Clustering Revealed in High-Resolution Simulations and Visualization of Multi-Resolution Features in Fluid-Particle Models

Krzysztof Boryczko, Witold Dzwinel,
AGH Institute of Computer Science, al. Mickiewicza 30, 30-059, Kraków, Poland

David A. Yuen
Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, Minnesota
MN 55455, USA

Abstract

Simulating natural phenomena at greater accuracy results in an explosive growth of data. Large-scale simulations with particles currently involve ensembles consisting of between $10^6$-10$^9$ particles, which cover $10^2$-10$^6$ timesteps. Thus, the data files produced in a single run can reach from tens of gigabytes to hundreds of terabytes. This data bank allows one to reconstruct the spatio-temporal evolution of both the particle system in whole and each particle separately. Realistically, for one to look at large data set at full resolution at all times is not possible and, in fact, not necessary. We have developed here an agglomerative clustering technique, based on the concept of mutual nearest neighborhood (MNN). This procedure can be easily adapted for efficient visualization of extremely large data sets from simulations with particles at various resolution levels. We present the parallel algorithm for MNN clustering and its timings on the IBM SP and SGI/Origin 3800 multiprocessor systems for up to 16 million of fluid particles. The high efficiency obtained is mainly due to the similarity in the algorithmic structure of MNN clustering and particle methods. We show various examples drawn from MNN application in visualization and analysis on the order of a few hundred gigabytes of data from discrete particle simulations, using dissipative particle dynamics and fluid particle model. Because data clustering is the first step in this concept extraction procedure, we may employ this clustering procedure to many other fields such as data mining, earthquake events and stellar populations in nebula clusters.

Keywords: large-scale data sets, visualization, feature extraction, parallel clustering, dissipative particle dynamics, fluid particle model

1 corresponding Author. Minnesota Supercomputing Institute, University of Minnesota, USA e-mail: davey@krissy.geo.umn.edu
Introduction

Visualization of multi-resolution structures occurring in large-scale simulations or experimental setups of natural phenomena involves invariably the processing of large-scale data sets with complex topologies in many spatio-temporal scales. For example, in geophysical fluid dynamics, dynamical processes are characterized by a strong variability over a wide range of spatial and temporal scales from nanoseconds and nanometers to million of years and kilometers [1]. We do not need to look at the data at full resolution all the time. In fact, looking at the full data set is sometimes not a viable option. Instead we really need techniques to view and interrogate data at various resolution levels, trading off resolution for interactivity or for focusing detailed attention on the scale of interest. In feature extraction of structured data, which can be represented by continuous fields with smooth derivatives, image processing techniques and filters, such as wavelets [2], can be employed. In medical imaging and computer graphics where data may originate from fluid and other materials which may be partially transparent and should be modeled as such, volume rendering that uses ray casting and ray tracing is currently in popular usage. These techniques, however, are not appropriate for discrete systems which are composed of various types of particles producing complex microstructures.

Simulations involving discrete particles are used for studying phenomena in microscopic and mesoscopic scales, where atomic interactions, thermal fluctuations and excluded volume effects go with large-scale features. Particle methods involve broad range of models [3-10] such as: molecular dynamics (MD), dissipative particle dynamics (DPD), fluid particle model (FPM), granular dynamics (GD), direct simulation Monte-Carlo (DSMC), particle-in-cell (PIC), smoothed particle hydrodynamics (SPH) and many other techniques in numerical simulations. Particles - which can represent atoms, molecules, "lumps" of fluid, grains of solids, micelles, moving grid nodes, rocks, faults, galaxies or abstract objects [11,12] - interact one with another, and evolve according to the Newtonian mechanics or other prescribed local rules such as cellular automata [13], as it is developed in the lattice Boltzmann gas (LBG) method [9,10]. The spatio-temporal scale of particle system depends on the particle definition and the number of particles in the system. The large-scale MD simulations in 3-D performed on current computers employ $10^6$-$10^9$ particles [3,4,14], depending on the range and the complexity of interactions. For simulations involving $10^6$ particles and covering $10^4$-$10^5$ timesteps, the data-set volume produced in a single run is about 450 GBytes - 4.5 TBytes, if the data were saved at each timestep. Non-linear interactions between particles resulting from conservative, dissipative, depletion forces and hydrodynamic interactions [15] result in clustering, which produces variety of multi-resolutional aggregates such as molecules, micelles, crystals, colloidal aggregates [16], polymeric chains and dispersion microstructures [17]. Extracting these clusters from the whole ensemble is very important for studying microstructural properties of complex and porous materials.

