Visual Comput (2007)
DOI 10.1007/s00371-007-0172-1

ORIGINAL ARTICLE

Jean-Baptiste Debard
Romain Balp
Raphaëlle Chaine

Dynamic Delaunay tetrahedralisation of a deforming surface

J.-B. Debard (✉) · R. Balp
Tsinghua University,
Computer Science Department,
Beijing, China
{jb.debard, romain.balp}@gmail.com

R. Chaine
LIRIS, Université Lyon 1, Lyon, France
raphaelle.chaine@liris.cnrs.fr

Abstract Reconstruction algorithms make it possible to retrieve a surface from the Delaunay tetrahedralisation (DT) of a point sampling, whose density reflects the surface local geometry and thickness. Most of these algorithms are static and some work remains to be done to handle deforming surfaces. In such case, we defend the idea that each point of the sampling should move with the surface using the information given by the motion to allow fast reconstruction. In this article, we tackle the problem of producing a good evolving sampling of a deforming surface S, and maintaining its DT along the motion. The surface is known only through a projection operator $(O_1) : \mathbb{R}^3 \rightarrow S$, and a normal operator (O_2) that returns the oriented normal at a point on the surface. On that basis, we offer some perspectives on how reconstruction algorithms can be extended to the tracking of deforming surfaces.

Keywords Delaunay tetrahedralisation · Surface reconstruction · Particle sampling

1 Introduction

Increasing computational power and new data acquisition techniques have created a growing interest in the visualisation of dynamic phenomena. The understanding of moving and deforming surfaces can find many applications in medicine, physics, the entertainment industry, etc., but the problem of tracking surfaces has of yet been seldom addressed, except for substantial contributions in the field of level-set methods. We should also mention the work of Cheng et al. [6], who addressed this problem for skin surfaces in the case of a very special motion. Previous works mainly focused on maintaining a point sampling or a surface triangulation, but to our knowledge most existing methods usually turn out to provide insufficient information on the surface. For instance, in the case of surface triangulation-based methods, topological changes can not be systematically detected.

Instead of maintaining a surface triangulation, we propose the alternative of continuously reconstructing the surface triangulation from the Delaunay tetrahedralisation (DT) of a moving surface point sampling. Therefore, the approach we develop here is twofold: it first consists in maintaining a point sampling bound to a moving surface, so that the point distribution favours a good behaviour of Delaunay-based reconstruction algorithms; secondly, it preserves the DT associated to the point sampling along the motion. Indeed, it has been shown that a topologically and geometrically sound approximation of a surface can be recovered from the DT of a point sampling if the latter fulfils special requirements (nerve theorem, Edelsbrunner and Shah [10]). Moving surface tracking can thus be reduced to maintaining the DT of an evolving good point sampling, and the present article provides a robust framework for localisation or reconstruction algorithms based on the DT.

Our method is yet another particle sampling with the particularity that it relies heavily on its underlying DT. The latter provides a proximity data structure and carries local and global information on geometric features. The particle sampling method was first introduced to the graphics community by Szeliski and Tonnesen [24] to

simulate deformable surfaces and by Turk [25] for surface texture generation (see Fig. 1 for example of deformation). It was spread by Witkin and Heckbert [28] who clearly presented the problem through a simple numerical scheme. The technique consists in attaching particles to the surface and assigning to each of them an energy deriving from the interactions with other particles. Starting from an initial distribution, an iterative optimisation process allows a local minimum to be found that corresponds to a uniform distribution over the surface.

Although our algorithm can be used in any static situation, it might be slower than several other recent algorithms in this case. It has been designed for a moving surface, where the solution of a point sampling at time t is a good initial configuration for the optimisation scheme at time $t + \Delta t$. We provide a fast method, first by using the DT to improve the numerical scheme involved in numerous existing methods, but also by setting up a strategy for fast update of the DT.

We assume the surface S to be without boundary and $\mathcal{C}^1$ at any time except when and where topological changes happen. The surface is probed only through two operators: a projection operator

$$O_1 : \mathbb{R}^3 \to S$$
$$x \mapsto \tilde{p}$$

where $\tilde{p}$ is the nearest point to x that belongs to the surface; and a normal operator

$$O_2 : S \to \mathbb{R}^3$$
$$p \mapsto \boldsymbol{n}$$

where $\boldsymbol{n}$ is the oriented normal to the surface at the coordinates of p. The results presented hereafter have been obtained using implicit surfaces ($f(x) = 0$), along with a Newton–Raphson scheme (cf. Appendix 2) for the projection operator and the normalised gradient $\boldsymbol{n} = \nabla f / |\nabla f|$ (computed using a first order finite differences scheme) for the normal operator. Our method handles motion in a discrete manner: at each time step, it maintains a point sampling which adapts to the local curvature and thickness, and maintains its DT. The adaptation has some limits, for instance, when and where topological changes occur, the point density does not grow to infinity: This is equivalent to the assumption that topological events arise between two time steps. Currently, our algorithm handles all kinds of topological changes except the appearance of a new connected component. This restriction follows from the very little knowledge we have of the surface through O_1 and O_2, but further information about the surface can be used to solve this problem.

Contributions. To our knowledge, this is the first time a DT is used along with particle sampling to track a deforming surface. It allows for very few assumptions to be made on the surface while providing a good approximation of the surface using curvature and thickness adaptation. In a purely technical point of view, the contributions of this paper are:

- A new local curvature and thickness estimation of a sampled surface
- An improvement of the particle sampling scheme, in the coordinate descent strategy, in the computation of the Levenberg–Marquardt step and in the method to adapt the sampling to the curvature and the thickness of the surface
- A new strategy to update the DT in a lazy manner such that it does not destabilise the optimisation scheme

Organisation. Section 2 provides a state of the art particle sampling method. Section 3 describes our adaptive particle sampling method. In Sect. 4, we explain our algorithm for fast update of the DT and in Sect. 5, we illustrate the validity of the framework, using it for the reconstruction of a moving surface, and to present some benchmarks.

2 Related work

The particle sampling (PS) method has been used for numerous purposes. Witkin and Heckbert [28] introduced it to tackle implicit surface visualisation and deformation problems. However, the PS method was later used to mainly handle static problems: implicit surface visualisation [17], texture mapping [29], implicit surface polygonisation [11, 15, 22, 26], surface simplification [20]. Actually, distributing a point set on a manifold is a widely investigated topic in applied mathematics [14]. Levet et al. [16] presented an anisotropic PS method based on the Witkin–Heckbert method to achieve point rendering [17] of an implicit surface. They later proposed a geometric method to perform a fast, almost optimal, particle sampling of a static surface, which needs several optimisation steps to reach an optimal configuration. Surprisingly, the PS method has been used mainly for static problems, although it appeared to be slow, as the iterative process might need a lot of steps before reaching a satisfying solution of the problem starting from a bad initial configuration. Nevertheless, it is obviously fast for motion problems, when the solution of the problem at time t is a good initial configuration of the problem at time $t + \Delta t$.

