State of the Art: Updating Delaunay Triangulations for Moving Points

Olivier Devillers, Pedro Machado Manhães de Castro

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

State of the Art: Updating Delaunay Triangulations for Moving Points

Olivier Devillers — Pedro Machado Manhães de Castro

N° 6665

October 2008

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--6665--FR+ENG

INRIA
SOPHIA ANTIPOLIS

State of the Art: Updating Delaunay Triangulations for Moving Points

Olivier Devillers , Pedro Machado Manhães de Castro

Thème SYM — Systèmes symboliques
Projets Geometrica

Rapport de recherche n° 6665 — October 2008 — 12 pages

Abstract: This paper considers the problem of updating efficiently a two-dimensional Delaunay triangulation when vertices are moving. We investigate the three current state-of-the-art approaches to solve this problem: –1– the use of *kinetic data structures*, –2– the possibility of moving points from their initial to final position by deletion and insertion and –3– the use of "almost" Delaunay structure that postpone the necessary modifications. Finally, we conclude with a global overview of the above-mentioned approaches while focusing on future works.

Key-words: Computational Geometry, Delaunay Triangulation, Moving Points

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Etat de l'Art: Mis à Jour des Triangulations de Delaunay des Points qui Bougent

Résumé : Ce travail considère le problème de mettre à jour de manière efficace une triangulation de Delaunay de dimension deux où les sommets subissent des perturbations. Nous examinons les trois approches principales dans la littérature pour résoudre le problème considéré: –1– l'utilisation des *structures de données cinétique*, –2– la possibilité de bouger les points de leurs positions initiales jusqu'à leurs positions finales par suppression et réinsertion et –3– l'utilisation de structures *pseudo-Delaunay* qui retardent les modifications nécessaires. Finalement, nous concluons avec une vision globale des approches mentionnées ainsi qu'en mettant en évidence des pistes pour des travaux futurs.

Mots-clés : Géométrie Algorithmique, Triangulation Delaunay, Points qui Bougent

1 Introduction

Delaunay triangulations of a set of points is one of the most famous data structures produced by computational geometry. Two main reasons explain this success: –1– computational geometers eventually produce efficient algorithms to compute it, and –2– it actually has effective usage such as meshing for finite elements methods or surface reconstruction from point clouds.

For several applications the data are moving and thus the triangulation evolves with time. It arises for example when meshing deformable objects, or in some algorithms computing the point's positions by variational methods. In this report we focus on existing algorithm to deal with Delaunay triangulations of set of moving points.

We first recall that the *Delaunay triangulation* $DT(\Omega)$ of a set Ω of n points in $\mathbb{R}^d$ is a *simplicial complex* such that no point in Ω is inside the circumsphere of any *simplex* in $DT(\Omega)$. Comprehensive texts can be found in [17, 5]. A lot of algorithms to compute the Delaunay triangulation are available in the literature [14, 10] most of them works in a *static* setting, that is the points are fixed and known in advance. It exists also a variety of so-called *dynamic* algorithms, in which the points are fixed but not known in advance and thus the triangulation is maintained under insertions and deletions of new points. If some of the points move continuously in $\mathbb{R}^d$ and we want to keep track of the modifications of the triangulations, we speak of *kinetic* algorithms. Finally an important variation is when the points move but we are only interested in the triangulation at some discrete times, we call that context *timestamps moving*.

The aim of this report is to describe and compare various techniques that can be used in the context of timestamps moving points in two and three dimensions. A first trivial solution consists in using an efficient static algorithm and recompute everything at each timestamp, we call this *rebuilding*. Then we discuss several state-of-the-art approaches to optimize the running time of updating Delaunay triangulations for moving points in two and three dimensions by exploiting the spatial coherence between two consecutive time stamps. We investigate the use of kinetic data structures, the possibility of moving points from their initial to final position by deletion and insertion and the use of "almost" Delaunay structure that postpone the necessary modifications. For each approach, we compare its performance against rebuilding's.

2 Kinetic Data Structures

To represent a continuous motion on geometric data structure, one could simply associate a function of time to each one of its primitives, which are usually points on the *euclidian space*. Such functions are called *trajectories*. Moreover, functions applied on those moving points become also a function of time.

A *predicate* is a function on a set of primitives which returns one of a discrete set of outputs (usually positive, zero, negative). The actual numerical value obtained in the predicate computation is the *predicate discriminant* and the sign of this value is the *result of the predicate*. When one or more predicates are used to evaluate whether a geometric data structure is valid or not, we call each one of those predicates a *certificate*. When the primitives move along a trajectory, certificates become *certificate functions* of time.

