ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV016

THE AUTOMATIC LHC COLLIMATOR BEAM-BASED ALIGNMENT SOFTWARE PACKAGE

G. Azzopardi*, G. Valentino[2], B. Salvachua[1]
[1] CERN, Geneva, Switzerland,
[2] University of Malta, Malta

Abstract

The Large Hadron Collider (LHC) at CERN makes use of a complex collimation system to protect its sensitive equipment from unavoidable beam losses. The collimators are positioned around the beam respecting a strict transverse hierarchy. The position of each collimator is determined following a beam-based alignment technique which determines the required jaw settings for optimum performance. During the LHC Run 2 (2015-2018), a new automatic alignment software package was developed and used for collimator alignments throughout 2018. This paper discusses the usability and flexibility of this new package describing the implementation in detail, as well as the latest improvements and features in preparation for Run 3 starting in 2022. The automation has already successfully decreased the alignment time by 70% in 2018 and this paper explores how to further exploit this software package. Its implementation provides a solid foundation to automatically align any new collimation configurations in the future, as well as allows for further analysis and upgrades of its individual modules.

INTRODUCTION

The CERN Large Hadron Collider (LHC) is the world's largest particle accelerator, built to accelerate and collide two counter-rotating beams towards an unprecedented center-of-mass energy of 14 TeV [1,2]. The LHC is susceptible to beam losses from normal and abnormal conditions, which can perturb the state of superconductivity of its magnets and potentially damage equipment. A robust collimation system handles beam losses of halo particles by safely disposing the losses in the collimation regions, with a 99.998% cleaning efficiency [3].

The collimation system consists of more than 100 collimators [4], each made up of two parallel absorbing blocks, referred to as jaws, inside a vacuum tank. The collimators are installed with a fixed rotational angle, depending on their location and functionality, which allows to clean in either the horizontal (H), vertical (V) or skew (S) plane. The jaws must be positioned symmetrically around the beam to ensure safe machine operation. Each jaw can be moved individually using two stepper motors at the jaw corners, allowing collimators to be positioned at different gaps and angles.

BEAM-BASED COLLIMATOR ALIGNMENT

Two types of beam instrumentation are available to align collimators; the Beam Position Monitoring (BPM) and the Beam Loss Monitoring (BLM) systems. BPM pick-up buttons are installed in 30% of the collimators, embedded in their jaws to provide a direct measurement of the beam orbit at the collimator location [5]. BPMs allow for a safe and fast alignment by analysing the electrode signals without needing to touch the beam. The remaining 70% of the collimators can only rely on dedicated BLMs positioned outside the beam vacuum, immediately downstream from the collimator [6]. BLMs are used to detect beam losses generated when halo particles impact the collimator jaws, such that characteristic spikes recorded in the losses indicate that the reference halo has been touched. The procedure of aligning collimators, with BLMs or BPMs, is referred to as beam-based alignment (BBA).

The beam-based alignment with BLM devices is performed via a four-step procedure established in [6]. This involves aligning a reference collimator in addition to the collimator in question (i). The reference collimator is taken to be the primary collimator (*TCP*) in the same plane (p) as collimator i. This creates a *reference halo* that extends into the aperture of collimator i. The procedure is to align the reference collimator before and after collimator i, as depicted in Fig. 1.

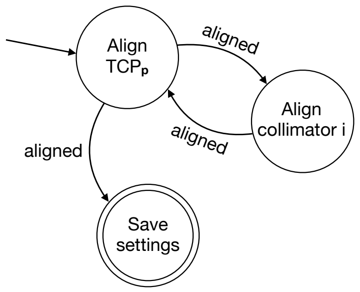

Figure 1: State machine of the beam-based alignment with reference collimator (TCP), from [7].

Once the beam has a well-defined halo amplitude shaped with the primary collimator, the alignment of a collimator can begin. A collimator is aligned by first aligning both jaws simultaneously towards the beam, followed by independently aligning the collimator jaws sequentially. The procedure, as depicted in Fig. 2, involves selecting a BLM threshold to stop the jaw movement when the recorded beam losses

* gabriella.azzopardi@cern.ch

surpass this threshold. Once the movement is stopped, the final step is to determine whether the jaw is aligned. A jaw is classified as aligned when an alignment spike pattern is detected in the losses, on two separate alignment instances. An alignment spike indicates that the collimator jaw has reached a transverse position closer to the beam than the reference collimator.

