applied sciences

MDPI

Article

Utilizing Ensemble Learning to Improve the Distance Information for UWB Positioning

Che-Cheng Chang *, Yee-Ming Ooi, Shih-Tung Tsui, Ting-Hui Chiang and Ming-Han Tsai

Department of Information Engineering and Computer Science, Feng Chia University, Taichung City 407, Taiwan
* Correspondence: checchang@fcu.edu.tw; Tel.: +886-4-24517250 (ext. 3764)

Abstract: An ultra-wideband (UWB) positioning system consists of at least three anchors and a tag for the positioning procedure. Via the UWB transceivers mounted on all devices in the system, we can obtain the distance information between each pair of devices and further realize the tag localization. However, the uncertain measurement in the real world may introduce incorrect measurement information, e.g., time, distance, positioning, and so on. Therefore, we intend to incorporate the technique of ensemble learning with UWB positioning to improve its performance. In this paper, we present two methods. The experimental results show that our ideas can be applied to different scenarios and work well. Of note, compared with the existing research in the literature, our first algorithm was more accurate and stable. Further, our second algorithm possessed even better performance than the first. Moreover, we also provide a comprehensive discussion for an ill-advised point, which is often used to evaluate the positioning efficiency in the literature.

Keywords: positioning system; distance information; ensemble learning; machine learning

check for updates

Citation: Chang, C.-C.; Ooi, Y.-M.; Tsui, S.-T.; Chiang, T.-H.; Tsai, M.-H. Utilizing Ensemble Learning to Improve the Distance Information for UWB Positioning. *Appl. Sci.* **2022**, *12*, 9614. https://doi.org/10.3390/app12199614

Academic Editors: Cristina Portalés Ricart, João M. F. Rodrigues and Pedro J. S. Cardoso

Received: 18 August 2022
Accepted: 22 September 2022
Published: 25 September 2022

1. Introduction

Sensors play significant roles in several modern applications [1–5]. Diverse sensors help us obtain various information for different applications. For instance, we use a depth camera to capture stereo vision and then calculate the corresponding depth information [4]. This is a method that requires line of sight (LOS) to obtain the distance information between the object and sensor(s). On the other hand, an ultra-wideband (UWB) positioning system is a non-line-of-sight (NLOS) example. Its architecture estimates the distance information as well as the location of an object in a specific space [4,5]. More specifically, via the UWB transceivers mounted on all devices in the system, the distance between each pair of devices can be obtained. Upon receiving enough distance information at a specific time interval, the object localization can be realized [5].

However, for practical applications, the uncertainty of the measurement may introduce incorrect measurement information, e.g., time, distance, positioning, and so on. Hence, in order to deal properly with uncertainty in measurement in practice, research have considered the UWB positioning issue by incorporating various machine learning approaches [5]. Their experimental results show some important and interesting properties. In particular, they refine the raw data before applying the classical trilateration; then, more accurate and stable positioning information is obtained.

According to the discussions presented in [5], the anchors with identical specification report various distance values even with the same real distance between themselves and the tag in the same scenario at the same time. Therefore, training distinct machine learning models for different pairs of anchors and tags is practical and better. Accordingly, we will use this concept in this work. On the other hand, after collecting enough data, the procedure in [5] enters the next phase, i.e., the learning phase. In this phase, all training data are refined first by sorting the data from large to small and then removing the largest and smallest 10% data, the authors claim that this filtering method helps them remove some extreme data and works well. Here, we have a question about these thresholds: are they the best threshold values for their filtering method? Hence, we will apply the concept of ensemble learning to set the thresholds independently and automatically and also present a comprehensive discussion with regard to different thresholds. Note that ensemble learning is a critical and interesting concept, which does not directly modify the original models, but combines several weak and independent models in order to achieve a stronger one [6,7]. In this paper, it is adopted to cascade machine learning algorithms.

