Eurographics Symposium on Point-Based Graphics (2007)
M. Botsch, R. Pajarola (Editors)

Efficient Refinement of Dynamic Point Data

B. Solenthaler †, Y. Zhang, R. Pajarola

Visualization and MultiMedia Lab, Department of Informatics, University of Zurich

Abstract

Particle simulations as well as geometric modeling techniques have demonstrated their ability to process and render points interactively. However, real-time particle-based fluid simulations suffer from poor rendering quality due to low surface particle resolutions. Surfaces appear blobby, surface details are lost, and features like edges are degraded due to smoothing effects. This paper presents a novel point refinement method for irregularly sampled, dynamic points coming from a particle-based fluid simulation. Our interpolation algorithm can handle complex geometries including splashes, and at the same time preserves features like edges. Point collisions are avoided resulting in a nearly uniform sampling facilitating surface reconstruction techniques. No point preprocessing is necessary, and point neighborhoods are dynamically updated reducing computation and memory costs. We show that our algorithm can efficiently detect and refine the surface points of a fluid and we demonstrate the improvement of rendering quality and applicability to real-time simulations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: – Animation and Virtual Reality I.3.5 [Computational Geometry and Object Modeling]: – Curve, surface, solid, and object representations

1. Introduction

Point representations have been used successfully in geometric modeling and in physically based particle systems. The lack of topological and connectivity information simplifies modeling interaction effects (eg. [MKN*04, MHTG05, MSKG05, PKA*05, SSP07]) as well as geometric manipulations (eg. [ZPKG02, AD03, PKKG03, PKG06]). However, it comes at a cost, as neighborhood information has to be computed. Nevertheless, [MCG03] succeeded in interactively simulating and rendering particle-based fluids and demonstrated its applicability to virtual reality simulators and 3D games [MST*04, Age05]. However, due to the real-time constraint, the number of particles has to be low which causes a loss of visual quality. Surface details are smoothed out as a result of surface reconstruction techniques and bumps related to the coarse particle distribution are visible. Using a point splatting approach as rendering technique is also not feasible as under-sampled geometries show artifacts at the silhouette and blur due to large splat radii. It would be desirable to improve the visual quality of low resolution particle simulations while still running at interactive frame rates.

The presented approach is to use an upsampling algorithm on the surface particles to optimize the visual appearance of particle simulations. Such a technique avoids the overhead of running a high resolution physical simulation which is substantial as the number of physical particles increases disproportionally with the desired number of surface particles. Even worse, the Courant condition requires smaller time steps for higher resolutions ([Mon89]) resulting in a computational effort of physical simulations increasing quadratically with the number of desired surface particles.

For static point geometries, refinement methods have already been demonstrated (e.g. [PGK02, ABCO*03, GBP04, GBP05]). Nevertheless, these methods cannot be applied to points evolved by a Smoothed Particle Hydrodynamics (SPH) simulation [Mon92] due to the real-time constraint as well as the challenging properties of such dynamic SPH particles. The challenges thereby are irregularly distributed particles, low-quality normals, as well as complex surfaces including splashes and isolated particles. Another issue is the robustness of the algorithm used for surface particle detection and upsampling in order to avoid visible artifacts resulting from the dynamic nature of particle simulations.

1.1. Our Contributions

To reveal all details present in (low resolution) particle-based fluid simulations (for interactive and real-time applications), we propose an efficient upsampling method applicable, but

† e-mail: {solenthaler, zhang}@ifi.uzh.ch, pajarola@acm.org

not limited to irregularly sampled, dynamic point data. The main features of our method are:

- **Low computational costs**: No point preprocessing is necessary, new points are added efficiently, and neighborhoods are updated dynamically instead of being determined from scratch each refinement step, reducing computation and memory costs.
- **Irregular inital sampling**: SPH particles are often irregularly distributed during the simulation. The refinement procedure effectively copes with this problem, and holes are generated only if indicated by the physics simulation.
- **Uniform sampling after refinement**: Point collisions are detected and avoided resulting in a nearly uniform sampling.
- **Details and sharp features**: Our interpolation yields many details of surfaces and splashes, and preserves features like edges.
- **Splashes and isolated particles**: Although isolated particles and particles in splashes posess low-quality normals, curvature and connectivity are preserved in a plausible way.

1.2. Related Work

Currently, many particle-based fluid simulation models are based on SPH, which was developed in astrophysics [Mon92, Mon05]. SPH has been successfully used in graphics to simulate soft objects [DC96], lava [SAC*99], multiple fluids [MSKG05], phase changes [MKN*04, KAG*05, SSP07] and interaction with solid objects [MKI03, MST*04]. It was also used for fluid control [TKPR06] and with adaptively sized particles [APKG07]. A real-time SPH simulation was demonstrated in [MCG03].

