

Spring Conference on Computer Graphics SCCG 2008

in cooperation with
Eurographics and ACM SIGGRAPH

Conference Materials and Posters

Budmerice
April 21 - 23, 2008



i n v e n t

Publisher:
Editor:
Cover page & logo design:
Production:

Comenius University, Bratislava
Martin Samuelčík
Matej Novotný, Jozef Martinka
Martin Samuelčík

Webpage:

<http://www.sccg.sk>

ISSN:

1335 - 5694

International Programme Committee

IPC chairman
Prof. Karol Myszkowski
Max-Planck Institute for Informatics,
Germany

Jeroen van Baar
Sebastiano Battiato
Alexander Belyaev
Bedřich Beneš
Peter Birkholz
David Breen
Jiří Bittner
Alan Chalmers
Pavel Chalmovianský
Ján Cíger
Silvester Czanner
Miguel Chover
Roman Ďurikovič
Miquel Feixas
Petr Felkel
Andrej Ferko
Franca Giannini
Markus Grabner
Helwig Hauser
Vlastimil Havran
Bert Jüttler
Maxim Kazakov
Ivana Kolingerova

Rafal Mantiuk
Simone Marini
Karol Myszkowski
László Neumann
Alexander Pasko
Giuseppe Patanè
Bernard Péroche
Tomas Plachetka
Werner Purgathofer
Przemyslaw Rokita
Mateu Sbert
Benjamin Schmitt
Peter Shirley
Pavel Slavík
Martin Šperka
Daniel Thalmann
Petr Vaneček
Ivan Viola
Michael Wimmer
Pavel Zemčík
Borut Žalik
Jiří Žára

List of Reviewers

Jeroen van Baar
Sebastiano Battiato
Alexander Belyaev
Bedřich Beneš
Peter Birkholz
David Breen
Jiří Bittner
Pavel Chalmovianský
Ján Cíger
Silvester Czanner
Miguel Chover
Piotr Didyk
Roman Ďurikovič
Miquel Feixas
Petr Felkel
Andrej Ferko
Franca Giannini
Markus Grabner
Thorsten Grosch
Helwig Hauser
Vlastimil Havran
Matthias Hullin
Matthias Ihrke
Bert Jüttler
Maxim Kazakov

Ivana Kolingerova
Rafal Mantiuk
Simone Marini
Karol Myszkowski
Alexander Pasko
Giuseppe Patanè
Tomas Plachetka
Werner Purgathofer
Tobias Ritschel
Przemyslaw Rokita
Mateu Sbert
Benjamin Schmitt
Peter Shirley
Elena Šikudová
Pavel Slavík
Martin Šperka
Kateřina Tátraiová
Daniel Thalmann
Zsolt Tóth
Petr Vaneček
Ivan Viola
Michael Wimmer
Pavel Zemčík
Borut Žalik
Jiří Žára

Closed Particle Search based on Cell Sorting with Keys

Juraj Onderik*
Comenius University †

Roman Ďurikovič‡
Comenius University †
University of Saint Cyril and Metod §

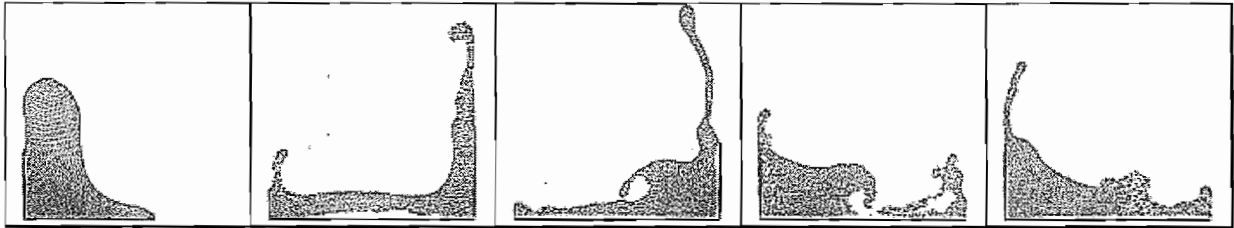


Figure 1: The classical dam-break test within our SPH fluid simulation environment.