The spatio-temporal hierarchy of "particles" and microstructures, which represents multi-resolutional character of matter, cannot be visualized employing classical techniques used in computer graphics e.g., volume rendering and ray casting algorithms. The large amount of unstructured data merging in different way in various levels of this hierarchy makes these time consuming techniques useless. Moreover, the particle features, which can be represented as N-dimensional vector, yield complex clustering patterns in higher dimensions, in which $N^{>>10}$, inaccessible for volume rendering techniques. Unlike continuum systems, in which the ultimate resolution can be achieved by increasing the number of grid points, the particles representation can display the finest resolution level [11]. They are singular entities and global features of the particle ensemble are unveiled after averaging. However, to obtain a single grid point of, e.g., velocity field, the velocities of at least 100 to 1000 particles should be averaged. Even for $10^9$ particles ensemble, the resulting resolution after averaging may appear too crude for extracting finer features, such as those found along grain-boundaries. Moreover, we will otherwise lose valuable information concerning the clustering of particles.
The large amount of data involves running the clustering procedure on high performance parallel architectures. For example, the Landsat data (about one terabyte per day - represented by 4 or 256-dimensional vectors) are analyzed by using efficient, parallel, continuous k-Means algorithm developed in Los Alamos and run on the top high-performance systems [18]. For clustering of large spatial data bases and image segmentation a number of workstations connected via Ethernet is also used [19]. Large-scale parallel clustering is performed in NOW (Network of Workstations) environment by using client-server model, in which the clustering task is divided among a set of clients that report their intermediate results to a single server process [20]. This parallel software is an ideal candidate for many GRID computing projects dealing with processing and concept extraction from large amount of data.

For the analysis of data from large-scale simulations with particles, the non-hierarchical algorithms cited above are inadequate due to the lack of an a priori knowledge about the number of clusters, high computational complexity of the algorithms and serious problems with avoiding local minima in minimization of the cost function. Due to the local nature of the particle microstructures the hierarchical agglomerative techniques fit better for their extraction. As shown in [21], optimal butterfly and tree parallel algorithms exist for hierarchical clustering for various proximity measures between clusters. The efficiency of agglomerative clustering can be greatly improved by using fast procedures for nearest neighbors (NN) search. For example, the Friedman's projection method [22] can be used in extracting high-dimensional clusters.

In this paper we present an efficient parallel algorithm devoted to the agglomerative clustering, which is based on the mutual nearest neighbor (MNN) concept [23] and linked-list method [24] for NN search. First, we discuss the concept of agglomerative clustering and we show its homogeneity with the particle model. This clustering scheme will be employed to conduct clustering of large-scale data sets, which consist of up to 16 million particles. Then, the MNN parallel algorithm is introduced and timings from IBM SP and SGI-Origin 3800 parallel systems are compared. We show the examples of application of MNN clustering in extracting features from simulations of complex fluids. Finally, to sum up, we discuss the conclusions and implications of this novel procedure.

**Serial clustering algorithm**

Clustering is a fundamental concept in pattern recognition [25-27] but also has many applications in fields such as earthquake physics, astrophysics and polymer fluid dynamics [30-35]. Clustering is basically a threshold phenomenon, which occurs in states far enough from equilibrium [16]. Clustering is used for classifying similar (or dissimilar) patterns represented by N-dimensional vectors. Depending on the data structures, different clustering schemes must be used. There are two principal approaches for classifying data. The first one consists in non-hierarchical extraction of clusters. This approach is used mainly for extracting compact clusters by using global knowledge of the data structure. The most known and simplest techniques base on K-means algorithm [18,25-27,36]. The main problem with K-means is that the final classification represents the local minimum of criterion function. Thus for multi-modal functions, starting from different initial iterations, one can obtain different cluster structures. Because the global information about the data is required for the non-hierarchical schemes, the non-hierarchical algorithms suffer also from a high computational complexity and most of them require an a priori knowledge about the number of clusters. In Fig.1 we show how the K-means algorithm searches for compact clusters with regular density. As shown in Fig.1a, it does not work for elongated, bridged clusters and clusters with a different density appearing in simulations with particles [23].

