4-2020

The CLAS12 Software Framework and Event Reconstruction

V. Ziegler

N. A. Baltzell

F. Bossù

D. S. Carman

P. Chatanon

See next page for additional authors

Authors

V. Ziegler, N. A. Baltzell, F. Bossù, D. S. Carman, P. Chatanon, M. Contalbrigo, J. Newton, and M. Ungaro

Nuclear Inst. and Methods in Physics Research, A 959 (2020) 163472

Contents lists available at ScienceDirect

Nuclear Inst. and Methods in Physics Research, A

journal homepage: www.elsevier.com/locate/nima

The CLAS12 software framework and event reconstruction

V. Ziegler [a,*], N.A. Baltzell [a], F. Bossù [b], D.S. Carman [a], P. Chatagnon [c], M. Contalbrigo [d], R. De Vita [e], M. Defurne [b], G. Gavalian [a], G.P. Gilfoyle [f], D.I. Glazier [g], Y. Gotra [a], V. Gyurjyan [a], N. Harrison [a], D. Heddle [h], A. Hobart [c], S. Joosten [i], A. Kim [j], N. Markov [a], S. Mancilla [k], M.D. Mestayer [a], J. Newton [l], W. Phelps [h], S. Niccolai [c], D. Sokhan [g], M. Ungaro [a]

[a] Thomas Jefferson National Accelerator Facility, Newport News, VA 23606, USA
[b] CEA-Saclay, Univ. Paris-Sud, Université Paris-Saclay, Gif-sur-Yvette, France
[c] Institut de Physique Nucléaire, CNRS-IN2P3, Univ. Paris-Sud, Université Paris-Saclay, 91406 Orsay Cedex, France
[d] INFN, Sezione di Ferrara, 44100 Ferrara, Italy
[e] INFN, Sezione di Genova, 16146 Genova, Italy
[f] University of Richmond, Richmond, 23173 VA, USA
[g] University of Glasgow, Glasgow G12 8QQ, United Kingdom
[h] Christopher Newport University, Newport News, 23606 VA, USA
[i] Argonne National Laboratory, Chicago, 60439 IL, USA
[j] University of Connecticut, Storrs, CT 06269, USA
[k] Universidad Técnica Federico Santa María, Valparaíso, 2390123, Chile
[l] Old Dominion University, Norfolk, 23529 VA, USA

ARTICLE INFO

Keywords:
Electron scattering
Hadronic physics
Event reconstruction software

ABSTRACT

We describe offline event reconstruction for the CEBAF Large Acceptance Spectrometer at 12 GeV (CLAS12), including an overview of the offline reconstruction framework and software tools, a description of the algorithms developed for the individual detector subsystems, and the overall approach for charged and neutral particle identification. We also present the scheme for data processing and the code management procedures.

1. Introduction

This paper describes the software framework, tools, and algorithms that were developed in support of event reconstruction and analysis of the CLAS12 (CEBAF Large Acceptance Spectrometer at 12 GeV) experiment at Jefferson Lab (JLab) [1]. Installed in experimental Hall B, CLAS12 is a large acceptance spectrometer based on two superconducting magnets and multiple detector subsystems that provides large coverage for the detection of charged and neutral particles produced by the interaction of an electron beam from the JLab CEBAF accelerator with a target located at the center of the spectrometer. A six-coil torus magnet defines the six-sector structure of the so-called Forward Detector that is outfitted with Drift Chambers [2] for charged particle tracking and multiple detector systems for particle identification. These detectors include threshold Cherenkov Counters [3,4] and Ring-Imaging Cherenkov Counters [5], scintillator-based time-of-flight hodoscopes [6], and electromagnetic calorimeters [7]. In the target region, a 5 T superconducting solenoid surrounds a central tracker based on silicon and Micromegas detectors [8,9], and subsystems for particle identification that include a time-of-flight scintillation counter barrel [10] and a neutron detector [11], forming the so-called Central Detector.

Fig. 1 shows a model representation of the CLAS12 spectrometer identifying the Forward and Central Detectors. In between the central and forward region, the CLAS12 Forward Tagger [12] extends the kinematic coverage for the detection of electrons and photons at polar angles from 2° to 5° (see Fig. 2). The Forward Detector covers the polar angle range from 5° to 40°, while the Central Detector covers the polar angle range from roughly 35° to 125°. The total number of readout channels of CLAS12 is larger than 100k. Typical trigger rates are 15 kHz. In 2018, data rates of 500 MB/s with a live time of >95% were achieved. A total of ~2 pB of data was accumulated in 2018.

The CLAS12 offline reconstruction and analysis framework was developed to cope with the complexity of the spectrometer and the related data volumes. It consists of an extensive library of software tools, of detector reconstruction packages, and a framework to chain the reconstruction and analysis applications for data processing. Software tools have been designed to support and standardize event reconstruction including detector calibration and monitoring, data analysis, I/O functionality, database access, detector geometry, and to handle magnetic field based calculations. Detector reconstruction packages are designed to extract from the raw data the relevant information for

* Corresponding author.
E-mail address: ziegler@jlab.org (V. Ziegler).

https://doi.org/10.1016/j.nima.2020.163472
Received 18 December 2019; Accepted 16 January 2020
Available online 22 January 2020

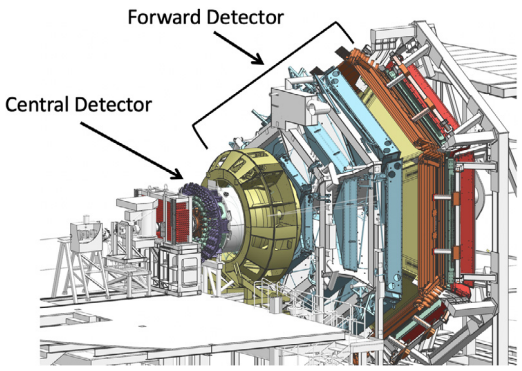

Fig. 1. Model representation of the CLAS12 spectrometer in Hall B at Jefferson Laboratory. The electron beam is incident from the left side of this figure. The CLAS12 detector is roughly 20 m in scale along the beam axis. The CLAS12 Forward and Central Detectors are identified.

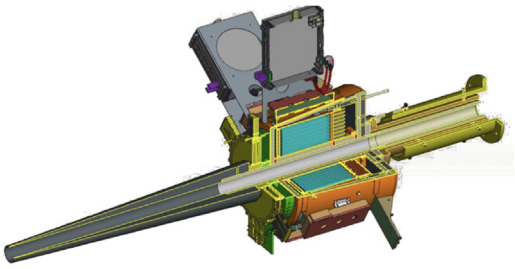

Fig. 2. Model representation of the CLAS12 Forward Tagger that is positioned just upstream of the torus magnet along the beam axis. Attached to the upstream face of the detector is the Møller electron shielding cone.

particle reconstruction, such as tracks, hits, or clusters. These are the input information for the CLAS12 Event Builder, which sifts through the reconstructed detector output to identify particles and form the reconstructed event. The reconstruction components are deployed in a service-oriented platform (see Section 2), which provides the functionalities for data processing for both event reconstruction and the subsequent analysis. While the software framework supports multiple programming languages, the CLAS12 reconstruction packages and tools currently in use are developed in Java.

This paper is organized as follows. The CLAS12 software framework and tools are described in Section 2. The raw and reconstructed data formats are presented in Section 3. The monitoring, calibration, and event display applications are described in Sections 4 and 5. Section 6 provides a detailed description of the detector and event reconstruction packages, including selected results from reconstruction of simulated data that have been used to develop and validate the algorithms. The reconstruction performance on beam data is presented in Ref. [1]. Finally, Sections 7 and 8 present the data processing and code management procedures adopted for CLAS12.

2. Software framework and tools

Nuclear and particle physics data processing applications must guarantee a long lifetime, larger than the multi-year duration of the corresponding experiment. The ability to upgrade and adapt technologies is therefore essential, so these applications should be organized in a way that easily permits upgrades of aged software components and inclusion of new ones, without need for major redesign or structural changes. Support for software evolution and diversification (e.g. compatibility with heterogeneous hardware structures, such as FPGAs and GPGPUs) is important to accommodate more efficient and robust data-processing applications in the future.

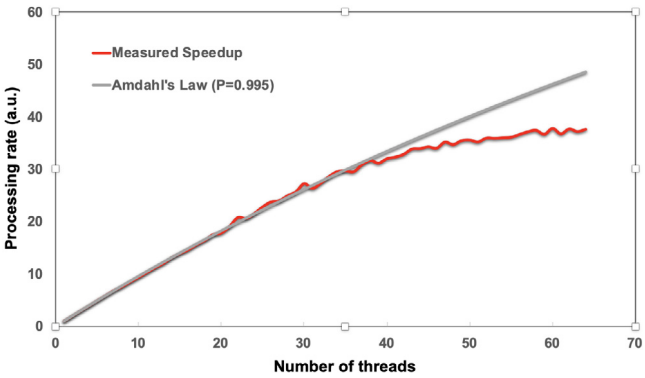

Fig. 3. Scaling of the CLAS12 full event reconstruction application as implemented in the CLARA framework. Tests were conducted on a Intel Xeon node (E5-2697A v4 @ 2.6 GHz). Comparison with Amdhal's law indicates 99.5% parallel efficiency over the 32 physical cores of the machine.

Following these principles, CLAS12 reconstruction and analysis relies on a data-stream processing framework called CLARA [13–16], which provides a service-oriented architecture in which to build the relevant software applications. Such applications are composed of interlocking building blocks called micro-services, which are linked together by data-stream pipes. The technology (e.g. a high-level programming language or hardware deployment details), as well as the algorithmic solutions used to process data, are encapsulated. The scope of a specific software application implemented in CLARA is determined by the micro-services that are included and by the order of their execution.

A micro-service receives input data, processes it, and produces output data, where the I/O is organized into tabular structures called "banks" whose structure is configured by the specific service developer. A micro-service reacts to an input data stream, processes it, and passes processed data to the next micro-service in the data-flow path. As a result, the CLAS12 data processing application is versatile and flexible, since the application building blocks can be improved individually and replaced with no need for structural changes in the framework. The CLAS12 micro-services are extensions of an abstract reconstruction engine, which includes common components such as initialization and event processing methods. This approach reduces and simplifies the development of an individual micro-service and enforces a common structure.

The CLARA data-stream pipe is a data bus based on the xMsg messaging system that supports various protocols such as MPI, pub-sub, p2p, inproc, and shared memory. The CLARA orchestrator, i.e. the process level workflow management system, controls the overall process execution.

The framework enables execution of software applications in multi-threaded mode. This is implemented via event-level parallelization for the CLAS12 reconstruction. The framework is specifically designed to do thread-based parallelization on multi-core machines, thereby allowing the simultaneous reconstruction of multiple events having as many active threads as the cores on the system. Fig. 3 shows the results of a scaling test on an Intel Xeon node (E5-2697A v4 @ 2.6 GHz). Comparison with Amdhal's law indicates 99.5% parallel efficiency over the 32 physical cores of the machine.

The CLARA framework provides service interface implementations in Java, C++, and Python languages. At present, all of the CLAS12 reconstruction services deployed using the CLARA framework are written in Java.

2.1. Common tools

The offline software of the CLAS12 project aims to provide tools that allow design, simulation, and data analysis to proceed in an efficient,

repeatable, and understandable way. Most software engineering details are hidden from users, allowing them to concentrate on the algorithms and physics. To facilitate code development for the detector subsystems of CLAS12, the software was designed to provide libraries that are commonly used by all of the reconstruction packages. These libraries, referred to as "common tools", contribute to software maintainability by avoiding code replication, which facilitates code maintainability.

The common tools consist of various packages, each having a specific purpose and functionality. Below we discuss the main packages used in the reconstruction software.

2.1.1. Geometry

Due to the complexity of the geometry of the CLAS12 detector subsystems, an interface was developed to provide classes and software tools that are used to describe the geometry of all subsystems in a unified way. A library of primitives provides geometrical objects needed to represent all detector subsystems (these include lines, planes, and various shapes such as cubes, trapezoids, etc.) and to provide the necessary transformations to accommodate misalignments and distortions. Furthermore, geometry tools provide methods to track particles through volumes for evaluation of track trajectories, such as line-to-surface intersections, ray tracing through objects, and evaluation of the distance of closest approach to a line or surface.

The CLAS12 geometry library is initialized from a database containing key geometry parameters and their variations for every detector. This maximizes flexibility, supports time-dependent experiment geometry conditions, and ensures consistency between the simulation, reconstruction, and event visualization packages.

To facilitate development of new detector geometries, visualization capabilities are included in the geometry library. Fig. 4 shows a view of part of the CLAS12 spectrometer using this functionality.

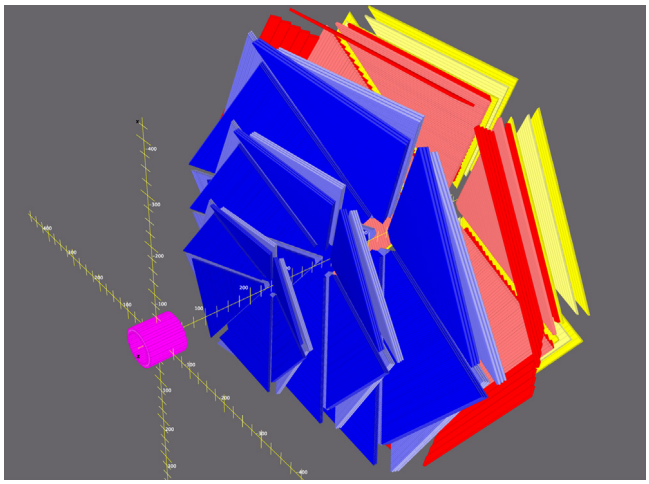

Fig. 4. Visualization of part of the CLAS12 spectrometer via the geometry package. From left to right, the Central Neutron Detector (CND) in magenta, the Drift Chambers (DC) in blue, the Forward Time-of-Flight (FTOF) in red, and the Electromagnetic Calorimeter (ECAL) in yellow are shown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.1.2. Databases

The Calibration Constant Database (CCDB) software package was developed at Jefferson Lab for the GlueX experiment in Hall D [17]. CCDB provides the functionality for storing and accessing structured tables in MySQL-based and SQLite portable databases. The CLAS12 reconstruction packages use the CCDB application programming interface to create and access tables that contain detector geometry and calibration constants, as well as maps used for decoding raw data. At the decoding stage, signals are converted from hardware notation (crate, slot, channel) into the CLAS12 notation (sector, layer, component).

The constants in CCDB tables are linked to specific runs (using time stamps), so that different variations of constants are stored depending on run conditions. CLAS12 software tools employ an Application Programming Interface (API) that parses database tables and creates structured maps of constants stored in memory by detector sector, layer, and component. This allows fast retrieval of the constants.

The CLAS12 database access tools have been developed to avoid bottlenecks that might result from multiple multi-threaded services accessing the database to retrieve constants. An interface has been designed to fetch the constants on demand and cache them for further requests. In this approach each service will request the constants it requires on one thread and each subsequent request by a new thread accesses the cached values.

2.1.3. Plotting and analysis tools

For ease of integration with the reconstruction software tools and packages, the plotting tools used for data calibration, monitoring, and analysis were developed in the Java programming language.

The plotting software, called *groot*, developed at Jefferson Lab for CLAS12 is tailored to have a programming interface similar to the CERN data analysis package, ROOT, and provides the necessary functionalities for histogram and graph creation, filling, and manipulation, as well as for fitting using the Java-based MINUIT library available from the JHEP repositories. This has been the base for the development of the detector monitoring and calibration suites (see Section 4).

These same tools can also be used for physics analysis. An additional analysis package containing classes for four-vector manipulations allows computation of event kinematics (e.g. Q^2 and W), Lorentz boosts, etc.

2.1.4. Magnetic field package

The magnetic field package, *magfield*, used by the CLAS12 reconstruction creates binary field maps from engineering models of the CLAS12 torus and solenoid [18]. It employs a common self-described binary format, with a header containing meta-data describing the pedigree of the field, its grid coordinate system, and the coordinate system of the field components. For example, the CLAS12 torus has a cylindrical grid but Cartesian field components. The same *magfield* package provides the trilinear interpolation of the field (a method of multivariate interpolation on a 3-dimensional regular grid). Given that the field is often requested at a sequence of points all contained within a single grid cell, *magfield* uses time-saving software "probes" to cache nearest neighbors.

2.1.5. Swimmer package

The *swimmer* package, in conjunction with the *magfield* package, is used in the CLAS12 reconstruction to propagate charged particles through the CLAS12 solenoid and torus fields. It uses a fourth-order (with fifth-order corrections) adaptive step-size Runge–Kutta integrator with single-step advancement that is achieved through a configurable Butcher tableau advancer. There are a number of convenience methods for swimming to a plane, to the closest point on a line, and to a specified value of a given (x, y, z) coordinate. For forward swimming in CLAS12, performance is improved by reducing the dimensionality of the track state vector that contains the main track parameters (Section 6.2.4), by changing the independent variable from the path length to the coordinate along the beamline, which defines the nominal CLAS12 z-axis.