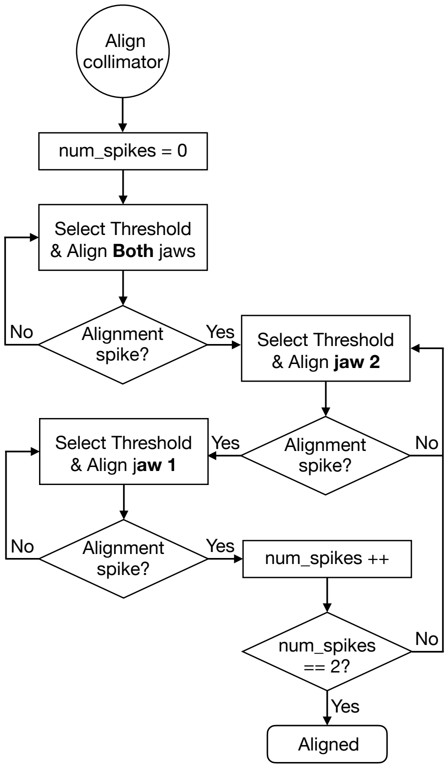

Figure 2: Flowchart of the beam-based alignment of an individual collimator, from [7].

The BBA procedure involves moving the collimator jaws towards the beam in steps of 5-20 μm whilst monitoring the beam loss signal recorded in the collimator's respective BLM. The alignment of any collimator relies on being able to identify an alignment spike when a jaw touches the reference halo: in this condition, the jaw reached the same aperture defined by the primary collimator with an accuracy of the step size. From this measurement, one can infer the local orbit position and the relative opening with respect to the primary collimator, which is used to establish the collimation hierarchy [8]. The BBA depends on being able to efficiently recognize alignment spikes and distinguish them from spurious signals that are often encountered, such that a collimator must continuously move towards the beam ignoring any spurious spikes, until a clear alignment spike is observed.

At the start of each year, and in case of major machine configuration changes during a run, the LHC goes through a commissioning phase to ensure that everything is correctly set up and ready for nominal operation. During the commissioning phase various alignment campaigns take place in order to set up the correct collimation hierarchy. Such campaigns make use of the BBA to align all collimators during their respective machine stages (injection, top energy, collisions, etc.).

To speed-up the procedure during such large alignment campaigns, the first step is to move all collimators in the same plane in parallel towards the beam, until the losses exceed the predefined thresholds. Once all collimators stop moving, each collimator is then aligned individually. The primary collimator of the plane is also aligned before and after moving the plane collimators in parallel to create the reference halo. The entire procedure is displayed in Fig. 3.

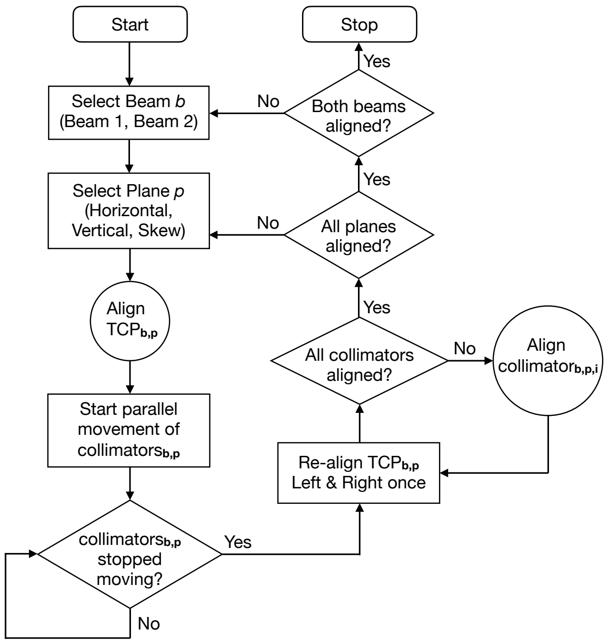
Start
Stop
Yes
Select Beam *b* (Beam 1, Beam 2)
No
Both beams aligned?
Yes
Select Plane *p* (Horizontal, Vertical, Skew)
No
All planes aligned?
Yes
Align TCP$_{b,p}$
All collimators aligned?
No
Align collimator$_{b,p,i}$
Start parallel movement of collimators$_{b,p}$
Re-align TCP$_{b,p}$ Left & Right once
collimators$_{b,p}$ stopped moving?
Yes
No

Figure 3: Flowchart of a collimator alignment campaign using the BBA in Fig. 2, from [7].

