Volume-Surface Trees

Tamy Boubekeur, Wolfgang Heidrich, Xavier Granier, Christophe Schlick

EUROGRAPHICS 2006 / E. Gröller and L. Szirmay-Kalos
(Guest Editors)

Volume 25 (2006), Number 3

Volume-Surface Trees

Tamy Boubekeur* Wolfgang Heidrich+ Xavier Granier* Christophe Schlick*

*: LaBRI - INRIA - CNRS - University of Bordeaux +:University of British Columbia

Abstract

Many algorithms in computer graphics improve their efficiency by using Hierarchical Space Subdivision Schemes (HS^3), such as octrees, kD-trees or BSP trees. Such HS^3 usually provide an axis-aligned subdivision of the 3D space embedding a scene or an object. However, the purely volume-based behavior of these schemes often leads to strongly imbalanced surface clustering. In this article, we introduce the VS-Tree, an alternative HS^3 providing efficient and accurate surface-based hierarchical clustering via a combination of a global 3D decomposition at coarse subdivision levels, and a local 2D decomposition at fine levels near the surface. First, we show how to efficiently construct VS-Trees over meshes and point-based surfaces, and analyze the improvement it offers for cluster-based surface simplification methods. Then we propose a new surface reconstruction algorithm based on the volume-surface classification of the VS-Tree. This new algorithm is faster than state-of-the-art reconstruction methods and provides a final semi-regular mesh comparable to the output of remeshing algorithms.

1. Introduction

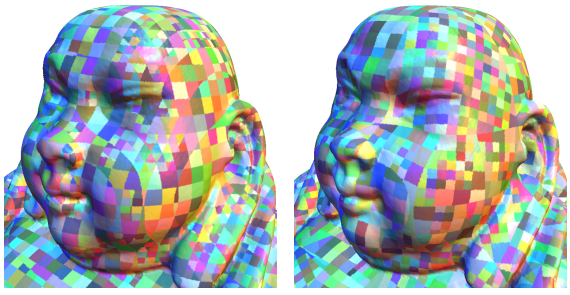
Hierarchical Space Subdivision Schemes (HS^3) are ubiquitous in computer graphics: simplification, reconstruction, compression, visibility, and many other processing steps are based on trees to partition and structure data sets. The initial space, often an axis aligned bounding box, is recursively subdivided until each cell satisfies a given error criterion. The root cell of the HS^3 can be either globally associated with the whole scene, or locally with each single object. Some of the most popular HS^3 are octrees, kD-Trees and axis-aligned BSP-Trees, which are easy to implement and to integrate in existing computer graphics frameworks.

Nevertheless, in the case of 3D surfaces, while HS^3 generate satisfying clustering at coarse subdivision levels, it is obvious that at finer levels, when the cells come closer to the surface, volume-based decomposition leads to imbalanced clustering in areas where the surface is not aligned with the main directions of the data structure (see Figure 1(a)).

In this article, we propose an alternative HS^3 which combines a 3D scheme for the first levels of the tree, and a 2D scheme as soon as the surface can be projected onto a plane without folding. We call such a tree a *Volume-Surface Tree* (or **VS-Tree**, for short). We show that VS-Trees achieve efficient and elegant surface-based partitioning that can be applied to a variety of applications, such as *surface simplification* and *surface reconstruction*. In the first part of this paper, we review previous work (Section 2) before introducing VS-Trees (Section 3) and show how this structure improves prior surface simplification algorithms based on *hierarchical vertex clustering* (Section 4). In the second part of the paper (Section 5), we describe a new efficient surface reconstruction algorithm based on the specific volume-surface behavior of VS-Trees.

2. Related Work

Hierarchical Space Subdivision Schemes: All HS^3 are based on a recursive subdivision of a root cell, as long as



(a) Octree clustering (b) VS-Tree clustering

Figure 1: *Comparison between (a) octree clustering and (b) VS-Tree clustering. The local 2D scheme used by VS-Trees produces much better alignment of clusters and reduces the total number of clusters within a given error bound.*

some user-specified criterion is not satisfied in every subcell. As outlined above, octrees, kD-Trees and BSP-Trees are by far the most popular HS^3. In the case of BSP Trees [FKN80], the space subdivision is diadic, using a simple split plane, often chosen axis-aligned for the sack of efficiency. The kD-Tree data structure [Ben75] performs orthogonal space separation and stores additional data elements at internal nodes. Finally, quadtrees and octrees [JT80, Sam89] express the dimension of the subdivided space directly in their structure: a 1-to-4 scheme for quadtrees in 2D, and a 1-to-8 scheme for octrees in 3D. The very simple construction of the octree, as well as its fast convergence toward the shape of the embedded 3D surface, makes it very popular when geometry processing methods, such as *surface simplification* and *surface reconstruction*, need to be scaled toward large data sets.