3. Data formats

EVIO (Event Input–Output) [19] is a data format designed and maintained by the JLab Data Acquisition Group, and is the data format of the raw data. For event reconstruction and analysis, the CLAS12

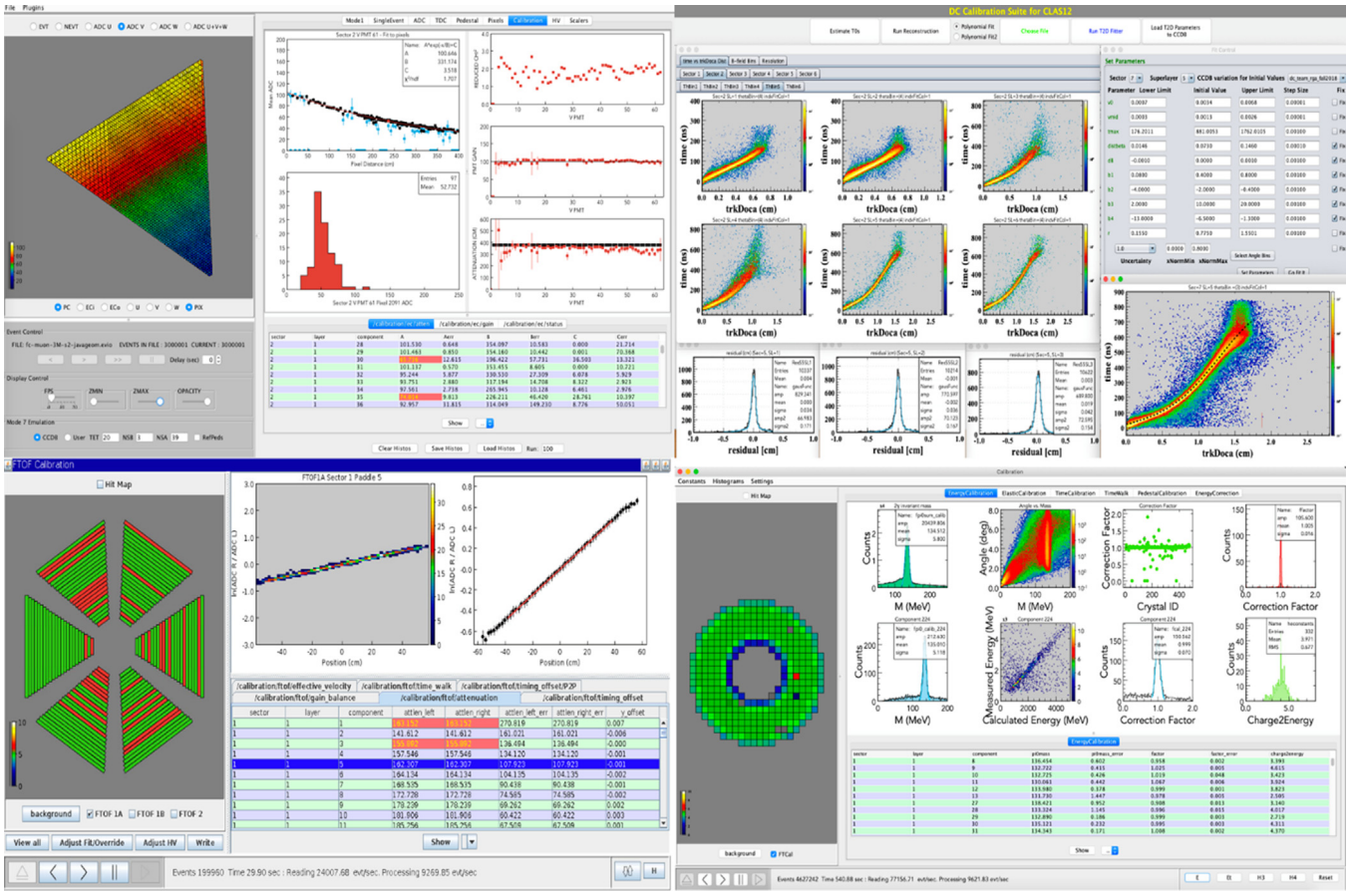

Fig. 5. Representative subsystem calibration GUIs for the Electromagnetic Calorimeter (ECAL) [7] (upper left), Drift Chambers (DC) [2] (upper right), Forward Time-of-Flight (FTOF) [6] (lower left), and Forward Tagger (FT) [12] (lower right).

data format was designed to provide a flexible data container structure, with features that minimize disk access for the most common tasks performed in data analysis. The High Performance Output (HIPO) format developed for CLAS12 was designed to provide data compression, using LZ4 (the fastest compression algorithm currently available), and random access.

HIPO stores data in separate records (with adjustable size), with tags associated with each record. Each record is compressed and a pointer to the record is kept in the file's index table. This feature allows separating events during reconstruction based on the content of the event, such as the number of reconstructed particles. Users can read portions of the file depending on the final states to be analyzed. The metadata of the file, describing detector and beam conditions, are common for all analyses.

The HIPO library has both Java and C++ implementations. On the basis of the C++ implementation, a library was developed extending ROOT base classes to allow for HIPO files to be read from ROOT frameworks. Additional tools are available to allow users to produce plots using native ROOT syntax.

4. Monitoring and calibration suites

4.1. Framework

A calibration framework was developed to implement visualization software tools needed for all detector systems. Standard views were developed using the Java *Swing* application to visualize detector components and to provide call-back mechanisms necessary to display detector-component specific information. These software tools provide functionality for data fitting, plotting, and displaying using a Graphical User Interface (GUI) environment.

The calibration framework makes use of the other CLAS12 libraries (the geometry and plotting packages, as well as database utilities) and provides a uniform GUI for all calibration applications. The framework provides a data-processing interface and a calibration constant database interface used for online and offline data analysis.

A common data-streaming interface is implemented with software-level abstraction that allows the calibration and monitoring codes to run on all of the supported data formats used in CLAS12, including data read in real-time from the CLAS12 DAQ system [20].

4.2. Calibration and monitoring suites

The software programs used for the CLAS12 detector subsystem monitoring, as well as the energy and time calibrations, are Java-based suites that employ the framework discussed in Section 2.1. The software tools provided by the framework facilitate the development of detector-specific suites. Fig. 5 shows representative views of the CLAS12 subsystem calibration suites.

The calibration applications take as input raw or reconstructed data files (from either beam data or Monte Carlo simulations) in either EVIO or HIPO data formats. They display and fit the various quantities and histograms relevant to the extraction of the calibration constants. The calibration analysis parameters are saved into ASCII files with the same structure as the tables defined in CCDB. The constants are then reviewed and uploaded to the database using CCDB commands.

5. CLAS12 event display

The CLAS12 Event Display (*ced*) is a diagnostic graphical application for displaying CLAS12 events. The primary element of *ced* is the

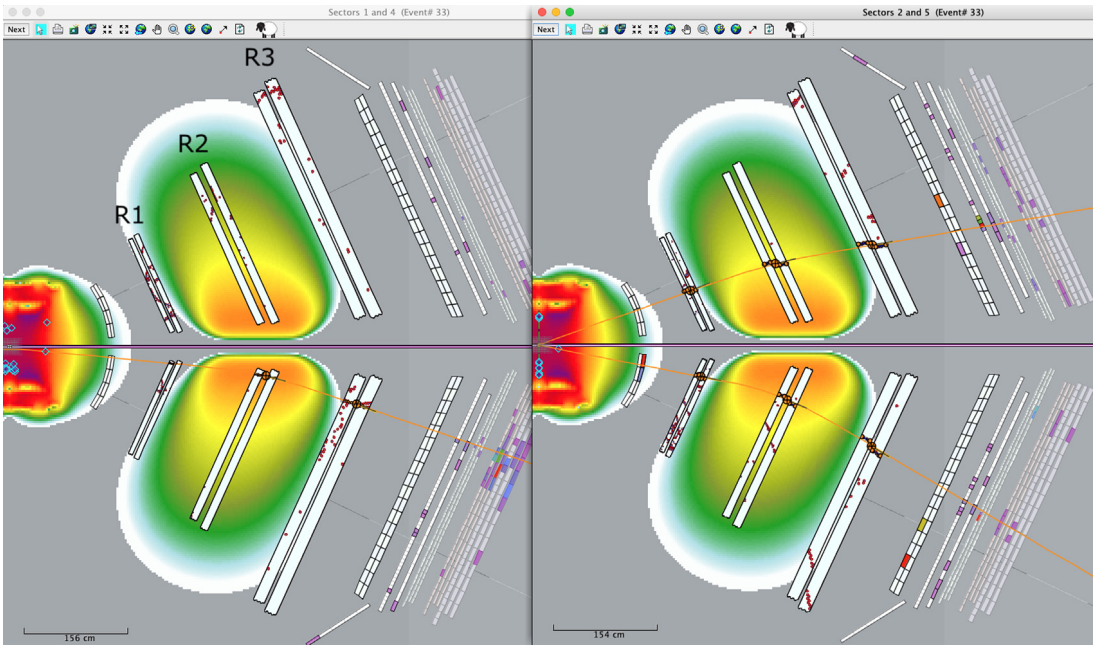

Fig. 6. Views from *ced* of charged particle tracks in the DC showing cut-views to highlight different pairs of sectors of the CLAS12 Forward Detector. The colored detector elements are the registered hits and the orange lines are the result of track reconstruction using the hits in the DCs. The colored areas about the detectors represent the regions of magnetic field from the torus and the solenoid. In these views the beam is incident from the left and the target is located in the middle of the solenoid (at the left edge of the image). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

"view", i.e. a graphical representation of CLAS12 in its entirety or a subset of its detector subsystems. For a given event, the primary purpose is to display the detector components that have recorded a signal, and, if available, the reconstructed tracks, to provide a visualization of the particle passage through the detector. In addition, *ced* can display information about the event such as the data banks, or information about the detector, such as the magnetic fields. Available views are both 2- and 3-dimensional with the possibility of disabling the latter for faster execution.

An illustration of views in *ced* is shown in Fig. 6, where a section of CLAS12 is displayed in a cut-view with a specific focus on the Forward Detector. The colored areas in the space around the detectors indicate regions where a significant magnetic field intensity is present from either the solenoid or torus; reconstructed tracks are shown by the orange lines. Similarly, Fig. 7 shows views of the Central Detector and of the Forward Tagger. Fig. 7(left) shows two tracks originating from the target as reconstructed from the fit of the available central tracker hits in correlation with signals in the outer detectors. Here, the color scale is representative of the recorded signal intensity. Fig. 7(right) shows a front view of the Forward Tagger calorimeter for an event where three clusters were recorded. *ced* is designed to be operated offline, reading either raw EVIO or HIPO events from a file, or online, reading events from the CLAS12 DAQ system [20] to allow for real-time monitoring of the detector during data taking.

6. Event reconstruction

The event reconstruction software has been designed and developed within the CLARA framework. As discussed in Section 2, the reconstruction of events for CLAS12 is separated into micro-services that execute data processing algorithms.

The data reader services access the detector decoded data stored in banks (see Section 2). Each entry for the decoded detector hits is a row in a bank. A row includes detector element identifiers (sector, layer, component, and order), and digitized detector data, such as signal charge, amplitude, time, or pedestal, depending on the specific system. Similar bank structures are created at the decoding stage for the various quantities needed for event reconstruction, such as hits, clusters, tracks, etc. The micro-services that implement the reconstruction algorithms pertaining to the CLAS12 subsystems fill these banks, which are subsequently appended and written out to a file by a data-persistency micro-service.

The services running the reconstruction algorithms access the various banks (transient data) as input and produce output banks needed for the subsequent algorithms in the reconstruction chain. The order in which the services are chained reflects the overall CLAS12 event reconstruction sequence and subsystem dependencies. First, charged particle tracks are reconstructed in both the Central and Forward Detector tracking systems based on the position of the recorded hits in the different detectors (i.e. using strip positions or wire locations). This procedure is referred to as "hit-based" tracking. In parallel, hits recorded in the other detectors are processed to reconstruct the energy and time of the associated particle interaction. These are matched to the reconstructed tracks by the Event Builder service, based on hit position and time information; unmatched hits are retained as neutral particle candidates. At this stage, the Event Builder can reconstruct the event "start time", i.e. the time of the interaction between the beam and target, and identify the reconstructed particles. Once the event start time is determined, a second iteration of forward tracking can be performed to implement the so-called "time-based" tracking (which also incorporates the drift times in the Drift Chambers). See Section 6.2.1 for more details on hit-based and time-based tracking.

The improved particle tracks from time-based tracking are the input for a second pass of the Event Builder, which leads to the final event reconstruction. Given this sequence, some services can run in parallel, while others need the reconstruction output provided by the preceding steps. For instance, hit-based tracking for the Central Vertex Tracker (CVT) using the CVT service and for the Drift Chambers using the DCHB service ("HB" is for hit-based) can run in parallel, while time-based tracking for the Drift Chambers using the DCTB service ("TB" is for time-based) must come after the first execution of the Event Builder service. An overview of the reconstruction application service composition detailing these dependencies is shown in Fig. 8.

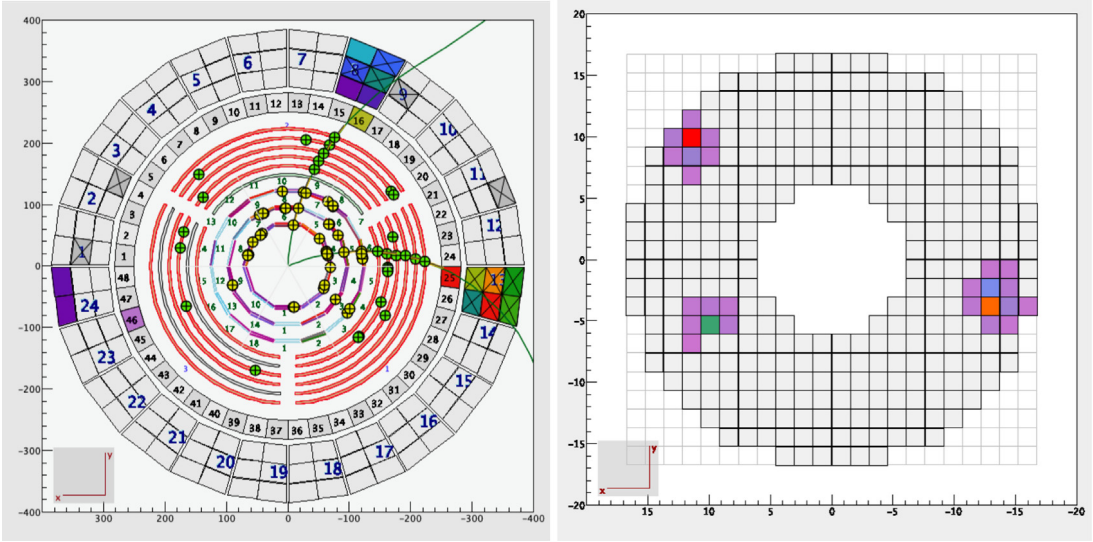

Fig. 7. Views from *ced* of the Central Detector (left) and the Forward Tagger (right) from a view looking down the beamline. In the Central Detector view (left), two tracks originating from the target are shown as reconstructed from the fit of the available central tracker hits in correlation with signals in the outer detectors (Central Time-of-Flight (CTOF) and Central Neutron Detector (CND)). Here the color scale is representative of the recorded signal intensity. The right figure shows a front view of the Forward Tagger calorimeter for an event where three clusters were recorded.

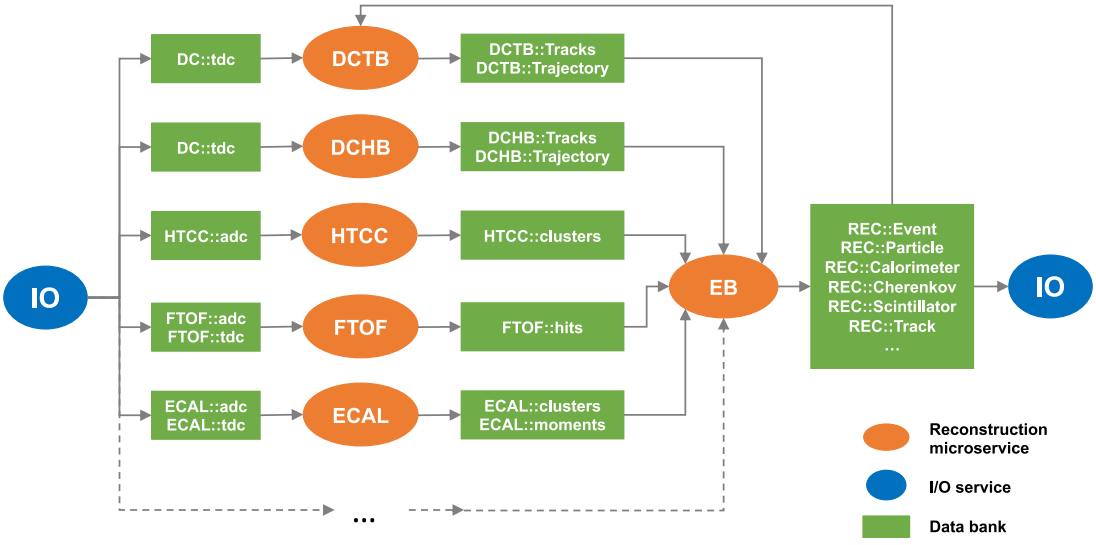

Fig. 8. Graphical representation of the CLAS12 interdependencies between services and banks. The I/O service reads events from the input file and distributes them to the reconstruction services chain for processing. Each service reads the relevant banks, applies the reconstruction algorithm, and provides output banks that are passed to the next service in the chain. The Event Builder (EB) service is executed as last in the chain; it collects the relevant banks from all CLAS12 subsystems services and produces event, particle, and detector response banks that are written to the output file.

6.1. Tracking overview

Charged particle tracking is the key element of the CLAS12 event reconstruction. It is separated into the reconstruction of tracks in the central tracker system (comprised of the Silicon Vertex Tracker — SVT [8] and the Barrel Micromegas Tracker — BMT [9]; together the SVT and BMT comprise the Central Vertex Tracker — CVT) and the forward tracking system (comprised of the Forward Micromegas Tracker — FMT [9] and the Drift Chambers — DCs [2]). In the forward region a torus magnet bends charged particles inward toward the beamline or outward away from the beamline depending on their charge. At full nominal current the $\int Bdl$ varies from roughly 2 Tm at 5° to 0.5 Tm at 40°. In the central region a 5 T solenoidal magnetic field bends charged tracks into helices. A view of the field intensities in the (z, x) plane and overlap region for the torus and solenoid fields is shown in Fig. 6.

