Automatic Beam Path Analysis of Laser Wakefield Particle Acceleration Data

Oliver Rübel[1,2,3], Cameron G.R. Geddes[4], Estelle Cormier-Michel[4], Kesheng Wu[1], Prabhat[1], Gunther H. Weber[1], Daniela M. Ushizima[1], Peter Messmer[5], Hans Hagen[2], Bernd Hamann[1,2,3], and Wes Bethel[1,3]

[1] Computational Research Division, Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720, USA.
[2] International Research Training Group "Visualization of Large and Unstructured Data Sets – Applications in Geospatial Planning, Modeling, and Engineering," Technische Universität Kaiserslautern, Erwin-Schrödinger-Straße, D-67653 Kaiserslautern, Germany.
[3] Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, University of California, Davis, One Shields Avenue, Davis. CA 95616, USA.
[4] LOASIS program of Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720, USA.
[5] Tech-X Corporation, 5621 Arapahoe Ave. Suite A, Boulder, CO 80303.

Abstract.

Numerical simulations of laser wakefield particle accelerators play a key role in the understanding of the complex acceleration process and in the design of expensive experimental facilities. As the size and complexity of simulation output grows, an increasingly acute challenge is the practical need for computational techniques that aid in scientific knowledge discovery. To that end, we present a set of data-understanding algorithms that work in concert in a pipeline fashion to automatically locate and analyze high energy particle bunches undergoing acceleration in very large simulation datasets. These techniques work cooperatively by first identifying features of interest in individual timesteps, then integrating features across timesteps, and based on the information derived perform analysis of temporally dynamic features. This combination of techniques supports accurate detection of particle beams enabling a deeper level of scientific understanding of physical phenomena than has been possible before. By combining efficient data analysis algorithms and state-of-the-art data management we enable high-performance analysis of extremely large particle datasets in 3D. We demonstrate the usefulness of our methods for a variety of 2D and 3D datasets and discuss the performance of our analysis pipeline.

1. Introduction

Laser wakefield particle accelerators (LWFAs) [1] can accelerate particles to high energy levels over very short distances. LWFAs utilize an electron plasma wave to accelerate charged particles (e.g., electrons) to high energy levels and can create and sustain electric and magnetic fields several thousand times stronger than possible using conventional technologies. Researchers at the LOASIS [2] program have demonstrated high-quality electron beams at 0.1 to 1 GeV using mm long plasmas [3, 4].

Analysis, understanding, and control of the complex physical processes of plasma-based particle acceleration is a challenging task and requires one to understand how particles become trapped in the plasma wave and how particle beams are formed and accelerated. These processes are best understood by tracing the particles that form a beam over time and investigating their temporal evolution [5, 6, 7, 8]. In real-world experiments it is, however, impossible to record the complete evolution of an experiment and much less to trace single particles within a plasma. Simulation of LWFA experiments is, hence, essential for the understanding of the fundamental physics of plasma-based acceleration, understanding of the processes observed in experiments, as well as improvement of experiments.

The datasets produced by LWFA simulations are (i) extremely large, (ii) of varying spatial and temporal resolution, (iii) heterogeneous, and (iv) high-dimensional, making analysis and knowledge discovery from complex LWFA simulation data a challenging task. One main feature researchers are interested in are beams of high-energy particles formed during the course of LWFA simulations. The particle beams of interest define very small subsets of the data making data analysis even more difficult.

Traditionally, detecting particle beams is a process performed manually. A researcher investigates plots of the complete time series, identifies a reference timestep at which a beam of interest exists, and defines proper thresholds to extract the particles of interest from the data. Manual selection of particle bunches is a time-consuming and complex process. While particle beams define a temporally evolving feature of the data, manual selection is commonly performed based on the information of a single reference timestep only.

To enable efficient and accurate analysis of particle beams in LWFA simulation data, dedicated mechanisms for detecting and selecting particle beams are needed. These methods must be efficient and deal with data of varying temporal and spatial resolution. Furthermore, analysis of selected particle beams and their temporal evolution requires effective analysis and visualization methods.

We present a novel approach for automatic detection of particle beams in LWFA simulation data and classification of their temporal behavior. By combining efficient data analysis methods and advanced data management using FastBit [9] we enable efficient analysis and accurate classification of particle beams based on the complete temporal history of the particles that form them. Our analysis pipeline is characterized by a step-by-step analysis process in which we derive in each step additional information about the particle bunches of interest while reducing the amount of data we need to consider in the subsequent analysis. We initially analyze each timestep independently to derive information about particle bunches for each timestep. We merge the information from this ensemble of particle bunch classifications to define a single consolidated description of the different bunches. We finally trace the detected particles over the complete time series and compute the distance of individual particles to a bunch and derive additional information about the different temporal phases of a bunch, e.g., the time when a bunch formed or accelerated. We use state-of-the-art, high-performance visualization based on VisIt [10] to investigate analysis results.

The specific contributions of this work are as follows:

- We present a novel analysis pipeline for automatic detection of particle bunches in LWFA simulation data. As part of this pipeline we describe: i) an efficient method for detecting particle bunches at single timesteps of a simulation; ii) a method for combining particle bunch classifications from different timesteps to define a single consolidated description of a particle bunch; and iii) a novel method for classification of particle bunches based on their temporal paths.
- We discuss the exploration of data analysis results using the visualization system VisIt.
- We show how our methods can be used to study the quality of particle bunches and report first results for applications of our methods for comparative analysis of particle bunches.
- We apply the proposed methods to a variety of 2D and 3D particle datasets, demonstrating the validity and effectiveness of our approach.
- We examine and report the runtime performance of our analysis pipeline for a variety of datasets. We also study the computational performance of new functions for computing 3D conditional histograms we specifically developed for this work and integrated into FastBit.

We first describe related work in Section 2. Section 3 provides an introduction to LWFAs, their simulation, and the data produced. We describe our analysis pipeline in detail in Section 4. In Section 5, we demonstrate applications of the proposed methods to address relevant scientific questions and describe the results we have obtained for various datasets. Section 6 discusses the performance of our analysis pipeline. In Section 7, we provide conclusions and discuss ideas for possible future research directions.

2. Related Work

In this work we make use of a wide range of methods. Before describing our method in detail in Section 4, we first discuss background material and related work. Section 2.1 introduces various data analysis methods which we use for detecting particle bunches of interest. To enable the analysis of even extremely large datasets we employ state-of-the-art data management and data mining methods (see Section 2.2). To investigate analysis results we utilize various visualization concepts described in Section 2.3.

2.1. Data Analysis

2.1.1. Data Analysis in High-Energy Physics

In high-energy physics experiments, it is usually not possible to measure the complete evolution of an experiment but only the end result. Thus, numeric simulations are commonly used to model these experiments computationally to gain insight into the complex physical processes. Knowledge discovery from large, complex simulation data is a challenging task. Visualization and statistical analysis are common tools to address this problem. A large number of analysis frameworks are available for this purpose, e.g., ROOT [11], AIDA [12], R [13] , IDL [14] OpenDX [15], VorpalView [16], ParaView [17], and VisIt [10]. None of these tools, however, addresses the problem of automatic analysis of beam paths.

Fonseca et al. [18] described recently a framework for particle tracing in the context of LWFA simulations based on the OSIRIS [19] framework. The sheer size of the data prohibits saving the complete information of an entire simulation. To get high-quality particle traces they, therefore, execute the simulation twice. After the first simulation run a researcher manually defines the particle subset of interest and then reexecutes the simulation to gather the data of the selected particle subset at a higher temporal resolution. Selection of the particles

of interest is performed manually by selecting, e.g., the n most energetic particles at the last timestep. Martins et al. applied these methods to investigate the ion dynamics and acceleration in relativistic shocks [20]. In contrast to Fonseca et al. and Martins et al. we focus in this work on automatic classification of particle bunches based on particle paths.

Several recent works explore automating different stages of the data analysis process. Bagherjeiran et al. [21] applied graph-based techniques for orbit classification in plasma simulations. Love et al. [22] conducted an image space analysis of coherent structures in plasma simulations using a number of segmentation and region-growing techniques to isolate regions of interest in orbit plots. Both approaches target the system dynamics in particle accelerator data in terms of particle orbits but do not address particle dynamics as a function of time nor inspect particle bunches. Hlína et al. [23] studied dynamic patterns and their velocities in thermal plasma jets via substraction and correlation analysis of succeeding images in a time series of CCD images recording the plasma-radiation. Hlína et al. and Love et al. focus on structures of the plasma itself rather than the dynamics and behavior of individual or groups of particles.

A recent publication by Ushizima et al. [8] describes beam detection, a method aimed at automating the analysis and classification process of single high-quality particle beams in 2D LWFA simulations. That approach combines a bunch lifetime analysis and fuzzy clustering to estimate spatially confined beams. The bunch lifetime analysis describes the location of high-density features over time indicating when and where potential particle beams may exist. Fuzzy clustering is then used to detect single particle beams at individual timesteps. In contrast to that work, our goal is the detection of multiple bunches in a single simulation instead of just a single high-quality beam. Furthermore, we describe the classification of particle bunches based on the complete temporal path of particles and analyze the temporal evolution of particle bunches.

2.1.2. Region Growing

Motivated by region growing approaches in image segmentation [24, 25], we use a combination of 2D and 3D region growing to identify single particle bunches of interest, i.e., particle bunches that are compact and have high momentum in x direction (px). The general approach is to start with a set of seed points and from these grow regions by appending to each seed those neighbors that have predefined properties similar to the seed (such as specific ranges of variables) [24].

In our analysis we impose a uniform analysis-grid on the particle data. The 3D analysis grid is a density-grid indicating the number of particles within each grid cell, whereas the

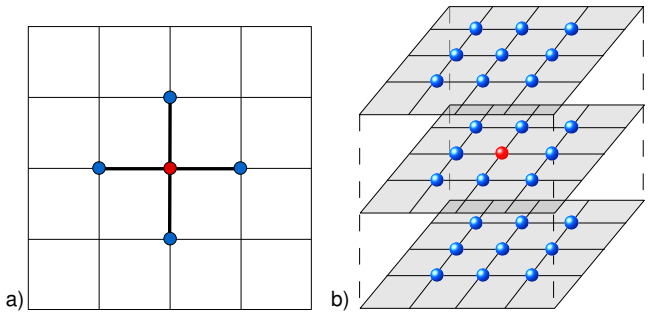

Figure 1. a) 4-neighbor stencil in 2D. b) 26-neighbor stencil in 3D. In both illustrations the reference grid-point is shown in red and its neighbors in blue.

2D analysis grid approximates the function $p_{max}(x, y)$ indicating the maximum value in the momentum in x direction (px) associated with each grid cell (see Section 4.4.1). In the region growing, we then use a single grid-point as seed. The seed point is automatically defined by the algorithm as selected maxima of the underlying function of interest (e.g., $p_{max}(x, y)$). Besides selection of proper seeds, criteria like topology (neighborhood) and stop criteria — described in detail in Section 4.4.2 — are important parameters of region growing algorithms. In the 2D case we use the 4-neighbor stencil and in the 3D case the 26-neighbor stencil (see Figure 1).

2.1.3. Ensemble Methods for Data Classification

The extreme size of LWFA simulation datasets prohibits the analysis of the complete data of the whole time series at once. Therefore, we first analyze each timestep separately to detect bunches of interest at each timestep. We then merge the information of this ensemble of particle bunch classifications to define a consolidated classification. Using this approach we: i) effectively reduce the amount of data we need to consider in later analysis steps, ii) increase performance of the analysis, iii) improve quality of initial analysis results, and iv) enable efficient analysis of the complete timeseries (see Section 4).

Methods that integrate multiple learned models into a single classification of the data are often referred to as ensemble methods. Ensemble methods are used in practice, e.g., for: i) knowledge reuse, i.e., several classifications are available which a user seeks to integrate into a common classification; ii) distributed data mining, e.g., in security sensitive applications where original data cannot be shared between different participating parties, or to iii) improve quality and robustness of classifications [26].

In pattern recognition applications, like hand writing recognition, ensembles of different classifiers help to improve quality, accuracy, and reliability of classifications [27]. In data clustering research, cluster ensembles help to improve the quality and robustness of traditional analysis methods such as k-means [28] as well as to improve the robustness and stability of instable classifiers such as neural networks [29]. We use the basic principle of ensembles to enable high-performance analysis and beam detection of extremely large 3D particle datasets.

2.2. High Performance Index/Query for Data Mining

In many cases, only a subset of the data is actually relevant to the data analysis. In the context of laser wakefield particle acceleration, e.g., only a fraction of all particles are accelerated to relevant energy levels and are of interest for the analysis. Therefore, to achieve good performance, we use at each stage of our analysis pipeline only a well-defined subset of the data, significantly reducing the amount of irrelevant data the algorithm needs to analyze. To maximize the benefit of such data reduction strategies, it is paramount that we are able to identify and access data subsets of interest fast. In this work we use state-of-the-art data management to be able to quickly: i) evaluate range queries, ii) compute conditional histograms, and iii) trace particles using ID queries; significantly improving the overall computational performance of the proposed analysis.

The commonly used strategy for accelerating these data accesses is called indexing in database terminology. A standard database indexing technique is the B-tree [30]. B-trees have properties favorable for applications that require frequent updates to the underlying base data and the index. Bitmap indices, in contrast, work best on read-only (or read-mostly) data — which is representative of most scientific and analytical applications — and can achieve faster query response times than B-trees in read-only applications [31, 32]. The core idea of a bitmap index is to use a sequence of bits to mark the positions of records satisfying certain

conditions. Each bitmap index consists of a set of bitmaps (or bit vectors), each of which represents a single or a range of values of the underlying data. Based on such an index, the data records that correspond to a range query like $X > 5$ can be identified without ever having to access the raw data [33, 34].

Without compression, the size of a bitmap index increases linearly with the number of bitmaps. To reduce index size, many different techniques have been proposed [35, 36]; the most common ones being binning, encoding and compression. Binning and encoding are different ways of controlling the number of bitmaps per index. Compression is used to reduce the size of individual bitmaps.

For this work, we make use of a bitmap index software called FastBit [9]. It implements the fastest known bitmap compression technique [37, 38], and has been demonstrated to be effective in a number of data analysis applications [39, 40]. In particular, it has a number of efficient functions for computing conditional histograms [41], which are crucial for this work. Furthermore, FastBit indices are relatively small compared to popular indices like B-trees [37, Fig. 7] and can be constructed much faster than others [33, Fig. 12]. Bitmap indices are well-known for their effectiveness on data having relatively small number of distinct values, such as gender. FastBit indices have also been demonstrated to be very efficient answering queries on data with a large number of distinct values through its unique compression [36] and binning [34].

We make extensive use of two additional features of FastBit specifically enhanced for the work in this paper: i) ID queries; and ii) a special function for computing 3D conditional histograms. To track particles of interest over time, we use FastBit to extract all particles with a given set of identifiers (IDs) from all timesteps via queries of the form $ID = 0||ID = 1||\ldots||ID = n$. This query may involve thousands or millions of IDs, which can take a long time just to parse the query string. FastBit provides a mechanism to directly input the list of IDs to reduce the query response time. This feature was also used in an earlier work by Rübel et al. [7].

FastBit has shown to be effective for computing 1D and 2D conditional histograms [41, 7]. In this work, we extend FastBit to enable fast computation of 3D conditional histograms and to export bit vectors for representing particles (data records) associated with the bins of a histogram. These bit vectors allow us to directly retrieve the information related to particles of interest, enabling fast data-access and leading to significant improvement of the computational performance of our analysis pipeline.

2.3. Visualization

2.3.1. High-Performance Visualization

We make use of state-of-the-art visualization methods to validate and investigate analysis results. Many commercial and open source visualization packages are available for visualization of large scientific data, such as, VisIt [10, 42], ParaView [17], or EnSight [43]. We use VisIt because it provides several unique features crutial for this work. First, VisIt provides direct integration of FastBit. Second, VisIt supports selection of particles based on particle IDs via the concept of *named selections* [7]. Once we define which particles form a bunch, this capability allows us to save the IDs of the selected particles in the form of a named selection, and then apply this selection directly to the original data. This concept enables a seamless integration of derived information from our analysis and the original data without having to modify the raw data. VisIt is also the project-wide visualization application of our collaborators at the LOASIS program reducing the cost for deployment of the analysis to the end user (e.g., training and maintenence cost).

2.3.2. Query-Driven and Feature-Based Visualization

The term *query-driven visualization* (QDV) is coined to describe a combination of high-performance indexing and querying capability with visual data exploration tools [44]. Aimed towards the analysis of massive datasets, QDV allows users to quickly search the data for features of interest. The visualization is then focused on the selected features, thus significantly reducing the amount of data presented to the user. The concept of QDV has been successfully applied to a wide range of applications, e.g., network traffic analysis [39] and visual exploration of LWFA simulation data [7].

Visualization techniques, like QDV, that focus on specific features of the data are also referred to as *feature-based visualizations*. In context of QDV, the features of interest are usually defined by the user via dedicated data queries. Other featured-based visualization methods automate the feature detection step by choosing an appropriate filtering process [45]. As part of this work, we present a novel method for automatic feature detection in LWFA data and present dedicated visualizations for analyzing the detected features, namely particle bunches.

3. Data Overview

Before describing our method in detail in Section 4, we first provide an introduction to laser wakefield particle acceleration and their simulation. The basic concept of an LWFA is to use a short ($\lesssim 100fs$), ultrahigh intensity ($\gtrsim 10^{18}W/cm^2$) laser pulse to drive waves in a plasma. In a hydrogen plasma, the radiation pressure of an intense laser pulse displaces the electrons while leaving the heavier ions stationary. Together with the space-charge restoring force of the ions, this displacement drives a wave (wake) in the plasma. If the wake is high enough in amplitude, electrons can become trapped and accelerated by the plasma wave, and eventually decelerate again as they outrun the wake. The electric and magnetic fields that can be achieved in an LWFA are thousands of times stronger than in conventional accelerators [5], enabling particles to be accelerated to high energies within very short distance. Recent experiments at the LOASIS program [2] have demonstrated high-quality electron beams [3] and energies up to 1GeV using cm long plasmas [4]. Figure 2 shows a snapshots of an LWFA simulation illustrating the acceleration process.

