Hypervolumetric Plasma-data Visualization

Florence Zara, Matthieu Haefele, Christophe Mion, Jean-Michel Dischler

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Hypervolumetric Plasma-data Visualization

Florence Zara — Matthieu Haefele — Christophe Mion — Jean-Michel Dischler

N° 5971

Septembre 2006

Thème NUM D

ISSN 0249-6399 ISRN INRIA/RR--5971--FR+ENG

Hypervolumetric Plasma-data Visualization

Florence Zara * , Matthieu Haefele † , Christophe Mion ‡ , Jean-Michel Dischler †

Thème NUM D — Modélisation, simulation et analyse numérique
Projet CALVI

Rapport de recherche n° 5971 — Septembre 2006 — 16 pages

Abstract: Several techniques have been proposed to explore hypervolumetric datasets but most of them fail to be efficient when very large fields are to be processed. This report describes an interactive visualization technique designed to explore at real-time framerates very large hypervolumetric 4-D+t scalarfields (with up to 16GB raw data per time step). We introduce a new out-of-core scheme aiming at generalizing to hypervolumes, brick-based techniques, already standartly used for large volumetric (*e. g.* 3D) datasets. For a given time step, we visualize the entire 4-D space by displaying directly 2-D arrays of height fields, each height field representing a 2-D hyperslice of the 4-D space. The contribution of this work consists in introducing an efficient partitionning scheme, that we called hyper-bricks, in such a way that it overcomes all hardware bottlenecks, namely the progressive load from disk, the decompression time (CPU) and the display time (GPU). We show that our technique, further using a cache system and a level of detail representation allows users to explore the full hypervolume at real-time framerates even on low-end PCs with basic graphics cards. We mainly apply our technique to the interactive exploration of plasma behaviors resulting from large numerical semi-Lagrangian simulations.

Key-words: scientific visualisation, hyper-volumetric data, plasma physics, compression scheme

* LIRIS, UMR CNRS-UCBL 5205, Lyon, France
† LSIIT-IGG, UMR CNRS-ULP 7005, Strasbourg, France
‡ INRIA, LORIA, ISA, Nancy, France

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 —Télécopie : +33 3 83 27 83 19

Visualisation de données hyper-volumétiques de plasmas

Résumé : De nombreuses techniques ont été proposées permettant d'explorer des données hyper-volumétriques. Mais la plus part d'entre-elles se révèlent inefficaces quand le volume de données est important. Ce rapport de recherche présente une nouvelle technique interactive de visualisation dédiée à l'exploitation en temps réel de larges volumes de données 4-D+t (avec plus de 16 Go de données par pas de temps). Nous introduisons une nouvelle méthode out-of-core visant à généraliser aux hyper-volumes, les briques techniques de base, actuellement usuelles pour des grandes masses de données volumétriques (*e. g.* 3D). Pour un pas de temps donné, un espace 4-D est visualisé en entier par tranches 2-D, chaque tranche représentant une coupe de l'espace 4-D. La contribution de ce travail réside dans l'introduction d'un schéma efficace de partitionnement, appelé hyper-brique, de manière à couvrir tous les goulets d'étranglements du matériel (chargement des données depuis le disque, temps de décompression du CPU, temps d'affichage de la GPU). Nous montrons que notre technique, utilisant un cache système et une représentation par niveaux de détails, permet d'explorer un large volume de données en temps réel. Notre technique est notamment utilisée pour étudier des données issues d'une simulation semi-lagrangienne de plasmas.

Mots-clés : visualisation scientifique, données hyper-volumétriques, physiques des plasmas, schéma de compression

1 Introduction

A large number of computer simulations generate multidimensional results and consequently need the analysis of multidimensional values. Since visualization constitutes a powerful data exploration tool, much work in information and scientific visualization has already been performed in this way. But multidimensional datasets can be very hard to categorize, according to the simulation context and parameters. In fact, different kinds of multidimensional visualization methods exist depending on the number of dimensions involved, the sampling rates and the size of the datasets.

In this report, we focus on the visualization of scalar functions of several variables evolving with time. These functions are defined by $y = f(x_1, ..., x_n, t)$, where x_i is a variable of the i-th dimension. More specifically, we consider the case of $n = 4$, that is, f is a function going to the space $\mathbb{R}^4 + t$ to the space $\mathbb{R} + t$. For each time step t, a densely sampled discretization of f in the 4-D space has to be visualized (at least $64 \times 64 \times 64 \times 64$). In addition, many time steps are involved (about a hundred). As a consequence, very large datasets (more than 20GB) have to be processed, which provides the motivation behind this out-of-core approach.

Before describing our method, let us first introduce the physical context of this application, since this is what has driven our main motivations and subsequent choices. The present work deals with the visualization of a simulation computing the evolution of a plasma over time [29]. Plasma can be considered as the fourth state of matter, which appears at certain condition of temperature and pressure (typically 10^4K or more). These conditions can be attained in various facilities and in particular in tokamak reactors and particle beam accelerators. To characterize plasma behavior, a kinetic description is used, governed by the Vlasov equation further coupled with Poisson or Maxwell equations:

$$\frac{\partial \vec{f}}{\partial t} + \vec{v} . \frac{\partial f}{\partial \vec{x}} + \frac{q}{m} \left(\vec{E} + \frac{\vec{v} \wedge \vec{B}}{c} \right) . \frac{\partial f}{\partial \vec{v}} = 0. \tag{1}$$

Equation (1) characterizes the evolution of particles distribution in time and phase space according to the electrical and magnetic fields $\vec{E}$ and $\vec{B}$, where the distribution function $f(\vec{x}, \vec{v}, t)$ represents the particle (mass m and charge q) densities at a time step t for a position $(\vec{x}, \vec{v})$ in phase space.

Two major numerical schemes exist to compute an approximate solution of this equation. The first one, called "Particle In Cell" (PIC), follows a large number of discrete particles ($\simeq 10^9$) in 6-D phase space. Some recent work has been proposed to visualize the large amount of generated time-space particles [21, 3].

The second scheme, used by the Vlasov solver described in [29], is based on a semi-Lagrangian method [13]. It approximates the particle distribution on a full discretization of the phase space. Consequently, at each time step this kind of simulation generates a multi-dimensioned regular grid, which contains the value of the distribution function for a given discrete position and velocity on each cell. Due to the tremendous amount of data, numerical semi-Lagrangian plasma simulations are generally not performed in full 6-D phase space but only in 4-D space using some symmetric properties and approximations. But even in 4-D, the amount of data remains very large. Regular grids of resolution 64 already require 128MB in double floating point precision for each time step.