Last but not least, the authors in [5] utilize the positioning error to conduct their evaluation. However, the positioning information and error are calculated by a trilateration, which may implicitly include some error-tolerant or optimization design, such as the digit-by-digit calculation. Obviously, a fairer evaluation mechanism is to evaluate the distance information. Consequently, in this paper, to avoid such a situation, we will consider the distance error to conduct our evaluation instead of the positioning error.

This paper is organized as follows: first, some preliminaries related to our research are reviewed in Section 2. Then our methods and the experimental results corresponding to different settings and scenarios are presented and discussed in Sections 3 and 4. Finally, this research is concluded in Section 5, and we suggest some feasible future work as well.

2. Preliminaries

2.1. Problem Formulation

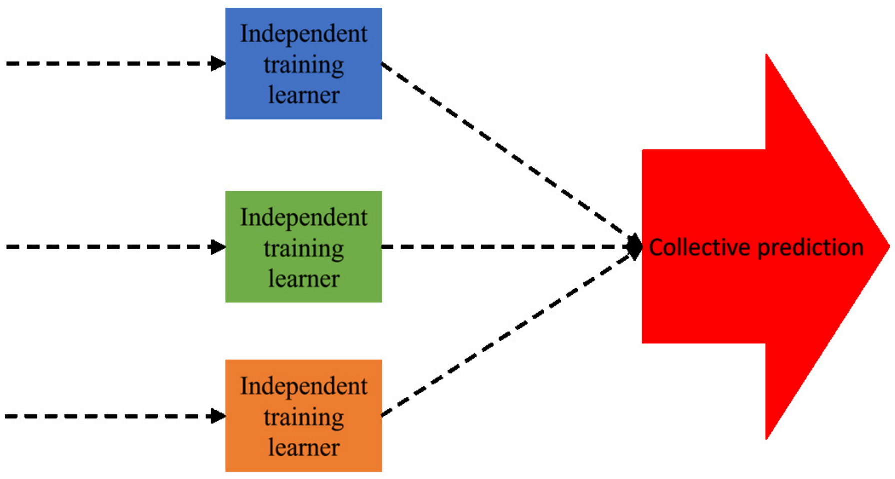
In the UWB positioning architecture, anchors can measure the distance from the tags to themselves via some radio ranging approaches. The measured distance can be expressed as follows:

$$d_{measure} = d_{real} + d_{noise}, \tag{1}$$

where the noise part is assumed as a white Gaussian noise generally. However, in [5], the authors illustrate that the white Gaussian noise may not fit the uncertain measurement well in practice. Thus, we need some novel ideas for overall consideration instead of merely using a deterministic process to simulate the uncertainty.

2.2. Ensemble Learning

Ensemble learning is growing in popularity due to its variability and efficiency [8,9]. Ensemble learning is composed of multiple weaker models that are independently trained; then, the collective prediction will be more accurate. The ensemble approach can simultaneously consist of several learners applying unsupervised algorithms, supervised algorithms, and so forth, in distinct ways, e.g., sequence, parallel, and mixture. Figure 1 shows a simple instance of ensemble learning with the parallel type.



Independent training learner
Independent training learner
Independent training learner
Collective prediction

Figure 1. An example of ensemble learning with parallel type.

2.3. *k-Means Clustering*

Clustering is a way bto group data into disjoint sets without labeling training data beforehand and is one of the most common unsupervised algorithms [4,8,9]. Moreover, classifying data into several disjoint sets should be purposeful. k-means clustering is one of the most utilized manners; it can group elements into k separate clusters for $k \geq 2$.

The pseudocode is abstracted below [10]:

- Place k centroids $c_1, c_2, \ldots, c_k$ randomly;
- For each point, which is not a centroid, find the nearest centroid and assign it to the cluster where the nearest centroid belongs;
- After assigning each point to its cluster, find a new centroid for each cluster;
- Repeat steps 2 and 3 until a predefined number of iterations or convergence.

k-means clustering is very simple, with low computational complexity.