To better understand nonlinear plasma response, beam trapping, self-consistent laser propagation, and beam acceleration — processes not accessible to analytic theory — LWFA experiments are computationally modeled. Traditionally explicit particle-in-cell (PIC) simulations [46] are used to model LWFA experiments [47, 48]. PIC simulations — using, e.g., VORPAL [49], OSIRIS [19] and others simulation codes [50] — self-consistently model the interactions of the laser, plasma, and particle bunch. This self-consistency property is important in the case of plasma accelerators because the shape of the laser pulse forms and evolves through its interactions with the wake: both the laser and the plasma evolve together, during which process a large portion of the laser energy is transferred into the plasma. Balancing this process is the basis for forming of a stable accelerating structure.

Here we concentrate on datasets produced by electromagnetic PIC simulations in VORPAL [49], a parallel, object-oriented plasma simulation code, which can model the behavior of charged particles in their self-consistent electromagnetic field. The particle motion, as well as the field evolution can be modeled with various approximations, ranging from fluids to fully kinetic particles and with an explicit electromagnetic or an electrostatic treatment of the field equations. The code has been successfully used across a number of disciplines within DOE's Office of Science, including ultra-high gradient laser-plasma acceleration of electron beams, electron cooling of heavy ion beams, and implicit

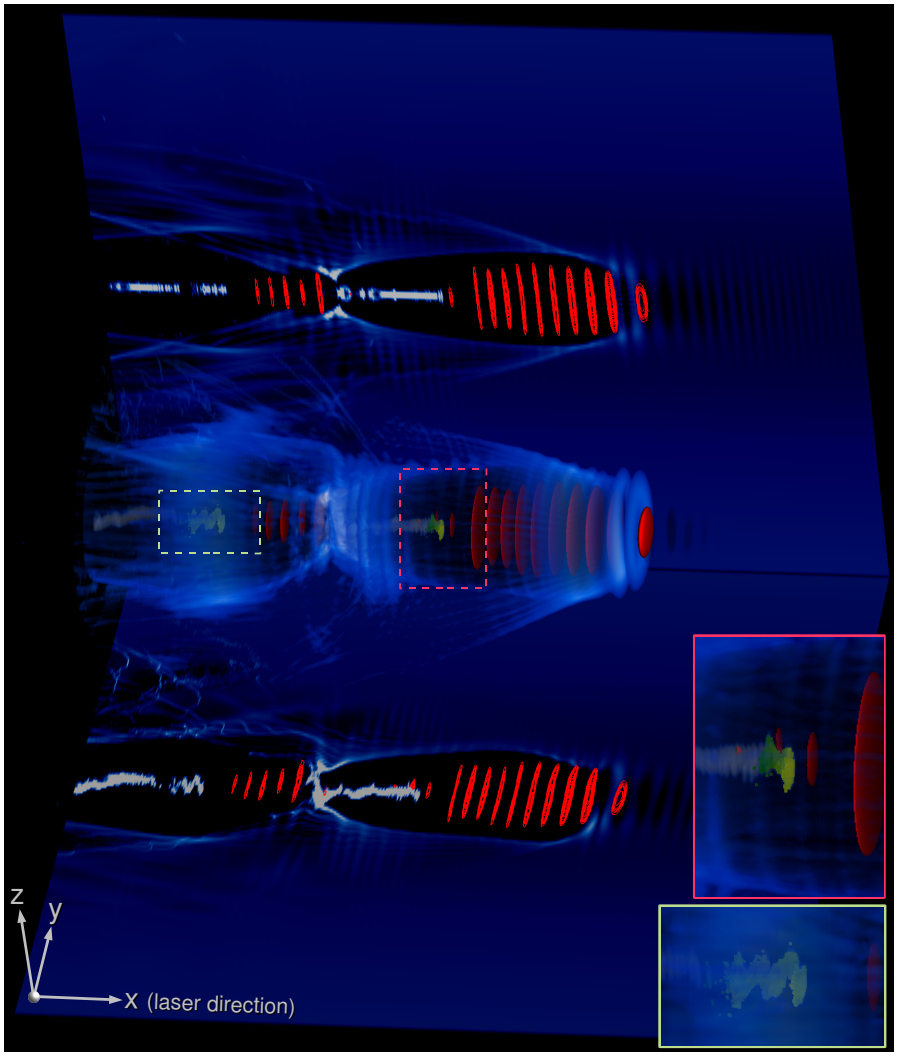

Figure 2. Volume rendering of the plasma density illustrating the three-dimensional structure of the wake (blue). A set of particles trapped in the wake that are accelerated to high energy levels ($px > 5 * 10^{10}$) are shown in green/yellow with green being medium and yellow being high momentum in x direction px. Contours (red) of the electric field strength in z direction (E_z) illustrate the laser pulse (which moves along the x axis from left to right). The panels at the bottom and back show a slice through the center of the volume in the x/y and the x/z plane, respectively. The inset views show close-ups of the two main bunches of accelerated particles. This visualization shows timestep $t = 22$ of the 3D dataset E used later for validation of the proposed methods (see also Table 1).

electromagnetic treatment of plasma edge phenomena in tokamak plasmas.

When modeling laser wakefield accelerators, VORPAL is used as PIC code. In this method, collections of real charged particles are modeled as computational macro-particles that can be located anywhere in the computational domain. The electromagnetic field is spatially discretized. Particles are moved under via Newton-Lorentz force obtained through interpolation from the fields. The current carried by the moving particles is then deposited

onto the simulation-grid to solve Maxwell's equations for the fields.

Due to the large computational resources required to accurately model a LWFA it is not practical to simulate the entire hydrogen plasma at once. To save computational resources and storage space VORPAL employs a moving window simulation approach. In this method, only a region around the laser pulse is simulated at each timestep. As the laser pulse is traveling through the plasma, the simulation window is moved along with the laser at the speed of light.

VORPAL uses two different kinds of output, one for storing snapshots of the entire simulation state at a particular point in time ('dumps') and the other being time histories of selected quantities. The dump data is both used for extracting simulation results, as well as for checkpoint/restarting operations. Dumps consist of particle data, field data, and auxiliary state data, all written according to Vizschema [51], a self-describing data organization scheme for scientific data designed to facilitate the visualization of its contained data.

For the analysis discussed here we concentrate on particle data. Each particle is represented as a vector of seven quantities in the 2D case $(x, y, px, py, pz, id, wt)$ and eight in the 3D case (including z). The quantities x, y, and z are measured in meters (m) and describe the physical location of a particle. px, py, and pz are in $\frac{m}{s}$ (γv) and describe the momentum of a particle in x, y, and z direction respectively. wt then describes the weight of each macro-particle defined by the number of electrons it represents. id is a unique identifier for each particle. Due to the large amount of data stored per dump, scientists have to find a compromise between available storage and the accuracy of the reconstructed trajectories.

The design of our analysis pipeline allows us to quickly reduce the amount of data we need to consider in the analysis. Being able to efficiently access the relevant data subsets is crucial to enable good computational performance. We augment the data with FastBit bitmap indices and use HDF5-FastQuery [52] — a data access API based on HDF5 [53], H5Part [54, 55] and FastBit [9]— to enable efficient access to the data. The same data interface was used in earlier work to enable high-performance visualization of LWFA simulation data [7].

4. Method

We now describe our algorithm for automatic detection and analysis of beam paths in laser wakefield particle accelerator simulation data. Section 4.1 defines the basic requirements our analysis needs to fulfill; we here discuss how a particle beam is defined and describe additional assumptions we are making in the analysis process. In Section 4.2 we then give an overview of the complete analysis pipeline and describe the different steps of the analysis in Sections 4.3- 4.7.

4.1. Feature Definition

In our analysis we seek to find particle beams that are characterized as follows:

- **F1:** A beam is defined by a compact bunch of accelerated particles (i.e., particle with high px values) condensed in x, y px, py space. In 3D simulations also z and pz. This means: i) the particles forming the bunch have high px values, and ii) the particle bunch is compact in physical as well as momentum space (see, e.g., Figure 5b).
- **F2:** In the simulations, the laser pulse is centered in the plasma at $y = 0$ and $z = 0$ and moves along the x axis. The laser pulse traveling through the plasma induces waves in the plasma which in turn accelerate the particle beams (see Figure 2). Particles are accelerated in the same direction as the laser pulse, here x direction.

- **F3:** While being trapped in the wave, the particles forming a bunch are accelerated over a period of time until they eventually outrun the wave and decelerate again. These particle bunches are therefore present over a period of time. Therefore, during the time they exist, the particle bunches themselves define a temporally coherent feature of the data.
- **F4:** The particle bunches created by the laser pulse will manifest as peaks in x, y, px space (see, e.g., Figure 5b or Figure 8).
- **F5:** In the analysis we use py and pz to assess the quality of particle beams as well as to define the distance of a particle to the beam . We do not, however, use py and pz to detect condensed particle bunches at single timesteps because a particle beam is usually not defined by a single condensed group of particles in y/py (or z/pz) space. Due to the characteristic oscillating transverse motion of accelerated particles, a particle beam usually constitutes as two (or more) condensed groups of particles in y/py (or z/pz) space (see, e.g., Figure 17 b and c). The transverse momenta py and pz are hence symmetric variables centered around zero.
- **F6:** Particles may become trapped in different periods of the plasma wave. In practice, several particle beams may, therefore, form in different periods of the wave and possibly coexist at the same time. Furthermore, after a beam has decelerated (see F3), a new beam may form later in time within the same period of the wave, i.e., at different times of the simulation, we may find different particle beams at similar locations within the simulation window.

Further, we make the following assumption:

- **A1:** Within each main peak in $x/y/px$ space, we find only one high quality bunch of interest at a time. As we show later, in cases where several bunches may exist within the same peak, the algorithm makes an automatic, implicit decision and retrieves the bunch that defines the highest density feature within that peak for the longest period of time. This bunch is often the one exhibiting the highest acceleration.

As described in Section 3, only a fraction of all timesteps of a simulation are saved to file. Even though the temporal resolution of the saved data is usually not high enough to resolve the oscillation frequency of the wave, it is selected by the user to be sufficient to resolve acceleration and dephasing (i.e., the process during which a beam loses its coherency). It is, therefore, safe to assume that:

- **A2:** Between two consecutive timesteps, a particle bunch does not disappear while a new bunch appears at the same location (see also F6).

4.2. Overview of the Algorithm

The number of particles in a beam of interest is much smaller than the total size of the data. To detect these bunches efficiently, we designed an analysis pipeline that allows us to quickly reduce the amount of data we need to consider. Using state-of-the-art data management based on HDF5 [53], H5Part [54, 55] and FastBit [9, 52] we are able to efficiently extract the portions of the data relevant for the analysis. In order to be able to accurately detect and classify particle beams, we need to consider information of the complete timeseries. We initially analyze each timestep separately to collect information on the particle beams. This information allows us to significantly reduce the amount of data we need to consider in the later analysis of the temporal particle paths.

Figure 3 provides a high-level overview of the general structure of the analysis pipeline. The main steps of the analysis pipeline are explained in detail in Sections 4.3 to 4.7. In the

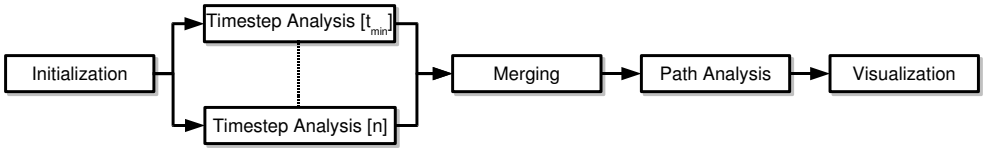

Figure 3. Overview of the algorithm design: In the initialization step we compute which timesteps need to be analyzed. Afterwards, each timestep is analyzed independently in order to identify the most prominent particle bunch at each timestep. Based on this information about the particle bunches at individual timesteps, the merging step defines a single consolidated description of the data. In the merging step we: i) Identify the number of detected bunches; ii) Compute for each bunch a set of candidate particles; and iii) Compute for each bunch a discrete frequency function describing how often the individual candidate particles were found to be part of the bunch. In the particle path analysis we then trace all candidate particles over the complete timeseries and compute for each bunch a reference path. Based on the reference path we then identify the different phases of a bunch, such as acceleration and deceleration, and define for each candidate particle the distance to the bunch. After completion of the analysis process we investigate analysis results using dedicated visualization methods.

analysis, we distinguish between two different types of particles. *Candidate particles* are all particles that were ever detected by the analysis as being part of a particular bunch, i.e., particles that met the bunch criteria described in Section 4.1 at least once in any timestep. *Reference particles* are a subset of candidate particles found with a high bunch *frequency*, i.e., particles with a high degree of temporal persistence within a bunch.

For illustration purposes, we use a medium-sized 2D dataset, labeled dataset C, to describe the different steps of the analysis (see Table 1 for more details). In the later validation of the analysis, we then present results for a large 3D dataset and various 2D datasets (see Section 5). The example dataset shows a fairly complex acceleration behavior and contains bunches of different quality at high as well as low energy levels. In the example dataset, our analysis detects two main particle bunches that we refer to as *first* and *second* bunch in the following: the first bunch is formed earlier in time and is, therefore, detected first by the analysis.

4.3. Initialization

The main purpose of the initialization step is to calculate the minimum timestep t_{min}, which indicates those timesteps that are relevant for the initial timestep analysis. As described earlier, a particle beam consists only of accelerated particles, i.e., particles with high px values. At early timesteps of the simulation, no particles at a sufficient level of acceleration exist since the plasma waves —induced by a laser pulse traveling through the plasma— are just forming. Hence, these early timesteps are not relevant in the initial timestep analysis described later in Section 4.4. Often the user has already a good understanding of when the first particle beams are forming in the data based on earlier analysis. We, therefore, allow the user to either: i) define t_{min} manually as input parameter of the analysis or ii) have the algorithm estimate a good value for t_{min} automatically.

In cases where the user has no prior knowledge of the value for t_{min}, we use the following approach. We first compute for each timestep t the number of particles $h(t)$ that satisfy the condition $px > 10^{10} ms^{-1}$. This condition ensures that we only consider particles with a sufficient level of acceleration. The condition $px > 10^{10} ms^{-1}$ is explained further in Section 4.4.1. Based on $h(t)$, we compute the average number a_p of accelerated particles at timesteps t with $h(t) > 0$, i.e., $a_p = \frac{\sum_{t=0}^{n} h(t)}{m}$, with n being the total number of timesteps

and m being the number of timesteps with $h(t) > 0$. We define t_{min} as the first timestep t with $h(t) > a_p$. This ensures that we consider only timesteps ($t \geq t_{min}$) with a sufficient number of accelerated particles where we can expect to find a well defined bunch that meets the criteria defined in Section 4.1.

This automatic approximation works especially well in those cases where the number of accelerated particles $h(t)$ reaches a relatively stable state. In complex cases where the number of accelerated particles $h(t)$ varies largely over time, the condition $h(t) > a_p$ may, however, lead to a suggestion of a too late minimum timestep t_{min}, in which case a different threshold of, e.g., $h(t) > (0.5 * a_p)$ could be used. After the initialization is complete we perform for each relevant timestep the analysis described in the next section.

4.4. Timestep Analysis

The initial timestep analysis is aimed at identifying the most prominent particle bunch at each timestep, i.e., the bunch associated with the particle wave having the highest px value. The goal is to identify a group of candidate particles for the different bunches, i.e., a set of particles that potentially belong to a bunch. Focusing only on these much smaller sets of candidate particles greatly reduces the workload for the later particle tracing. The algorithm executes the timestep analysis independently for each timestep. As illustrated in Figure 4, the timestep analysis consists of two main steps: i) data preparation and ii) bunch segmentation described in Sections 4.4.1 and 4.4.2, respectively.

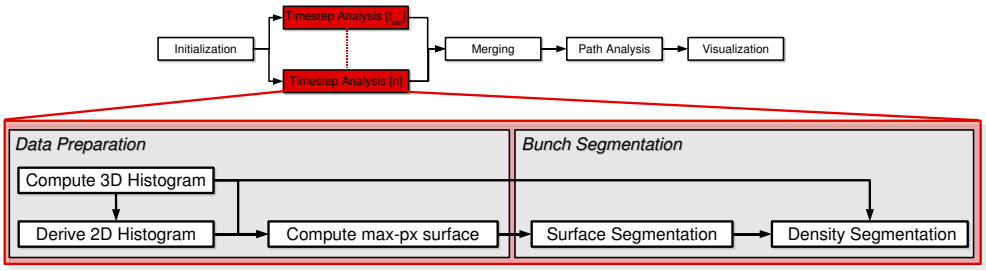

Figure 4. Overview of the timestep analysis: We use a grid-based segmentation approach to identify particle bunches of interest at each timestep. We first compute a 3D histogram in $x/y/px$ of all accelerated particles ($px > 10^{10} ms^{-1}$). From the 3D historgam we then also derive the 2D histogram in x/y. Based on the information from the two histograms we then compute the function p_{max} defining the maximum px value found at each (x, y) location of the used analysis-grid. We segment the surface defined by p_{max} to identify the region of interest (ROI) in physical space (x/y) and then identify the most condensed particle bunch within the ROI using a second density-based segmentation approach.