Agglomerative clustering schemes [25-27] consist in the subsequent merging of smaller clusters into the larger clusters, basing on a proximity criterion. Depending on the definition of the proximity measure, there exist many agglomerative schemes such as: average link, complete link, centroid, median and minimum variance algorithm. The hierarchical schemes are very fast...
for extracting localized clusters with non-spherical shapes. All of them suffer from the problem of not having properly defined control parameters, which can be matched for the data of interest and hence can be regarded as invariants for other similar data structures.

**Fig.1** A schematic drawing showing the results of K-means and MNN clustering on the same 2-dimensional synthetic data set.

**Fig.2** The results of K-means and MNN clustering on the same 256-dimensional data set representing vibration modes of the IBR-2 (Russia-Dubna) nuclear reactor. The non-linear Sammon’s mapping [37-38] from 256 into 3-dimensional space was employed for dimensionality reduction. Discrimination surfaces are shown for both cases. Visualization made by using the visualization package Amira 2.3 [43].

In the MNN algorithm proposed by Gowda and Krishna in [23] invented almost 25 years ago, the proximity measure between merging clusters is constructed on the basis of two types of distances: mnn-distance and the classical Euclidean metrics. The mnn-distance comes from mutual nearest neighbors (MNN) concept which can be outlined as follows.

1. Let us consider an ensemble consisting of $Np$ particles and $i=1,2,...,Np$.
2. Find the lists $L_i$ of $Ncut$ nearest neighbors $j$ in $RCN$ radius for each particle $i$. 


3. Sort out the neighbors \( j \) in the lists in ascending order according to the Euclidean distance between \( i \) and \( j \) particles. Thus \( L_i(k) = j \), where \( k \) is the position of the particle \( j \) in the \( i \)'th list and \( L_j(m) = i \), where \( m \) is the position of the particle \( i \) in the \( j \)'th list.

4. Because \( L_i(k) = j \) and \( L_j(m) = i \), compute \( \text{mnn}(i, j) \) distances defined as: \( \text{mnn}(i, j) = m + k \). The maximum \( \text{mnn} \)-distance is less than \( 2 \cdot N_{\text{cut}} \).

Different from the original scheme we have introduced additional threshold value \( RCN \). The values of \( N_{\text{cut}} \) and \( RCN \) control the size of clusters (e.g., the length of polymeric chains). The \( \text{mnn}(i, j) \) distances are sorted out in ascending order. The particle pairs \( (i, j) \) having the same \( \text{mnn}(i, j) \) distance are also sorted out the same way but according to the Euclidean distance between the particles.

A serial agglomerative clustering algorithm is given as follows:

1. **INITIALIZATION - NEIGHBORS SEARCH**
   a. Choose \( R_0 = \{ C_i = \{ x_i \}, \ i = 1, \ldots, N_p \} \)
   b. Create the list \( \Omega(t) \), of particle pairs \( (i, j) \) where \( t = 0, 1, 2, \ldots, M \) and \( M \) is the number of pairs, sorted out in increasing \( \text{mnn} \) and the Euclidean distances between \( i \) and \( j \)
   c. \( t = 0 \)

2. **REPEAT - MERGE CLUSTERS**
   a. Check if the particles \( (i, j) \) from \( \Omega(t) \) belong to different clusters \( C_i \) and \( C_j \), respectively.
   b. If not: \( t = t + 1 \) and go to 2
   c. If yes: merge clusters, i.e., \( C_i = C_i \cup C_j \)
   d. \( R_t = R_{t-1} - \{ C_j \} \)
   e. \( t = t + 1 \)

3. **FINISH** if \( t = M \)

The value of \( M \) is usually chosen as being equal to the greatest value of \( \text{mnn} \).

