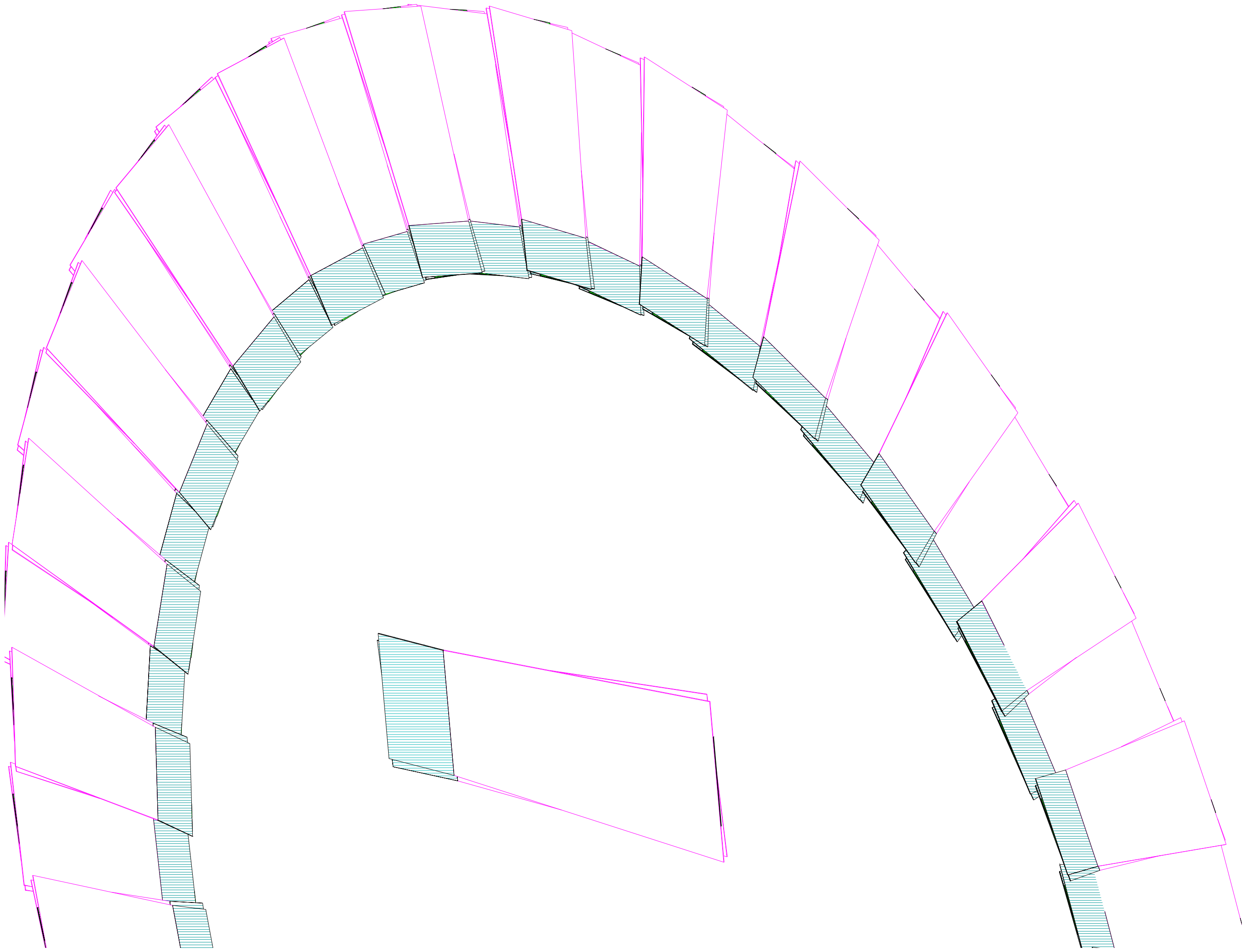
Endcaps:

- trapezoid-shaped crystals
- 9 discs from z=83.5cm to z=277.8cm
- radial coverage beginning from 26.0cm –43.8cm out to 56.0cm
- active overlap in r and ϕ

