SOFTWARE ARCHITECTURE FOR AUTOMATIC LHC COLLIMATOR ALIGNMENT USING MACHINE LEARNING

G. Azzopardi*,1,2, G. Valentino[1], B. Salvachua[2], S. Redaelli[2], A. Muscat[1]
[1] University of Malta, Msida, Malta
[2] CERN, Geneva, Switzerland

Abstract

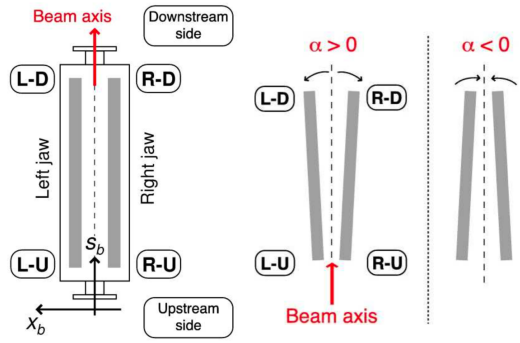
The Large Hadron Collider at CERN relies on a collimation system to absorb unavoidable beam losses before they reach the superconducting magnets. The collimators are positioned close to the beam in a transverse setting hierarchy achieved by aligning each collimator with a precision of a few tens of micrometres. In previous years, collimator alignments were performed semi-automatically, requiring collimation experts to be present to oversee and control the entire process. In 2018, expert control of the alignment procedure was replaced by dedicated machine learning algorithms, and this new software was used for collimator alignments throughout the year. This paper gives an overview of the software re-design required to achieve fully automatic collimator alignments, describing in detail the software architecture and controls systems involved. Following this successful deployment, this software will be used in the future as the default alignment software for the LHC.

INTRODUCTION

The Large Hadron Collider (LHC) at CERN is the largest particle accelerator in the world, built to accelerate and collide two counter-rotating beams at an unprecendented center-of-mass energy of 13 TeV [1, 2]. The LHC is susceptible to beam losses from normal and abnormal conditions, which can damage the state of superconductivity of its magnets. A robust collimation system handles beam losses of halo particles by safely concentrating them into room temperature collimation regions, with a 99.998% cleaning efficiency of all halo particles [3].

The LHC collimation system consists of around 100 collimators, each with two parallel absorbing blocks, referred to as jaws, inside a vacuum tank. The jaws are identified as *left* or *right*, depending on their position with respect to the incoming beam. The jaws must be positioned symmetrically around the beam and their coordinate system is displayed in Figure 1. Each jaw can be moved individually using two stepping motors at the jaw corners, allowing collimators to be positioned at different gaps and angles. The maximum possible operational angle in either direction is 1900 μrad [5]. The jaw corners are known as left-up (LU) and right-up (RU) when they are upstream of the beam and left-down (LD) and right-down (RD) when they are downstream of the beam.

Collimators provide halo cleaning using a multi-stage hierarchy, which is determined after aligning the collimators. Each year of LHC operation begins with a commissioning



(a) Jaw coordinate system (b) Jaw angular tilt convention

Figure 1: (a) The collimator coordinate system and (b) the jaw tilt angular convention as viewed from above, from [4].

phase which involves aligning all collimators and ensuring the correct operation to allow the LHC to achieve nominal operation [6]. Such alignments are performed to determine the beam orbit and beam size at each collimator location, which are otherwise not known sufficiently precisely as the actual beam orbit, collimator tank alignment and optics may deviate from the design orbit. This information is required to position the jaws within a certain number of standard deviations (beam σ) from the beam center [7].

Over the years various software and hardware upgrades were introduced to improve the alignment time and to simplify the alignment procedure. For six years collimators were aligned using a semi-automatic tool, however this reached its minimum alignment time of 3 hours at injection in 2017. This motivated the development of a fully-automatic tool, which was used for the first time in 2018 and has proved to be a beneficial advancement in view of the High Luminosity LHC (HL-LHC) upgrade [8].

LHC COLLIMATOR ALIGNMENTS

Collimator alignments are performed with a step precision of 5 μm. Each collimator has a dedicated Beam Loss Monitoring (BLM) device positioned outside the beam vacuum, immediately downstream, as shown in Figure 2. Such devices are used to detect beam losses generated when halo particles impact the collimator jaws. Recorded losses are proportional to the amount of beam intercepted by the collimator jaws and are measured in units of Gy/s. A collimator is considered aligned when a jaw movement towards the beam produces a clear loss spike in the BLM detector located further downstream [10].

* gabriella.azzopardi@cern.ch

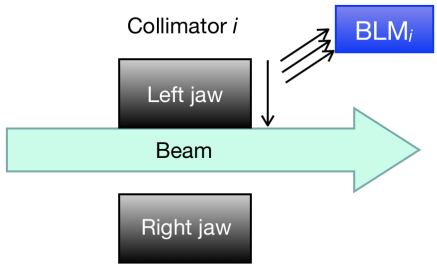


Figure 2: The jaws of collimator *i* around the beam, with its left jaw scraping the beam halo and the showers are detected by the corresponding BLM detector downstream, from [9].

