Algorithms and Software
for
Hardware Alignment Systems

Pedro Arce (CIEMAT)

LHC Detector Alignment Workshop

Outline

- The problem of optical alignment and how to solve it
- COCOA
- Use of COCOA
- Time and memory consumption

The problem

SIMULATION:

- **Error propagation:**
    - Calculate how much the errors of the calibrated parameters and of the measurements affect the errors of the parameters we want to measure

- **Redundancies:**
    - How much the errors change if some measurement disappears

- **Range:**
    - When a measurement will get out of the range of the measuring device if some objects move

The problem (II)

RECONSTRUCTION:

• Optical system takes measurements (2D sensors, 1D sensors, tiltmeters, distancemeters)

$\Rightarrow$ results are not what expected by extrapolating measured and calibrated parameters. **Why?**

- Wrong rotation / position of some objects
- Wrong internal calibration of some objects
  - wedge of a splitter
  - internal calibration of a distancemeter
  - deviation when traversing a sensor
  - ...

• **Software is the same for Simulation and Reconstruction**

- Only difference: for Simulation measurement is ideal, for Reconstruction measurement is real

How to solve it

➢ Get the equations of how each measurement depends on all these parameters

- positions, rotations and internal parameters

$M_1 = f_1(p_1, p_2, \ldots, p_m)$
$M_2 = f_2(p_1, p_2, \ldots, p_m)$
…
$M_n = f_n(p_1, p_2, \ldots, p_m)$

$M_1, \ldots, M_n$ = Measurements
$p_1, \ldots, p_m$ = parameters (known and unknown)
f_i are non linear equations

❖ You know the measurements and some calibrated parameters, you need to know the missing ones

⇒ Solve the system of equations: **Non-linear least squares fit**

- To solve a system of equations, you do not have to know the equations

How to solve it (II)

- **Only derivatives are needed**

⇒ **Get the derivatives with a numerical method**

- Reproduce a measurement with initial parameters (e.g. propagate a laser until the sensor)
- Move a parameter and see how the measurement value changes
- Repeat n times moving $1/2^i$, until it converges

Total CMS optical alignment system: 40000 parameters

⇒ **big and sparse matrices**

⇒ **sparse matrix library (meschach C library)**

COCOA

Cms Object-oriented Code for Optical Alignment

♦ General purpose software to simulate and reconstruct optical alignment systems composed of any combination of

laser, x-hair laser, source, lens, pinhole, mirror, plate splitter, cube splitter, rhomboid prism, optical square, sensor2D, sensor1D, COPS, distancemeter, distance target, tiltmeter, ‘user defined’

- Each object has internal parameters (planarity of a mirror, wedge between plates of a plate splitter, internal calibration of COPS...)
  - ´user defined´: you can tell COCOA how much light shifts and deviates in the ASCII file

➢ Reconstructs positions and angles of the objects from the measurement values

➢ Propagates the errors of the measurements and calibrations (including correlations)

