Alignment of LHCb Vertex Detector

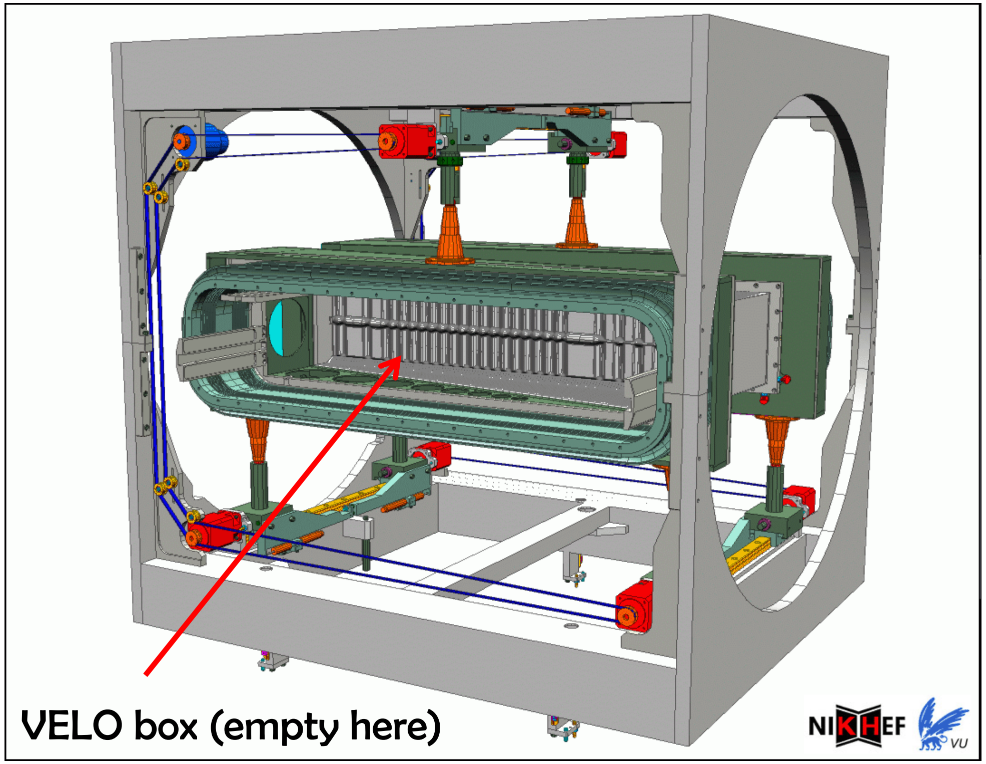
1. VELO alignment context
2. Software alignment strategy
3. Status & plans

1. VELO alignment context

2. Software alignment strategy

3. Status & plans

⇒ Vertex Detector (aka VELO) is a moving detector:



VELO box (empty here)

→ Divided into two boxes containing 21 modules each. A module is a pair of two sensors (r,ϕ) bonded together (*see S.Blusk introduction for more details*).

→ During LHC beam injection, each box is retracted by 3cm from its nominal position.

→ Then the boxes are moved back close to the beam, and data taking starts.

⇒ VELO position matters:

↳ LHCb first level trigger (Vertex Trigger) relies on a good VELO positioning (LHCb note 2005-056).

↳ VELO alignment has thus to be checked after each fill (*at least look at the residuals*), and correction might be necessary.

↳ Alignment should be reasonably fast, as for the moment we don't know if we will need to align nothing or the whole VELO on a fill-to-fill basis…

A precise and fast algorithm for VELO software alignment is thus necessary

⇒ Software alignment is just a part of the story:

❶ Precision Mechanical Assembly

❷ System Metrology & Initial Alignment ⇒ Alignment Challenge and Detector Calibration

❸ Software Alignment & Alignment Monitoring ⇒ Checking the residuals after each fill, then perform a new alignment if necessary.

