Lower Bounds For Kinetic Planar Subdivisions*

Pankaj K. Agarwal† Julien Basch‡ Mark de Berg§ Leonidas J. Guibas¶
John Hershberger‖

Abstract

We revisit the notion of kinetic efficiency for non-canonically-defined discrete attributes of moving data, like binary space partitions and triangulations. Under very general 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.

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 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 [9], and binary space partitions (BSP) [17]. Motivated by various applications, constrained subdivisions have been extensively studied in computational geometry and related application areas [10, 16].

In a growing number of applications, we do not have one fixed 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

*Part of this work was done while P.A. and M.d.B. were visiting Stanford University. Work by P.A. is 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. is supported by Army Research Office MURI grant DAAH04-96-1-0013. Work by L.G. is supported by National Science Foundation grant CCR-9623851 and by U.S. Army Research Office MURI grant DAAH04-96-1-0007.

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

‡Center for Geometric Computing, Department of Computer Science, Duke University, Box 90129, Durham, NC 27708-0129; jbasch@cs.duke.edu

§Department of Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, the Netherlands; markdb@cs.uu.nl.

¶Computer Science Department, Stanford University, Stanford, CA 94305; guibas@cs.stanford.edu.

‖Mentor Graphics Corp., 8005 SW Boeckman Road, Wilsonville, OR 97070-7777, USA; john_hershberger@mentorg.com.

the objects' features) will change only at discrete time instances. The maintenance of a subdivision should therefore proceed in discrete steps.

Relatively little attention has been focused so far on maintaining a constrained subdivision. In the case of BSPs, which are widely used in graphics where motion is pervasive, most existing work is based on using dynamic BSPs. These approaches discretize time into short intervals of fixed duration. 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, [11, 14, 18]. Such approaches suffer from the fundamental problem that it is difficult to know how to choose the length of the discretization interval. If the duration of an interval is too short, then the subdivision does not change combinatorially, and the deletion/re-insertion is just wasted computation; if the duration is too large, important intermediate events can be missed, with ill effects on applications using the subdivision. Such approaches cannot handle many moving objects, and have to adapt the interval duration to the fastest object.

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.* [8] is a general method to maintain a discrete attribute of objects in motion. It avoids a discretization of time in fixed intervals. It takes full 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. For each test, 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 kinetic data structure. For a given class of motions (in general, low-degree polynomial or pseudo-polynomial 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, 13, 15] for some other models for addressing problems involving motion.[1])

In the context of constrained subdivisions, the kinetic approach has been successfully applied by Agarwal *et al.* [3] 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.* [7] 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 kinetic data structures. In the context of kinetic BSPs, Agarwal *et al.* [3] 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 satisfactory because, as shown in Figure 1, for a set of points in the plane, the first BSP undergoes $\Omega(n^2)$ changes while the second one does not undergo any combinatorial change. It would be better to prove efficiency of a kinetic BSP by comparing it against all possible BSPs, 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 nevertheless possible to prove nontrivial lower bounds on the worst-case processing cost of a whole class of

[1] Atallah [5] and Ottmann and Wood [15] study kinetic geometric problems in an off-line setting, and Kahan [13] studies some problems under the assumption that the speed of the objects is bounded.

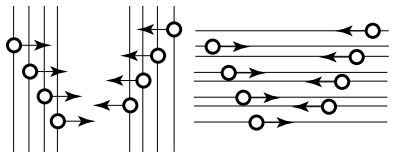


Figure 1: For n points moving horizontally, the cost of maintaining a cylindrical BSP with vertical separating lines is roughly quadratic, whereas it is zero if the separating lines are horizontal.

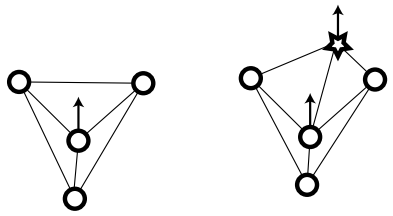


Figure 2: A triangulation of four points is forced to change when a point appears on the convex hull. With the introduction of one moving Steiner point, we obtain a triangulation that never changes.

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 or a triangulation 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.