Without loss of generality, a given certificate is valid while its corresponding certificate function is non-negative. For continuous trajectories, if a given geometric structure is valid, then it remains valid until at least one of its predicate functions value changes sign. Therefore, the roots of the certificate functions with respect to time represent an instability on their respective predicate discriminant. An *event* occurs, namely, when a certificate fails. Hence, the geometric data structure need to be updated in order to preserve its validity. Geometric data structures maintained under this motion model, which was introduced by Basch et. al. in 1997 [6], are called *kinetic data structures*.

Efficient kinetic data structure have been created to maintain a relevant number of geometric data structures, such as the maximum of a set of numbers [2], the sorted order [2, 23], convex-hull [6], closest pair [6], collision detection [1, 7], Delaunay triangulation in two and three dimensions [3, 26], and Regular Triangulation [20, 26] for instance.

General Algorithm. Primitives are allowed to change their trajectory arbitrarily at any time, as long as they remain continuous. Such events are called *trajectory update*. Whenever a trajectory update occurs all the certificates involved must be re-solved in order to guarantee the validity of the structure. We may simplify the continuous trajectories of the primitives to polynomials, thus each coordinate could be represented by a polynomial of time. Then, most predicate functions are also polynomials of time, and their roots can be handled exactly. Note that a motion need not to be infinite, and could be defined between two timestamps t_a, t_b. For any set $\mathcal{S}$ of certificate functions C_i, we may briefly schematize the kinetic data structures algorithm as follow:

- Input moving points $\mathbf{z}(t)$, as trajectory functions defined on the interval $[t_a, t_b]$.
- For any certificate function $C_i \in \mathcal{S}$ evaluated on the moving points input, compute a set of discrete certificate failure times. In other words, the roots r_{ij} of the certificate function C_i.
- Insert each root $r_{ij} \in [t_a, t_b]$ in a priority queue associated to an event. The head of the priority queue corresponds to the smallest root inserted.
- Take the first root in the priority queue. Fetch the associated event. Handle it by updating the geometric data structure, then removing the root from the priority queue. And so generate new certificate functions.
- When there is no more motion, no events hereby occurs. And the algorithm ends.

Four distinct measures should be taken into account when analyzing such algorithms:

- *Locality:* how many certificates depend on each primitive. This affects how much work needs to be done on a trajectory update.
- *Responsiveness:* worst case amount of work required to process a single event.
- *Efficiency:* the total cost of processing all events over some period.
- *Compactness:* the total number of certificates needed by the kinetic data structure.

Kinetic data structures performing well on those measures are said to be: local, responsive, efficient and compact. For a more detailed background on the general algorithm, please refer to [18] and [26].

Delaunay Triangulation. As aforementioned, scientific community had considered working with kinetic data structures model to solve the problem of updating Delaunay triangulations when its vertices are not fixed anymore [3, 20, 26]. To model Delaunay Triangulation with kinetic data structures, we should consider its certificate functions, event processing, event generation and its primitives trajectories, then we can instantiate the general kinetic data structure algorithm above-mentioned.

The correctness of a non-self-intersecting Delaunay triangulation in two or three dimensions with a single extra vertex at infinity connected to each hull edge [9, 8], can be checked using two different types of certificates. They are:

- *In-circle certificate.* In two dimensions, for each interior edge of the triangulation, a certificate that the circle through the three vertices on one side does not contain the vertex on the other side. Therefore, this certificate is applied to any four distinct points $\mathbf{z}_1(x_1, y_1)$, $\mathbf{z}_2(x_2, y_2)$, $\mathbf{z}_3(x_3, y_3)$, $\mathbf{z}_4(x_4, y_4)$ of the triangulation such that they belong to the same bi-face (two triangles sharing an edge). Mathematically, the certificate function equation for two dimensions is the determinant of the following matrix:

$$\begin{pmatrix} x_1(t) & y_1(t) & x_1(t)^2 + y_1(t)^2 & 1 \\ x_2(t) & y_2(t) & x_2(t)^2 + y_2(t)^2 & 1 \\ x_3(t) & y_3(t) & x_3(t)^2 + y_3(t)^2 & 1 \\ x_4(t) & y_4(t) & x_4(t)^2 + y_4(t)^2 & 1 \end{pmatrix} \tag{1}$$

