Algorithms for Dynamic Computational Geometry with Applications

Abstract

Most of the literature of computational geometry concerns geometric properties of sets of static points. M.J. Atallah introduced "dynamic computational geometry," concerned with both momentary and long-term geometric properties of sets of moving point-objects. This area of research seems to have been dormant recently. The current paper examines new problems in dynamic computational geometry: the "Too Close," "Too Far," and "3 Aligned" problems. Worst-case optimal solutions are given for these problems using the sequential and coarse-grained multicomputer (CGM) models of computation.

Key words and phrases: dynamic computational geometry, piecewise defined continuous function, analysis of algorithms, coarse-grained multicomputer

1 Introduction

Computational geometry is generally concerned with geometric properties of some object that is usually a set S of static points; less often, a set of line segments, curves, or surfaces. Examples of such properties include finding

- the nearest pair of distinct members of S;
- the diameter of S;
- the convex hull of S;
- maximal subsets of (exactly or approximately) collinear members of S
- subsets of S that are (exactly or approximately) congruent to, or similar to, a given set of points,

et al. Readers needing more background in computational geometry may find useful such sources as [12, 2, 9].

M.J. Atallah's paper [1] introduced "dynamic computational geometry," presenting efficient sequential algorithms for a variety of geometric problems concerned with systems of point–objects that are in simultaneous motion in Euclidean space $\mathbb{R}^d$; it was assumed for each object that its motion was described in each coordinate by known polynomials of time. Parallel versions of these algorithms were studied in [3, 4, 7, 8]. Another paper concerned with geometric properties of moving objects is [13]. The term "dynamic" is also applied to the study of a system of objects in which, over time, objects are inserted or deleted from the system, but this is not what is studied in the current paper.

In the current paper, we introduce efficient sequential and coarse grained parallel algorithms for additional problems in dynamic computational geometry (as introduced in Atallah's paper [1]), and discuss applications. Problems considered are:

- The "Too Close" problem: When, if ever, is the earliest time that one of our point objects, q_1, is too close to any other of our point objects, q_j? For moving objects, this seems to have safety implications.

- The "Too Far" problem: When, if ever, is the earliest time that one of our point objects, q_1, is too far from any other of our point objects, q_j? This seems to have implications for the ability to communicate or the ability to attack.

- The "3 Aligned" problem: When, if ever, is the earliest time that one of our point objects, q_1, is (approximately or exactly) collinear with any two others, q_i and q_j. This seems to have implications for the ability to communicate or the ability to attack.

2 Preliminaries

2.1 Motion assumptions

In the current paper, we consider a system $Q = \{q_i\}_{i=1}^n$ of moving point-objects in $\mathbb{R}^d$. The motion of the point-object q_i is described by a function $f_i : [0, M] \to \mathbb{R}^d$ such that if $p_j : \mathbb{R}^d \to \mathbb{R}$ is the projection to the j^{th} coordinate,

$$p_j(x_1, \ldots, x_n) = x_j,$$

then each function $p_j \circ f_i(t)$ is a polynomial in t. We further assume, as in [1], that polynomial equations of bounded degree can be solved in $\Theta(1)$ time. One can imagine applications, e.g., in the safety of airplanes circling an airport or satellites circling the globe.

2.2 On more general motion

The assumption of motion $f : [0, M] \to \mathbb{R}^d$ such that for each coordinate j, $p_j \circ f : [0, M] \to \mathbb{R}$ is a polynomial in the time variable t, may seem rather artificial. However, polynomials are dense in the set of continuous functions from $[0, M]$ to $\mathbb{R}$, i.e., given a continuous function $g : [0, M] \to \mathbb{R}$ and $\varepsilon > 0$, there exists a polynomial $G : [0, M] \to \mathbb{R}$ such that for all $t \in [0, M]$, $|g(t) - G(t)| < \varepsilon$. Further, depending on the function g, it is often possible to compute a corresponding function G efficiently, e.g., by using Taylor series, polynomial interpolation, et al.

Consider, for example, elliptic motion at constant angular velocity in the Euclidean plane. We describe a point in $\mathbb{R}^2$ as a pair (x, y) of real numbers. We consider motion described by

$$\begin{aligned} x &= R_1 \cos(at + \theta_0) + x_0 \\ y &= R_2 \sin(at + \theta_0) + y_0 \end{aligned} \tag{1}$$

for appropriate constants $R_1, R_2, a, \theta_0, x_0, y_0$, where t is the time variable. Note that both coordinates have the same period, and the x and y coordinates describe a planar ellipse (a circle if $R_1 = R_2$), with constant angular velocity.

Even if a library of functions containing the sin or cos is not available, the sin and cos functions can each be approximated by the sums of the first several terms of their respective appropriately centered Taylor series. Thus, motion as described in section (1) is included in the set of motions described in section 2.1. Similarly, a wide range of motion functions not described in the form of section 2.1 can be represented in this form with satisfactory approximation.