In order to explore at real-time framerates plasma simulations involving more than a hundred time steps, we have to face two major problems: multi-dimensional visualization and data compression. Despite a lot of work in both fields, their adequate linking still remains a highly challenging task, especially to meet real-time visualization constraints. In addition, due to the continuous increase of temporal data produced by simulations, this task likewise becomes more and more of immediate interest in scientific visualization. In this report, we propose an original out-of-core technique to tackle the problem of visualizing very large regular four-dimensional scalar fields evolving over time. This problem is particularly difficult since recent research activities show that it is already challenging to be able to visualize three-dimensional scalar fields evolving over time. In this report, our main contribution is to present new ways in which the data can be represented, compressed and visualized (that is, the way that all of these tasks are linked together).

The report is organised as follows. Section 2 presents some previous work on multidimensional data visualization and on data compression. Section 3 describes our visualization method for 4-D+t scalar data resulting

from plasma simulations. Section 4 deals with our compression method adapted to the visualization technique. Section 5 summarizes the implemented algorithm. Section 6 presents the results obtained from different experiments together with comments on them. Finally some concluding remarks are made and some future work is given in Section 7.

2 Previous Work

In order to reach interactive visualization framerates using an out-of-core approach, it is important to build an efficient compression scheme that limits both the time spent to load the data from hard disk and the time spent for decompression.

Run Length Encoding [14], Huffman codes [15] and dictionary methods [30] are **lossless compression algorithms**, *i. e.* initial datasets are exactly recovered after decompression. But the main drawback for these kinds of methods is the low compression ratio obtained for scalar values.

On the other hand, **lossy compression algorithms** can reach very high compression ratios. Among all the existing methods, we may distinguish two major classes of approaches:

- **Scalar quantization** reduces the data size by clustering some of the data together. It consists in building several intervals which cover data boundaries. Vector quantization [20, 5, 4] is based on such a principle, but data blocks and distances between these blocks are considered instead of single values and intervals.
- **Wavelet transforms** [7, 8, 22, 25] consist in projecting the data on a wavelet basis. Data are considered as a discrete function that is expressed as a linear combination of wavelet functions which form a basis of L_2. This procedure transforms the initial data into a set of wavelet coefficients, in a way that lots of them are near to zero. Then, thresholding these small coefficients according to a given ϵ_0 will compress the data.

Multidimensional visualization techniques can be subdivided into two main areas. The first one consists in database visualization, where all dimensions are represented equally as parallel coordinates [16], star coordinates [18] or using different glyphing techniques [26, 19, 2, 27]. These techniques enable a global representation of the whole database to be built by highlighting major dependencies among some variables. But this is not our purpose here. The second class of techniques performs the visualization of functions from $\mathbb{R}^m$ to $\mathbb{R}^n$ mainly by reducing dimensions. Such a dimension reduction can be done either by projecting the data along different hyperplanes [1, 6] or by extracting planes or volumes from the hypervolume. For a function from $\mathbb{R}^m$ to $\mathbb{R}$, the principle consists in affecting a value to $m-2$ (resp. $m-3$) variables, and to visualize the resulting 2-D (resp. 3-D) dataset. Worlds within worlds [12, 11, 9] and hyperslice [28] techniques represent different ways of selecting some given dimensions, affecting a value to these dimensions and visualizing the resulting subspaces. Some hybrid approaches, called "focus + context" methods provide an integrated overview of the entire multidimensional function space around a particular multidimensional focus point. A radial "focus + context" is proposed in [17], whereas a multidimensional table lens is proposed in [24].

3 4-D Data Visualization

3.1 Graphical Pipeline

At each time step t, the plasma simulation computes a 4-D scalar field ($y = f(x, y, v_x, v_y)$) from a regular grid. We store this data on a 2-D array of size $N \times N$ called D^t (or D) for "diagnostic at time t". Each component of D^t is a 2-D array of size $N \times N$ called $D^t_{u,v}$ with $(u, v) \in [1, N]$. This array will be called **subspace** of the diagnostic D^t. A component $D^t_{u,v}[i][j]$ of this subspace, with $(i, j) \in [1, N]$, represents the **particles density** at a (u, v) position and a (i, j) velocity for a given time t. The average density, $\overline{D_{u,v}}$, on the spatial position (u, v) is given by:

$$\overline{D_{u,v}} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} D_{u,v}[i][j].$$

Figure 1 depicts this data structure.

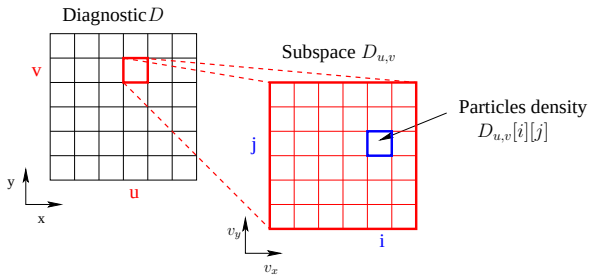

Figure 1: Data structure for a diagnostic D.

A "focus and context" approach combined with a "worlds within worlds" method is used to visualize this 4-D scalar field. The "focus" is materialized by a **lens**, which can be interactively shifted by the user. This lens L_{u_L,v_L,S_L} corresponds to a set of subspaces of D defined by:

$$L_{u_L,v_L,S_L} = \{D_{u_L+k,v_L+l}\}$$

$$\text{with } (k,l) \in [1,S_L] \text{ and } (u_L,v_L) \in [1,N-S_L+1].$$

S_L corresponds to the size of the lens and (u_L,v_L) to its position (left up corner) in the diagnostic D. At a given time t, the diagnostic D is visualized in 3-D as depicted on figure 2.

