Discrete Comput Geom 24:721–733 (2000)
DOI: 10.1007/s004540010060

Discrete & Computational
Geometry
© 2000 Springer-Verlag New York Inc.

Lower Bounds for Kinetic Planar Subdivisions*

P. K. Agarwal,[1] J. Basch,[2] M. de Berg,[3] L. J. Guibas,[2] and J. Hershberger[4]

[1]Center for Geometric Computing, Department of Computer Science, Duke University,
Box 90129, Durham, NC 27708-0129, USA
pankaj@cs.duke.edu

[2]Computer Science Department, Stanford University,
Stanford, CA 94305, USA
{jbasch, guibas}@cs.stanford.edu

[3]Department of Computer Science, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
markdb@cs.uu.nl

[4]Mentor Graphics Corp., 8005 SW Boeckman Road,
Wilsonville, OR 97070-7777, USA
john_hershberger@mentorg.com

Abstract. We revisit the notion of kinetic efficiency for noncanonically defined discrete attributes of moving data, like binary space partitions and triangulations. Under reasonable computational models, we obtain lower bounds on the minimum amount of work required to maintain any binary space partition of moving segments in the plane or any Steiner triangulation of moving points in the plane. Such lower bounds—the first to be obtained in the kinetic context—are necessary to evaluate the efficiency of kinetic data structures when the attribute to be maintained is not canonically defined.

* Part of this work was done while P.A. and M.d.B. were visiting Stanford University. Work by P.A. was supported by Army Research Office MURI Grant DAAH04-96-1-0013, by NSF Grants EIA-9870724, and CCR-9732787, by a Sloan fellowship, by an NYI award, and by a grant from the U.S.–Israeli Binational Science Foundation. Work by J.B. was supported by Army Research Office MURI Grant DAAH04-96-1-0013. Work by L.G. was supported by National Science Foundation Grant CCR-9623851 and by U.S. Army Research Office MURI Grant DAAH04-96-1-0007.

1. Introduction

Given a set S of n pairwise-disjoint polygonal objects in the plane, a *constrained convex subdivision* of the plane is a convex subdivision $\Pi = \Pi(S)$ of the entire plane so that each face of Π either lies inside an object of S or lies in the common exterior of the objects in S. Such subdivisions arise in several applications, most notably computer graphics and collision detection. For example, a common approach to answering ray-tracing queries amidst a set of polygons is to compute a constrained triangulation Π and trace each query ray ρ through Π, visiting only those triangles of Π that intersect ρ [1]. In many cases (e.g., answering point-location queries), we want Π to be hierarchical in the sense that Π is constructed by starting with the entire plane as a single polygon and subdividing a face of the current subdivision into $O(1)$ convex faces until a constrained subdivision is obtained. Examples of hierarchical constrained subdivision include quad-trees [12], kd-trees [8], and binary space partitions (BSP) [18]. Motivated by various applications, constrained subdivisions have been extensively studied in computational geometry and related application areas [9], [17].

In a growing number of applications, we do not have a single set S because the objects move over time. This is the case, for instance, in video games, virtual reality, and dynamic simulations. The set S is now replaced by a continuous family $S(t)$ indexed by time. A constrained subdivision Π computed for the initial scene $S(0)$ cannot be guaranteed to remain valid as the objects move, and it becomes necessary to *maintain* a subdivision $\Pi(t)$ over time. As objects move or deform continuously, we expect that a fixed subdivision (or, rather, its combinatorial description based on the objects' features) will change only at discrete time instances. The maintenance of a subdivision should therefore proceed in discrete steps.

Relatively scant attention has been paid so far in the literature on maintaining a constrained subdivision. In the case of BSPs, which are widely used in graphics, most existing work is based on using dynamic BSPs. These approaches discretize time into short intervals of fixed duration Δt. At the end of each interval, they delete the moving objects from the structure and re-insert them in their new positions; see, for example, [10], [15], and [19].

In the case of triangulations, the Arbitrary Eulerian–Lagrangian method [11] provides a way to integrate the motion of fluids and solids within a moving finite-element mesh. Here again, time is discretized and the mesh vertices are moved between each time step so as to respect the interfaces between the different media. Numerical problems arise when the mesh becomes too distorted. It is therefore necessary to check its consistency every Δt, for an appropriately chosen time interval Δt. Different techniques can be used to correct the distortions, ranging from completely rebuilding the mesh with interpolation [20] to more economical node relocation and multilevel mesh methods [13].

All these approaches suffer from the fundamental problem that it is difficult to choose the length of the discretization interval. If Δt is too small, then the subdivision does not change combinatorially, and the deletion/re-insertion (for the BSP) or the consistency check (for the moving mesh) are just wasted computations; if Δt is too large, important changes to the subdivision can be missed, with ill effects on applications using the subdivision. This problem is further exacerbated because the rate of change can vary across the subdivision, thus making it impossible to find a time step size that is globally good.