To reconstruct the surface from a set of fluid particles several techniques have been proposed, all without upsampling the surface points of a fluid. An efficient approach is presented in [MCG03] where they render the isosurface of a color field defined by the particles. A grid-based level-set simulation guided by particles is presented in [PTB*03], where they succeeded in achieving high visual quality but at the expense of computation time. [ZB05] presented a reconstruction technique using an implicit function defined on the center of mass of a local neighborhood of the particles leading to very smooth surfaces. Unfortunately, this method suffers from artifacts in concave regions which they propose to remove in a post processing step. In [SSP07], a method is presented to detect and avoid these errors on the fly. Another extension has been presented in [APKG07], where particle-to-surface distances are used for the reconstruction of surfaces from adaptively sized particles.

In geometry processing, a surface reconstruction based on the use of Radial Basis Functions with global support is presented in [CBC*01], whereas [OBA*03, TRS04] reduce the support by local approaches. MLS surface reconstruction [Lev03] has shown to be successful in surface editing [PKKG03], raytracing [AA03], and up- and down-sampling [PGK02, ABCO*03]. A sphere fit MLS improving the stability of the projection in low-sampled and curved regions has been presented in [GG07]. However, since the projection procedure of the MLS is quite expensive, it is unsuitable for upsampling a set of points in real-time. Real-time upsampling restricted to static, uniformly sampled point data was presented in [GBP04]. [GBP05] overcomes this weakness and presented a method which is able to fill large holes of static point clouds.

2. Particle-Based Physics Simulation

We briefly describe the particle-based fluid simulation model SPH which we use to demonstrate our upsampling method. SPH is an interpolation method for particle systems, where viscosity and pressure force fields are directly derived from the Navier-Stokes equations. The fluid is discretized using particles which carry field quantities A. At any position $\mathbf{r}$ in space, these quantities can be evaluated by summing up the weighted contributions of the neighboring particles p_j:

$$A(\mathbf{r}) = \sum_j \frac{m_j}{\rho_j} A_j W(\mathbf{r} - \mathbf{r_j}, r), \tag{1}$$

where m_j and ρ_j are the mass and the density of particle p_j, respectively. The radial symmetric weighting kernel $W(\mathbf{r}, r)$ has a finite support determined by the support radius r. In our simulation, we use the physics equations and kernel functions presented in [MCG03] and [MSKG05]. For a more detailed description about SPH we refer to [Mon92, Mon05].

3. Refinement

3.1. Surface Particle Detection

Since we are interested in visualizing the fluid surface, we only want to refine surface particles. The detection of free surface particles is a difficult problem and has shown to be a critical step in our refinement procedure as erroneously detected and erroneously undetected particles can lead to surface artifacts. In the particle simulation literature, several methods to detect surface particles have been proposed, often based on the number of neighbors in the support radius. These techniques do not satisfy our needs as particles are partly erroneously detected as surface, especially when high pressure forces are involved. This is ascribed to the irregular particle distribution and therefore to the non-constant number of particles in the neighborhoods.

We propose a method to detect surface particles which is based on the distance from the particle to the normalized center of mass **cm** of its neighborhood N. This criterion still investigates the particle distribution in the neighborhood of a particle but is independent of the actual number. See Figure 1 for an example. For each particle p_i, this distance $d_{i,cm}$ is calculated by

$$d_{i,cm} = \frac{\sum_j (\mathbf{r_i} - \mathbf{r_j}) m_j}{\sum_j m_j}. \tag{2}$$

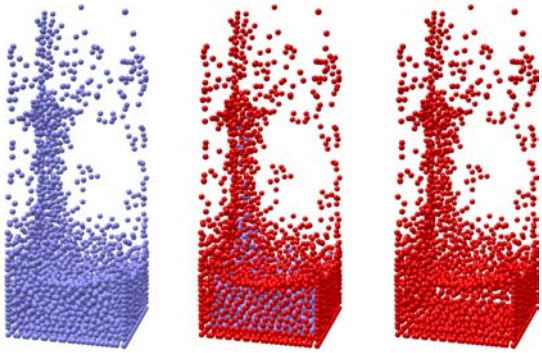

Figure 1: *Our surface particle detection method applied to a splashing scenario (left). The red particles are detected as being surface (middle) and correspond to the input set of the upsampling procedure (right).*

A surface particle is defined by having a $d_{i,cm}$ exceeding a certain threshold. In order to avoid oscillations between being surface and not being surface, we use a slow reaction for surface particles to turn into the state of not being surface. This is achieved by using two different thresholds, where a lower one is used for particles which are already marked as surface. Isolated particles have to be treated separately, where they are defined by having an empty neighborhood.