If the use of the PS method has been restricted to static cases, it is most probably because of localisation difficulties. Indeed, previous works rely on a localisation grid which needs to be refined, at a high computational cost, when it is too coarse. When dealing with an adaptive point sampling of a moving surface, such a method looses efficiency, as the grid needs to be refined almost as soon as curvature changes appear, and becomes totally inefficient if the grid is too refined. In this article we propose

to use the DT as an alternative powerful proximity structure. Hierarchical localisation structures such as octree or kd-tree used along with a k-neighbour algorithm, such as those proposed in [23], might be competitive compared with the DT, as such a data structure could be updated relatively quickly after the motion of one point. However, as we choose to maintain the DT for Delaunay-based reconstruction algorithms to be run on top of our algorithm, we find it elegant and valuable to use it to support the particle sampling.

The particle sampling method relies on an energy associated to the interaction between particles. Szeliski and Tonnesen [24] first used a Lennard–Jones potential function drawn from the physical Van der Waals forces to simulate this interaction. De Figueiredo et al. [11] used spring-mass force systems to sample and polygonise an implicit surface, and Turk used a reaction-diffusion model for textures generation [25] and surface retiling [26]. Witkin and Heckbert [28] proposed an optimisation scheme based on Gaussian repulsion to visualise and deform an implicit surface. The sampling produced is uniform with respect to the $\mathbb{R}^3$ Euclidean metric restricted to the surface. Karkanis et al. [15] compared this last method to with their polygonisation algorithm, but their use of the PS method leads to an unbearably slow algorithm. Crossono and Angel [8] later proposed some modifications of the Witkin–Heckbert method, in order to work on 3D density images. They used the same repulsion energy as in [28] but set up a strategy to progressively adapt the scaling factor to the local curvature. Rösch et al. [21] extended the Witkin–Heckbert method to algebraic implicit surfaces with boundaries and self-intersections. Meyer et al. [18] also proposed several improvements of the Witkin–Heckbert method, in order to avoid the tuning of too many parameters, and to make adaptive sampling: they derived a new energy which is less influenced by scale changes, and which can be tuned to control the number of particles needed or speed up computation. They also introduced the Levenberg–Marquardt method, therefore avoiding the tuning of the gradient step in the optimisation scheme, and added new coefficients in the energy expression so that it leads to adaptive sampling. Though this latter article provides deeper understanding of the particle sampling method and proposes some clever ideas, the results are still deceiving, as the algorithm remains slow and the adaptation to the curvature very limited.

3 Adaptive particle sampling

Given a surface S and a metric d_S defined on it, the particle sampling aims at distributing a set of points such that the distribution satisfies the best packing problem. This consists in maximising the smallest distance among N points bound to S

$$d_S^N = \max_{p_1,\dots,p_N \in S} \min_{1 \le i < j \le N} d_S(p_i, p_j). \tag{1}$$

According to [3], for N being sufficiently big, it can be considered that any best packing configuration is uniformly distributed on S relative to the metric d_S. The solution of the best packing problem thus always corresponds to a well spaced distribution over the surface S (Fig. 2). However this problem is never solved directly as the function to be maximised in Eq. 1 is non-regular (i.e. $\mathcal{C}^0$ but

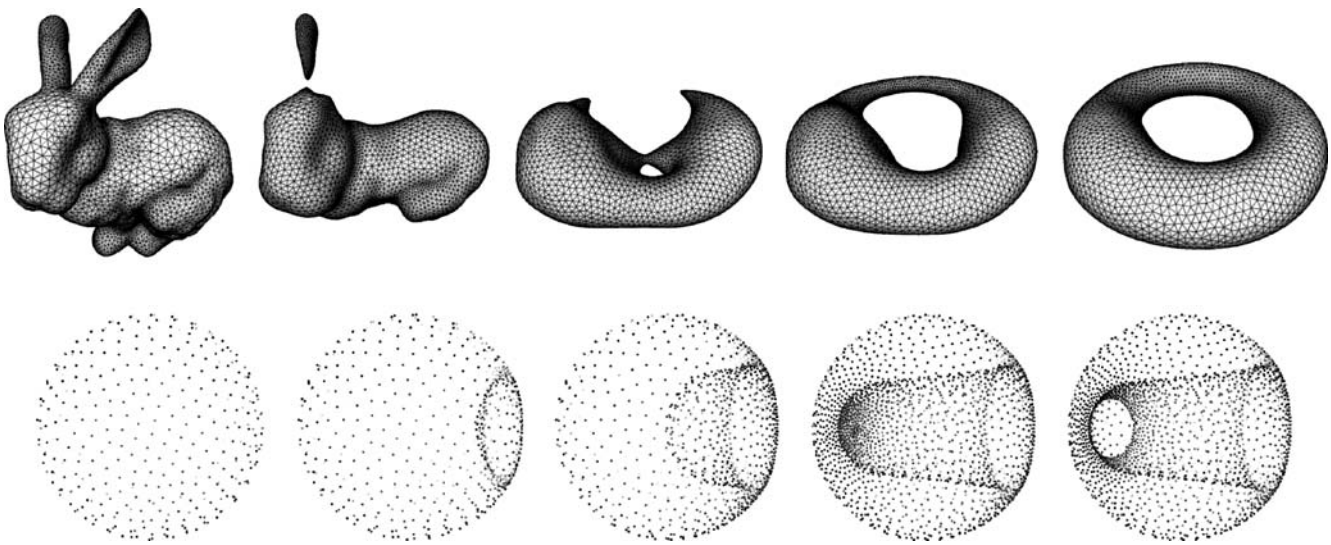

Fig. 1. Morphism between the bunny and a torus (*top*), and deforming sphere sampling (*bottom*), illustrating the dynamic adaptability of the sampling to the curvature and thickness changes of the surface

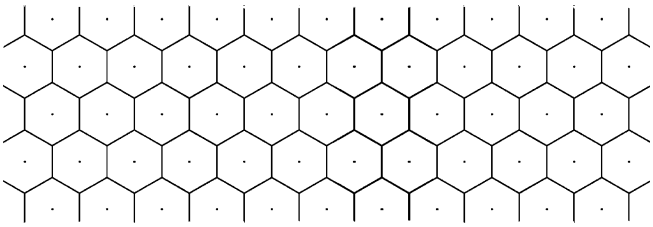

Fig. 2. Hexagonal packing, solution of the best packing problem in the plane with the Euclidean metric

not differentiable). Instead, it is a common approach to define an energy $E : (S)^{\mathrm{card}(\mathcal{P})} \longrightarrow \mathbb{R}$ on the point set $\mathcal{P}$, which is generated by mutual repulsion between each pair of points

$$E = \sum_{i \in \mathcal{P}} E_i = \sum_{i \in \mathcal{P}} \sum_{j \neq i} E_{ij} \tag{2}$$

where $E_{ij} = f(d_S(p_i, p_j))$ is a $\mathcal{C}^2(S \times S)$ function centered on point p_j. In practice, d_S is approximated by the Euclidean metric d in $\mathbb{R}^3$ and an optimisation scheme is used to find a minimum of this energy. The sampling is uniform in the sense of the geodesic metric d_S only if the support of the energy functions at each point is sufficiently small and the distribution on the surface is dense enough.