Combinatorial descriptions of subdivisions are discrete attributes of the input set, and it is natural to consider the problem of their maintenance in the context of kinetic data structures.

The kinetic approach of Basch et al. [7] is a general method to maintain a discrete attribute of objects in predictable motion. It avoids a discretization of time in fixed intervals. It takes advantage of the temporal coherence induced by the continuity assumption. The kinetic approach to maintain a given attribute $A(t)$ for a continuously changing scene $S(t)$ is as follows: at a given time t, we create a proof of correctness of the attribute. This proof is based on elementary tests called *certificates*. For each certificate, we compute the time at which it fails and put it in a global event queue. As the attribute cannot change while all tests remain valid, it is unnecessary to perform any computation until the first certificate fails. When a certificate fails (an *event*), the discrete attribute is updated if it needs to be, and a new proof of correctness is constructed by making some modifications to the previous proof of correctness. This is known as a *kinetic data structure* (KDS). The strength of the kinetic model is that it is on-line (some objects may change their motion), and allows us to perform a rigorous combinatorial time-cost analysis in the spirit of Atallah [5], with no need to assume any bounds on the velocities. The most important aspect of this analysis is the efficiency of a KDS. For a given class of motions (in general, low-degree polynomial or pseudopolynomial functions of time), we can compute the worst-case number of changes to the discrete attribute we wish to maintain, and the worst-case processing time spent to maintain the kinetic structure. If these quantities are comparable, the kinetic structure is said to be *efficient*. (See [5], [14], and [16] for some other models addressing problems involving motion.[1])

In the context of constrained subdivisions, the kinetic approach has been successfully applied by Agarwal et al. [4] for maintaining a valid BSP of segments moving in the plane. (This work is extended to triangles moving in space in [2].) Basch et al. [6] obtain kinetic solutions to the problem of collision detection between two polygons by maintaining a "pseudo-triangulation" of their convex hull. In both cases, however, the notion of efficiency is not clear, as there is no canonical discrete attribute against which to compare the performance of the KDSs. In the context of kinetic BSPs, Agarwal et al. [4] specify a static algorithm and ensure that at every moment the BSP they maintain is precisely the same as the one that the static algorithm would have computed for the current position of the input objects. They show that for any "pseudo-algebraic motion," the number of events is quadratic in the worst case and that this bound is optimal for their model. However, this model is not completely satisfactory. For instance, in Fig. 1, some points move horizontally: the left cylindrical BSP undergoes $\Omega(n^2)$ changes while the right one does not undergo any combinatorial change. It would be better to prove efficiency of a kinetic BSP by comparing it against all possible methods that maintain a valid BSP, and not just the ones constructed by a specific algorithm.

The main difficulty in proving a lower bound on constrained subdivisions, in general, is that they are not *canonically* defined. The purpose of this paper is to show that it is

[1] Atallah [5] and Ottmann and Wood [16] study kinetic geometric problems in an off-line setting, and Kahan [14] studies some problems under the assumption that the speed of the objects is bounded.

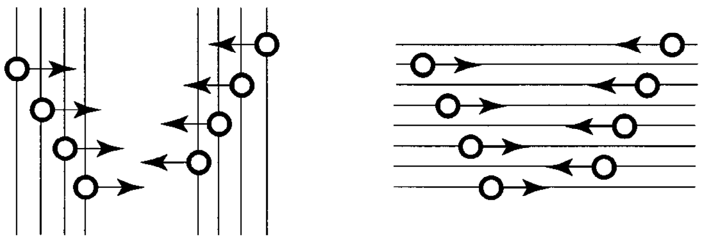


Fig. 1. For n points moving horizontally, the cost of maintaining a cylindrical BSP with vertical separating lines is roughly quadratic, whereas it is zero if the separating lines are horizontal.

nevertheless possible to prove nontrivial lower bounds on the worst-case processing cost of a whole class of constrained subdivisions. For a given class of constrained subdivisions, our method carefully constructs a set of moving points for which the kinetic maintenance of any constrained subdivision of the class will have a high processing cost. We illustrate this with two important examples: the class of all BSPs and the class of all triangulations. The models we use to capture these classes are interesting in their own right.

We first prove lower bounds on the number of changes required on a BSP of a set S of n points, each moving with fixed velocity. In Section 2 we exhibit a scenario of n moving points for which any kinetic BSP has to process $\Omega(n\sqrt{n})$ events.