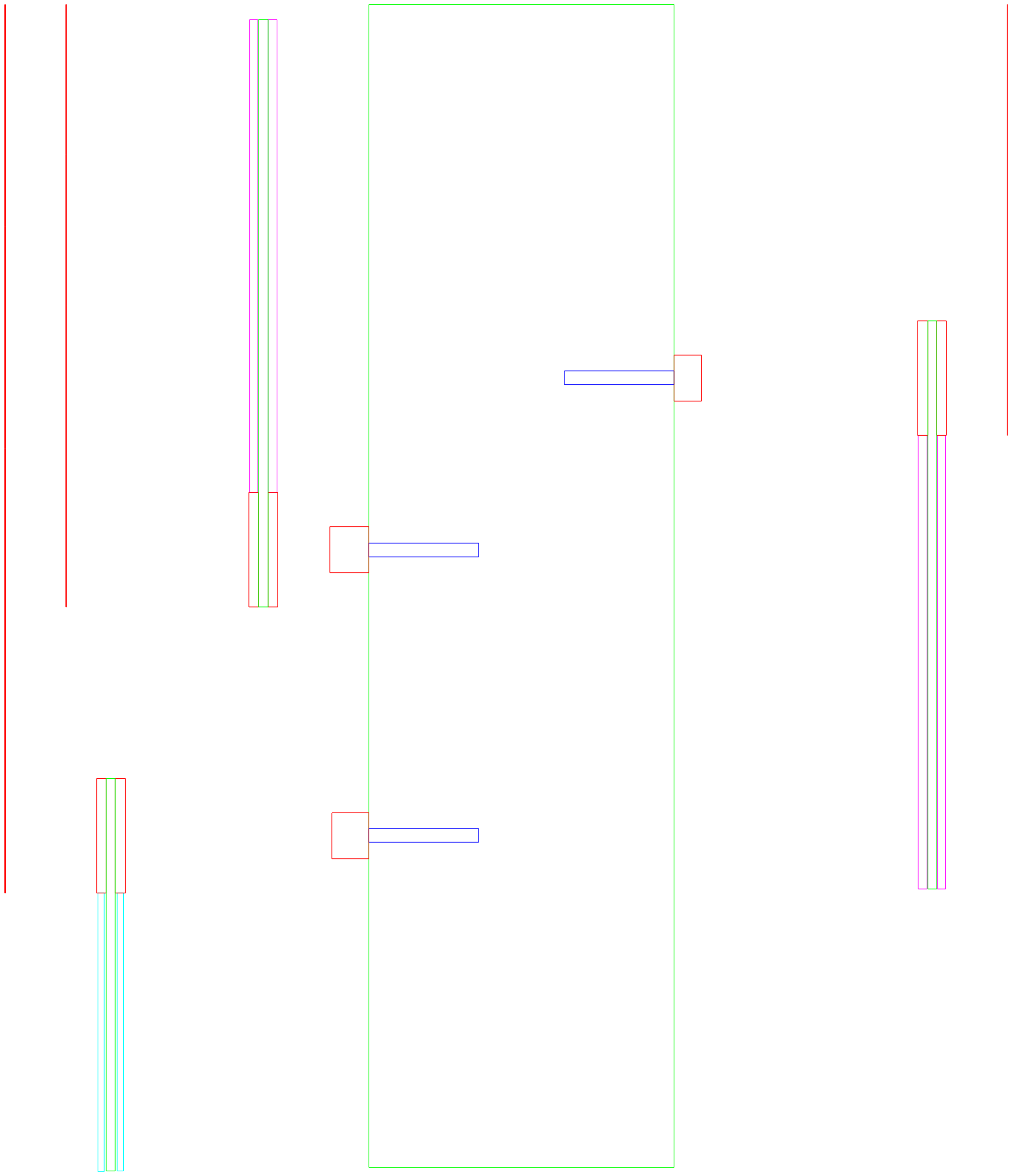
General:

- stereo angle achieved by rotation of modules
- support and services realistically represented

