Maintaining Deforming Surface Meshes*

Siu-Wing Cheng† Tamal K. Dey‡

Abstract

We present a method to maintain a mesh approximating a deforming surface, which is specified by a dense set of sample points. We identify a reasonable motion model for which a provably good surface mesh can be maintained. Our algorithm determines the appropriate times at which the mesh is updated to maintain a good approximation. The updates use simple primitives, and no costly computation such as line-surface intersection is necessary. Point insertions and deletions are allowed at the updates. Each update takes time linear in the size of the current sample set plus the new sample points inserted. We also construct examples for which, under the same model, no other algorithm makes asymptotically fewer changes to the mesh than our algorithm.

1 Introduction.

Digital simulations of deforming bodies often require a surface mesh to be maintained as time progresses. A major concern is the maintenance of a non-self-intersecting mesh that reflects the geometry of the deforming surface. In this paper we study the problem of maintaining a surface mesh to approximate an unknown deforming surface. We are given a set of dense sample points (ε-sample) on the deforming surface, a subset of which constitutes the vertex set of the mesh.

We do not deal with the generation of sample points to maintain appropriate density because it may be obtained by different mechanisms in different situations. For example, Pandya, Gao and Hwang [14] imaged a moving bioprosthetic heart valve's leaflet to determine sample points by stereoscopy. Successive images were used to sample the deforming leaflet. Glimm et al. [12] described a numerical method in which the sample points are obtained by intersecting the moving surface with the edges of a background uniform grid. We leave the generation of the sample set as a separate issue, and focus on the problem of maintaining a deforming mesh given a set of dense sample points.

We assume that the surface does not change topology. However, both the geometry and the sampling density may change. The mesh is continually deformed as its vertices move. The problem is to stop the simulation at appropriate times and update the mesh, in order to maintain the mesh quality. A mesh is good if its approximation error is small – pointwise in terms of the Hausdorff distance from the surface, and normalwise in terms of the deviations between normals of triangles and surface normals nearby.

The related problems of meshing and reconstructing a surface in the static case has been actively studied [2, 6, 9, 10, 11]. The maintenance of geometric configurations under motion has been extensively studied within the realm of kinetic data structures [1, 5, 13]. There is no previous work on maintaining a deforming surface mesh with theoretical guarantees when the surface is unknown. One approach is to reconstruct the surface from the set P of sample points at each update. One may use one of the several provable surface reconstruction algorithms available [2, 10, 11]. Most of these algorithms run in time $O(|P|^2)$ since they compute the 3D Delaunay triangulation of P [2, 10], or uses sophisticated tools to achieve a bound of $O(|P| \log |P|)$ [11]. Naturally, one would like to avoid rebuilding the mesh at each update and, instead, use the spatial coherence to (i) design a simpler and faster mesh update algorithm, and (ii) determine the update times.

Our main result is that, under some mild assumptions on the motion and sampling, we can update the mesh in $O(|P|)$ time, making $O(A/(\varepsilon\gamma)^2)$ changes to the mesh, where A is the surface area, γ is the reach, and ε is the *sampling factor* (i.e., ε is the smallest value such that P is an ε-sample). We only require P to be an ε-sample and P can be arbitrarily dense in some local regions. We guarantee a small approximation error and a bounded radius-edge ratio (ratio of the circumradius to the smallest edge length of the triangles) at all times. Our update algorithm uses relatively simple primitives such as edge flips in a triangulation and constructing the 2D constrained Delaunay triangulation of $O(1)$ points. It builds upon our recent edge flip result [8]. During the update, we can insert new sample points and delete ex-

*Research of the first author is supported by Research Grant Council, Hong Kong, China (612107). Research by the second author is supported by NSF grants CCF-0430735 and CCF-0635008.

†Department of Computer Science and Engineering, HKUST, Clear Water Bay, Hong Kong. Email: scheng@cse.ust.hk

‡Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA. Email: tamaldey@cse.ohio-state.edu

isting sample points requested by the user. At the end of the update, our algorithm computes the time for the next update. We show that the approximation error remains small until the next update. In Section 6 we construct examples for which, under the same model, no other algorithm makes asymptotically fewer changes to the mesh in the entire simulation.

2 Preliminaries.

2.1 Background. Let $\Sigma \subset \mathbb{R}^3$ be a smooth compact surface without boundary. The *medial axis* is the set of centers of all maximally empty balls. The *reach* γ of Σ is the infimum of the distances of the points in Σ to its medial axis. For any $\varepsilon > 0$, a discrete set $P \subset \Sigma$ is an *ε-sample* of Σ if for any point $x \in \Sigma$, there exists $p \in P$ such that $\|p - x\| \leqslant \varepsilon\gamma$. We call P a *tight ε-sample* of Σ if P is an ε-sample and the distance between some point $x \in \Sigma$ and its nearest sample point is $\varepsilon\gamma$. For a tight ε-sample, we call ε its *sampling factor*. We call P an $(\varepsilon, \alpha\varepsilon)$*-sample* for some $\alpha \leqslant 1$ if P is an ε-sample and the distance between any two points in P is at least $\alpha\varepsilon\gamma$. We use $\mathbf{n}_x$ to denote a unit normal of Σ at a point $x \in \Sigma$. For a triangle pqr with vertices on Σ, we use $\mathbf{n}_{pqr}$ to denote its normal. For any two vectors, line segments or lines d_1 and d_2, $\angle d_1, d_2$ denote the acute angle between the support lines of d_1 and d_2. Two triangles sharing an edge e subtend two angles θ_1 and θ_2 around the edge with $\theta_1 + \theta_2 = 2\pi$. We call $\min(\theta_1, \theta_2)$ the *edge-angle* at e. We need several geometric results from previous work.

LEMMA 2.1. ([3, 7]) *For any two points $x, y \in \Sigma$ such that $\|x - y\| \leqslant \varepsilon\gamma$ for some $\varepsilon \leqslant \frac{1}{2}$, $\angle \mathbf{n}_x, \mathbf{n}_y \leqslant \frac{\varepsilon}{1-\varepsilon}$ and $\angle \mathbf{n}_x, (y - x) \geqslant \arccos(\frac{\varepsilon}{2})$.*

LEMMA 2.2. ([10]) *Let pqr be a triangle with vertices on Σ and circumradius at most $\varepsilon\gamma$. Assume that p subtends a maximal angle in pqr. Then $\angle \mathbf{n}_{pqr}, \mathbf{n}_p \leqslant \arcsin\varepsilon + \arcsin(\frac{2}{\sqrt{3}}\sin(2\arcsin\varepsilon))$.*

COROLLARY 2.1. *Let τ be a triangle with vertices on Σ and circumradius at most $\varepsilon\gamma$. For any vertex p of τ, $\angle \mathbf{n}_\tau, \mathbf{n}_p \leqslant 7\varepsilon$ for sufficiently small ε.*

2.2 Motion model. We assume that a compact, smooth closed surface deforms within a fixed time interval $[0, I]$, carrying with it a dense set of sample points. Any simulation must progress at discrete steps, so we assume a time unit of 1 and any update of the mesh can only be performed at integral times within $[0, I]$. A mesh maintenance algorithm needs to select a subset of integral times within $[0, I]$, called *time steps*, at which the updates are performed. The motion is frozen during the update at a time step and resumed at the end of the update. The updates should keep the mesh a good approximation at all times within $[0, I]$.