4.4.1. Data Preparation

The data preparation step is executed once for each relevant timestep (i.e., $t \geq t_{min}$) and is aimed at initializing all data structures needed for the later bunch segmentation step (see Section 4.4.2). We use two grid-based data structures in the bunch segmentation step of the analysis pipeline: i) a 2D analysis-grid defined in the physical domain (x/y); and ii) a 3D analysis-grid defined in $x/y/px$ space. With the 3D analysis-grid we associate a scalar field defining how many particles belong to each grid point, i.e., we compute a 3D histogram defined over the domain $x/y/px$. The 2D analysis-grid approximates a second scalar function, p_{max}, describing the maximum px value of all particles at a given location. We compute p_{max} directly based on the information of the 3D histogram. In the later bunch segmentation step

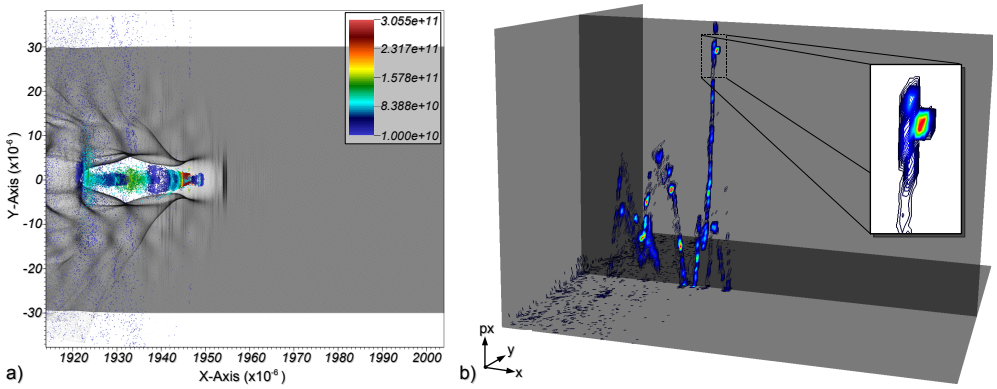

Figure 5. Visualizations of timestep $t = 40$ of the example 2D dataset consisting of $\approx 2.4 * 10^6$ particles. a) All particles shown in physical space (gray) and all particles that satisfy the condition ($px > 10^{10} ms^{-1}$) ($\approx 1.7\%$ of all particles) colored according to px. b) Iso-contours of the particle density shown on three slicing planes to illustrate the basic structure of the 3D particle density in x, y, px. The iso-contours are colored according to their value with black/blue being low, green/yellow being medium, and red being high density. The inset plot shows a close-up view of the main region of interest containing a condensed particle bunch.

we first identify a region of interest (ROI) in physical space via segmentation of p_{max} and then identify the particle bunch of interest within the ROI based on the 3D histogram.

3D Histogram Computation: In the 3D histogram computation we compute: i) the count of each bin of the 3D histogram, and ii) a set of bit vectors that indicate which particles are associated with each bin of the 3D histogram.

From **F1** (see Section 4.1) we know that a particle bunch of interest will consist only of accelerated particles. The expected wake oscillation is up to $px = 10^9 ms^{-1}$ and the particle beams of interest should be observed near $px = 10^{11} ms^{-1}$. At each timestep we therefore consider only particles that satisfy the condition $px > 10^{10} ms^{-1}$, which ensures that we only consider particles with a momentum in the x direction above the base oscillation of the wave while including all particles that are potentially of interest. As illustrated in Figure 5a, this condition significantly reduces the number of particles we need to consider, i.e, usually to only $\approx 1 - 3\%$ of all particles, significantly improving the performance of the analysis.

To be able to identify condensed particle bunches at single timesteps (i.e., bunches within the particles having $px > 10^{10} ms^{-1}$), we use a 3D histogram of x, y, and px with a resolution of typically 100 bins per variable and the condition $px > 10^{10} ms^{-1}$. Consistent with **F5** (see Section 4.1), we do not consider the transverse momenta py and pz at this stage of the analysis. In x and y, we compute the 3D histogram over the complete extents of the simulation window at the current timestep. Since the size of the simulation window is constant in x and y over time, we can correlate bins of 3D histograms from different timesteps directly via their index in (x, y). As illustrated in Figure 5b, the 3D histograms will always be sparse with accelerated particles appearing only behind the laser pulse — located roughly in the center of the simulation window — and most concentrated around $y = 0$.

Whereas the 3D histogram helps us to identify particle bunches of interest, the later bunch segmentation requires access to individual particles located within histogram bins (see Section 4.4.2). To accelerate the access to individual particles, we make use of a dedicated bit vector data structure in FastBit. We have added dedicated 3D histogram functions to FastBit in which we compute the actual counts of the 3D histogram as well as a set of bit vectors

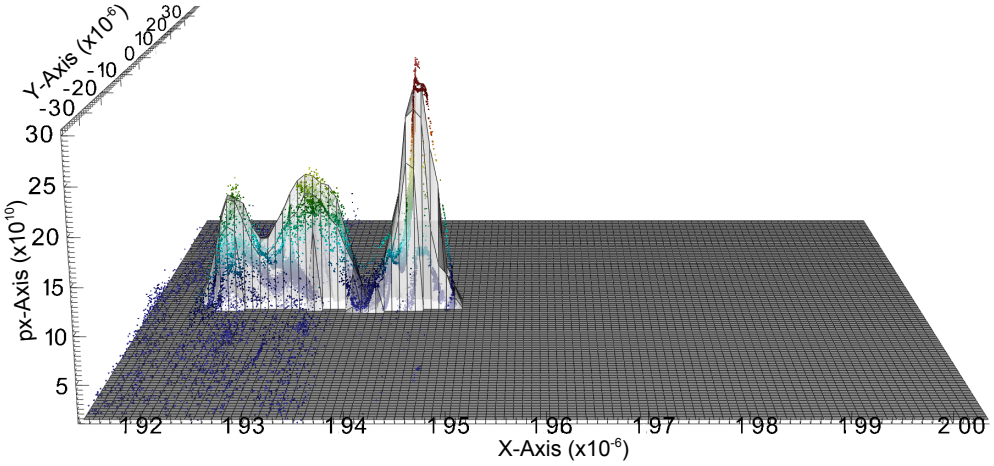

Figure 6. Surface plot showing the approximated maximum px function p_{max} defined over the x/y domain at timestep $t = 40$. Each point of the surface represents one column of the 3D histogram in (x, y) and is located in its center. The z coordinate of each point is defined based on the p_{max} function value computed for the according grid-point. In addition to the p_{max} surface all particles with $px > 10^{10} ms^{-1}$ are shown colored according to px. We can see that major peaks of p_{max} characterize the main regions of interest in physical space well.

that indicate for each non-empty bin which particles belong to that bin. These bit vectors are compressed and provide a memory and computationally efficient way for creating and storing the inverse mapping from a 3D histogram to the original data. In uncompressed form. a bit vector consists of n bits each representing one particle. Bit vectors can efficiently be merged using bitwise OR operations, which allows us to also efficiently access the data of many bins at once. Since the 3D histograms are always sparse, the memory overhead due to the bit vectors is in practice small, usually $\approx 2 - 5MB$ per timestep depending on the data distribution and size of the dataset.

Maximum *px* Function Computation: To be able to identify regions of high particle acceleration we compute, based on the information of the 3D histogram, the maximum px function:

$$p_{max}(x, y) = max(px(x, y)), \tag{1}$$

defined over the physical domain (x/y). This function associates with each point in physical space the maximum px value of all particles found at that location. We approximate this function by defining a 2D analysis-grid in physical space with the same resolution as the 3D histogram, i.e, usually 100 bins per variable. For each (x, y) column of the 3D histogram we compute the maximum px value by identifying the first bin (from top to bottom) with a count larger than zero using the upper bin-boundary in px as reference. In cases where outlier behaviors imposes a problem one could instead use the px value of the highest-density bin of each (x, y) column.

To ensure that the analysis is always focused on the areas of highest particle density, we perform a density-based filtering of p_{max} based on the 2D histogram in x/y. To avoid unnecessary accesses to the raw data and improve the performance of the algorithm, we derive the 2D histogram in x/y from the 3D histogram by adding up the counts of each (x, y) column of the 3D histogram. We then set the p_{max} function values at all grid-points with a particle density of less than 3% of the maximum 2D density to the minimum function

value. This density-based filter removes small peaks in the outer sparsely populated parts of the simulation window while preserving the main peaks of interest in the center.

To remove minor variations in the approximation of p_{max} and ease segmentation of the p_{max} function, we optionally allow the user to smooth the p_{max} function. For the results presented in this paper we approximate the values of the maximum px function p_{max} by smoothing the computed p_{max} function values using the following smoothing-kernel:

$$\begin{bmatrix} 0.075 & 0.075 & 0.075 \\ 0.075 & 0.4 & 0.075 \\ 0.075 & 0.075 & 0.075 \end{bmatrix}$$

derived through empirical study. Figure 6 illustrates the structure of the derived surface at timestep $t = 40$ of the example dataset. We can see that the derived surface approximates the general structure of the function well. The smoothing causes the maximum level of the surface to be lower than the peak px of the particles but does not affect the location of the maxima of p_{max}. Peaks with a low support are removed by the 2D density filter.

4.4.2. Bunch Segmentation

Bunch segmentation is aimed at identifying a single particle bunch of interest at a given timestep. This process is initially executed only once per timestep. Additional bunch segmentations may then be performed later at selected timesteps during the merging process. The segmentation of a particle bunch is performed in a two-step process. We first perform a 2D segmentation based on p_{max} to identify the region of interest (ROI) in physical space (x/y). To identify the most compact particle bunch within the ROI, we perform a second 3D density-based segmentation in $x/y/px$ space.

We use region growing for the segmentation itself. While the neighborhood and stop criteria are different for the 2D and 3D region growing, the basic algorithm is similar. In the region growing we maintain two lists: i) a list of selected points, and ii) a list of candidate points. The first list defines all points that have been identified as being part of the ROI and the second list contains all points that potentially belong to the ROI but still need to be checked. Initially the list of candidates contains only a single seed-point and the list of selected points is empty. We iterate through the list of candidates until no more candidates remain. For each candidate we check whether it satisfies a set of criteria (the so-called stop criteria). If the candidate point meets the criteria then it is moved to the list of selected points and its neighbors are added to the list of candidates, otherwise it is removed from the list of candidates.

To identify the ROI in physical space we first execute a 2D segmentation based on p_{max}. Using the global maximum of p_{max} as seed-point we perform a 2D region growing to identify the region in physical space associated with the seed. In this process we use the 4-neighbor stencil (see Figure 1) to define the neighbors of a given point of the 2D analysis-grid. We add a given candidate point to the list of selected points if it does not define a minima in x-direction of p_{max} in (i) $x/y/px$ space nor (ii) x/px space. The second condition prevents bleeding of the segmentation into secondary peaks of p_{max}. In this particular case we do not use the strict definition of a minimum, but define a point to be a minimum if it does not have any neighbor in x-direction with a p_{max} value smaller than its own. In the segmentation process we ensure that the ROI in physical space is closed and contains no holes.

The selection defined by the 2D segmentation is usually too large, i.e., it includes many particles in the vicinity of the bunch that do not actually belong to it. To ensure that the segmentation result is always centered around the most condensed bunch, we perform a secondary 3D density-based segmentation within the given 2D ROI. In this process we use

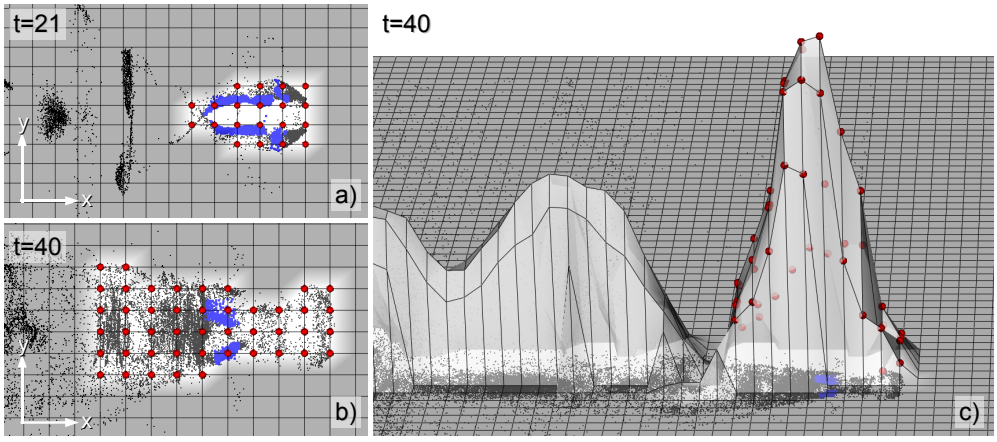

Figure 7. a) Overview of the segmentation process as performed at timestep $t = 21$ of the example dataset. The grid of the segmentation surface is shown in black. The points of the surface –each corresponding to one x/y column of the 3D histogram– selected by the initial surface segmentation step are shown in red and the corresponding area is highlighted in white. All particles with $px > 10^{10} ms^{-1}$ are shown in black and all particles selected by the final 3D density-based segmentation are shown in blue. b) Same as *a* but for timestep $t = 40$. c) Same as *b* shown as a 3D rendering. The surface is the same as shown in Figure 6. The initial surface segmentation defines the region of interest (ROI) in physical space. The density-based segmentation then identifies the most condensed group of particles within the ROI.

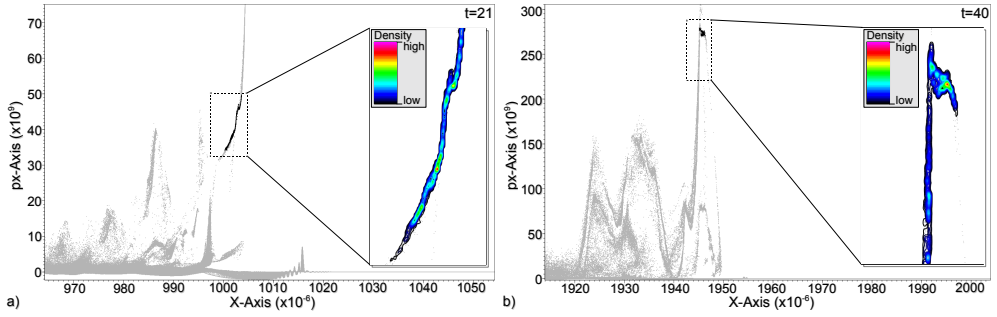

Figure 8. Example segmentation result at timestep $t = 21$ (a) and $t = 40$ (b) shown in x/px space (see also Figure 7). The particles identified by the segmentation process are shown in black, all other particles are shown in gray. The inset plots show a close-up view of the detected bunches with additional iso-contours of the particle density colored using the indicated color mapping. Due to the density-based segmentation approach the algorithm is able to detect condensed particle bunches at high (b) as well as low (a) energy levels.

the bin of the 3D histogram having the highest density within the identified region in physical space as the seed. In the 3D region growing, we use the 26-neighbor stencil (see Figure 1) and stop if the density value of a candidate point is below 20% of the density-value of the seed. This stop criterion of 20% of the seed-point density has shown in our experiments to provide a good tradeoff between high and low selectivity of the segmentation. In the later merging step of the analysis pipeline, the IDs of the particles selected by the bunch segmentation are used to identify corresponding particle at different timesteps. After completion of the segmentation we load the IDs of the particles associated with the selected bins. The bit vectors computed together with the 3D histogram enable us to directly access the according particle IDs.

Figure 7 illustrates the segmentation process for two selected timesteps from the example 2D dataset. Figure 8 then provides an overview of the result of the bunch segmentation at the same two timesteps. As we show in Section 4.5, the bunches shown in Figure 7 and 8 in fact define two different main bunches of interest in the example dataset.

Our bunch segmentation algorithm has several significant characteristics relevant for the later merging and path analysis. First, the bunch segmentation is always centered around the highest density of particles and, hence, centered around the main bunch of interest. This property of the segmentation ensures that the particles that define the "core" of a bunch of interest are always found as being part of the bunch at more timesteps than particles that are in its larger vicinity. Second, like most segmentation algorithms, the bunch segmentation may suffer from under-segmentation (too large a selection) and over-segmentation (too small a selection) errors. The potential effects of under- and over-segmentation are taken into account in the later particle path analysis so that our method deals robustly with these potential errors. In the context of a cluster ensemble, these effects are in some sense even desirable since they ensure diversity. In the context of the complete analysis pipeline, under-segmentation errors simply lead to inclusion of particles distant to the actual bunch in the list of potential candidates. These particles are identified later in the particle path analysis and do not affect the quality of the analysis. Our two-step segmentation process furthermore ensures that in all cases only particles within a relatively small region in physical space are selected so that the amount of potentially improperly selected particles is in general low. The potential effects of over-segmentation errors are accounted for when selecting a set of reference particles for each bunch described later in Section 4.6.2.

4.5. Merging

In the timestep analysis, we identified the single most prominent bunch for each timestep. For each of these bunches the timestep analysis computes i) a reference location describing the principal location of the bunch —here defined as the x-index $x_i(t)$ of the segmented maximum of the function p_{max}— and ii) the IDs of the particles that were found to form the bunch. Based on this information from the individual timesteps, the merging step: i) identifies those bunches in the various timesteps that represent the same physical bunch and ii) defines a single consolidated description for each of the different bunches. As described in Section 4.1 *F6*, in practice several bunches of interest may exist at the same time, e.g., one bunch in the first and another one in the second period of the wave behind the laser pulse. We therefore trace the identified bunches forward and backward in time — performing additional segmentations at the individual timesteps — to complete the information about each bunch and ensure accuracy of the initial analysis (see Figure 9).

A fundamental problem with the merging step is how to correlate bunches from different timesteps. In our algorithm, we identify corresponding bunches based on their reference location $x_i(t)$. The second fundamental problem we have to solve is how to combine the segmentation information from different timesteps. In order to define a single consolidated description of each of the different bunches, we define for each bunch i) a list of *candidate particles* containing all particles that were found at least once as being part of the bunch; ii) a so-called *bunch frequency* function describing how often each of the candidate particles was found as being part of the bunch. The term *frequency* is used here to describe at how many timesteps a particle is part of a particular bunch.