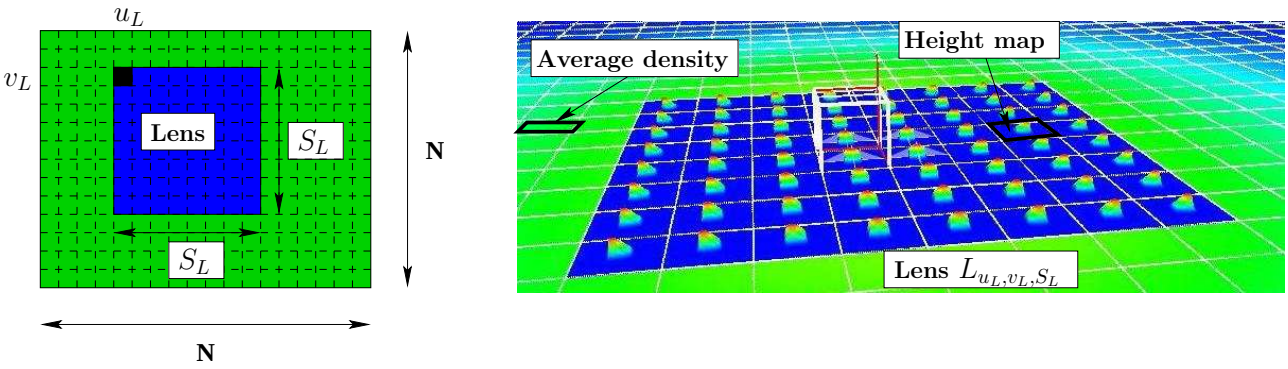

Figure 2: 3-D visualization of the diagnostic D using a rectangular lens L_{u_L,v_L,S_L} of size $S_L \times S_L$ placed inside D on (u_L,v_L) with $S_L = 8$.

- Each subspace $D_{u,v}$ exterior to the lens *i. e.* such that $D_{u,v} \notin L_{u_L,v_L,S_L}$, is displayed by a square "quad" with color $c_{u,v}$ defined by $c_{u,v} = F\left(\overline{D_{u,v}}\right)$ where F is a **transfer function** relating each scalar to a (R,G,B) color.
- Inside the lens *i. e.* such that $D_{u,v} \in L_{u_L,v_L,S_L}$, a full display of the subspace $D_{u,v}$ is made with a height map colored using the same transfer function F.

Figure 3 depicts the graphical pipeline of our 4-D visualization technique. When a time step t is selected by the user, the diagnostic D is visualized as described above. Firstly, a part of the compressed information concerning the diagnostic D is loaded from the hard disk. This information is then decompressed in memory and stays there as long as the time step t is not modified. Secondly, according to the (u_L,v_L) position of the lens, another part of the information, concerning only the lens, is loaded. This information is decompressed only on demand *i. e.* whenever we have to visualize the corresponding height map.

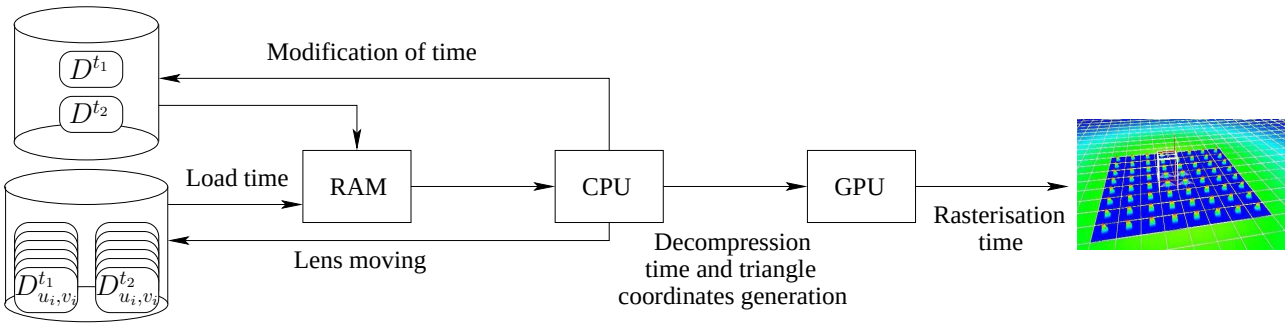

Figure 3: Pipeline of the 4-D+t "out-of-core" visualization system.

The number of generated triangles N_T for displaying the height maps can be very important, especially if a brute force approach is used, that is, each pixel of each subspace $D_{u,v}$ is displayed using a pair of triangles. To avoid sending too many triangles to the GPU, we propose to adapt the resolution of the $D_{u,v}$ subspaces according to their positions inside the lens as well as to the position of the observer. The decompression scheme implemented allows us to control precisely this resolution.

Figure 3 illustrates the three main bottlenecks of this system: (1) time spent to load files depending on the disk bandwidth and the size of the files (the latter also depends on the compression ratio), (2) decompression time depending on the processor and the decompression method and (3) time spent to display the height fields depending on the number of triangles. These three different timings added together have to fullfill the real-time constraint.

3.2 Generation of Height Map Triangles

As shown on Figure 2, the lens L determines the part of the diagnostic visualized using a full 3-D display *i. e.* with a height map.

We note that in the past, a large number of visualization methods have been proposed to efficiently display height maps such as for example ROAMing [10] (Real-time Optimally Adapting Meshes). These techniques adapt the number and sizes of the triangles according to the position of the observer as well as the underlying height function mainly by computing errors. Unfortunately, this error computation is usually costly. In our case, the decompression scheme already stresses nearly all the CPU resources. For this reason, we could not use straightforwardly existing algorithms.

To solve the problem, we propose to control the visualization resolution of the height maps directly by exploiting our compression scheme. This compression, based on a Haar wavelets transform, enables the subspaces $D_{u,v}$ to be formulated with a hierarchical structure. We define by $H^m[i][j]$ the height map for a given resolution level m. $H^m[i][j]$ is directly obtained using the inverse Haar transform, which is simply stopped at level m (see next section).

The resolution level m is fixed empirically according to the position of the observer and according to the position of the subspace $D_{u,v}$ inside the lens. The more the subspace is close to the center of the lens (on the center we use the highest resolution), the greater is the resolution since this mostly attracts the attention of the user.

4 4-D+t Data Compression

Our compression scheme certifies an important compression rate with a high precision. Moreover, this scheme is totally adapted to the hierarchical visualization method that we previously described. Indeed, a $2-D+2-D$ scheme, as proposed in this report, allows us to only load the necessary data in real-time. The scheme is applied independently on each diagnostic D at each time step t so temporal coherency is not considered.

For each diagnostic D, the compression is carried out at two levels:

- At a **local level** *i. e.* for each subspace $D_{u,v}$. Indeed, each subspace corresponds to a 2-D array which can be compressed with the usual techniques. We chose the Haar wavelet transform because of its hierarchical formulation, and its small computational cost.
- At a **global level**, in order to take into account the consistency of the $N \times N$ subspaces of the same diagnostic D.