❹ Software Alignment for offline data processing ⇒ Final 'best precision' alignment.

⇒ About the mechanical accuracies:

⇒ **Box** positioning estimates:

→ 50 microns accuracies for translations, 50 μrad for rotations.

→ Position reproducibility of 10 microns.

⇒ Expected accuracies for **modules** and **sensors**:

→ Sensor are positioned on a same module with ~10 microns accuracy (*values measured on the first production modules*)

→ Module will be positioned within a box with ~20 microns accuracy

⇒ Temperature and vacuum effects still have to be investigated

1. VELO alignment context

2. Software alignment strategy

3. Status & plans

⇒ How to define a strategy for software alignment ?

→ Try to be conservative: for the moment we don't know if the modules are moving when boxes are retracted, *so we need to include module alignment.*

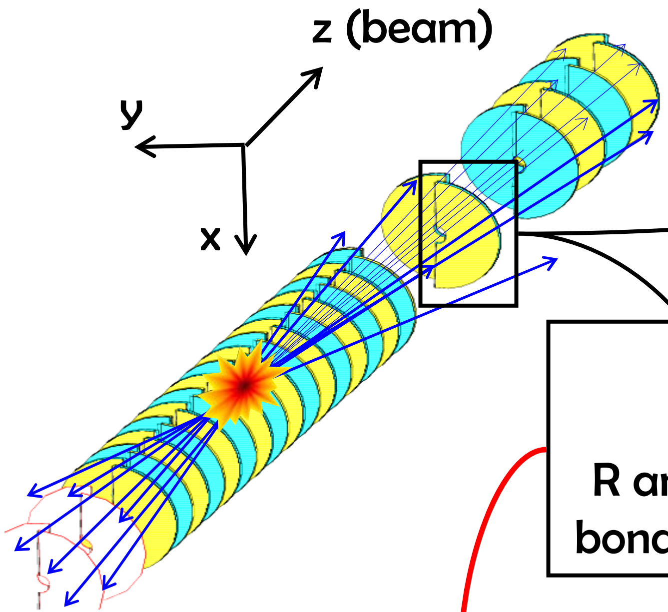
→ Try to be flexible: but if we don't need it, we should be able to turn it off without any problems, *so we need to separate the different alignment steps.*

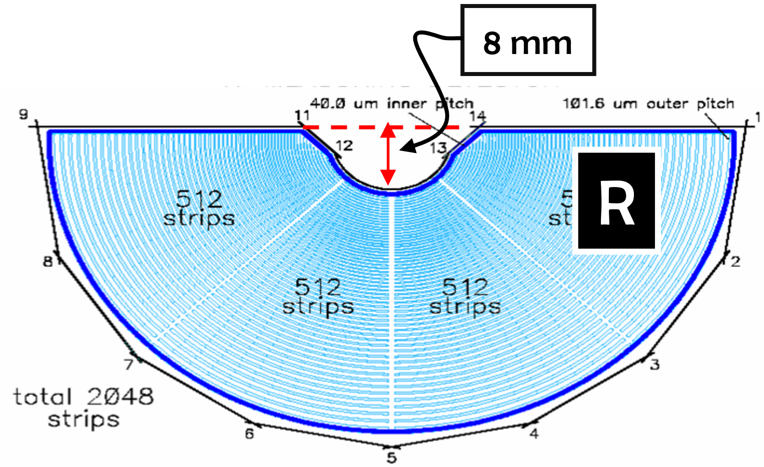
→ Try to be fast and robust: we have to be able to process the alignment in few minutes, constrained global fit method (*via Millepede*) seems a good candidate.

→ Linearize the problem: to use a global technique, we need to be able convert VELO (r,ϕ) information into a linear (X,Y) expression. Feasible as R and ϕ sensors are bonded together within a module.

