Vertical-Line-Based Incremental Algorithms for Moving Objects of the Monochromatic and Bichromatic Reverse Nearest Neighbors

Ye-In Chang, Chia-En Li*, Yu-Siang Tseng
National Sun Yat-Sen University/Computer Science and Engineering Department, Kaohsiung, Taiwan.

* Corresponding author. Tel.: +886-7-5254350; email: lice@db.cse.nsysu.edu.tw
Manuscript submitted December 25, 2015; accepted Feb 1, 2016.
doi: ???

Abstract: Due to wireless communication technologies, positioning technologies, and mobile computing develop quickly, mobile services are becoming important on the spatial database management. One of the most important topics in the spatial information query processing is the Reverse Nearest Neighbor query (RNN query). There are two types of the RNN query problem: Monochromatic and Bichromatic RNN queries. The IGERN algorithm can handle both types of the RNN query. In this paper, based on the revised version of the IGERN algorithm, we propose the MQMRNN and BQMRNN algorithms which are used to handle the Monochromatic query and Bichromatic RNN query, respectively. We use a vertical line which is passing through the query to improve the performance. From the simulation results, we show that our proposed algorithms are more efficient than the IGERN algorithm, when data objects are with the uniform distribution.

Key words: mobile service, reverse nearest neighbor, real-time systems, spatial database.

1. Introduction

The spatial database has been used in some location-based systems and Geographic Information Systems (GIS) which are widely used nowadays. One of the most important research topics in the spatial information query processing is the Reverse Nearest Neighbor query (RNN query) which retrieves zero to infinite data objects which treat the query object as their nearest neighbor. There are two types continuous evaluation of RNN queries, namely, monochromatic RNN and bichromatic RNN. In the monochromatic RNN, the query object and the data objects are of the same type [1]. That is, a data object o is considered a RNN to a query object q if there does not exist another data object o' where $dist(o, o') < dist(o, q)$. In the bichromatic RNN, all moving objects and queries are one of two distinct types, (e.g., A and B) [2], [3]. That is, a data object of type B, o_B, is considered a RNN to a query object of type A, q_A, if there does not exist another object of type A, o'_A, where $dist(o_B, o'_A) < dist(o_B, q_A)$. The RNN query on the static environments means that both of the query object and data objects are all static [4]-[6].

The difficulties of solving the RNN query problem on the dynamic environments are how to index the large data with the high moving frequency in the storage, and how to use a good algorithm which can find the accurate answers of the dynamic RNN query problem without costing too much time. Kang *et al*. proposed an incremental algorithm called IGERN to answer the dynamic RNN query problem [7]. It makes use of the bisectors between the query object and related data objects to reduce the number of candidates. Therefore, it is more efficient for the dynamic RNN query than the Voronoi diagram [8] and the six pies principle [9]. However, whenever the query object moves, it always reconstructs all of bisectors. In fact, some of bisectors are not necessary to be reconstructed.

The rest of the paper is organized as follows. Section II gives a survey of some algorithms of RNN query

problems. Section III, presents the proposed algorithms, the MQMRNN algorithm and the BQMRNN algorithm. In Section IV, we give the performance of the proposed algorithm and make a comparison between our algorithms and the IGERN algorithm. Finally, we give a conclusion.

2. Related Work

In this section, we introduce some algorithms for RNN query processing. Those algorithms include the approach in spatial databases [7], [8]. The IGERN algorithm maintains a grid data structure G of $N \times N$ equal size cells. Each cell $c \in G$ keeps track of the set of objects that lie within the cell boundary. In general, the IGERN algorithm has two main steps, namely, the initial and incremental steps [7].

2.1. Step 1: Getting the Initial Answer

The initial step of the IGERN algorithm has three main objectives: (1) Obtaining a bounded region r around the query object q and the Grid index G which will be monitored in the incremental step, (2) Identifying a set of objects *RNNcand* that need to be monitored in the incremental step, and (3) Identifying the set of initial reverse nearest neighbor objects ($RNN \subseteq RNNcand$) to the query object q. Initially, *RNNcand* is empty, while all grid cells in the grid data structure G are set as *alive*. Then, the initial step has the following two main phases: bounded region and verification.

