WEPV016

THE AUTOMATIC LHC COLLIMATOR BEAM-BASED ALIGNMENT SOFTWARE PACKAGE

G. Azzopardi*, B. Salvachua Ferrando, CERN, Geneva, Switzerland,
G. Valentino, University of Malta

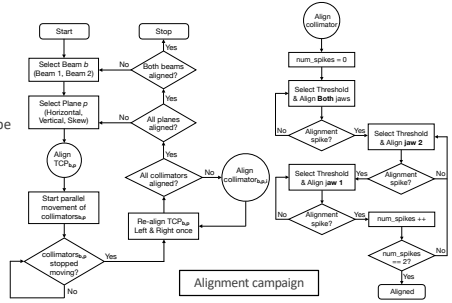
ICALEPCS 2021

Introduction

- Collimation system protects LHC.
- 100+ collimators, each made of 2 jaws inside a vacuum tank.
- Alignment campaigns required to set-up the collimation hierarchy.
- 30% of collimators have embedded Beam Position Monitoring (BPM) pick-up buttons.
  - BPMs directly measure the beam orbit at the collimator.
- All collimators have Beam Loss Monitoring (BLM) devices installed downstream, outside the beam vacuum.
  - BLMs detect losses when halo particles impact the jaws. A spike in the losses indicates the reference halo was touched.

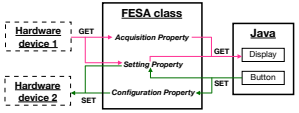
Beam-based Alignment (BBA) with BLMs

- Reference collimator aligned with collimator to create reference halo.
- The collimator jaws are moved towards the beam in steps of 5-20 µm whilst monitoring the BLM signal recorded in the collimator's respective BLM.
- The BBA allows to infer the local orbit position and the relative opening w.r.t. the primary collimator, to establish the collimation hierarchy.
- Collimators from the 2 beams can be aligned in parallel.
- Cross-talk must be considered, i.e. when losses generated by a collimator are detected by multiple BLMs around the LHC.



Software Architecture

- 3-tier structure (see Figure).
- The hardware is abstracted and controlled through FESA (Front-End Software Architecture).
- The control system communicates with FESA through devices.
  - FESA devices are typically abstractions of the hardware, grouped into a FESA class.
- The Java Swing GUI applications interact with the FESA class through the available Java API.



Implementation

- The fully-automatic alignment is implemented in a dedicated FESA class - CollAlignSupervisor.
- It relies on the automation of 3 main components:
  - **Collimator selection** for parallel alignment avoiding cross-talk.
  - **BLM threshold selection** to stop collimators moving towards the beam.
  - **Spike classification** using supervised machine learning to classify between alignment spikes and spurious spikes.
- These 3 components are developed as individual modules, independently available for any improvements.

Multi-threading

- Only 2 collimators can be aligned in parallel, 1 per beam (shared reference and cross-talk).
- The FESA class is assigned 2 devices, to run 2 instances of the software in parallel, i.e. 2 threads.
- Each thread communicates:
  - The beam/plane being aligned.
  - The reference collimator status, i.e.: moving/waiting status.
  - The current collimator ongoing alignment, for cross-talk purposes.
  - The global wait status, i.e. if any thread is waiting for an action from the other thread.

GUI Application Communication

FESA Class Property	States	Definition
Auto status	-1, 0, 1	Alignment: (error, paused/stopped, ongoing)
Align status	-3, -2, -1, 0, 1, 2	Parallel: (ongoing, done), Ignore, Collimator alignment: start, done, done + saved
Parallel status	-1, 0, 1, 2, 3, 4, 5	Deadlock, Ignore, Wait: (crosstalk, parallel, pause, TCP alignment, change collimator)
Parallel message	-	Any message to display in GUI
TCP status	-4, -3, -2, -1, 0, 1	Before collimator: (not aligned, aligned), Aligned before parallel, Ignore, Aligning: (before, after coll)
Collimator status	-	Name of collimator ongoing alignment
Jaw status	-1, 0, 1, 2	Ignore, Aligning: (first jaw, second jaw, both jaws)
Spike class	-2, -1, 0, 1	Ignore, Error, No spike, Spike

Available Features

- The software was designed to:
  - Be autonomous and efficient.
  - Independently "make decisions" in real-time based on the status.
  - Imitate users as much as possible, using "smart" features.
  - Ensure the correct alignment.
- The "smart" features include:
  - Equal priority for collimators from the two beams.
  - Limit the overall waiting time.
  - Reacting to user interrupts.