3.2. Point-Normal Interpolation

3.2.1. Initial Neighborhood

We use the surface particles as initial point set P_0 for the refinement procedure (Figure 1). Similar to [GBP04] we insert additional points which yield a new point set $P_1 \supset P_0$. The new point set P_1 is then used for the next refinement step. This procedure can be repeated until the desired point resolution is reached. During one refinement step, an additional point is inserted between each point p_i and its neighbors p_j, where it is avoided to generate points which are too close to points already created in order to achieve a nearly uniform sampling (Section 3.3). Figure 2 illustrates the points of the individual refinement steps. The neighborhood $N_i^{surface}$ of each point $p_i \in P_0$ is inherited from the physics simulation, where $N_i^{surface} \subset N_i$: if two surface particles are visible to each other in the physics calculation, meaning that they interact with each other, the pair is refined. Otherwise, if two particles have a distance larger than the support radius and do not interact with each other, no additional point is added. That is, the initial visibility radius r_0 used in the refinement procedure is equal to the support radius r used in the physics. Note that r_0 is the same for all points. In the following refinement step, the radius is reduced as described in Section 3.2.3. The point normals are determined either in the physics by using the method proposed in [MCG03] or by using a transformation invariant homogeneous covariance analysis as described in [Paj03]. The latter is more expensive but leads to higher quality normals which is crucial when using point splatting as rendering technique (see Figure 9).

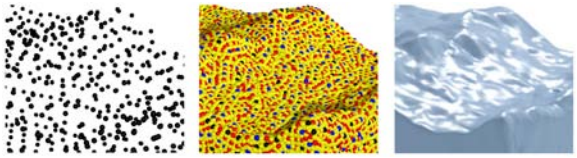

Figure 2: *A wavy scene is upsampled from 2.5k to 110k points. The point colors correspond to the different refinement steps: black P_0, blue P_1, red P_2, and yellow P_3.*

3.2.2. Spherical Interpolation

If a new point is added, its position and normal have to be determined. For real-time applications, it is important that we get these values at low computational costs. Nevertheless, surface details should be preserved as far as possible. While standard Bezier schemes are efficient ([GBP04, GBP05]) they often show insufficient uniformity of upsampled points. We therefore propose a spherical interpolation method which sets a new point (child $c_{1,2}$) onto a sphere which is defined by the two points being refined (parents p_1 and p_2), as illustrated in Figure 3. The child point is set onto the perpendicular bisector of $\mathbf{d} = \mathbf{r_1} - \mathbf{r_2}$, where the displacement from $\mathbf{d}$ onto the sphere is called h. The position of the child $\mathbf{r_c}$ is given by

$$\mathbf{r_c} = \frac{\mathbf{r_1} + \mathbf{r_2}}{2} + h\mathbf{n_c}, \tag{3}$$

where $\mathbf{n_c}$ is the normal of $c_{1,2}$ as defined in Equation 14.

Each parent computes a displacement h_1 and h_2, respectively, and the final h is a function of h_1 and h_2. There are different possibilities to determine h, one is to take the average, another one is to take the absolute minimum:

$$h = \frac{h1 + h2}{2} \tag{4}$$

$$h = min(|h1|, |h2|). \tag{5}$$

The first one (illustrated in Figure 3) leads to a smooth surface whereas the second one preserves edges and corners more accurately. We prefer the second one and our results are all computed using this approach.

To facilitate the calculation of h_1 and h_2, we determine the slopes $\frac{s_1}{t_1}$ and $\frac{s_2}{t_2}$. The displacements are given by

$$h_1 = \frac{s_1}{t_1}\frac{\|\mathbf{d}\|}{2} \tag{6}$$

$$h_2 = \frac{s_2}{t_2}\frac{\|\mathbf{d}\|}{2}, \tag{7}$$

where s and t are defined by

$$s_1 = |\mathbf{n_1} \cdot \mathbf{a}| \tag{8}$$

$$t_1 = \mathbf{n_1} \cdot \mathbf{b} + \|\mathbf{n_c}\| \tag{9}$$

and

$$s_2 = |\mathbf{n_2} \cdot \mathbf{a}| \tag{10}$$

$$t_2 = \mathbf{n_2} \cdot \mathbf{b} + \|\mathbf{n_c}\|. \tag{11}$$

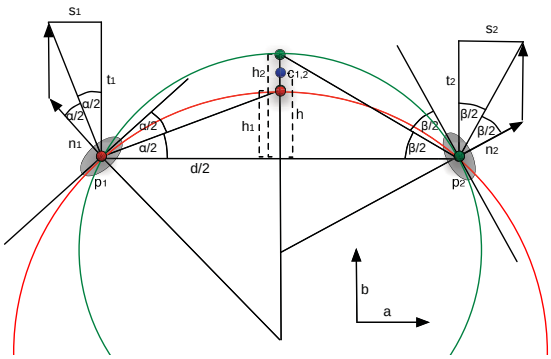

Figure 3: *A new child point $c_{1,2}$ is added between the parent points p_1 and p_2. Each parent computes its h1 and h2, respectively, where the final h is a function of h1 and h2.*