Let us now consider the case of triangulations for 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 the hull, 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 [4], any KDS maintaining a triangulation has to update the triangulation $\Omega(n^2)$ times. The question of effectively finding a kinetic triangulation that achieves these bounds is addressed in [6]. 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. It seems plausible that Steiner points can decrease the number of times a triangulation becomes invalid. See Figure 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, 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 non-trivial 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 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 ℓ_ν and is contained in Δ_ν. 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 could also have used tiny segments, or any other type of tiny objects. For our purposes, we modify the model slightly so as to never store points at internal nodes: if a point belongs to the splitting line at a node, it is arbitrarily given 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 re-organize 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),..., (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). 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*. We describe below a pattern consisting of eight moving points with different captures that create a breaking interval, and show how we can repeat this pattern $\Theta(n\sqrt{n})$ times with only $O(n)$ moving points.

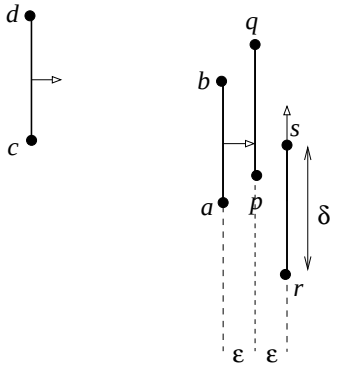
The local pattern. The individual pattern involves eight points moving in pairs. Each pair consists of two vertically aligned points at distance δ from each other at any time. Here $\delta > 0$ is a parameter to be determined later. Figure 3 shows the position of the points at time $t = 0$. The exact coordinates are:

$$\begin{array}{lcllcl} p & = & (0,0), & a & = & (-\varepsilon, -\delta/4), \\ r & = & (\varepsilon, -3\delta/4), & c & = & (-3\delta/2 - \varepsilon, \delta/4), \end{array}$$

and the other four points are placed δ above their associated point (Figure 3a). Here $\varepsilon > 0$ is a sufficiently small parameter, for instance, we can take $\varepsilon = \delta/8$.

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 setup, Figure 3b 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 will 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 $\{p, q, r, s, a, b, c, d\}$.*



(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)

Figure 3: (a) The configuration at time $t = 0$. The four pairs are shown as segments, which don't exist as objects in the construction. (b) The table shows some of the captures that happen during the motion.

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 non-empty subsets. We call the subset containing the point q the *red subset*; the other subset is the *blue subset*.

We can restate our earlier observation in color: if a point is captured by two red points, then it has to be red or the BSP needs to change at the time of capture. A similar statement is true for blue captures.

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. Recall that we defined the subset containing q to be 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 at last 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 in the paragraph above. Hence both b and d are blue, which is impossible as bd captures q. □

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 (Figure 4). 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.

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.*

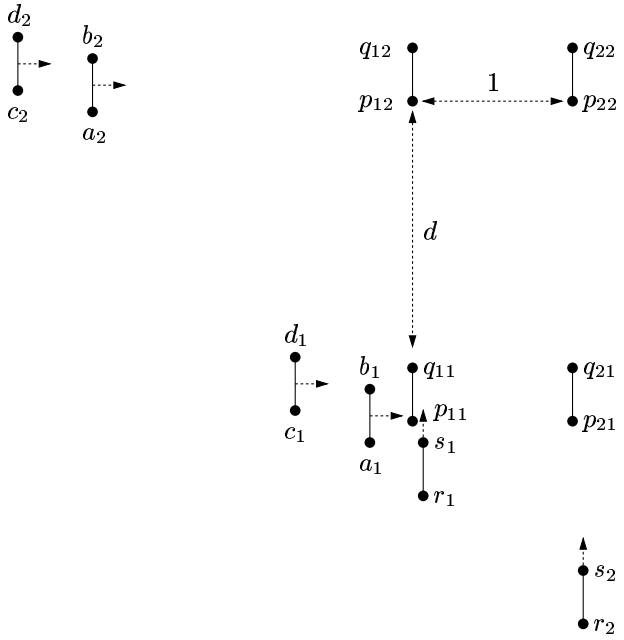


Figure 4: A global view of the lower bound construction for the BSP.