For any $t \in [0, I]$, we have the following notation: $\Sigma(t)$ = deforming surface, $P(t)$ = the set of sample points on $\Sigma(t)$, $\mathsf{Tri}(t)$ = surface mesh, $\mathsf{Vert}(t)$ = vertex set of $\mathsf{Tri}(t)$, $\gamma(t)$ = reach of $\Sigma(t)$, $\varepsilon(t)$ = sampling factor of $P(t)$. We require $\mathsf{Vert}(t) \subseteq P(t)$. Sometimes we abuse the notation $\mathsf{Tri}(t)$ to denote its underlying space as well.

The input sample $P(0)$ is required to be an ε-sample for sufficiently small ε. Notice that $\varepsilon(0) \leqslant \varepsilon$ as $P(0)$ is a tight $\varepsilon(0)$-sample by definition.

When the mesh is updated at a time step, the user is allowed to insert new sample points and delete existing sample points. It is a limitation that insertions and deletions cannot be be allowed at all times. If we run our mesh update algorithm whenever insertions and deletions occur, we can still maintain the surface mesh and its quality. However, we cannot show that our algorithm makes asymptotically the fewest changes to the mesh in this case. We discuss in Section 7 that if certain extra conditions are met, insertions and deletions can be allowed at all times.

Due to insertions and deletions, the set of sample points may change at a time step and so may the sampling factor. Thus, for a time step t, we distinguish between its beginning $\check{t}$ and its end $\hat{t}$. $P(\check{t})$, $\varepsilon(\check{t})$, and $\mathsf{Tri}(\check{t})$ may be different from $P(\hat{t})$, $\varepsilon(\hat{t})$, and $\mathsf{Tri}(\hat{t})$, respectively. The surface and its reach do not change from $\check{t}$ to $\hat{t}$, so they are still denoted by $\Sigma(t)$ and $\gamma(t)$.

At a time step t, $P(\check{t})$ may be changed by the insertions and deletions of sample points. Take the union of $P(\check{t})$ and the sample points to be inserted at t. We use ε_t to denote the sampling factor of this combined point set. Notice that the sample points to be deleted at t are ignored in defining ε_t.

We make several assumptions about the motion and sampling. First, we assume that we know $\Sigma(0)$, but not $\Sigma(t)$ for any $t > 0$. (In fact, we do not even need to know $\Sigma(0)$, but this simplifies our exposition.) We do not know the deformation of the surface. We only have access to the coordinates of the sample points at the current time. Second, for any $t \in [0, I]$, $\varepsilon(t)$ is sufficiently small and ε_t is greater than a fixed positive constant. We do not need to know $\varepsilon(t)$ or ε_t. Third, we are given an upper bound λ on the displacement of any point on the surface within a time unit. We assume that $\lambda \leqslant \min_{t\in[0,I]} 4\varepsilon(t)\gamma(t)/(5\kappa)$, where $\kappa \geqslant 468$ is a constant fixed in advance.

It is sometimes necessary for $\mathsf{Vert}(t)$ to be an ε-sample for some $\varepsilon \leqslant 1$ so that $\mathsf{Tri}(t)$ is a good approximation of $\Sigma(t)$; for example, when $\Sigma(t)$ is a sphere. If $\lambda > \varepsilon\gamma(t)$ for some t, one can construct an example in which a mesh triangle is inverted within a time unit.

This requires an upper bound of $\min_{t\in[0,I]} \varepsilon\gamma(t)$ on λ. Our bound on λ is within a constant factor $\frac{4}{5\kappa}$ of the best possible in the worst case.

We say that the *approximation error* at t is *small* if the Hausdorff distance between $\mathsf{Tri}(t)$ and $\Sigma(t)$ is $O(\varepsilon(t)\gamma(t))$, the normal of any triangle in $\mathsf{Tri}(t)$ makes an angle $O(\varepsilon(t))$ with surface normals nearby, and the edge-angles in $\mathsf{Tri}(t)$ are at least $\pi - O(\varepsilon(t))$.

2.3 Overview of our strategy. We first select a $(2\varepsilon(0), \varepsilon(0))$-sample $\mathsf{Vert}(0)$ from $P(0)$. In general, at any $t \geqslant 0$, the sample points not used as mesh vertices (i.e., $P(t) \setminus \mathsf{Vert}(t)$) are stored in the *dormant lists* of some vertices. For a vertex v, $\mathsf{dormant}(v)$ denotes its dormant list. (The points in dormant lists are still sample points on the surface and they also move with the surface. The sampling factor still refers to the sampling factor of $P(t)$.) We set $\mathsf{Tri}(0)$ to be either the restricted Delaunay triangulation of $\mathsf{Vert}(0)$ if $\Sigma(0)$ is known, or we can also run any provable reconstruction algorithm [10] on $\mathsf{Vert}(0)$ to construct $\mathsf{Tri}(0)$.

We keep some sample points in dormant lists so that the mesh vertices form a sparse sampling of the surface. This is necessary so that, at the next time step, we can repair the mesh using edge flips and perform insertions and deletions efficiently.

The next time step t_2 is computed at the end of the current time step t_1. For any $t \in [0, I]$, let $R(t)$ denote the maximum circumradius in $\mathsf{Tri}(t)$. We compute

$$t_2 = t_1 + \lceil R(\hat{t}_1)/(\kappa\lambda) \rceil.$$

That is, we will repair the mesh and perform the next batch of insertions and deletions specified by user at t_2. For $\lambda \leqslant \min_{t\in[0,I]} 4\varepsilon(t)\gamma(t)/(5\kappa)$, we show that the circumradii do not increase too much before t_2. This allows us to employ our recent edge flip result [8] at t_2 to bring the circumradii below some threshold. If a new sample point p to be inserted is close to a vertex v, we simply insert p into $\mathsf{dormant}(v)$. If p is far from all vertices, we split the triangle abc nearest to p and perform $O(1)$ edge flips to keep the circumradii small. The deletion of a vertex v is handled by projecting the link of v onto an appropriate plane, computing the constrained Delaunay triangulation of the projected link, and map the 2D triangulation to a triangulation of the link in 3D. Afterwards, $O(1)$ edge flips are performed to keep the circumradii small.

Theorems 2.1 and 2.2 summarize our main results.

THEOREM 2.1. *Assume that our motion model holds. For any time step $t > 0$, our update algorithm takes time linear in $|P(\check{t})|$ plus the number of new sample points inserted at t, and makes $O(\mathrm{area}(\Sigma(t))/(\varepsilon_t\gamma(t))^2)$ changes to the mesh. At any time t' from t to the next time step, $\mathsf{Vert}(t')$ is a $(6\varepsilon(t'), \frac{1}{5}\varepsilon(t'))$-sample and $\mathsf{Tri}(t')$ is a valid mesh with maximum circumradius at most $7\varepsilon(t')\gamma(t')$ and the approximation error at t' is small.*

THEOREM 2.2. *Assume that our motion model holds. There exists examples such that no algorithm can make asymptotically fewer changes to the mesh than our algorithm, while keeping the approximation error small at any $t \in [0, I]$.*

3 MeshUpdate.

Let $t > 0$ be a time step. We run a procedure MeshUpdate at t to restore the mesh quality.

MeshUpdate

1. Call RefineVert.
2. Call Insertion.
3. Call Deletion.

RefineVert improves the mesh quality. Insertion and Deletion handle the insertions and deletions requested by the user, respectively.

The goal of MeshUpdate is to guarantee the following *global invariants* at $\hat{t}$:

(i) $\mathsf{Vert}(\hat{t})$ is an $(3\varepsilon(\hat{t}), \frac{1}{15}\varepsilon(\hat{t}))$-sample.

