Defence Science Journal, Vol. 64, No. 1, January 2014, pp. 41-47, DOI:10.14429/dsj.64.2952

Fast Parameterless Ballistic Launch Point Estimation based on *k*-NN Search

Soojin Kim[*], Hyunjoong Kim, and Sungzoon Cho

Seoul National University, Seoul, Korea

[*]*E-mail: bum0528@snu.ac.kr*

ABSTRACT

This paper discusses the problem of estimating a ballistic trajectory and the launch point by using a trajectory similarity search in a database. The major difficulty of this problem is that estimation accuracy is guaranteed only when an identical trajectory exists in the trajectory database (TD). Hence, the TD must comprise an impractically great number of trajectories from various launch points. Authors proposed a simplified trajectory database with a single launch point and a trajectory similarity search algorithm that decomposes trajectory similarity into velocity and position components. These similarities are applied *k*-NN estimation. Furthermore, they used the iDistance technique to partition the data space of the high-dimensional database for an efficient *k*-NN search. Authors proved the effectiveness of the proposed algorithm by experiment.

Keywords: *k*-NN search, iDistance, trajectory similarity search, launch point estimation

NOMENCLATURE

S	- Sensed trajectory (Arbitrary partial trajectory detected by sensor)
TD	- Trajectory database (Database of ballistic trajectories from unified launch point)
Sub-TD	- Sub-trajectory database (Database of sub-trajectories partitioned into sensing time intervals and translated to origin)
MSST	- Most similar sub-trajectory (Sub-trajectory that has smallest distance from sensed trajectory)
MST	- Most similar trajectory (Whole trajectory that includes MSST)

1. INTRODUCTION

This paper addresses the problem of estimating the trajectory and launch point of enemy artillery projectiles. For tactical and technical reasons, the radar system of field artillery generally tracks an arbitrary trajectory at regular intervals for only a limited number of seconds. Real-time ballistic trajectory radar can track an arbitrary trajectory during a limited time by means of information with three degrees of freedom, whereas a ballistic trajectory with time varying aerodynamic characteristics such as drag coefficient and lift can be simulated by equations with six degrees of freedom. Therefore, using a kinetic model in a fire control system with radar detected information causes an inaccurate estimation of the ballistic trajectory and launch point.

Developing the more effective search algorithm in this area is an important matter because the time of search of launch point is a crucial in this type of applications. However, it is difficult to find appropriate references in this area of work. There are some existing research reports on launch point estimation. A maximum likelihood (ML) estimator based on the Levenberg-Marquardt algorithm was applied to estimate ballistic missile launch points by using line of sight (LOS) measurements from one or more passive sensors[1]. Batch data processing and nonlinear regression, augmented with an intercept as a multivariate parameter, have been proposed[2]. Launch point estimation based on data is founded, which necessitates a multi-parameter decision process and full search[3]. We proposed a *k*-NN search based estimation that is parameter less for a ballistic trajectory launch point[4]. But, it didn't still overcome the time complexity of full search. Therefore, in this paper, we adapt iDistance which is one of high dimensional vector space partitioning algorithms to offer efficient *k*-NN search method for estimating ballistic launch point.

We study the problem of estimating a ballistic trajectory and the launch point using a trajectory similarity search for a ballistic sub-trajectory observed by a radar system. The major difficulty of this problem is that estimation accuracy is guaranteed only when a similar trajectory exists in the TD. Accordingly, the TD must consist of numerous trajectories from every possible launch point under various launch conditions (such as speed and angle). Even if the TD consists of trajectories that reflect every possible launch point, time complexity poses a serious problem. Therefore, authors proposed a novel data-based ballistic launch point estimation process that utilizes a simplified TD and a simplified sub-TD. Furthermore, we use data-space partitioning to reduce the calculation time.

2. SIMPLIFIED TRAJECTORY DATABASE

The phrase trajectory similarity refers to temporal components as well as shape components. The TD introduces temporal issues related with derived parameters of motion such as speed and direction[5]. Figure 1 shows two hurricane trajectories that are not similar with respect to matters of speed, despite having the same shape and similar locations. All the

Received 20 December 2012, revised 21 November 2013, online published 23 January 2014

factors that characterize a trajectory (locality, temporality, directionality, and rate of change) can be flexibly formulated. This approach is effective not only to query for a similar trajectory but also to support trajectory clustering[6].

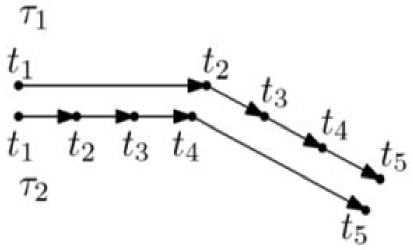


Figure 1. Similarity between two hurricane trajectories (van Kreveld & Luo[5]).