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. When processing an event, a kinetic Steiner triangulation can perform as many of these 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 don't 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.* [4] allows us to simulate a circular motion with linearly moving points, in the following sense: We can create a set S of points with position $S(t)$ 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 $\frac{\pi}{2n}$ radians so that a point in the upper-right quadrant corresponds to angles between 0 and n. This being said, we have a family of $2n+1$ red points on the red circle. Their time-dependent coordinates are, in our normalized polar coordinates:

$$r_i(t) = (1, i+t) \quad i = -n, \ldots, n\,.$$

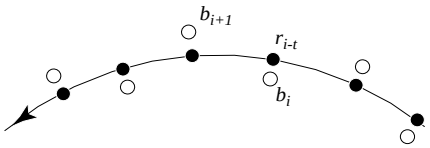


Figure 5: The static blue points (hollow) and the moving red points (filled). In reality, the inner and outer blue circles are very close to each other, and the blue points form a convex chain.

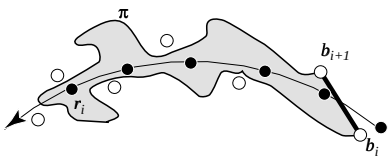


Figure 6: As time increases, the sleeve path is deformed, but it does not sweep over any blue point as its combinatorial description is fixed.

We also have a static family of n blue points[2], staggered on the inner and the outer blue circles:

$$b_i(t) = (1 + \varepsilon(1 - 2(i \bmod 2)), i) \quad i = 0, \ldots, n-1\,.$$

Hence, the red points move counter-clockwise, and r_0 passes close to b_i exactly at time i (Figure 5). We choose ε small enough so that when the red points are between two blue points (at half-integer times), the two sets form a convex chain of alternating red and blue vertices.

Consider a snapshot at a certain time t, and imagine that there is a path in the Steiner triangulation between b_0 and b_1. This path crosses the red circle. Now, if the triangulation does not change between t and $t + C$, the path, defined combinatorially, deforms continuously. As C red points pass the pair $b_0 b_1$ during that time, the path is "dragged along," and should consist of many edges to be able to weave between the red and the blue points (Figure 6). 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 $t_0 = 1/2$. We have a certain current triangulation at that instant. The *right sleeve path* between two 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 line $b_i b_{i+1}$ and shared by only one triangle (Figure 7). In a triangulation, we have $n - 1$ sleeve paths between consecutive blue points.

Lemma 3.1 *At time $t = 1/2$, the area between the segment $b_i b_{i+1}$ and its associated right sleeve path contains no blue point, and at most one red point, namely r_i.*

[2]To simulate this scenario with linear motion, the blue points will also move.

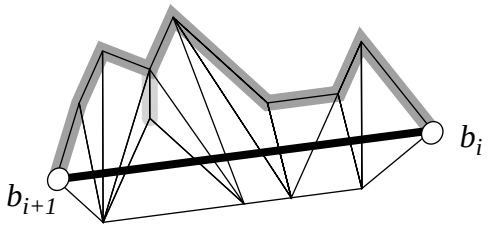


Figure 7: A right sleeve path between two points in a triangulation.

Proof: As noted earlier, the red and blue vertices form a convex chain at a half-integer time. All the edges of the sleeve path are to the right of the line supporting $b_i b_{i+1}$, while all points except one red (r_i) are to the left of that line. □

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

Proof: An edge borders two triangles. Consider a specific edge e bordering a triangle T. If e belongs to the sleeve path of a segment $b_i b_{i+1}$ because of T, then the two other edges of T cross $b_i b_{i+1}$. If they cross another segment $b_j b_{j+1}$, then, by convexity, the edge e is to the left of $b_j b_{j+1}$, and hence not in its right sleeve path. □

Lemma 3.3 *Let C be fixed and $i < n - C - 4$. If π is a sleeve path between b_i and b_{i+1} at time $1/2$ and it has C edges, then at least one edge of π disappears before time $C + 4$.*

Proof: Suppose the sleeve path remains valid until time $C + 4$. At time $1/2$, 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 $C + 4$, we have a path π' that has the same combinatorial description as π. The angular span of π' is at least $C + 3$ units as it passes within one unit of $r_i(C + 4)$. 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 $[i, i + C + 3]$, and not containing any blue point. Hence, π' intersects $b_j b_{j+1}$ twice for all $j \in \{i+2, \ldots, i+C+2\}$. 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 $C + 1$ edges, a contradiction. □

As we have $O(n)$ vertices (including the Steiner points), there is a constant λ such that we have at most λn edges in the triangulation. Each edge of the triangulation at time $1/2$ belongs to at most two sleeve paths, by Lemma 3.2. It follows that of the $n - C$ sleeve paths connecting b_i, b_{i+1}, for $0 \leq i < n - C$, at least half have at most 4λ edges. (Here we are ignoring a minor multiplicative factor of $1 + O(C/n)$.) For each of these paths, an edge has to disappear between time $1/2$ and time $C = 2\lambda + 4$. Each edge belongs to at most two sleeve paths, and so the total number of edge disappearances is at least half the number of paths.