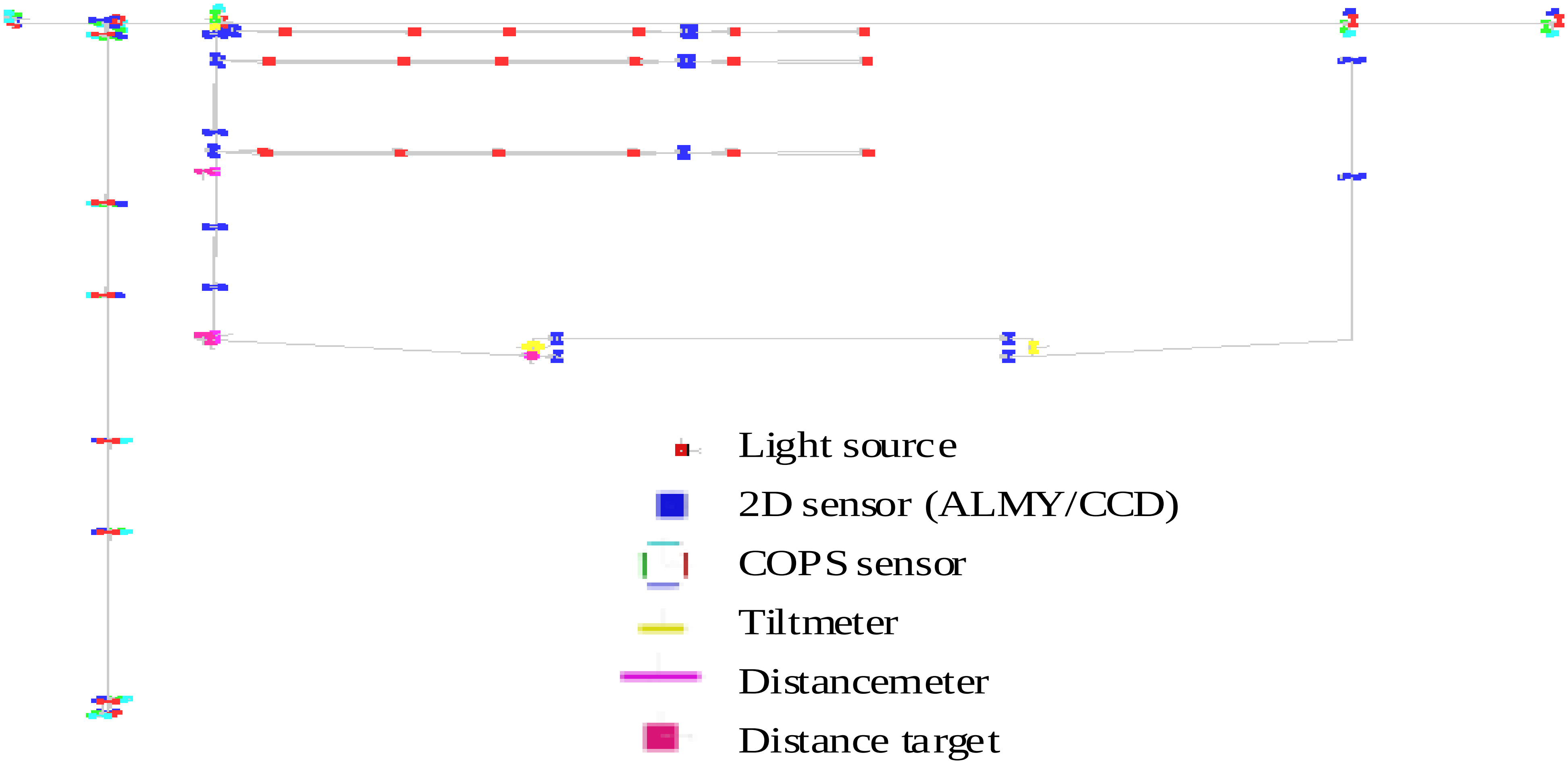
COCOA

- Interactive 3D view
  - VRML (Virtual Reality Modeling Language)
  - IGUANA (Interactive Graphics for User ANAlysis)
- Geometry
  - ASCII files
  - XML files (CMS format)
- Calibrated data can be read from Oracle DB to update data on geometry file
- Interface with DAQ measurements
  - ASCII files
  - ROOT tree files
- Output
  - ASCII file
  - Oracle DB

- Fully integrated with CMS software
  - Output interchangeable between COCOA and alignment with tracks sw

Full ISR setup in COCOA

(interactive 3D VRML view)



COCOA

COCOA

- **Documentation:**
  - Primer
  - User's Guide
  - Advanced User's Guide
  - Two examples explained with detail
  - dOxygen reference manual