Although they are same shapes, trajectories can be different by its starting points. If they are moved into the same starting point, they can be considered identical trajectory. In other words, there are innumerable trajectories which have identical shape and different starting point. Thus, to reduce the size of TD, we proposed simplified TD by using only the relative position of trajectory, i.e. the shape component of trajectory. The simplified TD assumed for a ballistic trajectory similarity search is the ideal case of a unified launch point obtained by parallel transformations along the *x*- and *y*axes. Figure 2 shows that the TD is made simple by unifying the launch points. Under the same conditions, Fig. 2(a) depicts trajectories from three different launch points and Fig. 2(b) depicts trajectories from a unified launch point.

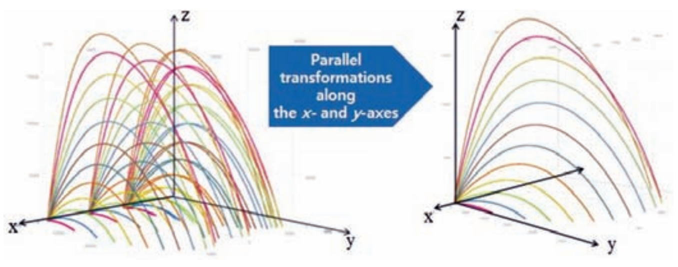
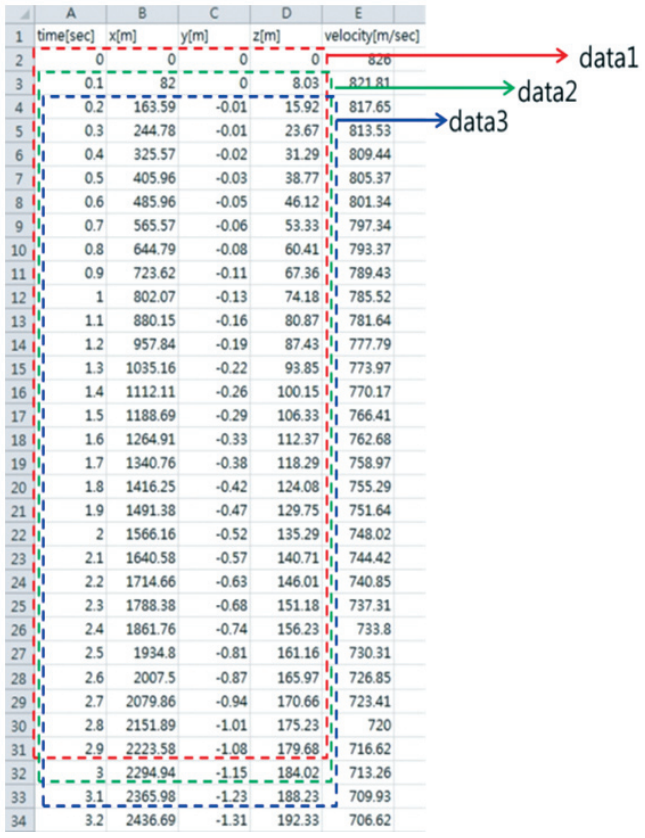


Figure 2. (a) Trajectories from six launch points (b) Trajectories from a unified launch point.

In our simplified TD, every trajectory has a relaxed location component so that similarities in the temporal and shape components can be considered. Some trajectory conversion processes have to precede the comparison of a sensed trajectory with trajectories in the simplified TD. For convenience of explanation, we define as follows:

- A real coordinate trajectory is the trajectory of a moving object in the real coordinate system.
- A regulated coordinate trajectory is the trajectory of a moving object in a relative coordinate system assuming that the object is launched from the origin.

As described above, trajectory similarity is represented by timestamps as well as route. For a trajectory similarity search, we must compare the direction and length of two trajectories at every timestamp. Generally, field artillery radar tracks an arbitrary ballistic trajectory using a fixed interval (t) for a short time period (T), for example 3 s. In this case, only if the time lengths of trajectories are identical, the trajectory similarity is worth to consideration. Thus, the trajectories in the TD should be partitioned into sub-trajectories with time length (T) and timestamp interval (t) identical to those of a sensed trajectory. The sub-trajectories with sensing time scale are the data by moving window. In Fig. 3, 'data 1' consists of 30 rows time table with 3-d coordinates at interval of 0.1 s. It can be represented by 90 dimensional vectors, the first 30 columns represent x coordinate, and the second 30 columns represent y coordinate. The sub-TD consists of these high dimensional sub-trajectory vectors. To check the temporal and shape components of each sub-trajectory for similarities with those of the sensed trajectory in relative coordinates, the sub-trajectories need to have position components relaxed from those given by the original whole trajectory. All sub-trajectories are also translated to the origin. Eventually, the simplified sub-trajectory database (sub-TD) consists of sub-trajectories extending from the origin by a sensing time interval. Figure 4 shows a conceptual example in which a TD is converted into a sub-TD. The 1,366 trajectories were divided into 2,188,591 sub-trajectories. For visibility, some sub-trajectories are described in detail.