(ii) $R(\hat{t}) \leqslant 3\varepsilon(\hat{t})\gamma(t)$ and for any $u, v \in \mathsf{Vert}(\hat{t})$, $\|u - v\| \geqslant R(\hat{t})/3$.

(iii) The dormant lists are disjoint. For any $v \in \mathsf{Vert}(\hat{t})$ and for any $p \in \mathsf{dormant}(v)$, $\|p - v\| \leqslant 5R(\hat{t})/4$.

(iv) The function that maps a point in $\mathsf{Tri}(\hat{t})$ to its nearest point in $\Sigma(t)$ is a homeomorphism.

In this section we show that the mesh quality does not deteriorate too much from $\hat{t}$ to the next time step, provided that the global invariants hold at $\hat{t}$. Based on this, we will show in Section 5 that RefineVert, Insertion, and Deletion can restore the global invariants at the end of the next time step. Theorem 2.1 then follows inductively.

3.1 Initialization. We need to initialize at time 0 so that the global invariants hold. Notice that $\varepsilon(0) = \varepsilon(\check{0}) = \varepsilon(\hat{0})$ as no point is inserted or deleted at time 0.

Assuming that $\Sigma(0)$ is given, one can compute $\varepsilon(0)\gamma(0)$ exactly from the restricted Delaunay triangulation. If $\Sigma(0)$ is not given, one may reconstruct a surface from $P(0)$, say with the Cocone algorithm [10], and then estimate $\varepsilon(0)\gamma(0)$ within a small constant factor using the maximum circumradius of the triangles. To simplify

the calculations, we assume that $\varepsilon(0)\gamma(0)$ is computed exactly from $P(0)$. After computing $\varepsilon(0)\gamma(0)$, we initialize the vertex set to contain an arbitrary point in $P(0)$. Then, we repeatedly insert the point in $P(0)$ whose distance from the current vertex set is largest. We stop growing the vertex set when this distance drops below $\varepsilon(0)\gamma(0)$. The final vertex set is $\mathsf{Vert}(0)$. For any point $x \in \Sigma(0)$, there exists $p \in P(0)$ such that $\|p - x\| \leqslant \varepsilon(0)\gamma(0)$. If $p \notin \mathsf{Vert}(0)$, by construction, there exists $v \in \mathsf{Vert}(0)$ such that $\|p - v\| \leqslant \varepsilon(0)\gamma(0)$. Thus, $\|x - v\| \leqslant 2\varepsilon(0)\gamma(0)$, implying that $\mathsf{Vert}(0)$ is a $2\varepsilon(0)$-sample. By construction, for any $u, v \in \mathsf{Vert}(0)$, $\|u - v\| \geqslant \varepsilon(0)\gamma(0)$. Hence, $\mathsf{Vert}(0)$ is a $(2\varepsilon(0), \varepsilon(0))$-sample. For any $p \in P(0) \setminus \mathsf{Vert}(0)$, we insert p into $\mathsf{dormant}(v)$, where v is the vertex nearest to p. We set $\mathsf{Tri}(0)$ to be the restricted Delaunay triangulation of $\mathsf{Vert}(0)$. (If $\Sigma(0)$ is unknown, reconstruct a surface from $\mathsf{Vert}(0)$ instead.) This completes the initialization.

LEMMA 3.1. *Let P be a tight ε-sample of a surface with reach γ. Let T be a mesh with vertex set $V \subseteq P$. If V is an $O(\varepsilon)$-sample, the maximum circumradius in T is at least $4\varepsilon\gamma/5$ for sufficiently small ε.*

Proof. Assume that the maximum circumradius is at most $\varepsilon\gamma$; otherwise, we are done. Let y be the point on the surface that achieves the distance $\varepsilon\gamma$ from P. Consider the 3D Voronoi diagram of V. Let a be the vertex whose Voronoi cell contains y. Because P is a tight ε-sample and V is an $O(\varepsilon)$-sample, $\|a - y\| \geqslant \varepsilon\gamma$ and $\|a - y\| = O(\varepsilon\gamma)$. For each triangle incident to a, we obtain an infinite planar wedge by extending the triangle beyond the edge opposite a. Let the wedge W induced by the triangle abc be the one closest to y. Let z be the point in W closest to y. We consider the case of z lying in the interior of W. The case of z lying on the boundary of W can be handled similarly. The acute angle between ay and W is at most $\angle \mathbf{n}_{abc}, \mathbf{n}_a$ plus the angle between ay and the tangent plane at a. By Lemma 2.1 and Corollary 2.1, this is $O(\varepsilon)$. Thus, $\|a - z\| \geqslant \|a - y\| \cos(O(\varepsilon)) \geqslant 4\varepsilon\gamma/5$ for sufficiently small ε. Consider the 2D Voronoi diagram of $\{a, b, c\}$ in the plane of abc. Because y projects orthogonally onto z and y lies in the 3D Voronoi cell owned by a, z lies in the 2D Voronoi region owned by a. Thus, the circumcenter of abc is further from a than z, i.e., the circumradius of abc is at least $4\varepsilon\gamma/5$. ◻

LEMMA 3.2. *The global invariants hold at* 0.

Proof. We showed before that $\mathsf{Vert}(0)$ is a $(2\varepsilon(0), \varepsilon(0))$-sample. Since $\mathsf{Tri}(0)$ is a restricted Delaunay triangulation, $R(0) \leqslant 2\varepsilon(0)\gamma(0)$. Thus, for any $u, v \in \mathsf{Vert}(0)$, $\|u - v\| \geqslant \varepsilon(0)\gamma(0) \geqslant R(0)/2$. By construction, the dormant lists are disjoint. Moreover, for any $v \in \mathsf{Vert}(0)$ and for any $p \in \mathsf{dormant}(v)$, $\|p - v\| \leqslant \varepsilon(0)\gamma(0)$, which is at most $5R(0)/4$ by Lemma 3.1. Since $\mathsf{Tri}(0)$ is a triangulated 2-manifold with maximum circumradius $O(\varepsilon(0)\gamma(0))$, the function that maps a point in $\mathsf{Tri}(0)$ to its nearest point in $\Sigma(0)$ is a homeomorphism [10]. ◻

3.2 Deterioration of the mesh. We quantify the deterioration of the mesh between two successive time steps in Lemma 3.6. We need some technical results.

LEMMA 3.3. *Let $t_1 < t_2$ be two successive time steps. Assume that the global invariants hold at $\hat{t}_1$. For any $t \in [t_1, t_2]$, $(t - t_1)\lambda \leqslant 2R(\hat{t}_1)/\kappa$.*

Proof. Follows from the definition of time steps. ◻

LEMMA 3.4. *Let $t_1 < t_2$ be two successive time steps. Assume that the global invariants hold at $\hat{t}_1$. Any edge in the surface mesh turns an angle no more than $\arcsin \frac{12}{\kappa - 12}$ from $\hat{t}_1$ to $\check{t}_2$.*

Proof. Consider an edge pq at $\hat{t}_1$. The length of pq decreases by at most $2(t_2 - t_1)\lambda \leqslant 4R(\hat{t}_1)/\kappa$ by Lemma 3.3. By the global invariants, the length of pq is at least $R(\hat{t}_1)/3$ at $\hat{t}_1$. So the length of pq is at least $(1/3 - 4/\kappa) \cdot R(\hat{t}_1)$ within $[\hat{t}_1, \check{t}_2]$. Let θ be the angle turned by pq. The displacement of q relative to p within $[\hat{t}_1, \check{t}_2]$ is at most $4R(\hat{t}_1)/\kappa$ by Lemma 3.3. Then, $\sin\theta$ is at most $4R(\hat{t}_1)/\kappa$ divided by $(1/3 - 4/\kappa) \cdot R(\hat{t}_1)$. ◻