COCOA

- **How it works:**
  - Describe the system in an input ASCII file
    - Also from an XML file
  - Select which parameters are unknown and which are known
  - For the known one write the values
    - They can also be read from an Oracle DB
  - Input the measurements
    - They can also be read from an ASCII file or a ROOT tree

➢ COCOA provides **best values for unknown parameters** (positions/rotations/internal parameters) compatible with measurements and **propagate the errors** from the measurements and the known parameters to the know and unknown parameters

➢ Also correct known parameters if current values do not provide a good fit

An example input file

```
// system composed of one laser, one periscope that
holds a plate splitter and a mirror and two 2D
sensors.

GLOBAL_OPTIONS
report_verbose 2
save_matrices 0
length_error_dimension 2
angle_error_dimension 2

PARAMETERS
pos_laser 0
posZ_periscope 1
posZ_sensor 1.1
err_pos 100
err_ang 100
prec_sens2D 5

SYSTEM_TREE_DESCRIPTION
object system laser periscope 2 sensor2D
object periscope plate_splitter mirror
```

```
SYSTEM_TREE_DATA
system s
 laser laser // this is the laser
  centre
   X  pos_laser 1000  unk
   Y  pos_laser 1000  unk
   Z  pos_laser 0.    fix
  angles
   X  0  err_ang  unk
   Y  0  err_ang  unk
   Z  0  err_ang  cal
 periscope peri
  centre
   X  0  err_pos  cal
   Y  0.25  err_pos  cal
   Z  posZ_periscope err_pos  cal
  angles
   X  0  err_ang  cal
   Y  0  err_ang  cal
   Z  0  err_ang  cal
```

```
plate_splitter spli
   ENTRY {
     length shiftX 0. 0. fix
     length shiftY 10. 0. fix
     angle wedgeX 0.0001 10 cal
     angle wedgeY 0.0001 10 cal
}
   centre
     X   0   err_pos  cal
     Y -0.25 err_pos  cal
     Z   0.  0.       cal
   angles
     X   0   err_ang  cal
     Y   0   err_ang  cal
     Z   0   err_ang  cal
 mirror mirr
   ENTRY {
     none planarity 0.1 0. cal
     }
   centre
     X   0   err_pos  cal
     Y 0.25  err_pos  cal
     Z   0.  err_pos  cal
   angles
     X   0  err_ang  cal
     Y   0  err_ang  cal
     Z   0  err_ang  cal
```

```
// now the two sensors
 sensor2D sens1
   centre
     X   0  err_pos  cal
     Y   0  err_pos  cal
     Z posZ_sensor err_pos  cal
   angles
     X   0  err_ang  cal
     Y   0  err_ang  cal
     Z   0  err_ang  cal
 sensor2D sens2
  centre
     X   0  err_pos  cal
     Y  0.5 err_pos  cal
     Z   0  err_pos  cal
   angles
     X   0  err_ang  cal
     Y   0  err_ang  cal
     Z   0  err_ang  cal

MEASUREMENTS
SENSOR2D
 s/laser & s/peri/spli:T & s/sens1
 H  0.1   prec_sens2D
 V -0.1   prec_sens2D
SENSOR2D
 s/laser & s/peri/spli:D & s/peri/mirr & s/sens2
 H  0.2   prec_sens2D
 V -0.1   prec_sens2D
```