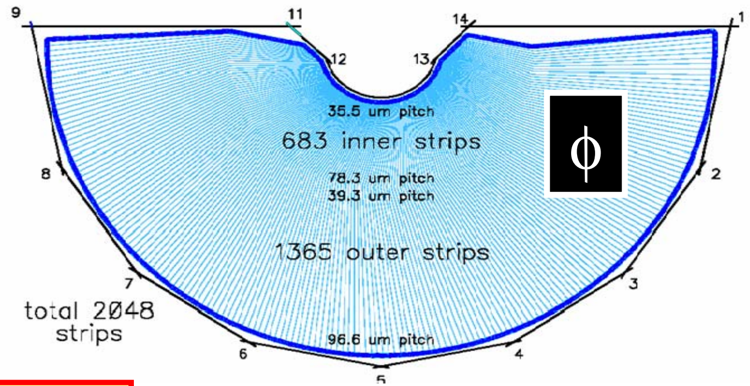
UNIVERSITY of GLASGOW

⇒ Vertex Module Definition:





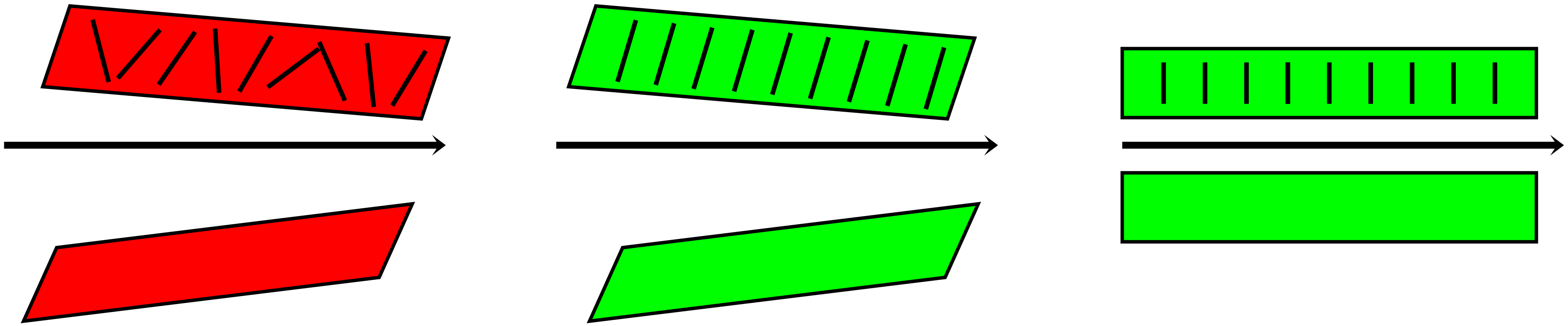
Module
=
R and ϕ sensors bonded together



First approx:
module = rigid body

252 modules parameters + 12 box parameters

⇒ The proposed method:



Step 0

Misaligned VELO

→

Step 1

Internally-aligned VELO

Millepede applied on tracks (**classic** & **halo**) in the two boxes

→

Step 2

Aligned VELO

Align the boxes using Millepede again on primary vertices, overlapping tracks,...

⇒ STEP1, along with preliminary results, is detailed in note **LHCb-2005-101** . STEP2 is described here (*note in preparation*):

http://ppewww.ph.gla.ac.uk/LHCb/VeloAlign/VeloApplication.html

1. VELO alignment context
2. Software alignment strategy
3. Status & plans

⇒ Alignment algorithm *flow* (integrated within LHCb software):

Retrieve Alignment Conditions → Retrieve constants from CondDB

↓

initialize() → All the alignment job properties (*modules to align, constraints to apply,…*) are set via jobOptions files

↓

InputTrack

↓

execute() → Each track is selected and transformed into a 'Millepede-friendly' track ((R/ϕ)→space-points conversion), using a TrackStore class.