Simplification by Clustering: The goal of simplification methods is to reduce the resolution of an object, while maintaining as much detail as possible from the original shape [HDD*93, GH97, CSAD04]. Clustering methods are a particular subset of simplification techniques, which cast the problem as a partitioning problem, where each partition only keeps one single sample that minimizes the error, in a given metric [GH97,CSAD04], with the original surface. Hierarchical approaches, such as BSP-based methods [SG01] or octree-based methods [SW03], provide adaptivity in the surface partitioning. This adaptivity allows for more accurate simplification of non-uniformly sampled surfaces than regular grid partitioning methods [RB93, Lin00], while remaining almost as efficient. Such techniques have originally been developed for meshes, but they can also be directly applied on point clouds, when the sampling density is high enough [PGK02]. In practice, it appears that the quality of the mesh simplified by hierarchical clustering is strongly related to the subdivision scheme, and we will show how the local 2D scheme used by VS-Trees offers a much more regular sample decimation than the 3D scheme induced by octrees (see Section 4).

Surface Reconstruction: The surface reconstruction problem arises in many applications such as processing of real-world range scanned objects, surface deformation, conversion between different surface representations, or even remeshing. To be as generic as possible, surface reconstruction techniques usually start from a sampling of the original surface in the form of a point cloud. Note that in addition to its position, each sample may also carry additional information, such as normal vector, that may (or may not) be exploited during the reconstruction. Since the seminal work of Hoppe et al. [HDD*92, HDD*94], various surface reconstruction approaches have been proposed in the literature, either based on the Delaunay triangulation [BC00, GKS00, ACK01, DGH01], deformable models [DQ01, DYQS04], implicit surfaces [CBC*01, TO02] or displaced subdivision surfaces [STKK99, JK02].

Recent advances in 3D acquisition techniques have dramatically improved the size and the density of available point clouds. In order to manage the intrinsic computational complexity of surface reconstruction from such huge point clouds, hierarchical data structures have been introduced. For instance, an implicit surface reconstruction can be obtained by splitting the input point cloud with an octree, computing a separate implicit surface for each leaf of the octree, and finally *blending* together the set of local implicit surfaces by using the *Partition Of Unity* method. This process has been successfully used for fast local polynomial fitting in the *Multi-Level Partition of Unity Implicits (MPU)* [OBA*03] as well as for *Radial Basis Functions* [TRS04]. Unfortunately, to get an efficient visualization by graphics hardware, the reconstructed implicit surface has to be converted into a mesh, which involves an expensive tessellation step [Blo94]. Moreover, the quality of the resulting mesh is generally poor and has to be improved using, for instance, an additional remeshing step, either based on parameterization [LSS*98], or fitting of subdivision surfaces [HDD*93, EDD*95, ZSS97, MK04]. Consequently, even if the computation of the implicit surface is efficient thanks to the space subdivision, the whole reconstruction process including the generation of an high quality semi-regular mesh becomes rather expensive.

Instead of using a completely automated reconstruction process that is likely to fail in some pathological zones of an object (e.g., undersampled areas, high curvature areas), we believe that the quality of surface reconstruction can be greatly improved by using an interactive user-controlled process. Obviously, aiming for user interaction implies that the whole process has to be extremely efficient. Thus, we propose here a performance-oriented surface reconstruction technique that takes fully benefit of the volume-surface organization of the VS-Tree, to generate a semi-regular mesh of arbitrary genus over an unorganized point-cloud, dealing both with noise and non-uniform sampling (see Section 5).

3. VS-Trees

3.1. Definition

A VS-Tree is a *surface-based* HS^3. The basic idea is to combine an octree and a set of quadtrees to describe a discrete 3D surface. During the recursive split involved in the octree construction, we switch to a quadtree as soon as the area of the surface associated with the current node is consistent with a scalar-valued function over a given ground plane (in other words, a *height field*). Figure 2 presents the three different kinds of VS-Tree nodes:

- **Volume Nodes** (V-Nodes): comparable to octree nodes. Each V-Node has 0 or 8 children, which can be V-Nodes or T-Nodes.
- **Transition Nodes** (T-Nodes): leaves of the 3D hierarchy which also are roots of the 2D hierarchies. Each T-Node has 0 or 4 children that are S-Nodes.

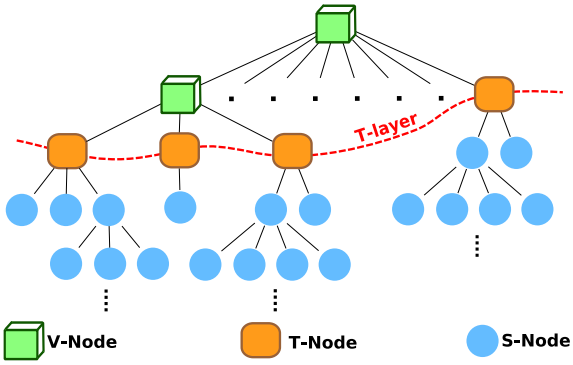


Figure 2: *VS-Tree structure.*

- **Surface Nodes** (S-Nodes): comparable to quadtree nodes. Each S-Node has 0 or 4 children that are S-Nodes.

Note that each T-Node carries a local frame that is used to align its corresponding sub-quadtree. The union of all T-Nodes defines the volumetric layer under which it becomes possible to implement 2D algorithms (see Figure 2); we call it the *Transition-layer* or **T-layer**.