In the following parts of this section we describe how we solve the two fundamental problems of correlating different bunches and combining their segmentation information, and how the initial merging is performed. Next, we describe the forward- and backward tracing

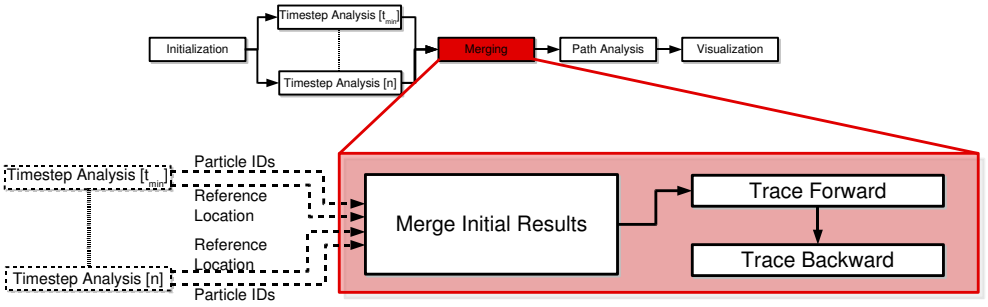

Figure 9. Overview of the merging process: In the merging step of the algorithm we first correlate the detected bunches via their reference location and define a single consolidated description for each of the different bunches. Instead of one bunch per timestep we now have a set of bunches each existing over a series of timesteps. Afterwards, we trace the identified bunches forward and backward in time to complete the information about each bunch and ensure accuracy of the initial analysis.

and present the merging results for the example dataset.

Correlating Bunches from Different Timesteps: In the merging, we correlate bunches from different timesteps via their reference location $x_i(t)$. As explained earlier in Section 3, the simulation employs a moving window approach in which the simulation window is moved along the cavity while the laser pulse is traveling through the plasma. The accelerated particles and the laser pulse both move roughly at the same speed as the simulation window, i.e., the speed of light c. Furthermore, the size of the simulation window in x is constant. The relative x-location of maxima in p_{max} are therefore expected to be relatively stable, i.e., their maximum movement in x-direction between two consecutive timesteps is restricted by the difference of the particle velocity and the speed of light, which is small in the case of relativistic particles. In the case of 100 bins per variable, the maximum slippage x_s in x between two time step is therefore usually 1-2 bins. In order to identify whether two bunches detected at timestep t and $t+1$ are the same we only need to check whether $x_i(t) - x_s \leq x_i(t+1) \leq x_i(t) + x_s$.

Correlating bunches based on $x_i(t)$ has shown in practice to be a reliable approach for identifying corresponding bunches since it does not make any hard assumptions on the minimal temporal resolution of the data. In terms of temporal resolution of the data we here only assume that the assumption $A2$:*"Between two consecutive timesteps a particle bunch does not disappear while a new bunch appears at the same location"* is true (see Section 4.1). Note, the algorithm does not make any assumptions about the acceleration/deceleration process itself, i.e., the algorithm deals robustly with changes in px.

By using $x_i(t)$ to correlate different bunches we furthermore assume that $A1$: *"Within each ROI in x/y space we find only one high quality bunch of interest at a time."* is true (see Section 4.1). In practice, $A1$ has shown to be a reasonable assumption. In cases where several potential bunches exist within the same peak in $x/y/px$ space, the merging procedure combines the two bunches. As explained below, in this case the algorithm will make an automatic, implicit decision and retrieve the bunch that defines the highest density feature for the longest period of time which usually is the bunch that shows the highest acceleration.

Combining the Segmentation Information from Different Timesteps: In order to define a single consolidated description for the different bunches, we need to combine the information from the different timesteps, i.e., the lists of particle IDs defining a bunch. The goal here is to define for each bunch: i) a list of candidate particles describing which particles

potentially belong to the bunch, and ii) identify a list of reference particles which we are certain belong to the bunch.

In order to define the list of candidate particles, we simply merge the particle ID lists from the different timesteps that define the same bunch. In this process, we count for each particle ID how often the corresponding particle was found as being part of the bunch. For example, when a particle is found at 10 different timesteps as being part of a particular bunch then this particle is assigned a count of 10. This function, $l_b(id_i)$, defines for each bunch b a discrete frequency measure also referred to as bunch frequency. The bunch frequency indicates the likelihood of a particle with $ID = id_i$ of being part of a particular bunch, i.e., the more frequently a particle was detected as being part of a bunch, the more likely it is that this is in fact the case. Based on l_b, we then identify a set of reference particles for a particular bunch b by selecting the particles with the highest bunch frequency values l_b. Further details are presented in Section 4.6.2.

To combine the segmentation information from all timesteps, we iterate through the results from the initial timestep analysis and check whether the bunch detected at timestep t corresponds to the bunch detected at timestep $t-1$. If the two bunches are the same then we merge their particle ID-lists and update l_b accordingly. If the two bunches do not correspond then we increase the number of detected bunches, create a new list of candidate particles for the new bunch, and create a new bunch frequency function l_{b+1}.

Forward- and Backward Tracing: In the initial timestep analysis we detected only the single most prominent bunch at each timestep. To account for the fact that several bunches may coexist at the same time (see Section 4.1, F6), we first trace the detected bunches forward and afterwards also backward in time in order to complete the information for each bunch. In the following, we refer to the last timestep at which the algorithm found a bunch b as $t_{last}(b)$.

Starting from $(t_{last}(b) + 1)$, the forward tracing checks if a maximum of the function p_{max} that can be segmented exists at approximately the same location ($\pm x_s$) where the bunch b was previously found (i.e., $x_b(t_{last}(b))$). If such a maximum of p_{max} exists, then we perform the bunch segmentation step for the identified location and merge the information of this new segmentation with the current description of the bunch b — i.e., we update the list of candidate particles, t_{last}, and l_b accordingly — and continue with the forward tracing. If we do not find a maximum that can be segmented within $x_b(t_{last}(b)) \pm x_s$, then we terminate the forward tracing for the current bunch b and switch to the next bunch $(b+1)$. In the forward tracing, we may also detect that two previously separated bunches are actually the same. This may be the case when two bunches (e.g. b_1, b_3) correspond via their principal location but the respective bunch was previously not segmented at all timesteps, e.g., the bunch was only segmented at timestep t and $t+2$. If we now detect in the forward tracing that the same bunch also exists at timestep $t + 1$ then we have closed the temporal gap in the description of the bunch indicating that the previously separated bunches b_1 and b_3 are actually the same. In this case we merge the descriptions of these bunches after closing the temporal gap and continue the forward tracing.

To ensure that we have segmented each bunch at all timesteps at which it exists, we now also have to trace the different bunches backward in time. Note, in contrast to the forward tracing we here do not have to check for whether two previously separated bunches are actually the same. All these cases have already been resolved in the forward tracing. The purpose of the backward tracing is only to gather additional information about the different bunches.

Merging Results for the Example Dataset: In the example dataset we detected two different bunches. Initially the first bunch was only detected at timesteps $t = [20, 22]$ and the second bunch at $t = [23, 57]$. In the forward tracing we then found that the first bunch also

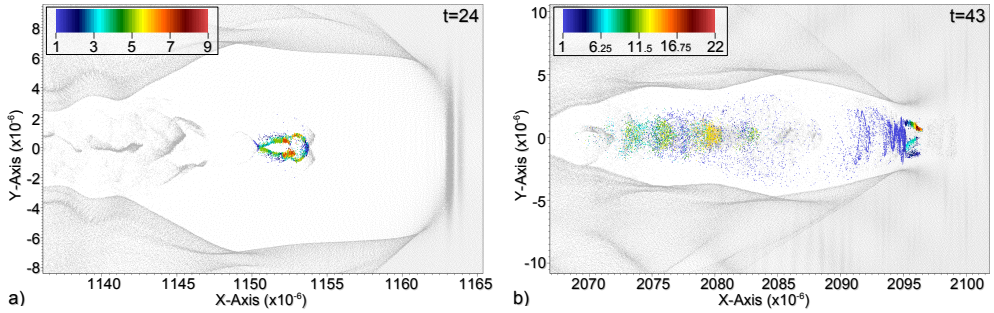

Figure 10. a) Bunch frequency l_1 of the first bunch shown at its peak px momentum at timestep $t = 24$. b) Bunch frequency l_2 of the second bunch shown at its peak px momentum at timestep $t = 43$. The candidate particles of the corresponding bunch are colored according to the associated bunch frequency values indicating how often a particle was found as being part of the bunch. All particles not detected as candidates for the respective bunch are shown in gray in both figures. Note, here only a subset of the complete simulation window containing all candidate particles of the displayed bunch is shown. We can see that in both cases the particles with the highest l_b values are highly localized defining the "core" of each bunch.

exists at timesteps $t = [23, 28]$ and added the acquired information to the initial description of the bunch. The backward tracing did not result in any additional information in this particular case. Note, when performing additional bunch segmentations in the forward and backward tracing we only need to re-execute the actual bunch segmentation (see Section 4.4.2) but not the expensive data preparation step (see Section 4.4.1) of the timestep analysis.

The merging already provides us with a first rough overview of the lifetime of the detected bunches. The different bunch frequency functions l_b shown in Figure 10 then describe a first approximated classification of the different bunches. We can see that in both cases all particles with very high l_b values — $l_1 \geq 7$ (with $max(l_1) = 9$) and $l_2 \geq 19$ (with $max(l_2) = 22$) — are located within a confined region in physical space. These are also the particles we will define later as references for the first and second bunch respectively. In case of the second bunch (see Figure 10b), we also see a larger number of particle with medium bunch frequency values $l_2 \approx 11$ appearing further in the back at a larger distance to the main bunch. This is due to the fact that these particles form a high-density feature in $x/y/px$ space during early timesteps $t \approx 23$. This bunch appears, however, at very low energies and decelerates quickly whereas the second main bunch appears at high energies and exists for a longer period of time. The particles with highest l_2 values are therefore all located within the main bunch of interest and are condensed in physical space.

4.6. Particle Path Analysis

The goal of the particle path analysis is to compute an accurate description of the detected particle bunches based on their complete temporal history. After the merging process the number of particles we need to consider has greatly been reduced to a set of candidate particles for each bunch, i.e., usually less than 1% of the number of particles per timestep. As illustrated in Figure 11 the particle path analysis consists of the following steps: i) particle tracing (see Section 4.6.1), ii) reference path analysis (see Section 4.6.2), and iii) path distance computation (see Section 4.6.3).

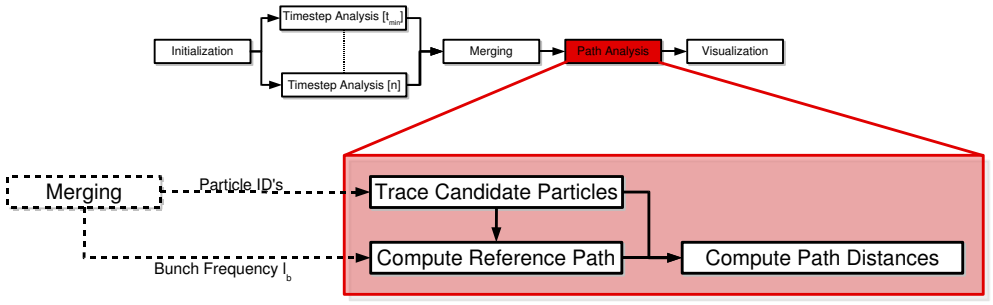

Figure 11. Overview of the particle path analysis: This step is performed independently for each identified bunch. We first trace all candidate particles of a bunch over the complete timeseries. Based on the bunch frequency function l_b we define a set of reference particles for the bunch and derive from their paths a single reference path approximating the temporal evolution of the bunch. Based on the reference path, we first compute the different temporal phases of each bunch, defining, e.g., when the bunch was formed or accelerated. Afterwards, we compute for each candidate particle the distance of its path to the reference path. Using the computed path distance fields we can accurately select the particle bunch.

4.6.1. Particle Tracing

As the first step of the particle path analysis we compute the complete temporal paths of all candidate particles of the current bunch. Based on the IDs of the candidate particles we can access the relevant data at each timestep of the simulation using FastBit by executing a corresponding equality query of the form $ID = id_1||ID = id_2||....||ID = id_n$ at all timesteps of the data set and then load the associated data.

The number of candidate particles per bunch largely depends on the resolution of the raw data, i.e., the number of particles per timestep. In practice, the number of candidate particles is usually on the order of a couple of thousand to a few tens of thousands of particles. The amount of data we need to access in order to define the complete temporal history of all candidate particles is relatively small and can be managed even on a regular desktop computer. The same basic method for tracing particles was also employed in earlier work to enable fast visual exploration of LWFA simulation data [7].

4.6.2. Reference Path Analysis

In order to be able to define the distance of a particle to the current bunch b, we compute a reference path approximating the temporal evolution of the bunch. Therefore, we need to first identify a set of reference particles that characterize the bunch.

As mentioned earlier, we use the bunch frequency function l_b computed during the merging step to identify a group of particles that define the "core" of the bunch. Particles with very high l_b values define a group that were consistently detected as being part of the bunch b. Furthermore, the bunch segmentation is always centered around the highest density feature within the corresponding peak in $x/y/px$ space, i.e., the density core of the bunch. Particles with very high l_b values define a compact group of particles centered at the density core of the bunch (see also Section 4.5 and Figure 10). Due to potential over-segmentation at single timesteps during the initial timestep analysis we may find only a few particles with the maximum l_b value for a bunch. Therefore, we cannot just simply select only those particles with the maximum l_b value, but need to ensure that we select a sufficient number of particles having high l_b values to describe the bunch accurately. We first compute the histogram for l_b defining how many particles were found at each discrete bunch frequency level. We then

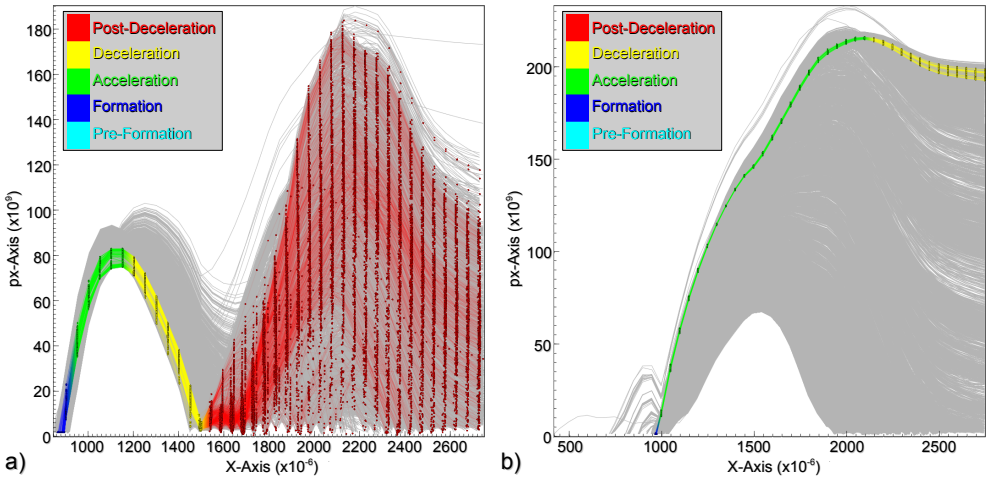

Figure 12. a) Temporal phases of the first bunch. b) Temporal phases of the second bunch. In both figures we show the paths of all candidate particles of the according bunch in gray. The reference particles and their paths are shown in addition colored according to the different beam-phases. The second bunch (b) does not show a post-deceleration phase because the simulation terminated before the bunch had completed deceleration.

detect the local maximum m of the function with the highest l_b value. All particles with $l_b \geq m$ are then chosen as reference for the current bunch b. In case of the example dataset, the reference levels for the two detected bunches are $l_1 = 7$ (with $max(l_1) = 9$) and $l_2 = 19$ (with $max(l_2) = 22$), respectively.

From the temporal paths of the reference particles, we then compute a single reference path representing the temporal behavior of the current bunch b by computing the average position and momentum (x, y, z, px, py, pz) of the reference particles at each timestep t. The reference path, hence, represents the path of the six-dimensional centroid of the bunch over time. Based on the reference path we define the different temporal phases of the bunch as follows:

- **Pre-Formation:** Early time frame during which less than 80% of the reference particles are present, i.e., the bunch is not yet well formed.
- **Formation:** Early time frame during which more than 80% of the reference particles are present but the momentum in x direction is still low, i.e., $px < 10^{10} ms^{-1}$.
- **Acceleration:** Time frame directly after the beam formation phase during which the beam is constantly accelerated until it reaches its peak energy, defined via px. Note, the peak px of a bunch may not be the global maximum in px for the particles of the bunch. As illustrated in Figure 12a, particles may undergo secondary phases of acceleration after the beam of interest has lost its coherency.
- **Deceleration:** Timeframe directly after the beam has reached its peak px momentum and is constantly decelerating.
- **Post-Deceleration:** This phase includes all timesteps after the beam has completed its deceleration phase. The beam has lost its coherency so that its behavior is undefined during this phase. For example, some of the particles may become trapped in secondary periods of the wave and undergo a secondary phase of acceleration (see Figure 12a), while other particles may leave the simulation window and are no longer traceable.

Figure 12 illustrates the different phases of the two bunches detected in the example dataset. The fact that we do not see a pre-formation phase along the reference paths of the two bunches indicates that most reference particles ($> 80\%$) enter the simulation window at the same timestep. When comparing the paths of the reference particles of the two bunches shown in Figure 12a and b, we can see that different bunches may show different acceleration behavior. Information on the different phases of a bunch provides the user with valuable information on which timesteps are of interest for visualization and other types of analysis. We also use this information in the later path distance computation in order to be able to accurately classify the detected particle bunches (see Section 4.6.3).

4.6.3. Path Distance Computation

In the path distance computation, which is the final step of the main analysis, the goal is to define for each candidate particle its distance to the bunch, i.e, the distance of the particle's path to the reference path of the bunch. The path distance provides a measure of how close a particle is to a bunch and is used to determine which particles belong to a bunch.