Forward SCT: Modules and Rings

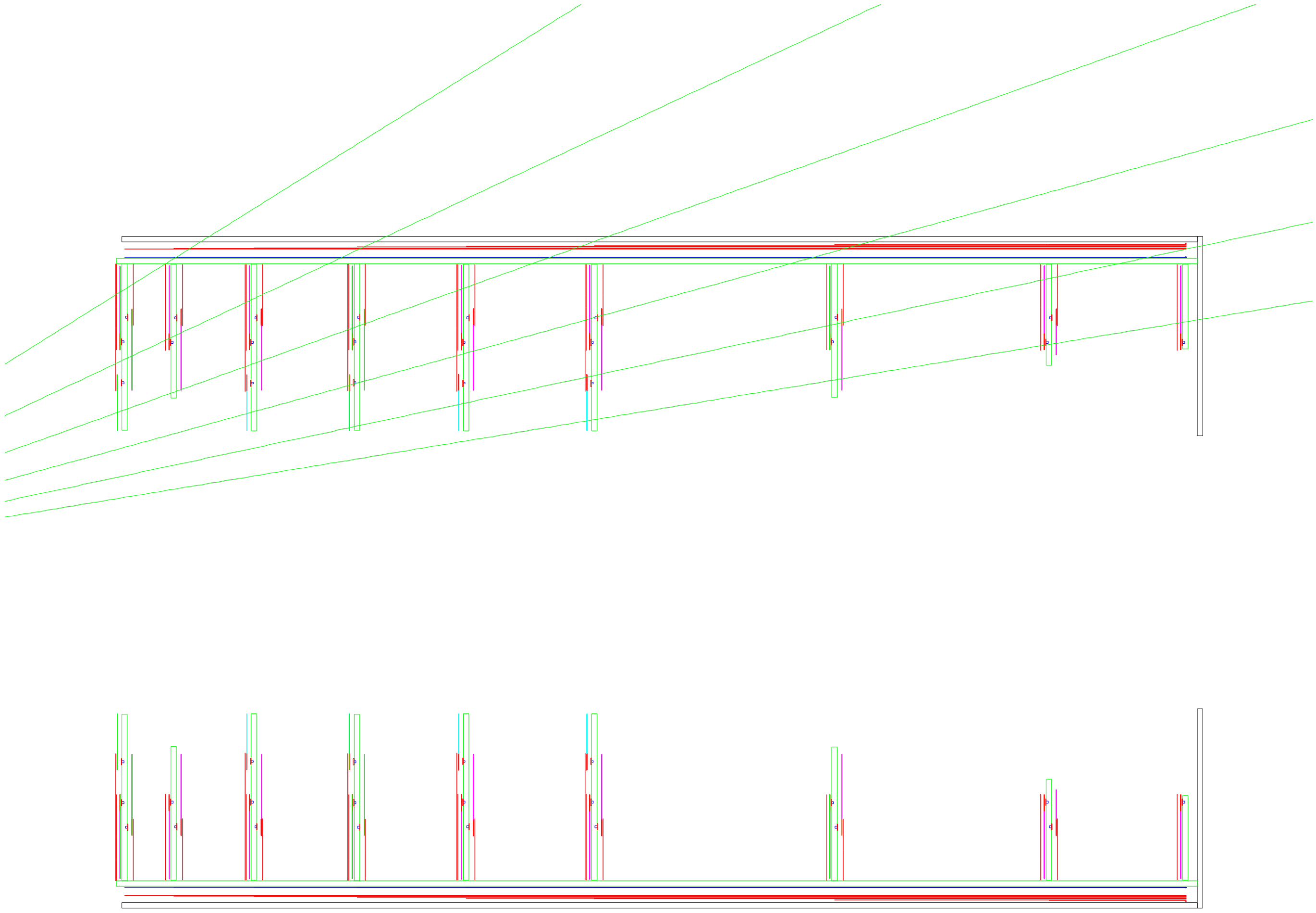


Forward SCT Wheel: Support and Services



Cross-section of Wheel 5
(horizontal axis expanded by factor 8)

Forward SCT Layout in Simulation



SCT Digitisation Simulation

Simulation Includes:

- Lorentz angle effect
- Diffusion
- Electronics crosstalk
- Dead channels
- Electronics noise
- Digital readout available

Changes in sct simulation since ID TDR

Current production version:

- changed inner ring from GaAs to Silicon
- adjusted barrel support cylinder length, construction of ends of cylinders and thickness of insulation to match TDR estimates
- routed services for first four disks through first TRT gap (were through second)

Changes foreseen in (not-too-distant) future:

- update material everywhere to latest engineers' estimates
- update barrel module design

TRT Geometry Simulation

Each straw is individual GEANT tube:

- copper wire
- drift gas
- kapton wall
- 50 μm CO_2 cooling
- foam radiator

Barrel:

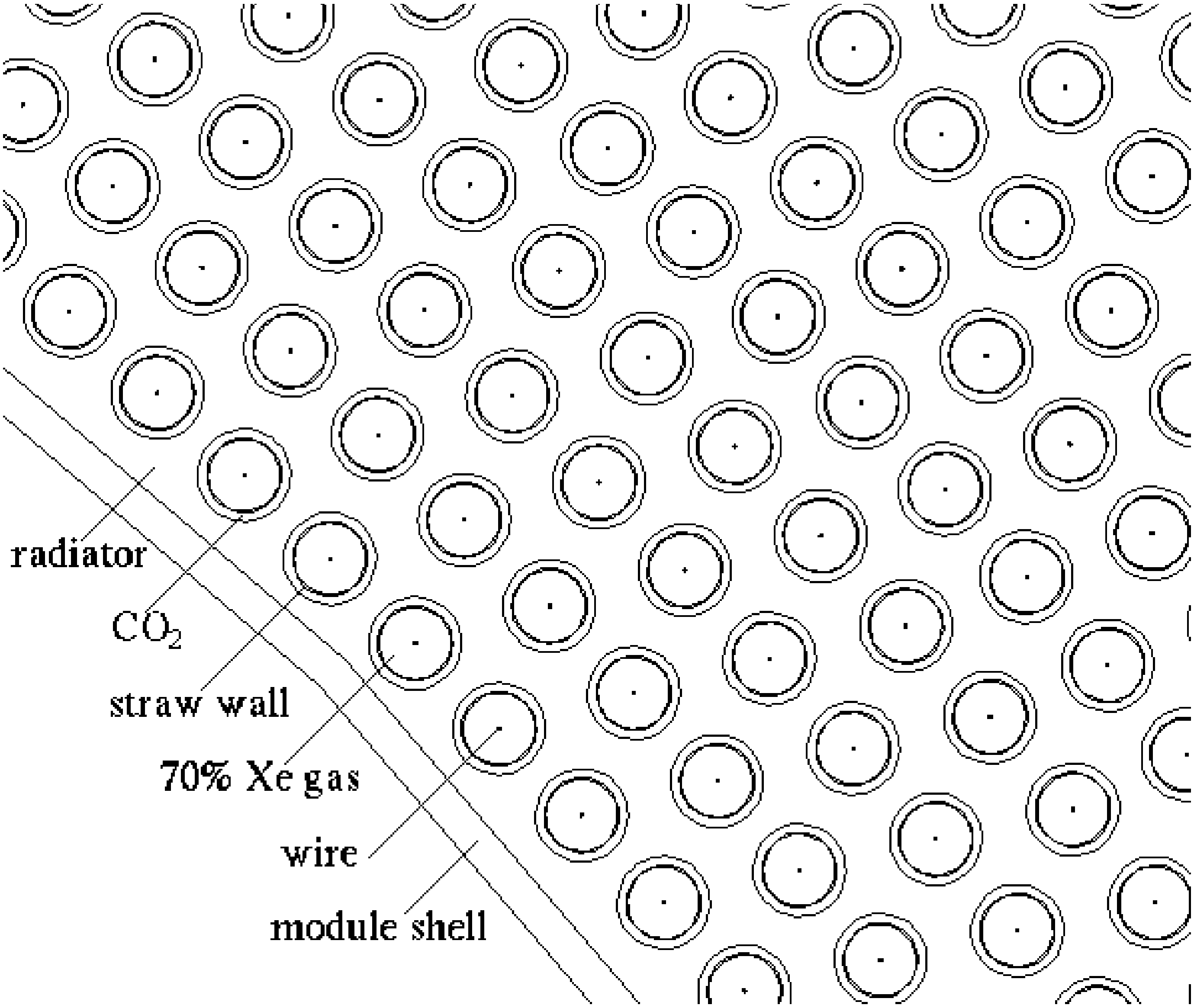
- 75 layers of straws lying parallel to beampipe
- each straw read out in two halves ($\pm$z)
- half-length of straws = 74cm
- inner radius 56cm
- outer radius 107cm