Collimators are aligned using a beam-based alignment (BBA) which is a four-step procedure established in [11]:

1. Align the reference collimator (primary collimator) by touching the beam halo as an initial step prior to the alignment of the collimator in question, collimator *i*.
2. Collimator *i* is then aligned with respect to the reference halo generated.
3. The reference collimator is then realigned to the beam.
4. Collimator *i* is retracted to its position in the hierarchy.

At each alignment step, the left and right jaws are aligned separately, to be able to associate the spike in the BLM signal to the particular jaw that was moving.

Semi-Automatic Alignment Procedure

The previous alignment implementation for the LHC collimators [12] uses a semi-automatic approach. This system makes use of a BLM-based algorithm which requires the user to make decisions as the alignment progresses. A typical semi-automatic alignment involves the following steps:

1. **The user** selects the collimator *i* to be aligned.
2. **The user** sets four input parameters:
   (a) Δx_i^L - Left jaw step size in μm, from a pre-defined list between 5 μm and 200 μm.
   (b) Δx_i^R - Right jaw step size in μm, from a pre-defined list between 5 μm and 200 μm.
   (c) S_i^{Thres} - Stop threshold in Gy/s, from a list of pre-defined thresholds between 1×10^{-7} and 2×10^{-4}.
   (d) t_i^s - Time interval between each step in seconds.
3. **The user** selects which collimator jaw to move towards the beam (left or right or both), and starts the alignment.
4. The jaw(s) of collimator *i* are automatically moved towards the beam in steps of Δx_i every t_i^s seconds.
5. The jaw movement automatically stops if the BLM losses exceed the threshold, S_i^{Thres}, by obtaining the BLM data S_i associated with collimator *i* after each jaw step.
6. Once the jaw(s) stop moving, **the user** is required to analyze the associated BLM losses in order to determine whether the collimator jaw(s) are aligned or not.
7. **The user** first aligns both jaws, then the left jaw followed by the right jaw, until a clear alignment spike is observed in each case, by repeating steps 2-6.
8. Once a collimator is aligned, **the user** must select to save the position of collimator *i* so that the beam center and beam size can be automatically calculated.

This system uses the BLM feedback only to automate the movement of collimators towards the beam and requires collimation experts to control the rest of the procedure, hence the term semi-automated.

Fully-Automatic Alignment Procedure

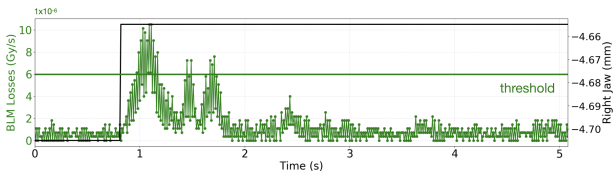
The semi-automatic beam-based alignment was fully-automated by closing the loop between automatically stopping the collimator movement after its losses exceed the threshold, and resuming the alignment based on the BLM loss signal. This involved using the feedback from the BLMs in real-time to replace the user steps from the previous section, with dedicated algorithms.

Crosstalk Analysis for Parallel Selection (Step 1)

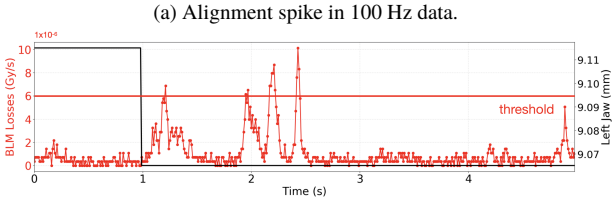
When a collimator reaches the beam envelope, beam losses propagate mainly in the direction of the beam, and are also observed by other nearby BLMs. This phenomenon is known as crosstalk. During alignment campaigns, collimators in the two beams are aligned in parallel to speed up the alignment process. This is possible using the semi-automatic approach by having the users manually select the collimators to align based on the crosstalk observed in their BLM signals. To fully-automate the alignment, it is critical to automate the task of collimator selection. This was done by analysing the crosstalk generated by each collimator when aligned individually. A data set of 650 samples was generated from sequential alignments, such that each sample contains the BLM signal of the aligned collimator and the signal of all other collimators' BLM detectors with losses larger than ten times the background losses. The BLM signals classified as being affected by crosstalk were identified by RMS-smoothing, and this was used to generate a list of collimators affected by crosstalk. This is an initial model for automatically handling the parallel alignment of both beams, with a more advanced analysis ongoing to quantify the level of crosstalk experienced at any collimator [9].

