Chapter 1

Automated detection and analysis of particle beams in laser-plasma accelerator simulations

Daniela M. Ushizima[1], Cameron G. Geddes[1], Estelle Cormier-Michel[1], E. Wes Bethel[1],
Janet Jacobsen[1], Prabhat[1], Oliver Ruebel[1,3], Gunther Weber[1],
Peter Messmer[2], Bernd Hamann[1], Hans Haggen[4]
[1]*Lawrence Berkeley National Laboratory, Berkeley, CA, USA*
[2]*Tech-X Corporation, Boulder, CO, USA*
[3]*University of Kaiserslautern, Germany*

1.1 Introduction

Numerical simulations [1] of laser-plasma wakefield (particle) accelerators [2] model the acceleration of electrons trapped in plasma oscillations (wakes) left behind when an intense laser pulse propagates through the plasma. The goal of these simulations is to better understand the process involved in plasma wake generation and how electrons are trapped and accelerated by the wake [3, 4]. Understanding of such accelerators, and their development, offer high accelerating gradients, potentially reducing size and cost of new accelerators.

One operating regime of interest is where a trapped subset of electrons loads the wake and forms an isolated group of accelerated particles with low spread in momentum and position [3], desirable characteristics for many applications. The electrons trapped in the wake may be accelerated to high energies, the plasma gradient in the wake reaching up to a gigaelectronvolt per centimeter [2]. High-energy electron accelerators power intense X-ray radiation to terahertz sources, and are used in many applications including medical radiotherapy and imaging [1].

To extract information from the simulation about the quality of the beam, a typical approach is to examine plots of the entire dataset, visually determining the adequate parameters necessary to select a subset of particles, which is then further analyzed. This procedure requires laborious examination of massive data sets over many time steps using several plots, a routine that is unfeasible for large data collections. Demand for automated analysis is growing along with the volume and size of simulations. Current 2D LWFA simulation datasets are typically between 1GB and 100GB in size, but simulations in 3D are of the order of TBs. The increase in the number of datasets and dataset sizes leads to a need for automatic routines to recognize particle patterns as particle bunches (beam of electrons) for subsequent analysis [5].

Because of the growth in dataset size, the application of machine learning techniques for scientific data mining is increasingly considered. In plasma simulations, Bagherjeiran et al.[6] presented a comprehensive report on applying graph-based techniques for orbit classification. They used the KAM classifier [7] to label points and components in single and multiple orbits. Love et al. [8] conducted an image space analysis of coherent structures in plasma simulations. They used a number of segmentation and region-growing techniques to isolate regions of interest in orbit plots. Both approaches analyzed particle accelerator data, targeting the system dynamics in terms of particle orbits. However, they did not address particle dynamics as a function of time or inspected the behavior of bunches of particles.

Ruebel et al. [9] addressed the visual analysis of massive laser wakefield acceleration (LWFA) simulation data using interactive procedures to query the data. Sophisticated visualization tools were provided to inspect the data manually. Ruebel et al. have integrated these tools to the visualization and analysis system VisIt [10], in addition to utilizing efficient data management based on HDF5 [11], H5Part [12, 13], and the index/query tool FastBit [14]. In [15], Ruebel et al. proposed automatic beam path analysis using a suite of methods to classify particles in simulation data and to analyze their temporal evolution. To enable researchers to accurately define particle beams, the method computes a set of measures based on the path of particles relative to the distance of the particles to a beam. To achieve good performance, this framework uses an analysis pipeline designed to quickly reduce the amount of data that needs to be considered in the actual path distance computation. As part of this process, region-growing methods are utilized to detect particle bunches at single time steps. Efficient data reduction is essential to enable automated analysis of large data sets as described in the next section, where data reduction methods are steered to the particular requirements of our clustering analysis.

Previously [5], we have described the application of a set of algorithms to automate the data analysis and classification of particle beams in the LWFA simulation data, identifying locations with high density of high energy particles. These algorithms detected high density locations (nodes) in each time step, i.e. maximum points on the particle distribution for only one spatial variable. Each node was correlated to a node in previous or later time steps by linking these nodes according to a pruned minimum spanning tree (PMST). We call the PMST representation “a lifetime diagram”, which is a graphical tool to show temporal information of high dense groups of particles in the longitudinal direction for the time series. Electron bunch compactness was described by another step of the processing, designed to partition each time step, using fuzzy clustering, into a fixed number of clusters. We combined the lifetime diagram with the clustering results to locate spatially confined beams, demonstrating the ability of the method to detect high quality beams, characterized by high energy and high degree of spatial coherence. A reported drawback of the method in [5] is the inability of detecting low quality beams due to mechanism of privileging high energy particles, therefore outputting incorrect scattered groups of particles with high energy instead of compact group of particles with low energy.

This paper extends previous work by addressing beam detection, independent of quality. We divide the investigation in two main steps: (a) detection of maximum density regions of particles using both longitudinal and transversal direction of variation and (b) multidimensional particle clustering over each time step to automatically detect isolated bunches of electrons within resulting partitions. We calculate these partitions using normal mixture models, followed by the selection of the best model according to a cluster compactness criteria. Each clustering algorithm uses a multivariate analysis to identify high density groups of electrons, iteratively searching for the best number of clusters to model the particle distribution. We employ data representation and partitioning to detect electrons undergoing acceleration, using the powerful R statistical tools [16] and have integrated the data management method of Fastbit [14] into the R analysis framework. Our main contribution is the automatic detection of compact groups of particles from large, complex and time-dependent scientific data sets of electron simulations. These groups are selected for coherence in both momentum and spatial coordinates, which are characteristic of electron beams that one wants to identify. In addition, we propose several graphical representations of data for fast information assessment that will help guide later feature extractors to derive simulation measurements. Our results show that the proposed framework can detect group of particles that belong to the electron beam even if the particle bunch is of low quality beam. This is important to allow comparison among many simulation runs with varying beam quality. We are able to automatically detect the beam and characterize it in terms of dispersion measurements, that identifies the time steps where the bunch is most condensed and under acceleration.

The next section (Sec.1.2) describes the datasets under investigation, the proposed approach and implementation details. Sec.1.3 presents the results of combining data transformation, geometrical modeling and analysis with classification of electrons from simulation time series. We conclude with discussions and future directions in Sec.1.4.

1.2 Material and methods

1.2.1 Particle acceleration simulations

LWFA simulations are used to model physical parameter variations, such as tuning of laser energy and plasma characteristics in order to determine the combination that will achieve the desired small energy spread bunch, hence guiding and improving understanding of laboratory experiments. The simulations model the properties of a hydrogen plasma, which is an ionized gas containing free electrons (not bound to a molecule) and positively charged ions [17]. As a laser pulse travels through the plasma, the electric field of the light separates electrons and positively charged ions. While the positive ions are heavy and stay in place, the light electrons are pushed away from the laser pulse creating a “bubble” of positively charged particles behind the pulse [3]. In this so-called

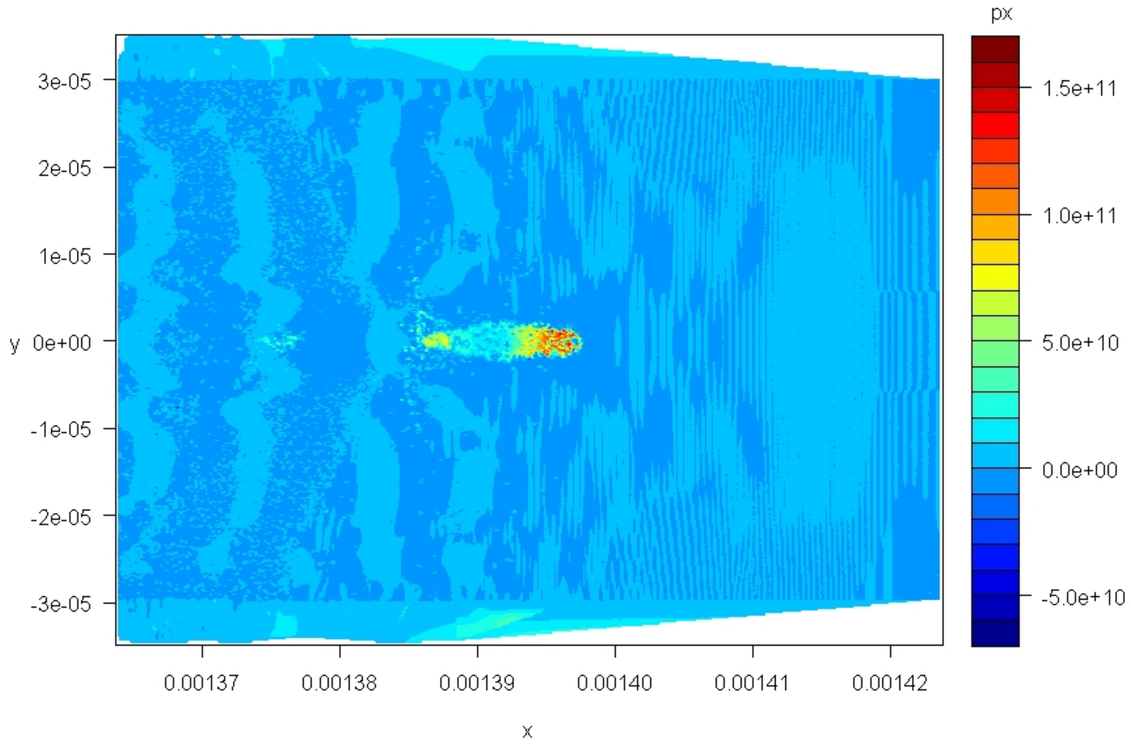


Figure 1.1: Level sets as a function of the momentum of simulation particles, for a time step known a priori to contain a high quality beam (red), characterized by high momentum (high px) in a compact envelope, given by x, y position.

blowout regime [18], a fraction of electrons can be trapped in the wave and be accelerated, in the same direction as the laser pulse, until they outrun the wave. As the accelerating structure travels at a speed less than the speed of light, the relativistic electrons eventually slip into a decelerating region of the wake, stopping the acceleration process.