3.1 Choice of the energy function

Needless to say, the energy function has to be carefully chosen so that the point distribution corresponding to the energy minimum is as close as possible to the solution of the best packing problem. The Gaussian energy

$$E_{ij} = \mathrm{e}^{-\frac{r_{ij}^2}{\sigma_i^2}}$$

(where $\boldsymbol{r_{ij}} = p_j - p_i$ and $r_{ij} = d(p_i, p_j)$) was first used by Witkin and Heckbert in [28]. This energy can lead to bad minimal energy configuration if the scaling factor σ_i is too small or too big as shown in Fig. 3. Witkin and Heckbert [28] set up a complex scheme to control σ_i, but we find that a direct method to scale the energy consists in defining σ_i as a function of $d(p_i, p_{j_0})$, where p_{j_0} is p_i's nearest neighbour (see Fig. 3)

$$\sigma_i = \frac{r_{ij_0}}{\sqrt{2}}.$$

With this "moving" definition of σ_i, p_i is submitted to the repulsion of at least one particle throughout the process. At equilibrium it will be stuck in a well defined minimum, which is equidistant from the closest particles. Other particles do not have any influence on the definition of that minimum since their contribution to E_i is negligible in that area. This means that they can be suppressed from the definition of E_i; this is of critical importance from a computational point of view. Moreover, the DT is a geometric structure that states proximity relationships on a set of points, and allows us to wipe out the problem encountered in previous PS methods of localising relevant points. For the energy expression E_i, we consider only the neighbours of p_i in the DT, and among them, we prune irrelevant ones, i.e. those whose normal to the surface clearly indicates that they belong to other parts of the surface and those located beyond a distance $2\sigma_i$ from p_i. The scaled energy now reads

$$E_i = \sum_{j \in \mathrm{neighbours}(p_i)} \mathrm{e}^{-\frac{r_{ij}^2}{\sigma_i^2}}.$$

This energy is stable, efficient and can be computed very quickly. It further simplifies the calculus of our optimisation scheme.

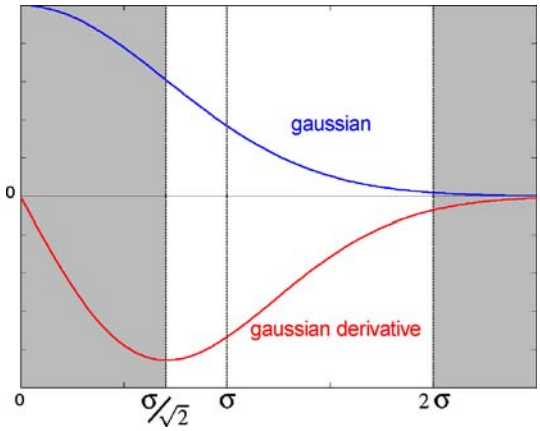

Fig. 3. To be a suitable repulsion energy, the Gaussian energy of a particle induced by its nearest neighbour should be such that $\frac{\sigma}{\sqrt{2}} < r < 2\sigma$, where r is the distance between the two particles

3.2 Optimisation scheme

Uniformly distributing a set of points by minimising an energy E on a surface is a difficult optimisation problem. The energy expression is far too complex to be readily used in an advanced optimisation method. Since variables (i.e. particles) are loosely coupled, it is worth applying a simple coordinate descent method where only one particle is free at a time and all the others remain fixed (cf. Fig. 4), as done by Meyer et al. [18].

3.2.1 Coordinate descent

Given the energy $E(k)$ associated to the point sampling's configuration k, if the displacement of at least one particle leads to $E(k+1)$ such that $E(k+1) < E(k)$, we proceed with the coordinate descent. When no particle displace-

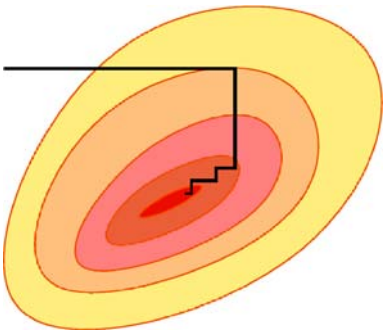

Fig. 4. Coordinate descent method in the plane along the directions of the axis

ment leads to a decrease of the energy, the minimum has therefore been reached, and the iterative process is stopped. This method can be quite inefficient if particles are selected one after the other, because some particles might be in a position where they can not minimise the energy any further. Therefore, we improved the scheme by only selecting particles that can lead to substantial decrease of the energy: when the displacement $|m_i|$ of a particle p_i is big with regard to its scale factor σ_i, i.e.

$$\frac{|m|}{\sigma_i} > THRESHOLD,$$

(in practice, we set the *THRESHOLD* to 0.05) we decide that its nearest neighbours and itself are good candidates for further steps. When there is no longer any candidate, all particles are stabilised and the algorithm terminates.

Algorithm 1. Coordinate descent

Require: E an energy function, $\{p_1, \ldots, p_N\} \subset \mathbb{R}^3$
Ensure: $E(p_1, \ldots, p_N)$ reaches a local minimum
  let *Candidate_list* be a FIFO list
  push every point in *Candidate_list*
  while *Candidate_list* is not empty **do**
    pop a candidate p_i and compute its new position p_i^{new}
    if $\frac{d(p_i^{\text{new}} - p_i)^2}{\sigma_i^2} > THRESHOLD$ **then**
      push p_i^{new} in *Candidate_list*
      push the nearest neighbours of p_i^{new} in *Candidate_list* if not already in it
    end if
  end while

When all particles are fixed except particle p_i, the latter is moved to a new position so that the diminution of E in this "direction" is maximum. If we assume $E_{ij} \approx E_{ji}\ \forall j$ (cf. Eq. 2), decreasing E is then equivalent to decreasing E_i and the problem is reduced to minimising the energy

$$E_i(p_i) : S \longrightarrow \mathbb{R}.$$

Note that, here, E_i is a function of p_i only, since all the other parameters are fixed. We now assume that if the distribution is dense enough, the surface can be considered to be almost plane in the region which contains p_i and its nearest neighbours. Let $\mathcal{T}_i$ be the corresponding tangent plane. Under such conditions, the problem is almost equivalent to minimising

$$F_i(p_i) : \mathcal{T}_i \longrightarrow \mathbb{R}$$

where $F_i = \sum_{j \neq i} F_{ij}$ and $F_{ij} = \mathrm{e}^{d(p_i, q_j)^2/\sigma_i^2}$. F_i refers to the "tangential"energy of particle p_i, and q_j is the projection of p_j on $\mathcal{T}_i$. Reducing the energy minimisation 3D problem to a problem in the tangential plane leads to substantial computational acceleration. This approach further ensures that particles are bound to the vicinity of the surface, preventing any displacement to the opposite side of the surface. However, this new optimisation process must be constrained by the decrease of $E_i(p_i)$.