Machine Learning for Spike Detection (Step 6)

The correct alignment of any collimator relies on being able to determine whether a collimator has touched the beam or not, based on the spikes in the BLM signal. An alignment spike, as shown in Figure 3a, indicates that the moving jaw touched the beam and is aligned. On the other hand, non-alignment spikes, as shown in Figure 3b, arise due to other factors such as; beam instabilities or mechanical vibrations of the opposite jaw, thus indicating that the jaw has not yet touched the beam and must resume its alignment. This process of spike recognition was cast as a classification problem by training machine learning models to distinguish between the two spike patterns in the BLM losses. A data set was assembled from previous alignment campaigns, from which fourteen manually engineered features were extracted and six machine learning models were trained, analyzed and thoroughly tested [13]. The suitability of using machine learning in LHC operation was confirmed during collimator alignments performed in 2018, whereby the models achieved a precision of over 95%.



(a) Alignment spike in 100 Hz data.



(b) Non-Alignment spike in 100 Hz data.

Figure 3: Typical BLM signals as a function of time showing examples of (a) an alignment spike and (b) non-alignment spikes, after a collimator movement towards the beam [13].

Automatic Threshold Selection (Step 2c)

A threshold is required to prevent collimators from moving too far into the beam, such that a collimator stops moving when its BLM device records losses that exceed this threshold. The ideal threshold must be: high enough to ignore noise spikes and allow the jaw to touch the beam without interrupting the movement, and low enough to immediately stop the jaws and generate minimal losses when the collimator actually touches the beam. Therefore the algorithm for automatic threshold selection was designed by applying an exponentially weighted moving root mean square on the latest BLM signal to smooth the signal and prioritise the most recent losses. The thresholds selected by users in previous alignments, were extracted to form a data set of 1778 samples, at injection and flat top. This data set was used to validate the algorithm, and the difference between the thresholds selected automatically and by the user were negligible for over 90% of the cases [14].

Highlighting the differences from the semi-automatic BBA, the fully-automatic BBA involves the following steps:

1. **The user** can now select a group of collimators to be aligned.
2. **The user** must then set **two** input parameters:
   - Δx_i - Jaw step size in μm, from a pre-defined list between 5 μm and 200 μm.
   - t_i^s - Time interval between each step in seconds, the minimum being 0.02 s.
3. **The automatic procedure** then selects; a collimator i, the jaws to move towards the beam (left or right or both), and a reasonable threshold, S_i^{Thres}. The alignment is then started automatically.
4. The jaw(s) of collimator i are automatically moved towards the beam in steps of Δx_i every t_i^s seconds.
5. The jaw movement automatically stops if the BLM losses exceed the threshold, S_i^{Thres}, by obtaining the BLM data S_i associated with collimator i after each jaw step.
6. Once the jaw(s) stop moving, **the automatic procedure** uses machine learning to classify the spike exceeding the threshold, to determine whether the collimator jaw(s) are aligned or not.
7. **The automatic procedure** first moves both jaws towards the beam simultaneously, then separately aligns each jaw twice, until a clear alignment spike is observed in each case. Therefore steps 3-6 must be repeated until these alignment spikes are observed, and must also be repeated for all selected collimators.
8. Once a collimator is considered aligned, **the automatic procedure** will save its position in a database, such that the beam center and beam size can be calculated.

LHC COLLIMATION SOFTWARE ARCHITECTURE

The alignment is performed remotely from the CERN Control Center using a top-level application implemented in Java, which allows users to move collimators whilst monitoring their BLM signal. These signals are logged at a frequency of 100 Hz, however they are shown to the user at a frequency of 25 Hz due to a limitation of the TCP (Transmission Control Protocol) communication protocol used.

The software architecture designed for the collimation system is implemented via a 3-tier structure as shown in Figure 4. The bottom level consists of actuators, sensors and measurement devices, which allow for adjusting a number

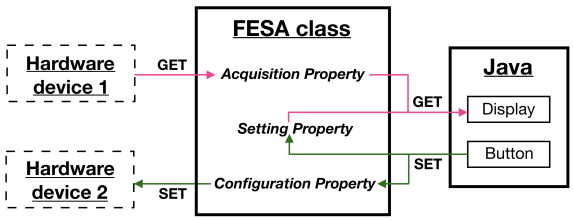


Figure 4: Software architecture diagram showing an example of FESA acting as the middleware between the Java application and hardware devices.

of parameters, including the collimators' left and right jaw positions and angles.

The hardware is abstracted and controlled in real-time through the use of FESA (Front-End Software Architecture) [15], the C/C++ framework used to develop LHC ring front-end equipment software which sends motor step commands [16]. FESA is a complete environment for equipment specialists to design, develop, test and deploy real-time control software for front-end computers (FECs). Its main objective is to standardize, simplify and optimize writing front-end software.

A hardware device is abstracted as a *FESA device* by exposing a public interface made up of *properties*. A property is made up of *value-items* which can be scalars, string, and other simple types. A client can *get*, *set* and *subscribe* to a property in order to read, write and be notified on change, respectively. FESA provides *Acquisition properties* and *Configuration properties* for getting and setting, respectively, and *Setting properties* allowing both getting and setting.

FESA devices are grouped into a *FESA class* which defines the: property interface, private data and real-time behavior, for all devices belonging to that class. The FESA class handles hardware access through the use of Real-Time Actions (*RTActions*) which are triggered by an interrupt to execute in real-time and are subject to tight timing constraints.