In addition to this, collimators from the two beams can be aligned in parallel to further speed-up the procedure. In such cases, one must consider the cross-talk across collimators, whereby losses generated by a collimator are not only detected by its corresponding BLM, but also by other BLM detectors around the LHC [9].

LHC COLLIMATION SOFTWARE ARCHITECTURE

Collimator alignments are performed from the CERN Control Center using a top-level application, allowing users to control collimators and monitor their BLM signal. The software architecture designed for the collimation system is implemented via a 3-tier structure as shown in Fig. 4.

The hardware consists of actuators, sensors and measurement devices. These allow for adjusting a number of parameters, including the collimators' jaw positions.

This hardware is abstracted and controlled in real-time through FESA (Front-End Software Architecture) [11], the

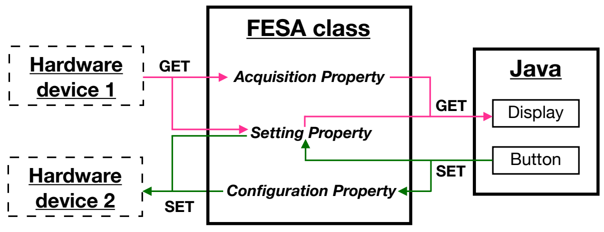

Figure 4: Software architecture diagram showing an example of FESA acting as the middleware between the Java application and hardware devices, from [10].

C/C++ framework used to develop LHC ring front-end equipment software which sends motor step commands [12]. FESA is a complete environment to design, develop, test and deploy real-time control software for front-end computers (FECs), standardizing front-end software development.

The high-level control system communicates with the FEC through devices. A device is the elementary control unit that exposes a public interface made up of properties. Most devices are software abstractions of the hardware (e.g. a collimator). 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.

Finally, the top level consists of Java Swing GUI applications. These interact with the FESA middleware framework through the Java API for Parameter Control (JAPC) [13], to control accelerator devices.

AUTOMATIC BBA IMPLEMENTATION

The fully-automatic alignment software introduced at the end of Run 2, was used throughout 2018 for all collimator alignments [14,15]. When provided with a list of collimators, this new tool is able to automatically align them following the procedure defined in Fig. 3.

The automation acts as a supervisor of the beam-based alignment, as it determines the order of tasks to be executed, controls the flow of collimator alignments, and reacts to the different states reached during an alignment campaign. The entire fully-automatic alignment is implemented on top of the beam-based alignment, within a dedicated FESA class - *CollAlignSupervisor* [10].

This software relies on the automation of three main components:

- Collimator Selection - The parallel alignment of collimators is automatically determined to minimize any cross-talk across the two beams. This makes use of offline analysis performed on 2018 commissioning data to determine the pairs of collimators that can be aligned in parallel without affecting each other [7]. The results of this analysis are available in an external file which can be updated in real-time for the FESA class to use directly.
- Threshold Selection - The collimator movement towards the beam is stopped when the BLM losses exceed this predefined threshold. The threshold is automatically selected and updated based on the real-time BLM losses detected at the collimator. Data from 2016 alignment campaigns was analysed to determine a suitable algorithm, which is implemented directly within the FESA class [16].
- Spike Classification - A “spike” is a signal triggered when the BLM losses reach the selected threshold. Supervised machine learning (ML) is used to automatically classify the BLM loss spikes into two classes; alignment spike or spurious spike. Features are extracted from the BLM loss signal and are used as inputs to the pre-trained ML models for classification [17]. The pre-trained models are available in external files which can be updated in real-time for the FESA class to use through Python. The FESA class handles the Python call for classification in a dedicated thread, which then communicates the final result to the original thread executing the fully-automatic alignment [10].