VS-Trees are proposed in order to increase efficiency of geometric processing usually combined with simple and efficient hierarchical structures such as octrees. In order to maintain a behavior as similar as possible to octrees, the ideal structure should have the following properties:

- Purely recursive construction: popular hierarchical structures have the strong advantage to be instanced through a simple recursive call, which is easy to implement;
- Efficient construction: rigid organization of data, such as the 1-to-8 split of octrees, allows efficient traversal and refinement of an hierarchical structure;
- T-layer at low depth: switching to quadtrees as soon as possible reduces the memory overhead thanks to the 2 dimensional structure, and speeds-up traversals and tests. Inclusion tests for arbitrary points are performed in 2D using the quadtrees placed under the T-layer;
- Graceful degradation: in the worst case of very small or under-sampled topological features, such as iso-surface extraction from physical simulation, the structure should behave no worse than an octree.

3.2. Construction

There are a large number of possible 3D surface decompositions that lead to a collection of 2.5D pieces. We propose to use the following simple recursive construction method that is easy to integrate in existing application softwares.

Input: Let S be the set of samples defining the input surface. Each sample s_i of S is defined by a position p_i and a normal vector n_i. For dense meshes, S can be chosen, for instance, as the original vertices of the mesh, or as the barycenters of the polygons. S can also be a point-based surface, with normals approximated with a Principal Component Analysis (PCA) [HDD*92, GKS00] if not available.

Clustering: The construction of a VS-Tree begins with the computation of a bounding box B of S recursively subdivided with a 1-to-8 octree scheme. At each level, the current set of samples S_i associated with the bounding box B_i at that level is classified against each child's bounding box. Let κ be a *height field indicator*, signaling whether S_i is consistent with a height field (i.e., it is 2.5D rather than 3D). This flag stops the recursive 1-to-8 subdivision process. When $\kappa(S_i)$ is true, the current node is set as a T-Node, and a local coordinate frame is computed. This local frame will strongly influence the final quality of the clustering, and must be carefully chosen. While it is generally impossible for a hierarchical structure to precisely recover all the anisotropic features present in the discrete surface, a well-aligned sub-hierarchy can often be computed by analyzing the underlying surface and considering its main directions (see Figure 1 and Figure 4). Thus, for constructing this local frame, we use a PCA on S_i, but rather than considering positions of samples [HDD*92, GKS00], we use their associated normals, a more relevant information when clustering surfaces [CSAD04].

Since we are looking for directions, we can perform the PCA in the normal space of S_i. The set of resulting eigenvectors is a good approximation of the tangent frame of the surface. We choose $\{n_i, u_i, v_i\}$ as a local frame, where n_i is the average

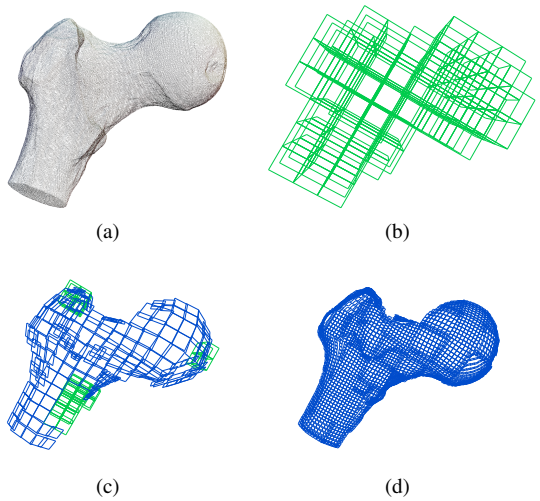


Figure 3: *Different levels of a VS-Tree. (a) The input discrete surface. (b) The upper levels of the tree are three-dimensional (in green). (c) The transition between 3D and 2D structure (in blue) is possible as soon as the surface can be locally expressed as a height field. (d) The lower levels of the VS-Tree are two-dimensional.*

Models	Samples	Octree	VS-Tree	Gain
Feline	49864	0.18s.	0.19s.	18%
Igea	134346	0.33s.	0.34s.	49%
Vase lion	200002	0.83s.	0.80s.	18%
Raptor	1000080	3.50s.	3.11s.	39%
XYZ dragon	3609601	11.82s.	9.88s.	52%
XYZ Statue	5000000	17.82s.	14.90s.	32%

Table 1: *Computation time to generate the HS^3 with L^2 error bounded at 10^{-4}. The gain is relative to the final number of partitions.*

normal of S_i, while u_i and v_i are the normalized projections of the two eigenvectors that minimize the dot product with n_i onto the plane Π_i defined by n_i and c_i (the centroid of S_i).

The set of samples S_i associated with the T-Node T_i is projected on Π_i. Finally, a bounding quad is computed for S_i and is recursively subdivided with a 1-to-4 quadtree scheme. The recursion is stopped when the error, computed over S_i, is below a threshold. Figure 3 shows the different steps involved in this construction. Note that the T-layer becomes independent of the discrete surface resolution when the sampling density is sufficient: typically, over-tessellating a dense mesh will not change the depth of the T-layer.