The top level consists of Java Swing GUI applications which interact with the FESA middleware framework through the Java API for Parameter Control (JAPC) [17]. JAPC is an API used to build Java applications that control accelerator devices by interacting with the get, set and subscription device properties.

Semi-Automatic Software Architecture

The semi-automatic BBA is implemented in the FESA class *LHCCollAlign* [18] which runs on its own Front-End Computer (FEC) and is responsible for:

- Acting as a BLM concentrator by collecting the 100 Hz data from the BLMs of all the collimators via UDP (User Datagram Protocol), which is converted to 25 Hz to decrease the amount of processing required. The 25 Hz data is used by Java to display the signal for the user to view, and the 100 Hz data is logged for offline access.
- Running the feedback loop during collimator alignments to stop any movement when the losses in the BLM signal exceed the predefined threshold.
- Moving the collimators towards the beam until the BLM losses exceed the threshold, using the *Align Collimator* property. This process is started from the Java application by the user, who must provide the required input parameters and select the jaw(s) to align.

The collimators are controlled using the *LHCCollimator* FESA class. This class is responsible for triggering and setting all collimator jaw movements, in particular functions which move collimators during different phases of the operational cycle, and for getting any data related to the collimator positioning system [19]. An overview of the two FESA classes and the Java application are shown in Figure 5.

Fully-Automatic Software Architecture

The fully-automatic BBA is implemented on top of the semi-automatic BBA, thus allowing for both alignment tools to be available together. It is implemented in the FESA class *CollAlignSupervisor* on its own FEC, and is responsible for:

- Automatically aligning the collimators towards the beam by calling the *Align Collimator* property in the *LHCCollAlign* FESA class, as used by the semi-automatic BBA. The difference being that the user does not provide the input parameters as these are computed automatically. The full-automation also selects the jaw(s) to align and the entire procedure is performed by calling the *Automatic Alignment* property.
- Automatically selecting the collimators to align in parallel based on the offline crosstalk analysis results, by providing the FESA class with the list of collimators affected by crosstalk. This is accessed by the algorithm before selecting any collimator to align, to first check which collimator (if any), is being aligned in the other beam and which collimators it affects. This ensures no collimators affected by crosstalk are aligned in parallel.
- Automatically selecting the threshold for aligning any collimator based on the latest BLM signal. This is implemented by accessing the latest 7.5 seconds of raw 25 Hz BLM data from the *LHCCollAlign* FESA class, and selecting the threshold accordingly. This algorithm keeps track of the previous threshold selected in the *CollAlignSupervisor* FESA class, to enhance the threshold selection process.
- Applying the machine learning model on the recorded BLM signal, to automatically detect alignment spikes and determine if the collimator is aligned. Machine learning model implementations are available in Python therefore a separate thread on the same FEC as the *CollAlignSupervisor* FESA class is dedicated to run the Python script developed for alignment classifications. This script is directly provided with the latest (4 seconds

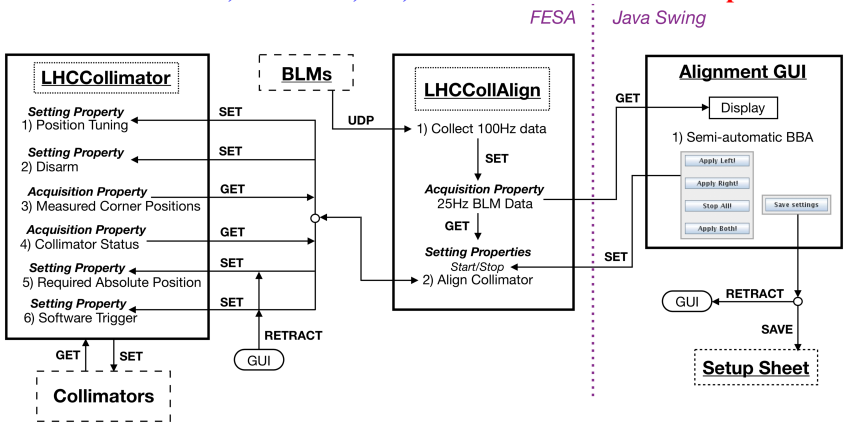


Figure 5: Overview of semi-automatic BBA implemented in FESA and Java.

at injection and 6 seconds at flat top, based on beam halo diffusion [20]) raw 25 Hz BLM data obtained from the *LHCCollAlign* FESA class and returns the classification result to the *CollAlignSupervisor* FESA class to resume the alignment based on the result.

The fully-automatic BBA is controlled from the same Java application, using three new buttons; Play, Pause and Stop. The full implementation of how the new FESA class interacts with the semi-automatic alignment and the Java application, is shown in Figure 6.

FESA allows for executing multiple instances of the same property in different threads. Therefore in order to allow for the alignment of the collimators in beam 1 and beam 2 to be done in parallel, two threads were assigned to the *CollAlignSupervisor* FESA class. This allows executing two instances of the same *Automatic Alignment* property in parallel. These two threads constantly communicate with each other before starting the alignment of any collimator, to ensure that the selected collimators to align in parallel are in accordance with the crosstalk analysis.