data1
data2
data3

	A	B	C	D	E
1	time[sec]	x[m]	y[m]	z[m]	velocity[m/sec]
2	0	0	0	0	826
3	0.1	82	0	8.03	821.81
4	0.2	163.59	-0.01	15.92	817.65
5	0.3	244.78	-0.01	23.67	813.53
6	0.4	325.57	-0.02	31.29	809.44
7	0.5	405.96	-0.03	38.77	805.37
8	0.6	485.96	-0.05	46.12	801.34
9	0.7	565.57	-0.06	53.33	797.34
10	0.8	644.79	-0.08	60.41	793.37
11	0.9	723.62	-0.11	67.36	789.43
12	1	802.07	-0.13	74.18	785.52
13	1.1	880.15	-0.16	80.87	781.64
14	1.2	957.84	-0.19	87.43	777.79
15	1.3	1035.16	-0.22	93.85	773.97
16	1.4	1112.11	-0.26	100.15	770.17
17	1.5	1188.69	-0.29	106.33	766.41
18	1.6	1264.91	-0.33	112.37	762.68
19	1.7	1340.76	-0.38	118.29	758.97
20	1.8	1416.25	-0.42	124.08	755.29
21	1.9	1491.38	-0.47	129.75	751.64
22	2	1566.16	-0.52	135.29	748.02
23	2.1	1640.58	-0.57	140.71	744.42
24	2.2	1714.66	-0.63	146.01	740.85
25	2.3	1788.38	-0.68	151.18	737.31
26	2.4	1861.76	-0.74	156.23	733.8
27	2.5	1934.8	-0.81	161.16	730.31
28	2.6	2007.5	-0.87	165.97	726.85
29	2.7	2079.86	-0.94	170.66	723.41
30	2.8	2151.89	-1.01	175.23	720
31	2.9	2223.58	-1.08	179.68	716.62
32	3	2294.94	-1.15	184.02	713.26
33	3.1	2365.98	-1.23	188.23	709.93
34	3.2	2436.69	-1.31	192.33	706.62

Figure 3. A way to segment a trajectory to serial sub-trajectories data.

To calculate the similarity between trajectories, a pair should be compared under identical conditions. Thus, through some conversion process, the coordinates of a sensed trajectory and trajectories in the TD are realigned and synchronized. Figure 5 shows the framework of the trajectory conversion process. In the upper box, the trajectories in the TD are converted into a sub-TD by a batch process in advance. In the lower box, the sensed trajectory is converted by the online process as soon as radar systems observe a trajectory and transmit its information. This will be explained in the next section. Then, a *k*-nearest neighbour (*k*-NN) search to compare the transformed sensed trajectory (*S′*) with trajectories in the sub-TD is possible.

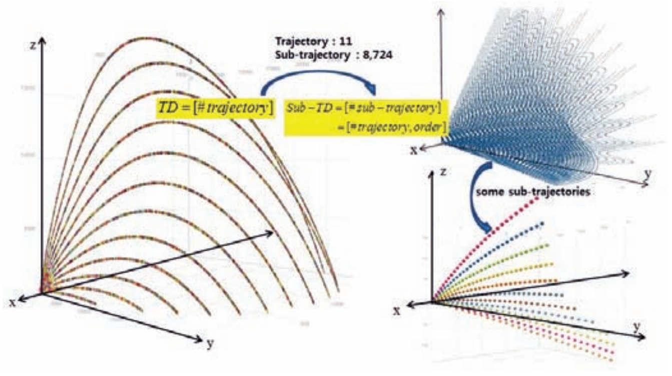


Figure 4. The concept of converting a TD into a sub-TD[4].

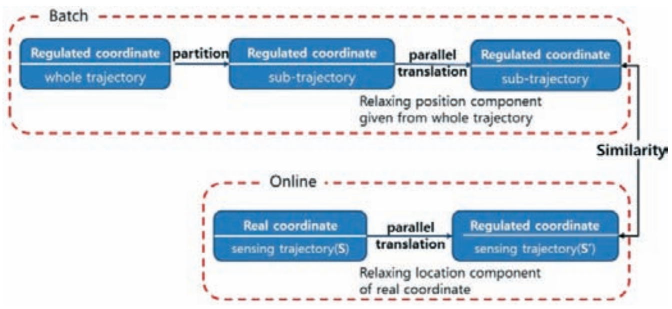


Figure 5. Trajectory conversion framework.

3. *k*-NN SEARCH

3.1 Trajectories Similarities in Shape and Time Dependence

First, we consider similarities in temporal and shape components between the sensed trajectory (*S*) and trajectories in the sub-TD. The temporal and shape components represent the rate of change in velocity. Where as *S* is obtained in real coordinates, the trajectories in the sub-TD are regulated coordinate trajectories in relative coordinates. Thus, the location component of *S* in real coordinates should be relaxed by translation to the origin. This translation function and the converted *S* are represented by *T*(*S*) and *S′* as follows

$$S' = T(S) \tag{1}$$

where

$$S = [(t_1, s_1), \cdots, (t_n, s_n)]$$

$$s_i = (x_i, y_i, z_i)$$

The similarity between two points is inversely proportional to the dissimilarity, which can be calculated from a distance measure. The most similar sub-trajectory (MSST) means the first-nearest neighbour; i.e., the trajectory with the shortest distance from S′ of those in the sub-TD. Let us define how to measure the distance between two trajectories. As a trajectory is a time series of points, a sub-trajectory can be regarded as a set of line segments whose lengths correspond with an identical time interval. The dissimilarity between the two sub-trajectories is defined as the integration of the line segment[7].