Height field indicator: Evaluating if a piece of surface will exhibit folding during a lower dimensional projection can be done by numerically integrating the curvature over this area. Nevertheless, such a test is computationally expensive even in the case of regular meshes, and more complicated for non-manifold meshes or topology-free representations such as point clouds. In order to make our approach more general and efficient, various heuristics can be used to define the height field indicator κ for such a predicate. Pauly et al. [PG01] propose a normal-cone test for allowing the projection of a set of surfels using the miniball algorithm. Boubekeur et al. [BRS05] extend this idea by introducing an additional displacement threshold to detect scan misalignment in dense acquired point sets. Although a formal proof is not available, since it would depend on some form of density and/or topology criterion, this indicator κ gives convincing results in practice. So we define κ to be *true* when:

$$\forall j \in [0,k_i[\left\{ \begin{array}{ll} n_{ij} \cdot n_i & > \delta_a \text{with } \delta_a \in [0,1] \text{ and} \\ \frac{|(p_{ij}-c_i)\cdot n_i|}{\max_{k_i}(||p_{ik_i}-c_i||)} & < \delta_d \text{with } \delta_d \in [0,1] \end{array} \right.$$

where k_i is the number of samples of the current cell i, n_i the average normal of the surfels in the cell, p_{ij} and n_{ij} are the position and the normal of the j^{th} sample of the cell i. δ_a (angle deviation) and δ_d (displacement deviation) are user-provided thresholds. In our implement, $\delta_a = 0$ and $\delta_d = 1/6$ has provided satisfying results in all our tests. Note that by increasing δ_a and decreasing δ_d, it becomes harder for κ to be true, and thus the T-layer is conservatively dropped to a lower level of the hierarchy.

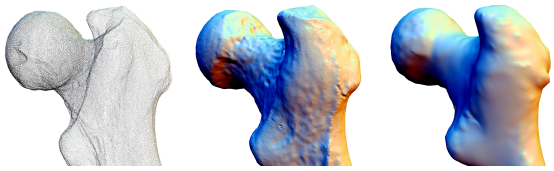
Error metrics: As usual with HS^3, an error metric ε has to be defined to control the recursive subdivision. A good error function should be monotonic and decreasing with the size of S_i. Geometric error metrics such as the L^2 metric used in the Quadratic Error Function [GH97] or the normal-based $L^{2,1}$ metric [CSAD04] are used to drive our VS-Tree construction. More complex combined metrics, such as the Sobolev one, may also be used. In the case of large objects, simple approximated metrics, such as the local density, may be chosen for efficiency.

4. Clustering and Application to Simplification

Balanced clustering: Figure 1 illustrates the difference of vertex clustering obtained with an octree and a VS-Tree. The volume-based behavior of octree decomposition frequently leads to very imbalanced clustering, mixing small clusters (when the surface is located near the corner of the octree cell) and large ones (when the surface passes near the center of the octree cell). Moreover, the *cuts* generated by the octree cell boundaries can be clearly identified within the clustering (see Figure 1(a)). VS-Tree decomposition strongly reduces both artifacts, as it provides a much better alignment of the cluster boundaries with the embedded surface (see Figure 1(b)). A very low variance can be observed in the size of the clusters, basically because the clustering only depends on the planarity, but not on the orientation, of the surface locally associated with each T-Node. For instance, the variance in the number of samples per cluster has almost been divided by 2 between Figure 1(a) and Figure 1(b). Additionally, an almost regular quad-like clustering can be observed. The few remaining non-quad clusters primarily come from the volume-based decomposition created at the top levels of the VS-Tree.

Computation efficiency: In addition to providing more balanced clustering, the VS-Tree is also more efficient than the octree when computing the HS^3. Moreover the advantage of the VS-Tree over the octree increases with the size of the input data, as shown on Table 1. For large objects, a 16% improvement can be observed in the computation time, as well as a reduction of the number of clusters between 18% and 52% for the same bounded error. This may appear quite surprising as octree decomposition is generally considered extremely efficient. In fact, the speedup observed by VS-Tree decomposition comes from two different properties. First, when the size of the input data increases (e.g., very dense

Figure 4: *Hierarchical mesh simplification with L^2 error bounded at 2.10^{-3}. Left: Original object (7M triangles). Middle: Octree simplification (1.75 sec. - 62856 triangles). Right: VS-Tree simplification (1.20 sec. - 52024 triangles).*



(a) (b) (c)

Figure 5: *Noise filtering. (a) Input point cloud (*137063 *samples). (b) Reconstruction with VS-Tree L^2 error bounded at* 10^{-4} *(*1.758 *sec, 125K triangles). (c) Reconstruction with VS-Tree L^2 error bounded at* 10^{-3} *(*0.987 *sec, 32K triangles).*

meshes), most of the data will be represented below the T-layer, and thus 1-to-4 splits will be much more frequent than 1-to-8 splits. To reach a given error threshold, the octree is thus usually much deeper, with significantly more empty cells compared to the corresponding VS-Tree. Second, below the T-layer, all the computations involved in the VS-Tree are done in 2D. When there is a large number of points in the sub-hierarchy of a given T-Node, these 2D computations more than compensate for the overhead involved in the projection to the local 2D frame.