3.2.2 Optimising F_i with regards to p_i

The Hessian matrix H_i of an energy F_i, expressed as a function of r_{ij} $(j = 1, \ldots, N_i,\ j \neq i)$ can be developed into

$$H_i = \frac{\partial^2 F_i}{\partial p_i^2} = \sum_{j=1, j\neq i}^{N_i} \left[\frac{\partial^2 F_{ij}}{\partial r_{ij}^2} \left(\frac{\boldsymbol{r_{ij}}}{r_{ij}} \otimes \frac{\boldsymbol{r_{ij}}}{r_{ij}} \right) + \frac{1}{r_{ij}} \frac{\partial F_{ij}}{\partial r_{ij}} \left(I - \frac{\boldsymbol{r}_{ij}}{r_{ij}} \otimes \frac{\boldsymbol{r}_{ij}}{r_{ij}} \right) \right]. \quad (3)$$

According to Meyer et al. [18], the second term inside the sum can be ignored, since first order derivatives tend to zero in the vicinity of the minimum. A first order approximation of the displacement h_i of particle p_i which ensures that $\nabla F_i(p_i + h_i) = 0$ is

$$h_i = -H_i^{-1} \nabla F_i.$$

This approximation alone might lead to erratic behaviour and might not converge when starting from a remote point, because the Hessian's approximation is not valid. This problem is solved by applying the Levenberg–Marquardt (LM) method. The latter is a trusted region-based method which proposes to condition the Hessian matrix in an adaptive way. The conditioned Hessian matrix reads

$$\tilde{H}_i^{lm} = \begin{cases} H_i^{ll}.(1 + \lambda_i) \\ H_i^{lm}, \qquad l \neq m. \end{cases}$$

The LM method ensures that there always exists a $\lambda_i > 0$ such that h_i lays in the neighbourhood of the initial position p_i (i.e. within the distance to the nearest neighbour).

This optimisation scheme is modified to find the minimum of F_i in $\mathcal{T}_i$, and the following constraint is further

added:

$$E_i(p_0, \ldots, O_1(p_i + h_i), \ldots, p_n) \leq E_i(p_0, \ldots, p_i, \ldots, p_n).$$

See Algorithm 2 for the pseudo-code of the LM method. The details of the calculus for the LM step are given in Appendix 1.

Algorithm 2. Levenberg–Marquardt (LM)

Require: $E_i \in \mathcal{C}^2(S)$, $p_i \in S$ and $\lambda_i > 0$
Ensure: $E_i(O_1(p_i + h_i)) < E_i(p_i)$
  let $h_i(F_i) \leftarrow \tilde{H}_i^{-1} \nabla F_i$
  while $E_i(O_1(p_i + h_i)) < E_i(p_i)$ **do**
    $\lambda_i \leftarrow 10 \times \lambda_i$
    compute $\tilde{H}_i$ (Eq. 3)
    $h_i \leftarrow \tilde{H}_i^{-1} \nabla F_i$
  end while
  $\lambda_i \leftarrow \frac{\lambda_i}{10}$
  return h_i

3.3 Changing the metric for adaptive sampling

The particle sampling method presented in the previous sections relies on the DT of the sampling. The DT provides direct information on neighbouring relationships and ensures that the geodesic metric can be approximated by the Euclidean one. Our article further aims at providing a robust method to produce a sampling configuration that favours reconstruction algorithms based on the DT. Amenta et al. [2] proved that, provided the input point-set density is proportional (with a sufficiently small coefficient) to the distance to the medial axis (ϵ-sample), their CRUST algorithm produces a surface approximation which is similar to the surface itself from both geometrical and topological points of view. A review of the algorithms which provide similar guaranties is given in [4, 9].

In practical cases, it is difficult to check whether a point-set is an ϵ-sample for a given surface. Allegre et al. [1] therefore proposed an alternative method which relies on other interesting properties of point samplings. Their idea consists in using a geometric radius based on the normal deviation, and a local thickness, rather than the distance to the medial axis. The geometric radius and the local thickness are both defined on continuous surfaces, but they can be readily approximated on point samplings. Exploiting the DT and the normal operator O_2, we propose a novel approximation of those quantities. These approximations are used to define a new metric, which is then used in our optimisation scheme, therefore ensuring that the point distribution adapts to the local curvature and to the thickness of the deforming surface. In the following sections, we give the definitions of the normal deviation and the thickness, and present the way they are used in the optimisation scheme.

3.3.1 Normal deviation radius

The normal deviation k_i^{nd} of the sampling at p_i is defined by

$$\left(k_i^{\text{nd}}\right)^2 = \max_{j \in N(i)} \frac{1 - \boldsymbol{n_i} \cdot \boldsymbol{n_j}}{r_{ij}^2}$$

where $N(i)$ refers to the neighbours of p_i in the DT which are on the same side of the surface (i.e. $\boldsymbol{n_i} \cdot \boldsymbol{n_j} > 0$); r_{ij} refers to the distance between p_i and p_j (as above), while $\boldsymbol{n_i}$ and $\boldsymbol{n_j}$ are the normals to the surface in p_i and p_j, respectively (cf. Fig. 5).

3.3.2 Thickness

The thickness at point p_i is a property defined on the surface, which corresponds to the distance between p_i and the nearest point that lies on an opposite region of the surface (see Fig. 6). It is estimated from the DT by

$$(th_i)^2 = \min_{j \in N_{\text{del}}(i), \boldsymbol{n_i} \boldsymbol{n_j} < 0} d_{ij}^2$$

where $N_{\text{del}}(i)$ refers to the Delaunay neighbourhood of p_i.

3.3.3 Adaptation coefficient k_i

The adaptation coefficient k_i is a combination of the Delaunay-based information carried by k_i^{nd} and the