a and **b** are two basis vectors, where the normal $\mathbf{n_c}$ of the child point corresponds to **b**:

$$\mathbf{a} = \frac{\mathbf{d}}{\|\mathbf{d}\|} \tag{12}$$

$$\mathbf{b}' = \mathbf{n_1} + \mathbf{n_2} - ((\mathbf{n_1} + \mathbf{n_2}) \cdot \mathbf{a})\mathbf{a} \tag{13}$$

$$\mathbf{b} = \mathbf{n_c} = \frac{\mathbf{b}'}{\|\mathbf{b}'\|}. \tag{14}$$

$\|\mathbf{b}'\|$ can be zero if the normals of both parents are parallel to **d** or if the normals sum up to zero. In these situations, we assume that the points belong to different surfaces and, therefore, the points are not refined.

3.2.3. Radius in the Next Refinement Step

For efficiency reasons, it is desirable to limit $|h|$ to h_{max}. We choose h_{max} so that 6 points may still refine to a perfect sphere assuming this is a reasonable requirement to the resolution of an underlying simulation. For an illustration in 2d see Figure 4. In this situation, h_{max} is given by

$$h_{max} = \rho - \frac{r_i}{2} = \frac{r_i}{2}(\sqrt{2} - 1), \tag{15}$$

where ρ is the sphere radius. As we know h_{max}, the radius r_{i+1} of the next refinement step is always well-defined, since it is required that the child is just in the neighborhood of its parent to avoid parasitic holes. Therefore, r_{i+1} is defined by

$$r_{i+1} = \sqrt{h_{max}^2 + (\frac{r_i}{2})^2} = \frac{r_i}{2}\sqrt{4 - 2\sqrt{2}} \tag{16}$$

which means that in each refinement step the radius is reduced by a factor of ≈ 0.54.

3.3. Point Collision Avoidance

As mentioned above, for each pair of points a new point is inserted. As we want to reach a uniform sampling after the refinement procedure, it is important to avoid adding points too close to existing points.

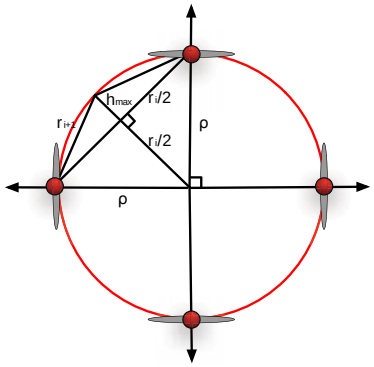

Figure 4: *Two-dimensional illustration of the limitation of h to h_{max}. In three dimensions, 6 points are refined to an accurate sphere.*

3.3.1. Collision Detection

We use a collision distance β which defines a sphere around a point which has to be empty. The collision volume of a child point for the case $h = h_{max}$ is two-dimensionally illustrated in Figure 5 as red circle. If any other point lies in the same volume the child point is rejected and not added. In each refinement step, β is adjusted proportionally to r_i. We use $\beta = 0.2r_i$ which we have determined heuristically to be adequate to approach a uniform sampling. There are three different collision configurations, see Figure 6 for an illustration. (for simplicity reasons, we refer to the different refinement steps as $s_0, s_1, s_2, \ldots, s_n$, and a point which is added in step s_i is called p^{s_i}):

1. A point created in the current refinement step s_i is closer than β to its parents.
2. A point created in the current refinement step s_i is closer than β to a (non-parent) point which was created in one of the last refinement steps $s_{j<i}$.
3. A point created in the current refinement step s_i is closer than β to another point which was also created in the current step s_i.

The first type where a child collides with its parents can be avoided by not refining a pair of points which are closer than r_{i+1}. This is valid as long as $\beta < 0.5r_{i+1}$, which is the case when using $\beta = 0.2r_i$. In this situation, the pair gets refined in the next refinement step s_{i+1}.