Angular Alignment Software Architecture

The current collimation hierarchy requires a 1.5 σ retraction between primary and secondary collimators of the betatron cleaning insertion, which corresponds to around 300 µm. In order to improve the LHC performance and achieve a smaller beam size at the points of collision, tighter collimator settings with smaller retractions are foreseen [21]. Until now, collimators have always been aligned with a zero tilt angle with respect to the beam. Recent beam tests, however, indicated that in reality collimator tank misalignments may introduce an angular tilt of up to a few hundred µm. As a result this approach will not be adequate to operate the system with retractions below 1.5 σ [22]. Therefore an automatic procedure to align collimators with different jaw tilts is required to be able to determine the best angle, i.e. angular alignments.

Three novel angular alignment methods were introduced to find the optimal angle of collimators [23]. These methods were initially implemented using the semi-automatic alignment tool by selecting a high threshold and keeping it fixed. This relied on the assumption that any losses which exceeded such a high threshold were a result of the collimator touching the beam. As expected there were a number of cases where the jaws stopped before touching the beam, due to the lack of spike recognition required to confirm if the jaw is aligned. This resulted in a number of misalignments, however enough results were collected to show that the methods were able to converge to the same angle for the collimators tested [24]. Since then, the semi-automatic alignment was transformed into a fully-automatic alignment, fully-automating the angular alignment.

The angular fully-automatic BBA is implemented in the same *CollAlignSupervisor* FESA class, as the standard fully-automatic BBA. This software was built on top of the semi-automatic BBA and provides the same functionality as the standard fully-automatic BBA, as depicted in Figure 7. The main difference is that the angular fully-automatic BBA is controlled from a separate application, however preserves the same play, pause and stop functionality.

User Interfaces

Two dedicated GUI applications are used for the standard fully-automatic BBA and the angular fully-automatic BBA. Both GUIs subscribe to the same 25 Hz BLM data and 1 Hz jaw position for display purposes.

The standard fully-automatic BBA GUI extends the previous semi-automatic BBA GUI [25], by introducing a new section as shown by the screen shot in Figure 8. The user can input the jaw step size, the time interval between each step and select between the nominal or measured settings when retracting the collimators to their final positions. The alignment can be paused, stopped and resumed by the user at any time.

The GUI lists the collimators to be aligned on the right, grouped by their orientation, such that each group ends with the primary/reference collimator of the plane. A collimator must be selected from this list to control it and to show its corresponding BLM signal and jaw position, with the selected collimator marked by a blue border. The state of each collimator is indicated by the various colour codes listed in Table 1.

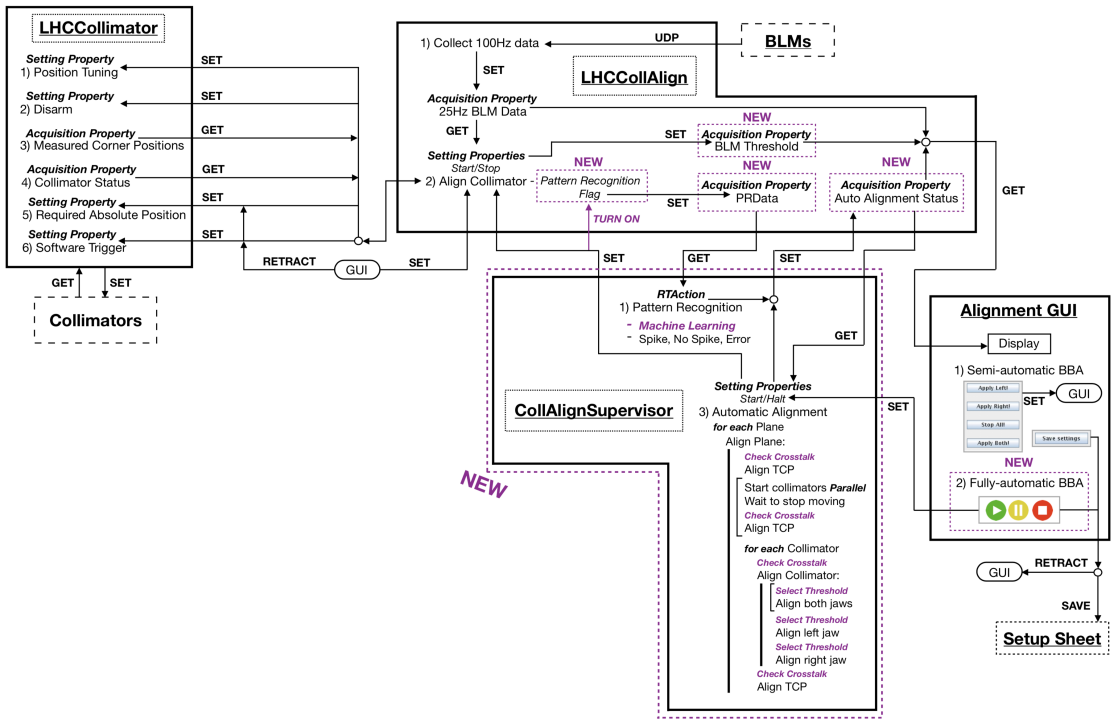


Figure 6: Overview of fully-automatic BBA implemented in FESA and Java.