2.2. Step 2: Incremental Maintenance

The incremental step of the IGERN algorithm is repetitively executed at each time unite for all time intervals T. The input is the query object q, the set *RNNcand* that came from either the initial step at time T_0 or a previous execution of the incremental step, and the Grid index G. Upon its execution, the incremental step checks for three different scenarios: (1) The query object q moves to a new location, (2) At least one of the objects in *RNNcand* moves to a new location, and (3) A new object moves into the *alive* cells. However, if any of these three scenarios take place, then the IGERN algorithm incremental step needs to perform more computations in order to efficiently maintain the query answer.

3. The MQMRNN and The BQMRNN Algorithms

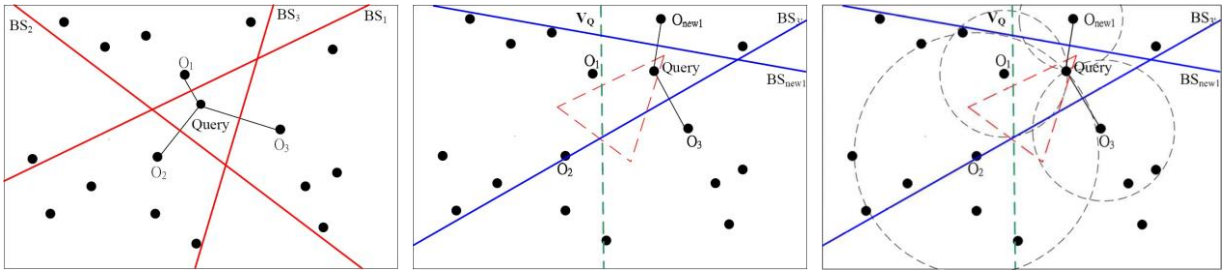
For the RNN query algorithms, how to get the answers of the RNN query efficiently, incrementally and correctly is really important. In this section, we propose new algorithms to solve the incremental RNN query problem. We use the Hilbert curve to help us to find the NN of the query [10]. We mainly focus on the incremental parts of the algorithms and we propose two algorithms named MQMRNN (Monochromatic Query Moving Reverse Nearest Neighbor) and BQMRNN (Bichromatic Query Moving Reverse Nearest Neighbor). We mainly separate the algorithms into two parts, *i.e.*, the initial step and the incremental step. The initial step is used to get the original answers of the RNN query and the incremental step is used to handle the situation of the data objects in the case of the data set move.

3.1. Processing of the MQMRNN Algorithm

The IGERN algorithm [7] is an event-based algorithm. There are some drawbacks of the IGERN algorithm. In the beginning, we introduce one of the most important concepts which we will use in our algorithm, *i.e.*, bisector. In the initial step, the main goal is to bound the *alive* and the *dead* region which are produced by the bisectors. There are three moving cases of the data objects. The first case is that query object q moves to a new location. The second case is that one of the candidates which we choose in the initial step moves to a new location. The third case is that the data objects which do not include query object q and the candidates move in the *alive* region.

Case 1: Fig. 1(a) represents the original situation of query object q. As shown in Fig. 1(b), query object q

moves to the right side of the plane, then we draw a vertical line named V_Q. Because query object q moves to the right side, we can filter out the data objects except original candidates whose location are at the left side of the vertical line V_Q. Then, we draw a new bisector $BS_{3'}$ between query object q and data object O_3 and get a new *alive* region. There are still some data objects in the *alive* region so we still need to find the NN of query object q (*i.e.*, O_{new1}) and draw a new bisector BS_{new1}. We keep doing such actions until there is no data objects in the new *alive* region. In the example, we have four candidates (*i.e.*, O_1, O_2, O_3, and O_{new1}). As shown in Fig. 1(c), there are another data objects in the circles whose centers are O_1, O_2, and O_3, so these three candidates will not be the answer of query object q. However, we discovery that there is no data object in the circle whose centre is O_{new1}. As a result, O_{new1} is the answer of query object q.