We now consider triangulations of a set of moving points described at any time by $S(t)$. If we do not allow Steiner points, then whenever a new vertex appears on the convex hull boundary of $S(t)$ or an existing vertex disappears from it, any triangulation of S has to change. Since it is possible to construct a set of n points, each moving at constant velocity, so that their convex hull changes $\Omega(n^2)$ times [3], any KDS maintaining a triangulation has to update the triangulation $\Omega(n^2)$ times. This argument, however, does not apply if we allow Steiner points, that is, additional moving points that are not part of the original point set, as we can enclose the moving points in a big "bounding box" such that the convex hull never changes. It seems plausible that Steiner points can decrease the number of times a triangulation becomes invalid. See Fig. 2 for a simple example. Globally, it seems that if we create a very fine mesh of the plane around the original point set, the subdivision will behave more like a flexible piece of cloth that can adapt to many movements of the original moving points without any combinatorial change. Of course, we might need an enormous amount of Steiner points,

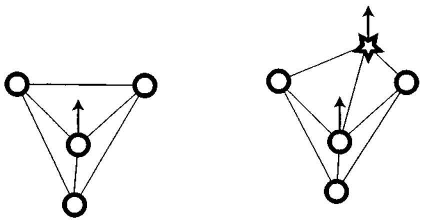


Fig. 2. A triangulation of four points is forced to change when a point appears on the convex hull. With the introduction of one moving Steiner point, we obtain a triangulation that never changes.

and the trajectory of each one might be quite intricate. Since memory limitations are often the most limiting aspect of an actual program, we focus on triangulations that use only $O(n)$ Steiner points. In Section 3 we present a scenario in which any KDS for maintaining a triangulation of n moving points with $O(n)$ moving Steiner points has to perform $\Omega(n^2)$ updates.

Our models are general enough to encompass all kinetic BSPs and kinetic Steiner triangulations. This is the first time nontrivial lower bounds are obtained that apply to a whole class of kinetic data structures at once.

2. Lower Bounds for Binary Space Partitions

A BSP $\mathcal{T}$ for a set S of segments in $\mathbb{R}^2$ is a binary tree with the following properties. A node ν of $\mathcal{T}$ is associated with an open convex polygon Δ_ν, called the *cell* of ν, and with the set of segments clipped within that cell. If ν is a leaf, no segment of S intersects its cell. If ν is an internal node, it is also associated with a *splitting line* ℓ_ν that partitions Δ_ν in two cells. The cell of the left (resp. right) child of ν is the intersection of Δ_ν with the negative (resp. positive) half-space bounded by ℓ_ν. The node ν also stores any part of a segment that lies on the line segment $\ell_\nu \cap \Delta_\nu$. If some of the segments in S are moving, at every moment t we want to maintain a *valid* BSP $\mathcal{T}(t)$ of $S(t)$.

In the lower bound we present below, the segments degenerate into points. We can also use tiny segments or any other type of tiny objects. For our purposes, we modify the model slightly so as never to store points at internal nodes: if a point belongs to the splitting line at a node, it is arbitrarily passed to the left child of the node. A leaf now contains at most one point or can be empty. Given a subset of points $S' \subset S$, we say that the deepest node of $\mathcal{T}$ whose cell contains S' is the *splitting node* of S'. This means that the splitting line of this node will be the first one along the tree not to leave all points of S' on the same side.

In this section we construct a scenario with a set S of n points moving at constant velocity such that any BSP defined over these points will have to be modified $\Omega(n\sqrt{n})$ times in order to remain correct. To obtain this result, we rely only on the general definition of a BSP given at the beginning of this section. In particular, we neither assume that the splitting lines have a finite number of possible directions, nor that they have to pass through any point of S. We also let the BSP use as much information about the motion as it likes, and let it reorganize itself arbitrarily when an event happens.

We consider only the case of disjoint objects, as intersecting objects add some static complexity that can arbitrarily raise the kinetic complexity; in our setting this means that we do not allow collisions between the points.

To prove a lower bound on the number of changes a BSP $\mathcal{T}$ for S must undergo, we proceed as follows. We show that there are k disjoint time intervals $(t_0, t_1), (t_1, t_2), \ldots, (t_{k-1}, t_k)$ such that the BSP must change during each time interval (we call this a *breaking interval*). This proves that the BSP must undergo at least k changes.

Our lower-bound construction is based on the following simple observation: suppose that we have three moving points p, q, r, and that at time t, point q passes through the segment pr. Let ν be the splitting node of $\{p, q, r\}$ at time t^- (i.e., just before time t).

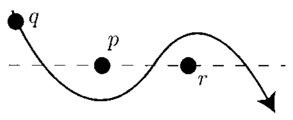


Fig. 3. A motion that forces a change to any kinetic BSP.

If ν_ℓ separates pr from q at t^-, then the kinetic BSP needs to be updated, as no line can separate pr from q at t. In this case, we say that *pr captures q*.