The path distance function should fulfill the following requirements:

- It should allow the user to effectively define the bunch with respect to pre-knowledge and current analysis requirements.
- The function should be continuous.
- The function should be physically meaningful.
- Function values should have a physically meaningful scale.
- Function values, and in this way classifications of different bunches, should be comparable.

In order to avoid any non-intuitive normalization, achieve comparability, and ensure that function values are at a physically meaningful scale, we compute two independent path distance functions. We essentially have two different main data spaces: i) physical space with the dimensions x, y, z; and ii) momentum space with the dimensions px, py, pz. In these two spaces we can directly define the standard Euclidean distance with no need for normalization. The Euclidean distance in physical space between two particles with index i and j at time t is defined as:

$$d_s(i,j,t) = \sqrt[2]{(x(i,t)-x(j,t))^2 + (y(i,t)-y(j,t))^2 + (z(i,t)-z(j,t))^2} \tag{2}$$

With $x(i,t)$, $y(i,t)$ and $z(i,t)$ being the location of the particle with index i at time t in x, y, and z respectively. The distance in momentum space is defined as:

$$d_m(i,j,t) = \sqrt[2]{(px(i,t)-px(j,t))^2 + (py(i,t)-py(j,t))^2 + (pz(i,t)-pz(j,t))^2} \tag{3}$$

With $px(i,t)$, $py(i,t)$ and $pz(i,t)$ being the momentum of the particle with index i at time t in x, y, and z direction respectively. In the case of a 2D simulation the terms referring to z and pz are ignored.

The distance between the temporal path of two particles with index i and j in physical and momentum space during the timeframe $[t_k, t_l]$ is defined as the average distance along their paths, i.e.:

$$d_s(i,j) = \frac{\sum_{t=t_k}^{t_l} d_s(i,j,t)}{n_t} \tag{4}$$

$$d_m(i,j) = \frac{\sum_{t=t_k}^{t_l} d_m(i,j,t)}{n_t} \tag{5}$$

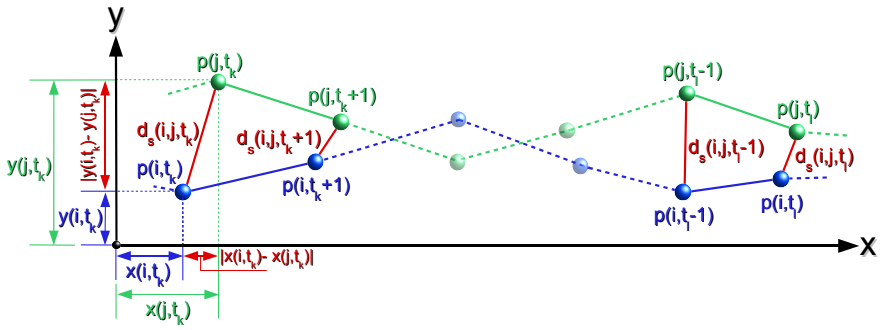

Figure 13. Illustration of the path distance computation for two particles with index i and j in 2D physical space x/y. The distance in momentum space is similarly defined.

With $n_t = t_l - t_k + 1$ (with $t_l > t_k$) being the number of timesteps considered in the computation of the path distance. Figure 13 illustrates the computation of the distance between the path of two particles in 2D.

The distance of a particle to a bunch is defined as the distance of the particle's path to the reference path of the beam. During the pre-formation and formation phase, a beam is not yet well defined. During the deceleration phase, a beam loses its coherency. The beam has fallen apart during the post-deceleration phase. The only timeframe during which a beam is well defined is during the acceleration phase. We, therefore, restrict the path distance computation to the acceleration time frame of the bunch by setting t_k and t_l to the start and end time of the acceleration phase, respectively. We then compute for each candidate particle the respective distances d_s and d_m to the reference path of the current bunch. These distances fulfill all basic requirements mentioned above; they are in a physically meaningful scale (d_s in $meters$, d_m in ms^{-1}), intuitive, physically meaningful, and continuous. Path distances from different bunches are also comparable, i.e., a basic distance of, e.g., $d_s = 10^{-6}m$ or $d_m = 10^{11}ms^{-1}$, have the same basic meaning for different bunches.

The bunch itself is then defined as the set of candidate particles that are within a given distance to the beam. The user, therefore, needs to specify thresholds in d_s and d_m to define the maximum allowed distance in physical and momentum space to the beam. In order to compare different bunches, a user may choose to use the same threshold values for different bunches.

While we here use the average distance along particle paths to define a bunch, other distance functions can easily be defined to describe other beam characteristics. Other distance functions of interest may, e.g., be the minimum distance along paths or also just the distance at the timestep where a bunch has reached its peak momentum in x-direction (px).

Figures 14 and 15 provide an overview of the two bunch classification functions d_s and d_m for the two bunches detected in the example dataset. We can see that in both cases the analysis was able to identify the bunch (blue particles) properly and that d_s and d_m define in each case the same principle bunch. As expected, we also find outliers in physical as well as momentum space, i.e., particles that show a similar acceleration behavior as the bunch but that are distant in physical space as well as particle that stay close to the bunch in physical space but are accelerated differently. By applying appropriate thresholds in d_s an d_m one can accurately define each of the two bunches (see also Figure 21C).

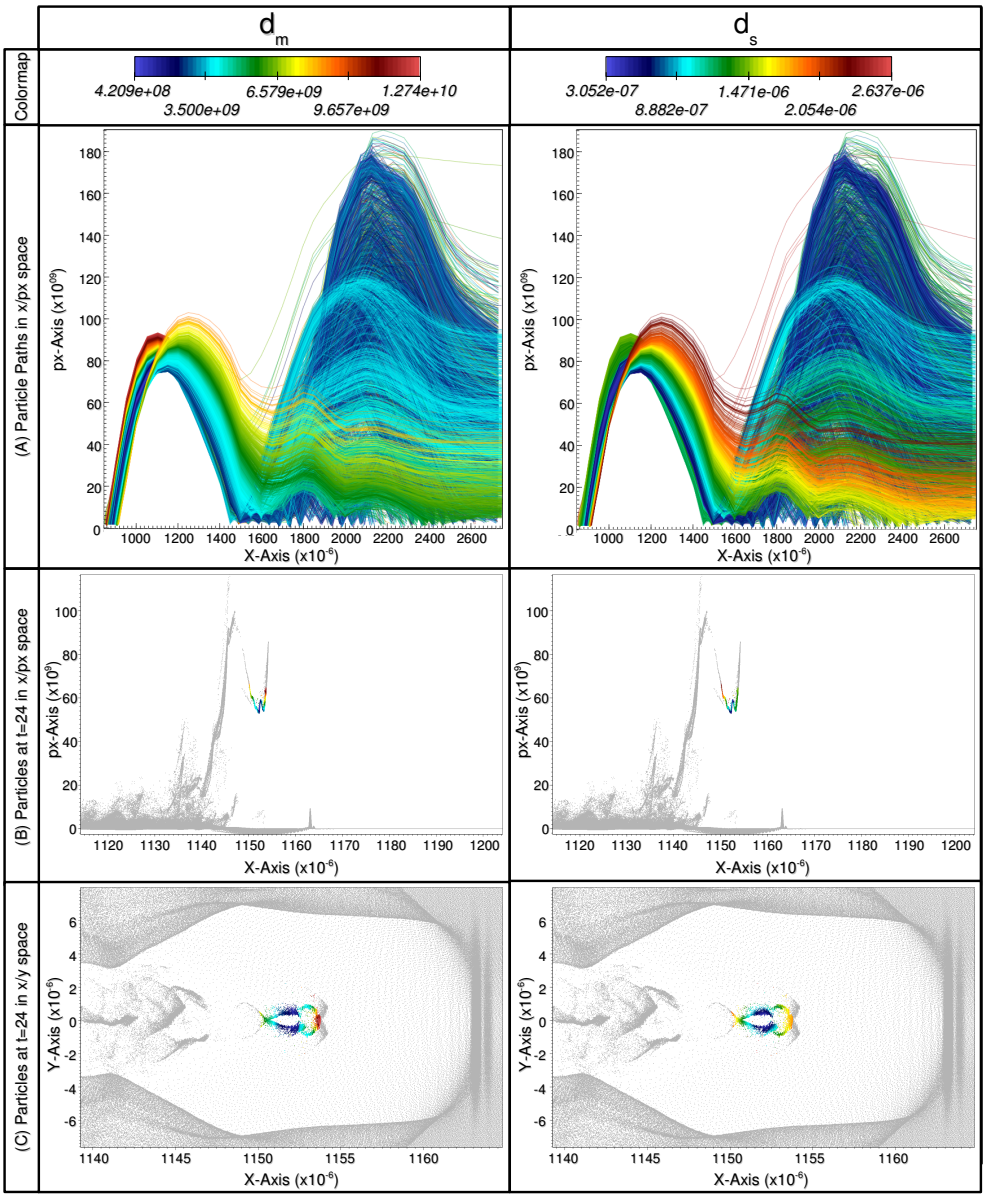

Figure 14. Overview of the path distance fields for the first bunch detected in the example 2D dataset (shown at $t = 24$) illustrating that d_m and d_s accurately classify the bunch. Note, here all candidate particles are shown, i.e., no thresholds in d_s or d_m have been applied in the images. The left column of the table shows the distance in momentum space d_m and the right column the distance in physical space d_s. Color indicates d_m and d_s respectively. We show the temporal paths in x/px space of all candidate particles in row (A). In row (B) and (C) we show the particles in x/px and x/y space at timestep $t = 24$ when the bunch has reached its peak momentum in px. Note, the bottom images show only a subset of the simulation window containing all detected candidate particles of the bunch. We can see that both distance functions define the bunch (blue particles) well. Results after thresholding are presented later in Section 5.2 (see Figure 21(C1)).

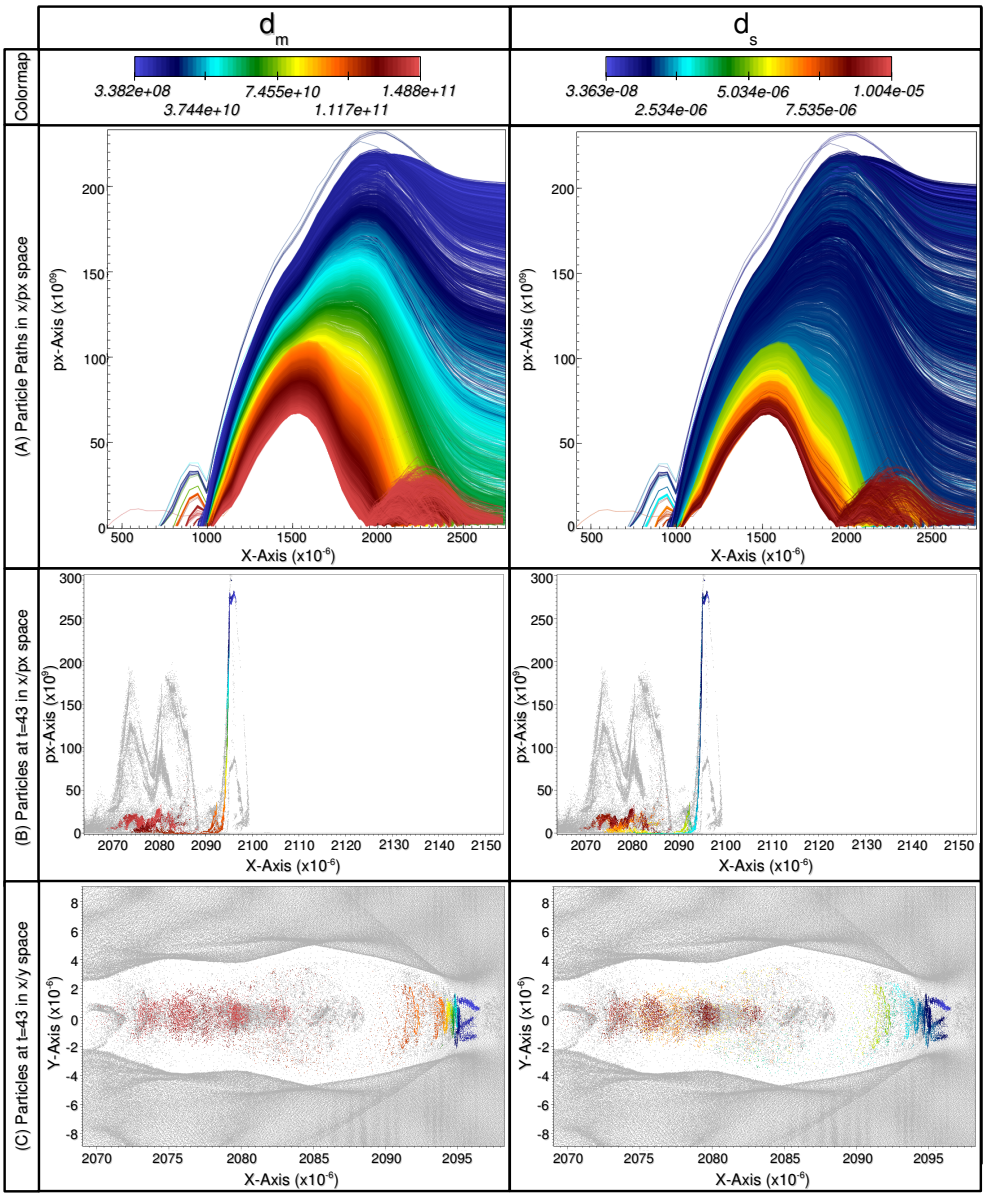

Figure 15. Overview of the path distance fields for the second bunch detected in the example 2D dataset (shown at $t = 43$) illustrating that d_m and d_s accurately classify the bunch. Note, here all candidate particles are shown, i.e., no thresholds in d_s or d_m have been applied in the images. The left column of the table shows the distance in momentum space d_m and the right column the distance in physical space d_s. Color indicates d_m and d_s respectively. We show the temporal paths in x/px space of all candidate particles in row (A). In row (B) and (C) we show the particles in x/px and x/y space at timestep $t = 43$ when the bunch has reached its peak momentum in px. Note, the bottom images show only a subset of the simulation window containing all detected candidate particles of the bunch. We can see that both distance functions define the bunch (blue particles) well. Results after thresholding are presented later in Section 5.2 (see Figure 21(C2)).

4.7. Visualization

In order to assist the user in the investigation of analysis results we make use of state-of-the-art visualization using VisIt. In the analysis process we create for each detected particle bunch a VTK file [56] of all computed particle paths including the derived beam phases and path distances. Using VisIt the user can investigate analysis results using a variety of high-quality visualizations, such as, 2D and 3D particle path visualizations, scatter-plots, or 1D/2D and 3D histograms. In earlier work we described the use of VisIt for visualization of the raw simulation data [7]. By visualizing analysis results in the context of the original data the user can effectively validate and investigate analysis results. All data visualizations shown here are created using VisIt.

VisIt also supports the concept of named selections. Named selections allow one to select a subset of particles based on their IDs. Using this concept it is possible to define persistent selections defining the same subset of particles throughout the time series. Based on the information created in the described beam path analysis a user can define such a named selection, e.g., via thresholding in d_s and d_m. By applying such a named selection to the original raw data a user can perform detailed analysis of the selected particle beam. Named selections are saved to file enabling later reuse.

5. Results and Validation

In Section 5.1 we describe how one can use our analysis to compare different bunches and assess their quality. We here use the same example dataset as in the previous sections. In Section 5.2 we present results of the beam path analysis for a variety of datasets. We use 2D as well as 3D particle datasets having varying spatial and temporal resolution that exhibit different acceleration behavior to demonstrate the effectiveness of our method under varying simulation conditions.

5.1. Beam Comparison and Visualization

In Section 4, we showed that our analysis is able to detect the main bunches of interest in the example dataset. Having detected the particle bunches of interest, one main question is to determine which bunch has the highest quality. In this section, we demonstrate how we can investigate and compare the quality of particle bunches. The quality of a bunch is characterized by several factors. A high quality beam should: i) have low energy spread (indicated by its compactness in px); ii) be compact in physical space; iii) reach high energy levels (indicated by high px); and iv) be focused (indicated by low transverse momentum py and in 3D also pz).

Using the path distance functions d_s and d_m, we can effectively compare the compactness of different bunches from the same simulation simply by comparing their respective histograms of d_s and d_m (see also Figures 14 and 15 for images of the two bunches in physical space and phase space). Figure 16a, shows, e.g., the histograms of the distance in momentum space d_m for the two bunches detected in the example dataset. For the first bunch (red) we find here many more particles with low values of d_m than for the second bunch, i.e., the first bunch will be more compact in momentum space and therefore show a lower energy spread. Similarly, we also find that the first bunch is more condensed in physical space than the second bunch by comparing the histograms of the distance in physical space d_s (see Figure 16b). These findings indicate that the first bunch is highly condensed in physical as well as momentum space and is likely of higher quality than the second bunch.

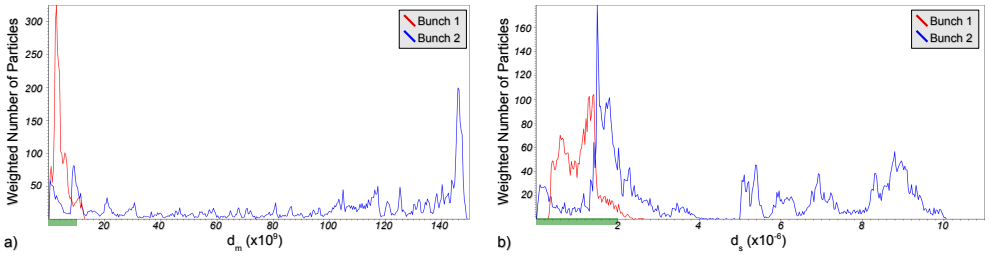

Figure 16. a) Histogram of the distance in momentum space d_m of all candidate particles detected for each of the two bunches. b) Histogram of the distance in physical space d_s of all candidate particles detected for each of the two bunches. The main region of interest below the default thresholds in d_m and d_s respectively is indicated in green in both figures. We find for the first bunch many more particles with low d_m and d_s values indicating that this beam is more compact in both physical and momentum space and therefore of higher quality.