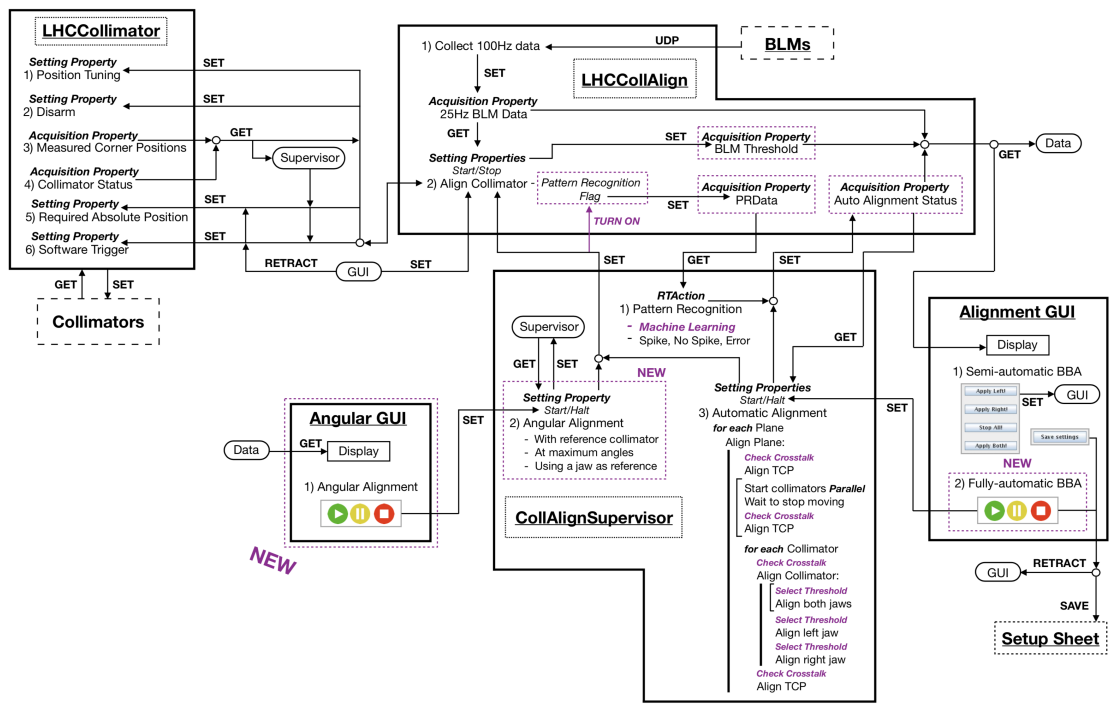


Figure 7: Overview of fully-automatic Angular BBA implemented in FESA and Java.

A standalone GUI was designed for the angular fully-automatic BBA which serves solely as a display, as shown in Figure 9. In this case, all commands must be provided directly to FESA. This was designed as a temporary solution to monitor the status of collimators aligned at different angles when performing beam tests. A more user-friendly version of this tool will eventually be incorporated with the rest of the collimation controls.

In both cases two instances of the same application can be opened at the same time, for easily monitoring the two beams. This allows for aligning a maximum of two collimators in parallel, one from each beam in the case of the standard fully-automatic BBA.

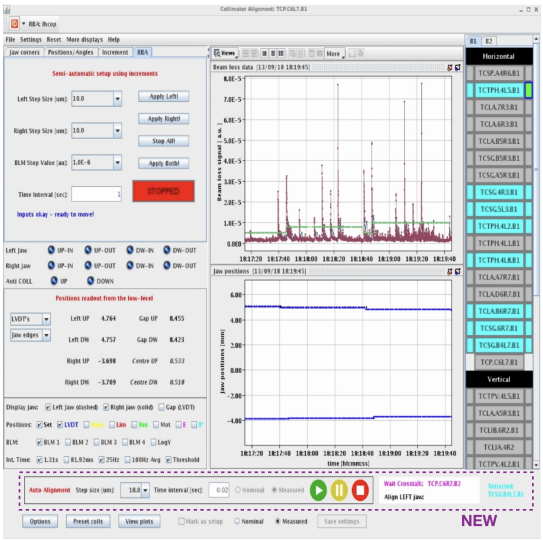


Figure 8: A screen shot of the collimator controller window providing standard fully-automatic alignments.