In Fig.2 we compare the clustering results by using K-Means and \( MNN \) schemes on the real data set. The use of inappropriate clustering scheme leads to serious errors in defining discrimination surface. As shown in [23] and in Figs.1b,2b, the \( MNN \) scheme can extract clusters of complicated shapes with irregular structures. Such the structures are found in discrete particle simulations of molecules, micelles, colloidal crystals and aggregates [16,17, 39-41].

Finding the nearest neighbors consumes most of the computational time of the \( \text{mnn} \) scheme, as in the case also for particle-based algorithms. For searching neighbors in 3-D space, however, there exist an efficient \( O(N_p) \) algorithm, which employ the linked-list concept [24]. Linked-list scheme is widely used in computations of short-ranged forces in particle simulations. Therefore, the neighbor tables obtained in \( \text{forces()} \) procedure can be reused for clustering or recomputed the same way.

To measure the efficiency of the serial clustering code written in FORTRAN 95, we performed our tests on IBM SP with Power3+ and for SGI-Origin 3800 system with R14000/500 CPUs. Both processors have a secondary cache with 8Mbytes. We used three data files representing particles positions from the same short simulation of blood cells flow in a capillary vessel by using fluid particle model (FPM) for different \( N_p \) – the number of particles. After 1,000 timesteps of thermalization, the FPM particles are simulated in the following 1,000 timesteps. We have performed clustering as a postprocessing procedure, at every 100 timesteps for different \( N_{\text{cut}} \) value. In Fig.3 we show the various number of particles as a function of clustering steps.
The most striking observation reveals that the computational time for clustering algorithm with linear computational complexity $O(Np)$ - experimentally confirmed for $Np$ less than one million particles - increases almost 5 times for $Np$ increasing from one million to two million particles. This sharp loss in efficiency can be observed both for IBM SP and for SGI/Origin systems. The size of data arrays for 2 million particles is about 60 MB. The arrays cannot fit in the cache. Because the particles that are the physical neighbors should also be closer one another in the computer memory, to avoid frequent cache misses the particles are renumbered every some period of time. As a consequence, the particles residing in the same cell have consecutive numbers. However, the gap between particle numbers still exists for the particles from different cells. This is due to the sequential numbering of particles in the computational domain. By increasing $2^{1/3}$ times the sizes of computational box in $x,y$ and $z$ directions, the gap between particle numbers from the neighboring cells increases almost 60%. The same effect we observed earlier for the code implementing fluid particle model [42]. However, the large clusters span more than along only neighboring cells. Therefore, the procedure for merging the clusters represents the most inefficient part of the code.

From the plots in Fig.3 we can conclude that the Power3+/375 processor is about two times faster than R14000/500 for $Ncut=6$ but the cache performance is better for SGI/Origin. Surprisingly, for $Ncut=4$ the SGI processor is only 50% slower. This may be due to different optimization of nested loops in merging cluster procedure, which uses only fixed point arithmetic, by the two compilers.

**Parallel algorithm**

As shown in Fig.4, the parallel algorithm for $MNN$ clustering has a similar structure to the algorithm for computing short-ranged forces in the particle ensemble [42]. We consider an isothermal two-dimensional system, which consists of $Np$ particles. The particles are enclosed within an elongated rectangular box. The box is divided into cubic cells with the edge size equal to the range of particle interactions $Rcut$. Because the cell size has been chosen to be slightly larger than $Rcut$, all the neighbors contributing to the total force acting on a given particle must be located within the cell containing the particle or within the adjacent cells. This key concept of the linked-list algorithm enable us to locate the nearest neighbors for all the particles by using the algorithm of $O(Np)$ complexity. These neighbors can be stored in auxiliary arrays and reused,
e.g., in the construction of higher order predictor-corrector numerical schemes time-stepping the equations of motion or deriving radial-distribution functions ($RDF$).

Parallel computing requires decomposing the computation into subtasks and mapping them onto multiple processors. The total volume of the box is divided into $P$ subsystems of equal volume, and each subsystem is assigned to a node in an array of $P$ processors. By using SPMD paradigm (single program multiple data), commonly used for MD code parallelization [3,14], each processor follows an identical predetermined sequence to calculate the forces on the particles within the assigned domain. The particles from cells, which are situated on the boundaries of processor domains, are copied to proper neighboring processors (see Fig.4). These boundary cells control the overhead in the communication. Each processor sends the message only in one direction to its closest neighbor.

