

Kinetic Nearest Neighbor Search in Black-Box Model

Bahram Sadeghi Bigham*

Maryam Nezami[†]Marziyeh Eskandari[‡]

Abstract

Proximity problems is a class of important problems which involve estimation of distances between geometric objects. The nearest neighbor search which is a subset of proximity problems, arises in numerous fields of applications, including Pattern Recognition, Statistical classification, Computer vision and etc. In this study, a nearest neighbor search is presented to move points in the plane, while query point is static.

The proposed method works in the black-box *KDS* model, in which the points location received at regular time steps while at the same time, an upper bound d_{max} is known on the maximum displacement of any point at each time step. In this paper, a new algorithm is presented for kinetic nearest neighbor search problem in the black-box model, under assumptions on the distribution of the moving point set P . It has been shown how the kinetic nearest neighbor will be updated at each time step in $O(k\Delta_k \log n)$ amortized time, where Δ_k is the k -spread of a point set P .

Key words: Computational Geometry, Black Box Model, Kinetic, Nearest Neighbor.

1 Introduction

In recent years, there has been a growing amount of research dealing with moving, or kinetic, data. Algorithms dealing with objects in motion traditionally discretized time and recompute the structure of interest at every time step from scratch. This can be wasteful, especially if time steps are small. Since objects will move more slightly, and the structure may not change at all. Ideally an object receives attention if and only if, its new location triggers an actual change in the structure. A basic assumption in the Kinetic Data Structure (KDS), which is introduced by Basch et al. [1], is that the object trajectories are known. This assumption severely limits the applications of KDS framework.

*Corresponding author: Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences (IASBS), Prof. Yousef Sobouti Blvd., Zanjan, Iran. b.sadeghi.b@iasbs.ac.ir

[†]Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences (IASBS), Prof. Yousef Sobouti Blvd., Zanjan, Iran. mnezami@gmail.com

[‡]Department of Computer Science and Information Technology, Alzahra University, Tehran Province, Tehran, Vanak Village Street, eskandari@alzahra.ac.ir

Sometimes in an online manner, the object locations are available only at time steps. So when future tracking moving object is not available, the basic assumption in the KDS model is not always valid (refer to surveys by Guibas for more information about KDSs [5, 6]). Certainly, there is a need for a hybrid model, which combines ideas from the KDS model with a traditional time-slicing approach. De Berge et al. [2] introduced kinetic data structures in black-box model with fewer restrictions, instead of knowing knowledge of the trajectories. They only assume that they know upper bounds on objects' speed and can compute their positions at regular time steps. They develop KDSs in the black-box model that consider under certain assumptions on the trajectories which is proved to be more efficient than recomputing the structure from scratch. Additionally, in recent years, some problems (e.g., convex hull, 2-center) have been studied in black box model [2, 3].

Proximity problems is a class of important problems in computational geometry which involve estimation of distances between geometric objects. The nearest neighbor search problem arises in numerous fields of applications, including Pattern Recognition, Statistical classification, Computer vision and etc.

In this paper, a black-box KDSs has been studied to find the nearest neighbor in a set P of n points moving to a static query point q in the plane. There is a need to make assumptions on the point movements and time steps to obtain proved efficient solutions. Time steps should be small enough to have coherency between the objects' positions in consecutive time steps. Otherwise, there is no better way than recomputing the structure from scratch. Also, it has been assumed that in most of our results that P is fairly distributed at each time step.

In the following, firstly, the black-box model which was introduced by de Berg et al. [2] and the concept of k -spread will be described. Furthermore, an algorithm that updates the nearest neighbor search at each time step in $O(k\Delta_k \log n)$ amortized time, where Δ_k is the k -spread of a point set P will be presented.

The Back-Box model: Let P be a set of moving points in plane. In black-box model, the exact motions of the points are unknown and a new location $p(t)$ for each point $p \in P$ is available at regular time steps $t =$

t_1, t_2, t_3, \dots . The goal is to update the nearest neighbor to q at each time step.

For any point $p \in P$, it is assumed a maximum displacement d_{max} that indicates how far the point can move between two consecutive time steps. For a (static) set S of points and a point $s \in S$, let $NN_k(s, S)$ denote the k -th nearest neighbor of s in $S \setminus \{s\}$. Let $dist(p_1, p_2)$ denotes the Euclidean distance between two points p_1 and p_2 , and

$$mindist_k(S) = \min_{s \in S} dist(s, NN_k(s, S)).$$

At any time step t , d_{max} is assumed the most $mindist_k(P(t))$.