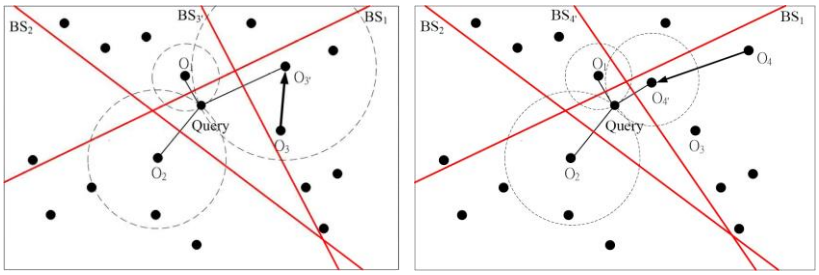


(a) (b) (c)

Fig. 1. The right-moving case of the query object: (a) the original location of the query object; (b) the new location of the query object; (c) the verification of the moving case of the query object.

Case 2: As shown in Fig. 1(a) and Fig. 2(a), data object O_3 moves to another location (denoted as $O_{3'}$), then we draw a new bisector $BS_{3'}$ and we get a new *alive* region. We find out that there is no data object in the new *alive* region so that we can execute the verification part directly. There is no data object in the circle whose centre is data object O_1. Therefore, data object O_1 is the answer of query object q.

Case 3: As shown in Fig. 1(a) and Fig. 2(b), we have an original *alive* region which is constructed by bisectors BS_1, BS_2, and BS_3. After data object O_4 moves into this *alive* region (*i.e.*, data object $O_{4'}$), we get a new *alive* region which is constructed by bisectors BS_1, BS_2, $BS_{4'}$. We discover that data object O_3 will no longer be candidate because its location is in the new *dead* region. We find out that there is no data object in the new *alive* region so that we can execute the verification part directly. As a result, data objects O_1 and $O_{4'}$ are the answers of query object q.



(a) (b)

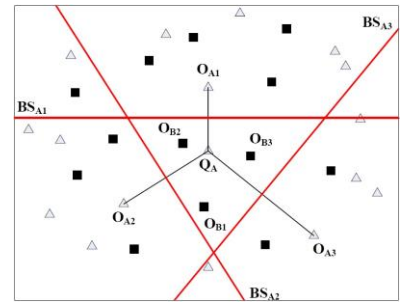
Fig. 2. The moving case: (a) the verification of the moving case of the candidates; (b) the verification of the moving case of the non-candidates.

3.2. The Processing of the BQMRNN Algorithm

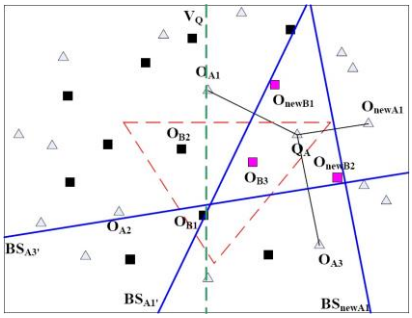
We regard such a query object q as the origin of the coordinate. That is, if the value of the X axis of every data object is greater than 0, we put these data objects into one group. Conversely, if the value of the X axis of every data object is less than 0, we put these data objects into another group. In the BQMRNN algorithm, we do not need to consider about whether candidates will be the answers of query object q or not because

candidates and query object q are of the same type (*i.e.*, type A). There are three moving cases of the data objects. The first case is that query object q moves to a new location. The second case is that one of the candidates of type A which we choose in the initial step moves to a new location. The third case is that the data objects which do not include query object q and the candidates of type A move in the *alive* region.

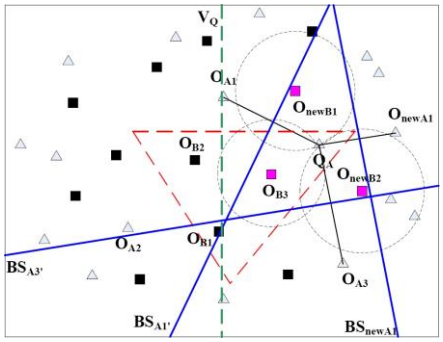
Case 1: Fig. 3(a) represents the original situation of query object q. As shown in Fig.3(b), query object q moves to the right side of the plane, then we draw a vertical line named V_{Q}. Because query object q moves to the right side, we can filter out the data objects whose location are at the left side of the vertical line V_{Q}. Then, we draw two new bisector $BS_{\mathrm{A1'}}$ and $BS_{\mathrm{A3'}}$. There are still some data objects of type A in the *alive* region so we still need to find the NN of type A of query object q (*i.e.*, O_{newA1}) and draw a new bisector BS_{newA1}. We keep doing such actions until there is no data objects of type A in the new *alive* region. As shown in Fig. 3(c), we discover that there is no data object of type A in the circles whose centres are O_{B3} and O_{newB1} so that these two data objects of type B are answers of query object q. However, we still find out there are some data objects of type A in the circle whose centre is O_{newB2}. Consequently, O_{newB2} is not the answer of query object q.