**Fig.4** Diagram of processor communication in fluid particle model and the agglomerative clustering procedure.

For clustering on-line with simulation, we have reused the auxiliary neighboring tables within the framework of MNN algorithm. In the case of off-line clustering the same linked-lists procedure can be applied for closest neighbors search. However, instead of the interaction range $Rcut$, the largest distance the nearest neighbor particles from single cluster can be separated out $RCN$ has to be considered. In these simulations we define $RCN=Rcut$. In original MNN scheme the points can merge into a cluster if its mnn-distance is less than a given value. In the scheme proposed here both criteria are taken into account. They support both a mechanism for extracting clusters with a different density ($Ncut$ – number of nearest neighbors considered) and control the size of clusters ($RCN$).
PARALLEL PART

Forall domains $P = 1, N$

- Divide Domains-onto_Cells;
- Renumerate_Particles_in_Linked_Cells;
- SEND from $P-1$ to $P$ particles from boundary_cells;
- Find Neighbors for all particles in $P$ domain;
- SEND from $P$ to $P-1$ neighbors of particles from boundary_cells;
- Find Clusters in $P$ domain;
- SEND from $P$ to $P-1$ particle_cluster_numbers from boundary_cells

endforall

ENDPAR

SEQUENTIAL PART

- Merge Large cross-processors clusters
- Get_statistics

ENDSEQ

MAIN CLUSTERING PROCEDURES

Find Neighbors

```
procedure Find_Neighbors_in_P()
    for icell = 1, num_of_cells_in P
        iparticle = first_in_icell_link_list()
        while (end_of_link_list_of_icell != 0)
            jcell = 1, num_of_neigh_cells_to_icell + 1
            jparticle = first_in_jcell_link_list()
            while (end_of_link_list_of_jcell != 0)
                dist = distance_between (iparticle, jparticle)
                if (dist < RCN) then
                    update_list_of_mnn_neighbours_of_iparticle
                    update_list_of_mnn_neighbours_of_jparticle
                endif
                jparticle = next_in_jcell_link_list()
            endwhile
            jcell = next_in_icell_link_list()
        endwhile
        iparticle = next_in_icell_link_list()
    endfor
```

Merge Clusters

```
procedure Find_Clusters_in_P()
    run mnn agglomerative procedure

procedure Merge_cross-boundary_Clusters_in_P_domain
    if (boundary_cell_particle_i ∈ Cluster_A_from_P-1 and, boundary_cell_particle_i ∈ Cluster_B_from_P) then
        Merge (A,B) → SEND_to_Merge_Large_cross-processors_clusters;
    endif
```

Fig.5 The parallel algorithm of MNN clustering.
In Fig.5 we present a more detailed meta-code of MNN parallel algorithm, which consists of parallel procedures for finding neighbors and merging local and medium cross-domains clusters. The communication between processor domains is more complex than in the fluid particle code [42]. In finding the nearest neighbor we allow communication to take place only for the adjacent processor. It is the same as in fluid particle simulation code [42]. Differences appear for merging medium cross-boundary clusters.

The communication remains local, i.e., it occurs for adjacent domains, but may span to the larger number of cells - not only these being close to the domain borders. Merging medium clusters in one large cluster spanning along several processor domains involves global communication. However, this occurs usually at the end of the computations. Final merging is fast and can be computed sequentially on this processor, which is defined as the master. Both linked-list and merging clusters procedures are of $O(Np)$ complexity. As shown in the previous section, due to inefficient use of the cache the performance of serial code decreases dramatically for $Np>10^6$. Dividing the large task onto several ones should result in more efficient use of the cache and considerably increase of the speed performance.

The MNN parallel code was written in FORTRAN 95 and was implemented on the MPI interface for both the IBM SP and SGI/Origin 3800 platforms. We performed our tests on IBM SP with Power3+ processors (4 processors in a single node) with 8GB of memory per node. For comparison we present the benchmarks for SGI/Origin 3800 system with R14000/500 CPUs (2 processors per node) 4 GB of memory per node.