The most common algorithm used to simulate LWFA is particle-in-cell (PIC) codes [19]. The PIC technique models the dynamics of particles and the electromagnetic field in a simulation window that travels at the speed of light [3, 20]. The properties of accelerated particle groups vary (e.g., duration in time, position, momentum and momentum spread) and, rather than being prescribed as inputs, are a consequence of a set of parameters defined before the simulation starts [3, 4, 21], similar to laboratory experiments [21].

One way to identify particles that were trapped at some point in time is by looking later on at the particles that have high energy. The particles with the highest energy levels are usually located in the first wake period, forming a compact bunch, as illustrated by the red region in Figure 1.1. The yellow level sets (center) show particles that can also be accelerated in wave buckets that follow the first wake. The selection of the particles of interest (the highly accelerated ones) has typically been done by looking at the later time-steps of a simulation and interactively selecting only those particles with a velocity that is larger than a defined threshold [20, 4, 9].

1.2.2 Simulation datasets

This chapter investigates datasets from 2D simulations that contain the (x, y) position of the particles as well as their momentum in the x and y directions (px, py). The laser pulse and accelerated particles propagate in the $x-$direction. Each simulation particle represents a group of electrons, with weight (*wt*). Since no identifiers (id) are stored for the particles, the particle weights are used as identifiers throughout the analysis. If, several particles have the same weight in the simulation, these are not traced. Table 1.1 presents details of the datasets used in our tests, from which we draw only traceable particles (unique identifier).

Dataset	Particles (10^6)	Timesteps	Total Size (GB)
A	0.4	37	1.3
B	1.6	37	4.5
C	0.4	38	1.3
D	3.2	45	11
E	6	39	28

Table 1.1: Tested 2D simulation datasets.

Data access requires efficient readers, provided by H5Part [22]. H5Part is a veneer API on top of HDF5 that considerably simplifies reading and writing simulation data to HDF5 files. In order to efficiently query large particle datasets, we utilize the capabilities of FastBit, a state-of-the-art index/query system [23, 24, 14]. FastBit resolves queries in a time proportional to the number of hits satisfying the query. This capability is essential when dealing with datasets of hundreds of millions of particles, where the interesting particles might only number in the hundreds or thousands. A naive (non-indexed) scheme would need to load up the entire dataset to resolve the query, which is prohibitively expensive for large datasets. This index/query system supports conditional queries, e.g. "*detect all particles such that* $(px > 1e10)\&(x > 0)\&(y > 5)$"; this can be used to select particles with interesting characteristics in multi-dimensional phase space. FastBit can also track selected particles across time steps by issuing queries of the form $id\ in$ $(5, 10, 31)$, which pulls out data for the three specific particles.

FastBit indices are stored within H5Part files and accessed using a custom C++ interface called HDF5-FastQuery [25]. All our analysis software is written in R. Therefore, in order to utilize FastBit's functionality within the R runtime, we extended the RcppTemplate package [26] to make function calls to the HDF5-FastQuery interface. This saves us considerable time to load subsets of particle data, at least 6.6 times faster than R-package *hdf5*. This is a considerable improvement over existing HDF5 packages in R, which often constrain the user to load the entire HDF5 file or complete groups within the file. In addition to efficient data access, our framework implements data reduction by using physical domain knowledge, data analysis algorithms and clustering techniques as described in the following sections.

1.2.3 Proposed framework

Unlike image data, composed of pixel values at regular spaces, laser wakefield simulations contain particles irregularly spaced in all dimensions. The scattered data is a common problem in scientific data mining [27] when trying to extract patterns in large datasets, particularly because the physical phenomenon is evolving over time. Data reduction of large datasets is often mandatory before applying clustering algorithms due to their inherent combinatorial complexity. Figure 1.2 shows our framework for detection of accelerated electron bunches in LWFA simulations; the algorithms for data partitioning and pattern detection are detailed in the next sections.

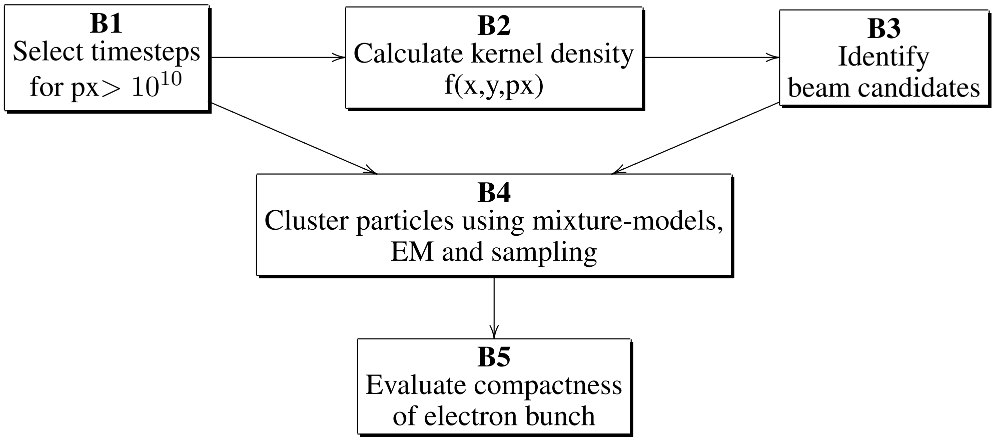
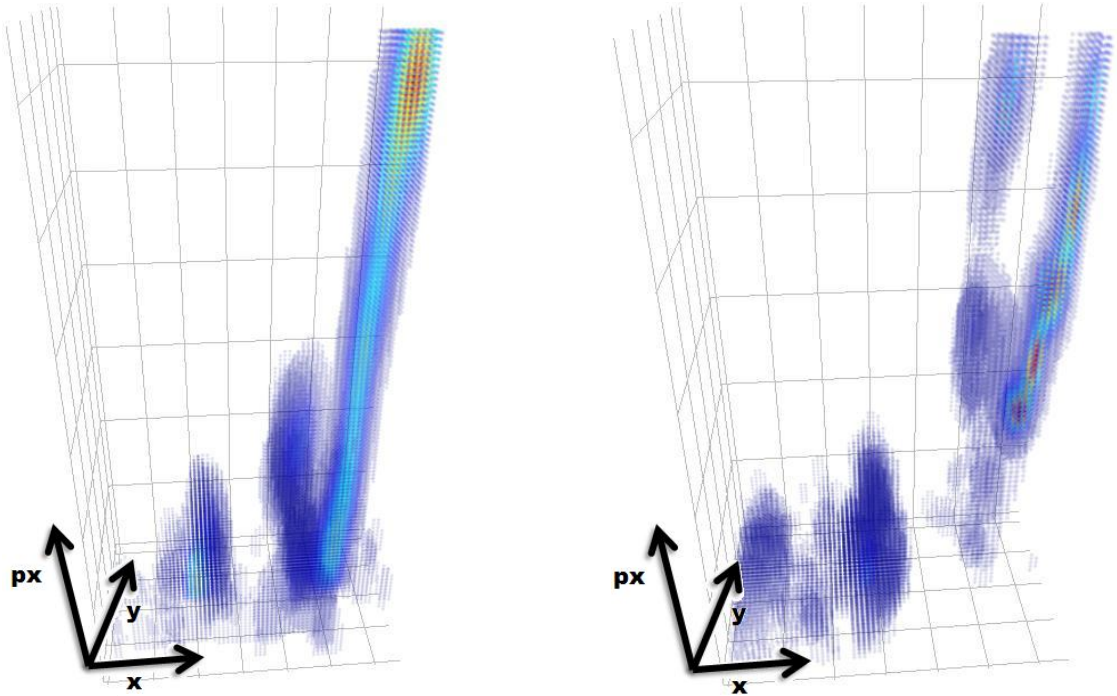


Figure 1.2: Framework for data reduction and beam detection applied to each time step of data sets generated by laser wakefield acceleration (LWFA) simulations.