Mesh simplification: We have implemented a mesh simplification algorithm similar to the octree clustering introduced by Schaefer and Warren [SW03], simply by replacing the octree with our VS-Tree. Here again, the more balanced cluster sizes provided by the VS-Tree reduce the mismatch of features for a given error threshold, without imposing an overly conservative mesh density. Moreover, the local frame computed independently for each T-Node roughly captures the anisotropy of the underlying mesh, while the octree completely ignores it. For instance, see the cheek on Figure 4. As expected, the VS-Tree introduces fewer clustering artifacts in the mesh topology, and better captures the original geometry (see near the eye, for instance).

5. Application to Fast Surface Reconstruction

5.1. Problem Analysis

Obviously, meshes have become the de-facto standard for 3D geometry processing and rendering. Development of 3D acquisition techniques, which have become widely available in recent years, has dramatically increased the need for robust and efficient point-to-mesh surface reconstruction techniques. Several mandatory properties for such a reconstruction processes have been introduced by recent work: (1) dealing with arbitrary genus; (2) offering intuitive de-noising control; (3) avoiding final remeshing by directly providing a semi-regular mesh; (4) providing error controlled output; and, of course, (5) being as efficient as possible. In this section, we propose to use the advantages of the VS-Tree decomposition in order to develop a point-to-mesh reconstruction technique that fulfills these five properties.

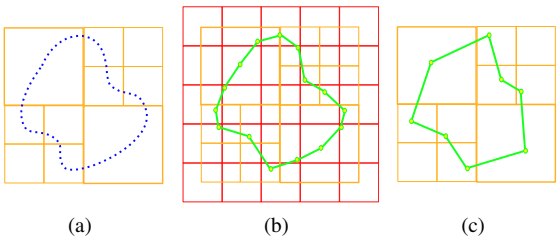
Intuitively, most of the global topological features of the surface can be recovered at the T-Layer of the VS-Tree. Thus, the T-Layer helps us to split the problem into two main steps: a coarse mesh M_0 is generated during the first step, and refined during the second step, to account for all the details included in the input point cloud. This second step uses a displacement process driven by the quadtree corresponding to each T-Node. The following algorithm summarizes our approach:

```
Mesh POINT_TO_MESH (PointSet S, float Threshold)
    VSTree T := buildVSTree (S);
    Mesh M := extractMeshAtTLayer (T);
    while (error (M, S) > Threshold) do
        M := refinePN (M);
        M := displace (M, T);
    return M
```

5.2. VS-Tree Based Surface Reconstruction

VS-Tree construction: Globally, we follow the construction process presented in Section 3. Similar to Pauly et al. [PGK02], the high frequency noise typically present in scanned data [NRDR05], is directly addressed at the *point* level by simply specifying an L^2 error threshold driving the VS-Tree creation. While more formal noise removal solutions exist [PG01, SFS05], this simple technique nicely smoothes out the noise, as shown on Figure 5, and is intuitive enough to be easily tuned by the user.

Base mesh reconstruction: The remainder of the algorithm will inherit the global topology of M_0, and in particular its genus. Since the geometry of S-Nodes does not change the global topology of the shape, M_0 is created using only the T-Layer (see Figure 6(a)). However, the set of T-Nodes composing the T-Layer can be sparse (e.g., large areas with low curvature), which does not allow the use of Delaunay-based reconstructions for this base-mesh. Moreover, ideally, we would like a watertight 2-manifold, homomorphic to the input point-based surface. This naturally leads us to choose a simple implicit surface reconstruction, by just considering the half space defined by the oriented frame of each T-Node (i.e., a linear polynomial acting as a distance function) and contouring it in similar fashion to Hoppe et al. [HDD*92].



(a) (b) (c)

Figure 6: *Coarse mesh generation. (a) Input point cloud (in blue) clustered in a VS-Tree (T-Layer in orange). (b) Marching cube dual contouring at the resolution of the deepest T-Node. (d) Coarse mesh M_0 generated by simplifying the mesh at the T-Layer level.*

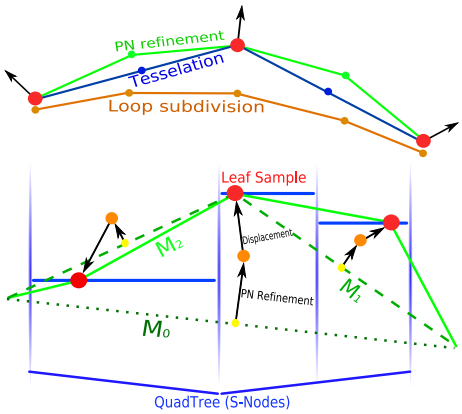


Figure 7: *Top: vertex insertion comparison. Bottom: VS-Tree refinement and displacement in a T-Node.*

However rather than directly contouring this simple localized distance function with a marching cube algorithm, we rather construct a smooth implicit surface using a *Partition of Unity* scheme. The octree structure of the upper levels of the VS-Tree allows consistent generation of overlapping zones that can be used to blend the local distance functions, in a similar fashion as the quadrics used by Ohtake et al. [OBA*03]. The mesh is then generated by applying a Bloomenthal polygonization [Blo94]. In order to guarantee that no topological feature of the VS-Tree will be missed, we use a dual contouring grid and set its resolution to that of the deepest T-Node (see Figure 6(b)). Finally, this mesh is simplified by clustering it at the T-Layer level. This leads to the final coarse mesh M_0, which contains only one vertex for each T-Node (see Figure 6(c)).