2.3 Piecewise defined functions

Definition 2.1. [1] Let $s \in \mathbb{N}$. Let $h : [0, M] \to \mathbb{R}$. Let $f_1, f_2, \ldots, f_s : [0, M] \to \mathbb{R}$ be distinct continuous functions. Suppose for all $t \in [0, M]$ there exists an index k and an interval $J = [u, v] \subset [0, M]$ such that $t \in J$ and $h|_J = f_k|_J$. A piece of $h : [0, M] \to \mathbb{R}$ *based on* $f_1, f_2, \ldots, f_s$ is a pair $(f_i, [u, v])$ such that $[u, v]$ is a maximal subinterval of $[0, M]$ for which $h|_{[u,v]} = f_i|_{[u,v]}$ identically.

Definition 2.2. Let $s \in \mathbb{N}$. Let $f_1, f_2, \ldots, f_s : [0, M] \to \mathbb{R}$ be distinct continuous functions. Let $h : [0, M] \to \mathbb{R}$ be a function defined by pieces *based on* $f_1, f_2, \ldots, f_s : [0, M] \to \mathbb{R}$. We say h *switches pieces at* $t_0 \in [0, M]$ if there are pieces of h, $(f_i, [u, t_0])$ and $(f_j, [t_0, v])$ such that $i \neq j$, i.e., t_0 is a common endpoint of the intervals of two pieces of h that have distinct functions. Such a point t_0 is a *switchpoint* of $\{f_i\}_{i=1}^s$ on $[0, M]$.

We say the set of pieces of the function h is its *description from* or *based on* $\{f_1, \ldots, f_s\}$, and this set of pieces *describes* h

Remark 2.3. *Note that by Definition 2.2, a switchpoint must be an interior point of* $[0, M]$.

Lemma 2.4. *A continuous function $f : [0, M] \to \mathbb{R}$ has s switchpoints based on $\{f_i\}_{i=1}^s$ if and only if f has $s+1$ pieces based on $\{f_i\}_{i=1}^s$ on $[0, M]$.*

Proof. This follows easily from Remark 2.3. Details are left to the reader. □

2.4 Coarse grained multicomputer

Material in this section is largely quoted or paraphrased from [6].

The *coarse grained multicomputer* (CGM) was described in [10] as a model of parallel computation capable of processing a great deal of data with relatively few processors. A $CGM(n, p)$ has p processors operating on $\Theta(n)$ data. Every processor has $\Omega(n/p)$ memory cells, each of $\Theta(\log n)$ bits. "Coarse grained" means the size $\Omega(n/p)$ of local memory is "considerable larger" than $\Theta(1)$; customarily, this is taken to mean $n/p \geq p$, i.e., $n \geq p^2$. We use this conventionin the current paper. Processors may share memory or may be arranged in some interconnection network.

Processors are indexed from 1 to p, and each processor "knows" its index.

We regard a $CGM(n, p)$ as a connected graph G in which vertices are processors. If the processors are in an interconnection network then edges are communication links between directly connected processors. If the processors share memory then we regard G as a complete graph, i.e., every pair of processors is regarded as joined by a communication link.

Processors were assumed in [10] to communicate data among themselves through sorting operations, but we will use the assumption of later papers that any pair of adjacent processors may exchange a unit of data in $\Theta(1)$ time. Similarly, in a shared memory system, we assume any pair of processors may exchange a unit of data in $\Theta(1)$ time.

We will use the following.

Theorem 2.5. [5] *A unit of data can be broadcast in a $CGM(n, p)$ from its source processor to all processors in $O(p)$ time.*

A set of values S distributed among the processors of a parallel computer G is said to be *gathered* to a processor P of G when all the values of S are copied to P.

Theorem 2.6. [5] *Let S be a set of N elementary data items distributed among the processors of a $CGM(n, p)$ G such that $N = \Omega(p)$ and $N = O(n/p)$. S can be gathered to any processor of G in optimal $\Theta(N)$ time.*

Theorem 2.7. [5] *Let $X = \{x_i\}_{i=1}^n$ be a set of elementary comparable data items distributed one per processor in a $CGM(n, p)$. The value of a semigroup operation* (e.g., $\min\{x \in X\}$) *can be computed and broadcast to all processors in optimal $\Theta(n/p)$ time.*

3 "Too close" problem

Problems studied in [1, 3, 4, 7] that may be interpreted as concerned with safety:

- The Nearest Neighbor Problem: which pair of objects are nearest, as a function of time?
- The Collision Problem: when (if ever) will some pair of objects collide?

Another important safety problem, which we call the "Too Close Problem": when is the earliest, if ever, the object q_1 is too close to one or more other objects q_j?

If d is a metric in which each point-object P_i has a distance g_i such that it is dangerous for

$$d(q_1, q_j) \leq \min\{g_1, g_j \mid j \neq 1\} \quad (2)$$

then we are particularly interested in predicting the time of the first occurrence of (2); perhaps the trajectories of one or both of the objects may be changed to avoid such an occurrence.

If for any index $j \neq 1$, inequality (2) holds at $t = 0$, then $t = 0$ is the earliest time of a "too close" occurrence. Otherwise, at $t = 0$, we have $d(P_1, P_j) > \min\{q_1, q_j\}$ for all $j > 1$, and since the trajectory functions f_i are all continuous, any instances of inequality (2) occur earliest when $d(P_1, P_j) = \min\{q_1, q_j\}$.