Since both parents know their neighboring points added in $s_{j<i}$ we can avoid collisions of type 2 by taking the intersection set I of both parent neighborhoods N_1, N_2, which is $I = N_1 \cap N_2$. As can be seen in Figure 5, the blue region B corresponds to the critical volume which has to be checked when h is not known. This volume encloses the red region R (collision volume) and is enclosed by the intersection set I: $R \subset B \subset I$. Therefore, all collisions of type 2 can be found by distance checking the child with all $p^{s_{j<i}} \in I$. Since the number of points in I is relatively small, the fact that I is larger than R is not very time critical.

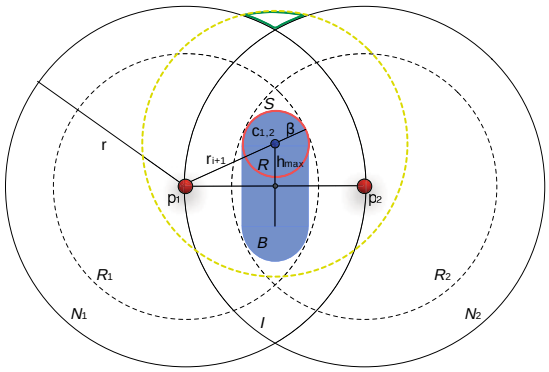

Figure 5: *Red: Collision region R for a child point in the case where $h = h_{max}$. Blue: Critical region B which has to be checked for collisions when h is not known. Dashed yellow: A new point p^{s_i} registers at all points $p^{s_{j<i}} \in N_1 \cup N_2$ if point is inside that region (except at points in the green region). Dashed black: R_1, R_2: Registered point sets of p_1 and p_2.*

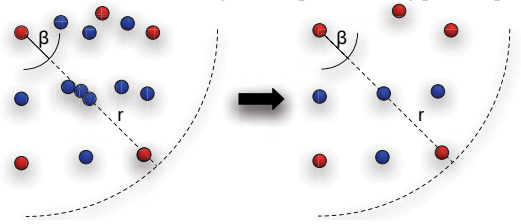

Figure 6: *Refined points without and with using our point collision avoidance algorithm. The red points were added in s_0 and the blue points in s_1.*

However, this is different for the third type, since the number of p^{s_i} is growing disproportionally. It turned out that an efficient way to solve type 3 collisions is to register an added child point at all points $p^{s_{j<i}} \in (N_1 \cup N_2)$ which are closer than $r_{i+1} + \beta$. For $c_{1,2}$ in Figure 5 these points lie all in the yellow dashed region. As a result, each $p^{s_{j<i}}$ knows the newly added points p^{s_i} inside the radius $r_{i+1} + \beta$. For the parent points p_1 and p_2 these registered point sets are illustrated by the black dashed regions R_1 and R_2. To check for case three collisions, we investigate the distances of each point in the intersection set S of R_1 and R_2 of both parents, thus $S = R_1 \cap R_2$. S fully encloses the blue region: $S \supset B \supset R$.

As can be seen in Figure 5, it cannot be guaranteed that all collisions are found. A new child does not register in the green region which is outside of the union $U = N_1 \cup N_2$. But since this volume is very small and is getting smaller or even disappears the smaller h is, this method is a good approximation and almost all collisions are avoided. Nevertheless, by using a smaller collision distance β or reducing h_{max} it is possible to detect and avoid all collisions.

3.3.2. Collision Handling

When we detect that a child point c collides with an existing point p and therefore is rejected, we move p into the direction of c in order to achieve that p is more equally distanced to the nearby points. This is done by averaging the positions and normals:

$$\mathbf{r_p} = \frac{\mathbf{r_p} w_p + \mathbf{r_c} w_c}{w_p + w_c} \tag{17}$$

$$\mathbf{n_p} = \frac{\mathbf{n_p} w_p + \mathbf{n_c} w_c}{\|\mathbf{n_p} w_p + \mathbf{n_c} w_c\|}, \tag{18}$$

where w is a weight which is assigned to a point at the time of creation. We chose this weight to be proportional to the distance from its parents, where a high weight is assigned to all $p \in P_0$ in order to preserve the original surface geometry. After a collision is detected, the weight of the point which is moved in the process is adjusted as well:

$$w_p = w_p + w_c. \tag{19}$$

This averaging positively affects the uniformity of the points after the refinement.