  The situation is analogous for three dimensions. The name of the certificate becomes *in-sphere* and it is applied to any five distinct points $\mathbf{z}_1(x_1, y_1, z_1)$, $\mathbf{z}_2(x_2, y_2, z_2)$, $\mathbf{z}_3(x_3, y_3, z_3)$, $\mathbf{z}_4(x_4, y_4, z_4)$, $\mathbf{z}_5(x_5, y_5, z_5)$ of the triangulation such that they belong to the same bi-cell (two tetrahedra sharing a triangle). The certificate function equation is the determinant of the following matrix:

$$\begin{pmatrix} x_1(t) & y_1(t) & z_1(t) & x_1(t)^2 + y_1(t)^2 + z_1(t)^2 & 1 \\ x_2(t) & y_2(t) & z_2(t) & x_2(t)^2 + y_2(t)^2 + z_2(t)^2 & 1 \\ x_3(t) & y_3(t) & z_3(t) & x_3(t)^2 + y_3(t)^2 + z_3(t)^2 & 1 \\ x_4(t) & y_4(t) & z_4(t) & x_4(t)^2 + y_4(t)^2 + z_4(t)^2 & 1 \\ x_5(t) & y_5(t) & z_5(t) & x_5(t)^2 + y_5(t)^2 + z_5(t)^2 & 1 \end{pmatrix} \tag{2}$$

- *convex-hull certificate.* In two dimensions, for each convex-hull edge, a test that the interior triangle adjacent to it is properly oriented. This test is applied to any three consecutive points $\mathbf{z}_1(x_1, y_1)$, $\mathbf{z}_2(x_2, y_2)$,

$\mathbf{z}_3(x_3, y_3)$ of the triangulation's convex-hull. The certificate function equation is the determinant of the following matrix:

$$\begin{pmatrix} x_1(t) & y_1(t) & 1 \\ x_2(t) & y_2(t) & 1 \\ x_3(t) & y_3(t) & 1 \end{pmatrix} \tag{3}$$

Again, the situation is analogous for three dimensions. And the certificate function equation is the determinant of the following matrix:

$$\begin{pmatrix} x_1(t) & y_1(t) & z_1(t) & 1 \\ x_2(t) & y_2(t) & z_2(t) & 1 \\ x_3(t) & y_3(t) & z_3(t) & 1 \\ x_4(t) & y_4(t) & z_4(t) & 1 \end{pmatrix} \tag{4}$$

In two dimensions, a *bad edge* is an edge such that the in-circle certificate fails for its corresponding bi-face. Analogously, in three dimensions, a *bad facet* is a facet such that the in-sphere certificate fails for its corresponding bi-cell. For Delaunay triangulations, an event being processed means updating its combinatorial structure when one or more certificates fail. In two dimensions, for an embedded Delaunay triangulation it is well-known that flipping the bad edges successively suffices to update the entire combinatorial structure. Thus, in two dimensions an event can be processed by simply flipping the bad edge and generating four new certificates each time. Some useful considerations on the triangulation embedding exist, one may reference to [26] and [3] for further details. Though, in three dimensions flip does not suffice to update the combinatorial structure [21]. In that case, one may directly consider vertex insertion and removal as a way to process events, then generate certificates for each new cells. Russel [26] has found, however, that flipping works much of the time. In three dimensions, flipping produces six near in-circle certificates each time.

At this point, remind that, in the motion problem we are considering here, primitives on a Delaunay triangulation are allowed to move discretely at a finite sequence of timestamps. However, in kinetic data structures points move continuously. The trajectory of primitives for each interval between two timestamps could be any function of time. Thus, we should consider what kind of interpolation would be adequate between two consecutive timestamps. This allows our problem to be handled by kinetic data structures.

A natural choice for the trajectory interpolant is a polynomial of time, mainly because exact algebraic equation solver is available [27] for the subsequent certificate functions polynomials. Moreover, we are interested in trajectory updates, which require continuity. Taking a look at Equation 1 and Equation 2, we may notice that using arbitrary polynomials as interpolant will lead to high dimension certificate functions. Since, at some point, we compute their roots, some strategy has to be considered to chose the right trajectory between two consecutive timestamps.

Several strategies was proposed by Russel's in his PhD Thesis [26]. We present them here as a small survey. Let $S(d)$ be the time needed for finding roots on a polynomial of degree d, f the number of cells in the initial triangulation and e_{motion} the number of events which occur when the points are moved to their displaced coordinates with a certain type of motion.