As we can anticipate from the previous section (see also Fig.6), we can obtain a super-linear speed-up obtained for $Np=2\times10^6$ due to the influence of the cache. It is clearly seen that the speed-up is almost two times greater for SGI/Origin system than for IBM SP for large number of particles (SGI: speed-up=40 for 16 processors and 2 million particles against SP: speedup=33 on 16 processors for 16 million particles). For $Np<10^6$ the two speed-ups are very similar. The decrease of speed-up is observed for larger number of CPU’s than 8 when CPUs from distant nodes are involved in computations.

![Graph](image)

**Fig.6** Timings on SGI/Origin 3800 (a) and IBM SP (b) with Power3+/375 chip. The data sets are consisted of FPM particles simulated in 1,000 timesteps. Clustering has been carried out off-line with the simulation, at every 100 timesteps.

**Simulation results**

For testing purposes we have employed the serial MNN clustering code for extracting 2-D patterns from three types of dissipative particle dynamics (DPD) simulations: phase separation in binary fluid, droplets condensation and agglomeration of colloids. As shown in Fig.7, we did not...
encounter any problems dealing with MNN clustering in the course of extracting both regular condensation patterns, irregular colloidal agglomerates and shapeless structures during the phase separation stage.

Fig.7 Snapshots from 2-D DPD simulation of condensation, colloid aggregates and phase separation in a binary fluid. The largest clusters extracted by MNN are shown in gray.

Fig.8 3-D results of clustering for the dispersion of slab in FPM particle fluid [8,17].
We have implemented clustering procedure for extracting clusters in 3-D from the large-scale data sets obtained from simulation of colloidal structures with fluid particle model [8]. We display in Fig.8 a few examples from the three different kinds of simulations. In Fig8a we show the clusters obtained for a colloidal slab accelerated in a particle fluid (black in Fig.8).

**Fig.9** 3-D visualization of clustering combined with volume rendering of the red blood cells (RBC) flowing through choking point in a narrow blood vessel simulated by using a two-level MD-FPM code. Different clot sizes are shown for normal RBC and sickle cells. The clots extracted by MNN clustering are colored in violet. Visualization of the blood flowing in bifurcating vessels is made by using Amira 2.3 [43].
Clustering allows for the visualization of different types of dispersive microstructures. The largest and the densest cluster of particles is depicted in white. The medium sized clusters are cast in red and single particles are shown in blue. Different dispersing structure can be seen in the figure on the right. Long red streaks of particles are produced behind the slab head. They are accurately extracted by this clustering code.

Dispersing structures, which are portrayed in Fig.8b, can be observed for fragmentation produced by the shattering mechanism. Clustering allows for tracing the evolution of the largest cluster, i.e., more solid part of the slab body, maintains the material strength. The temporal evolution of very complicated shapes of clusters obtained during colloidal agglomeration can be also clearly delineated by the method of \( MNN \) clustering. The extraction of even more complicated clusters is displayed in Fig.9.

Clustering becomes indispensable tool for the extraction of multi-level structures created by complex two-phase flows such as red blood cells (RBC) flowing in microscopic vessels. In Fig.9 we present two snapshots from simulations of RBC clotting close to the vein choking both for the “healthy” (left) and “sickle” (right) cells and choking of the flow in bifurcating vessels. We have neglected coagulation factors other than depletion forces. The RBC are made of particles coupled by harmonic forces. Two-level MD-FPM particle model was implemented.

Automatic identification of the clot size is difficult due to existence of hierarchy of multi-resolution structures: particles, RBC cells and clots of cells. The clots may be represented by medium and large between-processors clusters. Extracting such clusters consisting of flexible agglomerates of finer resolution, which consist of thousands of particles, is currently impossible employing the standard K-mean techniques or simple agglomerative clustering. As shown in Fig.9, the mnn parallel code can easily handle this prodigious problem.

Concluding Remarks

We have demonstrated here that the \( MNN \) hierarchical clustering scheme can extract efficiently distinct features from large-scale data sets produced in large-scale particle simulations. These schemes, usually applied for classifying N-dimensional patterns, are also truly promising tools for visualizing voluminous complex scientific data. Unlike the standard algorithms deployed in computer graphics, such as volume rendering, ray casting and ray tracing, the clustering schemes can be applied in extracting effortlessly valuable information from unstructured, irregular data, such as also found in finite-element calculations and wavelets besides the discrete particle simulations treated here.