The Euclidean distance between the two points in 3-dimension can be calculated as follows

$$D_{P,Q} = \sqrt{(Q_x - P_x)^2 + (Q_y - P_y)^2 + (Q_z - P_z)^2} \tag{2}$$

where $P_{i,}Q_i$ is the coordinate on the *i*-axis of the point *P*,*Q*

$i=x,y,z$

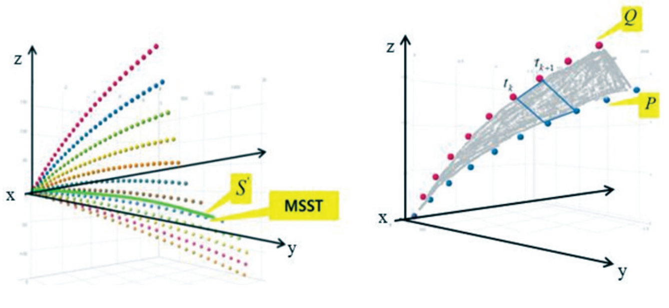
The distance between two sub-trajectories is calculated by integration of Eqn. (2) over a time interval. In order to reduce the computational complexity, the trapezoidal rule is used. The dissimilarity between two sub-trajectories can be approximated as follows

$$DISSIM(P,Q) \cong \frac{1}{2}\sum_{k=1}^{n-1}((D_{P,Q}(t_k) + D_{P,Q}(t_{k+1}) \bullet (t_{k+1} - t_k)) \tag{3}$$

where $D_{P,\,Q}(t_k)$ is the Euclidean distance between two points *P* and *Q* at the t_k

The distances between *S′* and all the trajectories in the sub-TD can be calculated from Eqn (3). Then, each dissimilarity between two trajectories can be used as a weight for *k*-NN estimation. Figure 6(a) shows a sub-trajectory that is determined the MSST of *S′* based on shape similarity. Figure 6(b) shows the concept of a distance measure between a pair of trajectories. To avoid overfitting to noise in the data, the idea of the MSST is extended to *k* neighbours ($k > 1$) as the *k*-MSST. In addition, averaging estimated launch points from several similar trajectories can prevent extreme wrong estimation that can occur by considering only small nearest neighbours.

d_i: dissmilarity between *S′* and each *k*-MSST ($1 \leq i \leq k$) (4)



(a) **(b)**

Figure 6. (a) Most similar sub-trajectory, (b) Concept of distance measure.

3.2 Weight of Elevation

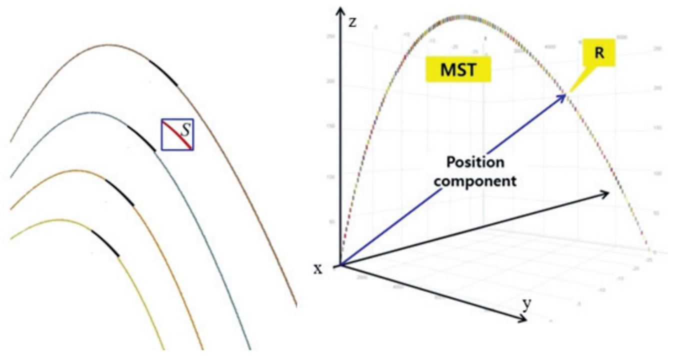
Figure 7(a) shows that sub-trajectories with only shape similarity can occur in various parts of several whole trajectories. Thus, location component in similarity should also be considered in searching for an exact *k*-NN. Authors proposed locational similarity as the similarity weight in *k*-NN estimation. If a sub-trajectory is determined the MSST of *S′*, its whole trajectory is estimated the most similar trajectory (MST) for the whole trajectory *S*. Thus, the location components of *S* and MSST in the TD should be compared. In the TD, the position components of the MSST are restored in the MST as *R*. In Fig. 7(b), *R* is the restored original position of the estimated MST in the TD. Since all trajectories in the database emerge from a unified launch point by parallel transformations in the *x*- and *y*-directions, only the difference in elevation is considered when comparing locations. The *z*-coordinate difference between the MSST and *S* is considered the elevation weight

$$h_i = \max\left|z_t(S) - z_t(R_i)\right| \quad (5)$$

where

$$z_t(S) = z - \text{coordiate value } S \text{ at } t$$

$$z_t(R_i) = z - \text{coordiate value } R_i \text{ at } t$$



(a) (b)

Figure 7. (a) Sub-trajectories with only shape similarity and (b) Location component of sub-trajectory.