- **Minimizing trajectory changes strategy.** They presented two different strategies focusing on not allowing trajectories changes. Hence, the multiplicative constant of f will not be greater than 1.
  - *Linear interpolation (LI).* Pick a single motion for each point that interpolates its initial and displaced positions. In other words, allow x, y, z to change linearly at the same time. The resulting certificates have degree four in two dimensions and five in three dimensions. The cost to maintain the structure is then $(f + 6e_{LI})S(5)$ in three dimensions.
  - *Lifted Linear interpolation (LLI).* Look at Equation 2, $l = {x_i}^2 + {y_i}^2 + {z_i}^2$ is the coordinate of (x_i, y_i, z_i) on the *lifted space* [16]. If the given kinetic data structure can also handle the added complexity of regular triangulations [4], which is the case of [26], the coordinate on the lifted space can be interpolated as well, thus decreasing the degree of the polynomial by one. The cost to maintain the structure becomes $(f + 6e_{LLI})S(4)$ in three dimensions.
- **Minimizing degree.** They showed two different strategies focusing on minimizing the certificate function polynomial degree. Hence, the argument of S will not be greater than 1.
  - *Point at a time (PAT).* If we manage to vary each row on Equation 2 one at a time, we get linear polynomials as certificate functions. It corresponds to move each point as in *LLI*, but one after

another. Since each certificate must be recomputed five times, in three dimensions, one for each point involved, the cost becomes $(5f + 6e_{PAT})S(1)$.

– *Coordinate at a time (CAT).* If, instead, we vary each column one at a time, it corresponds to linearly interpolate all points along each coordinate successively. The total number of trajectory changes becomes three, as we shift from x to y, y to z, and z to l. Therefore, the cost becomes $(4f + 6e_{CAT})S(1)$.

Several representative data sets from simulations were used to adequately estimate the performance of those methods. All these following data sets, tables and benchmarks can be found on the original work of Russel in [26] with much more details.

- From **molecular simulations**, *hairpin* and *protein A*. A short 177 atom beta hairpin and Staphyloccocal protein A, a 601 atom globular protein.
- From **muscle simulations**, *bicep*. A volumetric data from a simulation of a bicep contracting. The points move comparatively little between the frames.
- From **falling object simulations**, *falling objects* and *shifting objects*. Those sets are taken from a simulation of a collection of 1000 objects dropped into a pile. Initially, the objects fall through the air with occasional contacts, but later in the simulation they form a dense, although still shifting pile. The two data sets are related with those two distinct phases of the simulation, and are called *falling objects* and *shifting objects* respectively.

Further details on those data sets are shown in Table 1.

Table 1: *Attributes of the static Delaunay triangulation: Ephemeral cells* are cells created during the construction process which are not in the final triangulation. There were generally three time as many ephemeral cells as cells in the final triangulation. Their number gives some idea of the excess work done by the rebuilding process. *Tol.* is the average fraction of the local edge length that points must have to invalidate an in-circle certificate. The 20% *tol.* is the fraction of the local average edge length that points need to move to destroy 20% of the certificates. Very small motions compared to the edge length can invalidate cells in a Delaunay triangulation.

name	*points*	*cells*	*ephemeral cells*	*in-circle tests*	*tol.*	20% *tol.*
hairpin	177	1114	2554	7516	8.5	1.5
protein A	601	3943	10250	31430	8.6	1.5
bicep	3438	21376	66553	210039	9.0	1.6
falling objects	1000	6320	17137	52958	12	1.7
shifting objects	1000	6381	17742	55299	13	1.2

Naively implemented kinetic data structures are not competitive with rebuilding, even ignoring exactness issues as it is shown in Table 2. The running time is dominated by the initialization: generating the costs and solving the certificate function for each face of the initial triangulation. For the nonlinear certificates the computation is complicated by the need to multiply polynomials of unknown degree, requiring many loops and memory management. For the linear certificate function based motions they were able to use a specialized polynomial representation which removed much of this overhead.

The fastest naive kinetic Delaunay based method was the *coordinate at a time* interpolation with *point at a time* interpolation being almost as fast. The latter algorithm actually generally had fewer events, due to the smaller changes in the weights. However its extra initialization cost made it slightly slower. Both techniques were about a factor of two more expensive than recomputing the triangulation.

As stated before, even adopting anyone of the aforementioned strategies, the naive approach of kinetic data structure running time is far behind the rebuilding's one. One way to try to reduce the cost of maintaining the combinatorial structure, is to adopt some filtering techniques. Four layers of filtering were proposed by Russel [26]. Naturally, if none of the filters succeed, the sign of the roots of the certificate function are computed exactly, using an exact polynomial equation solver, [27] for example.

- *Layer 0: avoiding activation.* This filter acts on the certificate generation step of the general algorithm. It tries to avoid redundancies by allowing a small set of vertices to "jump" directly to their final position when it would not break any certificate on that position.