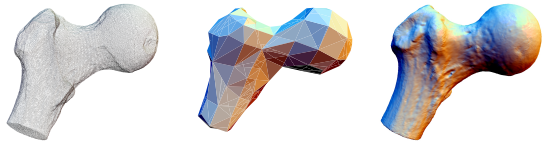
Mesh refinement: The goal of this second stage is to iteratively refine the mesh, until the geometric features of the input point cloud are recovered according to a given error threshold. For triangular meshes, the approximating subdivision scheme proposed by Loop [Loo87] is known to provide high quality mesh refinement. But in our quest for efficiency, we need to find a trade-off between speed and quality. We have found that local subdivision based on *Curved PN-Triangles* [VPBM01] are well suited to our constructive approach. This leads us to the following efficient two-step refinement technique:

1. each triangle of the mesh M_i is refined into four sub-triangles and the newly inserted mid-edge vertices are translated according to the cubic Bezier triangular patch computed by the PN-Triangle scheme;
2. these three mid-edge vertices are translated to their final position, according to the geometry stored in the local quadtree (see Figure 7).

This displacement procedure is the step that benefits most from the specific properties of our VS-Tree decomposition. Instead of having to define a smooth scalar field such as in implicit surface reconstruction methods, or a robust energy functional such as in dynamic model fitting [DYQS04], we simply use the quadtree defined at each T-Node to displace the inserted vertices accordingly. Let v denote an inserted vertex that has to be displaced. First, we find the highest S-Node s that only contains v. Then, we select the leaf l exhibiting the highest local variation in the quadtree built on s. Finally, we translate v toward the average sample carried by l. We mark l as locked, and will no more consider it for future displacement steps: as PN-Triangles provide an interpolating scheme, this vertex is now interpolated until the end of the refinement loop. This simple construction approximates the optimal displacement of v and avoids the mismatch of high-frequency features that would occur if a simple orthogonal displacement was performed (see Figure 7).

At each refinement step, the mesh is maintained *hole-free* since we only translate its vertices. In order to avoid the *surface aliasing* effect that could occur when many vertices are projected toward the same leaf, we do not displace v when no more leaf remains *unlocked* in the quadtree built on s. After each displacement pass, newly inserted vertices that have not been displaced are smoothed out according to the final position of neighbor vertices, using a simple tangential smoothing. Since the PN-refinement performs a 1-to-4 subdivision, each vertex v has at least two neighbors that have already been processed at a previous refinement step, and thus have reached their final position.

Adaptivity to curvature variation: In the case of point clouds sampled from a surface that exhibits large variations of curvature, one may think that an adaptive refinement scheme [ZSS97] would allow a better capture of the global shape. However, both the efficiency of vertex insertion, as well as the final semi-regular topology of the mesh, would be lost by such an adaptive refinement. Efficient adaptivity to curvature variation can be easily included in our scheme by letting the user tune the δ_a and δ_d thresholds used in the height field indicator κ. Indeed, increasing δ_a and decreasing δ_d induces a deeper T-layer in high-curvature areas and thus, a larger number of T-Nodes. Since, M_0 is generated by T-Node clustering, M_0 is itself denser, leading to a final mesh with higher resolution in high-curvature areas (see Figure 10(a)). Although this solution may break down for some pathological cases, it remains far less expensive than, for instance, the optimization of the L^∞ error [MK04]. Figure 5, 8, 9 and 11 shows some examples of surface reconstruction obtained with our algorithm.



(a) (b) (c)

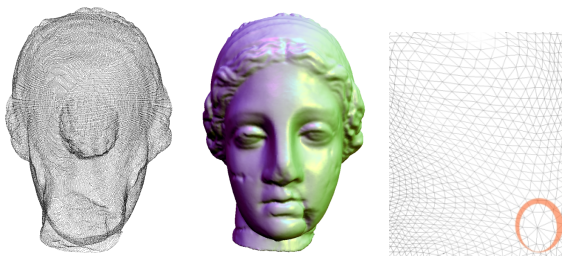
Figure 8: *Reconstruction of the ball-joint model (*137062 *points,* 1.758 *sec). (a) Input point set (b) Coarse mesh generated at the T-layer of the VS-Tree. (c) Final refined mesh.*

5.3. Performance

Table 2 provides some reconstruction timings for various models. The timing presented includes the VS-tree decomposition, the coarse mesh generation and the mesh refinement loop. Globally, this new algorithm is one order of magnitude faster than state-of-the-art *fast* surface reconstruction methods [OBA*03, GKS00], while directly providing a final mesh with semi-regular connectivity without any additional remeshing steps. For large point clouds, the VS-Tree construction becomes the bottleneck, since this is a non-linear operation. Figure 10(b) compares the final mesh quality of [OBA*03] to ours. In our implementation, the intensive use of pointers limits the size of in-core reconstruction. We are currently exploring out-of-core methods for performing the reconstruction with a constant and small amount of memory.