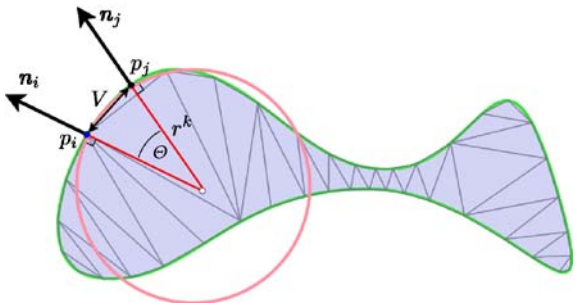

Fig. 5. Normal deviation $(k_i^{\text{nd}})^2 = \frac{1-\cos(\Theta)}{r_{ij}^2}$

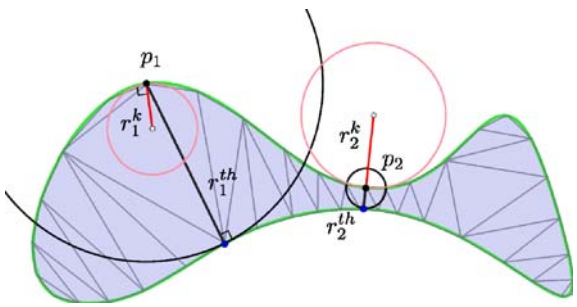

Fig. 6. Radius of curvature r^k (*red*) and thickness th (*grey*)

surface-based information carried by th_i. We use this coefficient to distort the metric locally

$$k_i^2 = \max\left(\left(k_i^{\text{nd}}\right)^2, \frac{1}{th_i^2}\right)$$

3.3.4 Changing the metric

In previous works, an adaptation of the point sampling to the curvature was achieved by adding weights in front of the E_i in the energy expression Eq. 2 [18], or by modifying the scaling radius σ_i [8, 15]. It turns out that modifying the scaling radius leads to extremely slow convergence, while adaptability is limited when adding weights. The key to handling adaptation is not to modify anything in the energy expression, but to change the metric instead (Fig. 7)

$$d^2(p_1, p_2) = \left(\frac{k_1^2 + k_2^2}{2}\right) |p_1 - p_2|^2.$$

Note that the scaling radius σ_i is defined using this new metric. The notion of surface neighbourhood is not affected by the metric change, therefore the DT is still defined using the Euclidean metric. Results of the adaptation can be seen on the deforming sphere of Figs. 1 and 8.

3.3.5 Controlling density

In order to provide an adaptive sampling while ensuring fast convergence of the PS method, some points need to be added or removed in the course of the algorithm. We thus define the following rules: As soon as a particle p_i is no longer a good candidate, i.e. if $\sigma_i < \Sigma_{\min}$ the particle is deleted; if $\sigma_i > \Sigma_{\max}$ the particle splits into two particles that are pushed in the candidate list, $\Sigma_{\min}$ and $\Sigma_{\max}$ being

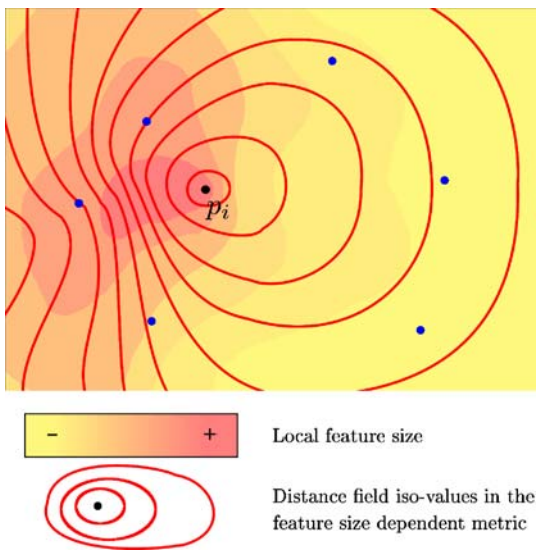

Fig. 7. Neighbourhood of p_i in the feature size dependent metric

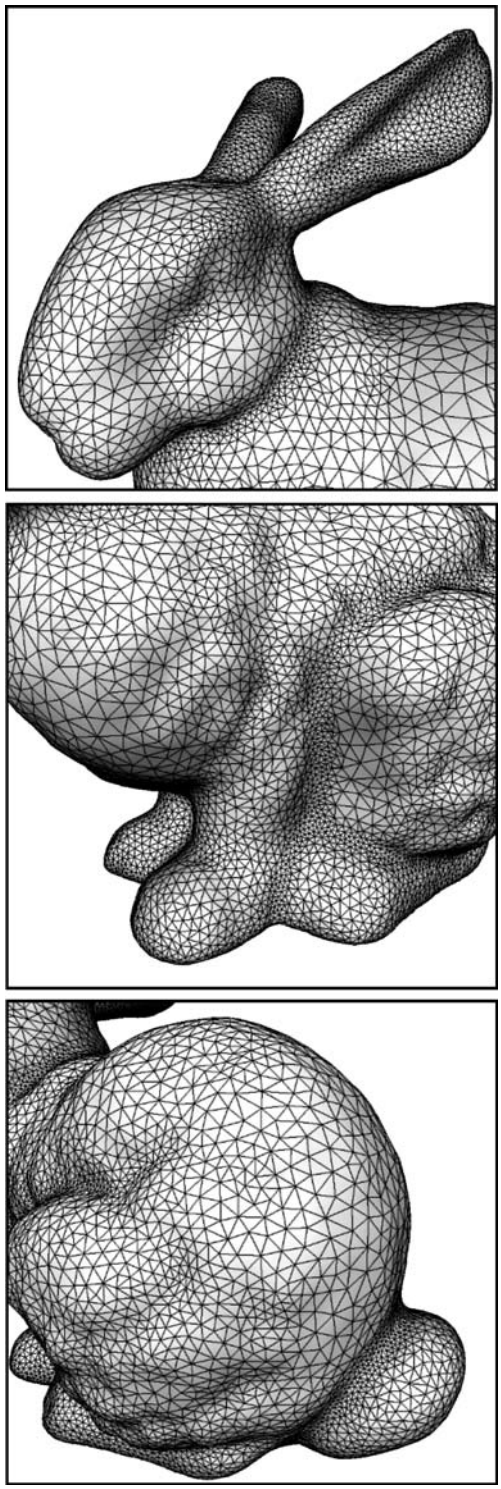

Fig. 8. Adaptive reconstruction of the Stanford bunny (13K points)

a user defined threshold. In our experiment we noted that good results were obtained with $\Sigma_{\max} \approx 2\Sigma_{\min}$.

4 Fast update of the DT

When a particle has been moved, the DT needs to be updated accordingly. No satisfying methods have yet been released to update the DT efficiently, but many strategies are still possible.

4.1 Overview of existing algorithms

Most existing works deal with the Delaunay update of several points simultaneously in motion. Guibas and Russel [13], proposed three different algorithms to update the vertices of the DT simultaneously. The first one is based on the kinetic data structures (KDS) introduced by Bash and Guibas [12]. The idea behind the KDS is to maintain a set of certificates which ensure that the Delaunay properties remain valid. When a certificate fails, the KDS recomputes the parts of the structure which need to be changed and updates the set of certificates. There is one certificate per facet. The KDS method is unbearably slow when all the points move simultaneously, since the algorithm needs to separate the roots of a degree-6 polynomial to determine a certificate's failure time. Moreover, the KDS can not handle degenerate cases where several events happen at the same time (for instance, when more than six points are cospherical or more than four points are coplanar).