For both systems, track reconstruction comprises algorithms for pattern recognition and track fitting. Hit objects, corresponding to the passage of a particle through a particular detector component, require the transformation of an electronic signal into a location of the track's position in the detector subsystem geometry. A hit is defined as a detector element represented by a geometric object, for example, a line representing a strip in the central tracker. These objects then form the input to the pattern recognition algorithms. This first step involves the identification of clusters of hits and the determination of the spatial coordinates and corresponding uncertainties for the hits and clusters of hits. At the pattern recognition stage, hits that are consistent with belonging to a trajectory (i.e. a particle track) are identified. This set of hits is then fit to the expected trajectory with their uncertainties, incorporating the knowledge of the detector material and the detailed magnetic field map.

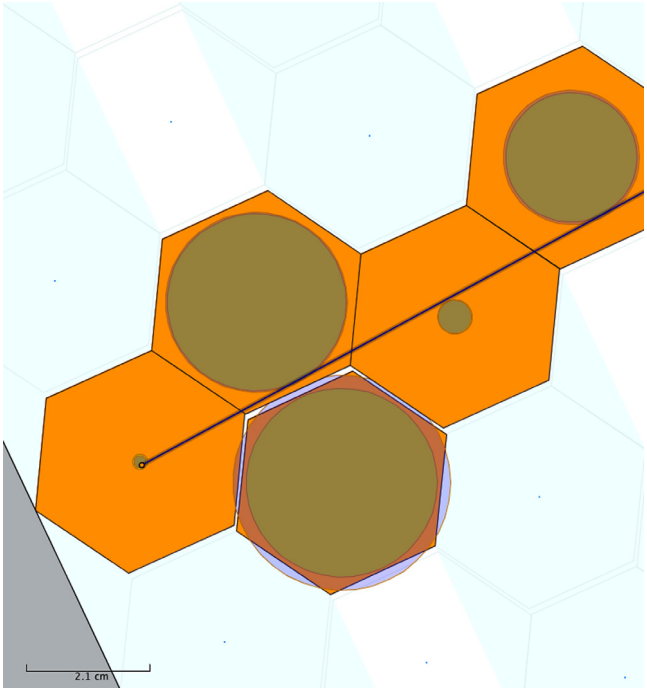

Fig. 9. Illustration of time-based tracking through a portion of a DC superlayer using the determined distance of closest approach to each sense wire indicated by the circles about the sense wires.

6.2. Forward tracking

6.2.1. Hit reconstruction

The Drift Chamber (DC) wire [2] hit information is given by the wire geometrical location and the drift time to the wire. Track-dependent corrections to the hit, such as the left–right ambiguity (to determine on which side of the sense wire the track passed) and time-walk (to account for the shift in time as a function of signal strength) must then be performed. Pattern recognition for the DCs is initially done using only wire position information and searching for groups of hits that form clusters. This portion of the algorithm is called hit-based tracking. In hit-based tracking, a hit is defined as a wire with a recorded signal. No timing information is incorporated at the preliminary stage of the reconstruction. After a hit-based track has been found, corrections to the raw times of the hits on the track resulting from the propagation time along the hit wire, the particle time of flight, the event start time, and the cable delays are applied to determine the corrected hit time. A distance of closest approach (DOCA) to the hit wire is estimated from the time. At this stage the tracking is redone using the calculated DOCAs in order to fit the track (see Fig. 9). This portion of the DC reconstruction phase is called time-based tracking. The calibration parameters entering in the function used to convert time to distance (see Ref. [2]) are extracted from the distance of local fits to the DOCAs using a linear function to the wire position.

In hit-based tracking, uncorrelated hit noise in the DCs is identified by a Simple Noise Removal (SNR) algorithm. Hits that are identified as noise are discarded from the list of hits passed on to the clustering algorithm. There are 112 sense wires in each of the 36 layers in each of the six Forward Detector sectors. The SNR stores all 112 wires for a given layer bit-wise in an extended 128-bit word, with "set" bits corresponding to hits. The extended words are objects that provide normal bit-wise operations on words of arbitrary (multiple of 64) length. The algorithm is configured through parameters specifying the maximum tilt of a track segment and the number of missing layers allowed in the formation of a segment. Using bit-wise operations on the extended words, the algorithm essentially operates as a parallel processor on all 112 sense wires in a layer. This parallelism precludes the need of a wire for-loop, which enables the algorithm to run in a negligible fraction of the total time for reconstruction. More to the point, the SNR actually saves time by reducing the combinations that must be explored in the pattern-recognition phase of the ensuing track-finding. An illustration of the SNR hit categorization in the DC is shown in Fig. 10.

6.2.2. Hit clustering

Within each of the six sectors of the CLAS12 Forward Detector, there are three sets of DCs that are referred to as Region 1 (R1) upstream of the torus, Region 2 (R2) within the torus coils, and Region 3 (R3) downstream of the torus (see Fig. 6). Each of the three detectors in each sector, R1, R2, and R3, consists of two so-called "superlayers", each containing six layers of 112 drift cells (or 6 wire layers). The hits remaining after the SNR algorithm are grouped into clusters. Clusters are made up of adjacent hits within the wire layers of a given DC superlayer. There can be at most two neighboring hits within a single wire layer, forming a "double-hit".[1] However, up to two wire layers can be missing within a superlayer when attempting to form a cluster. This is to reduce tracking inefficiencies resulting from possible wire malfunctions or intrinsic inefficiencies. It was found that requiring 4 out of 6 wire layers within a superlayer to form a cluster is sufficient to determine the cluster shape, which is subsequently used to find the track trajectory.

Additional "noise rejection" algorithms are applied to the clusters to remove spurious hits that do not come from a real track. So-called "curler" patterns as shown in Fig. 11 are typical for low-energy electrons in the DC. Therefore, a pruning algorithm was designed to remove them at an early stage of the reconstruction. The algorithm is a counting method of the number of contiguous hits within a single wire layer of a superlayer. In Figs. 11 and 12 we also see another typical noise pattern that looks like horizontal "strings" of hits along a wire layer. An algorithm was developed following the observation that high-momentum tracks from hadrons typically cross the superlayers at a large angle, while "curlers" from low-momentum background follow curling trajectories, with a significant part of the pattern lying within a single wire layer. Subsequent algorithms are employed for resolving overlapping segments.

Overlapping segments are produced when the trajectories of two tracks cross each other or when the tracks are almost parallel and very close to each other in a given region. A Hough Transform is employed to find hits on a line in the cluster, which allows the cluster to be split into segments. The resulting trimmed clusters are then fit to a straight-line hypothesis, and those hits with acceptable residuals are kept and identified collectively as a "track segment". An illustration of the Hough Transform cluster selection algorithm is shown in Fig. 13.

Subsequent hit pruning algorithms are employed at the time-based level. Fig. 14 illustrates the selected hits belonging to a cluster (orange) and the hits rejected by the noise-finding algorithms. In the zoomed view displayed in this figure, the cluster shown on the first superlayer illustrates the hit pruning algorithm and the remaining segment, while the rejected hits in the second superlayer are an example of a "looper" identified by the looper search algorithm.

6.2.3. Pattern recognition

Fits to the segments with a linear function are a preliminary step to estimating a track trajectory. The track parameters are estimated in the local coordinate system of the DCs from this trajectory.

Using the wire direction in a given superlayer along with the line fit to a segment in that superlayer, a plane can be constructed. Thus pairs

[1] An additional hit in a layer is due to noise coming either from an out-of-time hit that has a drift time that when converted to a drift distance exceeds the cell size, or hits not belonging to the track.

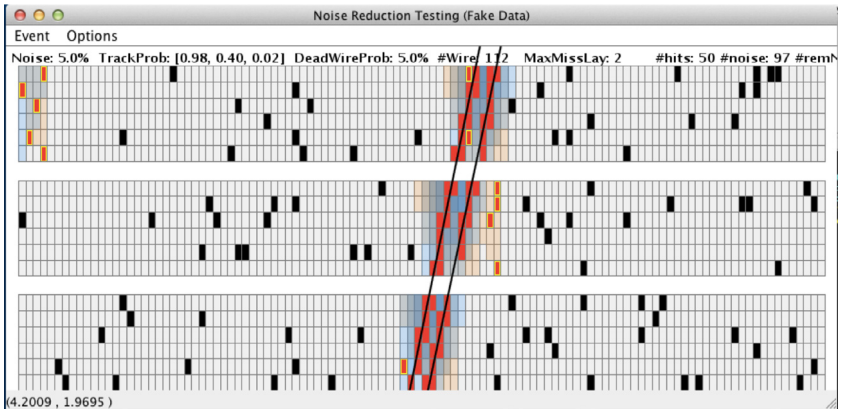

Fig. 10. Illustration of DC hits categorized by the SNR algorithm. This plot of wire layer vs. wire number shows three DC superlayers. The black hits are identified as noise and discarded and the red hits are saved for further evaluation by the subsequent hit selection algorithms. The orange hits are saved noise (false alarms) and the shaded areas correspond to possible clusters. The darker shades correspond to a higher quality factor, hence a higher probability for hits on a track.

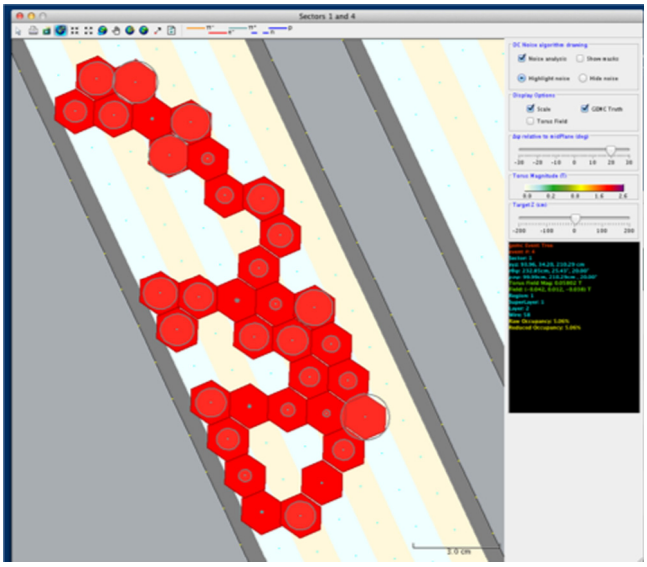

Fig. 11. Illustration of typical curler noise patterns in a single six wire layer superlayer in the DC displayed as seen using the CLAS12 Event Display *ced*. The hits shown are from a Monte Carlo electron event.

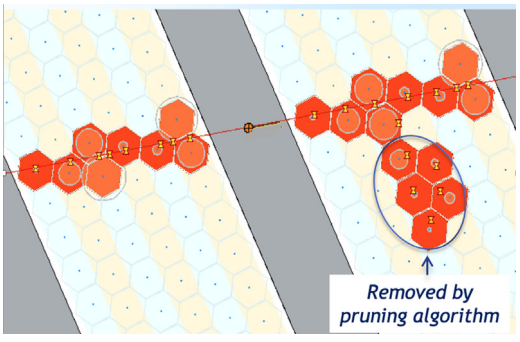

Fig. 12. Illustration of hits rejected by the pattern recognition pruning algorithm in a Monte Carlo electron event. The circles superimposed on top of the DC cells indicate the DOCAs computed from the fully corrected times. The group of hits encircled is removed by the pruning algorithm.

of segments in neighboring superlayers within one chamber (with superlayers of ±6° stereo angle) represent the intersection of two planes, which is a line whose coordinates are evaluated midway between the

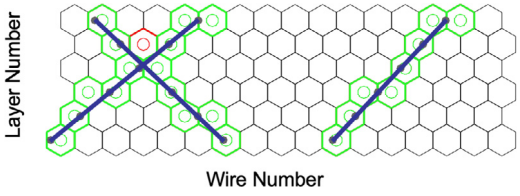

Fig. 13. Illustration of selected clusters (left-most selected hits with superimposed lines) using a Hough Transform. Two track segments cross each other. The right-most group of hits are selected using the nominal clustering algorithm. The hits are separated into cluster candidates and fit using a local coordinate system as a function of layer and wire number. The selection is done without employing timing information.

two superlayers, and is a 6-dimensional object (x, y, z, and 3 angles) that we call a "cross". A segment slope coincidence algorithm is used to match neighboring segments in a region (see Fig. 14). Selection cuts are subsequently applied on the reconstructed cross to ensure that it is within the detector fiducial volume within resolution.

There are instances when an entire superlayer can be missing from the list of hits available to fit a track. This can happen when inbending tracks are produced at low angle and miss the last superlayer of the chamber or when a segment has fewer than four valid (not out-of-time) hits. Therefore, in order to compensate for tracking inefficiencies due to this, an additional pattern recognition algorithm was designed. The algorithm matches segments within the even and odd numbered superlayers in a given sector, respectively. The matching algorithm returns an estimate of where the missing superlayer's hits should be and forms a "pseudo-segment" from the wire locations corresponding to these hits. Subsequently a "pseudo-cross" is formed using the pseudo-segment and the neighboring reconstructed segment in that region.

The first stage of pattern recognition consists of finding a track candidate from a set of 3 crosses (one each in R1, R2, and R3) that are fit to a parabolic functional form to give a "track candidate". Using the parameters of the parabolic function between the first and the third cross and obtaining the magnetic field intensity at each step along this trajectory, we obtain an estimate for $\int Bdl$. From the local angles of the crosses in the x–z plane for R1 (θ_1) and R3 (θ_3), we estimate the track momentum (p) in GeV and the particle charge (q) as:

$$\frac{q}{p} = \frac{\theta_3 - \theta_1}{\upsilon \int Bdl}, \tag{1}$$

where the angles are in radians, the magnetic field intensity (B) is in Tesla, and the path length (dl) is in cm. The conversion factor $\upsilon = 0.002997924580$ (GeV/c) $\mathrm{T}^{-1}\mathrm{cm}^{-1}$ corresponds to the speed of light. The cross position and angles in DC R1, together with the momentum and the charge, provide all of the necessary information to define the

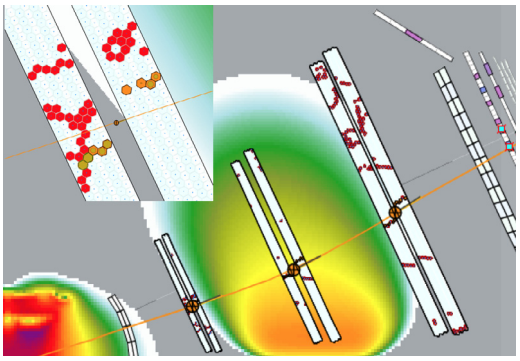

Fig. 14. Illustration of rejected hits (red hexagons) and accepted hits (orange hexagons) by the forward tracking pattern recognition algorithm using Monte Carlo data. The filled circle between the superlayers of a given region (R1, R2, or R3) represents the 3D point (called a "cross") obtained from the local fits to the DOCAs taking into account the direction along the wires. The track trajectory is projected at the $y = 0$ plane in this 2D view. The fitted track trajectory is represented by the orange line. The upper figure is a zoomed view into the track trajectory in R1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

track parameters at a given location in the detector, and therefore to start the track fitting.

6.2.4. Track fitting

The output of the pattern recognition is a seed with initial parameters used to start the track propagation from one measurement site to the next in the fit. The track fitting uses a Kalman Filter method with a 5-parameter track representation (x, y, tx, ty, Q) called the track state vector, defined in a local coordinate system with the z-axis perpendicular to the DC wire planes. Here, $Q = q/p$ (with q corresponding to the track charge), $tx = p_x/p_z$, $ty = p_y/p_z$, and p_x, p_y, and p_z represent the (x, y, z) components of the track momentum in the local coordinate system. In the analysis frame the state vector and the measurement are defined at each layer for which there is a hit on a track. Hence, as in Ref. [21], we can express the equations of motion of the track in the torus field and the propagation of the state vector covariance matrix as derivatives with respect to z. In the DC, the magnetic field components are mostly along the y coordinate (along the wires) in the analysis frame. The trajectory of the particle in the analysis frame is given by:

$$\begin{aligned} dx/dz &= tx, \\ dy/dz &= ty, \\ dtx/dz &= Q \cdot v \cdot \sqrt{1 + {t_x}^2 + {t_y}^2} \\ &\quad \cdot \; [t_y \cdot (t_x B_x + B_z) - (1 + {t_x}^2) B_y], \\ dty/dz &= Q \cdot v \cdot \sqrt{1 + {t_x}^2 + {t_y}^2} \\ &\quad \cdot \; [-t_x \cdot (t_y B_y + B_z) + (1 + {t_y}^2) B_y], \\ Q &= Q_0, \end{aligned} \tag{2}$$

where the initial values at the starting point $z = z_0$, corresponding to the measurement vector z-component at a give measurement site, are $x = x_0$, $y = y_0$, $t_x = t_{x0}$, $t_y = t_{y0}$, and $Q = Q_0$. The state vector is initialized at the first measurement layer.

The above equations are solved numerically using a fourth-order Runge–Kutta integration method in order to propagate the state vector from the DC plane at z_0 to the next one at z. The state vector covariance matrix is propagated along with it by computing the Jacobian matrices as in Ref. [21], again solving using a fourth-order Runge–Kutta method. The Jacobian matrix terms contribute to the propagator matrix used to compute the Kalman gain. The propagated covariance matrix takes into account multiple scattering through the known material layers of the DC tracking volume.

The non-zero components of the multiple scattering matrix are:

$$\begin{aligned} Cov(t_x, t_x) &= (1 + {t_x}^2) \cdot (1 + {t_x}^2 + {t_y}^2) \cdot {\theta_0}^2, \\ Cov(t_y, t_y) &= (1 + {t_y}^2) \cdot (1 + {t_x}^2 + {t_y}^2) \cdot {\theta_0}^2, \\ Cov(t_x, t_y) &= t_x t_y \cdot (1 + {t_x}^2 + {t_y}^2) \cdot {\theta_0}^2, \end{aligned} \tag{3}$$

where,

$$\begin{aligned} \theta_0 &= \frac{13.6}{\beta pc} \sqrt{\frac{l}{X_0}} \sqrt{1 + {t_x}^2 + {t_y}^2} \\ &\quad \times \left[1 + 0.038 \; \ln \left(\frac{t}{X_0} \sqrt{1 + {t_x}^2 + {t_y}^2} \right) \right] \end{aligned} \tag{4}$$

as given by the Highland-Lynch-Dahl formula [22]. The radiation length X_0 is computed as an effective radiation length corresponding to the gas mixture in the DC wire layer. Air is assumed outside of the DC volumes. The term l represents the path length traversed by the track.