The first step (B1) selects particles and time steps relevant for inspection from a n-time step simulation dataset, discarding those particles unlikely to belong to the physical phenomenon of interest. The pipeline obtains particle distribution (B2), using kernels to calculate an estimate ($f(x, y, px)$) of the probability density function. Next, we find parameters x, y, px for which f is maximum, selecting a subset of particles that may correspond to trapped bunches of electrons (B3). The following step (B4) then groups the simulation particles according to normal mixture models, before applying maximum likelihood estimation and Bayes criteria. The goal is to identify the most likely model and number of clusters that better refine the previous beam candidate particles. The simulation contains several time steps and varying number of particles per time step; we combine the result of beam detection for each time step and calculate statistics of the time series by applying moving averages (B5) to characterize the electron bunch.



(a) High quality beam

(b) Low quality beam

Figure 1.3: Kernel density estimation of Dataset A (left) and D (right) at a single time step, showing high-density bunches (red): 3D representation of $f(x, y, px)$ with heatmap colors representing the particle density.

High energy particles and densities (B1-B2)

Block B1 performs particle selection given a threshold in momentum in the $x-$direction, based on the fact that the bunch of electrons of interest should be observed near $px = 10^{11}$. We can then eliminate the low energy particles for which $px < 10^{10}$. The expected wake oscillation is up to $px = 10^9$. Therefore this threshold excludes particles of the background plasma, while including all particles that could be in an accelerated bunch. The precise choice of the threshold does not affect the result accuracy and a lower threshold could be used at higher computational cost [5]. After eliminating low momentum particles, some time steps (in general the first few time steps of the simulation) may not include a relevant amount of particles for inspection. We calculate the simulation average number of particles (μ_s) to determine the "representative" time steps, t_i, for which there is a number of particles greater than μ_s, determining an even smaller subset of time steps. We observed that this constraint eliminates initial time steps, but maintains consecutive time steps throughout the time series from t_{μ_s}, the first time step for which the number of particles is greater than μ_s. Again, this threshold can be adjusted to lower values.

It is necessary to compute the frequency of the particles given the (x, y, px) parameters of

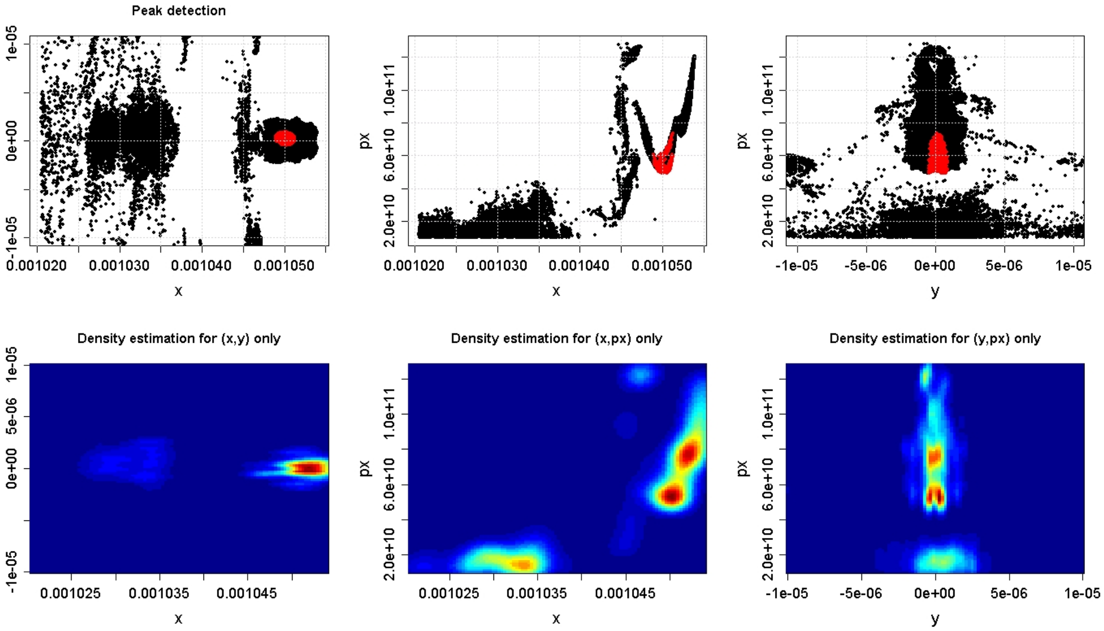
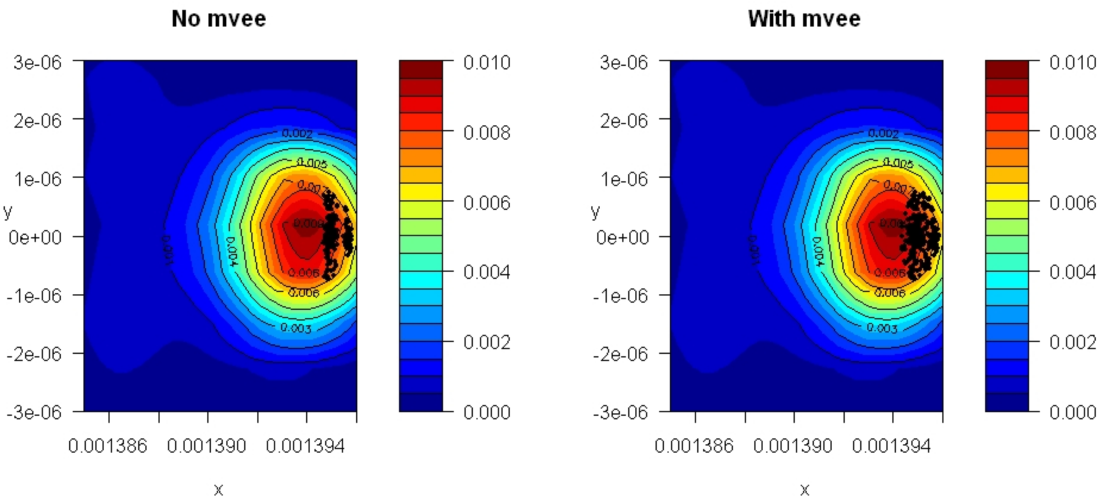


Figure 1.4: Projections of beam candidate region detection (block B3) from timestep in dataset D (top) and 2D particle density estimations (bottom) to confirm compactness of selected particles.

the particles in each time step. The most widely used nonparametric density estimator is the histogram, whose main disadvantage is its sensitivity to the placement of the bin edges, a problem not shared by kernel density estimators [28]. Kernel density estimators are hence a valuable tool to identify subgroups of samples with inhomogeneous behavior [29] and to recover the underlying structure of the dataset, while minimizing binning artifacts. The PDF estimation also depends on the number of particles and a set of smoothing parameters called bandwidth [29].

We estimate the probability density function (PDF) $f(x,y,px)$ for time steps $t = [t_{\mu_s}, T]$ in B2, where T is the original number of time steps in the simulation, before extracting beam candidate regions. An estimation of the PDF is calculated using kernel density estimation [30] and defined on a grid with spacing $\Delta x = 0.5\mu$m, $\Delta y = 0.5\mu$m and $\Delta px \approx 10^9$. These parameters are selected based on the physical expectation of electron beam size to be 2μm and momentum spread to be approximately 10^{10} [5]. This PDF will be used later for retrieval of the maximum value and the first adjacent bins.

Figure 1.3 shows 3D densities that are the result of the calculated multivariate kernel density estimators and illustrates the concepts of high and low quality beams. Notice that Fig.1.3(a) presents a concentrated region in (x,y,px) and higher values of px in comparison with Fig.1.3(b), which has scattered (red) groups with lower values of px, indicating a low-quality beam. Next, we propose a method to detect these groups of particles, independently of the range of energies they present.



(a) **(b)**

Figure 1.5: Comparison of particle selection with/without MVEE: extracting the orientation and the axes of an enclosing ellipse from (a) produces (b), increasing the number of particles from 173 to 263. Colors indicate the density of particles, using only (x, y)-coordinates, and black dots show potential particles to belong to the beam, according to the different methods.

Deriving maximum peaks (B3)

The task in B3 is to find particles at maximum values of f and their immediate vicinity to obtain compact electron bunches in space and with limited dispersion in momentum, as emphasized in red in Figure 1.4. These criteria determine the ability to characterize the quality of particle beams, which depends on the grouping of electrons in terms of their spatial parameters as well as momentum in the longitudinal (x) and transverse (y) directions. The binning used to calculate f may interfere in the beam quality descriptors if only the absolute maximum of the PDF is taken into account, e.g., the bins may separate a maximum peak into parts if the binning is too small to contain the particles of interest. To prevent this undesirable effect, we adopt a tolerance parameter to select compact bunches and extract more than one maximum (beam candidate region) per time step. In addition, this is a way of accruing more samples and detecting secondary beams when these are almost as prominent as the primary beam, associated to the maximum of f.