Unlike other agglomerative schemes, which employ local metrics, \( MNN \) distance does not depend only on the two particle positions. The mutual nearest neighborhood distance between two particles combines sorted lists of the nearest neighbors of these particles. Thus the mnn distance is not as local as, e.g., the Euclidean distance or \( L^2 \)-norm distance and not as demanding as the Mahalanobis metrics. It reflects a cluster structure in the neighborhood of a particle defined by \( Ncut \) - the number of neighbors considered. Therefore, \( MNN \) clustering can self-adapt to the local resolution of a bunch of complex clusters with multi-resolution structures.

High complexity of the algorithms for finding nearest neighbors makes the clustering methods slow for large-scale and highly inhomogeneous multi-dimensional data. Fortunately, due to high degree of locality of both agglomerative clustering and short-ranged N-body codes such as molecular dynamics and fluid particle methods (DPD, FPM, SPH) and low dimensionality of feature space, the same linked-list procedure of \( O(Np) \) complexity can be employed in searching for the neighbors. We show, however, that the serial version of the \( MNN \) algorithm is of limited use.

The clustering of more than one million particles for IBM Power3+/375 and R14000/500 processors result in large overheads coming from inefficient use of the processor cache. The
procedure of merging small clusters into larger ones is mainly responsible for this side effect. To allow for efficient clustering data along with particle simulation for larger number of particles, both procedures must be run in a parallel environment. This is especially important for the algorithms where longer-ranged interactions between the particles can be approximated by particle-cluster and cluster-cluster interactions and the clustering procedure must be performed frequently, such the algorithmic property of clustering scheme can considerably increase computational efficiency of the parallel code on the whole. We have demonstrated that the parallel MNN algorithm, running on the IBM SP and SGI/Origin 3800 systems, can produce a super-linear speed-up.

We show also various examples of MNN clustering application in the visualization of microstructures resulting from simulations of complex fluids represented by up to 16 million of fluid particles. Visualization of particle clusters, combined with averaged velocity or density fields, allows for studying complex microstructures appearing in both micro and mesoscopic scales. As shown in Fig.9, the clusters representing clouds of particles can be visualized in various spatio-temporal scales by using volume rendering algorithms or a standard visualization software equipped with this technique such as Amira [43]. This aspect makes both clustering and rendering complementary tools devised for the parallel visualization of multiresolutional patterns.

These methods can also be utilized beyond these physical scales, for classifying not only objects but also dynamical discrete events such as: earthquakes [28,29,32] and the micro-earthquakes induced by mining activities [31,33] or dissipation phenomena occurring in a dispersed stellar population [34-35]. Clustering of organisms has profound social consequences on social behavior [44]. Environmental variability can cause clustering for example, slums in cities or plankton growth from marine turbulence. The clustering and feature extraction software can be also applied in large collaborative GRID computing projects [45-47] dealing with processing and concept extraction from large-scale data sets.

Acknowledgements

We thank discussions with Dr Jim Rustad from PNNL, Professor Gordon Erlebacher from Florida State University and Professor Yehuda Benzion of University of Southern California. This research was supported by the complex fluids program of Department of Energy and by the Energy Research Laboratory Technology Research Program of the Office of Energy Research of the U.S. Department of Energy under subcontract from the Pacific Northwest National Laboratory and by the KBN (Polish Committee of Scientific Research) project 4 T11F 02022.

References


43. TGS – Visual Concepts For the New Millenium, URL: [www.tgs.com](http://www.tgs.com)


46. Fox G, Presentation on *Web Services and Peer-to-Peer Technologies for the Grid* ICCS Amsterdam April 24 2002 URL: [http://grids.ucs.indiana.edu/ptliupages/publications/presentations/iccsapril02.ppt](http://grids.ucs.indiana.edu/ptliupages/publications/presentations/iccsapril02.ppt)

47. Fox G, internal Presentation on *Collaboration and Web Services* April 4 2002 URL: [http://grids.ucs.indiana.edu/ptliupages/publications/presentations/collabwsncsaapril02.ppt](http://grids.ucs.indiana.edu/ptliupages/publications/presentations/collabwsncsaapril02.ppt)