Table 2: *Kinetic Delaunay update costs:* Speedups compared to rebuilding and event counts for different interpolating motions are shown. Note that all algorithms are slower than rebuilding and so all the costs are smaller than one. The coordinate at a time based methods are within a factor of two of rebuilding.

motion	*hairpin*	*protein A*	*bicep*	*falling objects*	*shifting objects*
linear	.04	.03	.04	.06	.06
lifted linear	.07	.08	.09	.10	.07
coordinate at a time	.50	.62	.39	.48	.54
point at a time	.40	.30	.40	.51	.44
rebuilding (in ms)	15	69	430	153	120

- *Layer 1: interval arithmetic.* Again, this filter acts on the certificate generation step of the general algorithm. In-circle predicates, Equation 2 with t as the current timestamp value, are first evaluated with interval arithmetic [24, 11], then if 0 is included on the output interval, they are computed exactly.
- *Layer 2: using the derivative.* Now, certificates are already generated. This filter uses a lower bound on the derivative of the certificate function, to try to prove that a certificate never fails, until the next timestamp.
- *Layer 3: approximate the failure time.* This filter uses Newton's method algorithm [25] to produce approximations of the timestamps where the certificate function vanishes. In other words, it finds the intervals where the roots of the certificate function are contained, the *root intervals*. If successful, this approach can prove that the certificate function has no roots in the interval over which it is being solved, or find an interval which contains the first root.

The following table indicate the filter failure ratio achieved by each filtering layer.

Table 3: *Filter failures:* The table shows the failure rates for each level of filters. On the bordeline data, there were many more failures of the early filters, which were caught by the derivative filter in level two (after the certificate function was generated). The inputs are divided into *good* and *borderline* sets. The former are data sets where the filters (especialy filter 0 and 1) perform well and the update procedure works effectively, the latter where they do not work as well.

data set	*filter 0*	*filter 1*	*filter 2*	*filter 3*
good	0.04	0.12	0.54	0.00
borderline	0.68	0.29	0.06	0.02

After mixing all those ingredients together, the kinetic data structure approach may be significantly faster than rebuilding with less than 1% bad certificates overall.

3 Static Techniques

When moving primitives from a structure between two discrete timestamps, one may not be interested in any intermediate states. In such a situation, there are a family of static techniques which only involves computing predicates on the initial and final coordinates and directly transforming the initial triangulation into the final answer. Hence, those techniques are comparatively easier to make them run faster than their kinetic counterpart, i.e. looking for all intermediate events during the motion.

One trivial instance of this family of schemes is the *placement* method. It consists into, first, taking each vertex and walking to the cell containing its displaced point coordinates, then inserting it in the cell, and, finally, removing the old vertex from the triangulation. Note that one removal and insertion are done. Recall that the incremental Delaunay triangulation construction algorithm [14], for each vertex to be inserted, would do a point location and an insertion. In practice, removal is more expensive than point location. Therefore, when all points move, placement is slower than rebuilding. Whenever points move far from its initial position,

it becomes even worse, since additional point locations are required. Guibas and Russel[19] worked on designing better approaches than placement. They came up with a few methods which we will present next.

Let $(\mathcal{T}, \Phi)$ signify assigning the coordinates of the point set Φ to the vertices of the triangulation $\mathcal{T}$, and $\mathcal{D}(\mathcal{T})$ signify the Delaunay triangulation of $\mathcal{T}$. Thus, for any two point sets Ω and Ω', the Delaunay combinatorial structures of Ω and Ω' are equal when $(\mathcal{D}(\Omega), \Omega')$ is Delaunay. To verify if $(\mathcal{D}(\Omega), \Omega')$ is Delaunay, check whether all cells have valid in-circle certificate or not, respectively. Whenever $(\mathcal{D}(\Omega), \Omega')$ is not Delaunay, one could remove a set of points Ψ one by one, in such a way that $(\mathcal{D}(\Omega \backslash \Psi), \Omega')$ is Delaunay. Then reinsert Ψ on the triangulation. Since $(\mathcal{D}(\emptyset), \Omega')$ is trivially Delaunay, removing $\Psi = \Omega$ suffices to guarantee the correctness of this technique. This kind of static technique is called *point removal*. Note: the order of removals matters. Therefore, the points could have some sort of score to evaluate the removal order. A total of three distinct heuristics to compute the score of points was considered on the work of Guibas and Russel[19], they are:

- *random.* Each vertex is given a score of one if it is involved in an invalid certificate, and zero otherwise. This means picking any points which is involved in an invalid certificate.
- *worst first.* Each vertex is assigned a score based on how many failed certificates it is involved in.
- *farthest first.* Each vertex is assigned a score based on how much the displacement has locally distorted the geometry of the point set. An example of that score's computation would be to compute, for each point $\mathbf{z}$ in Ω, the optimal rigid motion $\mathcal{O}$ from Ω to Ω' of its link vertices. And each point's score becomes the distance between $\mathcal{O}(\mathbf{z})$ and the location of $\mathbf{z}$ in Ω'. Clearly, this heuristic uses more expensive computations then the two above-mentioned ones, however it uses not only combinatorial properties of the sets Ω and Ω', but geometrical properties as well.