↓

AlignTrack Container

↓

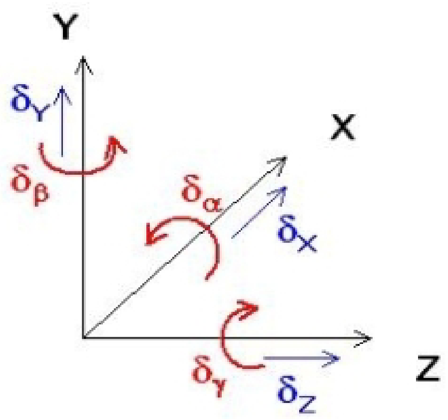
finalize() → The two alignment steps are performed using a Millepede C++ tool.

↓

Update Alignment Conditions → Check and if OK update CondDB

⇒ Methodology for the tests:

→ 200 runs of 2000 min. bias events were passed trough LHCb software with the following misalignments scales (*all 6 degrees of freedom are taken into account at each level*):

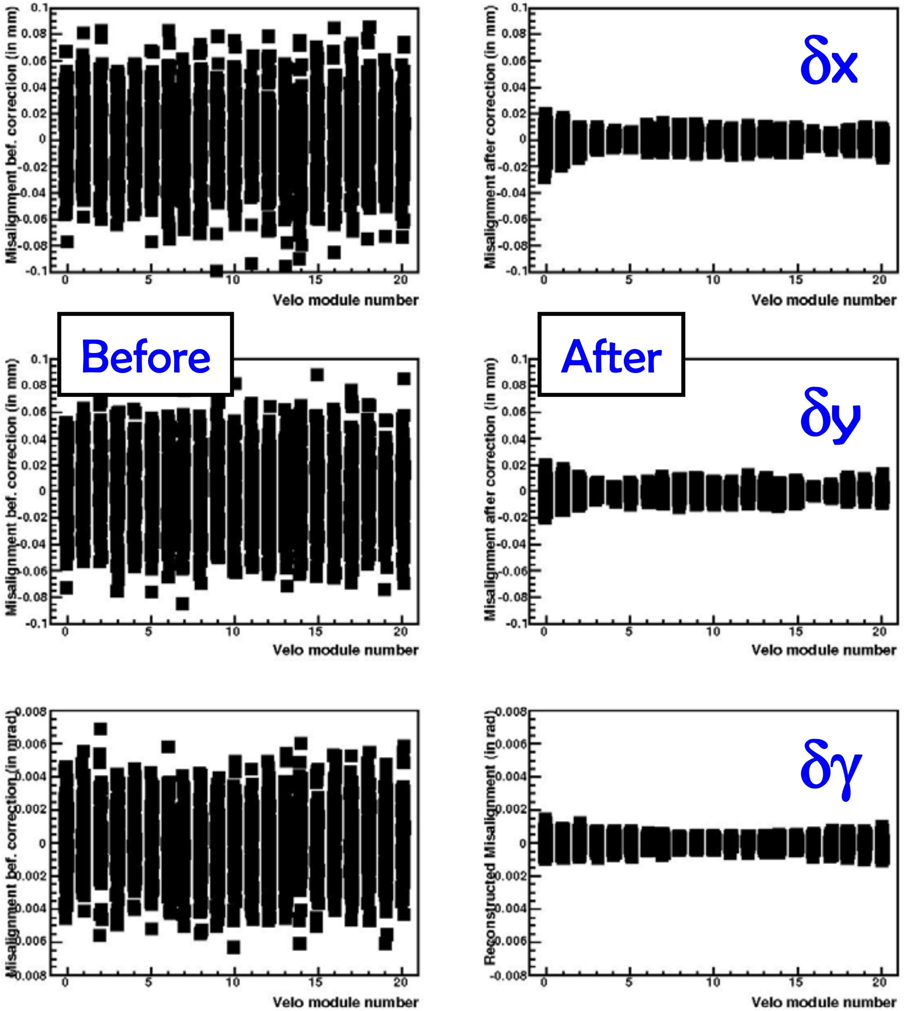


	Translations (in μm) (δx, δy, δz)	Rotations (in mrad) (δα, δβ, δγ)
Module	30	2
Box	100	2

→ Misaligned events are produced using alignment framework (*see J. Palacios talk*).