For instance, here is a way to create a breaking interval: by letting q weave through pr during an interval $[t_1, t_2]$ as in Fig. 3. Indeed, each of the three ways to split $\{p, q, r\}$ before time t_2 is incompatible with one of the three collinearities. If we create n pairs like pr and "shoot" n points like q that weave between each pr pair, we obtain a quadratic number of breaking intervals. However, such a scenario requires that the motions be described by polynomials of unbounded degree, which is unrealistic.

We describe below a pattern consisting of eight points moving at constant velocity. In this pattern, certain pairs capture certain points, so as to create a breaking interval. We show how we can repeat this pattern $\Theta(n\sqrt{n})$ times with only $O(n)$ moving points.

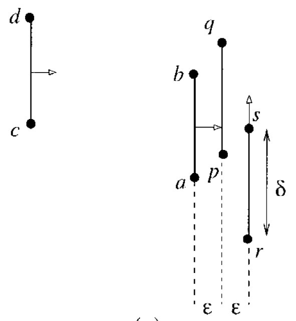
The Local Pattern. Let $\delta > 0$ be a parameter to be determined later, and $\varepsilon = \delta/8$. The individual pattern involves eight points moving in pairs. Each pair consists of two vertically aligned points at distance δ from each other.

Figure 4 shows the position of the points at time $t = 0$. The exact coordinates as a function of time are:

$$\begin{aligned} p(t) &= (0, 0), \\ a(t) &= (-\varepsilon + t, -\delta/4), \\ c(t) &= (-3\delta/2 - \varepsilon + t, \delta/4), \\ r(t) &= (\varepsilon, -3\delta/4 + t), \end{aligned} \tag{1}$$

and the other four points are placed δ above their associated point (Fig. 4(a)).

The points move as follows. The pair p, q is stationary, the pair r, s moves upward with unit speed, and the pairs a, b and c, d move to the right with unit speed. With this



(a)

point	captured by	point	captured by
p	ab, ac	a	rs, pr, qr
q	cd, bd	b	pq, qr, qs
r	cd, dq, bq	c	pq, pr, ps
s	ab	d	ps, qs, rs

(b)

Fig. 4. (a) The configuration at time $t = 0$. The four pairs are shown as segments, which do not exist as objects in the construction. (b) The table shows some of the captures that happen during the motion between time 0 and 2δ.

setup, Fig. 4(b) shows a number of captures that occur between time $t = 0$ and time $t = 2\delta$. For instance, the reader can verify that b, q, r become collinear at $t_1 \approx 0.14\delta$ and $t_2 \approx 1.73\delta$ (with our choice of $\varepsilon = \delta/8$). The first time, b is captured by qr, and the second time, r is captured by bq. There are more captures than indicated in the table, but these are the ones we exploit later.

Lemma 2.1. *The interval* $[0, 2\delta]$ *is a breaking interval for any kinetic BSP over a point set that includes the points* $\{p, q, r, s, a, b, c, d\}$.

Proof. Consider a BSP $\mathcal{T}$ at time $t = 0$. Let ν be the splitting node of our set of eight points. The split partitions $\{p, q, r, s, a, b, c, d\}$ into two nonempty subsets: a red and a blue.

We can restate our earlier observation in terms of colors: if a point is captured by two red points, then either it is red or the BSP changes at the time of capture. The same is true for blue points.

We will show that if $\mathcal{T}$ does not change between 0 and 2δ, all the points are of the same color, which is a contradiction by the choice of ν.

We assume without loss of generality that q is red. As bd captures q, either b or d is red. If b is red, then r is also red as it is captured by bq. Then qr captures a, ab captures p and s, pq captures c, and finally ps captures d. Hence all points are of the same color and we have a contradiction. Similarly, if d is red, dq captures r, qr captures b, and the rest follows as above, leading to a contradiction again. □

The Global Construction. Next we describe a set of $\Theta(n)$ moving points for which the above event sequence happens $\Theta(n\sqrt{n})$ times. Each point either moves at constant velocity or is stationary.

The idea is to repeat the local pattern on a $\sqrt{n} \times \sqrt{n}$ grid. Each cell of the grid contains a static pq pair. Each column contains an rs pair that moves up, and each row contains cd and ab pairs that move right. The grid is spaced and the initial positions of all pairs are chosen so that the local pattern is repeated in each cell at distinct times (Fig. 5). In this scenario, a kinetic BSP has to change linearly many times. As the pq pairs are static, we can repeat the same scenario later in time with $\Theta(\sqrt{n})$ fresh ab, cd, and rs pairs. With $\sqrt{n}$ repetitions of this scenario, we still use only linearly many moving points, and we have $\Theta(n\sqrt{n})$ distinct breaking intervals.