The mesh quality obtained by our technique is much higher as the one obtained by applying some octree-based tessellation on a reconstructed implicit surface (see Figure 10(b)) and approaches the quality obtained by mesh beautification techniques. However, they exhibit a few more extraordinary vertices, resulting from the initial clustering at the T-Layer level of the M_0 (see Figure 9). Nonetheless, it should be noted that the refinement process *does not* generate additional extraordinary vertices. So, if limiting the number of such vertices really matters for some specific application, one easy solution would be to apply mesh beautification *on the coarse mesh* M_0, which of course is dramatically faster than applying remeshing on the final dense mesh.

As said previously, even if our reconstruction technique generates high-quality meshes in almost every tested examples, we have biased each speed vs. quality tradeoff of our algorithm towards computation efficiency. Consequently, for difficult examples that exhibit poorly sampled areas with high curvature, slower reconstruction techniques based on implicit surfaces [OBA*03, TRS04], usually offer better recovering of thin features.



(a) (b) (c)

Figure 9: *Reconstruction of the Igea model. (a) Input point set. (b) Reconstructed surface (c) Close-up on the semi-regular mesh produced by our algorithm. Note, in the red circle, the limit of our technique which propagates high-degree vertices generated on* M_0.

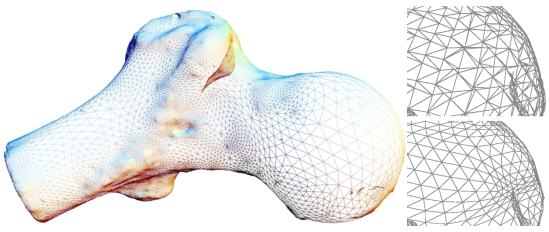
Models	Samples	Our method.	MPU.
Bunny	35949	0.852s.	4.272s.
Dino	56195	1.026s.	5.010s.
Santa	75783	1.067s.	7.135s.
Igea	134346	1.813s.	6.890s.
Male	303382	2.798s.	55.008s.
Dragon	437647	5.400s.	60.176s.
Happy Buddha	543652	6.384s.	80.866s.
Youthful	1728305	20.621s.	200.527s.
XYZ Dragon	3609601	41.844s.	480.693s.
XYZ Statue	5000000	53.298s.	475.551s.

Table 2: *Timing for VS-Tree surface reconstruction (with error set to* 5.10^{-4}*) and MPU (with error set accordingly).*

6. Conclusion

Hierarchical space subdivision schemes are the key ingredient to make efficient geometric processing methods in a large number of situations. In this paper, we have proposed the VS-Tree, a surface-based partitioning structure combining an octree with local quadtrees. This simple structure improves the quality of simplified meshes generated by vertex-clustering, while maintaining similar computation time compared to conventional octrees. We have also proposed an efficient point-to-mesh surface reconstruction algorithm based on the VS-Tree data structure. This algorithm recovers the global topology of the surface using the upper levels of the VS-Tree, while local geometric features are retrieved by a simple and efficient local displacement scheme induced by the lower levels of the VS-Tree.

Future research directions include quad-remeshing of very large meshes — a situation where parameterization methods are challenging. We also plan to further investigate the placement of expensive geometric processing at the T-layer level by, for instance, remeshing it before the refinement process in order to better handle sharp features and to upper bound the degree of extraordinary vertices. Finally, we believe that the VS-Tree has other applications in computer graphics. For example, it might be possible to use it in ray-tracing.



(a) (b)

Figure 10: *(a) Adaptivity by T-Layer contraint. (b) Close-up of the mesh obtained when reconstructing the ball-joint model with the* Multiple Partition of Unity *(top) method with our VS-Tree based method (bottom).*

References

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust. *Symposium on Solid Modeling and Applications* (2001). 2

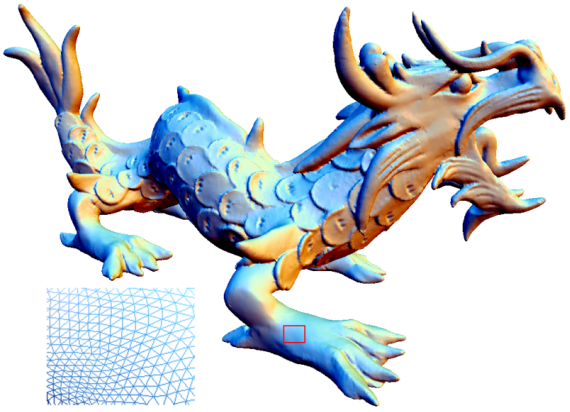


Figure 11: *Reconstruction of the XYZ Dragon model (3.6M points - 53.298 sec). Close-up on the semi-regular mesh produced by our algorithm.*

[BC00] BOISSONNAT J.-D., CAZALS F.: Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Symposium on Computational geometry* (2000). 2