On the aforementioned process, the vertex removal algorithm is likely to be called several times on each iteration. It is well known that the vertex removal algorithm is expensive if compared with flips and insertions, please refer to [15]. Moreover, whenever points are removed and placed anywhere else, many new cells and facets are created, leading to a significant quantity of certificates generated. On the other hand, flips are much cheaper and generate a constant number of certificates, four for two dimensions and six for three dimensions. If the triangulation is embedded, then, in two dimensions, any triangulation can be made Delaunay with such flips. In three dimensions, although it is not guaranteed, in most of the cases, it is still possible. Whenever it is not possible to make a three dimensional triangulation Delaunay, we call that case, namely, a *stuck case*.

With this in mind, an alternative algorithm is to, first remove a set of points Ψ from Ω, such that $(\mathcal{D}(\Omega \backslash \Psi), \Omega')$ is embedded, then try to use successive Delaunay edge flips to finish the conversion to Delaunay. For two dimensional triangulation this algorithm works well and fast. Conversely, for three dimensions, the previous solution, removing points, can no longer be safely applied in the stuck cases, since the triangulation is no longer necessarily embedded using the original coordinates. Moreover, after the displacements, the triangulation is not necessarily Delaunay anymore and the hole created by removing a point from a non-Delaunay triangulation in three dimensions may not be able to be filled with cells [28]. Thus, an alternative for stuck cases, would be to find a point which is adjacent to a non-Delaunay cell, but which can be removed from the triangulation using the new coordinates. If even that alternative would fail, then the method would just recompute the triangulation from scratch, using the preexisting connectivity to speed up point location. This static technique is called *hybrid*, and works well for both two and three dimensions, if only because, in practice, the stuck cases do not happen often enough.

We can find benchmarks of those methods applied on the same data sets as within Table 1. They come from the original work of Russell [26], where more details can be found. From those benchmarks, of the purely point removal techniques, the *worst first* heuristic achieved the best running time, followed closely by the *random* heuristic. In general, they outperformed rebuilding when 1-2% of the cells were invalid after displacement. The fastest method overall was the *hybrid* method which could outperform rebuilding when even around 10% of the cells were invalidated by the displacement. Nevertheless, the frequency of getting stuck cases was less than 10% per timestamp in average for almost every of their data sets. The performances are showed in Table 4. Their attempt to use the actual geometry of the motion to order points for removal was useless for many reasons, i.e. rigid transforms computation cost was too large, and so further results were omitted.

If we compare those methods with kinetic data structures, we learn that not taking intermediate states into account is well-suited for discrete timestamps motion. Which is indeed something to not overlook when designing further algorithms to maintain a Delaunay triangulation for moving points.

Table 4: *Static update performance:* The table shows the speedups to patch the triangulation compared to rebuilding and the number of removals for the presented heuristics. Entries in **boldface** are ones which are better than rebuilding (i.e. larger than one).

simulation	*step*	*random*	*worst first*	*hybrid*
hairpin	1	.65	.65	**1.4**
protein A	1	.32	.42	**1.5**
	2	.06	.24	**1.1**
bicep	1	**1.8**	**2.0**	**3.1**
	2	**1.3**	**1.3**	**2.7**
	4	.70	.78	**2.3**
	8	.08	.10	.73
falling objects	1	**1.4**	**1.6**	**3.5**
	2	**1.2**	0.93	**3.5**
	4	.38	.53	**2.0**
	8	.81	.22	**1.4**
shifting objects	1	.46	.60	**2.2**
	2	.42	.44	**1.2**

4 Quasi-Delaunay Structures

Not every application needs to maintain the Delaunay property on its triangulation at each single timestamp, but only some *"nice"* cells shape structures. In that case, it may be relevant to guarantee some significantly cheaper quasi-Delaunay property on the combinatorial structure, instead of the Delaunay's one.

Some effort has been done by the scientific community in this direction, specially for surface reconstruction and re-meshing problems. For instance, DeCougny and Shephard [12] used a quasi-Delaunay property based on an empty circumsphere criterion in the three-dimensional euclidean space as a sub-step to re-mesh a surface. Debard et al. [13] used also a quasi-Delaunay property to lessen the computation time of their reconstruction algorithm primitive displacement portion.