At each plane the state vector is mapped onto a measurement, which corresponds to the drift distance to the wire in a given DC plane. In instances where there are two hits associated with the track in a given wire layer (i.e. the track goes in between the wires), the information from both hits is included in the fit. The measurements used in the fit take into account the left/right position of the track with respect to the wire.

After the times are corrected, the drift distance is computed using tabulated distance-to-time multi-dimensional arrays. The drift distances are computed using a multi-dimensional interpolation method using the segment local angle (i.e. the entrance angle of the track in the cell), the value of the magnetic field at the location of the hit, and the corrected times. The Kalman fit is redone at the time-based level using the hits with corrected times and the computed drift distances. A graphical representation of tracks in *ced* is shown in Fig. 6. This is a typical event for the nominal running conditions of CLAS12.

After the last iteration of the Kalman fit that propagates the state vector to the initial site (corresponding to the first layer in which there is a hit), the track parameters are transformed into the lab frame and the track is swam through both the torus and the solenoid fields to the distance of closest approach to the beamline. The track parameters defined in the lab frame (x, y, z, px, py, pz) are reported at this location. A final track propagation from the reconstruction vertex point at the distance of closest approach to the beamline is performed to obtain the trajectory of the track as a series of points and path lengths corresponding to its intersection with all the detector surfaces. This is used for subsequent matching of the track to the detector responses.

In order to improve the accuracy of the vertex reconstruction at the distance of closest approach to the beamline, another tracking device was placed just downstream of the solenoid. This device is the Forward Micromegas Tracker (FMT) [9], which consists of 6 layers of Micromegas detectors and covers the polar angle range from 5° to 35°. Integration of this system in the reconstruction is currently ongoing. The reconstruction algorithms in place consist of clustering of Micromegas hits corresponding to active adjacent strips, taking into account the Lorentz angle correction and energy weighting, and of matching of the clusters to tracks found in the DC. The challenging aspect of this reconstruction task is the combining of the track representation in two different frames for the DC (tilted sector frame) and the natural frame of the FMT, which is the frame where the FMT disks are perpendicular to the beam axis.

6.3. Central tracking

Tracks whose polar angle is between 35° and 125° are reconstructed by the Central Vertex Tracker (CVT). The CVT consists of twelve cylindrical layers of tracking detectors, numbered from 1 for the innermost layer to 12 for the outermost layer. The subset of tracking detectors forming layers 1 to 6 are silicon strip sensors within the CLAS12 Silicon Vertex Tracker (SVT) [8]. Layers 7 to 12 are made of Micromegas

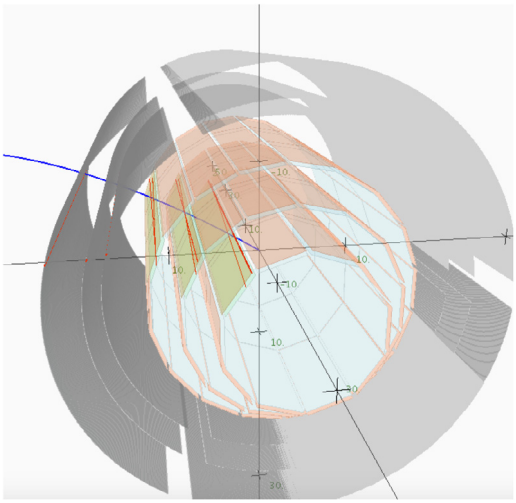

Fig. 15. Event display view of the CVT detector showing the 3 inner double layers of the SVT (in red) and the 3 innermost BMT layers (in gray). The red lines in the upper left of the SVT in this view represent active SVT strips corresponding to hits on a track. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

tiles within the Barrel Micromegas Tracker (BMT) [9]. The entire CVT surrounds the target and sits in the 5 T solenoid field. The SVT is made from 3 concentric rings of double-layer silicon sensors with a graded strip stereo angle from 0° to 3° (with 0° along the beamline z-axis) and a readout pitch of 156 μm. The BMT consists of 3 cylindrical detectors with strips along the z-axis (called the BMT-Z layers) and 3 cylindrical layers with circular layers with circular strips perpendicular to the z-axis (called the BMT-C layers). Each layer is divided into three 120° sectors.

The revolution axis of the CVT coincides with the ideal beam axis, which defines the z-axis of the CVT. The y-axis points upward in the laboratory frame and the x-axis is defined to form a right-handed coordinate system. The origin of the CVT coordinate system matches the center of the nominal CLAS12 target center. An illustration of the CVT detector with *ced* is shown in Fig. 15.

6.3.1. Hit clustering

The first step of the tracking algorithm is the formation of clusters from the raw hits. A cluster is a collection of contiguous hit strips. Its centroid, calculated using charge weighting, is either given by spatial information (a z-coordinate for the BMT-C detectors in which strips are arcs at constant z or xy-coordinates for the BMT-Z detectors in which strips are parallel to the z axis) or strip numbers for the SVT. Charge weighting is done by averaging the relevant strip information using the maximum of the ADC pulse for the Micromegas strips or the equivalent deposited charge for the SVT. The time information associated with each hit is currently not used.

Before feeding all of the CVT clusters to a pattern recognition algorithm, spatial coordinates must be associated with the SVT clusters. As described in Ref. [8], the six SVT layers are mechanically paired and consequently form three regions. The readout strips of the inner and outer layer of each region make a 3°-stereo angle. By associating one cluster of the inner layer with one cluster of the outer layer, and by assuming that an infinite momentum track perpendicularly crossed the two layers, a preliminary assignment for the (x, y, z) coordinates of the particle between the two layers is derived for this cluster pair. This pairing is performed over all clusters of the inner layer with all clusters of the outer layer. Pairs whose (x, y, z) coordinates are outside of the physical SVT sensor space are automatically removed from the list of candidates. If one of the two layers of a region has no hit that can be associated with a track, then the information of the active layer is simply ignored for the remainder of the reconstruction process.

6.3.2. Pattern recognition

The trajectory of a charged particle in a solenoidal magnetic field is an helix. Because the BMT detectors offer either xy- or z-coordinates but never both, the pattern recognition cannot be performed in 3 dimensions. For particles of large enough momentum (perpendicular momentum $p_\perp > 0.25$ GeV for a 5 T solenoidal magnetic field), the xy-projection of a helix is a circle, and the rz-projection is a straight line (where $r = \sqrt{x^2 + y^2}$). Therefore, a first pattern recognition algorithm is run in the xy-plane to look for circles and then a second pattern recognition algorithm is run in the rz-plane to search for straight lines.

The two pattern recognition algorithms are a modified version of the cellular automaton (CA) algorithm developed by the HERA-B Collaboration [23]. Here, the elementary cell of the CA is defined as a segment that connects two 2D points. In the (x, y) plane, cells are formed with SVT and BMT-Z xy-information. Two xy-clusters form a cell if the angular distance between them is lower than a defined threshold. This threshold has been derived by maximizing the reconstruction efficiency on a single track Monte Carlo simulation merged with background extracted from the data. Two clusters cannot form a cell if they are separated by more than one layer. Finally, the CA is run sector-by-sector in the BMT and, as a consequence, a cell cannot be formed with two clusters residing in different BMT sectors.

The subsequent step is the "neighbor" finding. Cell "a" is a neighbor of cell "b" if they share one cluster and if the layer numbers in "b" are higher than those in "a". Tuned on single-track Monte Carlo simulation data without background, cuts on the dot product between the cell directions are applied as neighbor-forming criteria. Once the neighborhood of a cell is defined, the CA is evolved over an N-evolution stage. For evolution stage n, the state of all cells is updated according to $S_n = max(S^j_{n-1}) + 1$, where S^j_{n-1} is the state of the jth-neighbor of the considered cell at evolution time $n-1$. Therefore, at evolution stage N, the cells with the highest state are further outward than the cells with a smaller state.

Track candidates in the (x, y) plane are then formed starting from the highest state cells and following the neighbor chain with $\Delta S = 1$. In case of multiple neighboring cells with the same state, the one that has the smaller dot product with the original cell is chosen.

Since the z-resolution of the BMT clusters is significantly better than the uncorrected (i.e. prior to obtaining a track direction) SVT 3D points, the search for candidates in the (r, z) plane is performed by only using the BMT-C information. The CA algorithm returns the track segments of two or three BMT-C clusters. Due to the orthogonality of the BMT-C and BMT-Z readout, all of the (r, z) segments of a BMT sector are combined with the (x, y) candidates in the same sector. A line is fit to the BMT-C hits and its intersections with the three SVT regions are computed. If the distance between the expected intersection and the preliminary 3D point in the SVT region is greater than two millimeters, then the two SVT clusters forming this preliminary point are removed from the track candidate.

6.3.3. Track fitting

Each track candidate is then passed to a Kalman filter. The state vector to describe a helix is formed by five parameters $(\varphi_0, d_0, \kappa, z_0, \tan\theta_{dip})$, where:

- d_0 is the (x, y) distance of closest approach to the CVT revolution axis;
- $\varphi_0 = \text{atan}(p_y/p_x)$ at closest approach angle to the CVT revolution axis;
- $\kappa = q/p_\perp$ and q is the electric charge of the particle and $p_\perp = \sqrt{p_x^2 + p_y^2}$ is the transverse momentum;
- z_0 is the distance along the z axis to the CVT center;
- θ_{dip} is the polar angle between the track and the xy-plane.

To initialize the Kalman Filter, a first estimate of these parameters is obtained from:

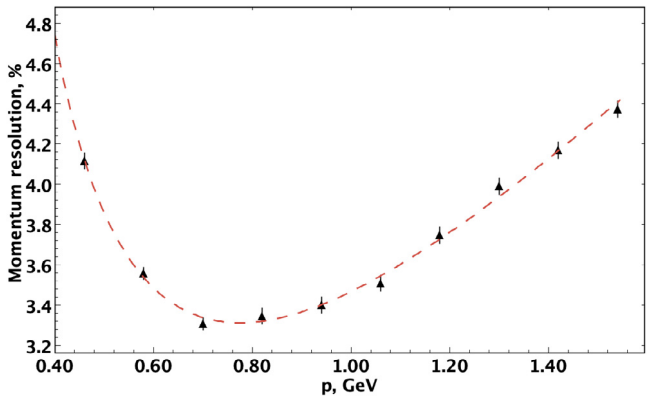

Fig. 16. Momentum resolution vs. momentum of simulated protons in the CVT without background.

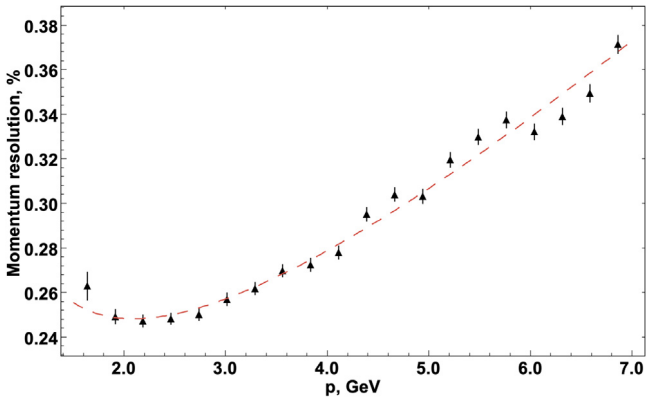

Fig. 17. Momentum resolution vs. momentum in the DC evaluated using pions simulated at $\theta = 15° \pm 5°$ and at $\phi = 0 \pm 5°$ without background.

- a circle fit in the xy-plane with preliminary SVT 3D points and BMT xy-clusters for d_0, φ_0, and κ. To improve the initialization of the fit, the point (0,0) (on the ideal beam z-axis) is included in the fit with an accuracy of 100 μm.
- a line fit in the rz-plane using only the z-clusters of the Micromegas to initialize z_0 and θ_{dip}.

The covariance matrices of the two fits are merged into a 5×5 matrix to initialize the covariance matrix for the Kalman filter. Following the transport equations in Ref. [24], the state vector is propagated from the CVT revolution axis to the outermost layer of the CVT, filtering at each measurement composing the track candidate. Once the last measurement is reached, the state vector and covariance matrix are brought back to the CVT revolution axis as they are and the transport/filtering process is re-run. A maximum of five iterations is performed to make sure of the convergence of the filtering process.

6.4. Tracking performance

The momentum resolutions in the central and forward trackers as a function of momentum are shown in Figs. 16 and 17, respectively. The distributions are fit with a function of the form $\sqrt{a + bx^2 + c/(1 + d/x^2)}$. In both distributions, the worsening of the resolution at low momentum is due to multiple scattering effects. The resolution also worsens as a function of momentum after a minimum is reached due to poorer track curvature resolution. The resolutions achieved are well within the design specifications and the difference in magnitude between the central and the forward trackers is due to the intrinsic resolutions of these systems

For central tracking, an average CVT reconstruction efficiency of 87.3% is obtained from a simulated proton sample with momenta in the range from 0.5 to 2.5 GeV. A slight drop of efficiency is observed for tracks with momenta less than 600 MeV. The higher curvature of small $p_\perp$ tracks results in an increase in inefficiency due to acceptance effects. The dominant source of inefficiency is the gaps between the sensitive volumes for the BMT and the SVT. These effects can be observed in the efficiency plots of Fig. 18.

For the forward tracking, the momentum resolution in the DC is evaluated using tracks simulated at $\theta = 15° \pm 5°$ and at $\phi = 0 \pm 5°$ (sample 1), to ensure that most tracks are within the sensitive volume. Furthermore, the DC momentum resolution is correlated with the polar angle since the track curvature is determined from the magnetic field intensity, which is higher at lower angles in the torus field, as can be seen from Fig. 19, corresponding to tracks simulated at $p = 4 \pm 1$ GeV, $10° \leq \theta \leq 25°$, and $\phi = 0 \pm 5°$ (sample 2).

These resolutions are obtained from a Monte Carlo sample that does not include out-of-time backgrounds or misalignments of the

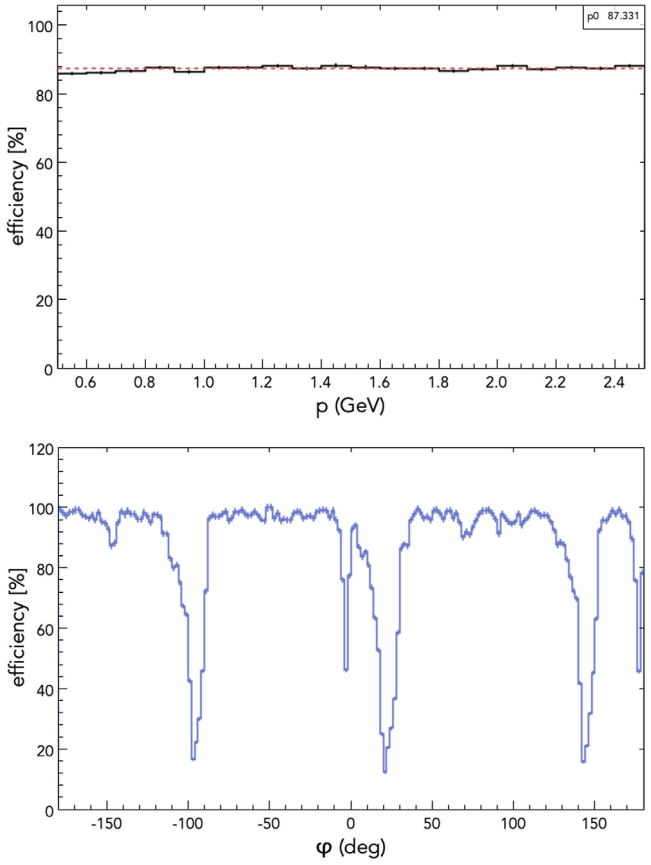

Fig. 18. Reconstruction efficiency vs. momentum (top) and azimuthal angle (bottom) of simulated protons in the CVT without background from a sample of simulated protons.

tracking volumes. A dedicated study that involves merging random background data with low-luminosity data is described in Ref. [1]. The tracking efficiency for inbending (negatively charged) and outbending (positively charged) pions in the torus field calculated from sample 1 is shown in Fig. 20 for tracks at $\theta = 15° \pm 5°$. Inbending tracks suffer from a loss in tracking efficiency for momenta generated below 1.8 GeV at the time-based level due to lack of matching with the outer detectors. These tracks miss the sensitive volumes of the Forward Time-of-Flight (FTOF) system, which is required to extract the time-correction information needed for time-based tracking. The tracks do however pass the hit-based tracking requirement. The efficiency loss

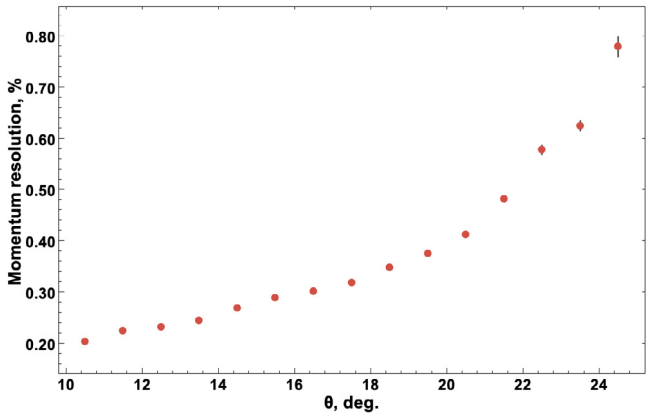

Fig. 19. Momentum resolution vs. polar angle in the DC evaluated using pions simulated at $p = 4 \pm 1$ GeV, $10° <\leq \theta \leq 25°$, and $\phi = 0 \pm 5°$ without background.

due to the aforementioned effect can be seen by comparing the (light) blue to the (dark) red distributions. In the momentum range from 1.8 to 7.5 GeV, the time-based tracking efficiency is 98%, while in the range from 1.4 to 7.5 GeV, the hit-based tracking efficiency is 99%. For outbending tracks (see Fig. 20(bottom)), both the hit-based and time-based tracking efficiencies are flat as a function of momentum and on the order of 99%.