→ No momentum cut applied for track selection *(try to rely on VELO information only)*

⇒ STEP1: Module alignment:



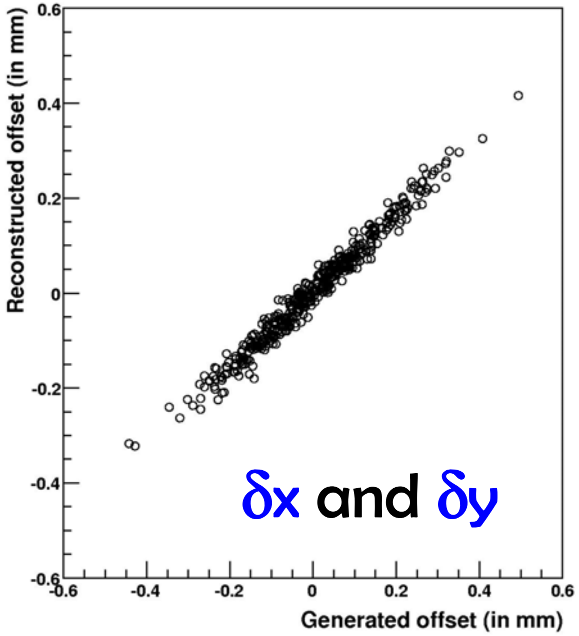
→ Only the 3 major DOFs are well corrected, sensitivity to other DOFs is smaller, but this is expected.

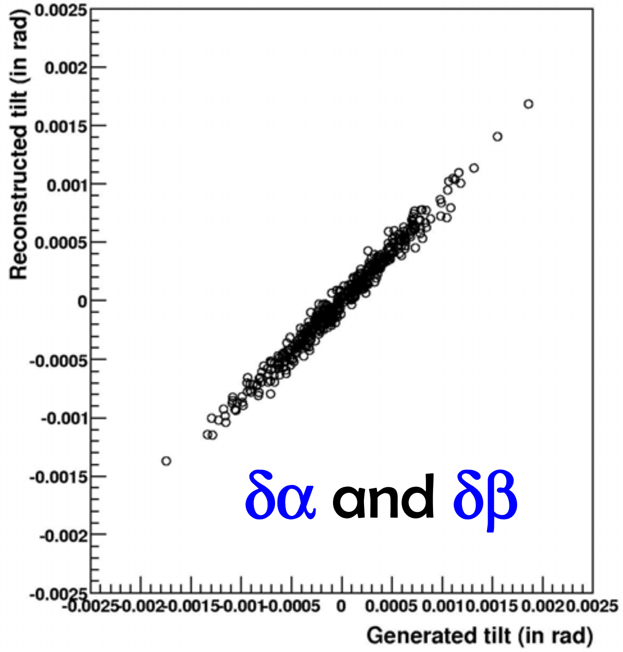
→ Resolution on alignment constants (with few 10000s tracks) are 3.8 μm (δx and δy) and 0.3 mrad (δγ)

→ Algorithm is fast (few minutes on a single CPU)

→ Improvement expected with the use of halo tracks.

⇒ STEP2: Box alignment *(with primary vertices)*:



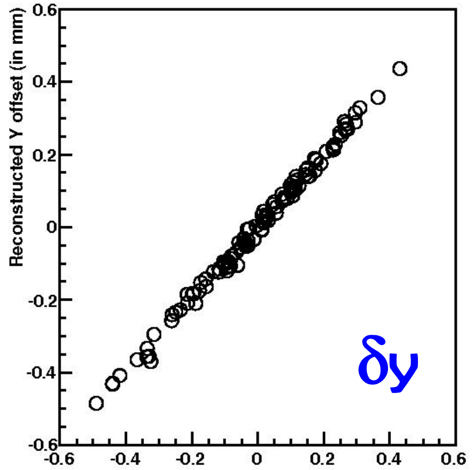


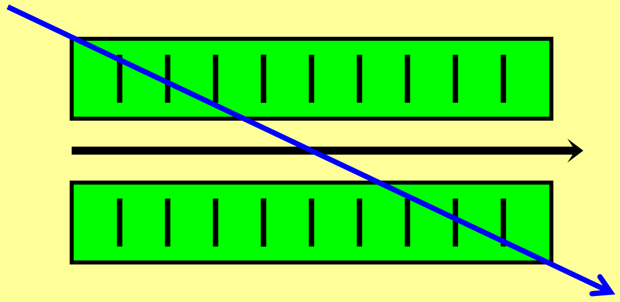
→ Use Millepede again, but local fit is now a PV fit using corrected track parameters…

→ Resolution obtained is still not satisfying (~30 µm for offsets, ~90 µrad for tilts), but give the position of each box w.r.t. the beam.

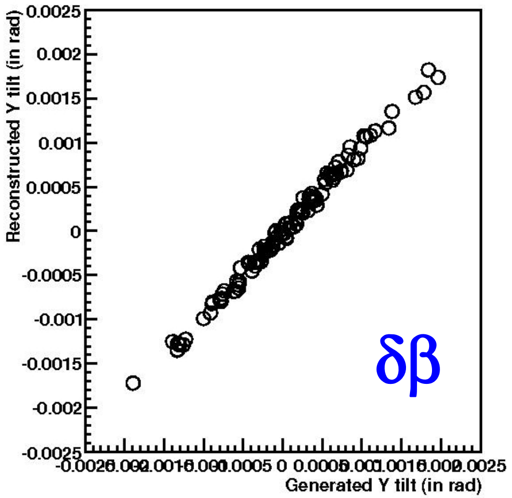
→ Still investigating possible improvements here….

⇒ STEP2: Box alignment *(with overlapping tracks)*:





Overlap track
=
Classic (or Halo) track with at least one space-point on the other side



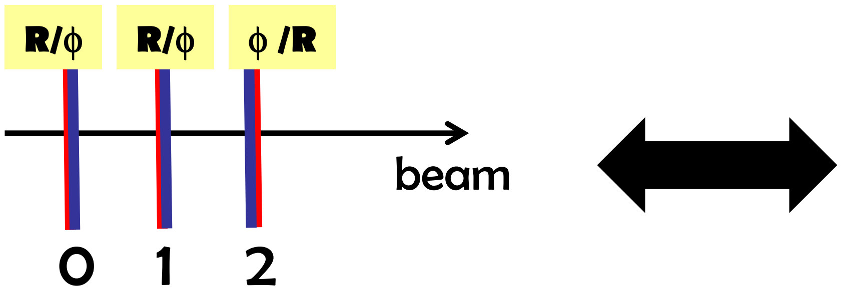
→ Difficult to obtain in the VELO (lot of work necessary on PR), but possible…

→ Preliminary results (particle gun events) are encouraging, **~10 μm** for offsets and **~40 μrad** for tilts could be obtained, with very few clean tracks.

→ But will not give VELO position w.r.t. the beam…

⇒ Alignment Challenge and Detector Calibration (ACDC):

→ Test beam using a 3 modules setup (*aka ACDC2*) in August.