Displacement Assumption: There is a maximum displacement d_{max} , such that at any time step t_i :

- $d_{max} \leq mindist_k(P(t_i))$, and
- $dist(p(t_i), p(t_{i+1})) \leq d_{max}$ for each point $p \in P$.

This means that there are no more than k points within a distance d_{max} of each other.

The k -spread of a point set: In this model, a conception called k -spread is utilized, as introduced by Erickson [4] for distribution of a set of point. The k -spread, Δ_k of a set P of static points is defined as:

$$\Delta_k(P) = \frac{diam(P)}{mindist_k(P)}$$

in which $diam(P)$ is diameter of P .

Lemma 1 *Let P be a set of points and R be a region in the plane such that $diam(R) < mindist_k(P)$, then R contains at most k points of P [2].*

2 Maintaining the nearest neighbor search

Let P be a set of n moving points in the plane and q be static query point that adheres to the Displacement Assumption with parameters k and d_{max} . The goal is to compute the nearest neighbor to q from moving points P at regular time steps $t = t_1, t_2, t_3, \dots$. Let $NNS(P(t))$ denotes the nearest neighbor in $P(t) := \{p(t) : p \in P\}$ to q . $NNS(t)$ is used as a shorthand for $NNS(P(t))$.

The maintenance of some structures always depends on all the points and the others are defined by only a subset of the points. The nearest neighbor search is latter type; so as previously mentioned, the algorithm does not need to ask for all new positions at every time step. It may ignore some points, if the new locations of these points cannot change the structure. Thus, an efficient algorithm is potentially exist. A subset $Q(t) \subseteq P(t)$ is introduced in this paper, so that $NNS(Q(t)) = NNS(t)$. A point in $P(t)$ is called active if it is part of $Q(t)$ at time step t ; and the points that are not in $Q(t)$ are

called inactive. $NNS(t)$ can be computed using only points from Q . If Q is much smaller than P , this may be much faster than computing $NNS(t)$ directly. In a kinetic setting, knowledge from previous time steps can be used to find Q quickly. The coherency between structures at successive time steps are to find Q quickly at each time step. To this end, a bound can be computed for each point, on the number of time steps that must pass before it can become part of Q .

Let P be a set of moving points and t be a time step. A function $\tau_{NNS}(p, t)$ is called an inactivity function if the point p is inactive at any time t_i with $t < t_i \leq t + \tau_{NNS}(p, t)$.

For the nearest neighbor search problem, it is defined

$$\tau_{NNS}(p, t) = \lfloor \frac{dist(p, q) - dist(q, NNS(t))}{2d_{max}} \rfloor$$

where $dist(p_1, p_2)$ denotes the minimum Euclidean distance p_1 to p_2 . The following lemma shows that this is a valid inactivity function.

Lemma 2 *For any point $p \in P$ and any time step t , $p(t_i)$ is inactive at any time step t_i with $t < t_i \leq t + \tau_{NNS}(p, t)$.*

Proof. In the black-box model, each point p can move at most d_{max} per time step. This includes the nearest neighbor of q , therefore the distance between a point p and q can decrease by at most $2d_{max}$ per time step. Also, the distance between $NNS(t)$ and q can increase by at most $2d_{max}$ in each time step; as $dist(p, q) - dist(q, NNS(t))$ can decrease by at most $2d_{max}$ per time step. As a result, p cannot be active point at any time step t_i with $t < t_i \leq t + \tau_{NNS}(p, t)$. \square

For each point $p \in P$, a time stamp T_p maintained that indicates the first time in the future at which p can be active. At the time step $t = T_p$, we say that the time stamp of p has expired.

The general algorithm thus works as follows: $NNS(t_0)$ initially is computed by scratch. Then each point $p \in P$ enter in a queue with the time stamp $T_p = t_0 + \tau_{NNS}(p, t) + 1$ as its key. At each time step t , the set $Q(t)$ of all points with key t retrieved from the queue. Then $NNS(Q(t))$ is computed and, hence, $NNS(t)$. Finally each point $p \in Q(t)$ reinserted into the queue with its new time stamp $T_p = t + \tau_{NNS}(p, t) + 1$.

To implement this algorithm, an array A is used where $A[t_i]$ contains the points which time stamps expire at time t_i . To restrict the amount of storage, an array $A[0 \dots n - 1]$ is used with n entries, where $A[i]$ stores all points with time stamp $i = T_p \bmod n$. The time stamps are bounded to be at most n steps. This approach enables us to add and remove points from the queue in $O(1)$ time per point as opposed to $O(\log n)$ with a standard priority queue structure. The proposed

approach is made explicit in Algorithm 1. Note that the algorithm needs to know only d_{max} to work properly and it does not need to know bounds on the k -spread.