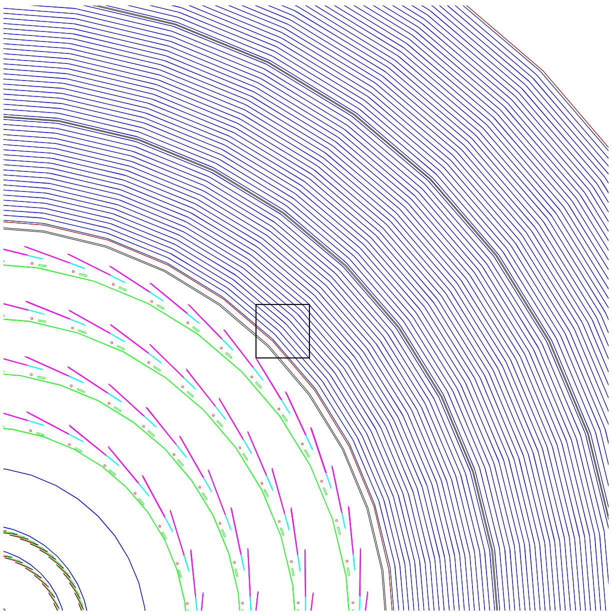
Endcaps:

- radial straws arranged in discs
- 14 "short-straw discs" 64cm $<$ r $<$ 103 cm
  83cm $<$ z $<$ 282 cm
- 4 "long-straw discs" 48cm $<$ r $<$ 103 cm
  282cm $<$ z $<$ 323 cm

The above is a somewhat simplified geometry. A more realistic geometry is available but not yet integrated with reconstruction.

TRT Geometry Simulation





TRT Digitisation Simulation

- Photo Absorption Ionization (PAI) Model used to simulate ionization (not standard GEANT) $\rightarrow$ reproduces test beam results
- radiator structure and photon absorption cross sections $\rightarrow$ probability of transition radiation photons
- detailed electronics response simulation

Released version of TRT simulation unchanged since ID TDR

Test version (private):

- changes in geometry to improve TR simulation
- changes in digitisation to better simulate “hardware parameters”, e.g. readout TDC bin size, electronics dead time, etc.
- minor bugfixes
- minor improvements in occupancy calculations

Simulation of Services between subdetectors

Changes for 97_6 version:

- updated material to match ID TDR
- included support rails (in calorimeter code)

Changes foreseen for future:

- update material to latest engineers' estimates (April 1998)
- include squirrel cage - ?
  - several very different designs, none are "official"
  - effect on calorimeters unclear, needs study

Outline

- Status of and Plans for ID Pattern Recognition packages
  - iPatRec
  - xKalman
  - PIXLRec
  - Astra
- Moving to OO...

iPatRec:

Pattern Recognition starting in the SCT layers

Algorithm:

- make lists of space points in precision layers
- make 'track candidates' from space points in outer sct layers + beamspot
- continue making track candidates with lower-radii space points (except b layer)
- build track with rest of precision layer hits
- add TRT hits to track
- now look for associations in b layer
- tracks are refitted at every stage

iPatRec: History and Status

- Official version unchanged since 97_3 release
  - used for Inner Detector TDR performance studies
  - 20'000 lines fortran + 20'000 lines C++.
    C++ is mixture of procedural and computer-generated OO (via agetocxx)
  - slow execution time for jets (30 seconds per 2-jet event)
  - very slow with pileup (12 minutes per 2-jet event)
  - works in slug framework, on alpha and hp

iPatRec: Recent Developments

- September 1997: geometry description and decoder converted to OO
  - gained factor 50 speed in decoding = factor 2 in operation
  - included database of dead modules and read-out chips
  - also module-level misalignment capability
- November 1997: fortran code for extending tracks into the TRT replaced by C++
  - reproduced TDR b-jet analysis results
  - this version used for 6-precision-layer studies (INDET-188)
- Since then: track finder rewritten in C++: reduction in wasted cpu due to recalculations $\rightarrow$ 15 seconds/2-jet event with pileup
  - this version still being tested
  - needs some cleanup before can be released
  - in principle can handle realistic magnetic field - needs testing
- Track fitter being extended to be useful to muon system also

iPatRec: Future

After release of current version:

- Finish conversion to OO by summer 1998
- integrate with other pattern recognition packages (common interfaces, common classes)
- reviews, documentation
- make it work in ARVE as soon as that is possible