LEMMA 3.5. *Let $t_1 < t_2$ be two successive time steps. Assume that the global invariants hold at $\hat{t}_1$. For any $t \in [\hat{t}_1, \check{t}_2]$, $\varepsilon(t)\gamma(t) \geqslant (\frac{1}{3} - \frac{4}{\kappa})R(\hat{t}_1)$.*

Proof. Let x be the point in $\Sigma(\hat{t}_1)$ whose distance from the nearest sample point in $P(\hat{t}_1)$ is $\varepsilon(\hat{t}_1)\gamma(\hat{t}_1)$. Therefore, at time t, the distance from x to the nearest sample point in $P(t)$ is at least $\varepsilon(\hat{t}_1)\gamma(\hat{t}_1) - 2(t - t_1)\lambda \geqslant \varepsilon(\hat{t}_1)\gamma(\hat{t}_1) - \frac{4}{\kappa}R(\hat{t}_1) \geqslant (\frac{1}{3} - \frac{4}{\kappa})R(\hat{t}_1)$ by Lemma 3.3 and the global invariants at $\hat{t}_1$. By the sampling condition, this distance is at most $\varepsilon(t)\gamma(t)$. ◻

LEMMA 3.6. *Let $t_1 < t_2$ be two successive time steps. Assume that $\kappa \geqslant 468$ and the global invariants hold at $\hat{t}_1$. The following hold for any $t \in [\hat{t}_1, \check{t}_2]$:*

(i) $R(t) < \frac{\kappa^2 - 144}{\kappa(\kappa - 348)} R(\hat{t}_1)$;

(ii) $\mathsf{Vert}(t)$ *is a* $(5\varepsilon(t), \frac{1}{15}\varepsilon(t))$*-sample;*

(iii) $4\varepsilon(t)\gamma(t)/5 \leqslant R(t) \leqslant 12\varepsilon(t)\gamma(t)$;

(iv) *for any* $v \in \mathsf{Vert}(t)$ *and for any* $p \in \mathsf{dormant}(v)$, $\|p - v\| \leqslant 4\varepsilon(t)\gamma(t)$.

Proof. Consider a triangle pqr at any time $t \in [\hat{t}_1, \check{t}_2]$. Let $p'q'r'$ denote this triangle at $\hat{t}_1$. Let ℓ' denote the shortest edge length of $p'q'r'$ at $\hat{t}_1$. By the global invariant, $\ell' \geqslant R(\hat{t}_1)/3$. Thus, the shortest edge length of pqr is at most $\ell' + 2(t - t_1)\lambda \leqslant \ell' + 4R(\hat{t}_1)/\kappa \leqslant (1 + 12/\kappa)\ell'$. Let θ' denote the smallest angle of $p'q'r'$ at $\hat{t}_1$. By Lemma 3.4, the smallest angle of pqr at t is at least $\theta' - 2\arcsin(12/(\kappa - 12))$.

Let R' and R denote the circumradii of $p'q'r'$ and pqr, respectively. Observe that $R' = \frac{\ell'}{2\sin\theta'}$ and $R \leqslant \frac{(1+12/\kappa)\ell'}{2\sin(\theta' - 2\arcsin(12/(\kappa-12)))}$. Therefore, $R \leqslant \frac{(1+12/\kappa)R'\sin\theta'}{\sin(\theta' - 2\arcsin(12/(\kappa-12)))}$. The right hand side is maximized when θ' is minimized. Since the global invariants imply that $\sin\theta' \geqslant 1/6$, $R \leqslant \frac{(1+12/\kappa)R'}{\cos\left(2\arcsin\frac{12}{\kappa-12}\right) - 6\sin\left(2\arcsin\frac{12}{\kappa-12}\right)\cos\left(\arcsin\frac{1}{6}\right)} \leqslant \frac{(1+12/\kappa)R'}{\cos\left(2\arcsin\frac{12}{\kappa-12}\right) - 6\sin\left(2\arcsin\frac{12}{\kappa-12}\right)}$. Then we simplify using the inequalities $\cos x > 1 - x$, $\sin x < x$, and $\arcsin x < 2x$ to obtain (i).

For any point $z \in \Sigma(t)$, there exists $p \in P(t)$ such that $\|p - z\| \leqslant \varepsilon(t)\gamma(t)$. If $p \notin \mathsf{Vert}(t)$, $p \in \mathsf{dormant}(v)$ for some $v \in \mathsf{Vert}(t)$. By the global invariant and the maximum displacement of points, $\|p - v\| \leqslant 5R(\hat{t}_1)/4 + 4R(\hat{t}_1)/\kappa$. Thus, $\|v - z\| \leqslant \varepsilon(t)\gamma(t) + (5/4 + 4/\kappa)R(\hat{t}_1)$. By Lemma 3.5, $\|v - z\| \leqslant \frac{19\kappa}{4\kappa-48}\varepsilon(t)\gamma(t) < 5\varepsilon(t)\gamma(t)$ for $\kappa \geqslant 468$. This proves the first half of (ii).

We prove (iii) first. Since $\mathsf{Vert}(t)$ is a $5\varepsilon(t)$-sample, by Lemma 3.1, $R(t) \geqslant 4\varepsilon(t)\gamma(t)/5$. By (i) and Lemma 3.5, $R(t) \leqslant \frac{\kappa^2-144}{\kappa(\kappa-348)}R(\hat{t}_1) \leqslant \frac{\kappa^2-144}{\kappa(\kappa-348)} \cdot \frac{3\kappa}{\kappa-12} \cdot \varepsilon(t)\gamma(t) \leqslant 12\varepsilon(t)\gamma(t)$ for $\kappa \geqslant 468$.

By the global invariant and the maximum displacement of points, the nearest neighbor distance in $\mathsf{Vert}(t)$ is at least $(1/3 - 4/\kappa)R(\hat{t}_1)$. By (i) and (iii), $R(\hat{t}_1) \geqslant \frac{\kappa(\kappa-348)}{\kappa^2-144}R(t) \geqslant \frac{4\kappa(\kappa-348)}{5(\kappa^2-144)}\varepsilon(t)\gamma(t)$. Substituting $\kappa \geqslant 468$, the nearest neighbor distance in $\mathsf{Vert}(t)$ is bounded from below by $\varepsilon(t)\gamma(t)/15$. This proves the second half of (ii).

The correctness of (iv) can be proved similarly using the global invariants, the maximum displacement of points, and Lemma 3.5. □

3.3 Faithful approximation. Is the mesh non-self-intersecting? For any $t \in [0, I]$, define the function $\mu_t : \mathsf{Tri}(t) \to \Sigma(t)$ that maps a point in $\mathsf{Tri}(t)$ to its nearest point in $\Sigma(t)$. Lemma 3.7 shows that μ_t is a homeomorphism, which implies that $\mathsf{Tri}(t)$ is non-self-intersecting. We omit the proof which is based on the continuity of the motion. A similar result was proved in [4] assuming that the triangles are Delaunay. This assumption may not hold in our case.

LEMMA 3.7. *Let $t_1 < t_2$ be two successive time steps. Assume that the global invariants hold at $\hat{t}_1$. For any $t \in [\hat{t}_1, \check{t}_2]$, μ_t is a homeomorphism.*

Lemma 3.7 implies that the edge-angles are greater than $\pi/2$ at any $t \in [\hat{t}_1, \check{t}_2]$. We can then apply Lemma 2.1 and Corollary 2.1 to show that the approximation error at any $t \in [\hat{t}_1, \check{t}_2]$ is small.