GUI Usability

- New options introduced:
  - Subsets of collimators in the list can be further grouped for alignments.
  - Collimators can be manually removed from the list during the alignment and re-added at a later stage.
  - Pre-set collimator subsets for alignments available.
  - Closing the application is an automatic stop if any alignment is ongoing.

Alignment Outlook

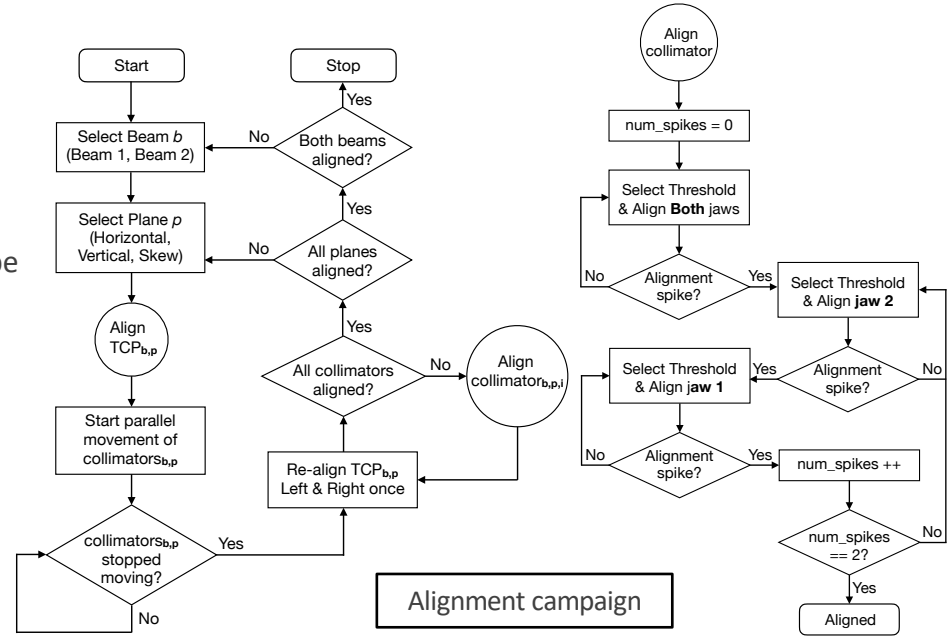
- New alignment configurations accessible and feasible:
  - **Angular alignments** for tighter collimator settings.
  - **Any combinations of collimators** can be aligned efficiently with minimal effort.
  - Subsets of collimators can be **aligned more frequently during operation.**
  - **Dedicated collimator configurations** e.g. ion beams no longer bound to identical setups as with protons.

Introduction

- Collimation system protects LHC.
- 100+ collimators, each made of 2 jaws inside a vacuum tank.
- Alignment campaigns required to set-up the collimation hierarchy.
- 30% of collimators have embedded Beam Position Monitoring (BPM) pick-up buttons.
  - BPMs directly measure the beam orbit at the collimator.
- All collimators have Beam Loss Monitoring (BLM) devices installed downstream, outside the beam vacuum.
  - BLMs detect losses when halo particles impact the jaws. A spike in the losses indicates the reference halo was touched.