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

It remains to see how we can modify this construction so that the points move at constant velocity. To this end, we use the linearization technique of Agarwal *et al.* [4]. Consider a point p moving at constant velocity along a chord of a circle of radius ρ. Let α be the angle of the starting point of the chord, and let $\alpha + \phi$ be the angle of its endpoint. If we let $\bar{c}(\theta) = (\cos\theta, \sin\theta)$, then the position of p as a function of time is given by:

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

This can be expressed in polar coordinates $(r_p(t), \theta_p(t))$. If we define $\psi(t) = \tan^{-1}((2t - 1)\tan\frac{\phi}{2})$, which varies in the range $[-\phi/2, \phi/2]$ as t varies in $[0, 1]$, we have:

$$\begin{aligned} r_p(t) &= \rho\frac{\cos\frac{\phi}{2}}{\cos\psi(t)} \\ \theta_p(t) &= \alpha + \frac{\phi}{2} + \psi(t)\,. \end{aligned}$$

The radius $r_p(t)$ doesn't depend on α. Hence, if a number of moving points move along chords of equal angular width ϕ, they will remain cocircular during the whole motion.

In our case, we can take our $2n$ points to move counter-clockwise and our blue points to move clockwise, along chords of equal angular width, but with starting and ending points on slightly offset circles as described earlier. In conclusion:

Theorem 3.1 *There exists a scenario 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.*

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 presented kinetic lower bounds for the number of changes of two non-canonically 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. At this point, we can only describe a scenario that achieves this bound, but some points have to accelerate and decelerate n times along their straight trajectory (any triangle still changes orientation only once). It might be the case that our model gives too much power to the adversary 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. And what if curved boundaries are allowed in the triangulation (but of bounded algebraic degree)?

Our lower bounds do not say anything about what happens for "natural" scenarios. Clearly natural point motions are not random and exhibit various degree 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, 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. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 107–116, Jan. 1998.

[3] P. K. Agarwal, E. F. Grove, T. M. Murali, and J. S. Vitter. Binary space partitions for fat rectangles. In *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 482–491, Oct. 1996.

[4] P. K. Agarwal, L. J. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving set of points. In *5th Workshop on Algorithms & Data Structures*, page 31–44, 1997.

[5] M. J. Atallah. Some dynamic computational geometry problems. *Comput. Math. Appl.*, 11:1171–1181, 1985.

[6] J. Basch, M. de Berg, O. Cheong, L. Guibas, and J. Hershberger. Kinetic maintenance of triangulations and pseudo-triangulations. In preparation.

[7] J. Basch, J. Erickson, L. J. Guibas, J. Hershberger, and L. Zhang. Kinetic collision detection for two simple polygons. In *Proc. 10th Sympos. Discrete Algorithms*, pages 102–111, 1999.

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

[9] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[10] M. Bern. Triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 413–428. CRC Press LLC, Boca Raton, FL, 1997.

[11] Y. Chrysanthou. *Shadow Computation for 3D Interaction and Animation.* Ph.D. thesis, Queen Mary and Westfield College, University of London, 1996.

[12] R. A. Finkel and J. L. Bentley. Quad trees: a data structure for retrieval on composite keys. *Acta Inform.*, 4:1–9, 1974.

[13] S. Kahan. A model for data in motion. In *Proc. 23th Annu. ACM Sympos. Theory Comput.*, pages 267–277, 1991.

[14] B. F. Naylor. Interactive solid geometry via partitioning trees. In *Proc. Graphics Interface '92*, pages 11–18, 1992.

[15] T. Ottmann and D. Wood. Dynamical sets of points. *Comput. Vision Graph. Image Process.*, 27:157–166, 1984.

[16] H. Samet. *The Design and Analyses of Spatial Data Structures.* Addison Wesley, MA, 1989.

[17] 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.

[18] E. Torres. Optimization of the binary space partition algorithm (BSP) for the visualization of dynamic scenes. In *Eurographics '90*, pages 507–518. North-Holland, 1990.