4 Edge flips.

Define the *diametric ball* of a triangle abc as the smallest ball containing a, b, and c on its boundary. We call the edge ab between two triangles abc and abd *flippable* if d lies in the diametric ball of abc. (Under the setting of this paper, if d lies in the diametric ball of abc, c lies in the diametric ball of abd too [8].) Flipping ab means that ab is flippable, and triangles abc and abd are replaced by acd and bcd. We extract the following theorem from [8].

THEOREM 4.1. ([8]) *Let V be an $(\varepsilon, \alpha\varepsilon)$-sample of a closed surface for some $\alpha \leqslant 1$ and sufficiently small ε. Let T be a mesh with vertex set V such that the maximum circumradius in T is $O(\varepsilon\gamma)$, where γ is the reach, and the edge-angles in T are at least $\pi - O(\varepsilon)$. Then: (i) among the triangles whose diametric balls contain a vertex $v \in V$, there is one with a flippable edge; (ii) if $abc, abd \in T$ and ab is flippable, the maximum circumradius of abc and abd is greater than the maximum circumradius of acd and bcd; (iii) repeated edge flips can make the diametric balls of all triangles empty of vertices; (iv) after any sequence of edge flips, the function that maps a point in the mesh to its nearest point on the surface is a homeomorphism.*

Theorem 4.1 implies that the surface mesh obtained by repeated edge flips is Delaunay. The lemma below strengthens the above result further. We omit its proof.

LEMMA 4.1. *Assume the same conditions in Theorem 4.1. For any $v \in V$, (i) there are $O(1)$ changes in the edges incident to v in any sequence of edge flips; (ii) we can perform $O(1)$ edge flips in $O(1)$ time to make v lie outside the diametric balls of all triangles.*

For any ball B and any $\beta \leqslant 1$, βB denotes the concentric ball with radius $\beta \cdot \text{radius}(B)$. The next result bounds the circumradius of a triangle τ if a "fraction" of its diametric ball is empty of any sample point.

LEMMA 4.2. *Let P be an ε-sample of a closed surface with reach γ. Let T be a mesh with vertex set $V \subseteq P$ such that V is an $O(\varepsilon)$-sample. Let B be the diametric ball of a triangle in T. If βB is empty of points in P for some $\beta \geqslant 2/5$,* $\text{radius}(B) \leqslant 3\varepsilon\gamma$ *for sufficiently small ε.*

Proof. Let B be the diametric ball of the triangle pqr. Let $R = \text{radius}(B)$. Let z be the center of B. It is known that $B \cap \Sigma$ is a topological disk that lies in a thin slab of width $O(\varepsilon^2\gamma)$ parallel to the equator of B. So the line through z orthogonal to the plane of pqr intersects Σ. Let x be the intersection point closest to z. By Lemma 2.1 and Corollary 2.1, the angle between px and pqr is $O(\varepsilon)$. Thus, $\|x - z\| \leqslant R\tan(O(\varepsilon))$. For $\beta \geqslant 2/5$ and sufficiently small ε, $\beta R - R\tan(O(\varepsilon)) \geqslant R/3$. So the ball B' centered at x with radius $R/3$ lies inside βB. As βB does not contain any point in P, neither does B'. So $R/3 \leqslant \varepsilon\gamma \Rightarrow R \leqslant 3\varepsilon\gamma$. □

5 Update routines.

Let t denote the current time step. We describe and analyze the procedures REFINEVERT, INSERTION, and DELETION. They assume certain conditions on the input and intermediate results, which we call the (k, α, ε)-*conditions* for some constants k and α:

(i) the set P of sample points is an ε-sample of $\Sigma(t)$ and the sampling factor of P is $\Omega(\varepsilon)$;

(ii) the vertex set $V \subseteq P$ is a $(k\varepsilon, \alpha\varepsilon)$-sample;

(iii) the maximum circumradius is at most $3k\varepsilon\gamma(t)$;

(iv) for any $v \in V$ and for any $p \in \mathsf{dormant}(v)$, $\|p - v\| \leqslant \frac{2k}{3}\varepsilon\gamma(t)$.

5.1 REFINEVERT**.** When REFINEVERT is called in MESHUPDATE, the $(5, \frac{1}{15}, \varepsilon(\check{t}))$-conditions hold by Lemma 3.6. Since REFINEVERT will also be called in INSERTION and DELETION, we analyze REFINEVERT assuming that the input satisfies the (k, α, ε)-conditions for general values of k, α, and ε, and that the sampling factor is ε.

REFINEVERT keeps inserting points in dormant lists as vertices until their distances from the current vertex set becomes less than a certain fraction of the maximum circumradius. So the largest diametric ball of triangles in the final mesh is empty of vertices and almost empty of sample points, which allows us to bound the maximum circumradius in terms of $\varepsilon\gamma(t)$. Note that the set of sample points do not change and neither does the sampling factor. Only the vertex set is updated. We maintain the $(k, \alpha', \varepsilon)$-conditions throughout the execution of REFINEVERT, where $\alpha' = \min\{\alpha, 1/15\}$.

REFINEVERT

1. For each vertex v, set S_v to be an empty list.
2. Loop
   (a) Set R to be the maximum circumradius of triangles in the current mesh.
   (b) For each vertex v, scan $\mathsf{dormant}(v)$ to remove the subset S_v of vertices p such that $\|p - v\| > R/3$.
   (c) While some S_v is non-empty, do:
      i. Pick a non-empty S_v and remove a point p from it.
      ii. Determine the vertex w and the triangle abc nearest to p.
      iii. If $\|p - w\| \leqslant R/3$, insert p to $\mathsf{dormant}(w)$; else call ADD(p, abc).
   (d) Flip edges to make the diametric balls of all triangles empty of vertices.

   until the maximum circumradius of triangles in the current mesh is greater than $2R/3$.

ADD(p, abc)

1. Split abc using p.
2. Flip edges to make the diametric balls of triangles incident to p empty.

Except for step 2(c) and ADD, the rest of REFINEVERT is self-explanatory and, in each iteration of the loop in step 2, they take time linear in the number of sample points. The maximum circumradius starts out to be at most $3k\varepsilon\gamma(t)$. For the loop in step 2 of REFINEVERT to iterate again, the maximum circumradius must decrease to two-third of its value or less. By Lemma 3.1, it is at least $4\varepsilon\gamma(t)/5$ at the termination of REFINEVERT. So there are $O(1)$ iterations of the while-loop at step 2.

In step 2(c), using the inductive assumption of the $(k, \alpha', \varepsilon)$-conditions, we can show that for any vertex v and any $p \in S_v$, $\|p - v\| \leqslant \frac{2k}{3}\varepsilon\gamma(t)$. So we can search the mesh from p in $O(1)$ time to find the nearest vertex w. We also find the nearest triangle abc in $O(1)$ time.

We elaborate on ADD. The splitting of abc works as follows. Let $\tilde{p}$ be the point on abc nearest to p. We assume that $\tilde{p}$ is in the interior of abc and so $p\tilde{p}$ is perpendicular to abc. The case of $\tilde{p}$ lying on the boundary of abc can be handled similarly. We connect $\tilde{p}$ to the vertices of abc to split it into three triangles. Consider $ab\tilde{p}$. If $\tilde{p}$ and the circumcenter of $ab\tilde{p}$ lie on opposite sides of ab and ab is flippable, then we flip ab.