3.1 Sequential solution

Algorithm 3.1. *Let Q be a set of n point-objects as in section 2.1. Assume for each i we know f_i, the function that describes the motion of q_i; and g_i, the minimum safe distance between q_i and any other q_j. We solve the Too Close Problem for q_1 as follows, where the return values of the algorithm are in the variables minTime and earliestIndices.*

1. *earliestIndices* $:= \emptyset$ *and minTime* $:= \infty$.
2. *For* $j = 2$ *to* n
   (a) *Let* $d_j = \min\{g_1, g_j\}$
   (b) *Compute the vector difference function*
   $$S_j(t) := f_j(t) - f_1(t).$$
   /* Note $|S_j(t)|$ is the distance between q_j and q_1 at time t. */
   (c) *If* $|S_j(0)| \leq d_j$ *then*
   minTime := 0
   earliestIndices $:=$ *earliestIndices* $\cup \{j\}$
   End If $|S_j(0)| \leq d_j$
   Else If equation $S_j(t) = d_j$ *has a smallest solution* $t_j \in [0, M]$ *then*
   If $t_j = minTime$ *then*
   /* *we have a tie for earliest solution found so far* */
   earliestIndices $:=$ *earliestIndices* $\cup \{j\}$
   Else If $t_j < minTime$ *then*

/ we have a new earliest-so-far solution */*
$minTime := t_j$
$earliestIndices := \{j\}$
End Else If $t_j < minTime$

End Else If smallest solution t_j

End For

Theorem 3.2. *Algorithm 3.1 executes sequentially in optimal* $\Theta(n)$ *time.*

Proof. Clearly, step 1 executes in $\Theta(1)$ time.

Step 2 consists of a loop whose body executes $\Theta(n)$ times. It is elementary that each step of the loop body executes in $\Theta(1)$ time. Therefore step 2 requires $\Theta(n)$ time.

Thus, the algorithm has a running time of $\Theta(n)$. This is worst-case optimal, since in the worst case, each member of Q must be considered. □

Remark 3.3. *Note if all we want is the time of the first instance of too-closeness, and not the index set of the too-close objects, we can change the For loop to a loop governed by "While* $minTime > 0$ *and* $j \le n$*". In this case we have a best-case running time of* $\Theta(1)$*, realized in the case* $|S_2(0)| \le d_2$*.*

3.2 CGM solution

We give a CGM solution that is only concerned with finding the earliest instance, if one exists, of two members of Q being too close.

Algorithm 3.4. *Let* Q *be a set of* n *point-objects as in section 2.1. Assume each of the* p *processors has descriptions of* $\Theta(n/p)$ *of the pairs* (f_i, g_i)*. Without loss of generality, processor* P_j *has descriptions of the records in the set* $\left\{(f_i, g_i)\}_{i=(j-1)n/p+1}^{jn/p}\right\}$*. We solve the Too Close Problem for* q_1 *as follows, where the return values of the algorithm are in the variables* $minTime$ *and* $earliestIndices$*.*

1. *Broadcast the pair* (f_1, g_1) *from processor* P_1 *to all processors.*
2. *Use Algorithm 3.1 so that each processor* P_i *solves its portion of the problem, finding the earliest time* T_i *that any of* $\{q_i\}_{i=(j-1)n/p+1}^{jn/p}\}$ $(i \ne 1)$*, if any, is too close to* q_1*.*
3. *Gather* $\{T_i\}_{i=1}^{p}$ *to* P_1*.*
4. *Processor* P_1 *computes* $m = \min\{T_i\}_{i=1}^{p}$ *in* $\Theta(p)$ *time and notes the set* Y *of indices at which the minimum occurs. I.e.,* $j \in Y$ *if and only if* $T_j = m$*.*
5. *If desired, broadcast* m *from* P_1 *to all processors.*

Theorem 3.5. *Algorithm 3.4 runs in optimal* $\Theta(n/p)$ *time.*

Proof. 1. By Theorem 2.5, the broadcast requires $O(p)$ time.

2. By Theorem 3.1, the local partial solutions require parallel $\Theta(n/p)$ time.

3. By Theorem 2.6, the gather step requires $O(p)$ time.

4. The minimum computed in P_1 requires $\Theta(p)$ sequential time.

5. By Theorem 2.5, the broadcast requires $O(p)$ time.

Thus the algorithm runs in $\Theta(n/p)$ time.

The worst-case optimality of this running time follows from the worst-case optimality of $\Theta(n)$ as the running time of the sequential solution. □

Remark 3.6. *If we wish to modify Algorithm 3.4 to return the index set of the Too Close objects at the earliest instance of Too-closeness, notice that in the worst case, all members of* $Q \setminus \{q_1\}$ *could be too close to* q_1 *at that time, so the index set would have cardinality* $n-1$*, too large for the application of Theorem 2.6. We can obtain such an algorithm with the same asymptotic running time as Algorithm 3.4 by not gathering the index set.*

4 "Too far" problem

If our point-objects are expected to communicate with each other, we might have an occurrence of a pair being too far to communicate; or if q_1 is required to be within a certain distance of each of the others, we might have an occurrence in which some q_j gets too far away. We are particularly interested in predicting the time of the first occurrence of such an event, since the trajectories of one or more of the objects may be changed to avoid such an occurrence.

This problem can be solved by algorithms similar to Algorithm 3.1 and 3.4. The sequential execution time is therefore a worst-case optimal $\Theta(n)$, and the execution time for a $CGM(n,p)$ is therefore a worst-case optimal $\Theta(n/p)$.