People

- A.Poppleton - main designer/developer
- R.Clifft - testing and C++ design
- P.Hendriks - extension to muon system
- J.Loken - TRT code
- A.Pickford - testing
- G.Stavropoulos - fortran $\rightarrow$ procedural C++, extension to muon system
- M.Tadel - testing

xKalman:

Pattern Recognition starting in the TRT

Algorithm:

- pattern recogntion started in TRT using a histogramming method → "initial helix"
- initial helices used for local track search (via kalman filter) in precision layers
- tracks found in precision layers extrapolated back into TRT

xKalman: History and Status

- used for Inner Detector TDR performance studies
- current release is entirely fortran
- works in slug and atlsim frameworks
- user guide and algorithm description available on web

xKalman stable since ID TDR (April 1997)

xKalman++:

xKalman being converted to OO C++:

- xKalman track reconstruction code converted to OO, work starting on conversion of road-definition code (I.Gavrilenko)
  - runs standalone, one event at a time
  - output stored in banks
- work going on in parallel (S.Almehed, S.Qian) on OO kalman filters

Future work:

- Finish conversion to OO
- put into a framework (atlsim? slug?) for high-statistics testing
- test - reproduce results of fortran version
- integrate with other pattern recognition packages (common interfaces, common classes)
- make it work in ARVE as soon as that is possible
- reviews, documentation

PIXLRec:

Pattern Recognition starting in the pixel layers

T.Mouthuy, A.Rozanov, L.Vacavant

Algorithm:

- define roads using primary vertex + points in innermost precision layer
- add points in successive precision layers
- add TRT information à la XKalman

PIXLRec: History and Status

- first released September 1996
- goal: to provide another pattern recognition algorithm $\rightarrow$ better understanding of ID performance
- exploits fine granularity of pixel detectors, especially inside jets
- mainly for precise tracking in small regions
- used for some ID TDR performance studies
- current release is entirely fortran
- works in slug and atlsim frameworks
- user guide and brief algorithm description available on web

Astra: an Object-Oriented Trackfinder

H.Drevermann, C.Linder, P.Luthaus

Algorithm:

- Use histogramming method in precision layers to 'filter' space points likely to belong to tracks
- Group these space points to reduce the combinatorial background - fast!
- Do combinatoric track finding on each group
- Depends on knowledge of the vertex, however vertex can be determined with the filter

Astra: History and Status

- started “conventionally” in 1995 with goal of providing pattern recognition starting from the pixel layers
- later decided to go object-oriented
- only a track finder - no code for getting detector constants or data, or for track fitting
- being developed in Atlantis (event display) - one event at a time - only low-statistics testing so far

Astra: Future

- test the algorithm with statistics
  - run in slug, in iPatRec “frame”
  - tune parameters for different types of physics
- go through design review (document ready now), then code review
- Use TRT information (further future)

Toward modular pattern recognition code

Goal for ID OO Pattern Recognitions: only the pattern recognition itself will be done by package-specific code.

Working meetings in December and January →

- what should be handled by common code?
- prioritisation - what needs to be made common first?

The following tasks will be handled by common classes:

1. interfaces to databases, magnetic fields - high priority
   - work underway in those domains
   - we will incorporate that work as it becomes ready
2. clustering - high priority
   - needs to be made common ASAP - fortran code exists, must be converted to OO C++
3. track fitting, other “utilities” - can wait a while
4. output - i.e. track class - high priority

Other code available for Inner Detector

- xConver,XHouRec (U.Egede, D.Barberis) - conversion reconstruction
- id_cuts.g (S.Haywood) - standard cuts to select good tracks
- bfsmear.for (D.Rousseau) - additional smearing to high-pT tracks from solenoidal B-field
- calorimeter brem fits (I.Gavrilenko) - use calorimeter position as extra hit in track fit
- presol.f/geom.f (A.Poppleton) - produce analytic curves of the muon track resolution

Name in parentheses = name of author/responsible.

For more information see the Inner Detector Layout and Performance Studies web page.

SUMMARY

- Inner Detector software stable since early 1997
- Some changes to come:
  - GEANT3 simulation changes should be minor (except possibly in pixel system)
  - improvements/new features expected soon in iPatRec
  - pattern recognitions “going OO” - needs to be done in an organised way