The idea of the second method is to update only what needs to be updated to recover the DT, relying on two predicates:

- The embedding predicate which is true for a vertex if all the tetrahedra incident to this vertex are embedded in $\mathbb{R}^3$
- The Delaunay predicate which is true for a vertex if every union of two adjacent tetrahedra incident to this vertex are of Delaunay (meaning the circumscribing sphere of one tetrahedron does not contain any vertex of the other tetrahedron in its interior)

All the vertices are moved to their final position, and in a first step, the vertices whose embedding predicate has failed are successively removed from the DT and put in a list until the tetrahedralisation is embedded in $\mathbb{R}^3$. In a second step, the same strategy is applied on vertices which the Delaunay predicate has failed. At the end of these loops, the remaining tetrahedralisation verifies the Delaunay properties, and all the removed points are reinserted in the DT.

The removal operation and the localisation task required for reinserting a point are the most time-expensive routines of this strategy. Guibas and Russel finally proposed an improvement of the second step of the second method, by trying to flip in order to recover the DT from the embedded tetrahedralisation. The flipping heuristic can fail as reported in [13]: in this case, the DT has to be recomputed from scratch. Fortunately enough, this last strategy did not fail so often in their experiments and appeared to be the fastest presented in their benchmarks.

4.2 Moving vertices one after another

In the present situation, vertices are moved one after the other, and the information on the previous position of a vertex can thus be used for fast localisation of its target position. If a vertex violates the embedding or the Delaunay predicate, the inserting-removing strategy can then be performed on that basis (see Algorithm 3). This strategy reduces the localisation cost to zero in the case of small particle displacement. However, the point removal routine monopolizes 9/10 of the global computational time. In our method, the Levenberg–Marquardt scheme ensures that a particle can not escape its first-neighbour ring in one step. Therefore, when a point moves, its neighbourhood is only slightly altered and we can update the DT in a lazy manner, without disturbing the optimisation scheme.

Algorithm 3. move_point

Require: $v \in E$ a vertex of the DT $DT(E)$, $p_{\text{new}} \in \mathbb{R}^3$
Ensure: DT is the DT of E when v has moved to p_{new}
  let p_{old} be the position of v
  set v to the position p_{new}
  if v does verify the *embedding predicate* **then**
    if v does verify the *Delaunay predicate* **then**
      return
    end if
  end if
  set v to its old position p_{old}
  insert p_{new} in DT
  remove v from DT

4.3 Lazy update

We propose to associate two distinct positions to each vertex of the DT: the first one corresponds to the true position which is used in the sampling optimisation, while the second one is fictitious and allows the Delaunay properties to be satisfied. We also define a strategy to ensure some DT-based relevance between the two positions. A score is associated to each vertex, indicating how strongly its true position violates the embedding and Delaunay properties after a move:

- If the vertex does not verify the embedding predicate at its true position, its score is incremented by s_1.
- If the vertex does verify the embedding predicate, but does not verify the Delaunay predicate at its true position, its score is incremented by s_2.

When the score of a vertex reaches s_{limit}, the vertex is moved by insertion/suppression (as we stated experimentally that it is faster to insert first and then remove than the inverse).

Table 1. Algorithm profile, with basic (*top*) and lazy (*down*) Delaunay update, on the following experiment: a set of 3000 points is distributed on the ellipsoid, starting from a very bad configuration and ensuring that particles can be neither added nor deleted. All the particles have to move until they reach their equilibrium position and the algorithm thus meets all situations from the worst case to the best one as shown in Fig. 9. Hence, the experiment reflects the relative cost of each subroutine on the average

Name	Subroutine	CPU time (%)
Move_point		92.59
	Remove	74.31
	Delaunay_predic	6.58
	Insert	2.90
	Embedding_predic	0.62
	Locate	0.06
LM		7.41
Lazy_move_point		81.15
	Remove	49.00
	Delaunay_predic	11.59
	Embedding_predic	2.70
	Insert	2.06
	Locate	0.07
LM		18.85

The laziness consists in deferring the updating of the neighbourhood of a point as long as possible. In our algorithm, the values of s_1, s_2 and s_{limit} were set to 2, 1 and 4, respectively. We experimentally stated that if the maximum score value is higher than 4, the update scheme is "too lazy" and influences the optimisation scheme. In fact, the latter converges more quickly, but the final configuration is too different from the solution of the best packing problem.

This strategy might not win the prize of elegance, but it reduces the calls to the vertex removal routine by more than 3, and reduces the computation time by a factor of 3. The Delaunay update routine still remains the bottleneck of the whole algorithm as shown in Table 1.

5 Applications and results

We have tested our algorithm on various examples including static and dynamic shapes defined by different implicit surface representations. Indeed, implicit surfaces provide direct expressions for the operators O_1 and O_2. Simulation results show that our method ensures excellent adaptation

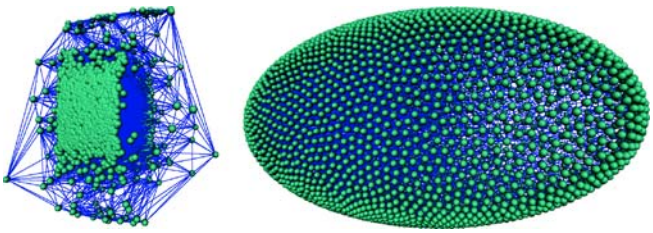

Fig. 9. Very bad initial distribution (*left*), and final configuration (*right*), on the ellipsoid with 3000 points

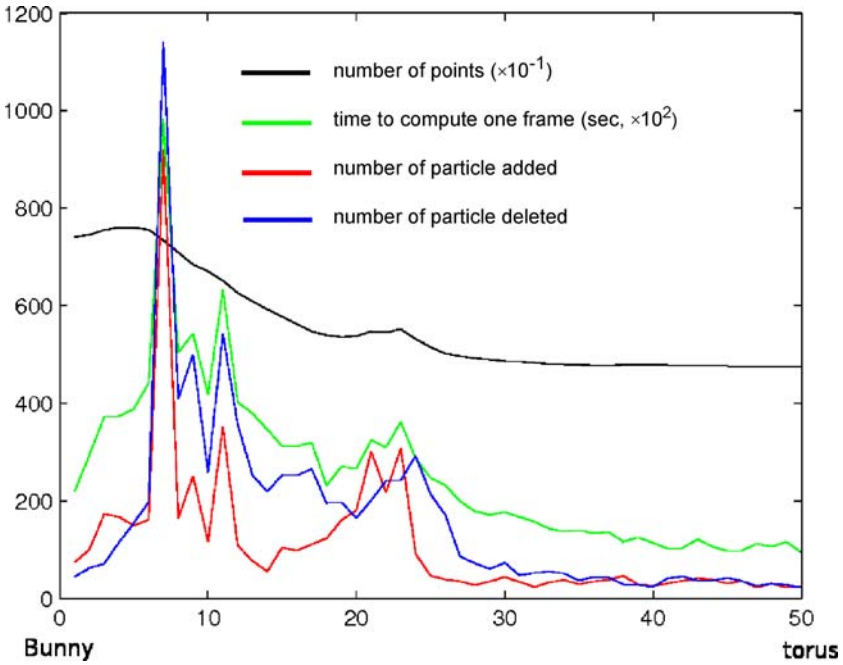

Fig. 10. Morphism between the bunny and the torus (50 frames), the *blue* and *red lines* show the number of particles removed and added during the computation of the ith frame, respectively; the *green line* represents the CPU time required to compute one frame

to the curvature and thickness of the surface, and that geometrical and topological changes are always handled properly. Three different kinds of representation were used, namely skeletal implicit surfaces, frequency-based representation using spherical harmonics [19], and variational implicit surfaces using radial basis functions. A good test for our algorithm is the morphism of two shapes. We used the work of Turk and O'Brien on variational implicit surfaces [27] to compute a morphism between the Stanford bunny and a torus (Fig. 1, top).