5 "3 aligned" problem

Another problem that has safety implications is the "3 aligned" problem.

- When 3 of our point-objects are aligned (or nearly aligned), the middle one may interfere with communications between the outer pair.
- If the two outer points of a (nearly) collinear triple represent hostile combatants, the middle object may prevent the outer combatants from attacking each other.

We are particularly interested in the first predictable occurrence of, say, q_1 and 2 other point-objects being (nearly) aligned, as perhaps we can call for at least one of them to change its trajectory before this occurs. We give a worst-case optimal sequential solution for this problem.

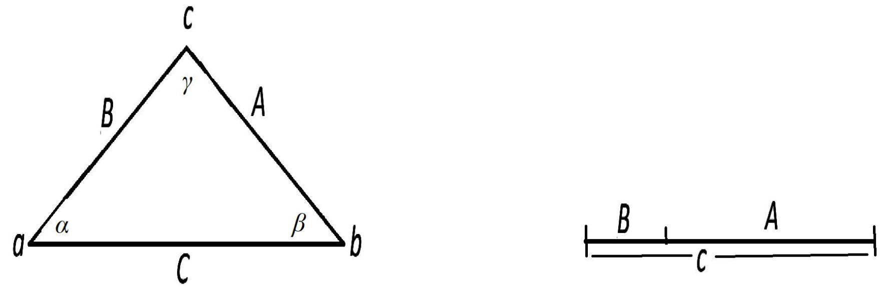

Figure 1: Assume we use the Euclidean or the Manhattan metric.
Left: For a triangle as shown, with non-collinear vertices, $C < A + B$.
Right: For collinear points as shown, $C = A + B$.

5.1 Preliminaries

We use the Manhattan metric as our distance function for this problem: for $x = (x_1, \ldots, x_d),\ y = (y_1, \ldots, y_d) \in \mathbb{R}^d$,

$$d(x, y) = \sum_{i=1}^{d} |x_i - y_i|.$$

It will be useful to determine pieces of the absolute value of a polynomial. If $f : [0, M] \to \mathbb{R}$ is a polynomial function of degree at most s, then $|f(t)|$ may have pieces on which the function is $f(t)$ and other pieces on which the function is $-f(t)$. We have the following.

Lemma 5.1. *Let $f : [0, M] \to \mathbb{R}$ be continuous such that for all $t \in [0, M]$, $f(t)$ can be computed in $\Theta(1)$ time. Let $[u, v]$ be a subinterval of $[0, M]$ such that $u < t < v$ implies $f(t) \neq 0$. Let $f(t)$ and $-f(t)$ be functions used to determine pieces of $|f(t)|$. Then we can determine in $\Theta(1)$ sequential time whether $|f(t)| = f(t)$ for all $t \in [u, v]$, or $|f(t)| = -f(t)$ for all $t \in [u, v]$.*

Proof. In $\Theta(1)$ time, we can pick t' such that $u < t' < v$ (e.g., t' could be the midpoint $(u + v)/2$ of $[u, v]$). By assumption, $f(t') \neq 0$. Since there are no zeroes of $f(t)$ on the interval (u, v), the continuity of f implies

$$|f(t)| = \begin{cases} f(t) \text{ for all } t \in [u, v] & \text{if } f(t') > 0; \\ -f(t) \text{ for all } t \in [u, v] & \text{if } f(t') < 0. \end{cases}$$

The assertion follows. □

As an immediate consequence, we have the following.

Corollary 5.2. *Let $f : [0, M] \to \mathbb{R}$ be continuous. If τ is a switchpoint for $|f(t)|$ based on $\{f, -f\}$ on $[0, M]$, then $f(\tau) = 0$.*

Example 5.3. The converse of Corollary 5.2 is not in general true, as shown by the following. Let $f : [0, 2] \to \mathbb{R}$ be given by $f(t) = (t - 1)^2$. Note $f(1) = 0$. However, since $f(t) \geq 0$ for all $t \in [0, 2]$, there is only one piece of $|f(t)|$ based on $\{f, -f\}$, namely the function f on the entire interval $[0, 2]$.

Lemma 5.4. *Let $f : [0, M] \to \mathbb{R}$ be a polynomial function.*

- *If f is a constant function with value $c \geq 0$, then $(f, [0, M])$ is the only piece of $|f|$ based on $\{f, -f\}$, on $[0, M]$.*
- *If f is a constant function with value $c < 0$, then $(-f, [0, M])$ is the only piece of $|f|$ based on $\{f, -f\}$, on $[0, M]$.*
- *Let $s \in \mathbb{N}$ be bounded. If f has degree $s > 0$, then $|f|$ has at most s switchpoints. If r is the number of switchpoints of f on $[0, M]$, then f has at most $r + 1$ pieces based on $\{f, -f\}$, on $[0, M]$.*

These pieces can be computed in $\Theta(1)$ sequential time.

Proof. For a constant function $f(t)$, $|f(t)|$ clearly has only one piece, either $(f, [0, M])$ or $(-f, [0, M])$, respectively according as the constant value c of f satisfies $c \geq 0$ or $c < 0$.

Suppose f is not constant. If the equation $f(t) = 0$ has no solutions in $[0, M]$, by assumption this can be determined in $\Theta(1)$ sequential time. Then $[0, M]$ is the interval of the only piece of $|f(t)|$. By Lemma 5.1 in an additional $\Theta(1)$ sequential time, the function of this piece, either f or $-f$, can be determined.