During the searching for values that are approximately equal to $max(f)$, we keep not only the maximum, but all bins where $f \geq u * max(f)$, where u is an uncertainty or tolerance parameter, here empirically set to 0.85. While this value enables the detection of the main and the secondary beams (when present), lower values of u could be used to control the amount of particles to be selected at a lower accuracy of beam position. From this point, we refer to the subset of particles conditioned to $u * max(f)$ and its adjacency, calculated for each time step, as “beam candidates”.

Figure 1.4 (top) presents projections of Figure 1.3.b with their calculated beam candidates emphasized in red. These are the result of our first attempt to improve particle selection by using an algorithm known as minimum volume enclosing ellipsoid [31], which is able to enclose previously selected particles and to include others based on a geometrically defined polytope. Figure 1.5 illustrates the algorithm when applied to LWFA data, showing the selected particles as black dots; these particles are not in the most dense region (red) once the colors refers to (x, y)-density calculation. When including compactness in px, the most dense region happens further ahead. As distinct from calculating center of mass and forcing an *ad hoc* diameter or semi-major/minor axes, the minimum volume enclosing ellipsoid (MVEE) algorithm [31, 32, 33] takes the subset of points and prescribes a polytope model to extrapolate a preliminary sub-selection to other particles likely to be in the bunch. The MVEE algorithm is a semidefinite programming problem and consists of a better approximation to the convexity of subsets of particles that correspond to compact groups of electrons. After querying hypervolumes similar to the one in Figure 1.3, we applied the geometrical model to adjust the particle selection as illustrated in Figure 1.5. By running the MVEE algorithm, we determine an ellipse as compact as possible covering the data points of the beam candidate region, increasing the number of samples without increasing the binning parameters. Here, we consider the problem of finding a MVEE, that minimizes the logarithm of the determinant of H such that

$$(x - c)^T H(x - c) \leq n \tag{1.1}$$

for a set of points x in R^n, an ellipsoid with center c and shape H [31].

In addition to methods to select beam particles and graphics for each time step, it is often useful to track the bins occupied by the beam candidates by using lifetime diagrams. Figure 1.6 shows earlier time steps containing a bunch of particles that remains at constant speed, with dispersion around $t = 32$ and formation of a second bunch around $t = 35$. This diagram show the whole simulation, i.e. a global representation of the temporal evolution of beam candidate in bins. The diagram relates the time steps (t) to the relative position in the simulation window (x), which corresponds to the maximum of f for each time step of the simulation, as calculated in block B3. The algorithms described in this section focused on the location of the beam. The next section complements the search for the beam by finding subsets of simulated particles, according to clustering techniques.

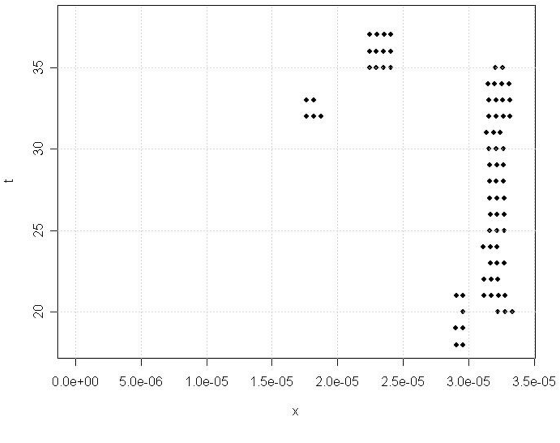


Figure 1.6: Lifetime diagram of peaks evolving in time for simulation A: temporal representation of beam candidates.

Clustering particles (B4)

Data partitioning is a conceptually intuitive method of organizing simulation particles into similar groups given the absence of class labels. Since clustering is an unsupervised learning method, evaluation and cluster quality assessment are valuable in interpreting classification results. We include both clustering methods and cluster validity techniques applied to particle acceleration to illustrate the applicability of dispersion measures to accurately evaluate an intrinsic structure, namely, a coherent bunch of electrons.

In order to determine the number of clusters in each time step of the simulation while testing statistical models with different numbers of components, we perform cluster analysis using model-based clustering, where the model and the number of clusters are selected at run time by *Mclust* [34]. The model-based clustering algorithm postulates a statistical model for the samples, which are assumed to come from a mixture of normal probability densities. The calculation of normal mixture models considers different covariance structures and different number of clusters [35] given an objective function (score). The assumption of the number of clusters k entails a loss of generality, so we consider a range of k in addition to parameters that control the shape of the class. These parametric models are flexible in accommodating data as shown in [36] and consider widely varying characteristics in estimating distributions.

By assuming a normal mixture model as in [37], we represent the data d, with n samples and k components, considering a τ_k probability that an observation belongs to the kth component and a multivariate normal distribution φ_k with mean vector μ_k and covariance matrix Σ_k, namely

$$\prod_{i=1}^{n} \sum_{k=1}^{G} \tau_k \varphi_k(d_i|\mu_k, \Sigma_k) \tag{1.2}$$

with priors conditioned to

$$\tau \geq 0; \sum_{k=1}^{G} \tau_k = 1. \tag{1.3}$$

The maximum likelihood is equivalent to the maximum log-likelihood function L [38] of $\theta_k = (\mu_k, \Sigma_k)$ with respect to the data samples d_i, namely

$$L(\theta_k) = \sum_{i=1}^{n} ln\, p(d_i|\theta_k) \tag{1.4}$$

The expectation-maximization (EM) algorithm relies on iterative two-fold processing: an E-step for calculating the conditional probability that an observation belongs to a certain group given the parameters θ_k, and a M-step for computing the parameters that maximize the log-likelihood given the previously calculated conditional probability function [36]. In other words, EM determines the most likely parameters $\theta_1, ..., \theta_k$ to represent a problem consisting of multivariate observations given by a mixture of k underlying probability distributions [37].

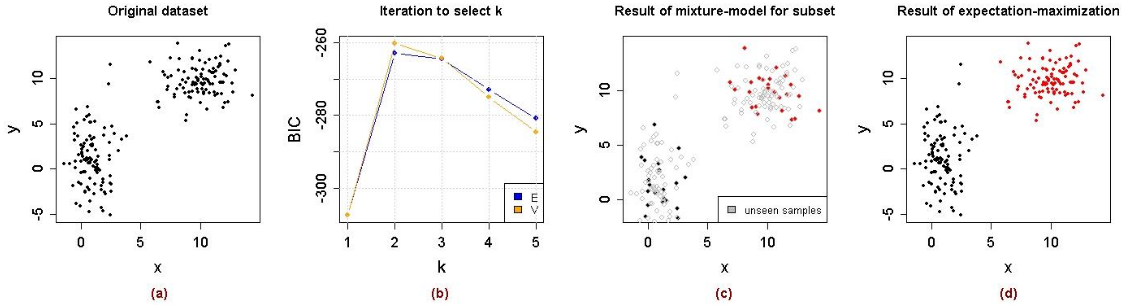


Figure 1.7: Example of model-based clustering to clouds of points: (a) two Gaussian distributions with no label assignment; (b) Bayesian information criteria calculation for different number of clusters (k) and different models (E=equal volume and V=varying volume); (c) result of classification using subset (25%) of the data; (d) generalization of model using expectation-maximization.

The size of the LWFA datasets can compromise the efficiency of mixture model-based algorithms due to *mclust* initialization [36], then we propose a random sampling technique. To illustrate such algorithm, Figure 1.7 uses artificial data, generated by two normal distributions $g1(x, y)$ and $g2(x, y)$ with 100 unlabeled samples each (Figure 1.7.a). In this example, we subsample the data by extracting a quarter of its original samples and calculate mixture-models, varying the structure and the number of the clusters (Figure 1.7.b). The result of the clustering provides labels for a quarter of the samples (black and red dots in Figure 1.7.c) and these labels support a supervised learning to classify the remaining samples as in Figure 1.7.d, a generalization procedure to extrapolate the "learned" models to the full dataset by using expectation-maximization.

Instead of imposing k, which is not known a priori, the objects are associated to each other according to a score that comes from the parameters, unknown quantities to be estimated from the probability distributions [39]. Figure 1.7.b shows the calculation of a score for different k and the maximum value of the curves imply the number of k the best describe the samples.