Algorithm 1 UPDATENNS

```

 $Q(t) \leftarrow$  set of points stored in  $A[t]$ 
Compute  $NNS(Q(t))$  and set  $NNS(P(t)) \leftarrow NNS(Q(t))$ 
for each  $p \in Q(t)$  do
    compute  $\tau_{NNS}(p, t)$ 
    Add  $p$  to  $A[t + \min(\tau_{NNS}(p, t) + 1, n) \bmod n]$ 
end for
 $t \leftarrow (t + 1) \bmod n$ 

```

It is worth mentioning that how can compute $NNS(t)$ and τ_{NNS} . Computing $NNS(Q(t))$ can be done by nearest neighbor search algorithm at static mode in $|Q(t)|$ time and clearly computing τ_{NNS} is done in $O(1)$ time, so the time to update algorithm 1 at time step t , depends on the size of $Q(t)$. In the worst case, all points may expire in a single time step, but when the k -spread of P is low, it is possible to show that on average only a small number of points expire.

Lemma 3 *At each time step t , UPDATENNS updates the nearest neighbor in P to q in $O(|Q(t)|)$ time.*

The number of points that can expire in a single time step -the size of $Q(t)$ - can be bounded by amortized analysis using $\gamma_{NNS}(P)$ as a parameter, which will define a bound on the maximum number of points $p \in P$ at any time t which τ_{NNS} has the same value.

Lemma 4 *The number of points $p \in P$ for which $\tau_{NNS}(p, t) = i$ is bounded by $\gamma_{NNS}(P) = O(k\Delta_k)$ for any $0 \leq i \leq n$.*

Proof. Let G_i denotes the set of points $p \in P$ for which $\tau_{NNS}(p, t) = i$, according to the definition $\tau_{NNS}(p, t)$ for any point $p \in P$ can conclude the following statement:

$$i \cdot 2d_{max} \leq \text{dist}(p, q) - \text{dist}(q, NNS(t)) < (i + 1) \cdot 2d_{max}$$

So, the number of points which distance $\text{dist}(p, q) - \text{dist}(q, NNS(t))$ is between $i \cdot 2d_{max}$ and $(i + 1) \cdot 2d_{max}$ should be bounded. The points in G_i reside in area C that which are centered at the same point q , but differ in radius length $2d_{max}$ as illustrated in Figure 1. Now consider overlaying this area with an axis-aligned grid \mathcal{G} of which each cell has edge length $\text{mindist}_k(P)$. From Lemma 1, it is clear that each cell contains $O(k)$ points. This region C can be subdivided into four axis aligned rectangles R_n , R_w , R_s and R_e with edge lengths at most $2d_{max}$ and $\text{dam}(P)$ (see Figure 2). Firstly, the maximum number of cells in R_n are computed. From the definition of the k -spread

and the Displacement Assumption, it is obvious that each rectangle of R_n intersects at most $O(\Delta_k)$ cells of this grid and similar argument can be made for R_w , R_s and R_e and so, it follows that $R = R_n \cup R_s \cup R_e \cup R_w$ contains $O(\Delta_k)$ cells of this grid. Then, according the areas defined in Figure 3, should be fixed out the maximum number of cells in C is less than R .

Let N_C denotes the maximum number of cells of C (and similarly for other regions). The areas of r_{ne} , r_{se} , r_{nw} and r_{sw} are pink regions, c_{ne} , c_{se} , c_{nw} and c_{sw} are yellow regions and red regions in Figure 3 are s_{ne} , s_{se} , s_{nw} and s_{sw} . In the following, we define:

$$N_C = N_R + N_{c_{nw}} + N_{c_{ne}} + N_{c_{se}} + N_{c_{sw}} - N_{r_{nw}} - N_{r_{sw}} - N_{r_{se}} - N_{r_{ne}} - N_{s_{nw}} - N_{s_{sw}} - N_{s_{se}} - N_{s_{ne}}.$$

It is clear that the area of c_{nw} is less than the area of r_{nw} ; therefore $N_{c_{nw}} < N_{r_{nw}}$. Similarly it can be seen that $N_{c_{ne}} < N_{r_{ne}}$, $N_{c_{se}} < N_{r_{se}}$ and $N_{c_{sw}} < N_{r_{sw}}$. So, according to statement , the maximum number of cells in C is less than R . As a result, C contains $O(\Delta_k)$ cells. As already mentioned, each cell contains $O(k)$ points from P , so set G_i contains $O(k\Delta_k)$ points. \square