(a)



(b)



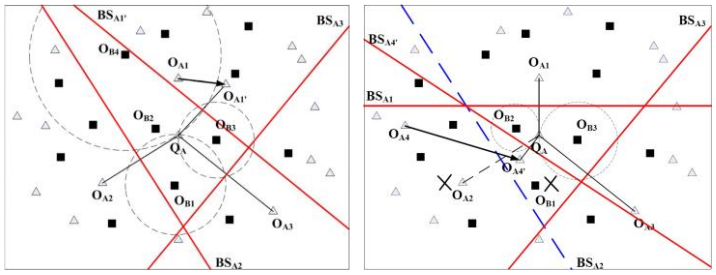
(c)

Fig. 3. The right-moving case of the query object: (a) the original location of the query object; (b) the new location of the query object; (c) the verification of the moving case of the query object.

Case 2: As shown in Fig. 3(a) and Fig. 4(a), data object O_{A1} (*i.e.*, candidate O_{A1}) moves to another location (denoted as $O_{\mathrm{A1'}}$), then we draw a new bisector $BS_{\mathrm{A1'}}$ and we get a new *alive* region. We find out that there is no data object in the new *alive* region so that we can execute the verification part directly. The data object O_{B2} is guaranteed to be the answer of query object q. We only discover that there is some data objects of type

A in the circle whose centre is O_{B4}. As a result, data objects O_{B1}, O_{B2}, and O_{B3} are answers of query object q.

Case 3: As shown in Fig. 3(a) and Fig. 4(b), we have an original *alive* region which is constructed by bisectors BS_{A1}, BS_{A2}, and BS_{A3}. After data object O_{A4} moves into this *alive* region (*i.e.*, data object $O_{A4'}$), we get a new *alive* region which is constructed by bisectors BS_{A1}, BS_{A3}, $BS_{A4'}$. We discover that data object of type A, O_{A2}, will no longer be candidate of type A, because its location is in the new *dead* region. Furthermore, we also discover that data object O_{B1} is in the new *dead* region, so we do not need to verify it. We find out that there is no data object of type A in the new *alive* region so that we can execute the verification part directly. The data objects O_{B2} and O_{B3} are answers of query object q.



(a) (b)

Fig. 4. The moving case: (a) the verification of the moving case of the candidates; (b) the verification of the moving case of the non-candidates.

4. Performance

In this section, we present the results of our experimental study. We make a comparison between our algorithm and the IGERN algorithm. We execute the experiments under the condition of an AMD Phenom(tm)II X4 965 Processor 3.4GHz with 4GB memory, Windows 7 Professional version, and coded in JAVA.

4.1. The Performance Model

We compare the two algorithms in term of CPU time. We focus on the number of data objects (denoted as *DN*) and the times of the moving of the data objects/query (denoted as *Inc*) in 2-D space, then to compute the CPU time. The CPU time means the time to compute the whole process of getting the answers of the RNN query problem. A summary on the ranges of the parameters used in this simulation is described as follows.

- *DN*: the number of data objects in database. The range is from 6000 to 18000.
- *Inc*: the number of Incremental execution. The range is from 20 to 70.
- *RatioDoverS*: the ratio of data objects located at two half planes.
- *BiRatio*: the ratio between the numbers of type A and type B data objects.

4.2. Experiment Results

In this section, we show the experimental results. We have different kinds of data sets as shown in Table I. For Case 1 and Case 2, we focus on the change of the number of data objects (*DN*). For Case 3 and Case 4, we focus on the change of the times of the incremental execution (*Inc*).

Table 1. Cases of Performance Graphics

Case	Uniform data set	Higher density of data objects in a half plane	Cumulative data objects	Cumulative incremental number
1	✓	✗	✓	✗
2	✗	✓	✓	✗
3	✓	✗	✗	✓
4	✗	✓	✗	✓

From Fig. 5(a), we show that the processing time of our MQMRNN algorithm is faster than that of the

IGERN algorithm. However, as the number of data objects increases, the growth rate of the processing time of our algorithm is slower than that of the IGERN algorithm. The reason is that the IGERN algorithm always needs to check all data objects while our algorithm only need to check barely a half of all data objects on the average. For instance, when the number of data objects is 6000/8000, the IGERN algorithm needs to check 6000/8000 times in the query moving case. But our algorithm only needs to check 3000/4000 times.