This process establishes the inference on the sample rather than on the full population [36]. This decision circumvents the bottleneck of the *mclust* initialization by a sampling strategy to partition large datasets: we propose a biased sampling process that ensures that the beam candidate region in the sampled subset by guaranteeing that at least 10% (empirically chosen) of the samples belong to the high density particle volumes. We cluster the particles using the normal mixture models for values of $k \in [1, 10]$ to follow an ellipsoidal model with variable volume (VEV) [40, 34]. We have tested other models as spherical, diagonal and ellipsoidal, which can have equal or varying volume and shapes. However, VEV was the best algorithm for most of the time steps in all the datasets, according to the Bayesian information criteria [41]. We re-run the experiments to have clustering results using VEV only, but a varying number of k. The resulting clusters from VEV are considered as the training set to classify all the remaining samples by using EM to extrapolate parameters from training samples.

The result of the clustering (B4) is combined with B3 by calculating the intersection between the beam candidates and the a cluster that contains most of the particles from the beam candidates. In other words, we determine which cluster is most likely to contain the beam candidates by majority voting among all possible clusters, finalizing the tasks in block B4. The block B5 only analyzes the most compact group of particles that remains in the each time step.

Cluster quality assessment (B4-B5)

One of our goals in investigating particle simulations is to detect the electron beam and to characterize the dispersion of its particles in terms of spatial and momentum variables using clustering algorithms. Since we do not know a priori the number of clusters that best describe the particle grouping, we need some measure of goodness of fit to evaluate different clustering algorithms. A standard approach is to obtain the number of clusters (k) by maximizing a criterion function and to repeat the clustering procedure for different number of clusters.

We select k by maximizing the Bayesian information criterion (BIC) for a parametrized clustering algorithm using mixture models, following an ellipsoidal, varying volume model. The optimal BIC value considers the log-likelihood, the dimension of the data, and the number of mixture components in the model. The criterion function must describe how well a given clustering algorithm can match the data, defined as a function of the variable k.

Herein we will evaluate the goodness-of-fit of the clustering algorithms for k groups of particles from each time step using BIC to guide model selection for a set of parameterized mixture models with a varying number of classes. BIC adds a penalty to the log-likelihood by considering the number of parameters in a certain model M and the number of observations (n) in the data set [37], with the form

$$BIC \equiv 2\ loglik_M(d, \theta_k^*) - (\#params)M\ log(n) \tag{1.5}$$

where $loglik_M(d, \theta_k^*)$ is the maximized log-likelihood of the model with estimated parameters θ_k^* from the observations d and (#params) number of independent parameters to be estimated in M [36].

In addition to the evaluation of the clustering method (B4), we also want to verify if our framework can capture the physical phenomena of trapping and acceleration, when the beam is expected to be more compact. We propose the inspection of the particles in adjacent time steps using moving averages [42] to identify if the electrons are grouped into stable bunches (B5).

The moving averages technique provides a simple way of seeing patterns in time series data, smooths out short-term fluctuations and highlights longer-term trends. This is physically motivated as the bunches of interest move at speed approximately equal to the speed of light, and hence are nearly stationary in the moving simulation window. We intersect particle bunches (b) at adjacent time steps, selecting the particles with the same identifier (id) and calculate statistical parameters (ρ) of a three-point moving average (mv_k), using the following algorithm:

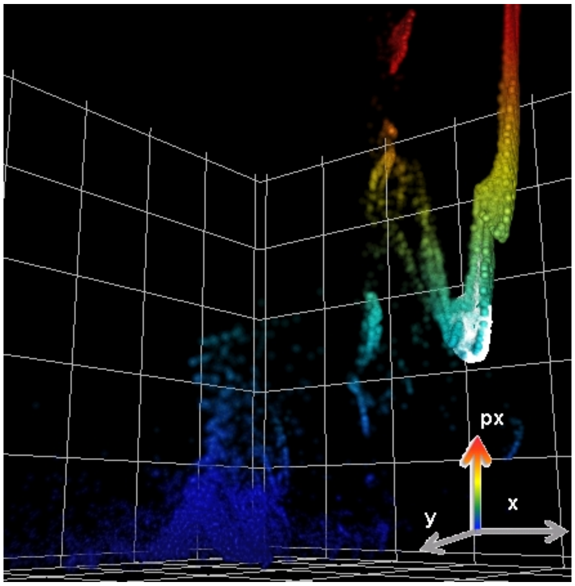


Figure 1.8: Result of locating high-density bunches for one time step of dataset D: 3D scatter plot of particles, color is proportional to the particle energy (px) and white blobs correspond to the preliminary detection of beam candidate region as described in Sec.1.2.3.

```
$k \leftarrow 1$
**for** $t = 2$ to n-1 **do**
    $id_k \leftarrow id(b_{t-1}) \cap id(b_t) \cap id(b_{t+1})$
    $mv_k \leftarrow (b_{t-1}|_{id_k} + b_t|_{id_k} + b_{t+1}|_{id_k})/3$
    $\rho \leftarrow statistics(mv_k)$
    $k \leftarrow k + 1$
**end for**
```

The particles of the bunch at time step $t-1$, b_{t-1}, indexed by id_k, are called $b_{t-1}|_{id_k}$ and the function *statistics* calculates parameters such as the mean, variance and maximum values from the moving averages. We use the plots in Figure 1.10, 1.13 and 1.14 to check the persistence of particle bunches by looking at the evolution of statistical parameters as discussed in the next section.

1.3 Results

Here, we apply the above-described algorithms to analyze laser-plasma wakefield acceleration simulations, using the clustering techniques as part of a completely automated pipeline to detect dense particle groups ("electron bunches"). The main contributions of this work, in comparison

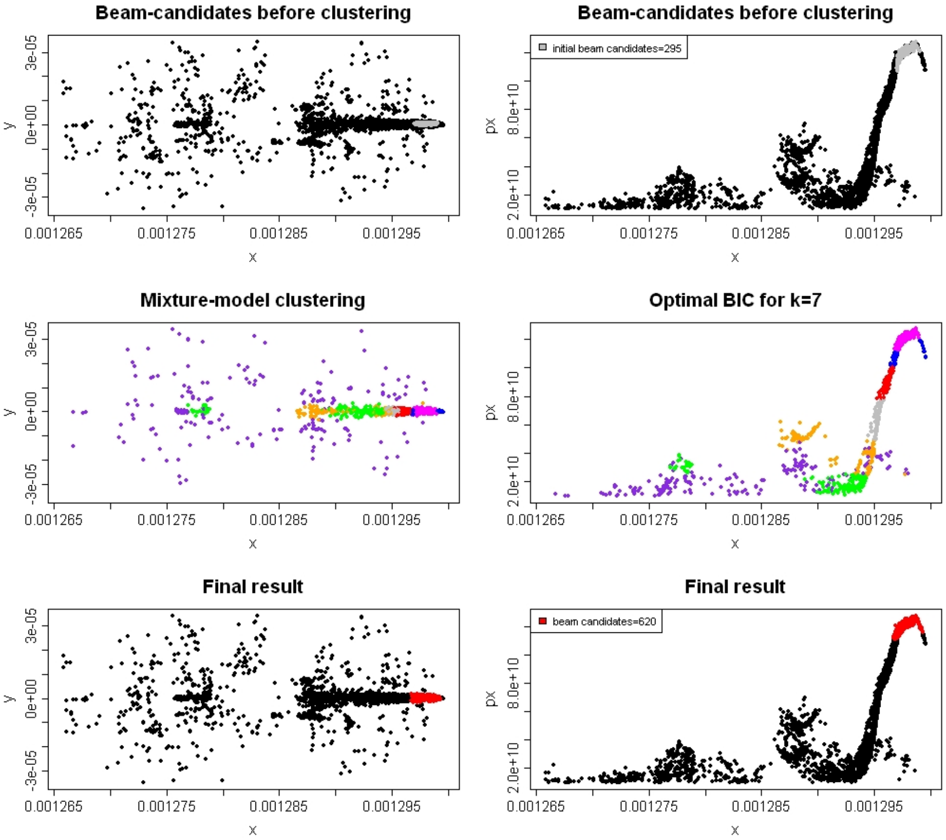


Figure 1.9: Result of beam detection for $t_i = 27$ from dataset A: beam candidates in gray from processing in block B3 (top), clustering using mixture models, with colors representing the different partitions over a sampled subset (center) and final result of electron bunch detection, with increased number of particles after generalization with EM-algorithm, from block B4 (bottom).

with the previous approach in [5], are that we can perform beam detection independent of the quality or energy of the beam. Thus compact groups of particles can have either high momentum or low momentum, instead of only being able to correctly detect groups of particles exhibiting high momentum. This improvement stems from determining particle distribution using kernel density estimators, which minimizes the sensitivity of bin size assumptions and placement, enabling accurate detection of maximum values of $f(x, y, px)$. This is in contrast with the previous method that considered f as function of x only. Also, while [9] relies on user interaction, here we automatically detect compact groups of particles under acceleration.