3.3. *k*-NN Estimation of Launch Point

The distance from each k-MSST to *S′* is used as a weight to influence the launch point estimation. The main concept of launch point estimation is that R_0 represents the restored coordinate or starting point of the MSST in the MST. This concept is reversely applied to the starting point of the sensed trajectory (S_0). Each k-MSST contributes to the estimation as a dissimilarity. Finally, a launch point is estimated as follows:

$$\hat{LP} = S_0 - \frac{1}{\sum_{i=1}^{k} \frac{1}{d_i^2} \times \frac{1}{h_i^2}} \sum_{i=1}^{k} \frac{1}{d_i^2} \times \frac{1}{h_i^2} \times R_{i0} \quad (6)$$

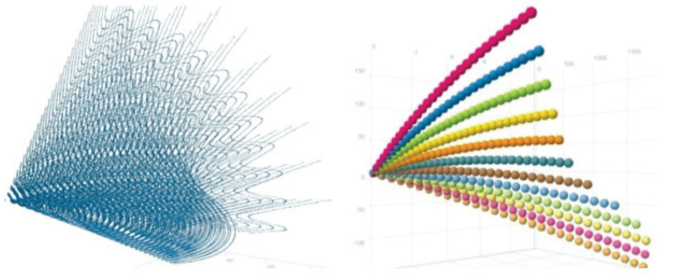
where

d_i : the shape component dissimilarity weight

h_i : the elevation difference weight

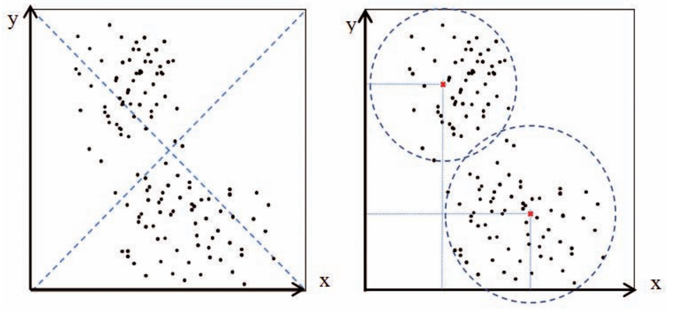
4. DATA SPACE PARTITIONING

Trajectories are represented by time series data, which are typically high-dimensional data. In Fig. 8(a), a ballistic sub-trajectory database consists of a dense pattern of similar trajectories. In Fig. 8(b), the sub-trajectory data are highly correlated because of directivity. The various structures for multidimensional data can be divided into two categories, which are known as R-trees [8-10]and X-trees [11]. These types of index structures are not appropriate for a high-dimensional dataset because of the 'curse of dimensionality': as the dimensionality increases, the performance deteriorates because of the overlap among bounding boxes[12,13]. Because sub-trajectory data consists of high dimensional vector, we apply suitable strategy for high-dimensional data structures to solve time complexity problem. In recent year, an alternative approach to indexing high-dimensional data has been the mapping strategy for data space partitioning. The pyramid-technique[14] was developed for hypercube range queries over a uniform data distribution and thus it can only perform the space-based partitioning described in Fig. 9(a). As the iDistance[15] technique is also capable of the data-based partitioning described in Fig. 9(b), it yields high performance on real data that are clustered and correlated.



(a) (b)

Figure 8. (a) Ballistic sub-trajectory database and (b) Ballistic sub-trajectory data.



(a) (b)

Figure 9. (a) Space-based partitioning and (b) Data-based partitioning.

4.1. *k*-NN Query Algorithm

A *k*-NN search probes the database for the *k* objects nearest to the given query *q*. In mapping strategy indexing, an NN search starts from a query sphere with radius Δr. The radius of the query sphere increases by Δr with each iteration until the search retrieves the complete answer set of minimum radius, which consist of the *k* nearest neighbours of the query point. The algorithm terminates when the vector space distance of the furthest object in the answer set is less than or equal to that of any object within the current search radius *r* with respect to the query *q*.

4.2. iDistance

iDistance was proposed for efficient k-nearest neighbour searches in high-dimensional data spaces[14]. First, the data space is partitioned and a reference point is defined for each partition. Second, the distance of each data point from the reference point of its partition is indexed with a one-dimensional value. Figure 10 shows a two-dimensional data space that is partitioned into three clusters before each data point is mapped to a one-dimensional value according to its distance from the reference point of its partition.

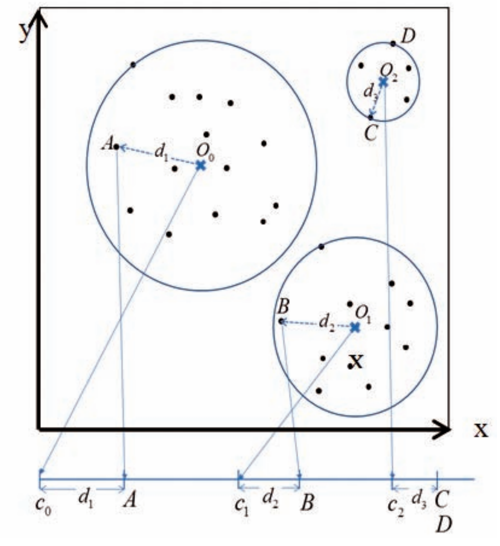
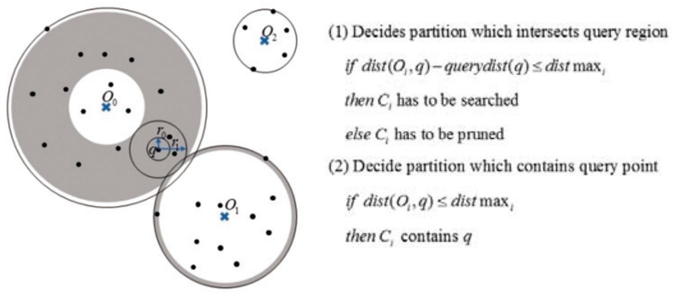


Figure 10. iDistance index.