4.1 Haar Wavelet Compression of Subspaces

Here, we do not precisely describe the algorithm of the well-known Haar wavelet compression [23], but only some main steps explaining our own implementation. Each subspace $D_{u,v}$ (2-D array) is iteratively re-written line by line and column by column as differences and averages. Consequently, as shown in Figure 4, we obtain for a given subspace $D_{u,v}$ a new 2-D array of size $N \times N$ called Haar wavelet transform $\mathcal{H}(D_{u,v})$, where the left upper corner corresponds to the average of the whole original array.

By applying the inverse transformation, the subspace $D_{u,v}$ can be rebuilt from $\mathcal{H}(D_{u,v})$. Moreover, the hierarchical Haar representation enables the subspace to be rebuilt only to a desired level m. The highest hierarchy level corresponds to a $N \times N$ resolution. The decompression time depends on the chosen level m.

Due to the fact that the $\mathcal{H}(D_{u,v})$ values result from difference computations, they are generally close to zero. The compression consists then in choosing a threshold ϵ_0 and to clamp to zero the values of $\mathcal{H}(D_{u,v})$ lower than ϵ_0. We call $\mathcal{H}'(D_{u,v})$ the resulting sparse structure (see Figure 4).

$\mathcal{H}'(D_{u,v})$ is stored as a couple denoted $(B_{u,v}[N][N], T_{u,v}[N_{u,v}])$, where $B_{u,v}$ is a bit mask identifying the zero and non-zero values of $\mathcal{H}'(D_{u,v})$, and $T_{u,v}$ is a linear array containing only non-zero values of $\mathcal{H}'(D_{u,v})$. For any subspace $D_{u,v}$ we need $N^2/8$ bytes to store $B_{u,v}$ while the size $N_{u,v}$ of $T_{u,v}$ depends on the threshold ϵ_0. Figure 4 summarizes this compression scheme for a subspace of size $N = 4$.

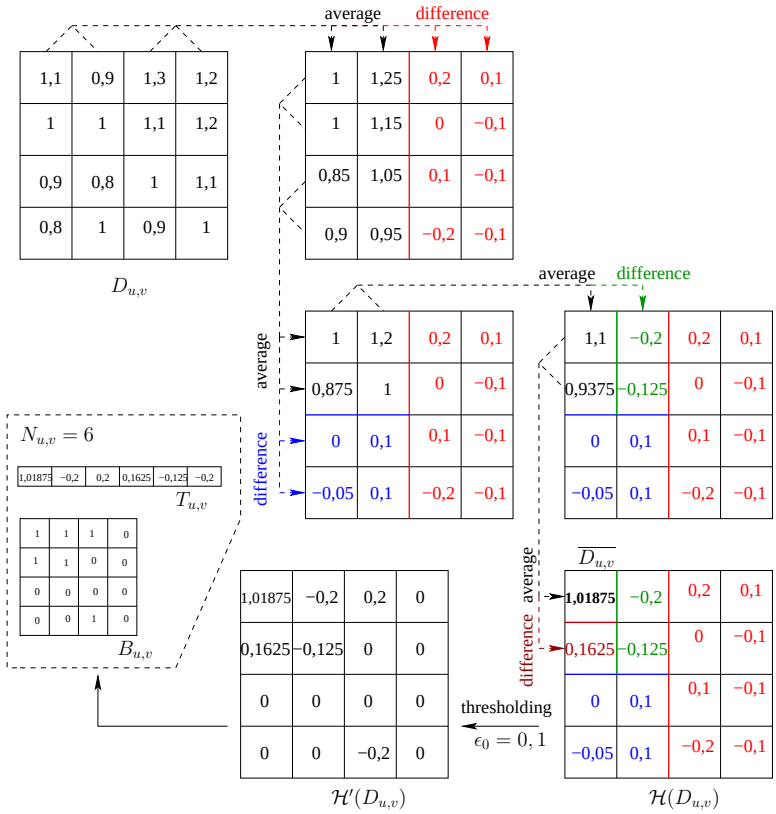

Figure 4: Compression example for a subpace of size $N = 4$.

4.2 Total Compression of a Diagnostic

Once the Haar wavelet compression algorithm has been applied, all subspaces $D_{u,v}$ are represented by a boolean matrix $B_{u,v}$ associated to an array $T_{u,v}$ containing only the non-zero values of $\mathcal{H}'(D_{u,v})$.

This first level of compression does not take into account the potential consistency among the different subspaces of the same diagnostic D. For this reason, a second compression scheme is applied at two levels:

1. **Creating a unified dictionary** by combining all the $T_{u,v}$ arrays according to a threshold ϵ_1. Consequently, we obtain a single array T_D for all the subspaces of a given diagnostic.
2. **Compression of the T_D dictionary** using an adaptive scalar quantization algorithm with a threshold ϵ_2.

To blend the arrays $T_{u,v}$, we first sort them, then we combine them by removing identical values according to ϵ_1. We obtain a single array T_D of size N_{T_D} sorted by decreasing order *i. e.* $\forall k, T_D[k] > T_D[k+1]$. Moreover, due to the ϵ_1 accuracy used for the blending, we have $\forall k, T_D[k] \geq T_D[k+1] + \epsilon_1$.

By blending the arrays $T_{u,v}$, we lose the connections among the matrices $B_{u,v}$ and the components of $T_{u,v}$. Consequently, we have to build a table $I_{u,v}$ which allows us to restore this connection. The components of $I_{u,v}$ are defined so that $T_D[I_{u,v}[k]]$ is equal to $T_{u,v}[k]$ plus or minus ϵ_1. So, the size of the array $I_{u,v}$ depends on $N_{u,v}$. Practically, the integers of $I_{u,v}$ are encoded on 24 bits which we found experimentally to be enough. Indeed, this allows for $2^{24} = 16\,777\,216$ different entries in T_D.

To summarize, at this stage of the compression scheme, a subspace $D_{u,v}$ is now defined as a pair $(B_{u,v}, I_{u,v})$ associated to a dictionary T_D common to all subspaces of the same diagnostic D.

Next, the **compression of the common dictionary** T_D is performed. Our new compression scheme takes advantage of the non uniformity of the values in T_D by performing an adaptative scalar quantization with an ϵ_2 error on L intervals. A linear course of T_D can be used to build the different intervals of size $d = M \times \epsilon_2$ where M is the sampling rate at each interval (see Figure 5).

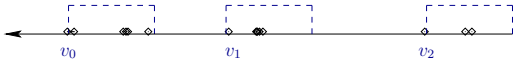

Figure 5: Quantization of the common dictionary.