We show that using the particle x-coordinate relative to the window size, we can keep track of the maximum values of the kernel density functions and represent these points using lifetime diagrams. Figure 1.6 shows the evolution of peaks from $f(x, y, px)$, which will support future work to restrict the search for compact bunches using clustering to specific regions around the maximum values. Figure 1.8 illustrates the result of identifying beam candidate from block B3 at a

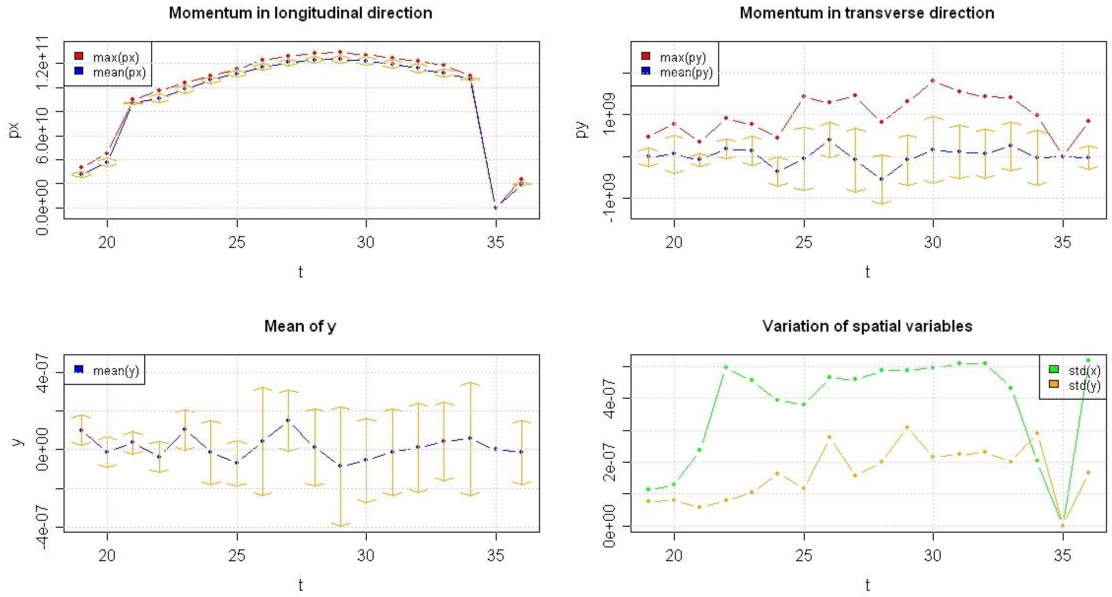
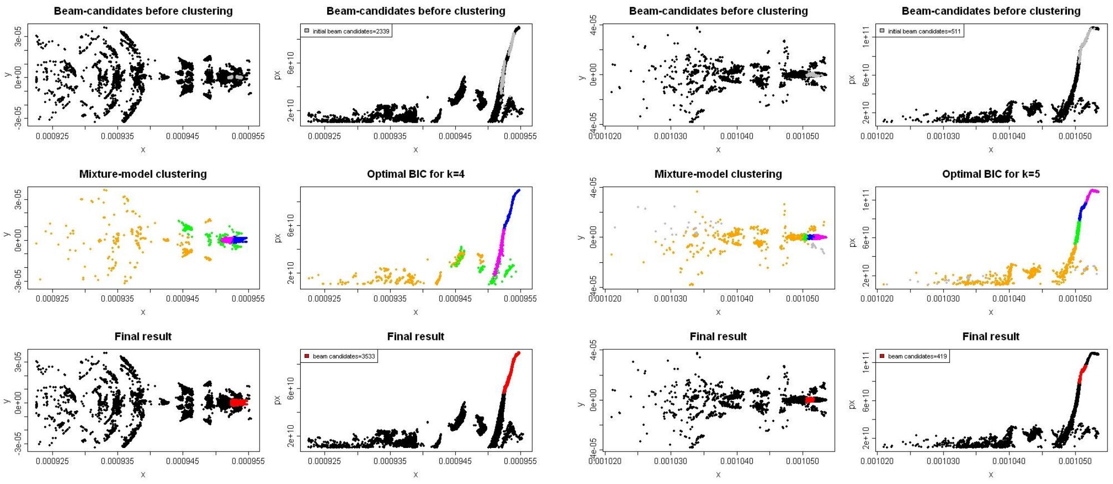


Figure 1.10: Beam quality assessment to evaluate the dispersion of particle parameters using the time series in dataset A: the curves show the history of one bunch that forms around $t = 22$, reaching maximum energy around $t = 27$.

single time step from the dataset D, showing the (x, y, px)-coordinates of particles and respective detected compact groups. The beam candidate region is represented by a cloud of white dots, containing all the particles for which $0.85 * max(f)$ holds.

The application of geometrical models such as the MVEE to enclose the detected beam candidates shows how structure assumptions may interfere in the number of particles selected, as illustrated in Figure 1.5. The advantage of this method is that it expands a previous restrictive selection to other potential points that should be included in the beam candidate region. As opposed to an approach that sets a fixed diameter, it also avoids an undesirable impact on the particle spread. We report results considering a geometrical model that encompass the beam candidate region by calculating the MVEE applied to preliminary selection of particles, which was mostly consistent with the shape of the bunch. The geometry assumption may result in inclusion of outliers if the beam present different shapes; however, we eliminate outliers during the moving averages procedure, keeping particles more likely to be part of the electron bunch.

We calculate model-based clusters for each time step, after retrieving the results from block B1 and B3. We illustrate the partitions of one time step of all datasets in Figure 1.9, 1.11 and 1.12, showing the phase space of a time step where the beam was expected to be compact. In Figure 1.9, the two top plots show the result of beam candidate selection, in gray, for dataset A as output by block B3. The two center plots present different compact groups of particles given by the mixture model, and the bottom plots give the final result of electron bunch selection (B4),

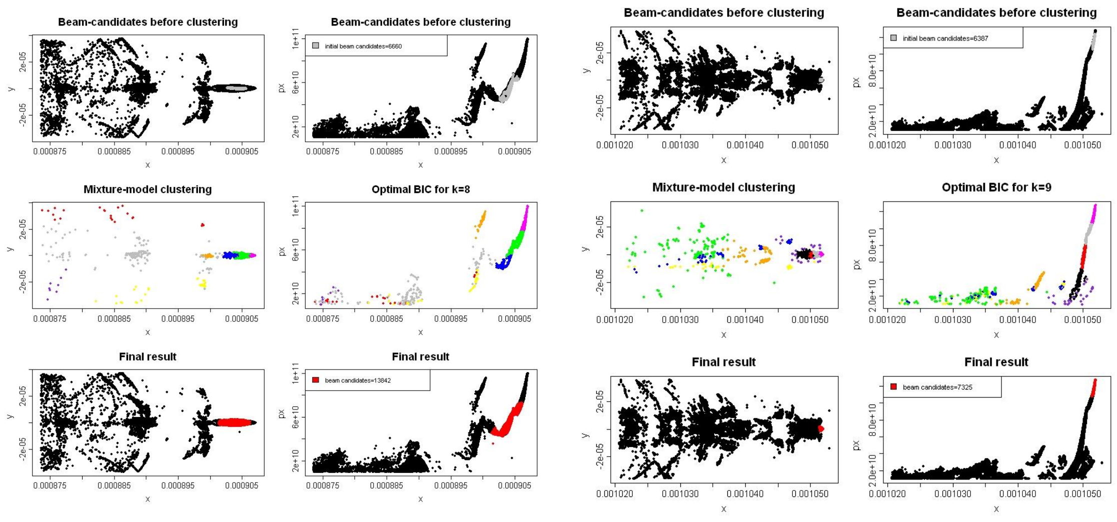


(a) Dataset B, t=21 **(b)** Dataset C, t=22

Figure 1.11: Result of beam detection for $t_i = 21$ from dataset B and $t_i = 22$ from dataset C: beam candidates in gray from processing in block B3 (top), clustering using mixture models, with colors representing the different partitions over a sampled subset (center) and final result of electron bunch detection, after generalization with EM-algorithm, from block B4 (bottom).

emphasized in red color. The result of B3 indicates potential clusters of particles, important to guide the sampling and identify the cluster position, but the definition of particle partitions that are connected and compact given x, y, px is only accomplished after B4, which finds the k-component varying-volume ellipsoidal mixture model clustering that best represent the particles, using BIC as criterion function.