Figure 11(a) shows how the *k*-NN search algorithm with iDistance works for a data space with three types of partitions. The data space is partitioned into three clusters (C_0, C_1, C_2) and each cluster has a reference point (O_0, O_1, O_2). The first partition, C_0, contains the query point, *q*. The second partition, C_1, has no intersection with the query region for search radius r_0 but intersects the query region when the search radius is increased to r_1. The third partition, C_2, does not intersect the query region. A *k*-NN search in iDistance can shorten the calculation time, because only partitions that intersect the query region are included. Figure 11(b) shows rules that decide the search space. Rule 1 decides which partitions intersect the query region and Rule 2 decides which partition contains the query point.



(1) Decides partition which intersects query region

if $dist(O_i, q) - querydist(q) \le dist\max_i$

then C_i has to be searched

else C_i has to be pruned

(2) Decide partition which contains query point

if $dist(O_i, q) \le dist\max_i$

then C_i contains *q*

(a) **(b)**

Figure 11. (a) *k*-NN search with iDistance and (b) Decision rules for partition type.

5. RESULTS

We conducted an experiment to demonstrate the efficiency of our algorithm using ballistic trajectories simulated by commercial software (PRODAS). The 1,366 trajectories were divided into 2,188,591 sub-trajectories. We assume that radar system senses the position of object at interval of 0.1 s during 3 s. Therefore, each trajectory was represented by 90-dimension variables (3D × 30). When a sub-trajectory was selected as test data and considered a sensed trajectory, the whole trajectory was eliminated from the TD. The accuracy was evaluated by means of the following definitions:

$$Accuracy = \frac{\#of\ error < \varphi^*}{\#of\ testing} \quad (7)$$

where *error* is the distance between estimated launching point and real starting point; φ^* : tolerance < casualty radius

First, to evaluate the quality of the proposed algorithm, we compare the estimation error of the proposed algorithm with that of the AN/TPQ-37 Firefinder mobile radar system. We examined the behavior of our algorithm by varying the number of neighbours from 2 to 50 by increments of 2. In each setting, we sampled 100 sub-trajectories randomly. Next, we calculated the average estimation error. Figure 12 shows that the average estimation error of the proposed algorithm is definitely smaller than the baseline of 105 m (average error of AN/TPQ-37[4]). Moreover, its performance is robust to changes in *k*, once *k* reaches a certain size. These experiments also performed with 100 random samples for each setting. Figure 13 shows the estimation accuracy versus *k* for three different tolerances.

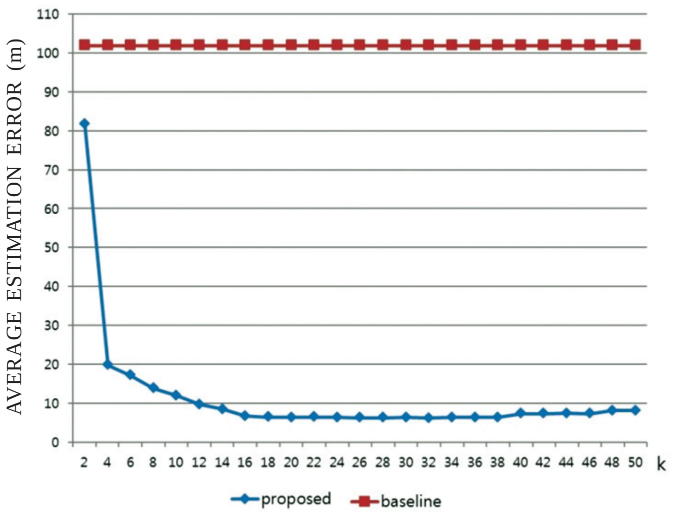


Figure 12. Average estimation error.

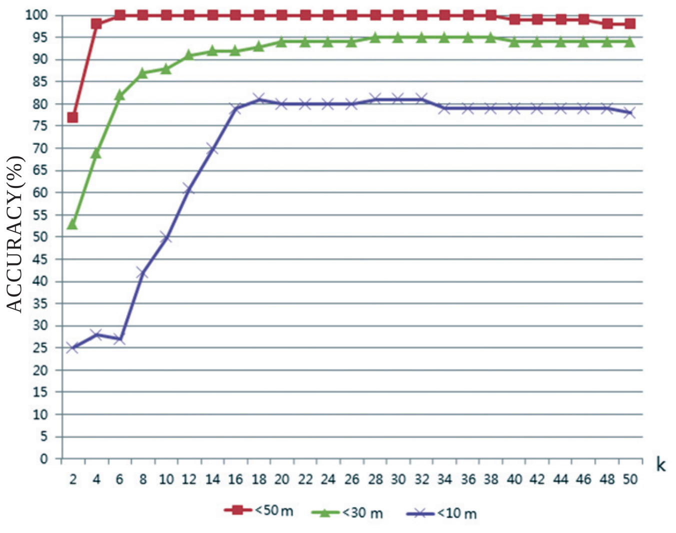


Figure 13. Estimation accuracy by tolerance.