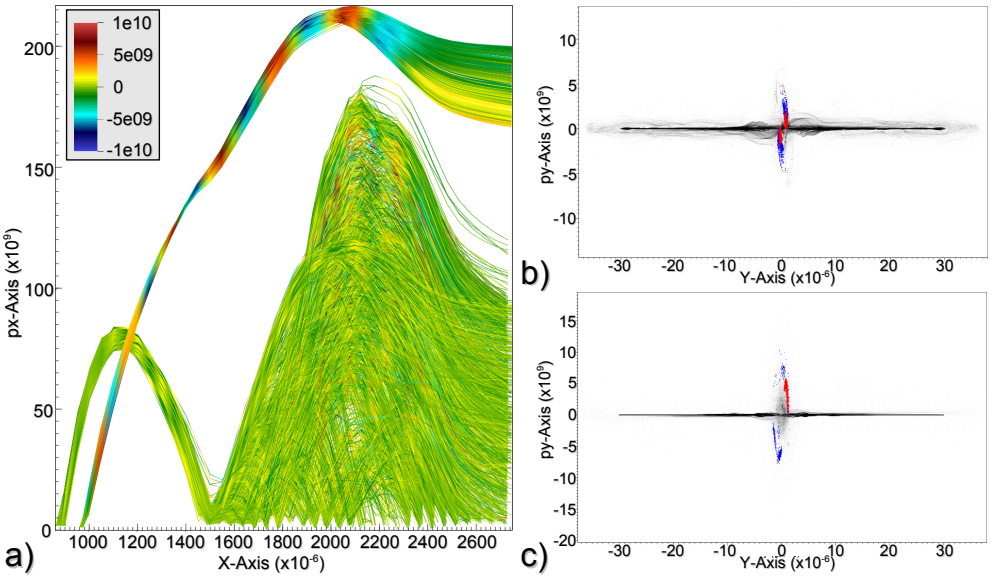

Figure 17. a) Paths of both bunches in x/px space after applying the thresholds $(d_s < 10^{-6})\&\&(d_m < 10^{10})$. Paths are colored according to py using the same color mapping with blue being high negative momentum in y, yellow/green being low py, and red being high positive momentum in y. Although the second bunch achieves overall higher levels of acceleration the first bunch shows much less variation in py while it exists. b) Density plot of y/py space at $t = 24$ (gray) and all particles of the first bunch with $(d_s < 10^{-6})\&\&(d_m < 10^{10})$ shown in red and with $(d_s < 2 * 10^{-6})\&\&(d_m < 10^{10})$ shown in blue. c) Density plot of y/py space at $t = 43$ (gray) with particles of the second bunch shown using the same coloring scheme as in *b*. We selected the timesteps $t = 24$ and $t = 43$ because these are the timesteps at which the according bunch has its peak momentum in px. When comparing figure *b* and *c* we can see that the first bunch exhibits much less dispersion in y/py space than the second bunch.

By comparing the temporal paths of the particles that form the two bunches, we can assess i) which energy level the bunches achieve and ii) how focused they are. To define the two particle beams, we first apply the same thresholds in d_m and d_s for each of the two bunches. Figure 17a shows the temporal paths of the selected particles in x/px space colored

according to the transverse momentum py. The second bunch achieves higher levels of px than the first bunch but also shows much more variation in py. This behavior indicates that while the second bunch reaches much higher energy levels than the first bunch, it also shows more spread in py. The fact that the second bunch is less condensed than the first bunch is even more apparent when comparing their structure in y/py space at the time when they reach their peak energy. Figure 17b shows the respective plot for the first bunch at timestep $t = 24$ and Figure 17c for the second bunch at timestep $t = 43$. When comparing the two plots, we can clearly see that the second bunch shows higher dispersion in y/py space at its peak energy than the first bunch. Overall this analysis indicates that the first bunch: i) has a low energy spread, ii) is compact in physical space, and iii) is highly focused. Even though the first bunch has a lower peak energy, these findings indicate that it is in some measures of higher quality than the second bunch. Our method enables the separation of different bunches and also the analysis of their quality and of processes contributing to the bunch quality.

Investigation of the paths of the beam particles also enables analysis of the temporal evolution of two particle beams. From the beam phases computed in the reference path analysis (see Figure 12), we already know that the first bunch is formed earlier than the second bunch. It then accelerates over a short period of time, then outruns the wave relatively quickly, and then slips into the deceleration phase. On the other hand, the second bunch is formed later and is accelerated over a longer period of time. When comparing the traces of the two beams shown in Figure 17a we also see that the two bunches show a very different acceleration/deceleration behavior. While the paths of the first bunch show a characteristic horseshoe-like shape, the second bunch shows a less smooth acceleration behavior and then decelerates more gradually than the first bunch.

Using 3D particle path visualizations, we can effectively investigate the temporal evolution of particle beams in up to four data dimensions at once. Figure 18 shows the particle paths of the first bunch in $x/y/py$ space colored according to px. Figure 19 shows an example of particle paths based on the relative position of particles within the simulation window for a bunch detected in dataset D (see Table 1). Visualizations of relative instead of absolute particle paths enable investigation of particle motion within the simulation window. With VisIt animations of the complete time series, a user gains a better understanding of the temporal evolution of particle beams.

5.2. Validation

Here we present results of our analysis for a variety of datasets listed in Table 1. Figures 20 to 22 give an overview of the results our method achieves on these different datasets. For each bunch, we show the resulting phase space diagram (x/px) at the timestep where it reached its peak momentum in px. We show in these plots all candidate particles in blue and all particles that satisfy the default selection condition $(d_s < 2 * 10^{-6})\&\&(d_m < 10^{10})$ in red. This condition ensures that we only select particles that are close to the bunch in momentum as well as physical space. Depending on the current requirements, a researcher may in practice choose lower or higher thresholds for d_s and d_m. As the different plots of d_s and d_m illustrate, both functions are smooth and enable the user to accurately define each bunch.

Traditionally thresholding in px is used to identify the high energy particles that form the beam(s) of interest. In order to identify proper thresholds and timestep, a researcher examines movies of a variety of plots, a complex and time consuming process. In many cases a single threshold may not be sufficient to isolate the beam particles of interest, e.g., in dataset C two high energy bunches exist at the same time. The main deficiencies of manual thresholding are that it is arbitrary, time consuming, and requires manual inspection of the dataset.

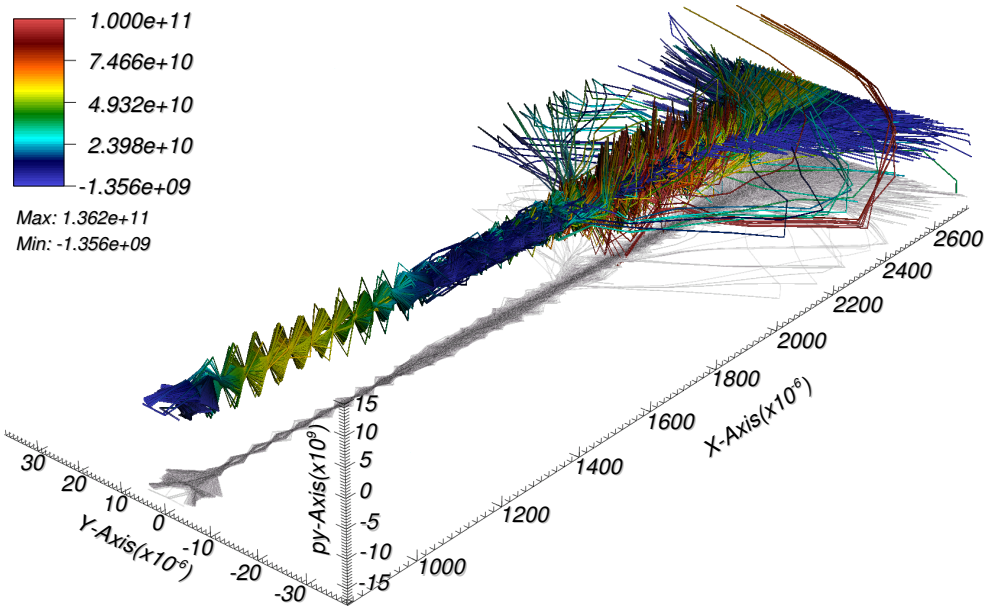

Figure 18. Paths of the first bunch in $x/y/py$ space after applying the thresholds $(d_s < 2 * 10^{-6})\&\&(d_m < 10^{10})$. Paths are colored according to px using the indicated color mapping. The bunch reaches its peak energy at $x \approx 1200 * 10^{-6}m$, afterwards the bunch decelerates. At $x > 1500 * 10^{-6}m$ the bunch has lost its coherency, while some particles leave the simulation window and are no longer traceable, others undergo a secondary phase of acceleration. While it exists the beam particles show a very regular motion in y/py. Particle path visualizations in 3D enable investigation of the evolution of particle beams in up to four dimensions.

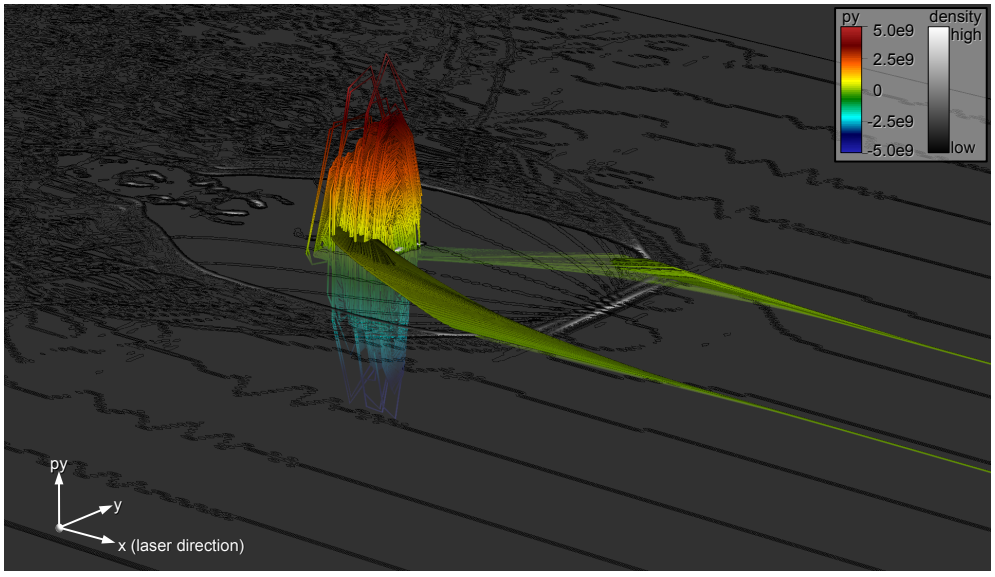

Figure 19. Relative particle paths of the first bunch detected in a high-resolution 2D dataset (see also Figure 21(D1)) shown in $x_{relative}/y/py$ space colored according to py. Instead of the absolute x location we here use the relative x location of particles within the simulation window to illustrate the motion of particles within the wake of the laser. Traces are shown up to timestep $t = 95$ when the bunch reaches its peak momentum in px. The x/y plane shows iso-contours of the particle density at timestep $t = 95$. We can see particles being injected from the sides and oscillating while being accelerated.

Dataset	Type	Size per timestep in MB≈	Total size in MB≈	Number of timesteps	Particles per timestep≈
A	2D	35	1,320	38	405,000
B	2D	128	4,606	36	1,600,000
C	2D	190	11,034	58	2,400,000
D	2D	62	13,990	226	610,000
E	3D	23,999	623,964	26	229,850,000
F	3D	7,091	212,733	30	90,790,000

Table 1. Description of the used simulation datasets.

As summarized in Table 1, the datasets we use are of varying temporal and spatial resolution, ranging from 26 to 226 saved timesteps and $\approx 405,000$ to $\approx 229,850,000$ particles per timestep. Dataset F was produced using an older version of the simulation code and does not contain unique particle IDs. In this work, we use dataset F only to evaluate the performance of the computation of 3D histograms and 3D bin-queries, so particle IDs are irrelevant.

As Figure 20(A) shows, our method reliably detects particle bunches even in datasets with low spatial resolution. Dataset B and C are two 2D datasets of medium spatial resolution. Dataset D is also a 2D dataset but with high temporal resolution. The results for datasets B-D are shown in Figure 20(B) and 21(C,D) respectively. Dataset E is a massive 3D dataset having $\approx 229,850,000$ particles per timestep. Due to the large amount of data that needs to be stored just for a single timestep, only few timesteps are written to file. The sheer size of this dataset, as well as its low temporal resolution, makes detecting particle bunches a challenging task. As shown in Figure 22, even in this most challenging case our method is able to reliably detect the two main bunches of interest.

6. Performance Evaluation

In this section we analyze the performance of our analysis approach. In Section 6.1, we analyze the performance of the 3D histogram computation used in the timestep analysis (see Section 4.4). Afterwards we characterize the performance of 3D bin queries in Section 6.2, i.e., how much time do we need to identify the particles located within a set of selected bins of a 3D histogram. Both, the 3D histogram computation and the 3D bin queries, are crucial parts of the initial timestep analysis (see Section 4.4) aimed at identifying particle bunches of interest at a particular timestep. For these performance tests, we are using a representative timestep of the medium sized 2D dataset (C) and a 3D dataset (F), described earlier in Table 1, containing a condensed particle bunch of interest. In Section 6.3, we then analyze the serial performance of the complete analysis algorithm using the datasets described in Table 1. We describe the overall performance of our approach as well as characterize the performance of the different parts of the pipeline. A detailed analysis of the performance of the particle tracing using FastBit is available in earlier work [7].

For all performance tests described in Sections 6.1 - 6.2, we used a workstation equipped with two $2GHz$ dual core AMD Operton^{TM}270 processor, $8GB$ running SuSE Linux. In these tests, only one of the cores is actually used by the application.

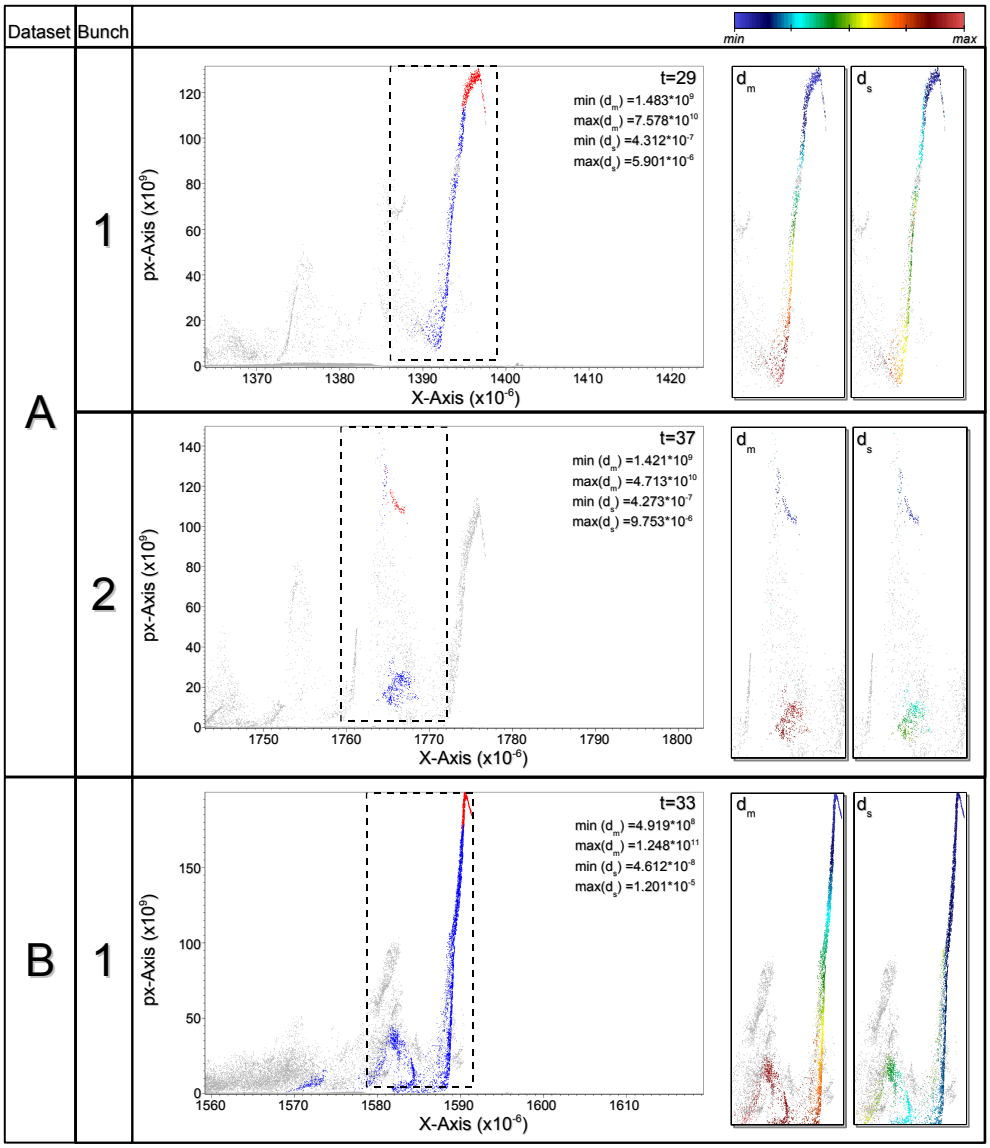

Figure 20. Overview of the analysis results for dataset A and B illustrating that our method is able to accurately detect the relevant bunches. In the main figure, all detected candidate particles are shown in blue and all particles that satisfy the condition $(d_s < 2 * 10^{-6})\&\&(d_m < 10^{10})$ are shown in red. This condition ensures that we select only particles close to the bunch. The cutout plots show the distance functions d_m and d_s, respectively. Each bunch is shown at the timestep when it has reached its peak px level.