Next, we evaluate the compactness of the electron bunch (B5) by calculating moving averages (mv_j) over the time series. Figures 1.10, 1.13 and 1.14 show the result of calculating statistics from mv_j, using the particles selected according to block B4. While the beam detection at each time step may contain outliers, the intersection with adjacent time steps returns the core subset of particles (id_j) that persists at least for three time steps. At the top left of Figure 1.10, we show the red and blue curves, with the maximum and mean value of px (red and blue, respectively), for each time step. The distance between the red curve from the blue curve, at each time step, is an indicator of the dispersion of the particles in the bunch as well as the length of the yellow arrows (standard deviation of the mv_j with respect to x, y, px or py). Also, notice that the moving averages capture the local behavior of a particle bunch that persists for at least three time steps, but does not guarantee that the bunch is present throughout the simulation. There are time steps where the algorithm does not capture any beam, which correspond to moving average equal to zero as in $t = [28, 34]$ from dataset D in Figure 1.14.a. The period of non-bunch detection, $mv_j = 0$, corresponds to the presence of peaks on f at different, non-adjacent positions, which is correlated to the dispersion of the particles for that period. It follows similar interpretation of the particle dispersion in terms of spatial parameters (x and y) and energy (px and py) to other datasets.



(a) Dataset D, t=19

(b) Dataset E, t=23

Figure 1.12: Result of beam detection for $t_i = 19$ from dataset D and $t_i = 23$ from dataset E: beam candidates in gray from processing in block B3 (top), clustering using mixture models, with colors representing the different partitions over a sampled subset (center) and final result of electron bunch detection, after generalization with EM-algorithm, from block B4 (bottom).

Figures 1.10, 1.13 and 1.14 demonstrate that the algorithm automatically identifies the bunch over a range of simulation conditions and resulting bunch qualities.

Our tests were conducted on an SGI Altix with 32 1.4 GHz Itanium-2 Processors and 180 GBytes of shared memory. The primary motivation for using this computing system is the large memory; the current implementation of the mixture model clustering algorithms in package *mclust* is fairly memory-intensive and does not work on standard workstations for large datasets. The SGI Altix is a multi-user machine, thus computing times in different stages of the framework are approximate. Our process of computing beam candidate regions (block B3) is reasonably fast and could be easily incorporated into routine inspection as a preprocessing step. The clustering computation is more expensive, and new implementations are necessary to improve performance. The approximate computing times of beam candidate (in seconds) and clustering (in minutes) for each dataset are organized as pairs with time in parenthesis: A=(15.6s, 31min), B=(66s, 20min), C=(24.3s, 42min), D=(295.8s, 116min) and E=(417.4s, 975min).

1.4 Conclusions and Future Work

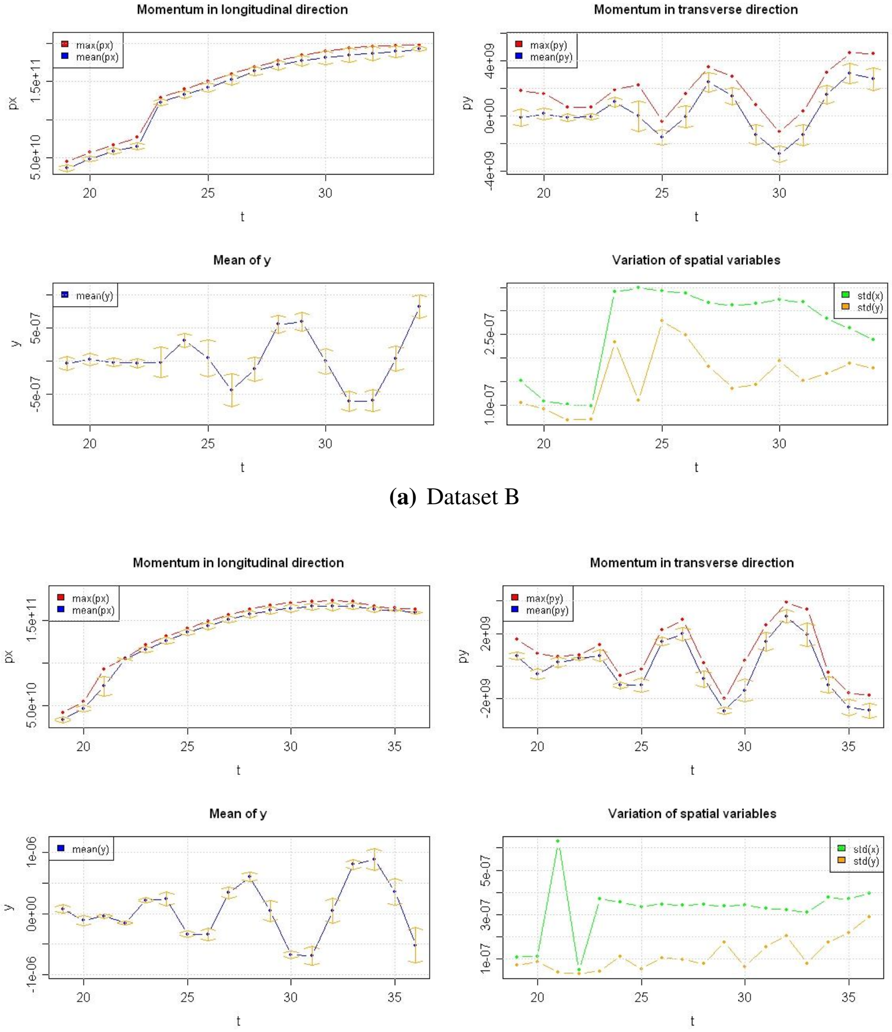
Previous works [9, 5] to find particle bunches reported results using fixed spatial tolerance around centers of maximum compactness and assumed *ad hoc* thresholding values to determine potential particle candidates involved in the physical phenomena of interest. In [5], we pointed out limitations inherent to techniques that detects maximum values using only one-dimensional spatial approach (x-axis), which did not capture the most condensed structure when confined to depressions between peaks in px or when dispersed in y.

The current approach circumvented most of these problems, since the algorithm searched for compact high density group of particles using both spatial information, x and y, and momentum in the direction of laser propagation, px. We improved the detection of a high density volume of particles by using the 3D kernel density, followed by the detection of its maximum and enclosing the particle subsets using MVEE, thus generating subsets of particles which are beam candidate regions. These subsets provided the position of the most likely cluster to contain a compact electron bunch in a time step. We proposed the use of moving averages to identify periods of bunch stability, in the time series, and we derived dispersion measures to characterize beam compactness and quality.

Our implementation of function calls to the HDF-FastQuery interface allowed to load data using FastBit in R, saving time while only loading subsets of particles that potentially participate to the phenomenon of interest. Our results showed that we can assess the beam evolution using both mathematical models and machine learning techniques to automate the search for the beam using large LWFA simulation datasets. Application of hierarchical approaches as in the R packages *hclust* and *mclust* are prohibitive if not combined with sampling methods. We present an algorithm to sample the simulation data, but Monte Carlo methods [40] could be used by adding a repetitive randomness process as a way of guaranteeing representation of a beam candidate region and improvement of accuracy. Future evaluations may consider more sophisticated methods as Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [43] and hierarchical clustering based on granularity [44], which are designed for very large data sets. Further investigation should also include subspace clustering [45] once the large simulation datasets contain target regions that can be determined using the techniques proposed in our framework.

1.5 Acknowledgments

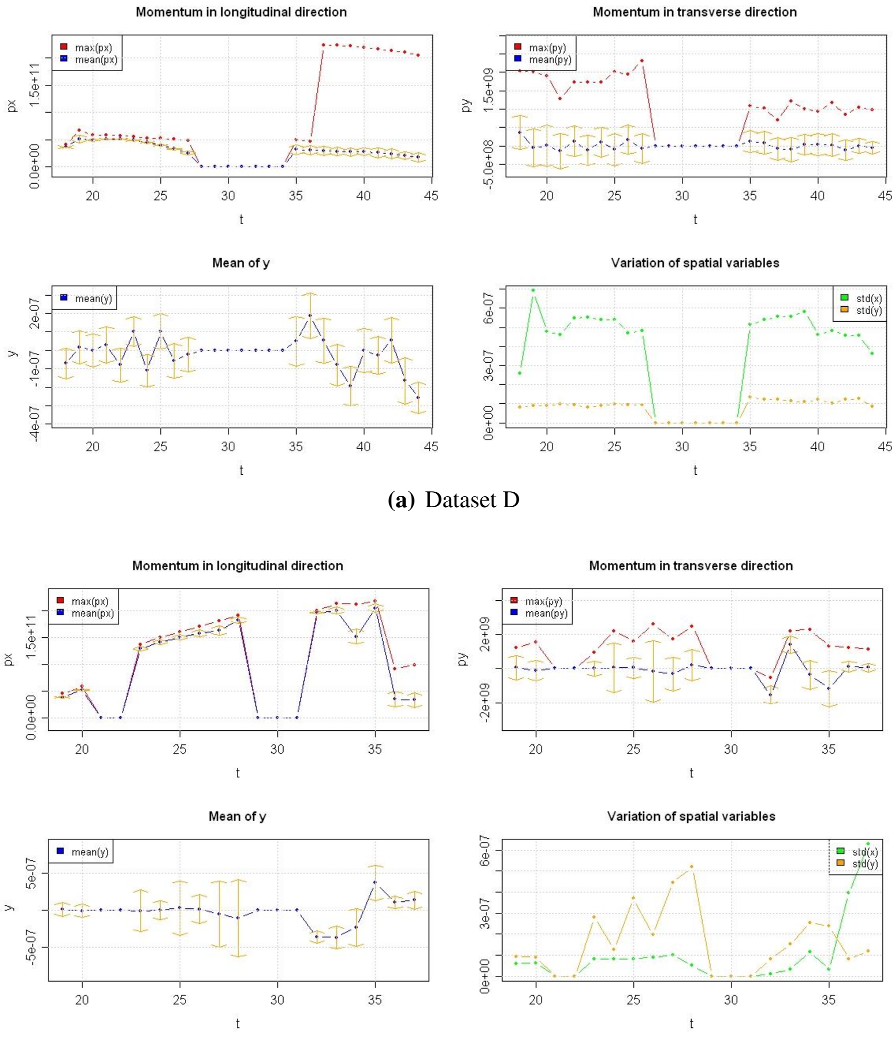
This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET) and by the U.S. DOE Office of Science, Office of High Energy Physics, grant DE-FC02-07ER41499, through the COMPASS SciDAC project and by the U.S. DOE Office of Energy Research by the Applied Mathematical Science subprogram, under Contract Number DE-AC03-76SF00098. This research used resources of the National Energy Research Scientific Computing Center, 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 development team for ongoing efforts in development and maintenance on a variety of supercomputing platforms, including those at NERSC [46].