Abstract

Lagrangian particle-based animation is a popular strategy for simulating fluids in computer graphics. Due to its inherent mesh-less nature a set of neighbor particles within a specified range must be efficiently found.

In this paper we propose *Cell Sorting* a novel approach for searching approximate neighbor particles necessary for efficient fluid simulation using SPH. Instead of storing particles into a fixed 3D grid or hash map, we encode their coordinates into *keys*. The list of keys is then sorted using linear time radix sort. A simple traversal using *H*-mask can quickly accumulate approximate neighbors without problematic cache misses of Spatial Hashing, large memory requirements of full 3D grids or $O(n \log n)$ time complexity of kd-trees. Furthermore we can achieve sub-cell precision by using larger *H*-masks, while having only constant factor slowdown. Using *H*-mask can substantially increase the precision of Spatial Hashing or 3D grids, however more cache misses or larger memory requirements arise.

We have demonstrated our approach within a standard SPH based fluid simulation.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism;

Keywords: Neighbor Search, Cell Sorting, H-Mask, Multi-Phase Smoothed Particle Hydrodynamics

1 Introduction

Realistic animation of complex natural phenomena such as fluids is an attractive and challenging research. For both Eulerian [Guendelman et al. 2005; Losasso et al. 2006] and Lagrangian approaches

[Keiser et al. 2005; Solenthaler et al. 2007] advanced unified solid-fluid coupling techniques has been proposed. Still a number of issues related to stability, accuracy, realistic boundary conditions and performance arise. Recent graphics hardware allows huge parallelization of many time consuming problems, thus simulation algorithms has to be adapted.

In Lagrangian particle-base simulation physical quantities are sampled on particle locations. The interaction among particles is only local, within a specified threshold distance. It is thus essential to quickly find for each particle the respective set of close neighbor particles. Since this highly affects further calculations of fluid dynamics it becomes usually the bottleneck of the overall animation.

This work contributes *Cell Sorting* an efficient algorithm for searching neighbor particles. By associating unique *key* to each particle we quickly build close particle pairs using a simple traversal of the sorted list of keys. In the results we give a comparison of our method to the popular *Spatial Hashing* [Teschner et al. 2003] technique and show scenarios where Cell Sorting performs almost twice as fast.

2 Related Work

Searching for all nearest neighbors (ANN) is a traditional problem among various research areas. Concerning molecular dynamics and particle-based fluid simulation traditional methods based on *locality graphs* (verlet tables) [Verlet 1967] and *uniform grids* [Allen and Tildesley 1989] has been well established. Hierarchical methods [Hernquist and Katz 1989] has been proposed to overcome large memory requirements of full 3D voxelization. However localizing particles in a logarithmic time inside the searching tree was insufficient for fast fluid simulations. Therefore spatial hashing techniques [Teschner et al. 2003] has been developed.

While adapting SPH simulation algorithms to graphics hardware, cache misses and bucket collisions arose using spatial hashing. Similar to our approach Staggered Grids [Kipfer and Westermann 2006] has been developed based on particle sorting along all three coordinate axes. In comparison our Cell Sorting technique sorts particles only once. Recently both fluid dynamics and neighbor search has been moved onto GPU storing only non-empty cells of full 3D grid [Harada et al. 2007].

*e-mail: juraj.underik@fmph.uniba.sk

†Faculty of Mathematics, Physics and Informatics, Comenius University, 842 48 Bratislava, Slovakia <http://www.fmph.uniba.sk>

‡e-mail: roman.durikovic@fmph.uniba.sk

§Faculty of Natural Sciences, University of Saint Cyril and Metod, 917 01 Trnava, Slovakia <http://www.ucm.sk/FPV>

3 Our Efficient Neighbor Search

Inspired by Staggered Grids, *Cell Sorting* uses coordinate sorting for searching neighbor particles efficiently. For each particle p we have to find the set $N(p, h)$ of neighbor particles p' whose distance to p is within a specified threshold h . Formally

$$N(p, h) = \{ p' \mid |\mathbf{r}' - \mathbf{r}| \leq h \} \quad (1)$$

where \mathbf{r}, \mathbf{r}' are positions of particles p, p' respectively.