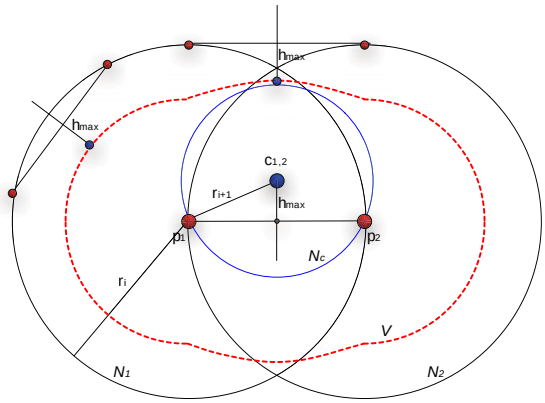

Figure 7: *A point created by two parent points outside U are always outside the red region V. As V fully encloses the child's neighborhood N_c of the next refinement step (blue) all candidate neighbor points are known and neighborhoods can be updated correctly.*

3.4. Neighborhood Update

To execute the next refinement step s_{i+1}, the new neighborhoods defined by r_{i+1} have to be determined for each existing point. Instead of using a search data structure and recomputing the neighborhoods in each step, we iteratively update the neighbors of each point which is less expensive concerning computation time and memory usage while maintaining correctness.

Neighborhood updates are done simultaneously to the registering described in Section 3.3 as in both steps we have to access all points $p^{s_{j<i}} \in U$. In order to save computation time, the distances which were already calculated in the collision test are reused, and neighborhoods are not updated after the last refinement step. We distinguish between three different neighbor relations (note that these are symmetric since the same r is used for all points):

1. Two points created in one of the last refinement steps $s_{j<i}$ are neighbors.
2. A point created in one of the last refinement steps $s_{j<i}$ and a point created in the current refinement step s_i are neighbors.
3. Two points created in the current refinement step s_i are neighbors.

The update of type 1 neighborhoods is straightforward: as $N_i^{s_{i+1}} \subset N_i^{s_i}$ and the distance to each neighbor is already known from the last refinement step, we can update the neighborhoods cheaply by comparing the distance to r_{i+1}.

Neighbor relations of type 2 are generated after having checked a child point for collisions. When a child does not collide with any other point and therefore is accepted, it is added to the neighborhood of all $p^{s_{j<i}} \in U | d \leq r_{i+1}$, and vice versa.

Additionally, the child has to be added to all points $p^{s_i} | d \leq r_{i+1}$ (type 3 relations). Figure 7 illustrates that it is sufficient to know all $p^{s_{j<i}} \in U$ to find all neighbor candidate points p^{s_i}, as long as each point $p^{s_{j<i}}$ knows the added children where it was involved as a parent, and therefore correct neighborhoods can be guaranteed. As we know h_{max}, r_{i+1}, and the maximal distance between two parent points $d_{max} = r_i$, we can show geometrically that it is not possible that two points $p^{s_{j<i}}$ both outside of U are refined yielding a new point which is closer than r_{i+1} to the child $c_{1,2}$. All points created by such parents would lie outside of the red dashed region V illustrated in Figure 7. As can be seen, V completely encloses the neighborhood N_c^{i+1} defined by r_{i+1} (illustrated in blue), even in the most extreme case where $h_{child} = h_{max}$: $V \supset N_c^{i+1}$. This means that all neighbor candidate points p^{s_i} are known by the child and can be distance checked.

4. Results and Discussion

We have tested our new refinement method on different irregularly sampled point scenes and rendered them either using the raytracing approach presented in [SSP07] or point splatting. All timings are given for an Intel Core2 2.66 GHz.

Three frames of a splashing column simulation consisting of 3k physics particles are shown in Figure 8. During the whole simulation, a point generation rate between 415k and 845k points per second is achieved (Table 1). A simulation sequence running at interactive rates is demonstrated in Figure 9, where the upsampled fluid with 14k surface points is running at 11fps (41 time steps per second) and the initial fluid with 1k surface points at 17fps (58 time steps per second). These timings include all computational costs (physics, normals, refinement, and visualization).

The effect of upsampling a textured fluid can be seen in Figure 10. Whereas the initial fluid appears blurry, the upsampled fluid is much sharper and detail-conserving. The introduced smoothing related to the original texture is visualized in the bottom row, where red and yellow correspond to low and high smoothing, respectively.

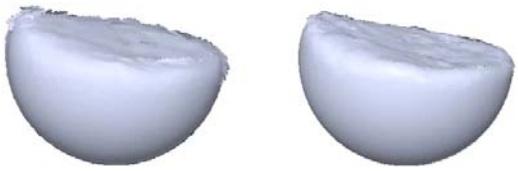

Figure 9: *Initial and refined points (left: 1k, right: 14k) simulated and rendered at 17fps and 11fps, respectively.*

Figure 10: *Left: inital points (12k). Right: upsampled points (140k). Bottom row: smoothing related to the original texture.*

In Figure 11, we applied our refinement method to 9k randomly chosen points of the static ball joint model (140k points), yielding 50k points. The initial and the upsampled points are visualized using point splatting. As can be seen, more surface details are visible after upsampling and artifacts at the silhouette can be reduced. The quality of the splatting could be further improved by optimally determining the splat radius for each point, whereas in our examples we use a constant radius for all points.