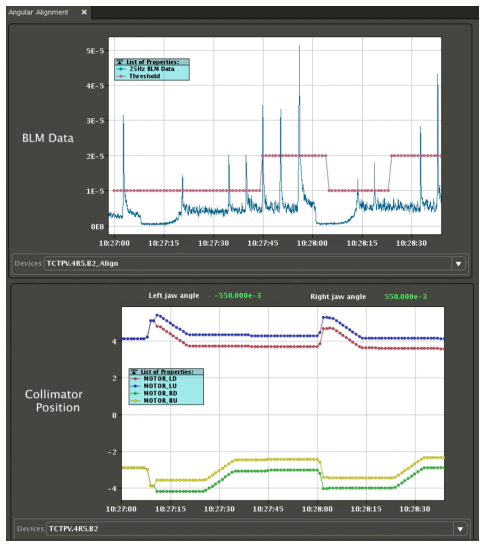


Figure 9: A screen shot of the GUI designed to monitor the angular fully-automatic alignments.

Table 1: Color Coding Used to Identify the Different Alignment States of an Individual Collimator

Color	State
Gray	Not yet moved
Green	Collimator jaw(s) moving
Yellow	Fully-automatic BBA paused by user when this collimator was mid-action.
Red	Fully-automatic BBA stopped by user when this collimator was mid-action.
Turquoise	Collimator aligned and jaws retracted to operational settings.
Magenta	Collimator jaw(s) waiting to align, due to crosstalk from another collimator.

FULLY-AUTOMATIC BBA RESULTS

The standard fully-automatic alignment software was used in LHC operation throughout 2018. During commissioning it was used to automatically align the collimators in the two beams sequentially, including the 79 collimators at injection and the 75 collimators at flat top. The alignment was a success, the beam centres and beam sizes measured at each collimator were consistent with those calculated during injection commissioning in 2017, and the measured settings were used for LHC operation in 2018.

A second version of the fully-automatic alignment was developed later in 2018, to incorporate the crosstalk analysis to align the collimators in the two beams in parallel. This software was tested with beam to align the same 79 collimators at injection. The resulting settings matched those obtained during commissioning in 2018, thus validating this new parallel software. This second version of the fully-automatic software decreased the alignment time of 79 collimators by 71.4% compared to the semi-automatic alignment in 2017, from 2.8 hours to 50 minutes [26].

Furthermore, the angular fully-automatic alignment software is reliable and its results are reproducible, as it is now able to determine the most optimal angle with minimal misalignments. Using this new tool a collimator can be automatically aligned at 41 different angles using 3 different methods in 25 minutes. This is 70% faster in comparison to semi-automatically aligning 1 collimator by manually aligning the 4 jaw corners individually. This was the only reliable approach available, which on average required 40 minutes [27].

SUMMARY

The LHC uses around 100 collimators to protect its sensitive equipment. They are aligned each year at the start of operation using feedback from BLM devices, to establish the correct hierarchy. Before 2018 the alignment procedure was semi-automated whereby the user must manually make decisions based on the recorded signal. The three main user tasks are; crosstalk analysis, spike detection and threshold selection. In 2018 these tasks were replaced by dedicated algorithms, to close the loop between BLM feedback and decision making, to fully-automate the alignment.

The alignment software is developed in FESA, having the semi-automatic and fully-automatic alignments executing on separate FECs. The new fully-automatic alignment was designed to work on top of the semi-automatic alignment, whilst allowing both tools to be available for use. The standard fully-automatic BBA GUI extends the semi-automatic GUI, whereas a separate interface was designed as a temporary solution to monitor the angular fully-automatic BBA.

The fully-automatic software significantly decreased the time required for both standard and angular alignments, and successfully does not require any human intervention. The full automation is a major step in enhancing operational efficiency and will be the default software in the future.

ACKNOWLEDGEMENTS

The authors would like to thank the collimation and operations teams for assisting during beam studies, as well as the support from the High Luminosity LHC project.

REFERENCES

[1] L. Evans, "The Large Hadron Collider", in *New J. Phys.*, vol. 9, no. 9, art. no. 335 (22 pages), 2007. doi:10.1088/1367-2630/9/9/335

[2] L. Evans and P. Bryant, "LHC machine", in *J. Instrum.*, vol. 3, no. 8, art. no. S08001 (158 pages), 2008.

[3] R. W. Assmann *et al.*, "Requirements for the LHC collimation system", in *Proc. 8th European Particle Accelerator Conf. (EPAC'02)*, Paris, France, Jun. 2002, pp. 197–199.

[4] S. Redaelli *et al.*, "Operational performance of the LHC collimation", in *Proc. 46th ICFA Advanced Beam Dynamics Workshop on High-Intensity and High-Brightness Hadron Beams (HB'10)*, Morschach, Switzerland, 2010, paper TUO2C05, pp. 395–399.

[5] D. Missiaen, R. J. Steinhagen, and J. P. Quesnel, "The alignment of the LHC", CERN, Geneva, Switzerland, Rep. CERN-ATS-2009-117, 2009.

[6] M. Lamont, "The LHC from commissioning to operation", in *Proc. 2nd Int. Particle Accelerator Conf. (IPAC'11)*, San Sebastian, Spain, Sep. 2011, paper MOYAA01, pp. 11-15.