The polar angular dependence of the DC tracking efficiency obtained from sample 2 is shown in Fig. 21. The green histogram corresponds to outbending tracks. The efficiency is flat in the angular range from 10° to 25° for outbending tracks, while there is a loss of tracks below 15° for the inbending tracks (shown in orange). As discussed above, this is due to tracks missing the outer detectors.

The vertex resolutions of reconstructed tracks from a sample of simulated semi-inclusive deep inelastic scattering events are shown in Fig. 22. The vertex is obtained for positively and negatively charged tracks reconstructed in the Central and Forward Detectors, respectively. The vertex resolutions for the Central Detector (blue histogram) is about 3 mm and for the Forward Detector (red histogram) is about 5 mm.

6.5. Electromagnetic calorimeters

The Electromagnetic Calorimeters (ECAL) [7] of the CLAS12 Forward Detector downstream of the torus and the FTOF are lead-scintillator strip sampling calorimeters used for the detection of electrons, photons, and neutrons. A pre-shower calorimeter (PCAL) is positioned in front of the EC calorimeter, which consists of two parts, EC-inner (ECIN) and EC-outer (ECOU). The ECAL reconstruction service provides a fast and efficient algorithm for grouping scintillator *strips* with *hits* into multiple *peaks* and *clusters* within the three submodules, PCAL, ECIN, and ECOU, for each of the six ECAL modules, while leaving cluster matching and particle identification to the Event Builder service.

Within the ECAL reconstruction service, these various elements exist as objects with methods, structures, and data members designed for calibration, pattern recognition, diagnostics, and serial output. For example, the service applies run-dependent calibration corrections for conversion of the raw ADC and TDC digitized data to energy and time, and also provides formatted output banks used by external services. Energy thresholds and cluster identification criteria can also be configured to optimize the reconstruction efficiency, suppress backgrounds, and avoid false or duplicate clusters arising from fluctuations at the fringes of the electromagnetic showers.

The cluster finding algorithm makes use of the unique geometry and stereo readout features of the ECAL. As discussed in Ref. [7], each triangular scintillator layer in the ECAL lead:scintillator sandwich is

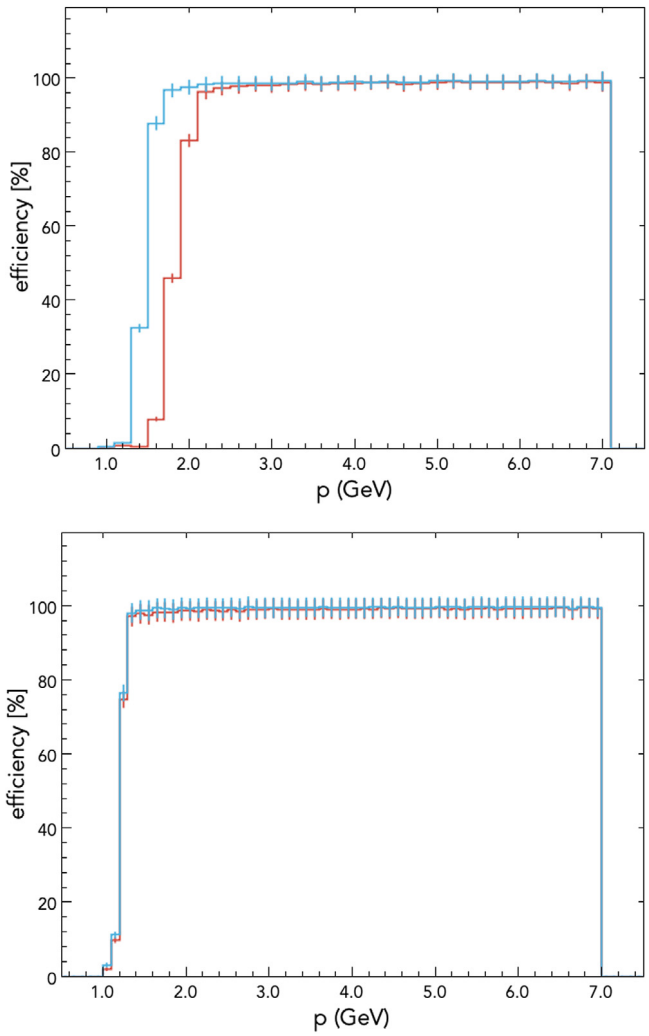

Fig. 20. DC tracking efficiency as a function of momentum evaluated using (top) negatively and (bottom) positively charged pions simulated at $\theta = 15° \pm 5°$ and at $\phi = 0 \pm 5°$. The (light) blue and (dark) red distributions correspond to the hit- and time-based tracking efficiencies, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

transversely divided into strips, with the shortest strip at the corners. The slice direction rotates by 120° for each successive layer, providing three *views* labeled U, V, and W. For each strip within a view, layers are optically ganged together into a stack. Individual photomultiplier tube (PMT) readout of each PCAL, ECIN, and ECOU stack provides a pulse proportional to the summed energy deposited in the stack.

The algorithm begins by finding collections of contiguous stacks having signals above a user-defined threshold for each of the three views. These groupings are called *peaks* and their member stacks are referred to as *hits*. Peak objects may be further subdivided based on the hit energy profile of the groupings. Each peak object is associated with one or more stacks of strips that belong to it, and the three-dimensional geometry of each stack is stored along with the peak data. The service uses this geometry data to determine which collection of peaks belong to *clusters*.

6.5.1. Cluster position

The criterion for defining a cluster requires the spatial intersection of three peaks, one from each of the U, V, and W views. Candidate peaks for a cluster search are based on a user-defined threshold for the summed peak raw energy. Each peak is represented geometrically as a directed line segment determined by the energy-weighted average

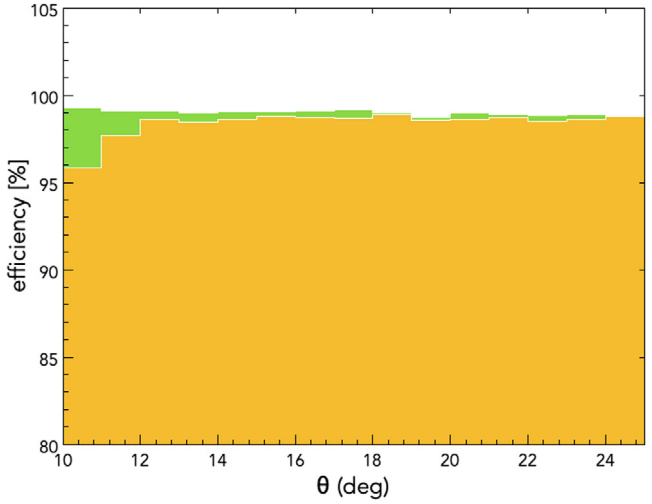

Fig. 21. DC tracking efficiency as a function of polar angle evaluated using pions simulated at $p = 4\pm1$ GeV, $10° \leq \theta \leq 25°$ and $\phi = 0\pm5°$. The green histogram corresponds to outbending tracks in the torus field and the orange histogram corresponds to inbending tracks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

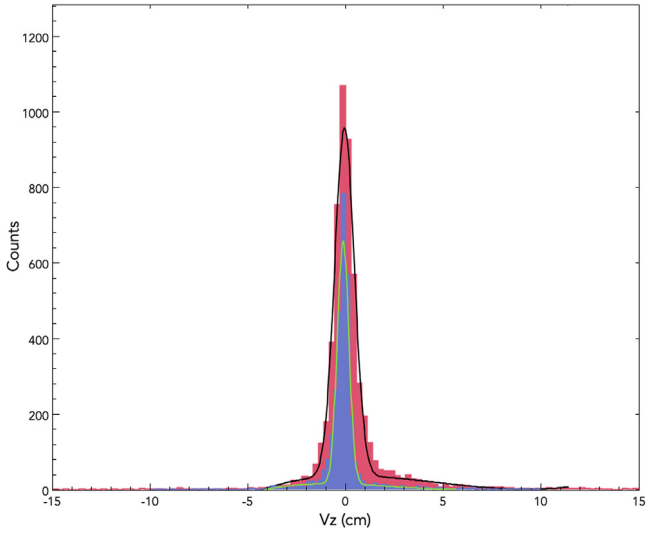

Fig. 22. The z (along the beamline) vertex resolutions of reconstructed tracks from a sample of simulated semi-inclusive deep inelastic scattering events. The vertex for positively and negatively charged tracks reconstructed in the Central and Forward Detectors is represented by the blue and red histograms, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of the mid-lines of each member strip. The degree of intersection of each U, V, and W peak triplet is determined by calculating the line of closest distance between a U and V peakline, followed by the line of closest distance between the midpoint of the UV line and the W peakline. A user-defined cut on this final UV-W distance identifies the cluster, and the midpoint of the UV-W line defines the transverse (x', y') coordinates of the cluster in the local coordinate frame (with the z' axis perpendicular to the ECAL planes). The longitudinal coordinate z' is set to coincide with the layer of maximum energy deposition to minimize parallax effects for tracks that are not perpendicular to the detector surface [7]. As the cluster reconstruction is performed before the matching with the other CLAS12 detectors that can provide particle identification, the same algorithm is applied to clusters originating from charged or neutral particles.

6.5.2. Cluster energy

Once the cluster is localized, the path from the cluster position to the PMT readout end is calculated for each U, V, W peakline and the peak energies are corrected for scintillator light attenuation. For isolated clusters, the cluster energy is then defined as the sum of the corrected energy from each of the U, V, and W peaks that define the cluster.

More complicated scenarios arise from the triangular geometry of the ECAL hodoscope, which creates the possibility of a single peak in the U, V, or W view that shares the summed energy from two or more clusters. For these cases, the energy in each cluster that shares that peak is assumed to be proportional to the relative partial energies of the multiple clusters as measured in the other views. For example, if there are two clusters, both of which share the same U peak, the summed energy $V + W$ is determined for each of the clusters, and the ratio of these summed energies determines how much of the U peak energy is assigned to each of the two clusters.

Finally, the clusters to be reported to external services are selected with a user-defined energy cut, and these clusters are sorted according to energy. Typical software thresholds applied at the stacks, peak, and cluster level are 1, 3, and 10 MeV, respectively.

6.5.3. Cluster time

Once the cluster is localized, the path from the cluster position to the PMT readout end is calculated for each U, V, W peakline and the peak timing is corrected for the propagation delay of the light, using the effective velocity of light determined for each scintillator from the calibration procedure. For isolated clusters, the cluster timing is then taken from the U, V, or W peak with the largest uncorrected raw ADC value. This minimizes the effect on the timing resolution from both the time-walk correction (i.e. the signal amplitude dependence of the hit time) and the photoelectron statistical fluctuations.

6.6. Threshold Cherenkov counters

The CLAS12 Forward Detector includes two threshold Cherenkov detectors for particle identification. The High Threshold Cherenkov Counter (HTCC) [4] is located upstream of the torus and is used for identification of the scattered electron in conjunction with the ECAL. The Low Threshold Cherenkov Counter (LTCC) [3] is positioned upstream of the FTOF and is used mainly to identify pions. Both the HTCC and LTCC are large gas-filled volumes (CO_2 for the HTCC, C_4F_{10} for the LTCC) with mirrors that direct light collection to the PMTs. The goal of the HTCC and LTCC reconstruction algorithms is to calculate the signal strength, time, and position from the raw ADC signals (read out with flash ADC boards — FADCs). The algorithm takes into account the properties of the HTCC and LTCC geometries, namely, the possibility for the signal from a single charged track to split into up to four mirrors. Hence, up to four separate signals (or hits) are produced. The final signal reconstruction is done in three steps: decoding, hit reconstruction, and cluster reconstruction. For each hit, the signal strength ($nphe_{hit}$ — the number of photoelectrons) is determined from the pedestal-subtracted integral of the FADC pulse and the associated time (t_{hit}) is determined from a fit of the position of the FADC signal threshold crossing time.

At the hit reconstruction stage, individual signals in terms of the ADC channels are converted into the number of photoelectrons ($nphe_{hit}$) for each hit using gain constants derived from the detector calibration and stored in CCDB:

$$nphe_{hit} = \frac{ADC}{gain}. \tag{5}$$

Geometry information on the PMT location is used to associate the angular coordinates (θ_{hit}, ϕ_{hit}) to the hit.

In order to reconstruct the real signal strength ($nphe_c$), split signals (hits) have to be combined into a single cluster. The algorithm starts by selecting the hit with the largest $nphe_{hit}$, which is used as a seed

for the cluster. Adjacent hits within a certain time window are then searched iteratively and, if found, added to the cluster. The total signal strength is determined as the sum of the individual signals, and the signal time is determined as the average between the individual signal times, weighted by the corresponding number of photoelectrons. The cluster angular coordinates are determined as the average between the individual hits forming the cluster. The cluster quantities are defined by:

$$\begin{aligned} nphe_c &= \frac{\sum_{i=1}^{N} nphe_{hit}}{N} \\ t_c &= \frac{\sum_{i=1}^{N} N * t_{hit}}{\sum_{i=1}^{N} nphe_{hit}} \\ \theta_c &= \frac{\sum_{i=1}^{N} \theta_{hit}}{N} \\ \phi_c &= \frac{\sum_{i=1}^{N} \phi_{hit}}{N}. \end{aligned} \tag{6}$$

The clustering algorithm is run iteratively until the full list of N hits is exhausted.

In the HTCC, the cluster coordinates, required for the matching of the hit with the reconstructed track in the Event Builder, are reconstructed by projecting (θ_c, ϕ_c) of the cluster on the surface of the ellipsoidal mirror of the detector. In the LTCC, an estimated cluster position is calculated based on a parameterization extracted from Monte Carlo simulations. The track that passes the closest to the cluster position is then chosen as the match for this cluster.

6.7. RICH detector

The CLAS12 Ring Imaging Cherenkov Counter (RICH) [5] presently replaces one LTCC counter in the Forward Detector (with a second RICH to be installed in the future replacing a second LTCC counter). When charged particles traverse the aerogel radiator in the RICH volume, Cherenkov radiation is emitted with a characteristic cone angle related to the particle velocity. These photons are distributed in a ring pattern that can be reconstructed by collecting the photons using mirrors and PMTs (see Fig. 23 for an example RICH event). The goal of the RICH reconstruction is to provide an estimate of the Cherenkov angle for each detected photon and intercepted particle track, to allow subsequent particle identification. This requires input from the Forward Detector tracking service, which defines the trajectory of particle tracks inside the detector and, in particular, the track intersection point and direction within the aerogel radiator and the photodetector plane composed of multi-anode PMTs (MaMPTs).

In the first phase, the RICH reconstruction identifies the cluster of hits produced by the charged particle in the sensor plane. In the second phase, the cross-talk signals are identified by means of an amplitude analysis (based on the time-over-threshold information) in conjunction with geometrical constraints, taking into account that a cross-talk hit should be in the proximity of a genuine hit. Finally, hits neither belonging to a cluster nor flagged as cross-talk are considered as Cherenkov photon candidates.

The photon path inside the RICH is reconstructed in two complementary ways, taking the middle point of the hadron trajectory inside the radiator as the emission point, and the hit pixel coordinates as the detection point. The first method uses an analytic formula that takes into account the refraction at the aerogel face and is only valid for directly detected photons. It provides an exact solution. The second method uses a ray-tracing algorithm that also takes into account the mirror reflections. It provides a numeric solution based on an iterative procedure. Both methods return the reconstructed Cherenkov angle in conjunction with the corresponding aerogel refractive index, which can vary slightly with respect to the nominal value due to the chromatic dependence on the unknown photon energy.

The relevant RICH components (aerogel, mirrors, MaPMT plane) are converted into ray-tracing planes or spheres where the photon can undergo refraction, reflection, or detection. Each ray-tracing element can be independently aligned. The alignment procedure uses as a benchmark the Cherenkov signal generated by electrons, as identified by the HTCC and ECAL. For these particles, the expected Cherenkov angle is given by the known particle momentum (from DC tracking) and mass. The position and orientation of the MaPMT plane is defined by minimizing the average distance that matches the RICH clusters to the charged tracks extrapolated to the MaPMT plane. Any other RICH component can be aligned with respect to the MaPMT plane by selecting the sub-sample of photons passing through that component. The alignment is done by minimizing the average distance between the ray-traced detection point (RdP) and the corresponding measured MaPMT hit over the selected sub-sample of photons.

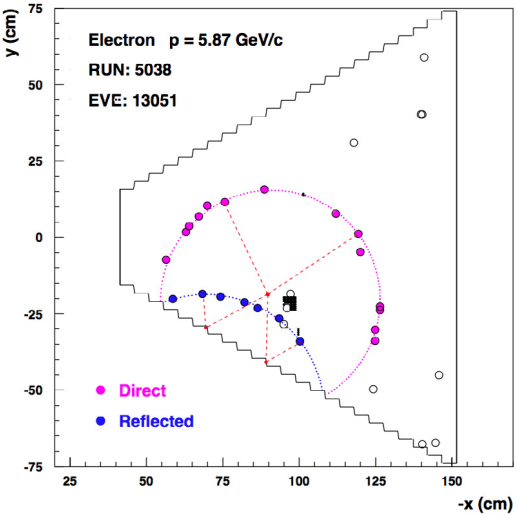

Fig. 23. Example of a reconstructed RICH event from beam data. Small points indicate the trial pattern expected for an electron, as identified by CLAS12. The dashed lines show examples of ray-traced photon paths from the common emission point (in the radiator) to the detected hit: two direct photons emitted upwards and two reflected photons emitted downwards. The open circles are the detected RICH hits. The circles are filled in the case that a viable traced solution has been found. The central cluster is generated by the track impact on the MaPMT plane.