When the tolerance is 50 m, 20 m, or 10 m, the accuracy is about 100%, 90%, or 80%, respectively. Finally, Figure 14 shows histograms of accuracy versus *k* for different estimation error ranges. The two-phase *k*-NN estimation performs most accurately when *k* is 30.

Second, to examine the efficiency of iDistance for a *k*-NN trajectory search, we compared the *k*-NN response time of the full search with that of the iDistance search and checked for agreement of the *k*-NN answer sets. We partitioned the data space into 4,096 clusters by Kmeans clustering and used $\Delta r = 10$ as the default. Figure 15 shows the 30-NN response times of the full search and the iDistance search. The average response times of the iDistance and full searches were 21.7267 ms and 12.188 s, respectively. Hence, combining *k*-NN with iDistance shortens the calculation time by a factor of roughly 560.

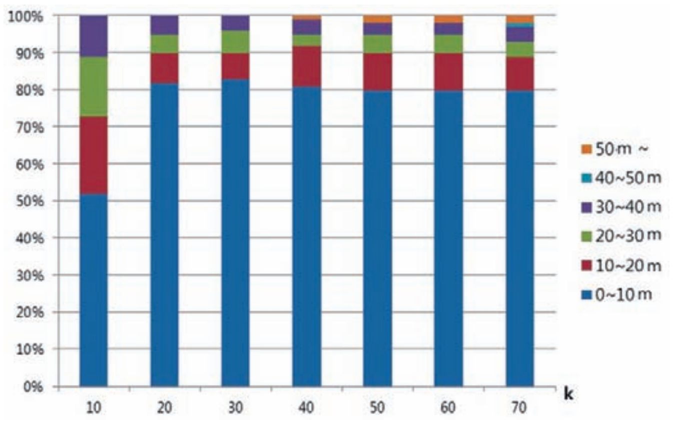


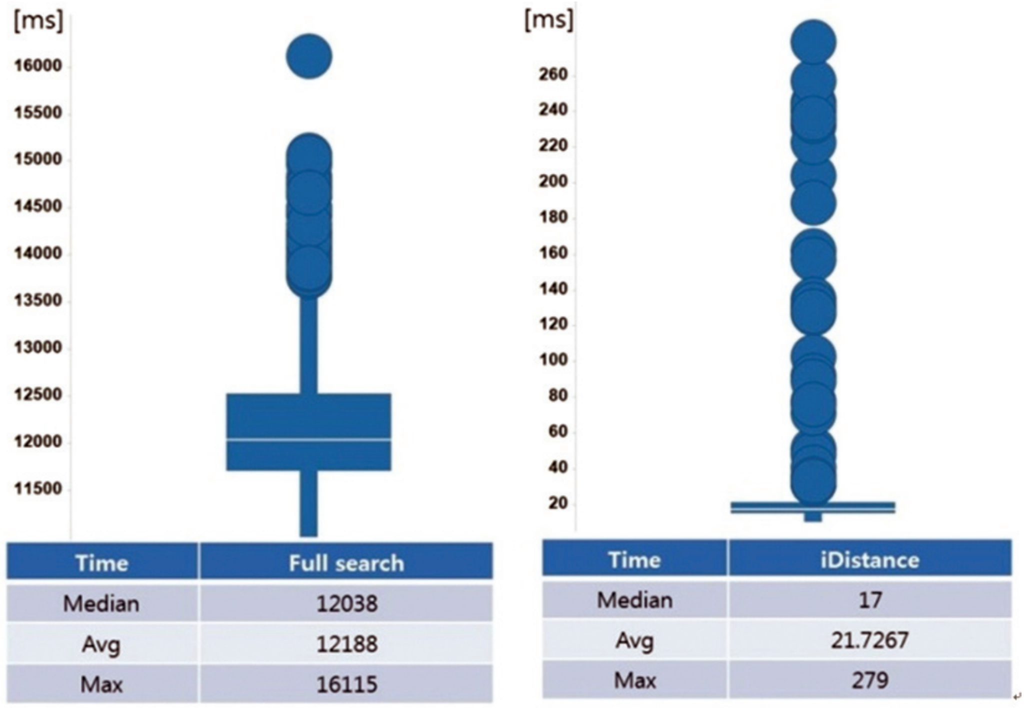
Figure 14. Histogram of two-phase *k*-NN estimation error ranges.