The key idea of our neighbor search algorithm is to assign a unique (number) *key* to each particle, *sort* all these keys in an ascending order and finally *traverse* efficiently the sorted list of keys and report close particle pairs.

3.1 Spatial Subdivision and Cell Sorting

Similarly to other grid methods we first calculate the smallest enclosing *Axis Aligned Bounding Box* (AABB) of all particles and (virtually) subdivide it into a 3D grid of cells with size h . Assuming the AABB is described by its *minimal corner* $\mathbf{c}_{min} = (x_{min}, y_{min}, z_{min})$ and *maximal corner* $\mathbf{c}_{max} = (x_{max}, y_{max}, z_{max})$, we define for each particle its (positive) *cell coordinates* $cell(x, y, z, h) = (i, j, k)$ with respect to the minimal corner as

$$cell(x, y, z, h) = \left\lfloor \frac{x - x_{min}}{h}, \frac{y - y_{min}}{h}, \frac{z - z_{min}}{h} \right\rfloor \quad (2)$$

where (x, y, z) is the particle position. To a given particle with cell coordinates $cell(x, y, z, h) = (i, j, k)$ we assign a unique number *key* $key(i, j, k)$ defined as

$$key(i, j, k) = i + I \cdot j + I \cdot J \cdot k \quad \text{and} \quad \begin{matrix} I \leq \lfloor B_x/h \rfloor \\ J \leq \lfloor B_y/h \rfloor \end{matrix} \quad (3)$$

where I, J are arbitrary depending on dimension of AABB $(B_x, B_y, B_z) = (x_{max} - x_{min}, y_{max} - y_{min}, z_{max} - z_{min})$ and cell size h . Assuming all particles lie inside the enclosing AABB the function $key(i, j, k)$ is *invertible*. Thus given a key associated with a particle we get directly the unique cell in which this particle lies. Actually in our implementation we set $I = J = 2^{16}$ and assume all cell coordinates are $i, j, k < 2^{16}$. Therefore we can simply form keys by storing the 16-bit cell coordinates (i, j, k) continuously in memory as 48-bit data chunks.

It is crucial to understand that sorting associated keys into ascending order allows us to efficiently report close particle pairs. Since keys are represented with positive integral numbers we can sort them using a simple linear-time *radix sort* algorithm [Terdiman 2000]. Notice we actually store only the permutation of indices of keys and do not move directly keys during the sort. This index permutation is important, since after the sort we can access particles in the same "sorted" order as the sorted keys.

3.2 Searching in 1D

We will explain the search algorithm first in 1D scenario, see figure (2). Given a referential particle p which lies in cell (i) having a key $q = key(i)$, we want to construct efficiently the set of neighbor particles $N(p, h)$.

Since the cell size is h , we need to *examine* only particles which lie in (neighbor) cells $(i-1), (i), (i+1)$. This means their keys q'

must be $q' = key(i-1)$ or $q' = key(i)$ or $q' = key(i+1)$. For each such particles we then simply compute the exact distance to the referential particle and report only close pairs, see figure (2).

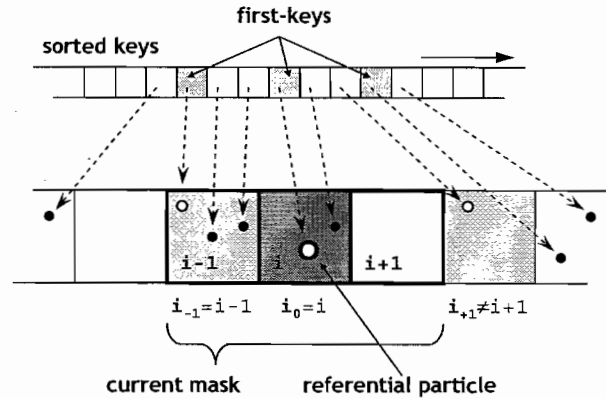


Figure 2: Cell Sorting principle in 1D. First row represents the array of sorted keys built on the respective set of particles (second row). The picture shows the situation, when the referential particle lies in i -th cell thus searching cell $(i-1), (i), (i+1)$. However the first key of cell $(i+1)$ is not within the investigated mask thus is automatically skipped.