Here are the details of the construction. We assume that n is a perfect square and let $M = \sqrt{n} + 1$. We have n stationary pairs:

$$\begin{aligned} p_{ij} &= (i, jM), \\ q_{ij} &= p_{ij} + (0, \delta), \end{aligned} \qquad 0 \le i, j < \sqrt{n}.$$

The distance between consecutive pairs in a row is 1, and the distance between consecutive rows is M.

The events happen in $\sqrt{n}$ batches, each consisting of n events—one per pair i, j. Batch k involves $\sqrt{n}$ pairs r_i^k, s_i^k and $\sqrt{n}$ pairs a_j^k, b_j^k and c_j^k, d_j^k. Restarting from (1),

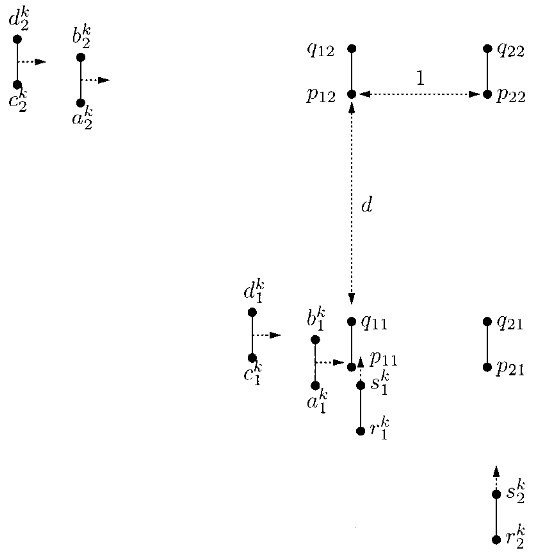


Fig. 5. A global view of the lower bound construction for the BSP.

we set

$$
\begin{aligned}
a_j^k(t) &= a(t) + (-jM - 2kn, jM),\\
c_j^k(t) &= c(t) + (-jM - 2kn, jM),\\
r_i^k(t) &= r(t) + (i, -i - 2kn),
\end{aligned}
$$

and b_j, c_j, s_j are placed δ units above their associated point.

The points $a_j^k, b_j^k, c_j^k, d_j^k$ are moving right at unit speed, and r_i^k, s_i^k are moving up at unit speed. Hence, they form the local pattern with p_{ij}, qij in the time interval $[2kn + jM + i, 2kn + jM + i + 2\delta]$. If we take $\delta < \frac{1}{2}$, we obtain $n\sqrt{n}$ disjoint breaking intervals.

Theorem 2.1. *There is a set of n points moving with constant velocities such that any BSP on that set of points undergoes $\Omega(n\sqrt{n})$ changes over the course of the motion.*

3. Lower Bounds for Triangulations

In this section we present lower bounds for the worst-case processing cost of a kinetic Steiner triangulation. Our model assumes unit cost for the insertion or deletion of an edge, or the insertion or deletion of a Steiner point. Note that an implicit representation of the triangulation could allow the change of the endpoint of linearly many edges in constant time, but we do not allow that in our model. When processing an event, a kinetic Steiner triangulation can perform as many elementary operations as it desires, but we assume that once the event is processed, the updated structure is still a triangulation. The other assumption we make about our kinetic triangulation is that it has linearly many Steiner points at any time. We do not assume anything about the total number of

Steiner points that ever arise, and each Steiner point is free to move along an arbitrary (continuous) path.

In this model we construct a scenario of $O(n)$ points in linear motion such that any linear-space kinetic Steiner triangulation requires a processing cost of $\Omega(n^2)$.

A technique of Agarwal et al. [3] (described later in this section) allows us to simulate a circular motion with linearly moving points, in the following sense: We can create a set S of points with positions $S(t)$ at time t, such that each point moves at constant velocity, and the set is always cocircular along a circle of fixed center (but with varying radius). For ease of exposition, we present the scenario with circular motion around fixed circles, and we restrict the motion to span a quarter of a circle so that the conversion can be made to linear motion.

We consider three concentric circles: the *red circle* of radius 1, the *inner blue circle* of radius $1-\varepsilon$, and the *outer blue circle* of radius $1+\varepsilon$. Here, ε is a small quantity that depends on n. We use polar coordinates to specify the position of a point, but we use a unit of $\pi/2n$ radians so that a point in the upper-right quadrant corresponds to angles between 0 and n, i.e., when we say that a point p is at position (r,α), we mean that p is at distance r from the origin O and that the vector Op makes an angle of $\alpha(\pi/2n)$ with the horizontal axis.