Debard et al. illustrates well how one could reduce the cost of reconstructing deforming surfaces with quasi-Delaunay structures. Generally, a reconstruction algorithm, as the one they used, based on minimizing energy functions, works as follow:

(1) sampling a set of points Ω from the surface;

(2) discretizing the surface by triangulating Ω (typically, the chosen triangulation is the Delaunay triangulation);

(3) minimize an energy function such as the Gaussian energy [29] on the cells of the triangulation. Since the third step is hard, they use a scheme, as done by Meyer et al [22], which consists of a simple coordinate descent method where only one point is free at a time.

It is exactly in step (3) that their algorithm would benefit from a faster Delaunay triangulation update for moving points. They relaxed the Delaunay condition for their cells, and then to use a pseudo-Delaunay scheme. Actually there is a trade-off between convergence speed and the iterations running time, for details please refer to [13].

In their pseudo-Delaunay scheme, they proposed a lazy update, which associates a score to each vertex, indicating how strongly its position would violate the embedding and Delaunay properties after a move:

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

When the score of a vertex reaches s_{limit}, the vertex is moved by insertion followed by a removal. Experimentally, they have founded that $(s_1, s_2, s_{limit}) = (2, 1, 4)$ is a good trade-off between *Delaunayness* of the structure and convergence speed.

Their benchmark consisted of moving three thousand points from a random initial distribution of points to a well-shaped ellipsoid. The experiment reflected the relative cost of each subroutine on the average, using placement and using their quasi-Delaunay scheme. As we will see in Table 5, the greatest portion of calculation is the removal in both cases. However, the alternative method could save around half the time due to removal operations. Note that, in both case, moving points is the bottleneck of their reconstruction algorithm.

Table 5: Benchmarks, taken from the work of Debard et al. [13].

algorithm	subroutine	CPU time (%)
placement move		92.59
	remove	74.31
	in_circle_predicate	6.58
	insert	2.90
	embedding_predicate	0.62
	locate	0.06
internal		7.41
quasi_Delauany move		81.15
	remove	49.00
	in_circle_predicate	11.59
	insert	2.70
	embedding_predicate	2.06
	locate	0.07
internal		18.85

With such efforts, Debard et al. achieved a three time factor enhancement on the computation time of moving points, if compared with the insert/remove scheme. Although this is a significant improvement on speed, we should remind that the combinatorial structure is not Delaunay for each timestamps, which was good enough for them, however it may be not sufficient for general applications.

5 Summary

We have reviewed several methods to compute the Delaunay triangulation of a set of moving points at some timestamps. The main objective is to use spatial coherence between two consecutive positions to increase the time performance compared to rebuilding from scratch.

Three families of method have been reviewed that can be classified in kinetic methods, methods using checking and point's insertion and deletion and weak Delaunay approaches.

The main idea of kinetic methods is to move all points from their positions at one timestamp to the next one, tracking all topological changes in the triangulation during the motion. If we are only interested in timestamps and not eventually by the complete motion, several strategies can be used to move between the two positions, moving all points according to one coordinate at a time being the most efficient one. Such method is efficient if the triangulation is almost unchanged, that is less than 1% of the simplices are modified.

The second strategy consists in checking the validity directly at the final position and make some repairs where it is necessary. Compared to the previous method, the drawback is that the change between initial and final position is more difficult to compute but the advantage is that we avoid to sort the topological changes along time and that we may save some unnecessary modifications appearing and disappearing between the two timestamps. If we compared to rebuilding from scratch, the main disadvantage is that repairing or removing a point is much more difficult than inserting and thus such method is competitive only if the modifications remains small enough, that is if less than 10% of the simplices are modified.

First strategy looks between timestamps, the second strategy only at timestamps, finally, the third strategy goes one step forward and may forget some modifications if the result is not too far from Delaunay triangulation. Of course this strategy can be used only for some special applications where an approximated Delaunay triangulation can be enough.

As we could see, new efficient algorithms design in this domain should not overlook the overheads suffered by kinetic data structure based methods. Future works include improving the static techniques. As we saw, we gain in the point location computation time, though we spend more time regenerating the data structures

between two timestamps. The time regenerating the data structure can be improved. Also, we shall analyze wether *geometric filtering* techniques enhance the performance of the in-sphere certificates or not.

6 Acknowledgements

The authors thank the *ANR Triangles* [1], contract number `ANR-07-BLAN-0319`, and region PACA support for this work.

References

[1] M. A. Abam, M. de Berg, S.-H. Poon, and B. Speckmann. Kinetic collision detection for convex fat objects. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 4–15, London, UK, 2006. Springer-Verlag.

[2] Mohammad Ali Abam, Pankaj K. Agarwal, Mark de Berg, and Hai Yu. Out-of-order event processing in kinetic data structures. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 624–635, London, UK, 2006. Springer-Verlag.