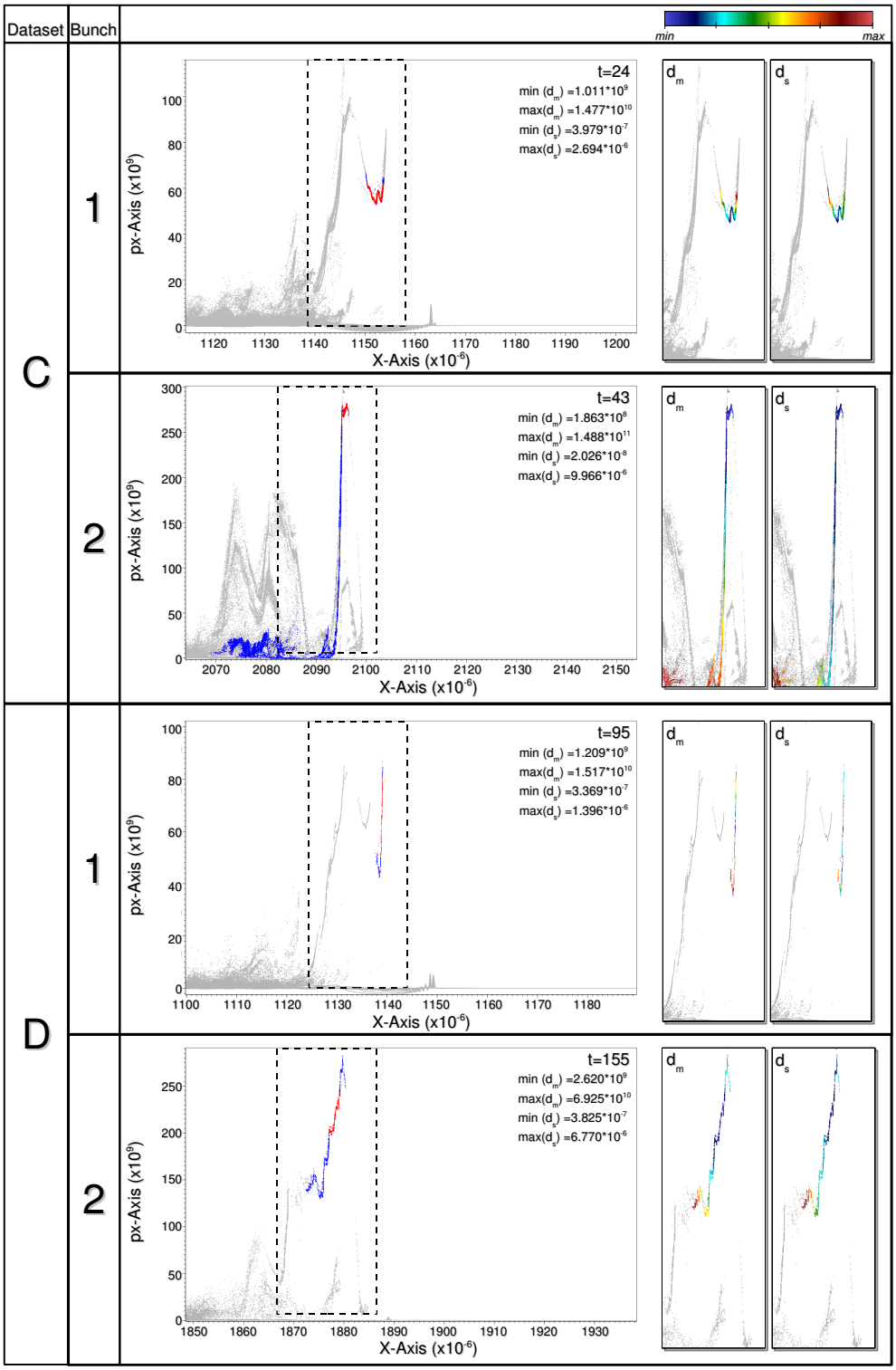

Figure 21. Overview of the analysis results for dataset C and D illustrating that our method is able to accurately detect the relevant bunches (see also Figure 20).

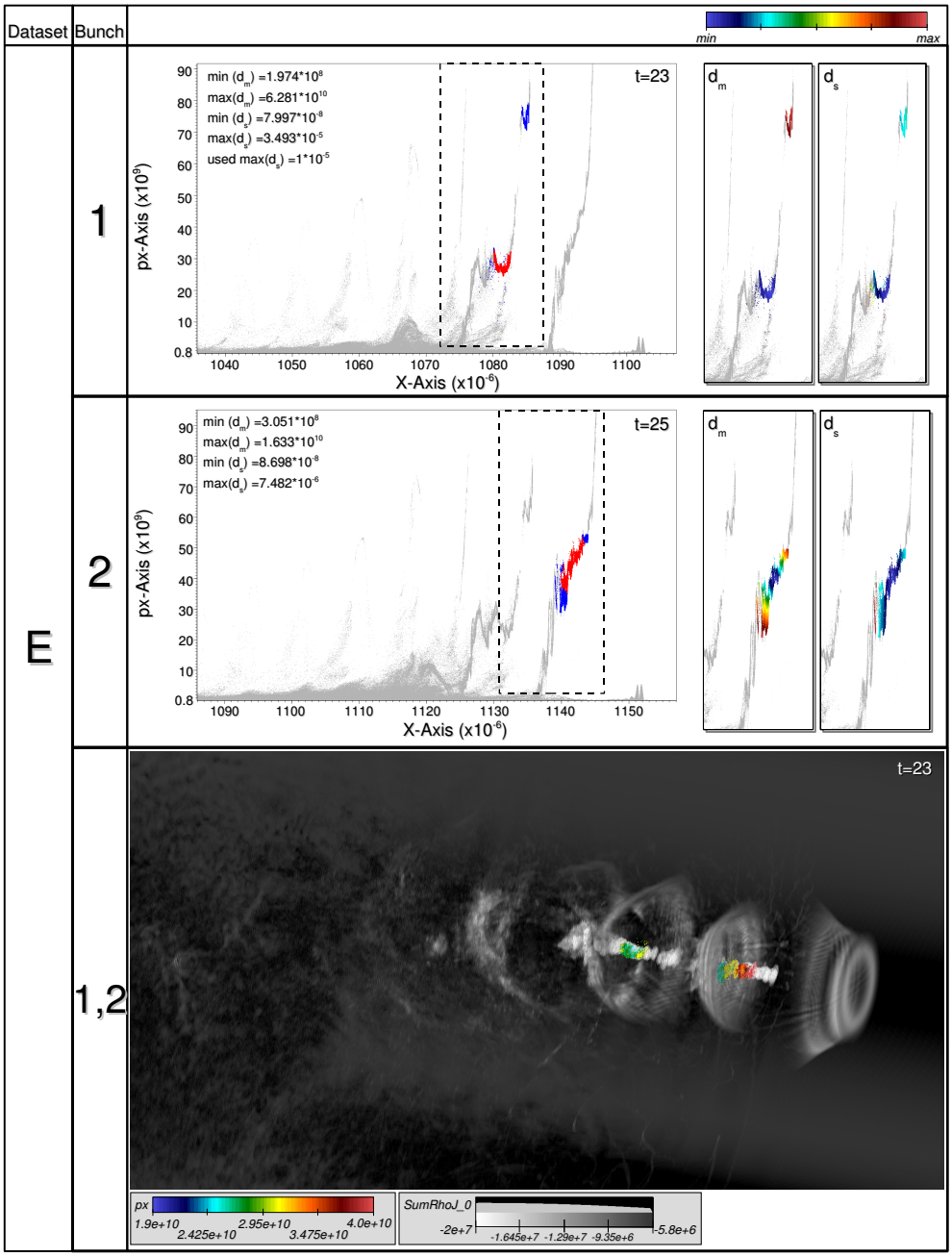

Figure 22. Overview of the analysis results for dataset E illustrating that our method is able to accurately detect the relevant bunches (see also Figure 20). The bottom image shows a volume rendering of the plasma density (gray) and the two selected bunches colored according to px at timestep $t = 23$ illustrating the location of the two bunches within the plasma wave.

6.1. Computing 3D Histograms

This performance test is aimed at characterizing the performance of the computation of 3D conditional histograms. The goal is to characterize the speed-up we achieve through the use of FastBit for computing the histograms as well as the overhead for computing the bit vectors in addition to the bin counts. We compare the performance of the following different implementations:

- **H1: Bitvectors:** This is the implementation used in the analysis code. We use FastBit to evaluate the condition and to compute for each non-empty bin a bit vector indicating which particles belong to that bin. From these bit vectors, we derive the counts of the 3D histogram.
- **H2: FastBit:** This implementation uses FastBit to first evaluate the condition of the histogram. Afterwards, the counts of the histogram are computed based on the selected particles only.
- **H3: Sequential:** In this variant, we do not make use of FastBit. We perform a sequential scan through all particles and check each against the given condition and update the counts of the 3D histogram accordingly.

For each implementation, we compute a series of 3D histograms in x, y, and px while varying the number of bins per variable. As in the timestep analysis, we use the condition $(px > 10^{10})ms^{-1}$. All histograms are computed over the complete data range of the respective variables. We repeated each measurement twenty times and report the average of all runs.

Figure 23 shows the performance of the different implementations for the example timesteps of dataset C and F. For the smaller 2D dataset, H2 and H3 show similar performance. For the larger 3D dataset, H2 then shows a much better performance than H3. This result is expected since in the case of H3, the cost for traversing the additional particles that do not satisfy the condition will increase significantly with increasing file size. For less than ≈ 150 bins per variable our approach is in general slower than H2 by a roughly constant factor of ≈ 1.3 to ≈ 2. For larger numbers of bins, the performance of H1 decreases faster than for H2. This behavior is due to the fact that in addition to the counts of the histogram, we also have to maintain a secondary data structure (the bit vectors) that requires random access to

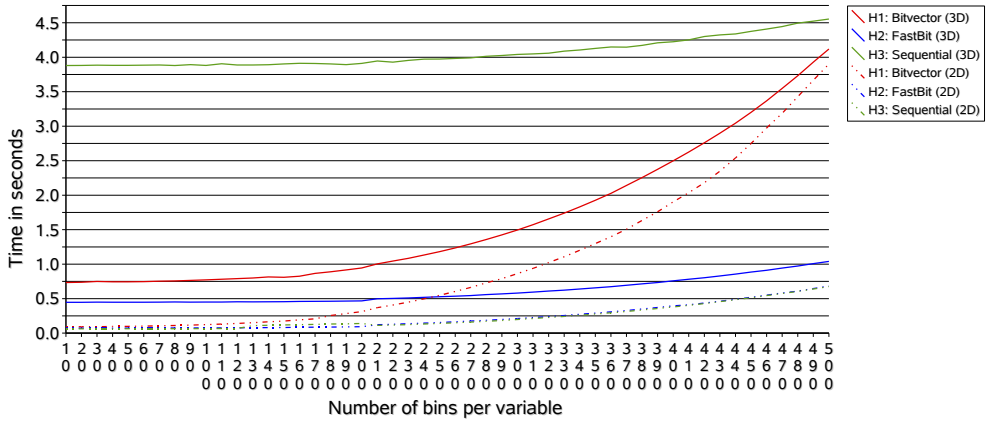

Figure 23. Timings for serial computation of conditional 3D histograms.

memory. With increasing number of bins also the number of non-empty bins and, hence, the number of bit vectors H1 has to maintain, increases. Updating this secondary data structure becomes increasingly expensive the more bit vectors need to be maintained. In the analysis we typically use only ≈ 100 bins per variable where the cost of maintaining the bit vectors is modest. As we show in the next section, the use of bit vectors significantly improves the performance of 3D bin queries and the additional time needed to compute and maintain the bit vectors is well spent.

6.2. Evaluating 3D Bin Queries

As part of the bunch segmentation process, we need to identify which particles are located within selected bins of a 3D histogram. As described earlier, we use a set of bit vectors —computed during the 3D histogram computation— to access the data of these particles efficiently. In case that this inverse mapping from the 3D histogram back to the original data space is not available, one would have to evaluate a corresponding query in order to decide which particles are associated with the selected bins. For each bin, such a query takes the form of:

$$[(x_i \geq x)\&\&(x < x_{i+1})\&\&(y \geq y_i)\&\&(y < y_{i+1})\&\&(px \geq px_i)\&\&(px < px_{i+1})] \quad (6)$$

The parameters x_i, y_i, px_i and x_{i+1}, y_{i+1}, px_{i+1} indicate the lower and upper boundaries of the bin with index i respectively. In a typical segmentation, not one, but several bins are selected so that many of these queries need to be combined via OR ($||$). For example, if 10 bins are selected then one has to evaluate a query consisting of 60 conditions combined via AND ($\&\&$) and OR ($||$).

In this test, we simulate the segmentation process by selecting the n most populated bins. We then increase n to analyze the performance with increasing size of the selection and query complexity. In this test we compare the following different implementations:

- **Q1: Bitvectors:** This is the version used in our analysis code. We first merge the bit vectors of the selected bins and then use FastBit to load the IDs of the selected particles.
- **Q2: FastBit:** We use FastBit to evaluate the segmentation query and then load the IDs of the selected particles only.
- **Q3: Sequential:** This implementation performs a sequential scan through the data to evaluate the query. We first load the data of all particles in x, y, and px and then check for each particle whether it satisfies the query. Since the different bin-queries are combined with OR, we can stop this process for each particle as soon as the particle has been identified as being part of a selected bin. Furthermore, the selected bins are sorted in decreasing order with respect to their counts. If N is the total number of particles, M the number of selected particles, and B the number of selected bins then one needs to perform in the worst case $(N - \frac{M}{2}) * (B * 6)$ compare operations to evaluate the query. Afterwards we again load the IDs of the selected particles.

Figure 24 shows the performance of the different implementations for the example timesteps of dataset C and F. We repeated each measurement ten times and report the average timings. In the 2D case, Q2 and Q3 show a similar performance. For the larger 3D data file the sequential scan (Q3) performs better than the implementation using FastBit (Q2). To answer 3D bin queries of the form shown in Eq. (6.2), FastBit invokes the indices on x, y and px separately and combines the results from the corresponding sub-queries to compute the final query result. Currently, FastBit is not able to share intermediate results computed for different bins and therefore performs a considerable amount of redundant work. Compared to both Q2

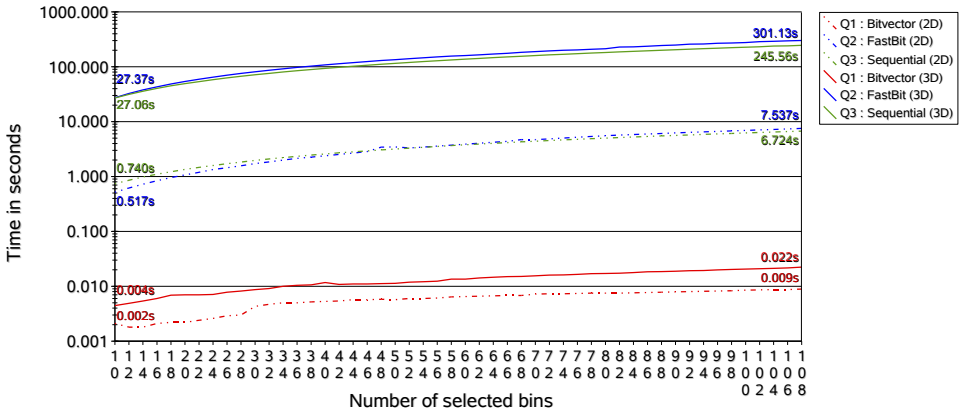

Figure 24. Timings for serial computation of 3D bin queries. Note the logarithmic scale on the y-axis showing the time in seconds. We can see that the data access using the bit vectors (Q1) is significantly faster than when having to evaluate a 3D bin query (Q2 and Q3).

and Q3, our implementation (Q1) shows an outstanding performance in all cases. Even in the case of the relatively small 2D dataset when selecting just 10 bins, we see a speed-up of ≈ 258 compared to Q2 and a speed-up of ≈ 370 compared to Q3. When comparing Q1 to Q2 or Q3 in the case of the larger 3D dataset, we see even higher speed-ups of ≈ 4500 when selecting only 10 bins and $> 11,000$ when selecting more than 100 bins. The additional time needed to compute the bit vectors is justified by the large speed-up we gain during the evaluation of segmentation results (and therefore also the merging).

6.3. Serial Performance of the Analysis Algorithm

For the serial performance tests of the algorithm we used a system equipped with eight $2GHz$ dual core AMD Opteron^{TM}Processor 870 with $8GB$ of memory per core running Ubuntu Linux. For the serial analysis, we use only one core and the memory limit was set to $8GB$ while the peak memory usage of the analysis did not exceed $2.5GB$ in any case. The peak memory usage is reached in the case of dataset D due to its many time steps and hence the large amount of data and information that is acquired in the initial timestep analysis.

Figure 25 shows the timings for our analysis of the datasets described in Table 1. We repeated each measurement ten times. Due to the complexity of the algorithm the variation of the timings is $\approx 1\%$ for larger and up to 5% for smaller datasets. To account for these variations we report the timing of the run with the median total analysis time. We can see that even when run in serial, our approach shows very good performance even for very large datasets. In order to give an overview of the complete performance of our method, we executed the minimum timepoint computation for all datasets. As mentioned earlier, in practice a user often has already a good understanding of when the first particle beams are forming so that the minimum timepoint computation is often omitted.