We have a family of $2n+1$ red points on the red circle. Their time-dependent coordinates are, in our normalized polar coordinates,

$$r_i(t) = (1, i+t), \qquad i = -n, \dots, n.$$

We also have a family of n static blue points, staggered on the inner and the outer blue circles:

$$\begin{aligned} b_{2i}(t) &= (1+\varepsilon, 2i), \\ b_{2i+1}(t) &= (1-\varepsilon, 2i+1), \end{aligned} \qquad i = 0, \dots, \lfloor n/2 - 1 \rfloor.$$

Hence, the red points move counterclockwise, and r_0 passes close to b_i exactly at time i (Fig. 6). We choose ε very small, e.g., $\varepsilon = 1/n^3$. In this case, at time $i+\frac{1}{2}$ for any integer i, the two families of points form a convex chain of alternating red and blue vertices (on the region of overlap).

We first review informally the idea of the construction. Consider a snapshot at a certain time t, and choose a path between b_0 and b_1 in the Steiner triangulation. This path crosses the red circle. Now, if the triangulation does not change between t and $t+\tau$, the path, defined combinatorially, deforms continuously. As τ red points pass the pair b_0b_1 during that time, the path is "dragged along," and should consist of many edges to

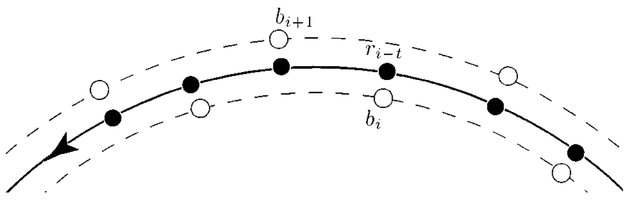


Fig. 6. The static blue points (hollow) and the moving red points (filled). In reality, the inner and outer blue circles are very close to each other, and the blue points form a convex chain.

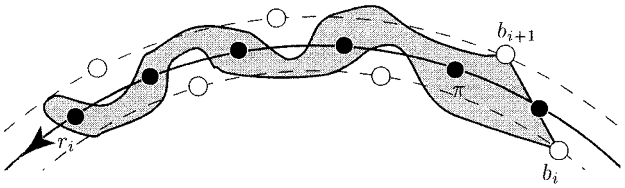


Fig. 7. As time increases, the sleeve path is deformed, but it does not sweep over any blue point as its combinatorial description is fixed.

be able to weave between the red and the blue points (Fig. 7). Conversely, the existence of a short path between b_0 and b_1 implies that the triangulation will need to be updated before too much time passes. We now need to find enough short paths to prove our lower bound.

Consider the situation at time $t_0 = \frac{1}{2}$. We have a certain current triangulation at that instant. The *right sleeve path* of a pair of consecutive blue points $b_i b_{i+1}$ is defined as follows: consider all the edges of all the triangles that intersect the segment $b_i b_{i+1}$, and keep only those that are to the right of the oriented line $b_i b_{i+1}$ and that are shared by only one triangle (see, e.g., the thick polygonal path in Fig. 8). In a triangulation, we have $n - 1$ right sleeve paths, one for each pair of consecutive blue points.

Lemma 3.1. *At time $t = \frac{1}{2}$, the right sleeve path of $b_i b_{i+1}$ crosses the red circle between r_{i-1} and r_i or between r_i and r_{i+1}.*

Proof. Let ℓ be the oriented line from b_i to b_{i+1}. As the right sleeve path connects a point inside the disk bounded by the red circle to a point outside that disk, it crosses the red circle at least once. By definition, the right sleeve path is fully contained in the half-space to the right of ℓ. It therefore crosses the red circle to the right of ℓ.

As noted earlier, the red and blue points form a convex chain at a half-integer time. Hence all these points except one (the red point r_i) are to the left of ℓ. Thus the points of the red circle that are to the right of ℓ are all between r_{i-1} and r_{i+1}. $\square$

Lemma 3.2. *At $t = \frac{1}{2}$, an edge of the triangulation belongs to at most two sleeve paths.*

Proof. An edge is adjacent to two triangles. Consider a specific edge e adjacent to a triangle Δ. If e belongs to the sleeve path of a segment $b_i b_{i+1}$ because Δ intersects

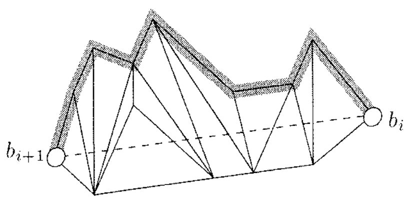


Fig. 8. The right sleeve path of $b_i b_{i+1}$.