2.4. *Regression Analysis*

Regression analysis is used to estimate the relationship between the dependent variable and independent variable(s) [11–13]. The most common form is linear regression. In particular, if a linear line fitting the data closely can be found in an analysis, we can make a prediction via calculating the weighted sum of the input features and bias term. The concept is shown in the following equation:

$$\hat{y} = a_n x_n + a_{n-1} x_{n-1} + \ldots + a_0, \tag{2}$$

where $\hat{y}$ is the prediction, n is the number of features, x_i the ith feature, and a_j the jth parameter.

Polynomial regression is another form, where the relationship between the dependent variable and independent variable(s) is as a nth degree polynomial for $n > 1$ [13]. In particular, if the analysis is more complicated than a linear line style, the power of each feature is added as a new feature. Thus, we can train this modified model with these extended features.

2.5. *Artificial Neural Network*

An Artificial Neural Network (ANN) is a machine learning model inspired by the networks of biological neurons found in animals' brains. It is powerful, scalable, and versatile. Due to its characteristics, it can be collocated with other approaches to tackle large and highly complex tasks, e.g., classifying images, speech recognition, recommending videos, and so on [13].

One of the simplest ANN architectures is Perceptron, which is based on artificial neurons. For each neuron, the inputs and output are numbers, and each input is associated with a weight and an extra bias feature may be added. Then, the weighted sum is applied to an activation function to output the result. Furthermore, Multi-Layer Perceptron (MLP) is an improvement of Perceptron. It is composed of one input layer, one or more hidden layers constructed from the concept of Perceptron, and one output layer. Thus, every layer except the output layer is fully connected to the next layer [13]. The computation of Perceptron is shown in the following equations:

$$y_i = x_n w_n + x_{n-1} w_{n-1} + \ldots + x_1 w_1 + x_0 w_0 + bias_i, \tag{3}$$

$$z_i = act_i(y_i), \tag{4}$$

where x_i is the input, w_i is the weight of x_i, $bias_i$ is the bias, y_i the weighted sum, and z_i the output produced from the activation function $act_i()$.

Here, most importantly, MLP can be applied to complex nonlinear problems. In [5], the authors illustrated that the white Gaussian noise may not fit the uncertain measurement well in practice. Thus, they utilized the concept of linear and polynomial regression to refine the distance information. However, it is obvious that the refinement of the UWB positioning topic is not a linear or simple polynomial issue due to the effects of signal attenuation and multipath propagation [14,15]. Hence, this is another motivation for us to design a new architecture including ANN to refine the distance information further.

2.6. The Original Design

The main concept in [5] is that in practice, the white Gaussian noise may not fit the uncertain measurement for UWB positioning well. Hence, instead of utilizing a deterministic model to evaluate the uncertainty, they utilized several machine learning methods for overall consideration. The following Figure 2 shows the operating procedure presented in [5].

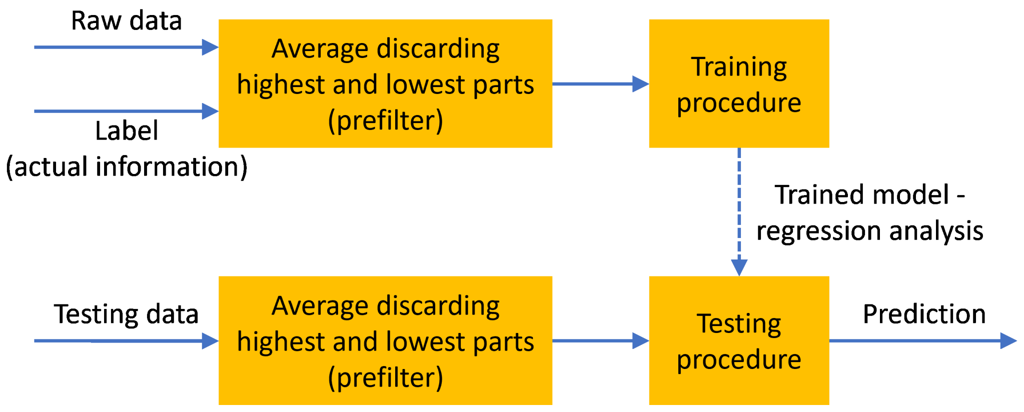

Raw data
Label
(actual information)
Average discarding highest and lowest parts (prefilter)
Training procedure
Trained model - regression analysis
Testing data
Average discarding highest and lowest parts (prefilter)
Testing procedure
Prediction

Figure 2. The refinement for UWB positioning in [5].