Currently, we reach a point generation rate of up to 34k points per frame at 25fps. It is possible to improve the computation times or the visual quality even further by integrating more sophisticated methods to optimally select the points which are going to be refined. Selection operators based on curvature and level of detail information could be applied, additionally, the computation costs of the refinement could be approximately halved by omitting the upsampling of occluded points. While we present the performance of our algorithm on irregular point samples it is to be noted that it can be easily applied to regular point samples as well. In fact, regular point samples facilitate the point generation process and actually improve the performance as the refinement radii can be smaller, eliminating many of the potential point collisions.

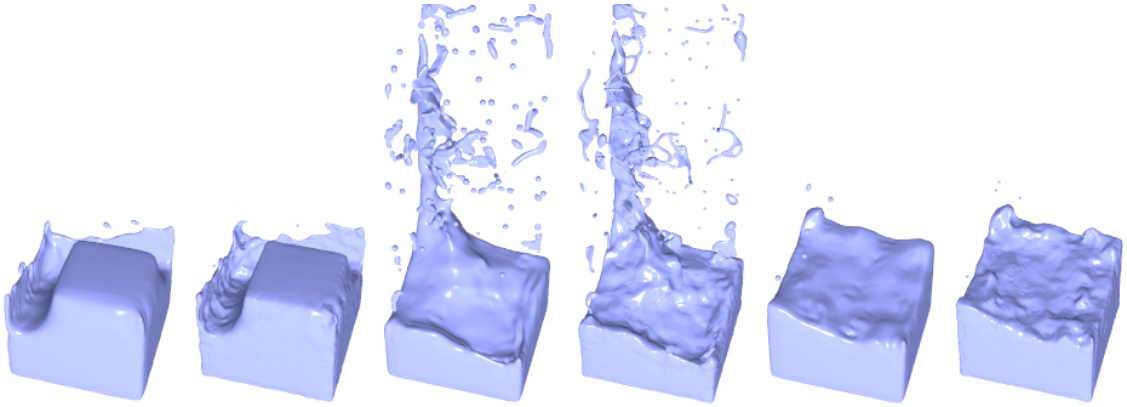

Figure 8: *Three frames of the column splashing simulation. The left and right image of each pair show the raytraced surface of the inital surface points and the surface after 3 refinement steps, respectively.*

Figure 11: *From left to right: splatted surface of the original point model (140k), initial points (9k) randomly chosen from the original model, upsampled points (50k), splatted surface of the initial point set, splatted surface of the upsampled point set.*

	P_0	P_3	$t_{refine}[s]$	$points/s$
Initial block	1'208	42'342	0.05	846'840
Splashing	1'916	103'840	0.25	415'360
Equilibrium	1'359	35'547	0.07	507'814

Table 1: *Point numbers and performances of 3 individual frames of the column splashing simulation sequence.*

Certain ring and ribbon like features are visible in sparse particle regions. While these might look unrealistic, it is not clear to us how they can be avoided in a consistent way. Furthermore, our refinement technique does not make use of temporal coherence. This may lead to problems in splashing areas where isolated particles merge and split and the rendered topology might undergo sudden changes. Although the integration of time-coherent aspects would reduce this problem, it would come at the expense of processing time as connectivity information would need to be stored and reused in each simulation step. As processing speed was one of our major constraint, temporal coherence is currently not integrated in our implementation.

5. Conclusion

We have presented a refinement method which is suitable to efficiently improve the visual quality of low resolution particle-based fluids used in interactive and real-time applications. Our algorithm is able to robustly detect and refine surface points, particularly if the input points are irregularly sampled dynamic points coming for example from an SPH fluid simulation. No preprocessing is necessary and due to our fast neighborhood update we can reduce computation and memory costs. Our interpolation method can handle complex surfaces and splashes, and features like edges can be preserved. Point collisions are avoided yielding a nearly uniform point distribution, which also supports surface reconstruction techniques. Our algorithm can generate up to 34k points per frame at 25fps which makes it suitable to apply the technique to the computationally expensive field of fluid simulation. As was shown, our algorithm improves the visual quality of the surface while preserving surface details. As future work, we intend to integrate a more robust normal computation, as the low quality normals currently used adversely affect the reconstructed surfaces. Additionally, we intend to use more sophisticated methods to optimally select the points which are going to be refined in order to reduce the computation costs even further.