From Fig. 5(b), we show that the processing time of our MQMRNN algorithm is faster than that of the IGERN algorithm. Moreover, the growth rate of the processing time of our algorithm is slower than that of the IGERN algorithm. The reason is the same as the Case 1. However, the difference of the checked data objects of our algorithm is much smaller than that of the IGERN algorithm when we execute the query moving case.

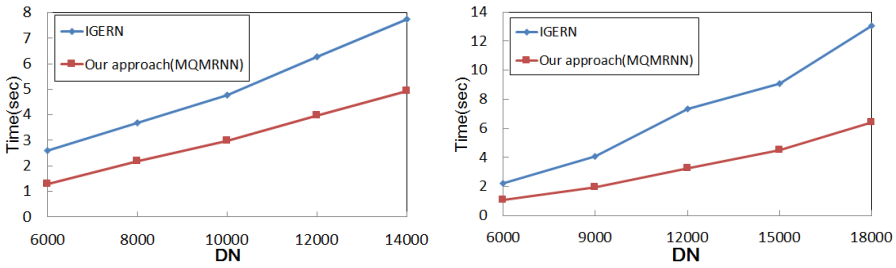


Fig. 5. A comparison of the processing time: (a) Case 1, *Inc* = 10; (b) Case 2, *Inc* = 10, *RatioDoverS* = 2.

Fig. 6(a) and Fig. 6(b) show the simulation results of the case that the number of data objects is fixed (*DN* = 12000). We can see that the performance of our MQMRNN algorithm is better than that of the IGERN algorithm. The processing time of both algorithms increases and the reason is the same as the one explained before.

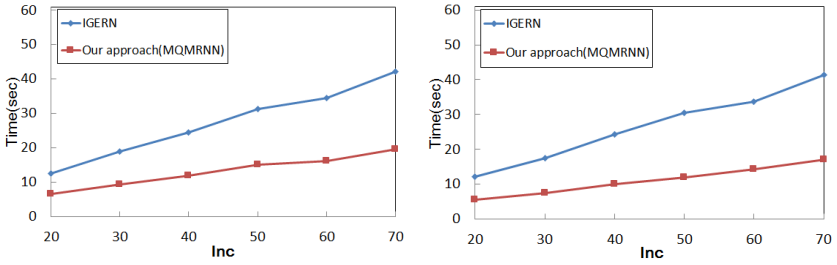


Fig. 6. A comparison of the processing time: (a) Case 3, *DN* = 12000; (b) Case 4, *DN* = 12000, *RatioDoverS* = 2.

Fig. 7(a) and Fig. 7(b) show the simulation results of using the data set of Case 1 (with *BiRatio* = 1) and the data set of Case 3 (with *BiRatio* = 1), respectively. The processing time of our BQMRNN algorithm is still faster than that of the IGERN algorithm. The reasons that our BQMRNN algorithm can provide such good performance are the same as the ones of Case 1 and Case 3 in the monochromatic type, respectively.

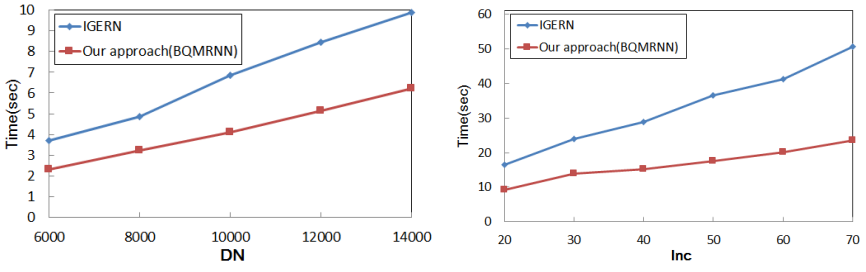


Fig. 7. A comparison of the processing time: (a) Case 1, *Inc* = 10, *BiRatio* = 1; (b) Case 3 , *DN* = 12000, *BiRatio*

= 1.