The quantization is then performed on each interval $I_l = [v_l - M \times \epsilon_2; v_l]$, by turning a real value $T_D[k]$ into an $m \in [1, M]$ integer defined as :

$$v_l - m \times \epsilon_2 \geq T_D[k] \geq v_l - (m+1) \times \epsilon_2.$$

Instead of storing N_{T_D} double precision values, we now store for each interval:

- Its index l ($Log_2(L)$ bits).
- Its right bound value v_l (64 bits).
- The index m ($Log_2(M)$ bits) for each value within the interval.

The number of generated intervals now hardly depends on the non-uniformity of the data.

5 Summary of the Method

Figure 6 presents an overview of our method and table 1 sums up the symbols used.

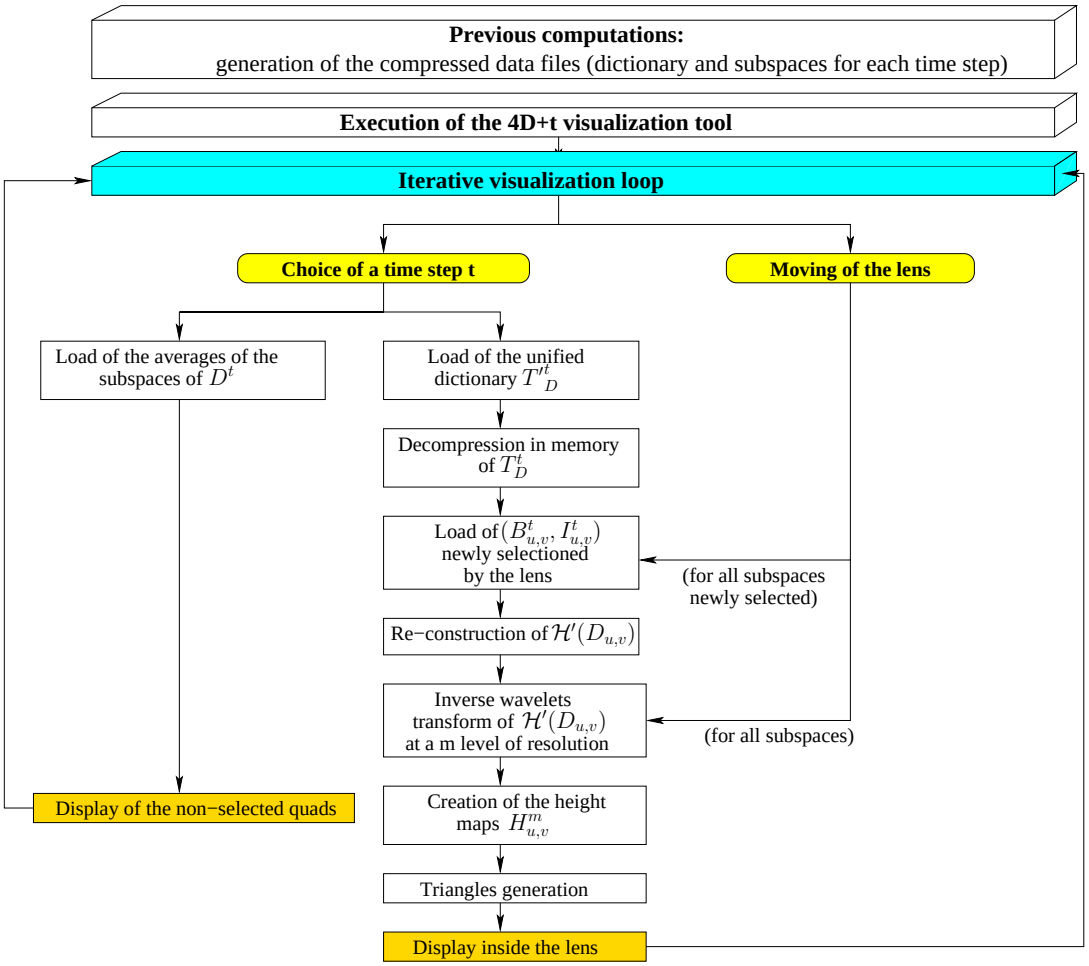

Figure 6: Draft of the implemented method.

First of all, some pre-computations are made only once for a given simulation in order to perform the compression of the 4-D+t data. The result is stored into several different files. Note that all diagnostics D^t are considered independently. At this stage, we have for each diagnostic D^t:

- A unified dictionary T^t_D stored in a file in a compressed form.
- A set of couples $(B^t_{u,v}, I^t_{u,v})$ computed for each subspace $D_{u,v}$ of the diagnostic D. These couples are stored in several files.

In the case where N is huge, and to avoid the creation of too large a number of files, the subspaces $(B^t_{u,v}, I^t_{u,v})$ are clustered into the same file by blocks of size 4×4.

After the pre-computations, the user can start the visualization application. Firstly, the system loads the compressed dictionary T'_D of the desired diagnostic and decompresses it to obtain the unified dictionary T_D. This dictionary will stay in memory unless the time step is changed. Moreover, averages of all subspaces, which have been stored into a single file, are also loaded to perform the display outside the lens.

Symbols	Significations
n	Space dimension (in our case n=2)
N	Discretisation resolution
t	Time
$\vec{x}$	Positions vectory
$\vec{v}$	Velocities vectory
$f(\vec{v},\vec{x},t)$	Distribution function of the charged particles
(u,v)	Coordinates on grid of the physical space (x,y)
(i,j)	Coordinates on grid of the velocity space (v_x,v_y)
D^t or D	Diagnostic at time t
$D_{u,v}$	Subspace of the diagnostic D
$D_{u,v}[i][j]$	Component of the subspace $D_{u,v}$
$\overline{D_{u,v}}$	Average density of $D_{u,v}$
L_{u_L,v_L,S_L}	Lens
S_L	Size of the lens
(u_L,v_L)	Position of the lens inside D
$c_{u,v}$	Color of the square "quad" (u,v)
F	Transfer function
N_T	Number of generated triangles
m	Resolution level
$H^m[i][j]$	Height map of $D_{u,v}[i][j]$
$\mathcal{H}(D_{u,v})$	Haar wavelets transform of $D_{u,v}$
ϵ_0	Threshold associated to the Haar algorithm
$\mathcal{H}'(D_{u,v})$	Result of the $\mathcal{H}(D_{u,v})$ thresholding
$N_{u,v}$	Number of the non-zero values of $\mathcal{H}'(D_{u,v})$
$T_{u,v}$	Array of the non-zero values of $\mathcal{H}'(D_{u,v})$
$B_{u,v}$	Bit mask of $\mathcal{H}'(D_{u,v})$
ϵ_1	Threshold associated to the blending
T_D	Unified dictionary of D
N_{T_D}	Size of the unified dictionary T_D
$I_{u,v}$	Index table to the unified dictionary T_D
ϵ_2	Threshold associated to the compression of T_D
L	Number of partitions of the unified dictionary T_D
v_l	Borders of the partitions of the unified dictionary
m	Subsystems of the unified dictionary T_D
T'_D	Compressed unified dictionary of D
ϵ	Error of the total compression scheme

Table 1: Overview of the used symbols.