If ab is flipped, this completes the splitting; otherwise, repeat the above to $ac\tilde{p}$ and $bc\tilde{p}$. (It is possible that no edge of abc is flipped in the end.) Afterwards, we move the vertex $\tilde{p}$ to the position of p, which deforms the mesh slightly. We claim that all new triangles in the final mesh have $O(1)$ radius-edge ratio. The proof is omitted. By Lemma 4.1(ii), step 2 of ADD can be done in $O(1)$ time. ADD has the following effect.

LEMMA 5.1. *If the $(k, \alpha', \varepsilon)$-conditions hold and the distance from p to any other vertex is at least $\alpha'\varepsilon\gamma(t)$, ADD$(p, abc)$ preserves the $(k, \alpha', \varepsilon)$-conditions.*

Proof. The insertion of p in step 1 of ADD may possibly violate the bound of $3k\varepsilon\gamma(t)$ on the maximum circumradius in the $(k, \alpha', \varepsilon)$-conditions. Nevertheless, step 1 ensures that the new triangles incident to p have $O(1)$ radius-edge ratio. Therefore, we can apply Lemma 4.1 for the flip sequence in step 2 of ADD. Then, we invoke Lemma 4.2 with both P and V being equal to the current vertex set. So the maximum circumradius in the star of p is at most $3k\varepsilon\gamma(t)$ at the end of ADD. If a pair of triangles not in the star of p are flipped, since the $(k, \alpha', \varepsilon)$-conditions held before the insertion of p, their maximum circumradii were at most $3k\varepsilon\gamma(t)$ before the flip and it can only decrease after the flip by Theorem 4.1. Other conditions are inherited or guaranteed by assumption. □

By Lemma 3.1, R is always greater than or equal to $4\varepsilon\gamma(t)/5$ in REFINEVERT. So when p is inserted in step 2(c)(iii), $\|p - w\| > R/3 \geqslant 4\varepsilon\gamma(t)/15 > \alpha'\varepsilon\gamma(t)$, where $\alpha' = \min\{\alpha, 1/15\}$. Then, the inductive maintenance of the $(k, \alpha', \varepsilon)$-conditions follows from Lemma 5.1 and the fact that step 2(d) can only decrease the maximum circumradius. At the termination of REFINEVERT, step 2(d) allows us to bootstrap from the $(k, \alpha', \varepsilon)$-conditions to obtain the following stronger guarantees.

LEMMA 5.2. *Assume that the (k, α, ε)-conditions hold and the sampling factor is ε. REFINEVERT performs $O(\text{area}(\Sigma(t))/(\varepsilon\gamma(t))^2)$ edge flips and takes $O(|P|)$ time, where P is the set of sample points. A mesh T with vertex set $V \subseteq P$ is returned such that: (i) V is an $(3\varepsilon, \alpha'\varepsilon)$-sampling of $\Sigma(t)$, where $\alpha' = \min\{\alpha, 1/15\}$; (ii) the maximum circumradius R in T is at most $3\varepsilon\gamma(t)$; (iii) for any $p \in P \setminus V$, $p \in \mathsf{dormant}(v)$ for some $v \in V$ and $\|p - v\| \leqslant R/2$.*

Proof. The last execution of step 2(b) ensures that for any vertex v and for any $p \in \mathsf{dormant}(v)$, $\|p - v\|$ is at most $1/3$ of the maximum circumradius at that time. Afterwards, the maximum circumradius may decrease to at least $2/3$ of its original value for the loop to terminate. Therefore, (iii) is correct.

Let pqr be a triangle with maximum circumradius at the termination of REFINEVERT. Let R be its circumradius. Let B be the diametric ball of pqr. We claim that $\frac{1}{2}B$ is empty of any sample point. If there is a sample point inside $\frac{1}{2}B$, it would be at distance greater than $R/2$ from other vertices. Thus, this sample point would be at distance greater than $R'/3$ from other vertices at the last execution of step 2(b), where $R' \leqslant 3R/2$ was the maximum circumradius then. This is a contradiction because this sample point would have been inserted as a new vertex in step 2(c). By our claim and Lemma 4.2, $R \leqslant 3\varepsilon\gamma(t)$, which proves (ii).

The lower bound of $\alpha'\varepsilon\gamma(t)$ on the intervertex distances follows from the inductive maintenance of the $(k, \alpha', \varepsilon)$-conditions. For any point $z \in \Sigma(t)$, there exists $p \in P$ such that $\|p - z\| \leqslant \varepsilon\gamma(t)$. If $p \notin \mathsf{Vert}(t)$, $p \in \mathsf{dormant}(w)$ for some $w \in \mathsf{Vert}(t)$. By (iii), $\|p - w\| \leqslant R/2$. Thus, if v is the nearest vertex in $\mathsf{Vert}(t)$ to z, then $\|v - z\| \leqslant \|p - z\| + \|p - w\| \leqslant \varepsilon\gamma(t) + R/2 < 3\varepsilon\gamma(t)$. This shows that the vertices form a $(3\varepsilon, \alpha'\varepsilon)$-sampling.

The total running time of one iteration of the loop in step 2 is clearly $O(|P|)$ based on our discussion of the implementation. By the $(k, \alpha', \varepsilon)$-conditions, the number of vertices on $\Sigma(t)$ is $\Theta(\text{area}(\Sigma(t))/(\varepsilon\gamma(t))^2)$ throughout one iteration of the loop. As discussed before, each call of ADD generates $O(1)$ edge flips. By Theorem 4.1, the number of edge flips in step 2(d) is linear in the number of vertices. So REFINEVERT performs $O(\text{area}(\Sigma(t))/(\varepsilon\gamma(t))^2)$ edge flips in one iteration of the loop. How many iterations are there? The maximum circumradius must decrease to two-third of its value or less for the loop to iterate again. We know that the maximum circumradius starts out to be at most $3k\varepsilon\gamma(t)$. By Lemma 3.1, it is at least $4\varepsilon\gamma(t)/5$ at the end. Therefore, there are $O(\log k) = O(1)$ iterations. This proves the time complexities stated. □

For the call of REFINEVERT in MESHUPDATE, we instantiate Lemma 5.2 with $\varepsilon = \varepsilon(\check{t})$, $k = 5$, and $\alpha = \alpha' = 1/15$. If there is no insertion or deletion at the time step t, REFINEVERT already restores the global invariants at $\hat{t}$: the working of REFINEVERT guarantees that for any $u, v \in \mathsf{Vert}(\hat{t})$, $\|u - v\| \geqslant R(\hat{t})/3$; the correctness of invariant (iv) follows from Theorem 4.1(iv); the other invariants follow from Lemma 5.2.

5.2 INSERTION**.** Let R be the maximum circumradius in the current mesh. For each new sample point p to be inserted, we find the vertex w and the triangle abc nearest to p. If $\|p - w\| \leqslant R/3$, insert p into $\mathsf{dormant}(w)$; otherwise, call ADD(p, abc). After inserting the new sample points, INSERTION calls REFINEVERT and returns.