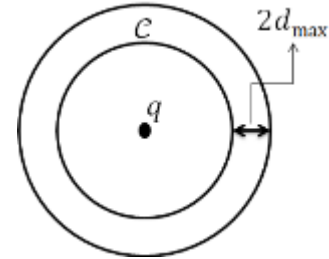


Figure 1: The area of C include set of points $p \in P$ which $\tau_{NNS}(p, t) = i$

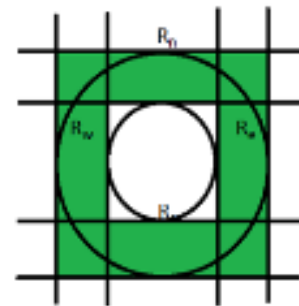


Figure 2: Rectangles R_n , R_s , R_w and R_e

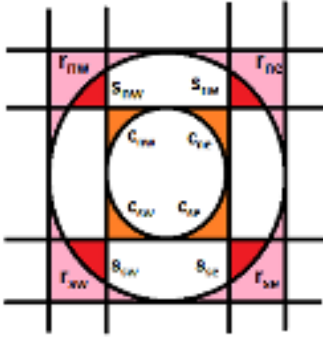


Figure 3: The areas of r_{nw} , r_{ne} , r_{sw} and r_{se} , s_{nw} , s_{ne} , s_{sw} and s_{se} , c_{nw} , c_{ne} , c_{sw} and c_{se}

In the worst case, all points will expire in a single time step. However, using an amortization argument and Lemma 4 it can be shown that on average, only $O(k\Delta_k \log n)$ point will expire in each time step.

Lemma 5 *If the number of points $p(t) \in P(t)$ with $\tau_{NNS}(p, t, P) = i$ is at most $\gamma_{NNS}(P)$ for any $0 \leq i \leq n$ and any t , then on average only $O(\gamma_{NNS}(P) \log n)$ points expire per time step [2].*

From Lemma 3 and 5 the following theorem can be concluded:

Theorem 6 *Under the Displacement Assumption, the nearest neighbor in a set P of n points moving to static query point q in the plane, can be maintained in the black-box model in $O(k\Delta_k \log n)$ amortized time per time step, where Δ_k is the maximum k -spread of P at any time.*

3 Conclusion

In this paper, an algorithm is presented to maintain the nearest neighbor in a set points moving to static query point in the KDS black-box model. The algorithm does not require knowledge of k or Δ_k . It only needs to know d_{max} , the maximum displacement of any point in one time step and also does not need to know the point trajectories. It is also shown that the proposed algorithm can update nearest neighbor in $O(k\Delta_k \log n)$ amortized time at each time step. Interesting open problems arise when someone talks about time bound in worst-case rather than amortized.

References

- [1] J. Basch, L. J. Guibas and J. Hershenberger, *Data structure for mobile Data*, Journal of algorithms, 31(1): 1-28, 1999.

- [2] M. de Berg, M. Roeloffzen and B. Speckmann, *Kinetic convex hulls, delaunay triangulations and connectivity structures in the black-box model*, Journal of Computational Geometry, 3 (1), 222-249, 2012.
- [3] M. de Berg, M. Roeloffzen and B. Speckmann, *Kinetic 2-center in the black-box model*, Proceedings of the 29th annual symposium on Symposium on computational geometry, 145-154, 2013.
- [4] J. Erickson, *Dense point sets have sparse delaunay triangulations or ... but not too nasty*, Discrete and Computational Geometry, Volume 33, Issue 1, pp 83115, 2005.
- [5] L.J. Guibas, *Kinetic data structures a state-of-the-art report*, In Proc. 3rd Workshop Algorithmic Found. Robot., pages 191-209, 1998.
- [6] L.J. Guibas. Kinetic data structures. In: D. Mehta and S. Sahni, *Handbook of Data Structures and Applications*, Chapman and Hall/CRC, 2004.
- [7] Agarwal, Pankaj K and Kaplan, Haim and Sharir, Micha. *Kinetic and dynamic data structures for closest pair and all nearest neighbors*, ACM Transactions on Algorithms (TALG), Volume 5 Issue 1, Article No. 4, 2008.
- [8] Rahmati, Zahed and Abam, Mohammad Ali and King, Valerie and Whitesides, Sue and Zarei, Alireza, *A simple, faster method for kinetic proximity problems*, Journal of Computational Geometry: Theory and Applications, Volume 48, Issue 4, 342-359, 2015.