Suppose the equation $f(t) = 0$ has solutions $t_1, \ldots, t_v$, where $1 \leq v \leq s$. Consider the following algorithm.

Algorithm 5.5. */* Algorithm to find pieces of $|f(t)|$ from functions $\pm f(t)$ on $[0, M]$, where f is a non-constant polynomial and has solutions in $[0, M]$ to the equation $f(t) = 0$. */*

- *Solve $f(t) = 0$ for $t \in [0, M]$ and sort the unique solutions $t_1, \ldots, t_v$ so*

  $$0 \leq t_1 < \ldots < t_v \leq M$$

  Then $1 \leq v \leq s$. Since we regard s as bounded, this takes $\Theta(1)$ time.
- *If $t_1 > 0$, insert $t_0 := 0$ into this list, Similarly, if $t_v < M$, insert $t_{v+1} := M$ into this list. This takes $\Theta(1)$ time. The list now has at most $s + 2$ entries, with first index $a \in \{0, 1\}$ and last index $b \in \{v, v + 1\}$.*
- *Use Lemma 5.1 to determine the function F of $pieceInterval_a$ in $\Theta(1)$ time. Thus,*

  $$(pieceInterval_a, pieceFunction_a) := ([t_a, t_{a+1}], F).$$
- $(previousLeft, Left) := (a, a + 1)$ */* prepare for next interval */*

- *For* $right := a + 2$ *to* b
  $pieceInterval_{left} := [t_{left}, t_{right}]$ *(*$\Theta(1)$ *time)*
  Use Lemma 5.1 to determine the function F *of* $pieceInterval_{left}$
  in $\Theta(1)$ *time.*
  If $pieceFunction_{previousLeft} = F$ *then*
  / same function; extend previous interval */*
  $pieceInterval_{previousLeft} :=$
  $pieceInterval_{previousLeft} \cup [t_{left}, t_{right}]$
  Else / a new piece */*
  $pieceFunction_{left} := F$
  $pieceInterval_{left} := [t_{left}, t_{right}]$
  $previousLeft := left$ */* prepare for next interval */*
  End Else / new piece */*
  $left := right$ */* prepare for next interval */*

  End For

- *Since we may have extended pieces to adjacent intervals, we may have undefined entries* $(pieceFunction_i, pieceInterval_i)$*. We can compress the defined entries to be consecutively indexed via a semigroup operation in* $\Theta(s) = \Theta(1)$ *time* [11].

It is easily seen that the body of the For loop executes in $\Theta(1)$ time. Therefore, the algorithm executes in $\Theta(s) = \Theta(1)$ time. □

Lemma 5.6. *Let* $F, G : [0, M] \to \mathbb{R}$ *be continuous functions. Let the members of* $\{F_i\}_{i=1}^m \cup \{G_j\}_{j=1}^n$ *be polynomials of degree at most* s*. Suppose* F *is described by pieces based on* $\{F_i\}_{i=1}^m$*, with* u *switchpoints. Suppose* G *is described by pieces based on* $\{G_j\}_{j=1}^n$*, with* v *switchpoints. Then for each of the functions* $H \in \{F + G, F - G, F \cdot G\}$*, if* τ *is the endpoint of a piece of* H *based on* $\{F_i + G_j \mid 1 \le i \le m, 1 \le j \le n\}$ *(respectively,* $\{F_i - G_j \mid 1 \le i \le m, 1 \le j \le n\}$ *or* $\{F_i \cdot G_j \mid 1 \le i \le m, 1 \le j \le n\}$*) then* τ *is an endpoint of a piece of a member of* $\{F_i\}_{i=1}^m \cup \{G_j\}_{j=1}^n$*. Thus* H *has at most* $u + v$ *switchpoints in* $[0, M]$*.*

Proof. We base our proof on the contrapositive: Suppose t_0 is not an endpoint of a piece of F or of G. Then we show t_0 is not an endpoint of a piece of $F - G$. The proof for $F + G$ and the proof for $F \cdot G$ are virtually identical.

If t_0 is not an endpoint of a piece of F then the function of F does not change its piece at t_0, i.e., for some $\varepsilon_1 > 0$, F is identically equal to some F_i on $[t_0 - \varepsilon_1, t_0 + \varepsilon_1]$. Similarly, if t_0 is not an endpoint of a piece of G then the function of G does not change its piece at t_0, i.e., for some $\varepsilon_2 > 0$, G is identically equal to some G_j on $[t_0 - \varepsilon_2, t_0 + \varepsilon_2]$. Therefore, $F - G$ is identically equal to $F_i - G_j$ on $[t_0 - \varepsilon_3, t_0 + \varepsilon_3]$, where $\varepsilon_3 = \min\{\varepsilon_1, \varepsilon_2\}$. This establishes our assertion. □

Corollary 5.7. *Let* $f : [0, M]^d \to \mathbb{R}$ *where each coordinate function* $p_i \circ f : [0, M] \to \mathbb{R}$ *is a polynomial of degree at most* s*. Then* $|f|$ *has at most* ds *switchpoints, hence by Lemma 5.6 at most* $ds + 1$ *pieces. The pieces of* $|f|$ *based on* $\{(\pm(p_1 \circ f), \ldots, \pm(p_d \circ f))\}$ *can be computed in* $\Theta(1)$ *time.*