[7] A. Mereghetti *et al.*, "Performance of the collimation system during 2016-hardware perspective", in *Proc. 7th Evian Workshop on LHC beam operation*, Evian, France, 2017, pp. 225–228.

[8] G. Apollinari *et al.*, "High-luminosity large hadron collider (HL-LHC): Preliminary design report" , CERN, Geneva, Switzerland, Rep. CERN-2015-005, 2015 and Fermi National Accelerator Lab. (FNAL), Batavia, IL, USA Rep. FERMILAB-DESIGN-2015-02, 2015.

[9] G. Azzopardi *et al.*, "Data-driven Cross-talk Modelling of Beam Losses in LHC Collimation", *Physical Review Accelerators and Beams*, vol. 22, no. 8, p. 083002, 2019.

[10] E. B. Holzer, *et al.*, "Beam loss monitoring system for the LHC", in *IEEE Nuclear Science Symposium Conference Record, 2005*, vol. 2, pp. 1052–1056, 2005. doi:10.1109/NSSMIC.2005.1596433

[11] R. W. Assmann *et al.*, "Expected Performance and Beam-based Optimization of the LHC Collimation System", in *Proc. 9th European Particle Accelerator Conf. (EPAC'04)*, Lucerne, Switzerland, Jul. 2004, paper WEPLT006, pp. 1825–1827.

[12] G. Valentino *et al.*, "Semiautomatic beam-based LHC collimator alignment" , *Physical Review Special Topics-Accelerators and Beams*, vol. 15, no. 5, p. 051002, 2012.

[13] G. Azzopardi *et al.*, "Automatic Spike Detection in Beam Loss Signals for LHC Collimator Alignment," *Nuclear Instruments and Methods in Physics Research Section A*, vol. 934, pp. 10-18, 2019.

[14] G. Azzopardi *et al.*, "Beam Loss Threshold Selection for Automatic LHC Collimator Alignment," presented at 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19), New York, USA, Oct. 2019, paper MOPHA010, this conference.

[15] A. Guerrero, J. J. Gras, J-L. Nougaret, M. Ludwig, M. Arruat, and S. Jackson, "CERN Front-End Software Architecture for Accelerator Controls", in *Proc. 9th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'03)*, Gyeongju, Korea, Oct. 2003, paper WE612, pp. 342–344.

[16] A. Masi, R. Losito, and S. Redaelli, "Measured Performance of the LHC Collimators Low Level Control System", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09)*, Kobe, Japan, Oct. 2009, paper WED001, pp. 612–614.

[17] V. Baggiolini *et al.*, "JAPC - the Java API for parameter control (designing for smooth evolution)", presented at the 10th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'05), Geneva, Switzerland, Oct. 2005.

[18] G. Valentino *et al.*, "Upgraded Control System for LHC Beam-Based Collimator Alignment", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 306–309. doi:10.18429/JACoW-ICALEPCS2015-MOPGF099

[19] A. Masi, R. Losito, and S. Redaelli, "Measured Performance of the LHC Collimators Low Level Control System", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09)*, Kobe, Japan, Oct. 2009, paper WED001, pp. 612–614.

[20] G. Valentino *et al.*, "Beam diffusion measurements using collimator scans in the LHC", *Physical Review Special Topics-Accelerators and Beams,* vol. 16, no. 2, p. 021003, 2013.

[21] R. Bruce *et al.*, "Calculations of safe collimator settings and β* at the CERN Large Hadron Collider", *Physical Review Special Topics-Accelerators and Beams*, vol. 18, no. 6, p. 061001, 2015.

[22] A. Mereghetti *et al.*, "β^*-Reach – IR7 Collimation Hierarchy Limit and Impedance", CERN, Geneva, Switzerland, Rep. CERN-ACC-NOTE-2016-0007, 2016.

[23] G. Azzopardi, A. Mereghetti, S. Redaelli, B. Salvachua, G. Valentino, and A. Muscat, "Automatic Angular Alignment of LHC Collimators", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 928–933. doi:10.18429/JACoW-ICALEPCS2017-TUPHA204

[24] G. Azzopardi *et al.*, "Automatic angular alignment of LHC Collimators", CERN, Geneva, Switzerland, Rep. CERN-ACC-NOTE-2017-0058, Sep. 2017.

[25] G. Valentino, R. W. Assmann, S. Redaelli, and N. J. Sammut, "LHC Collimator Alignment Operational Tool", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper TUPPC120, pp. 860–863.

[26] G. Azzopardi, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Operational Results of LHC Collimator Alignment Using Machine Learning", in *Proc. 10th Int. Particle Accelerator Conf. (IPAC'19)*, Melbourne, Australia, May 2019, pp. 1208–1211. doi:10.18429/JACoW-IPAC2019-TUZZPLM1

[27] G. Azzopardi *et al.*, "Operational Results on the Fully-Automatic LHC Collimator Alignment", *Physical Review Special Topics-Accelerators and Beams*, vol. 22, no. 9, p. 093001, 2019.