(a) Dataset B

(b) Dataset C

Figure 1.13: Beam quality assessment to evaluate the dispersion of particle parameters using the time series in: (a) dataset B: the curves show the history of one bunch that forms around $t = 23$, reaching maximum energy around $t = 33$; (b) dataset C: the curves show the history of one bunch that forms around $t = 21$, reaching maximum energy around $t = 33$.



(a) Dataset D

(b) Dataset E

Figure 1.14: Beam quality assessment to evaluate the dispersion of particle parameters using the time series in: (a) dataset D: the curves show the history of two bunches: one that forms at the beginning of the simulation, compact and lower energy ($t = [18, 27]$) and a second one with broader dispersion in px and higher energy ($t = [36, 44]$). The beam is not detected by the algorithm from $t = [28, 34]$, represented by zero values in the four graphs; (b) dataset E: the curves show the history of two bunches that form around $t = 23$, reaching maximum energy and compactness around $t = 34$. The beam is not detected by the algorithms from $t = [21, 22]$ and $t = [29, 31]$, represented by zero values.

Bibliography

[1] Laser plasma particle accelerators: Large fields for smaller facility sources. `http://www.scidacreview.org/0903/pdf/geddes.pdf`, 2009.

[2] T. Tajima and J. M. Dawson. Laser electron accelerator. Physical Review Letters, 43(4):267–270, 1979.

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

[4] C. G. R. Geddes. Plasma Channel Guided Laser Wakefield Accelerator. PhD thesis, University of California, Berkeley, 2005.

[5] 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 2008 Seventh International Conference on Machine Learning and Applications, Proceedings of IEEE ICMLA'08, 2008. LBNL-960E.

[6] A. Bagherjeiran and C. Kamath. Graph-based methods for orbit classification. In SDM, 2006.

[7] Kenneth Man-Kam Yip. KAM: A System for Intelligently Guided Numerical by Computer. MIT Press, 1991.

[8] 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, volume 6696, 2007.

[9] 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 (to appear).

[10] Visit. `https://wci.llnl.gov/codes/visit/`.

[11] 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. LBNL-59602.

[12] 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.

[13] 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 25–29 June, 2007. http://vis.lbl.gov/Publications/2007/LBNL-63042.pdf.

[14] Fastbit. `https://codeforge.lbl.gov/projects/fastbit/`, 2009.

[15] O. Rübel, C. G.R. Geddes, E. Cormier-Michel, K. Wu, Prabhat, Gunther H. Weber, D. M. Ushizima, P. Messmer, H. Hagen, B. Hamann, and W. Bethel. Automatic beam path analysis of laserwakefield particle acceleration data, 2009. in submission.

[16] The r project for statistical computing. `http://www.r-project.org`, 2009.

[17] K. Nakamura, E. Esarey, C. G. R. Geddes, A. J. Gonsalves, W. P. Leemans, D. Panasenko, C. B. Schroeder, and S. M. Hooker Cs. Toth. Performance of capillary discharge guided laser plasma wakefield accelerator. In PAC, pages 2978–2980, 2007.

[18] A. Pukhov and J. Meyer ter Vehn. Three-dimensional particle-in-cell simulations of laser wakefield experiments. Applied Physics B-Lasers and Optics, 74(4-5):355–361, 2002.

[19] C. K. Birdsall, A. B. Langdon, V. Vehedi, and John Paul Verboncoeur. Plasma Physics via Computer Simulations. Adam Hilger, Bristol, Eng., 1991.

[20] F.S. Tsung, T. Antonsen, D.L. Bruhwiler, J.R. Cary, V.K. Decyk, E. Esarey, C.G.R. Geddes, C. Huang, A. Hakim, T. Katsouleas, W. Lu, P. Messmer, W.B. Mori, M. Tzoufras, and J. Vieira. Three-dimensional particle-in-cell simulations of laser wakefield experiments. J. Phys.: Conf. Ser., 78(1):012077+, 2007.

[21] C. G. R. Geddes, D. L. Bruhwiler, J. R. Cary, W. B. Mori, S. F. Martins J.L. Vay, 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 optimization of wakefield accelerators. J. Phys.: Conf. Ser. 125, 125:1–11, 2008.

[22] H5part. `https://codeforge.lbl.gov/projects/h5part/`, 2009.

[23] Kesheng Wu, Ekow Otoo, and Arie Shoshani. On the performance of bitmap indices for high cardinality attributes. In VLDB, pages 24–35, 2004.

[24] Kesheng Wu, Ekow Otoo, and Arie Shoshani. Optimizing bitmap indices with efficient compression. ACM Transactions on Database Systems, 31:1–38, 2006.

[25] Hdf5-fastquery. `http://www-vis.lbl.gov/Events/SC05/HDF5FastQuery/index.html`, 2009.

[26] Rcpptemplate. `http://www.fastmirrors.org/cran/doc/packages/RcppTemplate`, 2006.

[27] C. Kamath. Scientific Data Mining: A Practical Perspective. Society for Industrial and Applied Mathematic (SIAM), Philadelphia, USA, 2009.

[28] M. P. Wand and M. C. Jones. Kernel smoothing. Chapman and Hall/CRC, 1995.

[29] R. Weissbach and O. Gefeller. A rule-of-thumb for the variable bandwidth selection in kernel hazard rate estimation, 2009.

[30] D. Feng and L. Tierney. Miscellaneous 3d plots. `http://cran.r-project.org/web/packages/misc3d/misc3d.pdf`, 2009.

[31] L.G. Khachiyan and M.J. Todd. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. Math. Program., 61(2):137–159, 1993.

[32] N. Moshtagh. Minimum volume enclosing ellipsoid. `http://www.mathworks.com/matlabcentral/fileexchange/9542`, 2009.

[33] P. Kumar and E. A. Yildirim. Minimum-volume enclosing ellipsoids and core. Journal of Optimization Theory and Applications, 126:1–21, 2005.

[34] C. Fraley and A. Raftery. Model-based clustering / normal mixture modeling: the mclust package. `http://www.stat.washington.edu/fraley/mclust`, 2009.

[35] D. Haughton, P. Legrand, and S. Woolford. Review of three latent class cluster analysis packages: Latent gold, polca and mclust. The American Statistician, 63(1):81–91, 2009.

[36] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. Journal of the American Statistical Association, 97:611–631, 2002.

[37] C. Fraley and A. Raftery. Mclust version 3 for r: Normal mixture modeling and model-based clustering. Technical Report no. 504, 2006.

[38] R. O. Duda, P.E. Hart, and D. G. Stork. Pattern Classification. John Wiley and Sons, Ltd, 2001.

[39] J.K. Vermunt and J. Magidson. Latent class cluster analysis. Applied latent class analysis, pages 89–106, 2002.

[40] J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. Biometrics, 49:803–821, 1993.

[41] D. Greene, P. Cunningham, and R. Mayer. Unsupervised learning and clustering. Machine learning techniques for multimedia, pages 51–90, 2008.

[42] R. H. Shumway and D. S. Stoffer. Time Series Analysis and Its Applications (Springer Texts in Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[43] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data, pages 103–114, New York, NY, USA, 1996. ACM.

[44] J. Liang and G. Li. Hierarchical clustering algorithm based on granularity. In GrC, pages 429–432. IEEE, 2007.

[45] H.P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data, 3(1):1–58, 2009.

[46] Nersc. `http://www.nersc.gov/`.