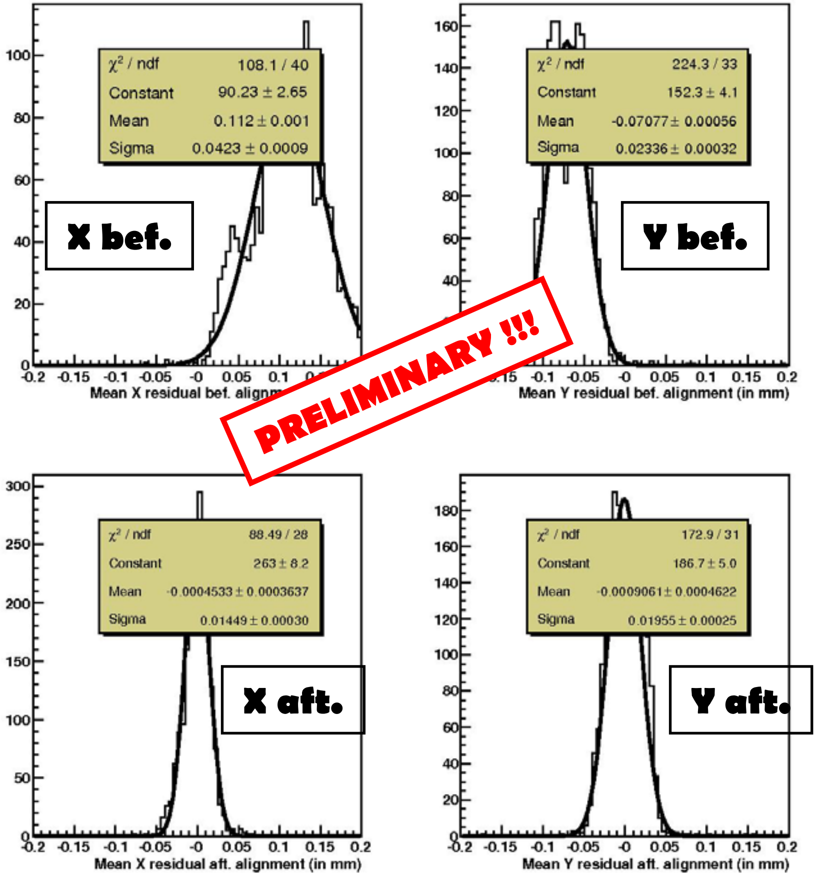


→ Just enough to get tracks, and then residuals, and then alignment...

→ Alignment performed using ~8000 tracks (*2% of the avail. dataset*)with 0° incidence angle. Angled tracks will be included in the future.

→ In parallel to the alignment process, a independent sample of 2000 tracks is collected. Space-points residuals before and after alignment are determined using this sample.

⇒ The first 'real' VELO alignment:

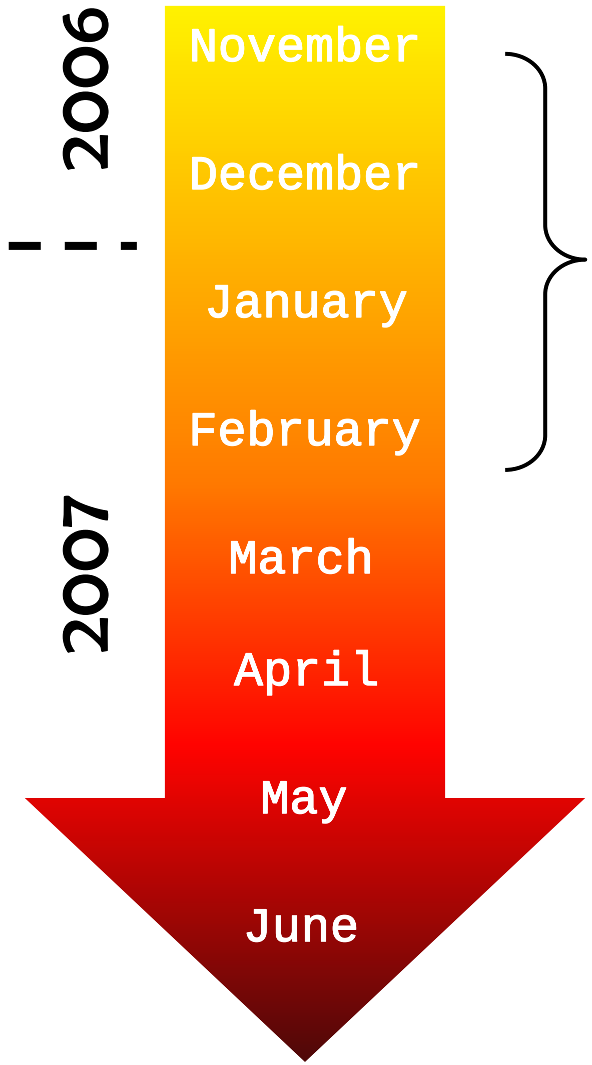


Space-points residuals for module 1

→ Mean value after alignment is close to zero, as expected, the code is doing his job…