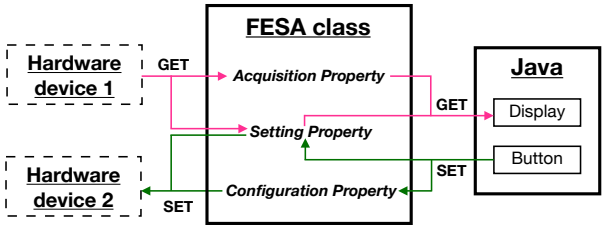
Beam-based Alignment (BBA) with BLMs

- Reference collimator aligned with collimator to create reference halo.
- The collimator jaws are moved towards the beam in steps of 5-20 µm whilst monitoring the BLM signal recorded in the collimator's respective BLM.
- The BBA allows to infer the local orbit position and the relative opening w.r.t. the primary collimator, to establish the collimation hierarchy.
- Collimators from the 2 beams can be aligned in parallel.
- Cross-talk must be considered, i.e. when losses generated by a collimator are detected by multiple BLMs around the LHC.



Software Architecture

- 3-tier structure (see Figure).
- The hardware is abstracted and controlled through FESA (Front-End Software Architecture).
- The control system communicates with FESA through devices.
  - FESA devices are typically abstractions of the hardware, grouped into a FESA class.
- The Java Swing GUI applications interact with the FESA class through the available Java API.



Implementation

- The fully-automatic alignment is implemented in a dedicated FESA class - CollAlignSupervisor.
- It relies on the automation of 3 main components:
  - **Collimator selection** for parallel alignment avoiding cross-talk.
  - **BLM threshold selection** to stop collimators moving towards the beam.
  - **Spike classification** using supervised machine learning to classify between alignment spikes and spurious spikes.
- These 3 components are developed as individual modules, independently available for any improvements.

Multi-threading

- Only 2 collimators can be aligned in parallel, 1 per beam (shared reference and cross-talk).
- The FESA class is assigned 2 devices, to run 2 instances of the software in parallel, i.e. 2 threads.
- Each thread communicates:
  - The beam/plane being aligned.
  - The reference collimator status, i.e.: moving/waiting status.
  - The current collimator ongoing alignment, for cross-talk purposes.
  - The global wait status, i.e. if any thread is waiting for an action from the other thread.

GUI Application Communication

FESA Class Property	States	Definition
Auto status	-1, 0, 1	Alignment: (error, paused/stopped, ongoing)
Align status	-3, -2, -1, 0, 1, 2	Parallel: (ongoing, done), Ignore, Collimator alignment: start, done, done + saved
Parallel status	-1, 0, 1, 2, 3, 4, 5	Deadlock, Ignore, Wait: (crosstalk, parallel, pause, TCP alignment, change collimator)
Parallel message	-	Any message to display in GUI
TCP status	-4, -3, -2, -1, 0, 1	Before collimator: (not aligned, aligned), Aligned before parallel, Ignore, Aligning: (before, after coll)
Collimator status	-	Name of collimator ongoing alignment
Jaw status	-1, 0, 1, 2	Ignore, Aligning: (first jaw, second jaw, both jaws)
Spike class	-2, -1, 0, 1	Ignore, Error, No spike, Spike

Available Features

- The software was designed to:
  - Be autonomous and efficient.
  - Independently “make decisions” in real-time based on the status.
  - Imitate users as much as possible, using “smart” features.
  - Ensure the correct alignment.
- The “smart” features include:
  - Equal priority for collimators from the two beams.
  - Limit the overall waiting time.
  - Reacting to user interrupts.

GUI Usability

- New options introduced:
  - Subsets of collimators in the list can be further grouped for alignments.
  - Collimators can be manually removed from the list during the alignment and re-added at a later stage.
  - Pre-set collimator subsets for alignments available.
  - Closing the application is an automatic stop if any alignment is ongoing.

Alignment Outlook

- New alignment configurations accessible and feasible:
  - **Angular alignments** for tighter collimator settings.
  - **Any combinations of collimators** can be aligned efficiently with minimal effort.
  - Subsets of collimators can be **aligned more frequently during operation.**
  - **Dedicated collimator configurations** e.g. ion beams no longer bound to identical setups as with protons.