These three components are developed as individual modules within the automatic alignment software package, independently available for any improvements/upgrades, as demonstrated in [18].

Multi-threading

At any point in time only two collimators can be aligned in parallel, one from each beam, due to the same reference halo used by collimators in the same plane, and cross-talk restrictions across planes. In order to provide this functionality, two devices are defined for the *CollAlignSupervisor* FESA class, allowing two instances of the fully-automatic alignment software to run in parallel, i.e. two threads.

Each thread constantly communicates the following:

- The current beam and plane being aligned.
- The status of the primary collimator in the plane, i.e: moving and waiting statuses.
- The 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

The *CollAlignSupervisor* FESA class provides a number of properties available to the GUI application for collimator control and monitoring. To begin a new alignment the user selects the list of collimators to be aligned, which are automatically sorted into the two beams and aligned sequentially. To align the two beams in parallel the user must open two instances of the GUI application, as this is common practice in the Control Centre. In this case, the user must select two separate lists of collimators to be aligned, one per beam.

Table 1: *CollAlignSupervisor* Properties Available for User Monitoring

Property	**State**	**Definition**
Auto status	ERROR_AUTO(-1)	Alignment halted with error.
	HALT_AUTO(0)	User paused/stopped alignment.
	PLAY_AUTO(1)	Alignment ongoing.
Align status	PARALLEL_ALIGN(-3)	Plane collimators moving in parallel.
	PARALLEL_DONE(-2)	Parallel plane movement ready.
	IGNORE(-1)	Property not in use.
	START_ALIGN(0)	Collimator alignment starting.
	DONE_ALIGN(1)	Collimator aligned.
	SAVE_SETTINGS(2)	Collimator aligned and settings saved.
Parallel status	DEADLOCK(-1)	Both beams waiting for each other.
	OK(0)	Property not in use.
	WAIT_CROSSTALK(1)	Wait due to crosstalk with alignment in other beam.
	WAIT_PARALLEL(2)	Wait parallel plane movement in other beam.
	WAIT_PAUSE(3)	Wait for other beam to pause alignment.
	WAIT_TCP(4)	Wait for TCP alignment in other beam.
	WAIT_CHANGE(5)	Wait for collimator to change in other beam.
Parallel message	-	Any message to display in GUI.
TCP status	TCP_NOT_DONE(-4)	TCP not yet aligned before collimator.
	TCP_DONE(-3)	TCP aligned before collimator.
	PARALLEL(-2)	TCP aligned before parallel plane movement.
	NOT_MOVING(-1)	Property not in use.
	BEFORE_COLL(0)	Ongoing TCP alignment before collimator.
	AFTER_COLL(1)	Ongoing TCP alignment after collimator.
Collimator status	-	Name of collimator ongoing alignment.
Jaw status	NO_JAW(-1)	Property not in use.
	FIRST(0)	First jaw alignment ongoing.
	SECOND(1)	Second jaw alignment ongoing.
	BOTH(2)	Both jaws alignment ongoing
Spike class	NO_CLASS(-2)	Property not in use.
	ERROR(-1)	Error on classification.
	NO_SPIKE(0)	BLM signal classified as non-alignment spike.
	SPIKE(1)	BLM signal classified as alignment spike.

Two FESA properties are used by the Java application to start new alignments:

- Multithreading status - FESA exposes which thread(s) are currently in use, to begin new alignments on available threads.
- Beam status - The beam(s) being aligned by each thread to ensure the user does not attempt to align collimators from the same beam in parallel. In cases of beam overlap the user is requested to wait until the ongoing alignment in the respective beam is completed or paused/stopped by the user.

Once an alignment campaign begins, the user is allowed to pause/resume or stop the alignment. The Java application keeps track of the subset of collimators which are still to be aligned, such that only these are sent to FESA. The user can monitor the status of the automatic alignment, as the FESA class constantly communicates the status through the properties listed in Table 1.