Secondly, only files containing the subspaces selected by the lens are loaded. Therefore it was important to use a $2-D+2-D$ compression technique and not a full 4-D one to only load the necessary information for the lens rendering. Note that, because of clustering, we may load some more subspaces than the number of subspaces actually covered by the lens, but these additional subspaces are used as cache. For all loaded subspaces, the couples $(B^t_{u,v}, I^t_{u,v})$ are used with the unified dictionary T_D to build the Haar wavelet transforms $\mathcal{H}'(D_{u,v})$. They correspond to 2-D arrays of real values of size $N \times N$. This information stays in memory until the lens is moved.

The third part concerns visualization. First, "quads" outside of the lens are displayed by using average values. Next, only subspaces covered by the lens are decompressed by using the inverse Haar transform for a given level m. This level depends on the position of the observer and the position of the given subspace inside the lens. This stage allows the creation of the height maps $H^m_{u,v}$ for different resolutions m. Then, these height maps are used to generate triangles that are sent to the graphics card.

Whenever the lens is moved, new blocks of subspaces can be loaded and decompressed. Since the lens can only be moved step by step (or subspace by subspace), a cache system allows us to re-use already decompressed subspaces.

When the time step is changed, all the methods have to be re-iterated. In this case the performance is mostly stressed. A correct choice of the size of the lens is important to maintain real time performances even when we change the visualized plasma diagnostics.

Note that we are not limited by the number of time steps but only by the capacity of the hard disk. The global performance depends on the simulation size N, the compression ratio, the size of the lens and the number of subspaces clustered in individual files.

6 Results

6.1 Visualization Method

In the previous sections, we described the data structure of the diagnostic D^t as an array of subspaces $D^t_{u,v}$, where (u, v) represented the Euclidean space (x, y). But it is naturally also possible to describe it using other integration spaces for (u, v), namely (x, y), (v_x, v_y), (x, v_x) or (y, v_y).

Figure 7 presents visualization results obtained for a same simulation, but using these four different integration spaces. In our system, the user can switch interactively between these four spaces. Figure 7 shows results without a lens, that is, without subspace visualization. This is the technique that physicists commonly use for analyzing plasma data.

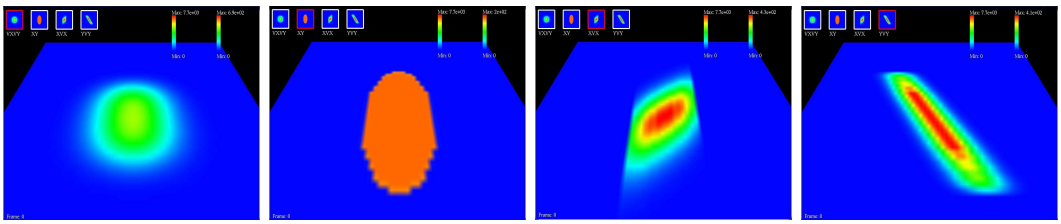

Figure 7: Visualization of a diagnostic on different integration spaces, respectively (v_x, v_y), (x, y), (x, v_x) and (y, v_y).

Figure 8 illustrates a complete visualization using the lens and the subspaces. Tag (1) presents the interval pointed to the lens in the initial integration space. This lens allows for a visualization of particles density variation (tags (3) and (4)) that would not be visible without the lens (tag (5)). It thus enables users to better understand the plasma evolution during the simulation, since it shows the full 4-D data.

Figure 9 presents a tool added to the application to pick up individual values of density (tag (2)) for a given point (tag (1)) within the 4-D data. The threshold of our compression scheme allows us to give a value with very low error. Moreover, the average of density is also given for the complete subspace (tag (3)).

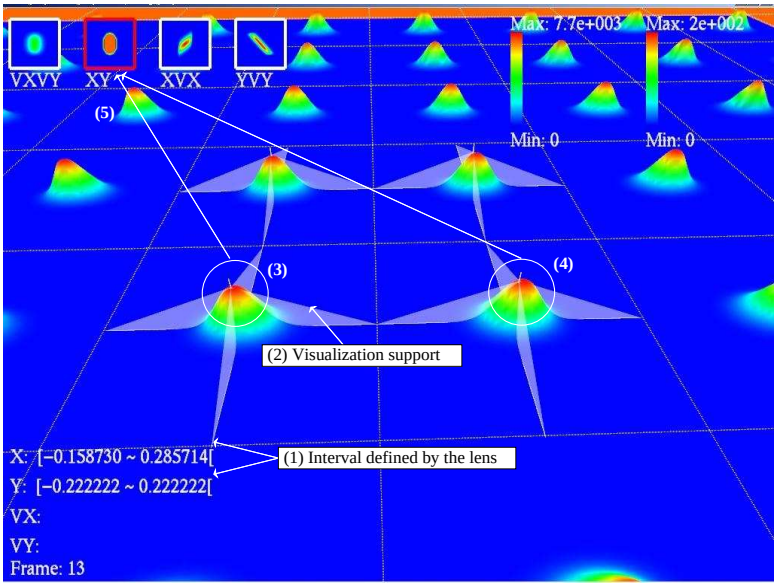

Figure 8: Visualization of the application with the lens.

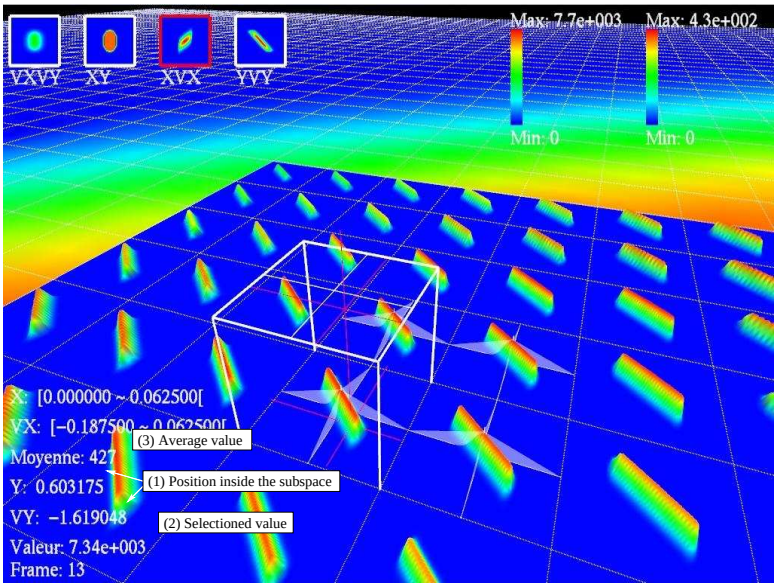

Figure 9: The lens and the pointing tool.