Proof. Recall we use

$$|f(t)| = \sum_{i=1}^{d} |p_i(f(t))|.$$

By Lemma 5.6, any switchpoint of $|f|$ must be a switchpoint of some $p_i(f)$. The assertion follows from Lemma 5.4. □

Corollary 5.8. *Let Q be a system of moving point objects as in section 2.1. For $i \neq j$, the distance function*

$$D_{i,j}(t) = \sum_{k=1}^{n} |p_k(f_i(t)) - p_k(f_j(t))|$$

has at most ds switchpoints, hence at most $ds + 1$ pieces based on the members of the set of polynomial functions

$$\left\{ \sum_{k=1}^{d} \pm[p_k(f_i(t)) - p_k(f_j(t))] \mid 1 \leq i < j \leq n \right\}.$$

These pieces may be computed in $\Theta(1)$ sequential time.

Proof. This follows from Corollary 5.7. □

5.2 Solving the 3 aligned problem

We are interested in the first occurrence (if any) of an ε-approximately collinear triple (q_1, q_i, q_j), where $1 < i < j \leq n$. As the time-dependent functions $A(t), B(t), C(t)$ we will use are continuous, either an ε-approximately collinear triple exists at $t = 0$ or the first occurrence (if any) exists, as in Figure 1 (Right) when

$$\varepsilon \in \{|A + B - C|, \quad |A + C - B|, \quad |B + C - A|\}.$$

Theorem 5.9. *(Suggested by Lemma 2.5 of* [3]*.) Let $F, G : [0, M] \to \mathbb{R}$ be continuous functions. Let F be described by pieces based on the members of $\mathcal{F} := \{f_i\}_{i=1}^{k}$. Let G be described by pieces based on the members of $\mathcal{G} := \{g_j\}_{j=1}^{m}$. Suppose for each pair (h_1, h_2) of distinct members of $\mathcal{F} \cup \mathcal{G}$, the graphs of h_1 and h_2 have at most s points of intersection in $[0, M]$. Then each of the functions $F + G$, $F - G$, and $F \cdot G$ has at most $(k + m)s$ switchpoints, hence at most $(k + m)s + 1$ pieces based on, respectively,*

$$\{f_i + g_j \mid f_i \in \mathcal{F}, g_j \in \mathcal{G}\}, \quad \{f_i - g_j \mid f_i \in \mathcal{F}, g_j \in \mathcal{G}\}, \text{ and } \{f_i \cdot g_j \mid f_i \in \mathcal{F}, g_j \in \mathcal{G}\},$$

and these pieces can be computed in $O(kms^2 + (k + m)s)$ time. If k, m, and s are regarded as bounded, the time estimate reduces to $\Theta(1)$.

Proof. We give a proof for $F - G$; the proofs for the others are virtually identical.

Let $\mathcal{F}'$ be the ordered union of switchpoints of members of $\mathcal{F}$ on $[0, M]$,

$$t_1 < t_2 < \ldots < t_u, \text{ for } u \leq ks.$$

Let $\mathcal{G}'$ be the union of switchpoints of members of $\mathcal{G}$ on $[0, M]$,

$$w_1 < w_2 < \ldots < w_v, \text{ for } v \leq ms.$$

Note

$$\#(\mathcal{F}' \cup \mathcal{G}') \leq ks + ms = (k+m)s.$$

By Lemma 5.6, a switchpoint of $F - G$ must come from $\mathcal{F}' \cup \mathcal{G}'$, so there are at most $(k+m)s$ switchpoints of $F - G$.

As in Lemma 5.4, there are at most $(k+m)s + 1$ pieces of $F - G$ based on the members of $\{f_i - g_j \mid f_i \in \mathcal{F}, g_j \in \mathcal{G}\}$, on $[0, M]$.

We give our algorithm for $F - G$; the others can be handled similarly.

Algorithm 5.10. *Compute the pieces of $F - G$ (or $F + G$ or $F \cdot G$), where F and G are functions as described above.*

If it is not known to have been done previously, sort the pieces of F in ascending order by the left endpoints of their intervals. Since s is bounded, this is done in $\Theta(1)$ time.

If it is not known to have been done previously, sort the pieces of G in ascending order by the left endpoints of their intervals. As above, this is done in $\Theta(1)$ time.

For each pair $(\mathcal{P}, \mathcal{Q})$ of pieces of F and of G, respectively, use Lemma 5.9 to compute the pieces of $F - G$ on the intersection of the interval of $\mathcal{P}$ and the interval of $\mathcal{Q}$ in $O(1)$ time. There are at most kms^2 such pairs and at most $(k+m)s+1$ switchpoints of $F-G$. The function $F-G$ has at most $(k+m)s+1$ pieces based on the members of $\{f_i - g_j \mid f_i \in \mathcal{F}, g_j \in \mathcal{G}\}$, on $[0, M]$. This step executes in $O((k+m)s+1)$ time.

Clearly, this algorithm runs in $O(kms^2 + (k+m)s + 1)$ time.