As far as the DT is concerned, we used the data structure design offered by the CGAL library [7], since it provides average constant time operation for the "neighbours request", "point insertion" and "point removal" routines. To illustrate the adequacy of dynamic particle sampling with Delaunay-based reconstruction methods, our algorithm was coupled with the Delaunay-based convection algorithm proposed by Chaine [5]. All the shapes presented in this article were extracted from the DT using this approach.

Figure 10 presents a synthetic benchmark of the algorithm running on a 50 frames morphism between the bunny and the torus, starting with about 8000 points and ending with 4800 points. We ran this experiment on a AMD64 3000+ with 2 Gb of memory. It took 3 s to compute a frame on average, 1.6 s on average when no big geometrical changes occurs, and a lot longer when such an event happens, namely, around topological changes, when the ears of the bunny collapse (three connected components appear and then collapse) and when handles appear (three handles appear and two of them then disappear). During these big geometrical changes, many particles have to be removed or inserted in some regions to satisfy the required density. Moreover, due to the bad shape of the implicit function near topological changes, the projection operator sometimes fails to reproject an inserted point at its nearest point on the surface, and some points are therefore inserted and immediately removed thereafter, which explains why the curve of removed points follows those of added points on the graph of Fig. 10.

6 Conclusion and future works

In this article we have coupled the particle sampling method to the problem of tracking the DT associated to the sampling. In particular, we have shown that the DT properties can be exploited in replacement of the usual localisation grid, and allows a parameter-free definition of the energy involved in the particle sampling method. The DT was further used to define a new metric which reflects the surface local curvature and thickness. This metric is used to adapt the sampling to surface geometrical properties, and permits the obtaining of a DT which is very well adapted to reconstruction purposes.

On that basis, we have improved existing numerical schemes by exploiting the local points' configuration, and speeded up convergence. As far as maintaining the DT is concerned, we have adopted a lazy update strategy that allows to postpone heavy operations as long as they are not considered necessary. The bottleneck of our algorithm is obviously the vertex removal and the update of the DT. Our present investigations aim at suppressing the need for point removal in the update algorithm. In particular, we worked on a KDS that moves one vertex at a time. The latter is expected to work well, since certificates are only degree-2 polynomials in this case. However, to date, our results with this KDS have not been as good as expected. The present work is naturally oriented towards kinetic surface reconstruction issues. However it shall also be extended to surfaces where the operators O_1 and O_2 are not easily supplied.

Appendix 1: Details of the calculus for the optimisation step

In [18], the optimisation step is calculated in $\mathbb{R}^3$ and projected in the tangent plane afterward, but when the step is computed in the tangent plane directly, the calculus is much more simple. For instance we avoid the inversion of a 3×3 matrix. Let $\{o \in \mathbb{R}^3, \boldsymbol{a} \in \mathbb{R}^3, \boldsymbol{b} \in \mathbb{R}^3\}$ be a coordinate system for the tangent plane $\mathcal{T}_i$, where $o = p_i^0$ (the subscript 0 refers to the initial position of the particle). We denote $v^{(1)}$ and $v^{(2)}$ the coordinates of a vector $\boldsymbol{v} \in \mathcal{T}_i$ along $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. At each time step of the coordinate descent, σ_i is constant. The tangential energy F_i is then C^∞ and the calculus of ∇F_i and H_i is very simple. Let $\boldsymbol{s_{ij}}$ be the projection of $\boldsymbol{r_{ij}}$ on the tangent plane $\mathcal{T}_i$, and $\boldsymbol{n}_{ij} = \frac{\boldsymbol{r}_{ij}}{r_{ij}}$, as shown in Fig. 11. The tangential energy reads

$$F_i = \sum_{j \in N_i} \mathrm{e}^{-\left(\frac{s_{ij}}{\sigma_i}\right)^2}.$$

The Hessian matrix $H_i(F_i)$ defined on $\mathcal{T}_i$ is

$$H_i^{lm} = \sum_{j \in N_i} \mathrm{e}^{-\left(\frac{s_{ij}}{\sigma_i}\right)^2} \left(\frac{4}{\sigma_i^4} s_{ij}^2 - \frac{2}{\sigma_i^2}\right) n_{ij}^l n_{ij}^m.$$

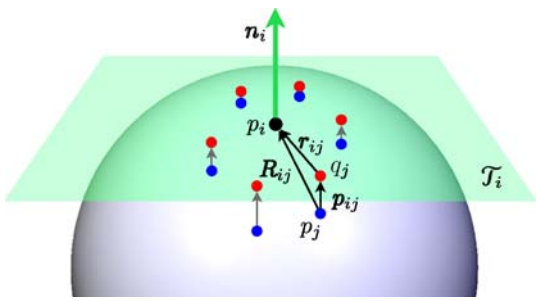

Fig. 11. Vicinity of p_i

The tangential energy gradient is

$$\nabla F_i = -\sum_{j\in N_i} \frac{2}{\sigma_i^2} \mathrm{e}^{-\left(\frac{s_{ij}}{\sigma_i}\right)^2} s_{ij}.$$

The displacement h_i in $\mathcal{T}_i$ therefore reads

$$h_i = -\frac{1}{\tilde{H}^{11}\tilde{H}^{22} - (\tilde{H}^{12})^2} \begin{pmatrix} \tilde{H}^{22} & -\tilde{H}^{12} \\ -\tilde{H}^{12} & \tilde{H}^{11} \end{pmatrix} \nabla E_i$$

and can be expressed as

$$h_i = h_i^{(1)}\boldsymbol{a} + h_i^{(2)}\boldsymbol{b}.$$

In practice, we do not use the Euclidean metric in the evaluation of h_i, but a modified metric instead (cf. Sect. 3.3.4).