The initialization step is where we first touch all the necessary files and access meta-data. The initialization time largely depends on the time needed for the first disk access. Overall, the 3D histogram computation and the particle tracing are the most expensive steps of the analysis pipeline. This result is expected since these are the two main steps where we need to access larger portions of the raw data. In the particle tracing, we then need to perform a series of equality queries to extract the data of a group of particles from each timestep based

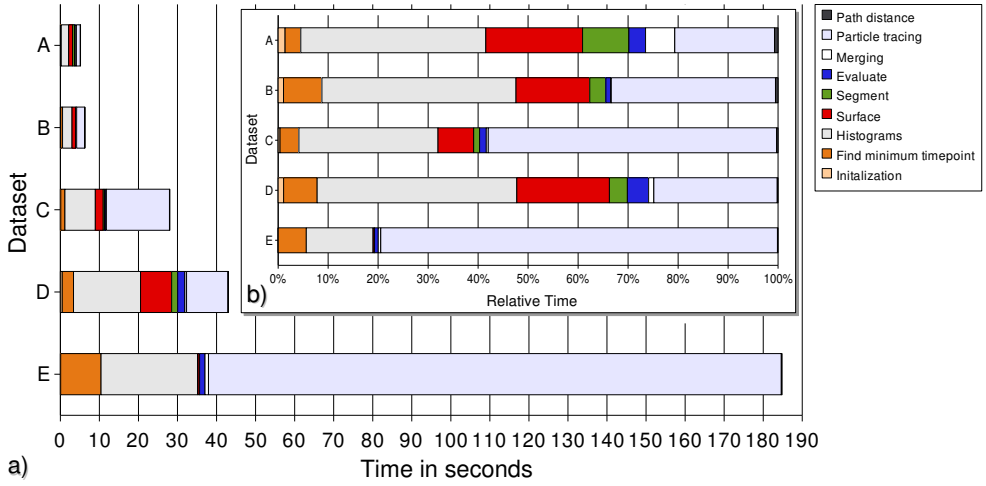

Figure 25. a) Absolute timings for the serial analysis of five different datasets using our method. The length of each horizontal bar represents the total time used for the complete analysis of the corresponding dataset. b) Relative timings for the different steps of the analysis pipeline as percentage of the total analysis time. In both figures color is used to indicate the timings of different parts of the algorithm. In all cases the histogram computation and the particle tracing are the most expensive analysis steps. This is expected since these are the two main steps during which the raw data needs to be accessed. We can see that the algorithm shows good performance in all cases and scales well with increasing dataset size.

on their IDs and merge the data to define the temporal particle paths. The particle tracing is particularly expensive in the 3D case and consumes the majority of the time. The performance of the particle tracing using FastBit is described in earlier work [7]. In a realistic use case scenario, we have seen speed-ups of two orders of magnitude using FastBit compared to a sequential scan method. Besides the 3D histogram computation and the particle tracing, the minimum timepoint computation also requires a considerable amount of time due to the fact that we need to perform a hit count of how many particles satisfy the query $(px > 10^{10})$ at all timesteps. Using FastBit, we are able to perform this hit count efficiently using only the bitmap index for px (without loading any raw data).

The actual analysis steps, i.e., segmentation, evaluation, merging, and path distance computation, are then very fast. This behavior is mainly due to the overall structure of the pipeline allowing the independent analysis of each timestep. The performance of the bunch segmentation (and surface computation) is independent of the size of the dataset and depends only on the resolution (i.e., number of bins per variable) of the underlying grid-based data structure and on the number of timesteps. As we have shown in the previous section, the outstanding performance of the evaluation of the segmentation — in which we need to identify the particles located within a set of histogram bins — is due to the bit vectors which allow us to directly access the required data. The time spent for merging is also very short ($< 0.9s$) in all cases. The performance of the merging step largely depends on the number of bunch segmentations executed in the forward- and backward tracing. However, the expensive data preparation — consisting of the 3D histogram and surface computation — is at this point already completed for all timesteps so that we here only need to execute the much faster segmentation and evaluation step, explaining the short execution times of the merging in all cases.

7. Conclusions and Future Work

Knowledge of the properties of particle beams and their temporal evolution is essential for the understanding and development of laser wakefield particle accelerators (LWFAs). We have presented novel computational methods for gaining insight into physical phenomena extracted from within large datasets produced by LWFA simulations.

We have presented a novel, efficient analysis pipeline for automatic detection and temporal classification of particle beams in LWFA simulation data. With this analysis we enable for the first time automatic classification of particle bunches based on the complete temporal history of the particles that form them. Our analysis provides the user with information about the different temporal phases of a bunch, which particles belong to a bunch, and their distance from the path of the bunch center. We have shown how state-of-the-art visualization using VisIt enables effective investigation of analysis results.

We applied our method to a variety of particle datasets in 2D and 3D space of varying spatial and temporal resolution showing different acceleration behavior demonstrating the effectiveness of our method under varying simulation conditions. We furthermore illustrated how a user can effectively explore and compare the quality of different particle bunches based on the results of the proposed analysis.

We examined and reported the runtime performance of our proposed analysis pipeline on a variety of datasets. For the example 2D datasets our implementation was able to complete the analysis in less than 45 seconds in all cases. Even in the case of the large 3D dataset — which was produced by a hundred-thousand-processor-hour class simulation— the analysis took only ≈ 185 seconds. Within the scope of the performance evaluation we also described the complexity of the different steps of the pipeline in detail.

As part of our analysis we also introduced dedicated methods for computing 3D histograms using compressed bit vectors to efficiently access the data associated with a set of bins. We integrated these methods directly in the FastBit library. We studied the runtime performance of these new functions under practical conditions and showed outstanding data-access performance.

With the increasing computational power of supercomputers and improved scalability of simulation codes, simulation datasets are expected to continue to increase in size and complexity. To address this challenge we are planning to investigate different avenues for parallelization of the analysis to further improve the performance of the proposed methods. Being able to objectively measure and define the quality of particle beams is essential for high-throughput analysis of large simulation datasets. Development and combination of methods for automated beam quality estimation with the described beam path analysis is expected to further improve versatility and value of automated analysis methods.

Acknowledgments

This work was supported by the Director, Office of Science, Offices of High Energy Physics and Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET) and the COMPASS project. This research used resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. We also thank the VORPAL team, in alphabetical order T. Austin, G. I. Bell, D. L. Bruhwiler, R. S. Busby, J. Carlsson, J. R. Cary, B. M. Cowan, D. A. Dimitrov, A. Hakim,

J. Loverich, P. Messmer, P. J. Mullowney, C. Nieter, K. Paul, S. W. Sides, N. D. Sizemore, D. N. Smithe, P. H. Stoltz, S. A. Veitzer, D. J. Wade-Stein, G. R. Werner, M. Wrobel, N. Xiang, W. Ye, for ongoing efforts to develop and maintain this parallel C++ framework on a variety of supercomputing platforms including those at NERSC.

References

[1] T. Tajima and J. M. Dawson, "Laser electron accelerator," *Phys. Rev. Lett.*, vol. 43, no. 4, pp. 267–270, Jul 1979.

[2] LOASIS, http://loasis.lbl.gov/.

[3] C. Geddes, C. Toth, J. van Tilborg, E. Esarey, C. Schroeder, D. Bruhwiler, C. Nieter, J. Cary, and W. Leemans, "High-Quality Electron Beams from a Laser Wakefield Accelerator Using Plasma-Channel Guiding," *Nature*, vol. 438, pp. 538–541, 2004, LBNL-55732.

[4] W. P. Leemans, B. Nagler, A. J. Gonsalves, C. Toth, K. Nakamura, C. G. R. Geddes, E. Esarey, C. B. Schroeder, and S. M. Hooker, "GeV electron beams from a centimetre-scale accelerator," *Nature Physics*, vol. 2, pp. 696 – 699, 2006.

[5] C. G. R. Geddes, "Plasma Channel Guided Laser Wakefield Accelerator," Ph.D. dissertation, University of California, Berkeley, 2005.

[6] F. S. Tsung, R. Narang, W. B. Mori, C. Joshi, R. A. Fonseca, and L. O. Silva, "Near-GeV-Energy Laser-Wakefield Acceleration of Self-Injected Electrons in a Centimeter-Scale Plasma Channel," *Physical Review Letters (PRL)*, vol. 93, 185002, October 2004.

[7] O. Rübel, Prabhat, K. Wu, H. Childs, J. Meredith, C. G. R. Geddes, E. Cormier-Michel, S. Ahern, G. H. weber, P. Messmer, H. Hagen, B. Hamann, and E. W. Bethel, "High Performance Multivariate Visual Data Exploration for Extemely Large Data," in *SuperComputing 2008 (SC08)*, Austin, Texas, USA, November 2008, LBNL-716E.

[8] D. Ushizima, O. Rübel, Prabhat, G. Weber, E. W. Bethel, C. Aragon, C. Geddes, E. Cormier-Michel, B. Hamann, P. Messmer, and H. Hagen, "Automated Analysis for Detecting Beams in Laser Wakefield Simulations," in *Proceedings of The Seventh International Conference on Machine Learning and Applications 2008 (ICMLA 08)*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2008, p. 382387, LBNL-960E.

[9] FastBit is available from https://codeforge.lbl.gov/projects/fastbit/.

[10] VisIt is available from https://wci.llnl.gov/codes/visit/.

[11] Root is available from http://root.cern.ch/drupal/.

[12] AIDA is available from http://aida.freehep.org/.

[13] R is available from http://www.r-project.org/.

[14] IDL is available from http://www.ittvis.com/ProductServices/IDL.aspx.

[15] OpenDX is available from http://www.opendx.org/.

[16] VorpalView is availble from http://www.txcorp.com/products/VORPAL/vorpalview.php.

[17] ParaView is available from http://www.paraview.org/.

[18] R. A. Fonseca, S. F. Martins, L. O. Silva, J. W. Tonge, F. S. Tsung, and W. B. Mori, "One-to-one Direct Modeling of Experiments and Astrophysical Scenarios: Pushing the Envelope on Kinetic Plasma Simulations," *Plasma Physics and Controlled Fusion*, vol. 50, 124034, November 2008.

[19] R. A. Fonseca, L. O. Silva, F. S. Tsung, V. K. Decyk, W. Lu, C. Ren, W. B. Mori, S. Deng, S. Lee, T. Katsouleas, and J. C. Adam, "OSIRIS: A Three-Dimensional, Fully Relativistic Particle in Cell Code for Modeling Plasma Based Accelerators," *Lecture Notes in Computer Science, Computational Science ICCS 2002*, vol. 2331/2002, pp. 342–351, 2002.

[20] S. F. Martins, R. A. Fonseca, L. O. Silva, and W. B. Mori, "On Dynamics and Acceleration in Relativistic Shocks," *Astrophysical Journal Letters (ApJ)*, vol. 695, L189-L193, April 2009.

[21] A. Bagherjeiran and C. Kamath, "Graph-based Methods for Orbit Classification," in *SDM*, 2006.

[22] N. S. Love and C. Kamath, "Image Analysis for the Identification of Coherent Structures in Plasma," in *Applications of Digital Image Processing. Edited by Tescher, Andrew G.. Proceedings of the SPIE*, vol. 6696, 2007.

[23] J. Hlína, V. Nwnicka, and J. Sonsky, "Identification of dynamic patterns and their velocities in thermal plasma jets," *Czechoslovak Journal of Physics*, vol. 54, no. 2, pp. 199–210, Feburary 2004, iSSN0011-4626 (Print) 1572-9486 (Online).

[24] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2007, vol. 3, ISBN 013168728X, 9780131687288.

[25] S. W. Zucker, "Region growing: Childhood and Adolescence," *Computer Graphics and Image Processing*, vol. 5, no. 3, pp. 382 – 399, 1976.

[26] A. Strehl and J. Ghosh, "Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions," *Journal on Machine Learning Research (JMLR)*, vol. 3, pp. 583–617, December 2002.

[27] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On Combining Classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, March 1998.

[28] A. Fred and A. Jain., "Combining Multiple Clustering Using Evidence Accumulation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, 2005, pp. 835–850.

[29] T. G. Dietterich, "Ensemble methods in machine learning," in *Lecture Notes in Computer Science*, vol. 1857/2000. Springer-Verlag, 2000, pp. 1–15.

[30] D. Comer, "The Ubiquitous B-Tree," *Computing Surveys*, vol. 11, no. 2, pp. 121–137, 1979.

[31] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology," *ACM SIGMOD Record*, vol. 26, no. 1, pp. 65–74, March 1997.

[32] P. O'Neil, "Model 204 Architecture and performance," in *2nd International Workshop in High Performance Transaction Systems, Asilomar, CA*, ser. Lecture Notes in Computer Science, vol. 359. Springer-Verlag, Sep. 1987, pp. 40–59.

[33] K. Stockinger, K. Wu, and A. Shoshani, "Strategies for processing ad hoc queries on large data warehouses," in *DOLAP'02*, McLean, Virginia, USA, 2002, pp. 72–79.

[34] K. Wu, K. Stockinger, and A. Shosani, "Breaking the Curse of Cardinality on Bitmap Indexes," in *SSDBM 2008*, 2008, pp. 348–365.

[35] C.-Y. Chan and Y. E. Ioannidis, "Bitmap Index Design and Evaluation," in *SIGMOD*, 1998, pp. 355–366.

[36] K. Wu, E. Otoo, and A. Shoshani, "On the Performance of Bitmap Indices for High Cardinality Attributes," in *VLDB*, 2004, pp. 24–35.

[37] ——, "Compressing Bitmap Indexes for Faster Search Operations," in *SSDBM'02*, Edinburgh, Scotland, 2002, pp. 99–108.

[38] ——, "Optimizing Bitmap Indices with Efficient Compression," *ACM Transactions on Database Systems*, vol. 31, pp. 1–38, 2006.

[39] E. W. Bethel, S. Campbell, E. Dart, K. Stockinger, and K. Wu, "Accelerating Network Traffic Analysis Using Query-Driven Visualization," in *Proceedings of 2006 IEEE Symposium on Visual Analytics Science and Technology*. IEEE Computer Society Press, October 2006, pp. 115–122.

[40] K. Wu, W. Koegler, J. Chen, and A. Shoshani, "Using Bitmap Index for Interactive Exploration of Large Datasets," in *SSDBM'2003*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 65–74.

[41] K. Stockinger, E. W. Bethel, S. Campbell, E. Dart, and K. Wu, "Detecting Distributed Scans Using High-Performance Query-Driven Visualization," in *SC '06: Proceedings of the 2006 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*. New York, NY, USA: IEEE Computer Society Press, October 2006, LBNL-60053.

[42] H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max, "A Contract-Based System for Large Data Visualization," in *Proceedings of IEEE Visualization 2005*, October 2005, pp. 190–198.

[43] EnSight Gold is available from http://www.ensight.com/ensight-gold.html.

[44] K. Stockinger, J. Shalf, K. Wu, and E. W. Bethel, "Query-Driven Visualization of Large Data Sets," in *Proceedings of IEEE Visualization 2005*. IEEE Computer Society Press, October 2005, pp. 167–174, LBNL-57511.

[45] C. D. Hansen and C. R. Johnson, *The Visualization Handbook*. Elsevier Academic Press, 2005.

[46] C. K. Birdsall and A. Langdon, *Plasma Physics via Computer Simulation*, 1st ed., ser. Series in Plasma Physics. Taylor & Francis, Inc., October 2004.

[47] C. G. R. Geddes, D. L. Bruhwiler, J. R. Cary, W. B. Mori, J.-L. Vay, S. F. Martins, T. Katsouleas, E. Cormier-Michel, W. M. Fawley, C. Huang, X. Wang, B. Cowan, V. K. Decyk, E. Esarey, R. A. Fonseca, W. Lu, P. Messmer, P. Mullowney, K. Nakamura, K. Paul, G. R. Plateau, C. B. Schroeder, L. O. Silva, C. Toth, F. S. Tsung, M. Tzoufras, T. Antonsen, J. Vieira, and W. P. Leemans, "Computational Studies and oOtimization of Wakefield Accelerators," *Journal of Physics: Conference Series V 125*, pp. 12 002/1–11, 2008, LBNL-2245E.

[48] E. Michel, B. A. Shadwick, C. B. Schroeder, C. G. R.Geddes, E. Esarey, W. P. Leemans, H. Ruhl, and T. Cowan, "Accurate Modeling of Laser-Plasma Accelerators with Particle-In-Cell Codes," in *Advanced Accelerator Concepts: 12th Advanced Accelerator Concepts Workshop*, ser. American Institute of Physics Conference Series, M. Conde and C. Eyberger, Eds., vol. 877, Nov. 2006, pp. 213–219.

[49] C. Nieter and J. R. Cary, "VORPAL: A Versatile Plasma Simulation Code," *J. Comput. Phys.*, vol. 196, no. 2, pp. 448–473, 2004.

[50] J. Wang, P. Liewer, and V. Decyk, "3D electromagnetic plasma particle simulations on a MIMD parallel computer," *Computer Physics Communications*, vol. 87, pp. 35–53, 1995.

[51] Vizschema is available from https://ice.txcorp.com/trac/vizschema.

[52] L. Gosink, J. Shalf, K. Stockinger, K. Wu, and E. W. Bethel, "HDF5-FastQuery: Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices," in *Proceedings of the 18th International Conference*

on Scientific and Statistical Database Management. IEEE Computer Society Press, July 2006.

[53] HDF5 is available from http://www.hdfgroup.org/HDF5/index.html.

[54] A. Adelmann, R. Ryne, J. Shalf, and C. Siegerist, "H5Part: A Portable High Performance Parallel Data Interface for Particle Simulations," in *Particle Accelerator Conference (PAC05)*, May 16-20 2005.

[55] A. Adelmann, A. Gsell, B. Oswald, T. Schietinger, E. W. Bethel, J. Shalf, C. Siegerist, and K. Stockinger, "Progress on H5Part: A Portable High Performance Parallel Data Interface for Electromagnetic Simulations," in *Particle Accelerator Conference PAC07*, June 2007.

[56] *VTK User's Guide*. Kitware, Inc. 5th Edition, September 2006.