5. Conclusion

In this paper, we have proposed the MQMRNN algorithm and the BQMRNN algorithm to solve the monochromatic and the bichromatic RNN query problem, respectively. We use a vertical line which is passing through the query point to improve the performance efficiently. We have studied the performance of our proposed MQMRNN and BQMRNN algorithms and made a comparison with the IGERN algorithm by simulation. The results have shown that our algorithms have better performances than the IGERN algorithm.

Acknowledgment

This research was supported in part by the Ministry of Science and Technology of Republic of China under Grant No. MOST 104-2221-E-110-077.

References

[1] Wu, W., Yang, F., Chan, C. Y., & Tan, K. L. (2008). Continuous reverse k-Nearest-Neighbor monitoring. *Proceedings of the 9th International Conference on Mobile Data Management*, (pp. 132 – 139).

[2] Nghiem, T. P., Maulana, K., Nguyen, K., Green, D., Waluyo, A. B., & Taniar, D. (2014). Peer-to-Peer bichromatic reverse nearest neighbours in mobile Ad-Hoc networks. *Journal of Parallel and Distributed Computing*, *74(11),* 3128 – 3140.

[3] Tran, Q. T., Taniar, D., & Safar, M. (2010). Bichromatic reverse nearest-neighbor search in mobile systems. *IEEE Systems Journal*, 4(2), 230 – 242.

[4] Cheema, M., Lin, X., Zhang, W., & Zhang, Y. (2011). Influence zone: efficiently processing reverse k-Nearest neighbors queries. *Proceedings of the 27th International Conference on Data Engineering*, (pp. 577 – 588).

[5] Luo, Y., Lian, C., Liu, J., & Chen, H. (2008). Finding RkNN by compressed straightforward index. *Proceedings of the 3rd International Conference on Intelligent System and Knowledge Engineering*, (pp. 279 – 284).

[6] Singh, A., Ferhatosmanoglu, H., & Tosun, A. S. (2003). High dimensional reverse nearest neighbor queries. *Proceedings of the 20th International Conference on Information and Knowledge Management*, (pp. 91 – 98).

[7] Kang, J., Mokbel, M., Shekhar, S., Xia, T., & Zhang, D. (2010). Incremental and general evaluation of reverse nearest neighbors. *IEEE Transactions on Knowledge and Data Engineering, 22(7),* 983 – 999.

[8] Bohan, L. & Xiaolin, Q. (2009). Research on reverse nearest neighbor queries using ranked voronoi diagram. *Proceedings of the 1st International Conference on Information Science and Engineering*, (pp. 951 – 955).

[9] Xia, T. & Zhang, D. (2006). Continuous reverse nearest neighbor monitoring. *Proceedings of the 22nd International Conference on Data Engineering*, (pp. 77 – 86).

[10] Kang, J., Mokbel, M., Shekhar, S., Xia, T., & Zhang, D. (2007). Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. *Proceedings of the 23rd International Conference on Data Engineering*, (pp. 806 – 815).

Ye-In Chang was born in Taipei, Taiwan, R.O.C. in 1964. She received her B.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1986. She received her M.S. and Ph.D. degrees in computer and information science from Ohio State University, Columbus, Ohio, in 1987 and 1991, respectively. From August1991 to July 1999, she joined the faculty of the Department of Applied Mathematics at National Sun Yat-Sen University, Kaohsiung, Taiwan. From August 1997, she has been a professor in the Department of Applied Mathematics at National Sun Yat-Sen University, Kaohsiung, Taiwan. Since August 1999, she has been a professor in the Department of Computer Science and Engineering at National Sun Yat-Sen University, Kaohsiung, Taiwan. Her research interests include database systems, distributed systems, multimedia information systems, mobile information systems and data mining.

Chia-En Li received his B.S. degree in information management from I-Shou University in 2007 and his M.S. degree in Computer Science from National Pingtung University of Education in 2009. He is currently a Ph.D. student in Department of Computer Science and Engineering at National Sun Yat-Sen University. His research interests include database systems, data mining and risk factor analysis.

Yu-Siang Tseng received her B.S. degree in computer science and engineering from National Sun Yat-Sen University in 2011, and his M.S. degree in computer science and engineering from National Sun Yat-Sen University in 2013. He is currently a system design engineer in Taiwan.