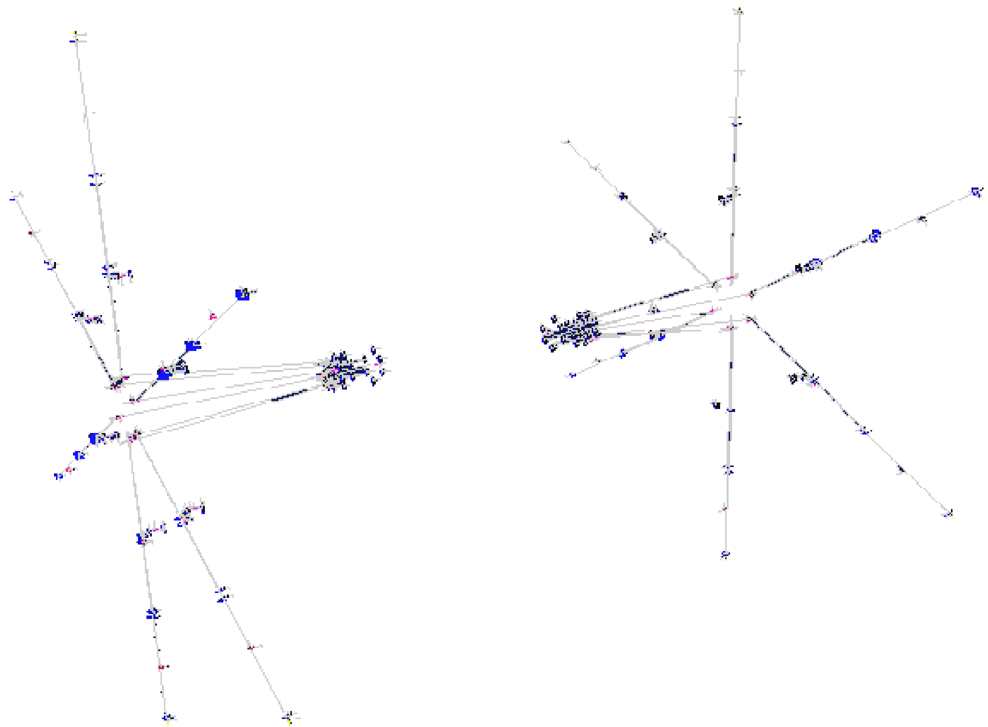
Use of COCOA

- Several desing studies

- Several test benches

- Simulation full CMS Link alignment system (3000 parameters)

-Simulation full CMS Muon Endcap system (6500 parameters)



- Reconstruction of ISR test (test of a full CMS muon alignment halfplane)

- Reconstruction of MTCC test (fraction of final design of CMS, with B field)

- Will be used in 2007 for final CMS hardware alignment systems

The CMS optical alignment system

CMS elements suffer movements and deformations from magnetic field, gravity and temperature
(≈ several mm)

We need precision ≈ 150 μm:

Monitor Muon Chambers relatively among them

- Align. internal Muon Barrel
- Align. internal Muon Endcap

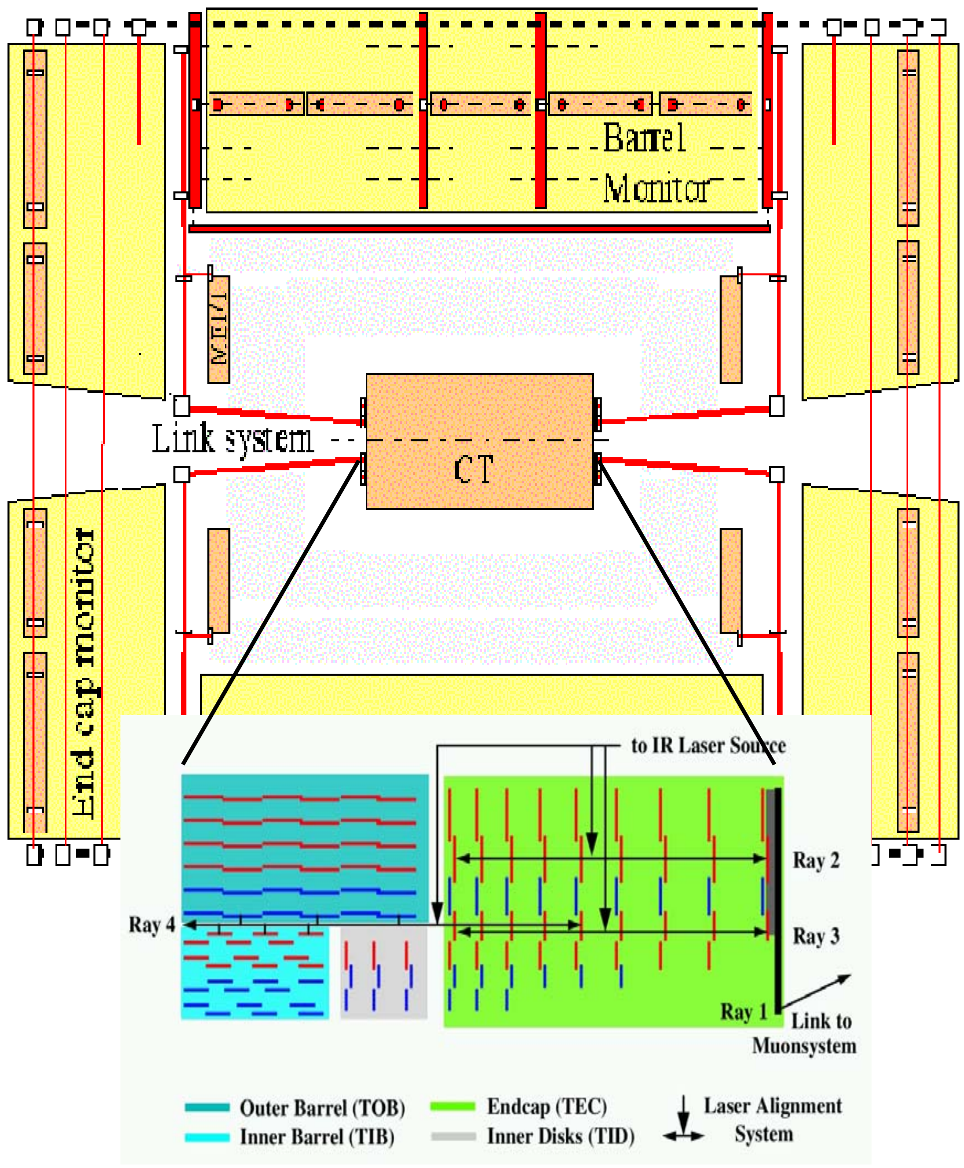
Monitor Muon Ch. w.r.t. Tracker

- Align. Muon ↔ Tracker (‘Link’)