For each hadron track, the ray-tracing algorithm progresses as described in the following. A trial photon is a hypothetical photon assumed to originate from the emission point at a Cherenkov angle θ_T and an azimuthal angle ϕ_T, with the corresponding RdP $T(\theta_T, \phi_T)$ defined by the ray-tracing algorithm. A limited ensemble (on the order of 100) of trials is initially traced having θ_T defined by a particle hypothesis, i.e. electron for a particle identified as an electron in CLAS12, pion otherwise, and ϕ_T uniformly distributed around the charged particle trajectory, see Fig. 23. For each MaPMT measured hit, the closest trial RdP is taken to be the starting point of the iterative ray-tracing procedure for that hit.

To initiate the iterative procedure, the closest trial RdP is required to stay at a distance from the hit smaller than 10 cm, which is twice the typical distance between the initial trial RdPs on the MaPMT plane. At each step, the closest trial is re-traced by varying its angles by the expected Cherenkov angle resolution σ to define the corresponding displaced RdPs $T_\theta(\theta_T + \sigma, \phi_T)$ and $T_\phi(\theta_T, \phi_T + \sigma)$. The distance vectors $\overrightarrow{TT_\theta}$ and $\overrightarrow{TT_\phi}$, connecting each rotated trial RdP to the initial trial RdP, naturally define a reference system in the MaPMT plane, see Fig. 24. The distance vector $\overrightarrow{TH}$ between the measured hit H and the closest trial RdP position T is projected onto the reference vectors to get an estimate of the next angular step. In particular, the scale factor f of the polar angle step $\Delta\theta = f\sigma$ is defined by projecting the distance vector $\overrightarrow{TH}$ onto the reference vector $\overrightarrow{TT_\theta}$: $f = (\overrightarrow{TH} \cdot \overrightarrow{TT_\theta})/|\overrightarrow{TT_\theta}|^2$. The factor f can be either positive or negative, depending on if the rotated point

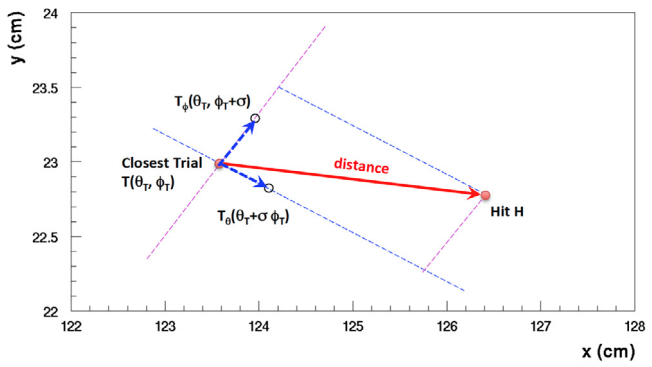

Fig. 24. Example of iteration of the ray-tracing photon path reconstruction in the RICH (x, y) plane. The emission polar θ_T and azimuthal ϕ_T angles of the closest trial photon are varied by the expected Cherenkov angle resolution σ to extrapolate the corresponding displacements of the detection point. The distance between the measured and the trial hit is projected onto such displacements to quantify the next angular step in units of σ. See text for details.

moves toward or away from the measured hit. The same is done for the azimuthal angle ϕ.

The angles of the trial photon are modified by the calculated $\Delta\theta$ and $\Delta\phi$ angular shifts, and the procedure is repeated. At each step, the trial RdP gets closer to the measured hit, but an exact solution cannot be found as the procedure uses a linear approximation relating the distances in the MaPMT plane with the angular rotations in the 3D space. The iterative procedure stops when the distance of the trial RdP from the measured hit is smaller than a fraction of the MaPMT pixel size, i.e. the RICH detector spatial resolution. The convergence is fast, typically within a few steps, so that the average reconstruction time of a RICH event is negligible, at the level of few tens of microseconds.

For each photon hit in the event, the RICH reconstruction procedure provides a measurement of the Cherenkov angle that does not depend on a given particle hypothesis, having as input only the emission point, the hit position, and the detector geometry. The initial particle hypothesis is only instrumental to define the starting ensemble of trials. As soon as a trial closer than 10 cm to the hit is found, the iterative procedure re-calculates the angles using only geometrical information and neglecting any previous assumption on the particle type.

As there is no a priori knowledge on which particle has emitted a given photon, the procedure is repeated for any charged particle intercepting the RICH radiator. The ensemble of such measured Cherenkov angles represents the basic experimental information provided by the RICH. Any particle identification method, from the most simple average at the track level to the most complicated likelihood using the full event information, can be derived from it.

6.8. Time-of-flight systems

The time-of-flight (TOF) detectors for CLAS12 include the Forward Time-of-Flight system (FTOF) [6] and Central Time-of-Flight system (CTOF) [10]. The FTOF consists of planes of scintillator counters located between the RICH/LTCC and the ECAL. Two parallel counter arrays in each Forward Detector sector are employed to achieve the desired time resolution in the polar angle range from 5° to 35°. The arrays are referred to as panel-1b (closer to the target) and panel-1a (farther from the target). A third set of counter arrays referred to a panel-2 covers polar angles from 35° to 45°. The different FTOF arrays can be seen in Fig. 6. The CTOF consists of a barrel of scintillator counters located just outside of the CVT within the solenoid.

The raw data from the detector PMTs read out during data acquisition include an ADC charge and hit time from a fitted flash ADC (FADC) waveform and a TDC time. The ADC and TDC information is read out and recorded only for those channels that are above the ~1 MeV hardware readout threshold for the FADCs and discriminators of both systems.

As the FTOF and CTOF counters employ double-ended PMT readout, the calibration procedures for these systems (described in detail in Refs. [6,10]) allow the reconstruction to report accurate hit times and deposited energy associated with both PMT signals above threshold. At this point the event reconstruction combines the PMT hit times and energies to give a hit time and energy deposition associated with the scintillation counter. In a second phase, hits in adjacent counters, due to particles that pass through multiple counters in the FTOF and CTOF systems (so-called "corner clippers"), are combined into clusters with an associated time, coordinate, and deposited energy. The algorithms for the hit and cluster definitions are detailed in the next sections.

6.8.1. Raw counter hits

Raw hits for the TOF systems are defined by matching the ADC and TDC information reported for each counter. This matching is based on the comparison of the TDC time with the time from the FADC waveform analysis. The latter is derived from fitting the leading edge of the FADC pulse shape during data decoding. Due to the choice of fast timing PMTs for the detector readout and the use of 250 MHz FADCs, the number of samples on the leading edge of the PMT pulses is only 3 to 4, hence the FADC timing resolution is only ~1 ns. The FADC and TDC times are then required to be within a selected window. The windows parameters, position, and width, as well as all other constants used by the reconstruction package, are loaded at run time from CCDB. Currently the window width used is 10 ns, which was found to be sufficient to reduce the probability of a mismatch of the ADC and TDC data for a given scintillation bar hit. This is especially important for the FTOF as the ADC value of the hit is used to compute the time-walk correction.

6.8.2. Reconstructed counter hits

Raw hits are processed to determine reconstructed hits with energy, time, and position information. The reconstructed hit times from the individual PMTs need to account for the time delays along the readout path that include the PMT signal transit time, the signal propagation times through the signal cables and the electronics, and any time-walk effects associated with the readout discriminators. For the FTOF readout, leading-edge discriminators are employed, while for the CTOF readout, constant fraction discriminators are employed and no external time-walk corrections are required. The hit times reconstructed by the TDC readout of the PMTs at the ends of each scintillation bar (referred to generically here as 1 and 2) are given by:

$$t_{1/2} = (CONV \cdot TDC_{1/2}) - t^{walk}_{1/2} \mp \frac{C_{12}}{2} + C_{RF} + C_{p2p}, \quad (7)$$

where $CONV$ is the TDC channel-to-time conversion factor (0.024 ns/bin), TDC is the measured PMT TDC value relative to the trigger signal, t^{walk} is the time-walk correction that accounts for the pulse amplitude dependence of the crossing times of the discriminator threshold (used only for FTOF), C_{12} is a time offset to center the PMT TDC difference distribution about 0, and C_{RF} and C_{p2p} are the time offsets to align all of the counter hit times with respect to the accelerator RF time and to each other, respectively. The paddle-to-paddle time offsets C_{p2p} mainly account for the signal delays along the cable lengths from the PMT output to the readout electronics.

The FTOF and CTOF particle hit times relative to the trigger signal can be determined separately from the times t_1 and t_2 measured by the PMTs of a given scintillation bar using:

$$t^{1/2}_{hit} = t_{1/2} - \frac{d_{1/2}}{v_{eff}}, \quad (8)$$

where $d_{1/2}$ represents the distances along the bar from the hit point to the PMT given by:

$$d_{1/2} = L/2 \pm y, \quad (9)$$

with y the hit coordinate along the bar (determined from forward tracking for the FTOF and central tracking for the CTOF) and L is the counter length. The average counter hit time is given by:

$$\bar{t}_{hit} = \frac{1}{2}(t^1_{hit} + t^2_{hit}) = \frac{1}{2}\left[t_1 + t_2 - \frac{L}{v_{eff}}\right], \tag{10}$$

where v_{eff} is the effective speed of light in the scintillation bar.

Using the timing information from the PMTs at the ends of each bar, the hit coordinate along the bar with respect to the center of the bar can be defined from the FTOF or CTOF information alone using:

$$y = \frac{v_{eff}}{2}(t_1 - t_2 - C_{12}). \tag{11}$$

It is this coordinate determination that is compared against the projected coordinate from tracking to determine if the time-of-flight hit matches to a projected track.

The algorithm detailed above and currently in use requires good ADC and TDC information for the PMTs at both ends of the counter to be available. However, if one of the PMTs of a counter is malfunctioning, Eq. (8) shows that the hit time recorded from the working PMT alone can be used to reconstruct the particle hit time using tracking information to correct for the light propagation delay along the counter. The loss of one PMT involves a $\sqrt{2}$ worse timing resolution for the counter. Algorithms to address these cases are already implemented in the reconstruction service but are presently disabled.

The reconstructed energies from the ADC values of the PMTs (1 and 2) for a given scintillator bar are given by:

$$E_{1/2} = (ADC_{1/2} - PED_{1/2})\left[\frac{\left(\frac{dE}{dx}\right)_{MIP} \cdot t}{ADC_{MIP}}\right], \tag{12}$$

where $(ADC - PED)$ is the measured pedestal-subtracted ADC integral, ADC_{MIP} is the ADC value for normally incident minimum-ionizing particles (MIPs) at the center of the scintillation bar, $\left(\frac{dE}{dx}\right)_{MIP}$ is the energy loss for MIPs in the scintillation bars (2.001 MeV/cm), and t is the scintillation bar thickness. The deposited energy is computed as the geometric mean of the deposited energy as determined from the two counter PMTs as:

$$E_{dep} = \sqrt{E_1 E_2}. \tag{13}$$

6.8.3. Hit clustering and matching

If there are multiple scintillation bar hits associated with a single incident charged particle track, a hit cluster can be defined. These clusters have associated with them a hit coordinate, deposited energy, and hit time. Hits are assigned as part of a cluster in either the FTOF or CTOF if their hit positions and hit times fall within selected matching windows. The clustering algorithm looks to define hit clusters matched to tracks separately in each of the counter arrays.

With hit clusters defined, the associated cluster coordinate along the counter length is defined as the energy-deposited weighted average of the reconstructed y coordinate from Eq. (11) as:

$$y_{cluster} = \sum_{i=1}^{N} y_i \cdot \Delta E_i. \tag{14}$$

Note that in both the FTOF and CTOF systems, the maximum cluster size is practically limited to $N = 2$. For the coordinate transverse to the counter length along the counter width, the coordinate is defined as the average of the coordinates associated with the middle of the bar.

The assigned cluster energy is the sum of the deposited energies in the counters associated with the defined cluster,

$$E_{cluster} = \sum_{i=1}^{N} E^i_{dep}. \tag{15}$$

In the FTOF when there is a defined hit or a defined cluster in both panel-1b and panel-1a, a second cluster matching algorithm is applied to determine if the hit or cluster in panel-1b and the hit or cluster in panel-1a are associated with the same incident track matched to the panel-1b hit or cluster. If they are associated, a corrected FTOF hit time based on the panel-1a and panel-1b cluster times is computed using a time resolution weighting according to the counter in each cluster with the largest energy deposition using:

$$t_{corr} = \frac{\frac{t^{cluster}_{1b}}{\delta_{1b}} + \frac{(t^{cluster}_{1a} - \Delta r/\beta c)}{\delta_{1a}}}{\left(\frac{1}{\delta_{1b}} + \frac{1}{\delta_{1a}}\right)}. \tag{16}$$

Here $\delta_{1a,1b}$ are the effective time resolutions measured for the counters determined during the FTOF calibration procedure and $t^{cluster}_{1a,1b}$ are the cluster hit times in panel-1a and panel-1b. The term $\Delta r/(\beta c)$ accounts for the path length difference between the panel-1b cluster hit coordinate and the panel-1a cluster hit coordinate and comes from forward tracking information. As β depends on the FTOF time, it is assumed that it is based on the panel-1b time information (the array with the better timing resolution).

Given the effective FTOF counter resolutions, the overall FTOF hit time resolution is improved by 15%–20% when combining the times from panel-1b and panel-1a in this manner. Of course, if the track interacts with only panel-1a or with only panel-1b due to the slightly different solid angles of coverage of the arrays, then only the single plane hit time is used in the event reconstruction.

Note that employing the cluster times has not yet been fully validated in the event reconstruction but is currently under test using Monte Carlo data samples. While this validation is in progress, the information passed from the time-of-flight systems to the Event Builder is based on reconstructed hits.

6.9. Central neutron detector

The Central Neutron Detector (CND) [11] is used to detect 0.2 to 1 GeV neutrons in the Central Detector. The CND consists of a barrel of three layers of scintillators coupled at their downstream ends with U-turn light guides and read out on their upstream ends with PMTs. The light readout from the scintillation bar in which a particle interacts is called “direct”, while the light that travels through the U-turn into the neighboring bar and read out in the coupled counter is called “indirect”.

The reconstruction of the CND is done in five steps:

- the choice of the direct and indirect paddle, by comparing the two PMT times (referred to as the left and right times) of a coupled pair of counters, after correcting them for relative and absolute offsets determined in the calibration procedure and accounting for light propagation times [11];
- the reconstruction of the deposited energy;
- the reconstruction of the time and position of the hit in the paddle;
- the matching of CND hits with CVT tracks coming from the interaction vertex;
- the clustering of multiple hits.

6.9.1. Energy reconstruction

For direct hits in the left paddle at a position z along the paddle, the two associated ADCs can be written as:

$$ADC_L = \frac{E_L}{E_0} \cdot MIP_D \cdot e^{\frac{-z}{\lambda_L}}, \tag{17}$$

$$ADC_R = \frac{E_R}{E_0} \cdot MIP_I \cdot e^{\frac{-(L-z)}{\lambda_L}}.$$

Here MIP_D (MIP_I) is the ADC-to-energy constant for direct (indirect) minimum-ionizing particles (MIPs), $E_{L/R}$ is half the energy deposited by the particle in the left/right paddle, z is the distance along

the left counter to the left PMT, L is the length of each paddle, and A_L is the coupled counter pair attenuation length. E_0 is given by:

$$E_0 = \frac{h \cdot 2.001}{2} \text{ MeV}, \tag{18}$$

where h is the thickness of each scintillator. In the case of direct hits in the right paddle, the applicable equations are obtained by switching the L/R indices. The energy reconstruction for each coupled paddle is obtained inverting Eq. (17). The total energy of the hit is then given by the sum of E_L and E_R.

6.9.2. Hit position and time reconstruction

The reconstruction of the time and position of a hit will be shown for the case of a hit in the left paddle. In case of a hit in the right paddle, the applicable equations are obtained by switching the L/R indices.

Starting from t_L and t_R, defined as

$$\begin{aligned} t_L &= t_{tof} + \frac{z}{v_{eff_L}} + t_S + t_{off} + TDC_j, \\ t_R &= t_{tof} + \frac{L-z}{v_{eff_L}} + \frac{L}{v_{eff_R}} + u_t + t_S + t_{off} + TDC_j, \end{aligned} \tag{19}$$

and subtracting the time offsets obtained from the calibration (t_{off}), the start time (t_S), and the time jitter (TDC_j), one can define the propagation times $t_{L_{prop}}$ and $t_{R_{prop}}$ to the left and right PMTs of the coupled pair as:

$$\begin{aligned} t_{L_{prop}} &= t_{tof} + \frac{z}{v_{eff_L}}, \\ t_{R_{prop}} &= t_{tof} - \frac{z}{v_{eff_L}} + \frac{L}{v_{eff_L}} + \frac{L}{v_{eff_R}} + u_t, \end{aligned} \tag{20}$$

where $v_{eff_{L/R}}$ is the effective light velocity in the left/right paddle and u_t is the propagation time of light to travel in the U-turn. Both of these quantities are obtained from CND calibration (see Ref. [11] for details).

The position of the hit z is obtained from the difference of the left and right propagation times:

$$\begin{aligned} z = {} & \frac{v_{eff_L}}{2}\left(t_{L_{prop}} - t_{R_{prop}}\right) \\ & + \frac{v_{eff_L}}{2}\left(L \cdot \left(\frac{1}{v_{eff_L}} + \frac{1}{v_{eff_R}}\right) + u_t\right). \end{aligned} \tag{21}$$

The x and y coordinates of the hit are obtained from the radius and the azimuthal angle of the hit, which are, in turn, determined by knowing the layer, sector, and component (left or right) of the hit. Finally, the time of flight of the particle that produced the hit is obtained from the sum of the left and right propagation times:

$$\begin{aligned} t_{tof} = {} & \frac{1}{2}\left(t_{L_{prop}} + t_{R_{prop}}\right) \\ & - \frac{1}{2}\left(L \cdot \left(\frac{1}{v_{eff_L}} + \frac{1}{v_{eff_R}}\right) - u_t\right). \end{aligned} \tag{22}$$

6.9.3. Hit/track matching

Tracks from charged particles crossing the CVT are associated with hits in the CND. This allows the position of each CND hit to be computed from the track extrapolated beyond the CVT to the location of the hit counter. This information is used in the detector calibration [11]. CVT tracks are extrapolated to radii corresponding to the entry point, midpoint, and exit point of the track in the paddle. These points are defined as the intersections between the helix of the track and cylinders of radii corresponding to the distances between the beamline and the three CND layers. A CVT track and a CND hit are matched if the hit coordinates and the extrapolated coordinates are within a user-selected distance. The path traveled by the particle in the paddle is approximated as the distance between the entry and exit points. The path length between the vertex and the hit is obtained from the helix parameters.