Automatic Alignment Features

The automatic alignment was designed to be autonomous. It must independently “make decisions” in real-time based on the alignment status, until the alignment of all selected collimators is complete. In addition to this, a number of “smart” features have been introduced to the automatic alignment when aligning the two beams in parallel. The aim of these features is to imitate, as much as possible, the decisions a user would take when aligning collimators. This software must align the collimators as efficiently as possible, whilst ensuring the correct alignment and must avoid classifying a collimator "aligned", if it has not reached the reference halo. The incorporated "smart" features include:

- Alternating between collimators from the two beams that must be aligned sequentially due to cross-talk.
- Automatically wait when the collimators in the other beam are performing the parallel plane movement.
- Before starting the parallel plane movement, wait for the collimator in the other beam to finish its alignment (with TCP) and save settings.
- If both beams are close to start the parallel plane movement, wait to move both planes together.
- On pause then resume, the collimators in the current plane are assumed to have already moved in parallel.
- On pause/stop, the other thread will automatically resume (if waiting), as expected.
- Any deadlocks are handled by resuming the alignment on thread 1.

GUI APPLICATION USABILITY

The users have various options newly available at the start of Run 3, including:

- After selecting the list of collimators to align, subsets of the list can be further selected for grouping alignments.
- Collimators can be manually removed from the list during the alignment and re-added at a later stage, in cases of issues with particular collimator or if different settings are to be used for different collimators.
- Preset selections for aligning subsets of collimators, e.g. aligning only collimators with/without BPMs.
- Aligning the TCP before/after collimators (recall Fig. 1) is now optional for the user to select. A combination of having a subset of collimators aligned with TCP and others without TCP is also available.
- Moving the collimators in the current plane in parallel is now optional. (By default, this is not selected at flat top due to a beam dump that occured in the past [7].)

Overall the user always has full control of the automatic alignment with the play/pause/stop buttons. To select subsets of collimators or to change any settings, the user must first pause or stop the alignment. Finally, closing the application is an automatic stop if any alignment is ongoing.

ALIGNMENT OUTLOOK

Introducing the automatic alignment software has made various alignment configurations accessible and feasible to execute on a regular basis:

- Aligning collimators at an angle potentially allows for tighter collimator settings [19]. However, this procedure is longer, as the BBA is applied at different jaw angles to find the most optimal one. The automatic software makes such angular alignments more accessible.
- Any combinations of collimators, with or without TCP, are now available for the users to perform more efficiently with minimal effort.
- Specific collimator alignments can be performed more frequently during LHC operation, rather than having to wait for dedicated beam time for alignment campaigns.
- Collimator configurations, e.g. for ion beams, can now become more independent. In this case, the automatic alignment allows for evaluating dedicated configurations for ion beams, rather than being bound to keeping the identical setup used for proton beams.

CONCLUSION

The LHC is equipped with more than 100 collimators to protect its sensitive equipment. In order to ensure safe beam operation, dedicated collimator alignment campaigns are performed at the start of each year and whenever the LHC configuration is changed, to determine the collimator settings for nominal operation. This is a critical phase to ensure safe operations and various improvements were carried out year after year, in order to reduce the required time while ensuring the accurate alignment necessary.

The majority of collimator alignments are applied using the BBA procedure relying on dedicated beam loss monitors. This procedure has been fully-automated by means of a new software package introduced in 2018, providing a solid foundation to align collimators automatically, yielding a 70% speed-up in collimator alignments [14, 15].

The automatic alignment software package relies on three components that have been developed as independent modules; collimator selection, threshold selection and spike classification. This implementation design provides the flexibility to independently extract each of these modules for further analysis and possibly replace with software upgrades.

This software package is readily available with various new features, to be used as the primary tool for collimator alignments during Run 3. This will enable the configuration of new collimator settings for different machine setups, that were not feasible to be determined in the past due to time restrictions. Moreover, this will make collimator alignments accessible during LHC operation to possibly align subsets of collimators more frequently, rather than waiting for large alignment campaigns to be scheduled.

ACKNOWLEDGEMENTS

The authors would like to thank the collimation team (BE-ABP-NDC) for their feedback as front-end users and the LHC operational crew (BE-OP) for support during software tests with beam.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV016

REFERENCES

[1] L. Evans, "The Large Hadron Collider", in *New J. Phys.*, vol. 9, no. 9, 2007.