Recall that ε_t is the sampling factor of the union of $P(\check{t})$ and the new sample points to be inserted at the time step t. Although the sampling factor drops from $\varepsilon(\check{t})$ to ε_t during the insertion, $\varepsilon(\check{t})/\varepsilon_t = O(1)$ by our motion model. When inserting a new sample point p, it may be reasonable to assume that a vertex v close to p is also given in the input. Then, it suffices to search the mesh from v in $O(1)$ time to find the vertex and triangle nearest to p. If such information is unavailable in the input, we can invoke a more expensive search as follows. We implicitly divide $\mathbb{R}^3$ into a uniform grid with cell width $30R(\check{t})$. By making one of the occupied cell the "origin", the cells can naturally be given integer coordinates, with each coordinate in the range $[-|P(\check{t})|, |P(\check{t})|]$. For each $v \in \mathsf{Vert}(\check{t})$, we compute the coordinates i_v of the grid cell occupied by $\mathsf{Vert}(\check{t})$ in $O(1)$ time. Clearly, at most $|P(\check{t})|$ grid cells are occupied. Therefore, we can radix-sort the i_v's for all $v \in \mathsf{Vert}(\check{t})$ in lexicographical ordering in $O(|P(\check{t})|)$ time. Then, in linear time we obtain a list $\mathcal{L}$ of sets: each set contains the vertices in the same grid cell and $\mathcal{L}$ is sorted in lexicographical order of the coordinates of the grid cells storing the sets. Similarly, we can compute the sorted list of coordinates of the cells occupied by the new sample points, as well as the cells adjacent to these occupied cells. By merging this list with $\mathcal{L}$, for each new sample point p, we obtain the grid cell containing p and its nine adjacent cells. We check all vertices in these ten cells to find the vertex w_p nearest to p. The above takes time linear in the size of the mesh plus the number of new sample points.

When a new sample point p is actually inserted, w_p may not be the nearest vertex as new vertices may have been added by previous insertions. Still, we can search the mesh from w_p in $O(1)$ time to find the vertex and triangle nearest to p. The correctness follows from an analysis similar to that of RefineVert.

Lemma 5.3. *At a time step t,* Insertion *takes time linear in $|P(\check{t})|$ plus the number of new sample points inserted, and it performs $O(\mathrm{area}(\Sigma(t))/(\varepsilon_t\gamma(t))^2)$ edge flips. Afterwards, the properties stated in Lemma 5.2 hold with $\varepsilon = \varepsilon_t$ and $\alpha = \varepsilon(\check{t})/(15\varepsilon_t)$.*

5.3 Deletion**.** By Lemma 5.2 or Lemma 5.3 depending on whether points have been inserted, the input to Deletion satisfies the $(3, \frac{1}{15}, \varepsilon_t)$-conditions. First, we mark the sample points to be deleted. The marked points in dormant lists are deleted right away. We cannot delete the marked vertices all at once. Otherwise, the density of the vertices may become very uneven and it is unclear how to obtain the global invariants afterwards. Instead, Deletion proceeds in iterations.

Deletion

While there is a marked sample point, do:

1. Delete the marked points in dormant lists.
2. Call UserDeletion.
3. Call Decimation.
4. Call RefineVert.

UserDeletion

1. Set R to be the maximum circumradius.
2. While there is a marked vertex, do:
   (a) Pick a marked vertex v.
   (b) Collect two sets S_1 and S_2 of marked vertices w such that $S_1 = \{w \text{ marked} : \|v - w\| \leqslant 2R\}$ and $S_2 = \{w \text{ marked} : 2R < \|v - w\| \leqslant 4R\}$.
   (c) Find an unmarked vertex u at distance $4R$ or less from v. Find a vertex $w^* \in S_1$ such that $\mathsf{dormant}(w^*)$ is non-empty.
   (d) If neither u nor w^* was found, call Remove(w) for each $w \in S_1$ and return.
   (e) For each $w \in S_1 \cup S_2$, call Remove(w).
   (f) If u was found, for each $w \in S_1 \cup S_2$, append $\mathsf{dormant}(w)$ to $\mathsf{dormant}(u)$. Otherwise, w^* was found and we do:
      i. Remove some q from $\mathsf{dormant}(w^*)$.
      ii. Find the triangle abc nearest to q. Call Add(q, abc).
      iii. Concatenate $\mathsf{dormant}(w)$ for all $w \in S_1 \cup S_2$ to form $\mathsf{dormant}(q)$.

Decimation

1. Set R to be the maximum circumradius. Color all vertices white.
2. For each white vertex v, color all vertices w such that $\|v - w\| < R/2$ black.
3. While there is a black vertex, do the following:
   (a) Pick a black vertex u.
   (b) For each point $p \in \mathsf{dormant}(u) \cup \{u\}$, find the white vertex w nearest to p and insert p into $\mathsf{dormant}(w)$.
   (c) Call Remove(u).

In each iteration we delete the marked points in dormant lists first. Then, we call UserDeletion which delete marked vertices until the sampling factor has increased by a certain factor greater than 1. After UserDeletion returns, Decimation is called to decimate the mesh to match the increased sampling factor. Then, RefineVert is called to restore the $(3, \frac{1}{15}, \varepsilon)$-conditions, where ε is the current sampling factor. This is repeated until no marked sample point is left.

UserDeletion and Decimation use a subroutine Remove which works as follows. Remove(v) assumes that the vertex set is a $(O(\varepsilon), \Omega(\varepsilon))$-sample, where ε is the current sampling factor. It takes the plane H of a triangle incident to v, project the link of v onto H, and compute the 2D constrained Delaunay triangulation of the projected link. Based on the sampling conditions, we can show that this 2D triangulation has constant size and bounded radius-edge ratio. We map the 2D triangulation to the triangulation of the link of v in 3D. Because of the sampling condition, the angles decrease by $O(\varepsilon)$ and so the triangulation in 3D also has bounded radius-edge ratio. Finally, we flip edges to make the triangles incident to vertices in the link of v empty of vertices. Notice that there are only $O(1)$ vertices in the link of v. Clearly Remove(v) can be implemented in $O(1)$ time. The edge flips allow us to show that the maximum circumradius at the completion of Remove(v) is at most $3k\varepsilon\gamma(t)$, if the vertex set is a $k\varepsilon$-sample for some $k \geqslant 1$.

Let ε be the sampling factor at the beginning of UserDeletion. Both UserDeletion and Decimation require and preserve the $(k_1, \alpha, k_2\varepsilon)$-conditions for some constants k_1, α and k_2 throughout their execution. The sampling factor ε' at the end of Decimation lies between ε and $k_2\varepsilon$. So we can rewrite the $(k_1, \alpha, k_2\varepsilon)$-conditions as the $(\frac{k_1 k_2 \varepsilon}{\varepsilon'}, \frac{\alpha\varepsilon}{\varepsilon'}, \varepsilon')$-conditions. Then, by Lemma 5.2, RefineVert restores the $(3, \frac{1}{15}, \varepsilon')$-conditions and so the next iteration in Deletion works correctly.

If neither u nor w^* is found in UserDeletion, there is no sample point within a distance of $2R$ from v. Then, we can show that the sampling factor has gone up by a factor greater than 1, and so UserDeletion returns. In Decimation, the current maximum circumradius R reflects the increased sampling factor. Decimation clearly enforces that the intervertex distance is at least $R/2$ at the end.

In Deletion, since the sampling factor increases by a constant factor greater than 1 from one iteration to the next, there are $O(\log(\varepsilon(\hat{t})/\varepsilon_t)) = O(1)$ iterations. Each iteration takes time linear in the number of sample points and performs $O(\text{area}(\Sigma(t))/(\varepsilon_t\gamma(t))^2)$ edge flips. Using Theorem 4.1(iv), one can show that $\mu_{\hat{t}}$ is a homeomorphism between $\mathsf{Tri}(\hat{t})$ and $\Sigma(t)$. The last call of Decimation ensures that any two vertices are at distance $R/2$ or more. By Lemma 5.2, the last call of RefineVert guarantees that the other conditions in the global invariants hold at $\hat{t}$.