→ Applying the corrections found at the pattern recognition level seems to improve the track quality.

→ Still a lot to understand here (*e.g. sensor to sensor misalignments*), but that looks promising!



⇒ Nov.06 : ACDC3: beam tests with more modules.

⇒ Nov.06 to Feb.07 : ACDC's analysis, try to fully understand the results, learn the lessons for the real experiment.

⇒ March/April 07 : The complete alignment software is available into LHCb official software, ready for first data.

⇒ Year 07 : Develop a method for the 'final-precision' alignment, taking sensor misalignments into account (*Kalman filter?*)

Backup Slides

UNIVERSITY of GLASGOW

→ Proposed solution:

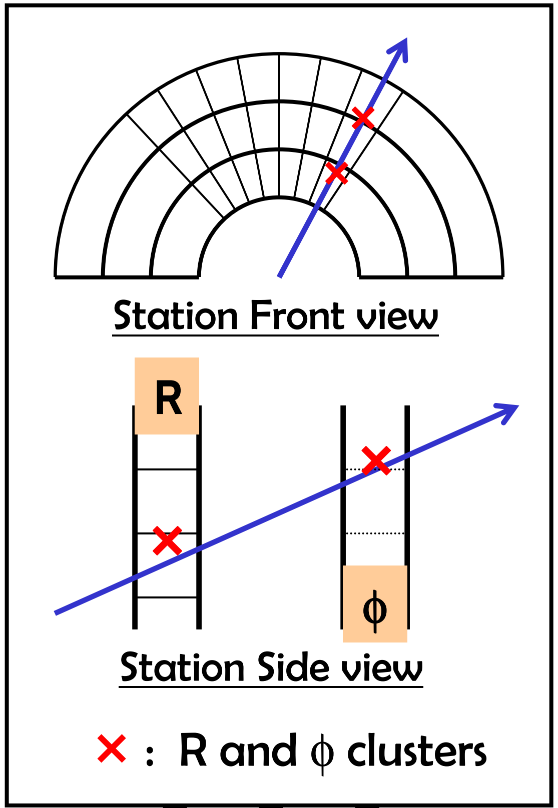
1. Take tracks and transform (R,ϕ) coordinates into (X,Y,Z) ones.

2. Precisely known parameters are $\phi(\phi_{sensor})$, $R(R_{sensor})$, Z_R, and Z_ϕ. Should we take Z_R or Z_ϕ for the Z coordinate?

3. Right figure describes why we choose Z_R.

4. Assuming this, we could obtain a precise (X,Y,Z) coordinate for each (R,ϕ) couple of clusters.

➡ **VELO is 'linearized'**



$$\phi(R_{sensor}) \approx \phi(\phi_{sensor})$$

$$R(R_{sensor}) \neq R(\phi_{sensor})$$

→ Step 2 algorithm: 'Millepede returns'

$$\begin{cases} X_i = m_x Z_i + X_o + \text{global params} \\ Y_i = m_y Z_i + Y_o + \text{global params} \end{cases}$$

Step 1: use coordinates in order to fit the tracks

Select events with at least one track in each part

$$\begin{cases} v_X = m^i_x v_Z + X^i_o + \text{global params} \\ v_Y = m^i_y v_Y + Y^i_o + \text{global params} \end{cases}$$

Step 2: use track parameters (*corrected acc. to step 1 results*) in order to fit the primary vertex