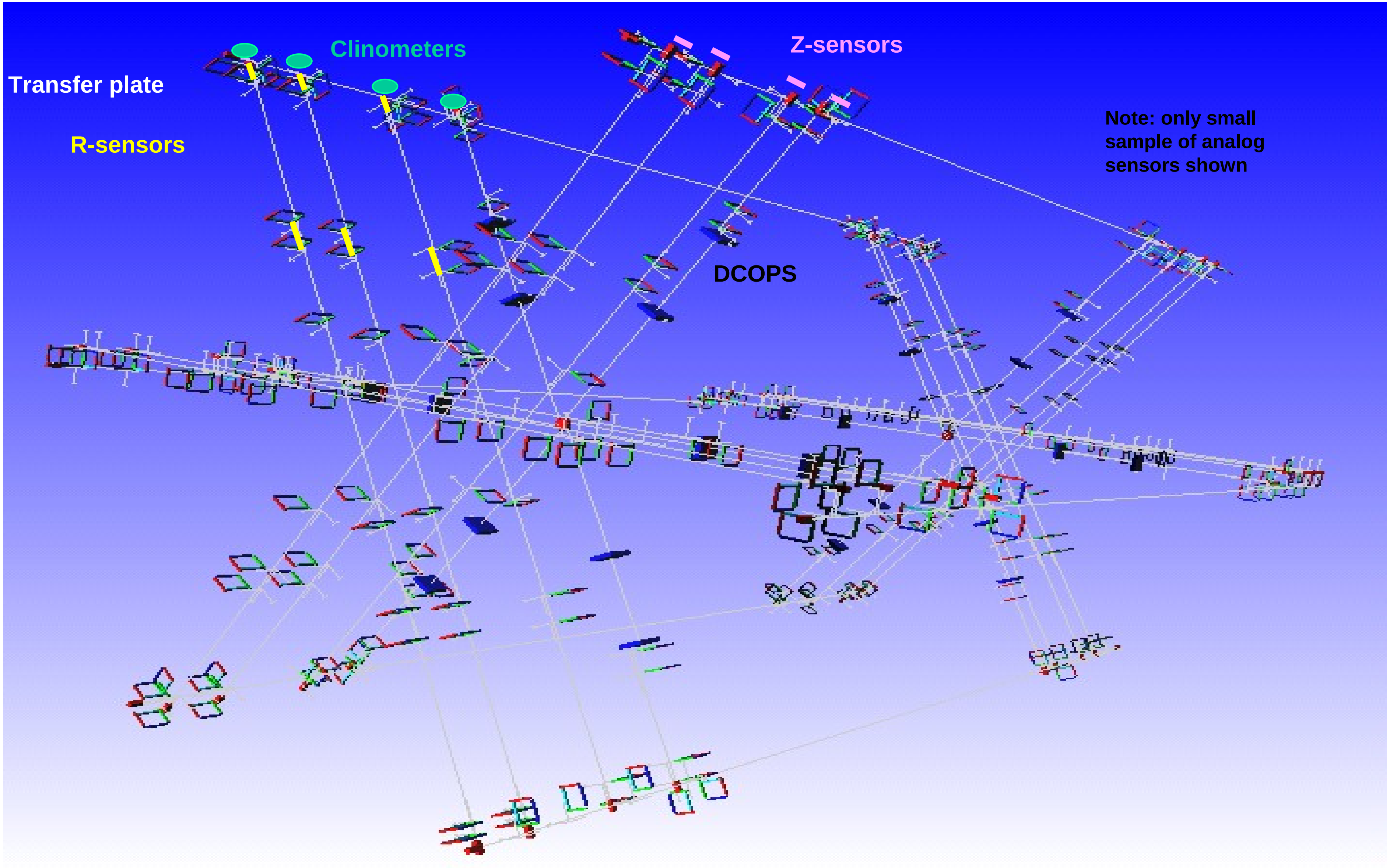
Monitor Tracker Sensors relatively among them