$b_i b_{i+1}$, then the two other edges of Δ cross $b_i b_{i+1}$. Suppose these two edges of Δ intersect another segment $b_j b_{j+1}$. Since the blue vertices are in convex position, edge e lies to the left of the oriented line $b_j b_{j+1}$, and therefore does not belong to its right sleeve path. □

Lemma 3.3. *Let $\tau > 0$ be a constant and $i < n - \tau - 4$. If π is the right sleeve path of $b_i b_{i+1}$ at time $\frac{1}{2}$ and it has τ edges, then at least one edge of π disappears before time $\tau + 4$.*

Proof. Suppose the right sleeve path remains valid until time $\tau + 4$. At time $\frac{1}{2}$, the sleeve path intersects the red circle between r_{i-1} and r_i or between r_i and r_{i+1} (Lemma 3.1). At time $\tau + 4$, we have a path π' that has the same combinatorial description as π (i.e., it passes through the same vertices, although these vertices may have moved). The angular span of π' is at least $\tau + 3$ units as it passes within one unit of $r_i(\tau + 4)$. By continuity, the area delimited by π' and the segment $b_i b_{i+1}$ is homotopic to a path containing the portion of the red circle $[i, i + \tau + 3]$, and not containing any blue point. Hence, π' intersects $b_j b_{j+1}$ twice for all $j \in \{i + 2, \ldots, i + \tau + 2\}$. As the blue points form a convex chain that can be intersected at most twice by each edge of π', the path π' (and hence π) should have at least $\tau + 1$ edges, a contradiction. □

Lemma 3.4. *Let $\lambda \geq 1$ be a constant and let T be a triangulation at time $\frac{1}{2}$ with $(\lambda - 1)n$ Steiner points. Then at least $n/4 - \tau/2 - 2$ edges have to be deleted to maintain T between time 0 and time $\tau + 4$, where $\tau = 12\lambda$.*

Proof. As we have λn vertices, we have at most $3\lambda n$ edges in T. Each edge of the triangulation at time $\frac{1}{2}$ belongs to at most two right sleeve paths by Lemma 3.2. Hence there are $n/2$ right sleeve paths that have at most $\tau = 12\lambda$ edges. Lemma 3.3 applies to at least $n/2 - \tau - 4$ of those, all of which have to change before time $\tau + 4$.

As edge deletion can affect at most two right sleeve paths, at least $n/4 - \tau/2 - 2$ edge deletions have to be performed between time $\frac{1}{2}$ and time $\tau + 4$ to maintain the triangulation. □

This implies that at least $\Omega(n)$ elementary operations have been performed on the triangulation between $\frac{1}{2}$ and $\tau + 4$. As τ is a constant, we can repeat this argument $\lfloor n/(\tau + 4) \rfloor$ times, at times $\{k(\tau + 4) + \frac{1}{2} \mid k = 0 \cdots \lfloor n/(\tau + 4) - 1 \rfloor\}$, restarting each time from a new snapshot of the triangulation.

Finally, we show how we can modify this construction to obtain the same effect with points moving at constant velocity. To this end, we use the linearization technique of Agarwal et al. [3]. If we let $\bar{c}(\theta) = (\cos\theta, \sin\theta)$, a point p moving along a chord of a circle of radius ρ from angle α to angle $\alpha + \varphi$ at constant velocity has coordinates

$$p(t) = \rho(1 - t)\bar{c}(\alpha) + \rho t \bar{c}(\alpha + \varphi) .$$

By rotational invariance and symmetry, it follows that the distance of $p(t)$ to the origin does not depend on α nor on the sign of φ. Hence, if a number of moving points move along chords of equal angular width (in absolute value), they will remain cocircular during the whole motion.

In our case, we can take our red points to move counterclockwise and our blue points to move clockwise, along chords of equal angular width, but with starting and ending points on circles of slightly different radii as described earlier. In conclusion:

Theorem 3.1. *There exists a set of n points moving at constant velocity so that any kinetic Steiner triangulation of linear size requires $\Omega(n^2)$ elementary changes over the course of the motion.*

Remark 3.1. The above argument can be generalized to Steiner triangulations with more vertices: if we allow $O(nf(n))$ vertices at any one time, then we can find linearly many sleeve paths of size $O(f(n))$. A path of this size will break after $O(f(n))$ time, and each broken edge belongs to at most two paths, so the number of changes is at least $\Omega(n^2/f(n))$.

4. Conclusion

In this paper we have presented kinetic lower bounds for the number of changes of two noncanonically defined combinatorial structures: BSPs and triangulations. Each lower bound presents room for improvements and variations.

In the case of the BSP, we would like to obtain a local pattern that can be repeated quadratically many times with linearly many points, so as to bridge the gap between our current lower bound and what we believe is the correct answer. It might be the case that our model gives too much power to the adversary (who is allowed to reconstruct the BSP completely at each event) and that the right bound for this computational model is not quadratic.

In the case of Steiner triangulations, our proof relies on the fact that we add only linearly many Steiner points. Indeed, does the lower bound hold if an arbitrary number of Steiner points is allowed? The answer to this question might turn out to be negative in our model, but not necessarily in a model where we request that each Steiner point has a motion of constant description complexity. What if curved boundaries are allowed in the triangulation (but of bounded algebraic degree)?