6.2 Compression Method

The results presented in this section are obtained on a low-end PC with an AMD Athlon XP 3000+processor with 1GB 300Mhz Dual Channel DDR and a Hitachi 160GB SATA hard disk with an 8MB cache and a ATI 9700Pro AGP 8x graphic board.

The performances are evaluated for four different physical cases, and for each one, the four possible projections are computed. But since compression results are almost similar for all of these four simulations, we only present results for the integration space (x, y) of one of these simulations. The different simulations have been run on 4-D phase space grids of size 64^4 over 40 time steps, which represents 128 MB for a single diagnostic. To simplify the parameterization of the compression, we always fix the three thresholds to the same value ($\epsilon_0 = \epsilon_1 = \epsilon_2 = \epsilon$). Experimentally, we observe that the resulting absolute error remains less than the ϵ we fixed for the compression.

The pre-processing step takes about $2-3$s to compress a single diagnostic for a particular projection, so the whole simulation compression can be computed in less than 6min for the four different projections.

In Figure 10 a compression ratio greater than 90% is obtained even for very low error tolerances ($\epsilon = 10^{-8}$). The resulting size of a single diagnostic is between 3MB ($\epsilon = 1$) and 11MB ($\epsilon = 10^{-8}$), which enables to load between respectively 5 and 25 time steps per second according to a 50MB/s hard disk bandwidth. The time spent to decompress the unified dictionary is at most $11-12\mu s$ ($\epsilon = 10^{-8}$).

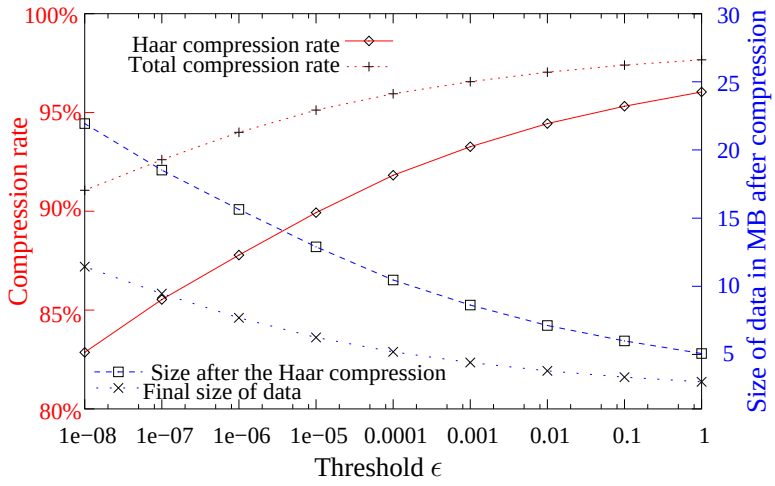

Figure 10: Compression ratio and size of a single diagnostic in MB after compression according to ϵ for the integration space (x, y).

In Figure 11 we present the time spent in the inverse Haar transform for several resolutions according to ϵ. This time, constant for the different thresholds, is between $426\mu s$ (64×64 resolution) and $2\mu s$ (4×4 resolution).

Figure 12 shows the maximal time spent for loading subspaces selected by a lens of size 8×8 and the maximal time spent to build the height maps in the case of maximal resolution ($\mathcal{H}'(D_{u,v})$ reconstruction + inverse Haar transform). The third plot (sum of the two first) corresponds to the total time spent when the lens is moved.

To conclude, once the pre-processing step is finished, our method enables us to visualize at real-time rates the 4-D data $f(x, y, v_x, v_y)$ computed by the plasma simulation. A 8×8 lens is considered for framerates evaluation. The resolution is highest (64×64) at the center, gradually decreasing towards the border (16×16). We measured the following framerates: $15-20$fps when the time step is changed, $15-20$fps when the projection space is changed and $25-34$fps when the lens is moved.

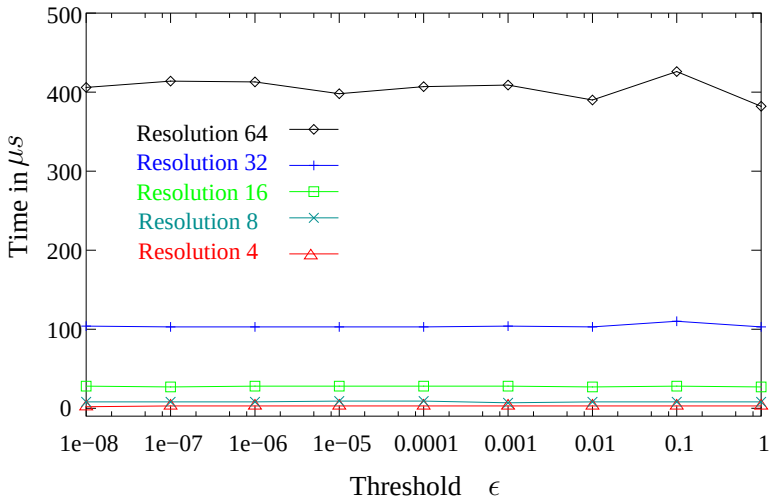

Figure 11: Time in μs spent in the inverse Haar transform for several resolutions $(64, 32, 16, 8, 4)$ according to ϵ for the integration space (x, y).