Lemma 5.4. *At a time step t,* Deletion *takes time linear in $|P(\check{t})|$ plus the number of new sample points inserted, and does $O(\text{area}(\Sigma(t))/(\varepsilon_t\gamma(t))^2)$ flips. The global invariants hold afterwards.*

Our first main result, Theorem 2.1, follows from Lemmas 3.6, 5.2, 5.3, 5.4, and 3.7.

6 Examples.

We describe examples for which no algorithm can perform asymptotically fewer changes to the mesh than our algorithm (Theorem 2.2). We ignore the initialization. Let $\mathcal{A}$ denote any other algorithm. The surface is a sphere with radius γ in our first example. Our algorithm and $\mathcal{A}$ are given the same set of sample points. To approximate the sphere well, $\mathcal{A}$ must maintain a mesh whose vertex set is an ε-sample for some $\varepsilon \leqslant 1$. Consider the time step 0. Let T be the mesh maintained by $\mathcal{A}$ at time 0. The size of T is $\Omega(1/\varepsilon^2)$. A constant fraction of the triangles in T have $O(1)$ neighboring triangles. This allows us to find a set S of $\Omega(1/\varepsilon^2)$ triangles in T so that no two share a vertex. Take a triangle $abc \in S$. We design the motion such that: (i) a first moves on the sphere to a point close to the midpoint of bc; (ii) a then moves slightly away from the sphere center; and (iii) a reverses the motion to return to its original position. Let the speed of a be $\lambda = \theta\gamma$ throughout. The motion of a can be accomplished in time $O(\varepsilon\gamma/\lambda) = O(\varepsilon/\theta)$. $\mathcal{A}$ must destroy abc before (iii); otherwise, (ii) will make abc nearly perpendicular to the sphere, which makes it impossible for $\mathbf{n}_{abc}$ to approximate any surface normal nearby. Hence, $\mathcal{A}$ must destroy all of the triangles in S within a time interval $\Gamma = [0, O(\varepsilon/\theta)]$. This amounts to $\Omega(1/\varepsilon^2)$ changes to the mesh within Γ.

Take any time step $t > 0$ scheduled by our algorithm that falls inside Γ. Our algorithm makes $O(1/\varepsilon_t^2)$ changes to our mesh at t, which is within a constant factor of $\Omega(1/\varepsilon^2)$. By Lemma 3.1, the gap between t and the next time step is $\Omega(R(\hat{t})/\lambda) = \Omega(\varepsilon(\hat{t})\gamma/\lambda) = \Omega(\varepsilon(\hat{t})/\theta)$, which is a constant fraction of Γ. The same holds for any other time step scheduled by our algorithm that falls inside Γ. So our algorithm schedules only $O(1)$ time steps in Γ. We charge the edge flips performed at these time steps to the mesh changes performed by $\mathcal{A}$ within Γ. We take the next time step t' of our algorithm not covered by Γ. This induces another time interval Γ' containing t' in which $\mathcal{A}$ must destroy $\Omega(1/\varepsilon^2)$ triangles. Γ' is disjoint from Γ because the motion progresses

forward in time. So we can repeat the analysis above. In all, $\mathcal{A}$ does not perform asymptotically fewer changes to the mesh than our algorithm in the entire simulation.

What if we had called MeshUpdate at every time unit instead? We would have $O(1/\theta)$ updates within each of the intervals Γ, Γ', etc. Recall that θ controls the speed of points and it can be very small. When θ is very small, the problem should be easier because the sample points move less. However, if we had called MeshUpdate at every time unit, we could have performed a factor of $O(1/\theta)$ more changes to the mesh than $\mathcal{A}$, which is arbitrarily large as θ approaches zero.

We can generalize the above construction to work for an arbitrary surface, if we further require $\mathcal{A}$ to keep the longest edge length at most ε times the reach for some constant $\varepsilon \leqslant 1$. In this case, the size of the mesh maintained by $\mathcal{A}$ is $\Omega(A/(\varepsilon\gamma)^2)$, where A is the surface area. The rest of the argument is the same as before.

Because the sample points can be arbitrarily dense in some local regions, the mesh size may be much less than the number of sample points. Thus, our examples do not say anything on whether our running time is small or large. It seems unlikely that one can avoid looking at all the sample points regularly though.

7 Discussion.

In our model, insertions of sample points are only allowed during updates at time steps scheduled by our algorithm. It is because we can only afford to run RefineVert at the time steps in order to prove that our algorithm makes asymptotically the fewest changes to the mesh. Suppose that we call Insertion but not RefineVert in the end to handle insertions between two successive time steps. The sampling factor may be reduced by a large constant factor, which will invalidate the lower bound on $\varepsilon(t)\gamma(t)$ in Lemma 3.5 and hence the rest of the analysis. If there is a known constant upper bound on the decrease in the sampling factor caused by insertions between two successive time steps, we can factor this constant upper bound into the analysis.

Stronger conditions are needed for deletions. In addition to a known constant upper bound on the increase in the sampling factor caused by deletions, we need to assume that deletions between two successive time steps do not make the density of mesh vertices highly uneven. Under this assumption, we can delete the vertices specified by the user all at once.

Given these assumptions, the number of changes to the mesh made by us is asymptotically no more than the number of insertions and deletions plus the number of changes to the mesh made by any other algorithm.

Some open questions ensue with this research. Can the sampling density condition be relaxed to be sensitive to the local feature size instead of the reach? Can insertions and deletions be allowed at any time without extra assumptions while making asymptotically minimum changes to the mesh? How can changes in the surface topology be accommodated?

References

[1] P.K. Agarwal, Y. Wang, and H. Yu. A 2D kinetic triangulation with nearly quadratic topological changes. *Proc. 20th Annu. Sympos. Comput. Geom.* (2004), 180–189.

[2] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete Comput. Geom.* 22 (1999), 481–504.

[3] N. Amenta and T. K. Dey. Normal variation with adaptive feature size. A note as an erratum to Lemma 2 in [2].

[4] N. Amenta, S. Choi, T. K. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Internat. J. Comput. Geom. Appl.* 12 (2002), 125–141.

[5] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *Proc. 8th ACM-SIAM Sympos. Discrete Alg.* (1997), 747–756.

[6] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67 (2005), 405–451.

[7] H.-L. Cheng, T. K. Dey, H. Edelsbrunner and J. Sullivan. Dynamic skin triangulation. *Discrete Comput. Geom.* 25 (2001), 525–568.

[8] S.-W. Cheng and T.K. Dey. Delaunay edge flips in dense surface triangulations. arXiv:cs.CG/0712.1959, 2007. Available from authors' web-pages.

[9] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. *Proc. 20th Annu. Sympos. Comput. Geom.*, 2004, 280–289.

[10] T. K. Dey. *Curve and surface reconstruction : Algorithms with mathematical analysis.* Cambridge University Press, New York, 2006.

[11] S. Funke and E.A. Ramos. Smooth-surface reconstruction in near-linear time. *Proc. 13th Annu. ACM-SIAM Sympos. Discrete Alg.*, 2002, 781–790.

[12] J. Glimm, J.W. Grove, X.L. Li, and D.C. Tan. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM J. Sci. Comput.*, 21 (1999), 2240–2256.

[13] L. Guibas. Kinetic Data Structures: a state of the art report. *Proc. 3rd Workshop Alg. Found. Robotics*, 1998.

[14] S. Pandya, B.Z. Gao, and N.H.C. Hwang. Bioprosthetic heart valve leaflet deformation monitored by double pulse stereo photogrammetry. *Proc. First Joint BMES/EMBS Conference*, 1999.