Appendix 2: The Newton–Raphson method

The Newton–Raphson scheme allows us to reproject a point on an implicit surface ($f(X) = 0$); it can be modified following the idea of the Levenberg–Marquardt method, with a mix between a good guess and a gradient descent:

Algorithm 4. Newton–Raphson

Require: $f \in C^1(\mathbb{R}^3)$ (implicit func.), $p \in \mathbb{R}^3$ and $\epsilon << 1$
Ensure: $f(p) < \epsilon$ (p is on the surface)
  let $\lambda \leftarrow 1 \in \mathbb{R}$
  while $|f(p)| > \epsilon$ **do**
    let $\tilde{p} \leftarrow p - f(p)\frac{\nabla f(p)}{(1+\lambda)\nabla f(p)\cdot\nabla f(p)}$
    if $|f(\tilde{p})| < |f(p)|$ **then**
      $p \leftarrow \tilde{p}$
      $\lambda \leftarrow \frac{\lambda}{10}$
    else
      $\lambda \leftarrow 10\lambda$
    end if
  end while
  return p

Acknowledgement The authors wish to thank Mohammed Mousa for providing them with the code of his spherical harmonic decomposition [19]. This work was supported partly by the National Basic Research Project of China (No. 2006CB303102) and the Natural Science Foundation of China (No. 60673004, 60333010), and partly by the EROS3D project from the French Agence Nationale de la Recherche (ANR).

References

1. Allègre, R., Chaine, R., Akkouche, S.: A flexible framework for surface reconstruction from large point sets. Comput. Graph. **31**(2), 190–204 (2007)
2. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: SMA '01: Proceedings of the 6th ACM Symposium on Solid Modeling and Applications, pp. 249–266. ACM, New York (2001)
3. Borodachov, S.V., Hardin, D.P., Saff, E.B.: Asymptotics of best-packing on rectifiable sets. ArXiv Mathematical Physics e-prints (2006)
4. Cazals, F., Giesen, J.: Delaunay Triangulation Based Surface Reconstruction. Springer, Berlin Heidelberg New York (2006)
5. Chaine, R.: A geometric-based convection approach of 3-d reconstruction. In: Symposium on Geometry Processing, pp. 218–229. Eurographics Association, Aire-la-Ville, Switzerland (2003)
6. Cheng, H.L., Dey, T.K., Edelsbrunner, H., Sullivan, J.: Dynamic skin triangulation. In: SODA '01: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 47–56. Society for Industrial and Applied Mathematics, Philadelphia, PA (2001)
7. Computational Geometry Algorithms Library: http://www.cgal.org. Cited Nov. 2006
8. Crossno, P., Angel, E.: Isosurface extraction using particle systems. In: VIS '97: Proceedings of the 8th Conference on Visualization '97, pp. 495ff. IEEE, Los Alamitos, CA (1997)
9. Dey, T.K.: Curve and Surface Reconstruction: Algorithms with Mathematical Analysis. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, New York (2006)
10. Edelsbrunner, H., Shah, N.R.: Triangulating topological spaces. In: SCG '94: Proceedings of the 10th Annual Symposium on Computational Geometry, pp. 285–292. ACM, New York (1994)
11. de Figueiredo, L.H., de Miranda Gomes, J., Terzopoulos, D., Velho, L.: Physically-based methods for polygonization of implicit surfaces. In: Proceedings of the Conference on Graphics Interface '92, pp. 250–257. Kaufmann, San Francisco (1992)
12. Guibas, L.: Kinetic Data Structures: State of the Art. Stanford University, Palo Alto, California, USA (2004)
13. Guibas, L., Russel, D.: An empirical comparison of techniques for updating Delaunay triangulations. In: SCG '04: Proceedings of the 20th Annual Symposium on Computational Geometry, pp. 170–179. ACM, New York (2004)
14. Hardin, D.P., Saff, E.B.: Discretizing manifold via minimum energy points. Notices Am. Math. Soc. **51**(10), 1186–1194 (2004)
15. Karkanis, T., Stewart, A.J.: Curvature-dependent triangulation of implicit surfaces. IEEE Comput. Graph. Appl. **21**(2), 60–69 (2001)
16. Levet, F., Granier, X., Schlick, C.: Fast sampling of implicit surfaces by particle systems. In: SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06), p. 39. IEEE, Washington, DC (2006)
17. Levet, F., Reuter, P., Schlick, C.: Anisotropic sampling for differential point rendering of implicit surfaces. J. WSCG 2005 **13** (2005)
18. Meyer, M.D., Georgel, P., Whitaker, R.T.: Robust particle systems for curvature dependent sampling of implicit surfaces. In: SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005, pp. 124–133. IEEE, Washington, DC (2005)
19. Mousa, M., Chaine, R., Akkouche, S.: Direct spherical harmonics transform of a triangulated mesh. J. Graph. Tools **11**(2), 17–26 (2006)
20. Pauly, M., Gross, M., Kobbelt, L.P.: Efficient simplification of point-sampled surfaces. In: VIS '02: Proceedings of the Conference on Visualization '02, pp. 163–170. IEEE, Washington, DC (2002)
21. Röch, A., Ruhl, M., Saupe, D.: Interactive visualization of implicit surfaces with singularities. Eurograph. Comput. Graph. Forum **16**(5), 295–306 (1997)

22. Rodrian, H., Moock, H.: Dynamic triangulation of animated skeleton-based implicit surfaces. In: Proceeding of the Annual Conference on Implicit Surfaces '96, pp. 37–52. ACM (1996)
23. Sankaranarayanan, H.S.J., Varshney, A.: Fast *k*-neighborhood algorithm for large point-clouds. In: Proceedings of the 3rd IEEE/Eurographics Symposium on Point-Based Graphics. ACM, Boston, MA, USA (2006)
24. Szeliski, R., Tonnesen, D.: Surface modeling with oriented particle systems. In: SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, pp. 185–194. ACM, New York (1992)
25. Turk, G.: Generating textures on arbitrary surfaces using reaction-diffusion. In: SIGGRAPH '91: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, pp. 289–298. ACM, New York (1991)
26. Turk, G.: Re-tiling polygonal surfaces. In: SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, pp. 55–64. ACM, New York (1992)
27. Turk, G., O'Brien, J.F.: Shape transformation using variational implicit functions. In: SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 335–342. ACM/Addison-Wesley, New York (1999)
28. Witkin, A.P., Heckbert, P.S.: Using particles to sample and control implicit surfaces. In: SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pp. 269–277. ACM, New York (1994)
29. Zonenschein, R., Gomes, J., Velho, L., de Figueiredo, L.: Controlling texture mapping onto implicit surfaces with particle systems. In: Proceedings of Implicit Surfaces '98, pp. 131–138. ACM, New York, USA (1998)

JEAN-BAPTISTE DEBARD has been a student at the Ecole Centrale Lyon (France) since 2002 and a MS candidate in the Computer Science Department of Tsinghua University (China) since 2004. His area of interest is computational geometry.

ROMAIN BALP has been a student of the Ecole Centrale Nantes (France) since 2003 and a MS candidate in the Computer Science Department of Tsinghua University (China) since 2005. His research interests include computational geometry and image-based modelling.

DR. RAPHAËLLE CHAINE is an Associate Professor of Computer Science at the University of Lyon I (France), and a member of the LIRIS (CNRS) research lab. She obtained her MS degree in Cognitive Sciences in 1995 at Paris VI University (France), and PhD in Computer Science in 2000 at University of Lyon I. Her research activities include surface reconstruction, surface decomposition, segmentation, mesh compression and computational geometry.