- Align. Internal Tracker

***4 subsystems* with quite different hardware**



Muon Endcap Alignment: Full simulation view



Reconstruction of ISR test

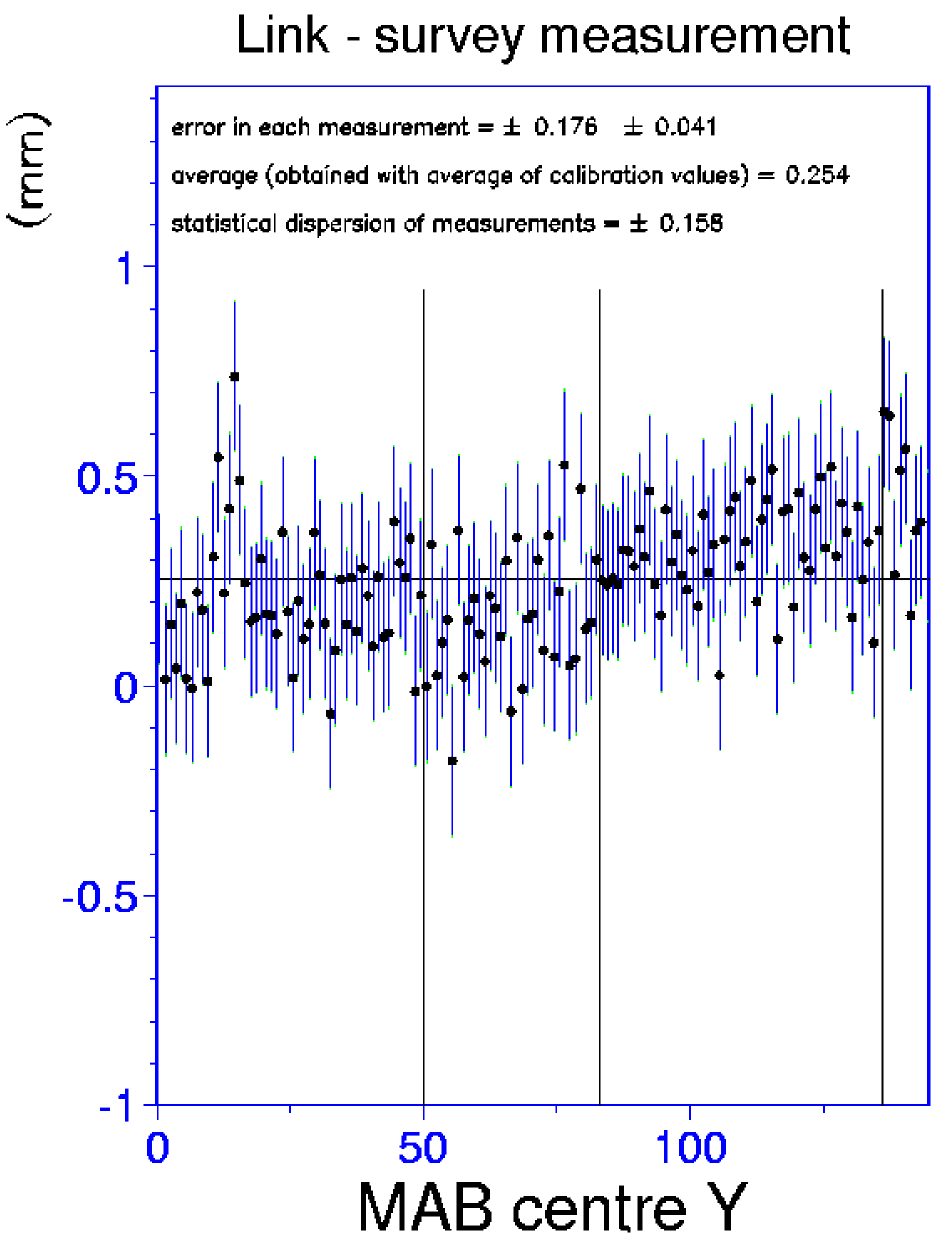
COCOA

- ‘Proof of concept’ test of CMS alignment system: one full half-plane

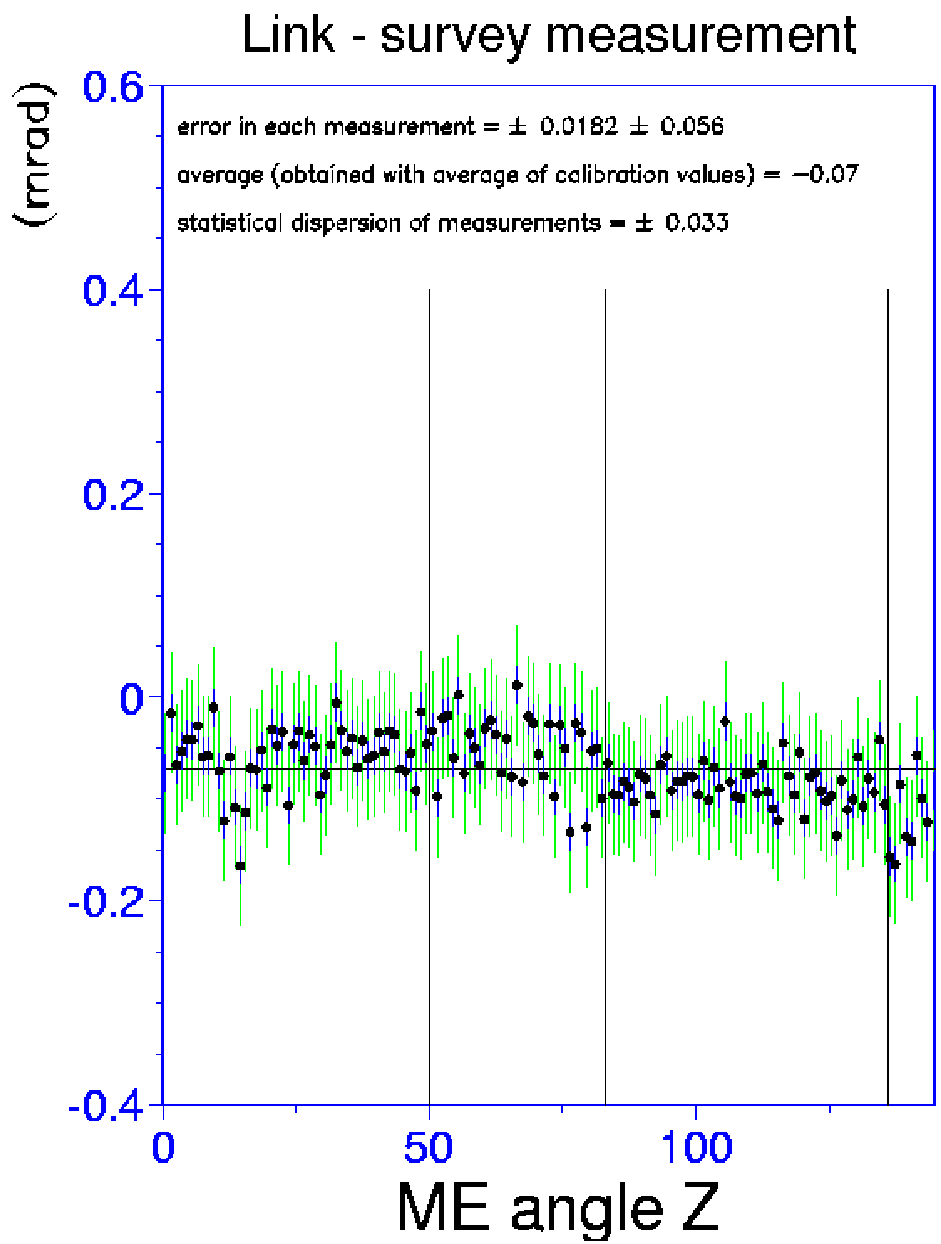
- Barrel
  - 18 forks (4 light sources each)
  - 3 double cameras
  - 3 single cameras on MAB+z
  - 120 measurements
- Endcap
  - 2 x-hair lasers
  - 7 COPS
  - transfer plate with 2 COPS
  - 1 COPS on MAB +Z
  - 1 COPS on fake MA -Z
  - 47 measurements
- Link
  - 2 laserboxes
  - laser level
  - 10 2D sensors
  - 2 tubes
  - 4 distancemeters
  - 4 tiltmeters
  - 312 measurements

- Input object parameters from calibrations
- Input object positions from survey
- Input measurements collected during August and September 2001

August



September



COCOA

Time and memory consumption

COCOA

Full CMS Link alignment system (2865 parameters):

- **31 minutes** in Pentium III 850 MHz
- Memory: **590 Mb**
  - Due to the size of matrices
- Time scales as ~(#param)^3 and memory scales as ~(#param)^2!

$\Rightarrow$ we cannot simulate full CMS (~40k params)

Time and memory consumption

COCOA

☺ Several solutions under study:

- **Diminish the number of parameters**
  - Many parameters have a negligible effect in the final result
  - Needs a thorough testing to avoid biasing
- **Split the system in N parts**
  - There is no really independent subsystem though…
- **Use other library packages**
  - Millipede II, …