References

[AA03] ADAMSON A., ALEXA M.: Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics Symposium on Geometry Processing* (2003), pp. 230–239. 2

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C.: Computing and rendering point set surfaces. *IEEE Transactions on Computer Graphics and Visualization 1*, 9 (2003), 3–15. 1, 2

[AD03] ADAMS B., DUTRE P.: Interactive boolean operations on surfel-bounded solids. In *Proceedings of ACM SIGGRAPH* (2003), pp. 651–656. 1

[Age05] AGEIA: *Physics, Gameplay and the Physics Processing Unit,*. White paper, 2005. 1

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. In *ACM SIGGRAPH* (2007). 2

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of ACM SIGGRAPH* (2001), pp. 67–76. 2

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation* (1996), pp. 61–76. 2

[GBP04] GUENNEBAUD G., BARTHE L., PAULIN M.: Dynamic surfel set refinement for high quality rendering. *Computer and Graphics 28*, 6 (2004), 827–838. 1, 2, 3

[GBP05] GUENNEBAUD G., BARTHE L., PAULIN M.: Interpolatory refinement for real-time processing of point-based geometry. *Computer Graphics Forum, Eurographics 2005 conference proceedings 24*, 3 (2005), 657–667. 1, 2, 3

[GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. In *ACM SIGGRAPH* (2007). 2

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRE P., GROSS M.: A unified lagrangian approach to solid-fluid animation. In *Proceedings of Eurographics Symposium on Point-Based Graphics* (2005), pp. 125–133. 2

[Lev03] LEVIN D.: Mesh-independent surface interpolation. *In Geometric Modeling for Scientific Visualization* (2003), 37–49. 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Symposium on Computer Animation* (2003), pp. 154–159. 1, 2, 3

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. In *Proceedings of ACM SIGGRAPH* (2005), pp. 471–478. 1

[MKI03] MONAGHAN J., KOS A., ISSA N.: Fluid motion generated by impact. *Journal of Waterway, Port, Coastal and Ocean Engineering 129* (2003), 250–259. 2

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Symposium on Computer Animation* (2004), pp. 141–151. 1, 2

[Mon89] MONAGHAN J.: On the problem of penetration in particle methods. *Comput. Phys. 81* (1989), 1–15. 1

[Mon92] MONAGHAN J.: Smoothed particle hydrodynamics. *Annu. Rev. Astron. Physics 30* (1992), 543. 1, 2

[Mon05] MONAGHAN J.: Smoothed particle hydrodynamics. *Rep. Prog. Phys. 68* (2005), 1703–1759. 2

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Symposium on Computer Animation* (2005), pp. 237–244. 1, 2

[MST*04] MÜLLER M., SCHIRM S., TESCHNER M., HEIDELBERGER B., GROSS M.: Interaction of fluids with deformable solids. *Journal of Computer Animation and Virtual Worlds 15*, 3-4 (2004), 159–171. 1, 2

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.: Multi-level partition of unity implicits. *ACM Transaction on Graphics 3*, 22 (2003), 463–470. 2

[Paj03] PAJAROLA R.: Efficient level-of-details for point based rendering. In *Proceedings IASTED Invernational Conference on Computer Graphics and Imaging* (2003). 3

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient simplification of point-sampled surfaces. In *Proceedings of IEEE Visualization* (2002), pp. 163–170. 1, 2

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRE P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3 (2005), 957–964. 1

[PKG06] PAULY M., KOBBELT L. P., GROSS M.: Point-based multiscale surface representation. *ACM Trans. Graph. 25*, 2 (2006), 177–193. 1

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modeling with point-sampled geometry. In *Proceedings of ACM SIGGRAPH* (2003), pp. 641–650. 1, 2

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. In *Proceedings of Eurographics* (2003), pp. 401–410. 2

[SAC*99] STORA D., AGLIATI P., CANI M. P., NEYRET F., GASCUEL J.: Animating lava flows. In *Graphics Interface* (1999), pp. 203–210. 2

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid-solid interactions. *Journal of Computer Animation and Virtual Worlds 18*, 1 (2007), 69–82. 1, 2, 6

[TKPR06] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. In *Symposium on Computer Animation* (2006), pp. 7–15. 2

[TRS04] TOBOR I., REUTER P., SCHLICK C.: Multiresolution reconstruction of implicit surfaces with attributes from large unorganized point sets. In *Proceedings of Shape Modelling International* (2004), pp. 19–30. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (2005), 965–972. 2

[ZPKG02] ZWICKER M., PAULY M., KNOLL O., GROSS M.: Pointshop 3d: an interactive system for point-based surface editing. In *Proceedings of Computer graphics and interactive techniques* (2002), pp. 322–329. 1