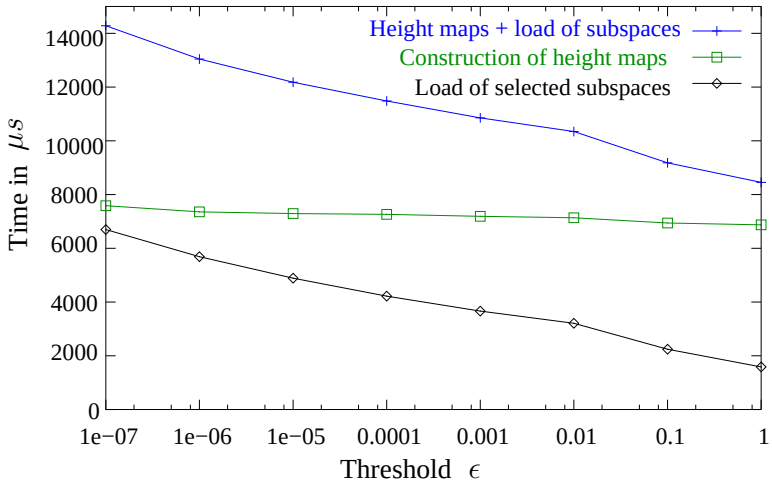

Figure 12: Maximal time (in μs) spent when the lens (8×8) is moved according to ϵ.

7 Conclusions and Future Work

This report proposes an original out-of-core technique to tackle the difficult problem of visualizing very large regular four-dimensional scalar fields evolving with time. The core contribution lies in the way the data are represented, compressed and then hierarchically visualized using a worlds within worlds technique (that is, 2-D subspaces on 2-D arrays). The method is simple and satisfies well the crucial real-time constraint in spite of the continuous load from the mass-storage device.

We based the 3-D visualization on the use of a lens, locally enabling the full display of a distribution function $f(x, y, v_x, vy)$ by means of height fields. This 3-D visualization turns out to be much more expressive than a 2-D one, which would only use 2-D textures for the subspaces instead of height maps.

We solved the major problem of dealing with very large datasets by implementing an efficient hierarchical compression scheme allowing us to only load the necessary data for our "focus + context" visualization. Moreover it keeps a high accuracy desired by plasma scientists who are usually looking for phase space regions where the the particles distribution is low. Indeed, small particle densities can have a large impact on the whole plasma behavior.

In the near future, and with the continuous increase of computational power it should be possible to compute full 6-D semi-Lagrangian plasma simulations. These will raise an even more difficult problem of storage and visualization. We are presently starting to tackle the problem of dealing with even more data, in particular using a parallelization of the current algorithm.

References

[1] Asimov. The grand tour: A tool for viewing mutlidimensional data. *SIAM the Journal of Scientific Statistical computing*, 6:128–143, 1985.

[2] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

[3] C.S. Co, A. Friedman, D.P. Grote, JL. Vay, and E. Wes Bethel. Interactive methods for exploring particle simulation data. *IEEE Visualization*, 2, 2004.

[4] C. Constantinescu and J. A. Storer. Improved techniques for single-pass adaptive vector quantization. *Proceedings of the IEEE*, 82(6):933–939, June 1994.

[5] C. Constantinescu and J. A. Storer. Online adaptive vector quantization with variable size codebook entries. *Inf. Process. Manage.*, 30(6):745–758, 1994.

[6] Dianne Cook and Andreas Buja. Manual controls for high-dimensional data projections. *Journal of Computational and Graphical Statistics*, 6(4):464–480, 1997.

[7] I. Daubechies. Orthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Maths*, 16:909–996, 1988.

[8] R. A. Devore, B. Jawerth, and B. J. Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2 (Part II)):719–746, March 1992.

[9] S. R. dos Santos and K. W. Brodlie. Visualizing and investigating multidimensional functions. In *Proceedings of the IEEE TCVG symposium on Data Visualisation 2002*, page 173. Eurographics Association, 2002.

[10] M. A. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. ROAMing terrain: real-time optimally adapting meshes. In *IEEE Visualization*, pages 81–88, 1997.

[11] S. K. Feiner and C. Beshers. Visualizing n-dimensional virtual worlds with n-vision. In *Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 37–38. ACM Press, 1990.

[12] S. K. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In Scott E. Hudson, editor, *User interface software and technology*. ACM Press, October 1990.

[13] F. Filbet, E. Sonnendrücker, and P. Bertrand. Conservative numerical schemes for the Vlasov equation. *J. Comput. Phys.*, 172(1):166–187, 2001.

[14] S.W. Golomb. Run-length encodings. *IEEE Transactions on Information Theory*, pages 399–401, 1966.

[15] D.A. Huffman. A method for construction of minimum redundancy codes. In *Proceedings of the Institute of Radio Engineers*, volume 40, pages 1098–1101, 1952.

[16] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st conference on Visualization '90*, pages 361–378. IEEE Computer Society Press, 1990.

[17] S. Jayaraman and C. North. A radial focus+context visualization for multi-dimensional functions. In *Proceedings of the conference on Visualization '02*, pages 443–450. IEEE Computer Society, 2002.

[18] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *IEEE Information Visualization, Hot Topics*, pages 4–8, 2000.

[19] H. Levkowitz. Color icons: merging color and texture perception for integrated visualization of multiple parameters. In *Proceedings of the 2nd conference on Visualization '91*, pages 164–170. IEEE Computer Society Press, 1991.

[20] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, 1:84–95, Jan. 1980.

[21] K.-L. Ma, G. Schussman, B. Wilson, K. Ko, J. Qiang, and R. Ryne. Advanced visualization technology for terascale particle accelerator simulations. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11. IEEE Computer Society Press, 2002.

[22] Y. Meyer, editor. *Wavelets and applications*, volume 20 of *RMA: Research Notes in Applied Mathematics*, Paris, 1992. Masson.

[23] C. Mulcahy. "image compression using the haar wavelet transform". *Spelman Science and Mathematics Journal*, 1(1):22–31, 1997.

[24] R. Rao and T. Tenev. Extending table lens to multidimensional data and olap operations. In *Euro-American Workshop: Visualization of Information and Data*, pages 24–25, June 1997.

[25] R. M. Rao and A. S. Bopardikar. *Wavelet Transforms: Introduction to Theory and Applications*. Addison-Wesley, 1999.

[26] S. Smith, G. Grinstein, and R. D. Bergeron. Interactive data exploration with a supercomputer. In *Proceedings of the 2nd conference on Visualization '91*, pages 248–254. IEEE Computer Society Press, 1991.

[27] H. Theisel and M. Kreuseler. An enhanced spring model for information visualization. In N. Ferreira and M. Göbel, editors, *Eurographics'98*. Blackwell, 1998.

[28] J. J. van Wijk and R. van Liere. Hyperslice: visualization of scalar functions of many variables. In *Proceedings of the 4th conference on Visualization '93*, pages 119–125, 1993.

[29] E. Violard and F. Filbet. Parallelization of a vlasov solver by communication overlapping. In *proceedings of PDPTA'02*, pages 1049–1055. CSREA Press, June 2002.

[30] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399