Since the list of keys is already sorted, all keys q belonging to a common cell (i) have the same value $q = key(i)$ and thus form a continuous sublist $Q(i)$ with the *first-key* denoted as $q(i)$. Formally

$$\begin{aligned} Q(i) &= (q_a, q_{a+1}, \dots, q_b) \\ q(i) &= q_a \end{aligned} \quad (4)$$

where $a \leq b$ are appropriate indices in the sorted key list. For empty cells we set $Q(i) = \emptyset$. Since keys of neighbor particles belong only to the union of $Q(i-1)$, $Q(i)$ and $Q(i+1)$ we need to enumerate these lists efficiently. This is achieved once we can quickly determine respective first-keys $q(i-1)$, $q(i)$ and $q(i+1)$. If we process referential particles p in the sorted order of keys, respective first-keys $q(i-1)$, $q(i)$ and $q(i+1)$ can be updated in a straightforward fashion. Suppose we have already found $q(i-1)$, $q(i)$ and $q(i+1)$ (e.g. in previous iteration) and constructed neighbor sets for all referential particles in cell (i) . Now the next referential particle lies in next cell $(i') = (i+1)$, therefore $q(i'-1) = q(i)$, $q(i') = q(i+1)$ and only $q(i'+1)$ must be determined. If we remember last key q_b in $Q(i+1)$, then simply $q(i'+1) = q_{b+1}$.

As shown in figure (2) if a cell (i) is empty (has no particles inside) it has no first-key $q(i)$ and we should skip processing it. Let $(i_{-1}) \leq (i_0) \leq (i_{+1})$ be the *last, referential and following* non-empty cells. Then $q(i_{-1}) \leq q(i_0) \leq q(i_{+1})$ must exist and have the same ordering. When constructing neighbor sets we examine particles in the last cell only when it is also the previous cell ($i_{-1} = i-1$) and in following cell only when it is also the next cell ($i_{+1} = i+1$).

Formally, we store first-keys $q(i_s)$ of non-empty cells (i_s) where $i_s \geq i + s$ and $s \in \{-1, 0, +1\}$. We process particles in cell (i_s) only when $(i_s) = (i + s)$.

¹while building neighbor set $N(p, h)$

3.3 Searching in 2D, 3D

The extension of the proposed 1D searching algorithm into higher dimensions is almost straightforward. For each referential particle we need to examine particles in all 9 (2D) or 27 (3D) neighboring cells and report exact particle pairs, see figure (3). In 3D we de-

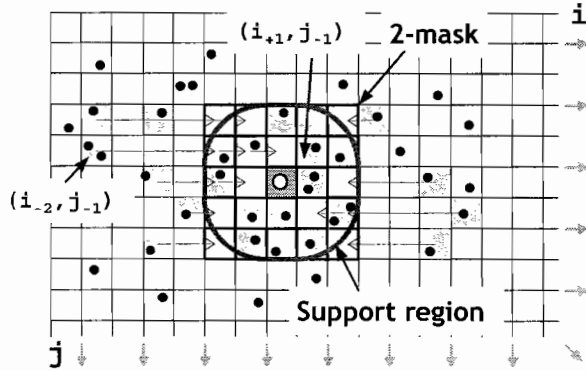


Figure 3: Cell Sorting principle in 2D. Searching on a sub-cell precision is shown by virtually splitting cells and introducing (5x5) 2-mask instead of classical 3x3 1-mask. Respective particles of all current first-keys lie in light-gray cells. Particles outside the support region can be clearly ignored, thus for larger masks some "corner" cells can be fully omitted.

fine the set of common keys $Q(i, j, k)$ and its respective first-key $q(i, j, k)$ as

$$\begin{aligned} Q(i, j, k) &= (q_a, q_{a+1}, \dots, q_b) \\ q(i, j, k) &= q_a \end{aligned} \quad (5)$$

Due to the chosen ordering of keys, the construction of the neighbor sets by selecting referential particles in sorted order simplifies update of first-keys in higher dimensions as well. Formally, we store first-keys $q(i_s, j_t, k_u)$ of non-empty cells (i_s, j_t, k_u) where $i_s \geq i + s$, $i_t \geq i + t$, $i_u \geq i + u$ and $s, t, u \in \{-1, 0, +1\}$. We process particles in cell (i_s, j_t, k_u) only when $(i_s, j_t, k_u) = (i + s, j + t, k + u)$.