[3] G. Albers, Leonidas J. Guibas, Joseph S. B. Mitchell, and T. Roos. Voronoi diagrams of moving points. *Internat. J. Comput. Geom. Appl.*, 8:365–380, 1998.

[4] F. Aurenhammer. Power diagrams: Properties, algorithms and applications. Technical Report F120, Techn. Univ. Graz, Graz, Austria, 1983.

[5] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, September 1991.

[6] J. Basch, L. J. Guibas, C. Silverstein, and L. Zhang. A practical evaluation of kinetic data structures. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 388–390, 1997.

[7] Julien Basch, Jeff Erickson, Leonidas J. Guibas, John Hershberger, and Li Zhang. Kinetic collision detection between two simple polygons. *Comput. Geom. Theory Appl.*, 27(3):211–235, 2004.

[8] CGAL Editorial Board. *CGAL User and Reference Manual*, 3.2 edition, 2006. http://www.cgal.org.

[9] Jean-Daniel Boissonnat, Frédéric Cazals, Frank Da, Olivier Devillers, Sylvain Pion, François Rebufat, Monique Teillaud, and Mariette Yvinec. Programming with CGAL: The example of triangulations. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 421–423, 1999.

[10] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. Triangulations in CGAL. *Comput. Geom. Theory Appl.*, 22:5–19, 2002.

[11] H. Brönnimann, C. Burnikel, and S. Pion. Interval arithmetic yields efficient dynamic filters for computational geometry. *Discrete Applied Mathematics*, 109:25–47, 2001.

[12] H. L. de Cougny and M. S. Shephard. Surface meshing using vertex insertion. In *Proc. 5th International Meshing Roundtable*, pages 243–256, PO Box 5800, MS 0441, Albuquerque, NM, 87185-0441, 1996. Sandia National Laboratories. Also Sand. Report 96-2301.

[13] Jean-Baptiste Debard, Romain Balp, and Raphaëlle Chaine. Dynamic Delaunay Tetrahedralisation of a Deforming Surface. *The Visual Computer*, page 12 pp, August 2007.

[14] Olivier Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[15] Olivier Devillers and Monique Teillaud. Perturbations and vertex removal in a 3d delaunay triangulation. In *Proc. 14th ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 313–319, 2003.

[16] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. In *Proc. 1st Annu. ACM Sympos. Comput. Geom.*, pages 251–263, 1985.

[1] http://www-sop.inria.fr/geometrica/collaborations/triangles/

[17] P. J. Green and R. R. Sibson. Computing Dirichlet tessellations in the plane. *Comput. J.*, 21:168–173, 1978.

[18] L. J. Guibas. Kinetic data structures — a state of the art report. In P. K. Agarwal, L. E. Kavraki, and M. Mason, editors, *Proc. Workshop Algorithmic Found. Robot.*, pages 191–209. A. K. Peters, Wellesley, MA, 1998.

[19] Leonidas Guibas and Daniel Russel. An empirical comparison of techniques for updating delaunay triangulations. In *Proc. 20nd Annu. ACM Sympos. Comput. Geom.*, pages 170–179, 2004.

[20] Leonidas J. Guibas, Feng Xie, and Li Zhang. Kinetic collision detection: Algorithms and experiments. In *ICRA*, pages 2903–2910, 2001.

[21] Barry Joe. Three-dimensional triangulations from local transformations. *SIAM J. Sci. Stat. Comput.*, 10(4):718–741, 1989.

[22] M. Meyer, P. Georgel, and R.T. Whitaker. Robust particle systems for curvature dependent sampling of implicit surfaces. In *In Proceedings of the International Conference on Shape Modeling and Applications (SMI)*, pages 124–133, June 2005.

[23] Victor Milenkovic and Elisha Sacks. An approximate arrangement algorithm for semi-algebraic curves. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 237–246, New York, NY, USA, 2006. ACM.

[24] Sylvain Pion. Interval arithmetic: An efficient implementation and an application to computational geometry. In *Workshop on Applications of Interval Analysis to systems and Control*, pages 99–110, 1999.

[25] N. Revol. Newton's algorithm using multiple precision interval arithmetic. *Numerical Algorithms*, 34(2):417–426, 2003.

[26] D. Russel. *Kinetic Data Structures in Practice*. PhD thesis, Stanford University, 2007.

[27] Daniel Russel, Menelaos I. Karavelas, and Leonidas J. Guibas. A package for exact kinetic data structures and sweepline algorithms. *Comput. Geom. Theory Appl.*, 38(1-2):111–127, 2007.

[28] Jonathan R. Shewchuk. A condition guaranteeing the existence of higher-dimensional constrained Delaunay triangulations. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 76–85, 1998.

[29] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 269–277, New York, NY, USA, 1994. ACM.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399