6.9.4. Clustering

The clustering of CND hits is based on the geometrical space–time distance between them. The determination of the maximal distance for clustering two hits together takes into account the measured resolutions for position and timing of the CND counters [11].

The algorithm uses standard hierarchical clustering [25]. A scan of all hits in an event is performed and only hits with a deposited energy greater than 1 MeV are considered for clustering. The two closest hits are combined into a single hit with associated energy defined as the sum of the energies of both hits. The position and timing of the cluster hit are defined as those of the hit with the highest energy, i.e. the seed hit. The same algorithm is recursively run on the remaining hits. Finally, the leftover hits that are relatively far from each other are called clusters. The sector, layer, and component of each cluster are those of the seed hit.

6.10. Forward tagger

The Forward Tagger (FT) [12] is placed between the HTCC and the torus magnet along the beamline and is designed to detect electrons and photons in the polar angular range from 2° to 5°. The FT is composed of an electromagnetic calorimeter based on $PbWO_4$ crystals (FT-Cal), a two-layer scintillator hodoscope (FT-Hodo), and a Micromegas tracker (FT-Trk) similar in design to the FMT [9]. The FT reconstruction service is designed to provide efficient algorithms to determine the energy, time, and positions of the signals associated with the incident particle. The reconstruction matches this information to determine the type and three-momentum of the particle. The package consists of four services, one for each of the sub-detectors and a global service that builds the particle information from the output of the detector reconstruction. In the following, we describe each of the FT services and their algorithms.

6.10.1. The FT-Cal reconstruction service

The calorimeter service has the role of reconstructing clusters associated with the incident particles from the detector raw information. These include the charge and time recorded by the FADC boards that read out the crystal signals. A cluster is defined as a contiguous ensemble of crystals within the calorimeter, in which a signals above a minimum energy threshold (10 MeV) are found within a selected time window (10 ns) from each other.

The first step to build a cluster is to reconstruct the energy and time of the individual crystal hits from the raw FADC information. For this purpose, the charge and raw time of the recorded pulse are converted to energy and time using calibration constants derived from data. A linear relationship between energy and charge is assumed. The hit time is defined from the raw time by applying an offset and a charge-dependent correction that accounts for time-walk effects.

Reconstructed hits are then ordered by energy and, starting from the maximum energy hit, subsequent crystals are associated with it based on their relative positions and time differences. Once all hits are associated with a cluster, the overall cluster energy, time, and positions are computed. The cluster energy E_{tot} is calculated as the sum of the individual hit energies, E_{rec}, plus a global correction to account for the hit thresholds and for shower leakages due to the finite length of the crystal and the overall calorimeter size. This correction is parameterized as a function of the measured cluster energy based on full Geant4 simulations of the detector response [12]. The cluster time is computed as the energy-weighted average of the individual hit times. Finally, the cluster position in the x–y plane (transverse to the beam z-axis) is computed as the logarithmic energy-weighted hit coordinates (x_i, y_i), i.e. the crystal position with the following functional form [26]:

$$\begin{aligned} x_{cluster} &= \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i}, \\ y_{cluster} &= \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i}, \end{aligned} \tag{23}$$

where the index i runs over the N crystals in the cluster and the weighting factors w_i are defined as:

$$w_i = max\left(0, w_0 + ln(E_i/E_{rec})\right). \quad (24)$$

The parameter w_0 was fixed to 3.45 after optimization based on Geant4 simulations. The z coordinate of the cluster is set to a constant depth from the crystal upstream face that was optimized based on Monte Carlo studies.

The resulting clusters are finally selected by applying cuts to exclude instances where the total and seed energies are less than a defined threshold or where the number of crystals in the cluster is below a defined limit.[2] All of these selection parameters, as well as the other constants used in the cluster reconstruction, are set at run time by reading the CLAS12 calibration constants database, CCDB.

The final list of clusters is saved to an output HIPO bank that is passed to the global FT reconstruction service for the particle reconstruction. The intermediate hit information is also saved to a HIPO bank for debugging purposes.

6.10.2. The FT-hodo reconstruction service

The FT-Hodo is used to discriminate photons and electrons. The system consists of two layers of plastic scintillator tiles read out with silicon photomultipliers. The FT-Hodo reconstruction service, which is similar to that for the FT-Cal, has the role of reconstructing hits and associating matching hits in the two layers of the detector to form clusters.

Hits are defined from the raw FADC information as the energy and time of the signals associated with the incident particles. These are computed assuming a linear relation for the charge-to-energy conversion and an additive offset between the raw and reconstructed time. The constants necessary for these conversions are derived for each individual detector component based on beam-data calibrations as discussed in Ref. [12] and set at run time by reading the values from CCDB. The reconstructed hits are then selected by applying a minimum energy threshold that was optimized based on data analysis.

The selected hits are then matched to form clusters consisting of scintillator tiles in the two detector layers, matched in space and time. The position matching distance is defined by the largest tile size, i.e. 3 cm, while the time matching parameter was optimized based on Geant4 simulations and is set conservatively to 8 ns. The resulting cluster parameters are the cluster size, position, total energy, and time. The cluster energy is calculated as the sum of the individual hit energies, while both the position in the x–y plane and time are calculated as the energy-weighted average of the corresponding hit parameters. The resulting information is saved to a HIPO bank that is passed to the global FT service. As for the calorimeter, the intermediate hit information is also saved to a HIPO bank for debugging purposes.

6.10.3. The FT-Trk reconstruction service

The FT-Trk is used to measure the angle of the scattered electron. It consists of two double-layers of Micromegas and is positioned upstream of the FT-Hodo. The FT-Trk reconstruction service is currently in the development stage and will be described in detail in a future publication, while here we discuss only the general principles. Algorithms for the conversion of the raw Micromegas detector information to hits and for matching hits to form clusters follows those developed for the CLAS12 BMT that are discussed in Section 6.3. All combinations of clusters identified in the x–y layers of each of the two sub-detectors are then built to form crosses. Finally, the crosses found in the two sub-detectors are matched based on their position and saved as input for the global FT service.

[2] Note that the seed crystal is the one with the largest signal.

6.11. The FT global service

The final step of the FT reconstruction is the matching of the information resulting from the three sub-detectors. Specifically, hodoscope and calorimeter clusters are matched to distinguish charged particles having a cluster in the hodoscope from neutrals that have a low probability of creating a signal in that detector. The matching is based on the relative position of the calorimeter and hodoscope clusters in the x–y plane and on their time difference. The position matching parameter is determined by the hodoscope component size, while the timing cut is set to 10 ns, similar to the cut value used in the lower levels of the FT reconstruction. The output of the matching is an FT *particle*, whose energy and position at the detector are determined from the calorimeter cluster parameters, while its charge is set by the presence of a hodoscope cluster. The particle three-momentum at the target for charged particles is then computed accounting for the bending in the solenoid field, while for neutrals it is computed assuming a straight path from the CLAS12 target center to the FT. When available, the tracker information will be used to refine the determination of the particle impact point on the FT front face and, therefore, to improve the reconstruction of the angles at the vertex. The resulting particle information is saved to a HIPO bank for use in the CLAS12 Event Builder service.

6.12. Event builder

The Event Builder is the last service in the reconstruction algorithm, and performs a series of functions:

- collects information from the upstream services;
- correlates information from the sub-detectors into particles;
- performs a general particle identification scheme;
- organizes the resulting information into a standardized, persistent data bank structure.

The service is run twice with identical algorithms, once using hit-based tracks, and later with time-based tracks, where the results of the hit-based Event Builder are used to initialize time-based tracking.

6.12.1. Forming particles

In defining a reconstructed charged particle in CLAS12, the Event Builder assumes that an assignment will be made for each reconstructed track in both the Forward Detector and the Central Detector. The associated calorimeter, scintillator, and Cherenkov detector responses are then assigned to that particle based on geometric coincidences between the detector responses and the track, with matching criteria corresponding to the resolution of a given detector. The geometric matching is based on the distance of closest approach between the track and the response, where an example is shown in Fig. 25.

A similar procedure is followed for creating neutral particles, except the seeding is presently with unassociated ECAL (for the Forward Detector) and CND (for the Central Detector) responses instead of tracks.

6.12.2. Event start time

A start time is assigned to the entire event and serves as our most precise reference time on which all time-based particle identification relies. This is based on the optimal charged particle candidate in the Forward Detector with an associated FTOF timing response. The Event Builder assigns the start time based on the highest energy electron in the ECAL. If there is no electron in the ECAL, it next looks for a positron in the ECAL. If there is no lepton, the next track in the priority list is a forward-going positive track (assumed to be a π^+). Finally, if there is no forward-going positive track, it looks for a forward-going negative track (assumed to be a π^-). When looking for π^+ or π^- tracks, only the candidate with the highest momentum in each group is considered.

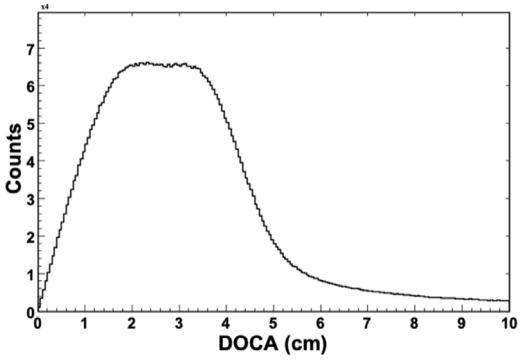

Fig. 25. Example of the geometric matching criteria showing the distance of closest approach between a charged track from the DC extrapolated to the ECAL and the cluster positions in the ECAL.

A parallel event start time is determined from the FT to facilitate physics analyses and triggers where the primary scattered electron is at very forward angles in the FT. In this case, all combinations of charged particles in the FT and the Forward Detector are considered. The particle in the FT is assumed to be an electron, whereas all hadron mass hypotheses are considered for the Forward Detector tracks. The combination with the best time coincidence is chosen. The timing of the resulting FT electron is then used to assign the start time.

A correction to the start time is then performed using the RF signal from the accelerator, combined with the reconstructed event vertex position. This effectively aligns the event start time to our best measure of the beam-bunch arrival time at the target.

The uncorrected, measured vertex time of a particle, t_v, can be written as

$$t_v = t - \frac{P_L}{\beta c}, \tag{25}$$

where t is the measured time response (e.g. in a scintillator), P_L is the path length between the primary interaction vertex and that response, and βc is the speed of the particle. We can then construct a correction to align this time with the closest beam bunch time at the target:

$$\begin{aligned} \Delta t_{RF} &= t_v + (z_0 - z_v)/c - t_{RF} - N/(2f_{RF}), \\ \Delta t'_{RF} &= mod(\Delta t_{RF}, 1/f^{RF}) - 1/(2f_{RF}), \end{aligned} \tag{26}$$

where f_{RF} is the frequency of the accelerator, 249.5 MHz or 499 MHz, corresponding to 2.004 ns or 4.008 ns bunch spacings, t_{RF} is the measured, calibrated RF time for the event, and z_0 is the target center and enters due to its use as a position calibration reference. The resulting RF- and vertex-corrected start time for the event is then given as

$$t'_v = t_v - \Delta t'_{RF}. \tag{27}$$

6.12.3. Particle identification

The next stage is a basic particle identification scheme. This is intended to be loose to accommodate a variety of physics analyses, while persisting the necessary information to easily tighten and improve the criteria later.

For charged particles, first calorimetry and Cherenkov information is used to positively identify e^-/e^+ candidates in the Forward Detector. If the measured energy deposition is consistent with the expected sampling fraction of the ECAL, and the photoelectron response from the HTCC is consistent with $\beta \sim 1$, the particle is assigned as an e^- or e^+ depending on sign of the curvature of the track from forward tracking with the DCs through the torus magnetic field.

The remaining charged particles are then assumed to be hadrons and assigned an identity based solely on timing information, where the $p/K/\pi$ candidate giving the smallest time residual is assigned. This time residual is computed from the difference between the measured particle flight time and that computed for a given mass hypothesis. Fig. 26 shows reconstructed β vs. momentum distributions from beam data for forward-going positively charged hadrons using information from the FTOF and DC subsystems, where the electron is reconstructed either in the Forward Detector (Fig. 26(top)) or in the Forward Tagger (Fig. 26(bottom)). The computed curves for the different mass hypotheses are overlaid.

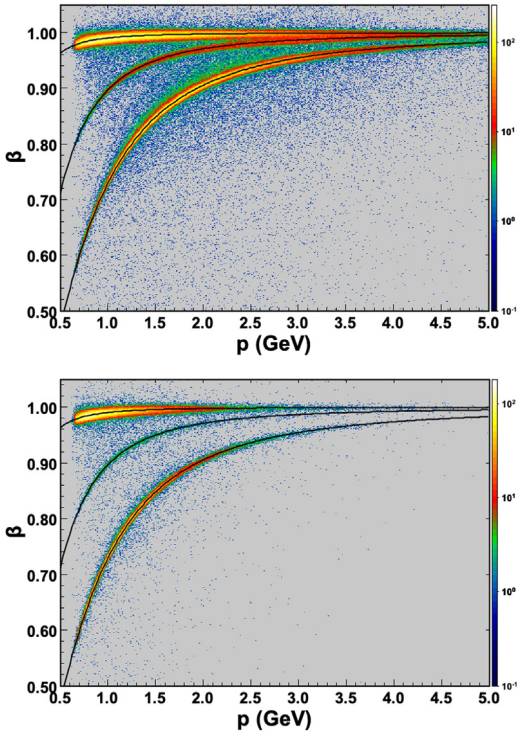

Fig. 26. Particle β vs. momentum from simulation data for positively charged tracks with their start time from an electron in the Forward Detector (top plot) or in the FT (bottom plot).

Identification of neutral particles assumes only neutrons and photons, differentiated only by timing and topological information. For the Forward Detectors this is based on the ECAL, while for the Central Detector it is based on the CND, and their reconstructed cluster positions are used to compute the particle travel path from the event vertex, assuming a straight-line trajectory. If the resulting measured β is close to 1, the particle is assigned as a photon, otherwise it is assigned as a neutron. For photons in the Forward Detector, the momentum is determined from its deposited energy and ECAL sampling fraction [7]. For neutrons, the momentum is assigned based on the measured β, assuming the neutron mass. Fig. 27 shows an example of β reconstructed for neutrals in the Forward Detector showing separation of photons and neutrons.

A particle identification quality factor in the form of a signed-χ, or pull, is assigned based on the individual contributing detector subsystem responses and their resolutions. For e^-/e^+ identification the resolution-normalized distance from the expected ECAL sampling fraction is used, while for charged hadrons the resolution normalized time-difference is used. The resulting information is organized into standardized output bank structures for physics analysis, see Section 7.2. This includes the particle four-vectors, the associated detector responses, and global event information such as beam RF and helicity information.

6.12.4. Particle identification performance

The accuracy of the particle identification algorithm that is currently implemented can be estimated from Monte Carlo simulations

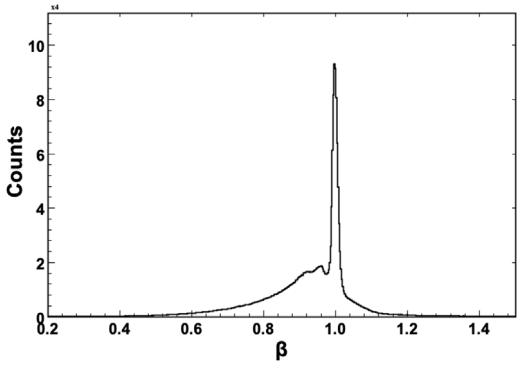

Fig. 27. β distribution for neutral particles as measured by the ECAL from simulation data, showing a sharp peak at $\beta = 1$ from photons and a broader, slower distribution from neutrons.

Table 1
Particle identification matrix for the CLAS12 Forward Detector based on simulated hadrons and photons with momentum between 1 and 2.5 GeV, and electrons up to 9 GeV. The diagonal elements are correctly identified, while the off-diagonal elements are misidentified. Detector inefficiencies are included.

	Truth					
	e	π	K	p	n	γ
e	0.98					
π		0.93	0.10	0.00		
K		0.03	0.80	0.00		
p		0.03	0.02	0.98		
n					0.66	0.01
γ					0.14	0.95

Table 2
Particle identification matrix for the CLAS12 Central Detector based on simulated hadrons with momentum between 0.3 and 1.1 GeV. The diagonal elements are correctly identified, while the off-diagonal elements are misidentified. Detector inefficiencies are included.

	Truth			
	π	K	p	n
π	0.84	0.14	0.00	
K	0.11	0.80	0.01	
p	0.03	0.04	0.95	
n				0.11
γ				0.00

where the assigned particle identification can be compared to the true one. Tables 1 and 2 show the particle identification matrix for the Forward and Central Detectors, respectively. The values are based on simulations of electron–hadron or electron–photon pairs with hadron and photon momenta in the range from 1 to 2.5 GeV and electron momenta in the range from 1 to 9 GeV. The diagonal elements correspond to the cases where the particle is correctly identified and the off-diagonal elements to the cases where the particle is misidentified. Future improvements are anticipated and discussed in Section 9.3.

Another measure of the particle identification performance for neutrals is given by the reconstruction of π^0 decays to two photons. Fig. 28 shows the $\gamma\gamma$ invariant mass reconstructed from the ECAL and from the Forward Tagger.