If k and m are regarded as bounded, the time estimate reduces to $\Theta(1)$. □

Corollary 5.11. *Let $F, G, H : [0, M] \to \mathbb{R}$ be continuous functions. Let F be described by k switchpoints based on the members of $\{f_i\}_{i=1}^{a}$. Let G be described by m switchpoints based on the members of $\{g_j\}_{j=1}^{b}$. Let H be described by r switchpoints based on the members of $\{h_u\}_{u=1}^{c}$. Then each of the functions $\pm F \pm G \pm H$ has at most $k + m + r$ switchpoints based on the members of*

$$\{\pm f_i \pm g_j \pm h_u \mid 1 \leq i \leq a, \ 1 \leq j \leq b, \ 1 \leq u \leq c\}.$$

If k, m, r, s are regarded as bounded, then all these switchpoints can be computed in $\Theta(1)$ time.

Proof. This follows by applying Theorem 5.9, first to F and G, then to $F - G$ and H. □

Remark 5.12. *Assume moving point-objects as in section 2.1. Using the Manhattan metric, $i \neq j$ implies*

$$D_{i,j}(t) := d(f_i(t), f_j(t)) = \sum_{k=1}^{d} |p_k(f_i(t)) - p_k(f_j(t))|$$

The polynomial function of a piece of $|p_k(f_i(t)) - p_k(f_j(t))|$ *is either*

$$p_k(f_i(t)) - p_k(f_j(t)) \quad or \quad -[p_k(f_i(t)) - p_k(f_j(t))].$$

There are at most s *solutions of* $p_k(f_i(t)) - p_k(f_j(t)) = 0$. *Let zeroes in [0,M] of* $p_k(f_i(t)) - p_k(f_j(t))$ *be* $Z_k = \{t_{k,1} \ldots, t_{k,u_k}\}$, *where* $0 \le u_k \le s$. *Then the pieces of* $D_{i,j}(t)$ *have at most* ds *switchpoints in* $[0, M]$ *at which some* k^{th} *component switches its base function. Therefore,* $D_{i,j}(t)$ *has at most* $ds + 1$ *pieces based on* $\{\pm[p_k(f_i) - p_k(f_j)]\}_{k=1}^{d}$.

We are ready to consider our measure of (approximate) collinearity: Since by the Triangle Inequality we must have

$$|f_i(t) - f_j(t)| \le |f_i(t) - f_1(t)| + |f_1(t) - f_j(t)|,$$

q_1 is approximately collinear with, and "between", q_i and q_j, if for sufficiently small $\varepsilon > 0$,

$$|f_i(t) - f_j(t)| + \varepsilon \ge |f_i(t) - f_1(t)| + |f_1(t) - f_j(t)| \tag{3}$$

Similar inequalities apply for approximate collinearity with either q_i "between" q_1 and q_j, or q_j "between" q_1 and q_i.

We have that either

- Approximate collinearity is satisfied at $t = 0$, i.e.,

$$|f_i(0) - f_j(0)| + \varepsilon \ge |f_i(0) - f_1(0)| + |f_1(0) - f_j(0)|; \tag{4}$$

  or similarly for the other "between" possibilities, or

- by continuity, (3) is first satisfied for

$$\varepsilon = \sum_{k=1}^{d} |p_k[f_i(t) - f_1(t)]| + \sum_{k=1}^{d} |p_k[f_1(t) - f_J(t)]| - \sum_{k=1}^{d} |p_k[f_i(t) - f_j(t)]| \tag{5}$$

  or similarly for the other "between" possibilities. In this case, Corollary 5.7 and Corollary 5.11 yield that equation (5) has at most $3ds$ switchpoints and the pieces of the right side of (5), based on the members of

$$\{\pm p_k[f_i(t) - f_1(t)] \pm p_k[f_1(t) - f_k(t)] \pm p_k[f_1(t) - f_u(t)] \mid 1 \le k \le d\}$$

  can be computed in $\Theta(ds)$ sequential time. If we assume d and s are bounded, then the running time reduces to $\Theta(1)$; or

- (3) has no solution in $[0, M]$.

Recall the following, in which the symbol f' stands for the first derivative of f.

Lemma 5.13. *Let $f : [a, b] \to \mathbb{R}$ be a differentiable function. Then each of* $\min\{f(t) \mid t \in [a, b]\}$ *and* $\max\{f(t) \mid t \in [a, b]\}$ *occurs at some member of*

$$\{a, b\} \cup \{t \in [a, b] \mid f'(t) = 0\}.$$

Remark 5.14. *Note in the following Algorithm 5.15 we consider*

$$\begin{aligned} candidates := \{t \in [0, M] \mid |a(t) + b(t) - c(t)| = \varepsilon\} \\ \cup \{t \in [0, M] \mid |a(t) + c(t) - b(t)| = \varepsilon\} \\ \cup \{t \in [0, M] \mid |b(t) + c(t) - a(t)| = \varepsilon\}. \end{aligned}$$

This enables us to consider instants in which any of $p_1(t)$, $p_i(t)$, or $p_j(t)$ is the middle point of a (near) collinear triple. If, instead, we wish to consider only instants when $p_1(t)$ is the middle point of a (near) collinear triple, we would use

$$candidates := \{t \in [0, M] \mid |a(t) + b(t) - c(t)| = \varepsilon\}.$$

5.3 Sequential solution

In the following algorithm, we compute, for all index pairs (i, j) such that $1 < i < j \leq n$, $f_1(0)$, $f_i(0)$, and $f_j(0)$. This lets us determine from (4) any (if existing) solution occurs at $t = 0$. If no such solution exists, we test for another solution.