3.4 Algorithm Summary

Assuming all first-keys $q(i_s, j_t, k_u)$ for $s, t, u \in \{-1, 0, +1\}$ are properly computed for a the given referential particle p function REPORTNEIGHBORS(p, h, δ) will report all pairs of close particles (p, p_m) whose distance is within the support h . As shown in algorithm (1) we enumerate all particles within lists $Q(i_s, j_t, k_u)$. This is achieved by condition on line (6).

Reporting all pairs of particles satisfying the distance condition is now straightforward. As shown in algorithm (2) function REPORTALLNEIGHBORS(h) will report desired pairs by first blindly initializing all 27 first-keys to the very first key in the sorted list of keys. Next calling REPORTNEIGHBORS($p_m, h, 1$) ensures that all first-keys will be properly set for the first referential particle. Finally we enumerate all particles in the sorted order and report their close neighbors by iteratively calling REPORTNEIGHBORS($p, h, 0$).

In: referential particle p , support length h , reporting switch δ

```

function REPORTNEIGHBORS( $p, h, \delta$ )
1: ( $x, y, z$ )  $\leftarrow$  POSITION( $p$ )
2: ( $i, j, k$ )  $\leftarrow$  cell( $x, y, z, h$ )
3: foreach  $s, t, u$  in  $\{-1, 0, +1\}$  do
4:    $n \leftarrow$  KEYINDEX( $q(i_s, j_t, k_u)$ )
5:    $q \leftarrow$  key( $i + s, j + t, k + u$ )
6:   while SORTEDKEYS( $n$ )  $\leq$   $q - \delta$  do
7:      $m \leftarrow$  PARTICLEINDEX( $n$ )
8:     if  $\delta = 0 \wedge$  DISTANCE( $p, p_m$ )  $\leq$   $h$  then
9:       REPORTPAIR( $p, p_m$ )
10:    end
11:     $n \leftarrow n + 1$ 
12:  end
13:   $q(i_s, j_t, k_u) \leftarrow$  SORTEDKEYS( $n$ )
14: end
end

```

Algorithm 1: ReportNeighbors computes close neighbors for the referential particle p . Parameter δ is responsible for reporting pairs ($\delta = 0$) or updating first-keys ($\delta = 1$)

In: support length h

```

function REPORTALLNEIGHBORS( $h$ )
1: foreach  $s, t, u$  in  $\{-1, 0, +1\}$  do
2:    $q(i_s, j_t, k_u) \leftarrow$  SORTEDKEYS(0)
3: end
4:  $m \leftarrow$  PARTICLEINDEX(0)
5: REPORTNEIGHBORS( $p_m, h, 1$ )
6: foreach  $p$  in SORTEDPARTICLES do
7:   REPORTNEIGHBORS( $p, h, 0$ )
8: end
end

```

Algorithm 2: ReportAllNeighbors finds all pairs of close particles. By setting ($\delta = 1$) ReportNeighbors only initializes first-keys.

4 Results

To measure the efficiency of our algorithm we have performed following testing scenario. Given 15000 randomly sampled particles we get $15000 \times 15000 = 225000000$ all possible pairs of particles. Depending on the sampling domain size, only a fraction of them belong to close particle pairs. We define the *pair ratio* as the fraction between *close pairs* and *all pairs*. Setting $h = 1$ we started with a sampling domain $2 \times 2 \times 2$ (pair ratio = 0,9999) and end with domain $15 \times 15 \times 15$ (pair ratio = 0,0069).

Figure (4) shows measured total time for reporting all close particle pairs using three different algorithms namely *SortGrid* (Cell Sorting), *HashGrid* (Spatial Hashing) and *BruteForceN2* (All-Pair-Test). As shown in figure (5) for low pair ratio Cell Sorting is almost twice as fast as Spatial Hashing. All simulations have been performed on a Mobile P4 1.7 GHz with GeForce 4 Go.