[Ben75] BENTLEY J. L.: Multidimensional binary search trees used for associative searching. *Commun. ACM 18* (1975). 2

[Blo94] BLOOMENTHAL J.: An implicit surface polygonizer. In *Graphics Gems IV*. Academic Press, 1994. 2, 6

[BRS05] BOUBEKEUR T., REUTER P., SCHLICK C.: Visualization of point-based surfaces with locally reconstructed subdivision surfaces. *Shape Modeling International* (2005). 4

[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. *ACM SIGGRAPH* (2001). 2

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM SIGGRAPH* (2004). 2, 3, 4

[DGH01] DEY T. K., GIESEN J., HUDSON J.: Delaunay based shape reconstruction from large data. *IEEE Symposium on Parallel and Large-Data Visualization and Graphics* (2001). 2

[DQ01] DUAN Y., QIN H.: Intelligent balloon. *ACM Symposium on Solid Modeling and Applications* (2001). 2

[DYQS04] DUAN Y., YANG L., QIN H., SAMARAS D.: Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. *ECCV* (2004). 2, 6

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. *ACM SIGGRAPH* (1995). 2

[FKN80] FUCHS H., KEDES Z. M., NAYLOR B. F.: On visible surface generation by a priori tree structures. *ACM SIGGRAPH* (1980). 2

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. *ACM SIGGRAPH* (1997). 2, 4

[GKS00] GOPI M., KRISHNAN S., SILVA C.: Surface reconstruction based on lower dimensional localized delaunay triangulation. *Eurographics* (2000). 2, 3, 7

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *ACM SIGGRAPH* (1992). 2, 3, 5

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. *ACM SIGGRAPH* (1993). 2

[HDD*94] HOPPE H., DEROSE T., DUCHAMP T., HALSTEAD M., JIN H., MCDONALD J., SCHWEITZER J., STUETZLE W.: Piecewise smooth surface reconstruction. *ACM SIGGRAPH* (1994). 2

[JK02] JEONG W., KIM C.: Direct reconstruction of displaced subdivision surface from unorganized points, 2002. 2

[JT80] JACKINS C., TANIMOTO S.: Oct-trees and their use in representing three-dimensional objects. *CGIP 14* (1980). 2

[Lin00] LINDSTROM P.: Out-of-core simplification of large polygonal models. *ACM SIGGRAPH* (2000). 2

[Loo87] LOOP C.: *Smooth subdivisions surfaces based on triangles*. Master's thesis, Department of Mathematica, University of Utah, August 1987. 6

[LSS*98] LEE A. W. F., SWELDENS W., SCHRÖDER P., COWSAR L., DOBKIN D.: MAPS: Multiresolution adaptive parameterization of surfaces. *ACM SIGGRAPH* (1998). 2

[MK04] MARINOV M., KOBBELT L.: Optimization techniques for approximation with subdivision surfaces. *ACM Symposium on Solid Modeling and Applications* (2004). 2, 6

[NRDR05] NEHAB D., RUSINKIEWICZ S., DAVIS J., RAMAMOORTHI R.: Efficiently combining positions and normals for precise 3d geometry. *ACM SIGGRAPH* (2005). 5

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.: Multi-level partition of unity implicits. *ACM SIGGRAPH* (2003). 2, 6, 7

[PG01] PAULY M., GROSS M.: Spectral processing of point-sampled geometry. *ACM SIGGRAPH* (2001). 4, 5

[PGK02] PAULY M., GROSS M., KOBBELT L.: Efficient simplification of point-sampled surfaces. *IEEE VIS* (2002). 2, 5

[RB93] ROSSIGNAC J., BORREL P.: Multi-resolution 3d approximation for rendering complex scenes. *Modeling in Computer Graphics* (1993). 2

[Sam89] SAMET H.: *Quadtree, Octrees, and Other Hierarchical Methods*. Addison Wesley, 1989. 2

[SFS05] SCHEIDEGGER C. E., FLEISHMAN S., SILVA C. T.: Triangulating point set surfaces with bounded error. *Eurographics Symposium on Geometry Processing* (2005). 5

[SG01] SHAFFER E., GARLAND M.: Efficient adaptive simplification of messive meshes. *IEEE VIS* (2001). 2

[STKK99] SUZUKI H., TAKEUCHI S., KIMURA F., KANAI T.: Subdivision surface fitting to a range of points. *Pacific Graphics* (1999). 2

[SW03] SCHAEFER S., WARREN J.: Adaptive vertex clustering using octrees. *Geometric Design and Computing* (2003). 2, 5

[TO02] TURK G., O'BRIEN J. F.: Implicit surfaces that interpolate. *Shape Modeling International* (2002). 2

[TRS04] TOBOR I., REUTER P., SCHLICK C.: Multiresolution reconstruction of implicit surfaces with attributes from large unorganized point sets. *Shape Modeling International* (2004). 2, 7

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. *ACM I3D* (2001). 6

[ZSS97] ZORIN D., SCHROEDER P., SWELDENS W.: Interactive multiresolution mesh editing. *ACM SIGGRAPH* (1997). 2, 6