6. CONCLUSION

Authors proposed a novel parameterless data-based estimation approach for ballistic trajectory launch points. The proposed approach uses a simplified trajectory database and *k*-NN estimation. Its average estimation error is definitely smaller than the baseline. Moreover, its performance is robust to changes in *k*, once *k* reaches a certain size. Thus, this algorithm overcomes the greatest difficulty of selecting *k* in *k*-NN estimation. Under an ordinary tolerance, the proposed search algorithm yields almost perfect accuracy. Because we carried out this study with assumed sensing data, we experimented under tolerances much tighter than real operating conditions and we inferred that the accuracy of our algorithm is at least 80%. As ballistic sub-trajectories are high-dimensional correlated data, we used a *k*-NN search with iDistance to reduce the time complexity. The average response time was decreased by more than 22 ms. Our ballistic trajectory launch point estimation approach is highly useful for real-time processes of field artillery, because it offers good performance in terms of both accuracy and speed.

ACKNOWLEDGEMENTS

This work was supported by the second stage of the Brain Korea 21 Project in 2012, the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0030814), and the Engineering Research Institute of SNU.



Time	Full search
Median	12038
Avg	12188
Max	16115

Time	iDistance
Median	17
Avg	21.7267
Max	279

Figure 15. Response times of full search and iDistance search.

REFERENCES

1. Lih, Y.; Kirubarajan, T.; Bar-Shalom, Y. & Yeddanapudi, M. Trajectory and launch point estimation for ballistic missiles from boost phase LOS measurements. *In* the IEEE Proceedings on Aerospace Conference, Snowmass at Aspen, CO, USA, 1999, pp. 425-442.
2. Nelson, E.; Pachter, M.& Musick, S. Projectile launch point estimation from radar measurements. *In* the Proceedings of the American Control Conference, Pasadena, CA, USA, 2005. pp.1275-1282.
3. Jeong, W.-Y.& Cho, S. The estimation method for the starting point of moving objects using the sub trajectory similarity. *In* the International Conference of Data Mining, Phuket, Thailand, 2011, pp.125-127
4. Kim, H.; Kim,S. & Cho, S. Two-Phase *k*-NN on sub-trajectory similarity for efficient ballistic launching point estimation. Industrial Conference on Data Mining, Berlin, Germany, 2012. pp.37
5. Van Kreveld, M. & Luo, J. The definition and computation of trajectory and subtrajectory similarity. *In* the Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems, Seattle, Washington, USA, 2007, **44**. pp.44
6. Pelekis, N.; Kopanakis, I.; Marketos, G.; Ntoutsi, I.; Andrienko, G. & Theodoridis, Y. Temporal Representation and Reasoning. 14th International symposium on IEEE, Alicnate, Spain, 2007, pp.129-140.
7. Frentzos, E.; Gratsias, K.; Pelekis, N. & Theodoridis, Y., Algorithms for nearest neighbour search on moving object trajectories. *Geoinformatica,* 2007**, 11**(2), 159-193.
8. Beckmann, N.; Kriegel, H. P.; Schneider, R. & Seeger, B. The R*-tree: an efficient and robust access method for points and rectangles. *In* the Proceedings of the ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, 1990, **19(2),** pp.322-331.
9. Guttman, A. *R*-trees: A dynamic index structure for spatial searching. *In* the Proceedings of the ACM SIGMOD International conference on Management of data, New York, USA, 1984, **14**(2) pp.47-57.
10. Kamel, I. & Faloutsos, C. Hilbert. R-tree: An improved R-tree using fractals. Institute for Systems Research Technical Reports: 1993.
11. Berchtold, S.; Keim, D.A. & Kriegel, H.P. The X-tree: An index structure for high-dimensional data. *In* the VLDB Conference, Mumbai (Bombay), India, 1996, pp.451
12. An, J.; Chen, Y. P.; Xu, Q. & Zhou, X. A new indexing method for high dimensional dataset. Database Systems for Advanced Applications, Springer Berlin, Heidelberg, 2005, pp.385-397.
13. Zhang, R.; Ooi, B.C. & Tan, K.L. Making the pyramid technique robust to query types and workloads. *In* the Proceedings of the 20th International Conference on Data Engineering, Istanbul, Turkey, 2004, pp.313-324.
14. Berchtold, S.; Böhm, C. & Kriegal, H.P. The pyramid-technique: towards breaking the curse of dimensionality. *In* the ACM SIGMOD Record, 1998, pp.142-153.
15. Yu, C.; Ooi, B. C.; Tan, K.L. & Jagadish, H. Indexing the distance: An efficient method to *k*-NN processing. *In* the Proceedings of the International Conference on Very Large Databases, Zurich, Switzerland, 2001, pp.421-430.

CONTRIBUTORS

Dr Soojin Kim received her Masters degree (Inderstrial Engineering) from KAIST, in 2002 and PhD (Datamining) from Seoul National University, in 2013. Presently, she is working as Army Major in Defense Agency for Technology and Quality.

Mr Hyunjoong Kim recieved his Masters degree of Industrial Engineering from Seoul National University in 2013. Presently, he is doctoral student of Industrial Engineering in Seoul National University.

Prof. Sungzoon Cho received his doctoral degree in computer science at the University of Maryland in College Park with research on machine learning. Presently, he is deputy director of Big Data Center at Seoul National University and an editor of International Journal of Operations Research and Information Systems (IJORIS) and International Journal of Cognitive Biometrics (IJCB).