It is also worth noticing that the models for the BSP and the triangulation are of a very different nature. The BSP can be completely rewritten when an event occurs, whereas the triangulation is charged for each insertion or deletion of an edge or vertex, which is more realistic. Moreover, our BSP model encodes only the hierarchical partition of the point set, and not any geometric information about the cells defined by the partition lines. If the geometry of a cell changes, an event is processed in existing methods [2], [4] but not in our model for lower bounds. We were not able to define a more constrained model of computation for the BSP that could give better bounds. Moreover, although there is clearly a relation between BSPs and Steiner triangulations (a BSP is a subdivision of the plane defined with some vertices in addition to the original data, even though the cells might not be triangular), we were not able to use our proof technique for the Steiner triangulation and adapt it to obtain a quadratic bound for the BSP.

Our lower bounds do not say anything about what happens for “natural” scenarios. Clearly natural point motions are not random and exhibit various degrees of coherence.

Is there an intuitive measure of how coherent the motion of a point set is? Can we get coherence-sensitive algorithms for this problem? Given, say, the knowledge that a kinetic BSP with few changes exists, is it possible to find it or get an approximation?

References

1. P. K. Agarwal and J. Erickson, Geometric range searching and its relatives. In *Advances in Discrete and Computational Geometry* (B. Chazelle, J. E. Goodman and R. Pollack, eds.), AMS Press, Providence, RI, 1998, pp. 1–56.
2. P. K. Agarwal, J. Erickson, and L. Guibas. Kinetic binary space partitions for triangles. In *Proc*. 9*th ACM–SIAM Sympos*. *Discrete Algorithms*, 1998, pp. 107–116.
3. P. K. Agarwal, L. J. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving set of points. In *Proc*. 5*th Workshop Algorithms Data Structures*, 1997, pp. 31–44.
4. P. K. Agarwal, L. J. Guibas, T. M. Murali, and J. S. Vitter. Cylindrical static and kinetic binary space partitions. *Comput*. *Geom*. *Theory Appl*., **16** (2000), 103–127.
5. M. J. Atallah. Some dynamic computational geometry problems. *Comput*. *Math*. *Appl*., **11** (1985), 1171–1181.
6. J. Basch, J. Erickson, L. J. Guibas, J. Hershberger, and L. Zhang. Kinetic collision detection for two simple polygons. In *Proc*. 10*th Sympos*. *Discrete Algorithms*, 1999, pp. 102–111.
7. J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Algorithms*, **31** (1999), 1–28
8. J. L. Bentley. Multidimensional binary search trees used for associative searching. *Comm*. *ACM*, **18** (1975), 509–517.
9. M. Bern. Triangulations. In *Handbook of Discrete and Computational Geometry* (J. E. Goodman and J. O'Rourke, eds.), CRC Press LLC, Boca Raton, FL, 1997, pp. 413–428.
10. Y. Chrysanthou. Shadow Computation for 3D Interaction and Animation. Ph.D. thesis, Queen Mary and Westfield College, University of London, 1996.
11. J. Donea. *Computational Methods for Transient Analysis*, volume 1. North-Holland, Amsterdam, 1983, chapter 10: Arbitrary Eulerian–Lagrangian methods.
12. R. A. Finkel and J. L. Bentley. Quad trees: a data structure for retrieval on composite keys. *Acta Inform*., **4** (1974), 1–9.
13. S. Ghosh and S. Raju. R-S adapted Arbitrary Lagrangian–Eulerian finite element method for metal-forming problems with strain localization. *Internat*. *J*. *Numer. Methods Engrg*., **39** (1996), 3247–3272.
14. S. Kahan. A model for data in motion. In *Proc*. 23*th Annu*. *ACM Sympos*. *Theory Comput*., 1991, pp. 267–277.
15. B. F. Naylor. Interactive solid geometry via partitioning trees. In *Proc*. *Graphics Interface* '92, 1992, pp. 11–18.
16. T. Ottmann and D. Wood. Dynamical sets of points. *Comput*. *Vision Graph*. *Image Process*., **27** (1984), 157–166.
17. H. Samet. *The Design and Analyses of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1989.
18. R. A. Schumacker, R. Brand, M. Gilliland, and W. Sharp. Study for Applying Computer-Generated Images to Visual Simulation. Technical Report AFHRL–TR–69–14, U.S. Air Force Human Resources Laboratory, 1969.
19. E. Torres. Optimization of the binary space partition algorithm (BSP) for the visualization of dynamic scenes. In *Proc*. *Eurographics* '90, 1990, pp. 507–518.
20. H. T. Y. Yang, M. Heinstein, and J.-M. Shih. Adaptive 2d finite element simulation of metal forming processes. *Internat*. *J*. *Numer. Methods Engrg*., **28** (1989), 1409–1428.

Received June 10, 1999, *and in revised form March* 9, 2000. *Online publication August* 21, 2000.