Algorithm 5.15. *Algorithm to find first (if any) instance of an ε-approximately collinear triple of q_1 and two other members of Q, sequentially. Assume $\varepsilon > 0$ is given.*

```
tFirst = ∞, i = 2, j = 3
While tFirst > 0 and i < n
        Compute descriptions of functions
            a(t) := d(f_1(t), f_i(t)),    b(t) := d(f_1(t), f_j(t)),
                c(t) := d(f_i(t), f_j(t))
        If a(0) = 0 or b(0) = 0 or c(0) = 0 then tFirst = 0
        Else /* min not found at t = 0 for current i, j */
            candidates := {t ∈ [0, M] | |a(t) + b(t) − c(t)| = ε}
                ∪{t ∈ [0, M] | |a(t) + c(t) − b(t)| = ε}
                ∪{t ∈ [0, M] | |b(t) + c(t) − a(t)| = ε}
            If candidates ≠ ∅ then
              newCandidate := min{t ∈ candidates}
              If newCandidate < tFirst then tFirst = newCandidate
            End If candidates ≠ ∅
            j = j + 1
            If j > n then i = i + 1 and j = i + 1
        End Else /* min not found at 0 */
End While
Return tFirst
```

Theorem 5.16. *Algorithm 5.15 runs in worst-case optimal* $\Theta(n^2)$ *time and in best-case* $\Theta(1)$ *time..*

Proof. Since each of $a(t), b(t), c(t)$ has at most ds switchpoints, each computation of *candidates* requires $\Theta(1)$ time. Since in the worst case there are $\Theta(n^2)$ pairs (i, j) that must be considered, it follows that the worst-case sequential running time of the algorithm is $\Theta(n^2)$, which is optimal.

In the best case, we discover a solution at $t = 0$ for $i = 2$ and $j = 3$, so that the While loop body is executed just once. Thus, in this case, $\Theta(1)$ time is used. □

6 Further remarks

We have studied the Too Close, Too Far, and 3 Aligned problems in dynamic computational geometry, for n point-objects in motion as described in seccion 2.1. We have given algorithms for these problems whose running times are all worst-case optimal. In particular, the running times are as follows.

Problem	*Model of computation*	*Worst − case time*
Too Close	*Sequential*	$\Theta(n)$
Too Close	$CGM(n, p)$	$\Theta(n/p)$
Too Far	*Sequential*	$\Theta(n)$
Too Far	$CGM(n, p)$	$\Theta(n/p)$
3 *Aligned*	*Sequential*	$\Theta(n^2)$

We are grateful to the anonymous referees for useful suggestions.

7 Declarations

- Competing Interests: none
- Funding Information: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.
- Author contribution: The author wrote this entire paper.
- Data Availability Statement: Not Applicable
- Research Involving Human and/or Animals: Not Applicable
- Informed Consent: Not Applicable

References

[1] M.J. Atallah, Some dynamic computational geometry problems, *Computers and Mathematics with Applications* 11 (12) (1985), 1171 – 1181.

[2] J.-D. Boissonnat and M. Teillaud, eds., *Effective Computational Geometry for Curves and Surfaces*, Springer, 2006

[3] L. Boxer and R. Miller, Parallel Dynamic Computational Geometry, *Journal of New Generation Computer Systems* 2 (1989), 227–246

[4] L. Boxer and R. Miller, Dynamic computational geometry on meshes and hypercubes, *Journal of Supercomputing* 3 (1989), 161–191

[5] L. Boxer and R. Miller, Coarse grained gather and scatter operations with applications, *Journal of Parallel and Distributed Computing*, 64 (2004), 1297–1320

[6] L. Boxer and R. Miller, Efficient coarse grained data distributions and string pattern matching, *International Journal of Information and Systems Sciences* 6 (4) (2010), 424–434.
Reprinted as
L. Boxer and R. Miller, Efficient coarse grained data distributions and string pattern matching, *International Journal of Information and Systems Sciences* 7 (2) (2011), 214–224

[7] L. Boxer, R. Miller, and A. Rau-Chaplin, Scaleable parallel algorithms for lower envelopes with applications, *Journal of Parallel and Distributed Computing* 53 (1998), 91–118.

[8] L. Boxer, R. Miller, and A. Rau-Chaplin, Scalable Parallel Algorithms for Geometric Pattern Recognition, *Journal of Parallel and Distributed Computing* 58 (1999), 466-486

[9] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd Ed., Springer, 2008

[10] F. Dehne, A. Fabri, and A. Rau-Chaplin. Scalable parallel geometric algorithms for multi-computers, *Proceedings 9th ACM Symposium on Computational Geometry* 1993, 298–307

[11] R. Miller and L. Boxer, *Algorithms Sequential and Parallel, A Unified Approach*, 3rd ed., Cengage Learning, Boston, 2013.

[12] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, New York, Springer–Verlag, 1985

[13] G. Trajcevski, H. Ding, P. Scheuermann, R. Tamassia, and D. Vaccaro, Dynamics-aware similarity of moving objects trajectories, *Proceedings of the 15th International Symposium on Advances in Geographic Information Systems*, 2007.