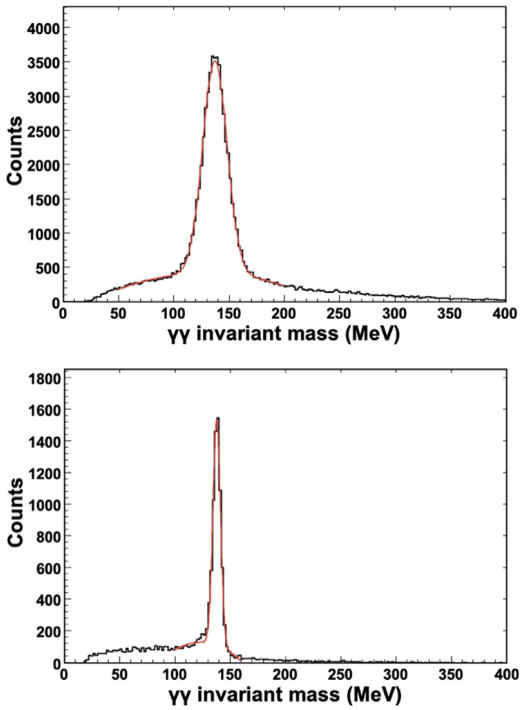

Fig. 28. Reconstructed $\pi^0 \to \gamma\gamma$ candidates using photons detected in the ECAL (top plot) and the FT (bottom plot). The plots are based on simulations of semi-inclusive deep inelastic scattering events generated based on the PYTHIA event generator [27].

7. Data processing

7.1. Workflow

The raw data from the detector subsystems is currently first pre-processed in what is called the decoding stage. This is an I/O-heavy, single-threaded process and involves extracting hits from waveforms, translating data-acquisition/hardware nomenclature (associated with crate/slot/channel labels) into physical detector objects, performing special analyses dependent on serial event access, and converting from the input EVIO format to the HIPO data format. This phase includes registering beam helicity state changes and special scaler events, and populating their results into special tagged HIPO events to facilitate later analysis. The result is a factor of ∼5 reduction in size and a file format optimized for I/O.

The second data processing stage is a CPU-heavy reconstruction phase, including all of the tracking, clustering, calorimetry, time-of-flight, and event building described in the previous sections. It runs multi-threaded in the CLARA framework and can be configured to output various data schema depending on the purpose, see Section 7.2, during full-scale data processing, or larger, special-purpose banks during preliminary calibration phases.

The final stage of data processing involves the running of I/O-heavy analysis trains that perform event skimming (e.g. filtering out specific final state event topologies), and accommodate various corrections and common analysis plugins. It splits the data into multiple output files based on different event selections, each optimized for a group of physics analyses. An example schematic is shown in Fig. 29. This stage is designed to be run repeatedly as selection criteria and physics analyses mature. The reduction factor of the input file size generated by the analysis trains depends directly on the applied filtering conditions for the specific output. Selecting events with an electron identified in the ECAL provides a reduction factor of ∼0.3, while for events with an electron in the ECAL and a positive hadron in CLAS12, the reduction factor is ∼0.1. A typical 2 GB EVIO file gets reduced to a 200 MB HIPO data file with banks for physics analysis (see Section 7.2).

7.2. Data summary tapes

The final data output is provided by the Event Builder in the form of data summary tapes (DSTs), a standardized selection of HIPO banks for physics analysis. The trains mentioned above are run on input DST files to produce skimmed output DSTs. These include:

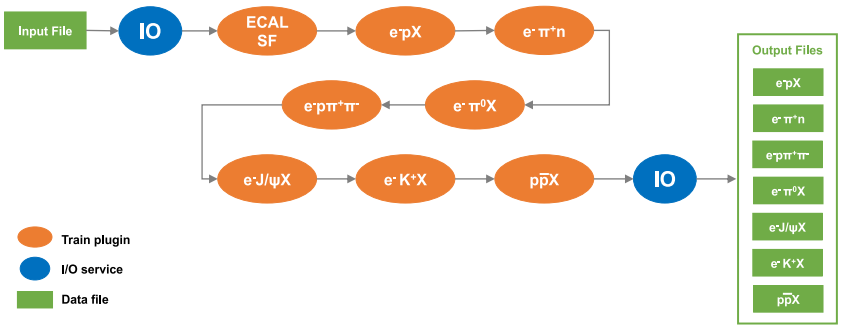

Fig. 29. Schematic flow of analysis trains. The example shows a train composed of a plugin to correct the ECAL sampling fraction (SF) and several analysis filters for different final states. Events from a HIPO file are read by the IO service and processed through the analysis chain that applies the selected corrections and labels them according to the different filters. The labeled events are written to the corresponding output file.

- global event information, e.g. run number and event time stamp, integrated beam charge, beam helicity state, event start time;
- particle information, e.g. momentum four-vector and vertex position, particle type and identification quality, and status words that encode information on the sub-detectors involved in the particle formation;
- high-level detector response information associated with each particle, e.g. detector identifier, response position and time, and track trajectory in each detector layer.

The DST banks are organized such that the large detector information banks can easily be dropped to leave only the data essential for a high-level physics analysis, without leaving unassociated references or unnecessary information.[3]

7.3. Computing resources

Reconstruction of all CLAS12 data is performed on Jefferson Lab's batch computing system [28]. It currently consists of about 400 computing nodes of various types, with a total of about 21,000 available jobs slots and half as many cores. The input raw data and analyzed output data are stored on JLab's tape silo [29], which provides sufficient cold storage for all of JLab's activities. Data for physics analysis are also stored live on JLab's Luster filesystems [30], which currently amounts about 2 PB of disk, but will be increased to almost 7 PB in the near future. Analysis of the reconstructed data is performed on the JLab batch and interactive farm nodes, and also exported to other institutions for final physics analysis. CLAS12 currently has 450 TB available on different file systems, and a fair share computing resource of 36M core-hr/yr.

8. Code management

8.1. Repositories

The software is managed in a github repository [31], and branches and forks are utilized to accommodate parallel development by several groups. Two main branches, *master* and *development*, are utilized to store code ready for production and for validation, respectively. For the main branches, all modifications are made through pull requests after passing the automated tests described in Section 8.3 and require approval by a designated CLAS12 software expert.

[3] Currently all DST banks are saved to a file.

8.2. Releases

There are three reconstruction code release types: test, validation, and production. A tagging scheme has been implemented to indicate the type of change with respect to previous releases. Test releases, identified by the letter "c", are tagged from branches other than the master or development branches and are intended to validate a specific code change or algorithmic improvement. Usage of these releases is typically limited to the developers. Validation releases, identified by the letter "b", are tagged from the development branch to test code updates before merging to the master branch. Production releases are tagged from the master branch after code updates for production data processing.

The release designator scheme uses the format $X(b/c).Y.Z$, where increments of X, Y, or Z are applied in the following cases:

- X: introduction of new technology, major algorithmic improvements, or changes that are not backward compatible;
- Y: extension of interfaces, new implementations, or major bug fixes;
- Z: minor bug fixes.

8.3. Code tests and validation

In addition to automatic builds, the software includes both basic unit tests and advanced tests for several packages. These are designed to verify the correctness and reproducibility of the reconstruction output for a specific package or for the overall event, respectively. Unit tests involve, for example, reconstructing a simulated track or particle hit in a specific detector and comparing the result to the truth information. Advanced and extended tests are run on either a Monte Carlo or beam data sample, comparing to the Monte Carlo truth information in the first case or to the results obtained in previous releases in the second case. A portion of the tests are run automatically at build time, using the TravisCI system linked to the github repository. These automatic tests take about 30 min to run and have proven invaluable in overseeing software development.

In addition to unit and advanced tests, every new release is subject to extensive validation on both Monte Carlo and beam data. Samples of Monte Carlo and beam events for different beam energies and detector configurations were chosen to test event reconstruction over the entire detector acceptance. Reconstruction of these samples is performed and results are compared to previous code releases. The comparison focuses on several parameters, from processing time, to momentum resolution, to particle reconstruction efficiency. A new release is accepted for production only if it results in globally improved event reconstruction performance.

9. Ongoing developments

The software framework and event reconstruction described in the previous sections are based on the code that is currently being utilized for data processing or will be deployed in the near future in an upcoming release. Nevertheless, as CLAS12 data are being analyzed, several potential improvements have been identified and are either in the process of being implemented or planned for the near future. In this section, we discussed the most relevant developments.

9.1. Artificial intelligence assisted forward tracking

Recent progress in the field of machine learning offers a promising alternative to conventional algorithmic tracking methods. While the conventional methods provide algorithms that are well understood and well studied, there are some algorithms in the data reconstruction process that can be substituted with neural networks to reduce data processing times. For CLAS12, tracking is the most time-consuming aspect of experimental data processing. Tracking in the DC takes up to ∼90% of the total data processing time, which includes finding track candidates and iterating through track-forming segments to find the best combinations of segments that can form a track. This time increases with luminosity as the number of noise segments increases and can ultimately lead to processing time degradation. We have started to address this issue by employing machine learning techniques to find the best track candidates in each event and to reduce the number of combinatorics.

With increased luminosity, the number of potential DC cross candidates increases. This implies that the Kalman Filter fitting algorithm must be run for all possible combinations of crosses.

Reconstructed track segments from both positive and negative tracks from the currently reconstructed data samples are used to train the neural network. We are currently testing three types of neural networks: boosted decision tree [32], multilayer perceptron [33], convolutional [34].

Preliminary results indicate that the convolutional neural network performed competitively with the multilayer perceptron with about 97% accuracy and 3% false positives.

The hits identified as on-track by the neural network are saved in a bank and the DC reconstruction package was adapted to read these data as an input to hit-based tracking. Benchmark results of reconstruction speed for hit-based tracking show a factor of ∼5 improvement.

Implementing the neural network software into the CLAS12 reconstruction workflow is under development. The second stage of the machine learning project will concentrate on efficiency improvements using artificial intelligence assisted tracking.

9.2. Improvements to event reconstruction

The CLAS12 detector began beam operations for physics in early 2018 after a several month commissioning phase. Since that time the event reconstruction code has continued to improve to meet issues as they have arisen. However, the code and the framework are already performing well enough for advanced physics analysis of the collected data to proceed. A broad survey of reconstruction results using the current CLAS12 software framework and event reconstruction code are presented based on beam data in Ref. [1]. As might be expected, there are still areas where development, testing, and validations are in progress in order to continue improvements. In this section, several areas of ongoing work are highlighted.

9.2.1. Improvements to central tracking

Improvements to tracking in the CVT are currently being studied. These include:

- improvements to the tracker geometry implementation and fitting algorithm — the combination of these code modifications is expected to improve the fit residuals, which are indicative of a bias in the current version of the code as seen through systematic shifts in their distributions;
- implementation of geometrical distortions derived from detector alignment;
- the use of the beam offset information (relative to the nominal beam z-axis) in the track fit initialization;
- the use of SVT clusters instead of crosses in the seeding.

These updates aim at enhancing the robustness of the tracking algorithm and improving resolution and efficiency.

9.2.2. Improvements to time-of-flight reconstruction

As discussed in Section 6.8, the output of the time-of-flight reconstruction are hits that are used as input to the Event Builder algorithms. The use of clusters for particles that go through two adjacent TOF paddles (either in the FTOF or CTOF systems) is expected to yield improved timing resolution, as is combining the hit times in FTOF for tracks that go through both forward counter hodoscopes as discussed in Section 6.8.3. A quantitative estimate of the timing resolution improvements and a validation of the clustering algorithm are currently ongoing using on Monte Carlo simulations.

9.3. Improvements to the event builder

The matching of tracks to detector responses is currently based on the distance of closest approach between the tracks and the response coordinates. Improvements to this matching may be obtained using track trajectories, i.e. intersections of the track with the relevant detector planes where the responses are reported, potentially reducing the uncertainty on the path length determination that relies on the response coordinates. Additionally, the use of timing information in matching will reduce the effect of accidentals in high rate detectors such as the HTCC.

In the future, the particle identification scheme will be improved by exploiting additional detector information. This includes the ECAL shower profile to improve electron–pion separation for momenta above ∼4.9 GeV where the HTCC becomes sensitive to charged pions, and RICH responses to improve charged-particle identification in the forward direction.

10. Conclusions

We have presented the software framework and event reconstruction algorithms that are currently being utilized for the processing of data collected by the CLAS12 experiment in Hall B at Jefferson Lab. The framework was developed to allow processing of CLAS12 data for reconstruction and analysis based on a service-oriented architecture. The specific software applications leverage an extensive set of common libraries for handling I/O, geometry, databases, and magnetic field that are designed to support data monitoring, calibration, reconstruction, and analysis.

Full event reconstruction is implemented in the framework as a chain of micro-services that perform reconstruction of the individual CLAS12 subsystems and whose output information is collected by the Event Builder service to form and identify particles. While the current reconstruction chain already supports reconstruction of all subsystems and the creation of full events with performance consistent with expectations, upgrades to the existing software implementation and algorithms are under study. However, the current status of event reconstruction based on data collected during the first production data

runs with CLAS12 with the electron beam are reported and discussed in detail in Ref. [1] that show the efficacy of the developed reconstruction framework, common tools and detector calibration applications, and the associated algorithms required for event reconstruction.

Acknowledgments

We would like to thank the JLab IT Division and CODA group for their support. This work was supported in part by the Chilean Comisión Nacional de Investigación Científica y Tecnológica (CONICYT), the Italian Istituto Nazionale di Fisica Nucleare, the French Centre National de la Recherche Scientifique, the French Commissariat à l'Energie Atomique, the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under contract DE-AC05-06OR23177, the National Science Foundation, the Scottish Universities Physics Alliance (SUPA), United Kingdom, the United Kingdom's Science and Technology Facilities Council, and the National Research Foundation of Korea.

References

[1] V.D. Burkert, et al., The CLAS12 spectrometer at Jefferson laboratory, Nucl. Instrum. Methods A 959 (2020) 163419.
[2] M.D. Mestayer, et al., The CLAS12 drift chamber system, Nucl. Instrum. Methods A 959 (2020) 163518.
[3] M. Ungaro, et al., The CLAS12 low threshold Cherenkov counter, Nucl. Instrum. Methods A (2020) (in this issue).
[4] Y.G. Sharabian, et al., The CLAS12 high threshold Cherenkov counter, Nucl. Instrum. Methods A (2020) (in this issue).
[5] M. Contalbrigo, et al., The CLAS12 RICH detector, Nucl. Instrum. Methods A (2020) (in this issue).
[6] D.S. Carman, et al., The CLAS12 forward time-of-flight system, Nucl. Instrum. Methods A (2020) (in this issue).
[7] G. Asryan, et al., The CLAS12 forward electromagnetic calorimeter, Nucl. Instrum. Methods A 959 (2020) 163425.
[8] M.A. Antonioli, et al., The CLAS12 silicon vertex tracker, Nucl. Instrum. Methods A (2020) (in this issue).
[9] A. Acker, et al., The CLAS12 micromegas vertex tracker, Nucl. Instrum. Methods A (2020) (in this issue).
[10] D.S. Carman, et al., The CLAS12 central time-of-flight system, Nucl. Instrum. Methods A (2020) (in this issue).
[11] P. Chatagnon, et al., The CLAS12 central neutron detector, Nucl. Instrum. Methods A 959 (2020) 163441.
[12] A. Acker, et al., The CLAS12 forward tagger, Nucl. Instrum. Methods A 959 (2020) 163475.
[13] V. Gyurjyan, et al., CLARA: A contemporary approach to physics data processing, J. Phys. Conf. Ser. 331 (2011) 032013.
[14] J. Carbonneau, M. Moog, J. Gilfoyle, et al., APS Division of Nuclear Physics Meeting Abstracts, EA.024, 2011.
[15] Component Based Dataflow Processing Framework, IEEE, ISBN: 978 1-4799-9926-2, 2015, http://dx.doi.org/10.1109/BigData.2015.7363971.
[16] CLARA: the CLAS12 reconstruction and analysis framework, J. Phys. Conf. Ser. 762 (2016) 012009.
[17] The GlueX Collaboration, The the GlueX experiment in Hall D, presentation to JLAB PAC36, 2010, http://www.gluex.org/docs/pac36_update.pdf.
[18] R. Fair, et al., The CLAS12 superconducting magnets, Nucl. Instrum. Methods A (2020) (in this issue).
[19] Event IO Data Format, https://coda.jlab.org/drupal/content/event-io-evio.
[20] S. Boyarinov, et al., The CLAS12 data acquisition system, Nucl. Instrum. Methods A (2020) (in this issue).
[21] A. Spiridonov, Optimized integration of the equations of motion of a particle in the HERA-B magnet, 2005, arXiv:physics/0511177.
[22] V.L. Highland, Nucl. Instrum. Methods 129 (1975) 497; Nucl. Instr. and Meth. 161 (1979) 171.
G.R. Lynch, O.I. Dahl, Nucl. Instr. and Meth. B58 (1991) 6.
[23] I. Abt, I. Kisel, S. Masciocchi, D. Emelyanov, CATS: a cellular automaton for tracking in silicon for the HERA-B vertex detector, Nucl. Instrum. Methods A 489 (2002) 389.
[24] Rainer Mankel, Rep. Progr. Phys. 67 (2004) 553.
[25] W.H.E. Day, H. Edelsbrunner, J. Classification 1 (1984) 7.
[26] R. Niyazov, S. Stepanyan, CLAS/DVCS Inner calorimeter calibration, in: CLAS-Note 2005-021, 2005, https://misportal.jlab.org/ul/Physics/Hall-B/clas/viewFile.cfm/2005-021.pdf?documentId=213.
[27] T. Sjostrand, S. Mrenna, P.Z. Skands, A brief introduction to PYTHIA 8.1, Comput. Phys. Comm. 178 (2008) 852.
[28] Jefferson Lab Batch Farm, https://scicomp.jlab.org/docs/ExpPhyComp.
[29] Jefferson Lab Tape Silo, https://scicomp.jlab.org/docs/node/9.
[30] Jefferson Lab Lustre Filesystem, https://scicomp.jlab.org/docs/node/17.
[31] CLAS12 Reconstruction Software Repository, https://github.com/jeffersonlab/clas12-offline-software.
[32] B.P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, G. McGregor, Nucl. Instrum. Methods A 543 (2005) 577.
[33] K. Hornik, M. Stinchcombe, H. White, Neural Netw. 2 (1989) 359.
[34] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, p. 326.