[2] L. Evans and P. Bryant, "LHC machine", in *J. Instrum.*, vol. 3, no. 8, 2008.

[3] R. W. Assmann *et al.*, "Requirements for the LHC collimation system", in *Proceedings of the European Particle Accelerator Conference*, Paris, France, pp. 197 – 199, 2002.

[4] G. Azzopardi, *et al.*, "LHC Collimation Controls System for Run III Operation", in the 18th International Conference on Accelerator and Large Experimental Physics Control Systems, Shanghai, China, 2021, paper THPV012, this conference.

[5] G. Valentino, *et al.*, "Final implementation, commissioning, and performance of embedded collimator beam position monitors in the Large Hadron Collider", *Phys. Rev. Accel. Beams*, vol. 20, no. 8, p. 081002, 2017. doi:10.1103/PhysRevAccelBeams.20.081002

[6] R. W. Aßmann, *et al.*, "Expected performance and beam-based optimization of the LHC collimation system", in *Proceedings of the European Particle Accelerator Conference*, Lucerne, Switzerland, 2004, pp. 1825–1827.

[7] G. Azzopardi, "Automation of the LHC Collimator Beam-Based Alignment Procedure for Nominal Operation", Ph.D. thesis, Univeristy of Malta, 2019.

[8] R. Bruce, *et al.*, "Calculations of safe collimator settings and β^* at the CERN Large Hadron Collider", *Phys. Rev. Accel. Beams*, vol. 18, no. 6, p. 061001, 2015. doi:10.1103/PhysRevSTAB.18.061001

[9] G. Azzopardi, B. M. Salvachua Ferrando, and G. Valentino, "Data-driven cross-talk modeling of beam losses in LHC collimators", in *Physical Review Accelerators and Beams*, vol. 22, no. 8, p. 083002, 2019. doi:10.1103/PhysRevAccelBeams.22.083002

[10] G. Azzopardi, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Software Architecture for Automatic LHC Collimator Alignment Using Machine Learning", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 78–85. doi:10.18429/JACoW-ICALEPCS2019-MOCPL04

[11] A. Guerrero, J. J. Gras, J-L. Nougaret, M. Ludwig, M. Arruat, and S. Jackson, "CERN Front-End Software Architecture for Accelerator Controls", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Gyeongju, Korea, 2003, paper WE612, pp. 342–344.

[12] A. Masi, R. Losito, and S. Redaelli, "Measured Performance of the LHC Collimators Low Level Control System", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Kobe, Japan, 2009, paper WED001, pp. 612–614.

[13] V. Baggiolini, *et al.*, "JAPC - the Java API for parameter control (designing for smooth evolution)", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Geneva, Switzerland, 2005.

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

[15] G. Azzopardi, B. M. Salvachua Ferrando, G. Valentino, and S. Redaelli, "Operational Results on the Fully-Automatic LHC Collimator Alignment", *Physical Review Accelerators and Beams*, vol. 22, no. 9, pp. 093001, 2019. doi:10.1103/PhysRevAccelBeams.22.093001

[16] G. Azzopardi, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Automatic Beam Loss Threshold Selection for LHC Collimator Alignment", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 208–213. doi:10.18429/JACoW-ICALEPCS2019-MOPHA010

[17] G. Azzopardi, G. Valentino, A. Muscat, and B. M. Salvachua Ferrando, "Automatic spike detection in beam loss signals for LHC collimator alignment", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 934, pp. 10–18, 2019. doi:10.1016/j.nima.2019.04.057

[18] G. Azzopardi *et al.*, "New Machine Learning Model Application for the Automatic LHC Collimator Beam-Based Alignment", presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2021), Shanghai, China, 2021, paper THPV040, this conference.

[19] G. Azzopardi, A. Mereghetti, A. Muscat, S. Redaelli, B. Salvachua, and G. Valentino, "Automatic Angular Alignment of LHC Collimators", in *Proc. ICALEPCS2017*, Barcelona, Spain, Oct. 2017, paper TUPHA204. doi:10.18429/JACoW-ICALEPCS2017-TUPHA204