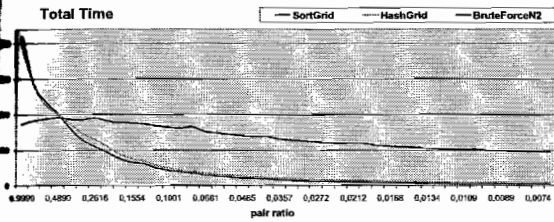


Figure 4: Relation between pair ratio and the respective total time required to find close particle pairs using Cell Sorting (red), Spatial Hashing (green) and All-Pair-Test (blue).

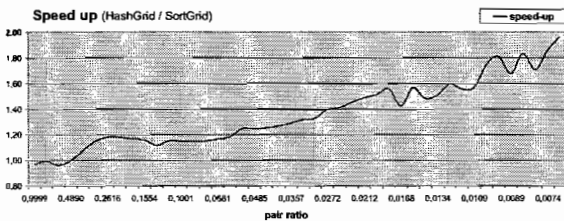


Figure 5: Depiction of the speed up ratio between Cell Sorting and Spatial Hashing again for decreasing pair ratio.

5 Conclusion and Future Work

We have proposed Cell Sorting a novel approach for searching neighbor particles and demonstrated it within our SPH fluid simulator based on [Müller et al. 2005], see figure (1). Due to cache misses, Spatial Hashing has been slightly outperformed by our method, without large memory requirements of a Full 3D voxelization. Our approach is inherently linear, thus will outperform hierarchical methods² for larger data sets. We have introduced a H -mask to achieve searching on a sub-cell resolution.

As a future work we will investigate the possibility of involving spatial and temporal coherence into our method and implement it on GPU. Further we will focus on decreasing the compressibility of fluid in SPH by solving the pressure implicitly using iterative techniques.

6 Acknowledgements

This research was sponsored by grant from EU-FP6-MC-040681-APCOCOS.

References

- ALLEN, M. P., AND TILDESLEY, D. J. 1989. *Computer simulation of liquids*. Clarendon Press, New York, NY, USA. 1
- GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.* 24, 3, 973–981. 1
- HARADA, T., KOSHIZUKA, S., AND KAWAGUCHI, Y. 2007. Sliced data structure for particle-based simulations on gpus. In

²They usually need $O(n \log n)$ to rebuild.

GRAPHITE '07: *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, ACM, New York, NY, USA, 55–62. 1

HERNQUIST, L., AND KATZ, N. 1989. Treesph - a unification of sph with the hierarchical tree method. 419–446. 1

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRE, P., AND GROSS, M. 2005. A unified lagrangian approach to solid-fluid animation. In *Symposium on Point-Based Graphics*, Eurographics Association, Zurich, Switzerland, M. Gross, H. Pfister, M. Alexa, and S. Rusinkiewicz, Eds., 125–133. 1

KIPFER, P., AND WESTERMANN, R. 2006. Realistic and interactive simulation of rivers. In *Proceedings Graphics Interface 2006*, Canadian Human-Computer Communications Society, S. Mann and C. Gutwin, Eds., 41–48. 1

LOSASSO, F., IRVING, G., AND GUENDELMAN, E. 2006. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics* 12, 3, 343–352. Member-Ron Fedkiw. 1

MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. 2005. Particle-based fluid-fluid interaction. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 237–244. 4

SOLENTHALER, B., SCHLÄFLI, J., AND PAJAROLA, R. 2007. A unified particle model for fluid-solid interactions. *Comput. Animat. Virtual Worlds* 18, 1, 69–82. 1

TERDIMAN, P., 2000. Radix sort revisited. Online Paper. URL: <http://www.codercorner.com/RadixSortRevisited.htm>. 2

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANTES, D., AND GROSS, M. H. 2003. Optimized spatial hashing for collision detection of deformable objects. In *VMV*, 47–54. 1

VERLET, L. 1967. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.* 159, 1 (Jul), 98. 1