First, in the data collection procedure, for each reference point, they initialized the locations of the sole tag tag and anchor $anchor_i$. Then, they collected the real distance between tag and $anchor_i$ as well as the corresponding distance provided by the UWB positioning system, i.e., $real\_dis(tag, anchor_i)$ and $sys\_dis(tag, anchor_i)$. Hence, the actual distance and its inaccurate version provided by the UWB positioning system were obtained as the training data.

Next, the prefilter was used to discard the highest and lowest parts from the raw data and then average them before entering the training procedure. After the training procedure was finished, this trained model was utilized in the testing procedure to evaluate the performance, including accuracy and stability. Here, the trained model was only regression analysis [5].

3. Our Architecture and Algorithm

The uncertain measurement in the real environment may introduce incorrect information. Generally speaking, the noise part is often assumed as a white Gaussian noise. However, the white Gaussian noise may not fit the uncertain measurement well in practice. Thus, we need some novel ideas for overall consideration instead of merely using a deterministic process to simulate the uncertainty.

Now, we begin to detail our algorithm, which is shown in Algorithm 1. Foremost, $anchor_i$ and tag represent the devices with UWB transceivers in the experiment (line 1). Then we have some data sets, which are set to ϕ initially, i.e., $DATA$, $TRAINING$, and $TESTING$ (lines 2–5). Moreover, we define several functions (lines 6–9):

- **initialize**(x) is used to put device x at a specific location for the purpose of obtaining training and testing data;
- **sys_dis**(x, y) is utilized to return the distance information between device x and device y provided by the UWB system;
- **real_dis**(x, y) is utilized to return the actual distance information between device x and device y;

- **prefilter**(a) is a method to discard unreasoned parts from data a and then return the residual part.

Algorithm 1. The improved algorithm for *tag* in a specific scenario.

```
Variable:
(01) $anchor_i$ and tag are devices with UWB transceivers;
(02) $DATA$, $TRAINING$ and $TESTING$ are data sets;
(03) $DATA \leftarrow \emptyset$;
(04) $TRAINING \leftarrow \emptyset$;
(05) $TESTING \leftarrow \emptyset$;
Function:
(06) initialize($x$): put device $x$ at a specific location;
(07) sys_dis($x$, $y$): the distance between device $x$ and device $y$ provided by the UWB system;
(08) real_dis($x$, $y$): the actual distance between device $x$ and device $y$;
(09) prefilter($a$): a method to discard unreasoned parts from data $a$;
Collecting phase:
(10) for all combinations of $anchor_i$ and $tag$ do
(11)     initialize($anchor_i$);
(12)     initialize($tag$);
(13)     for $1 \le x \le n$ do
(14)         $DATA \leftarrow DATA \cup$ (sys_dis($anchor_i$, $tag$), real_dis($anchor_i$, $tag$));
(15) separate $DATA$ into $TRAINING$ and $TESTING$;
Learning phase:
(16) $TRAINING \leftarrow$ prefilter($TRAINING$);
(17) train the model via $TRAINING$;
Evaluating phase:
(18) $TESTING \leftarrow$ prefilter($TESTING$);
(19) evaluate the trained model via $TESTING$;
```

In the collecting phase, for all combinations of anchor $anchori_i$ and the tag tag, we initialized the locations of $anchor_i$ and tag (lines 10–12). Then, the information about the distance between the $anchor_i$ and the tag provided by the UWB system and the real distance between the $anchor_i$ and the tag are stored, i.e., **sys_dis**(x, y) and **real_dis**(x, y). This means that we obtained the actual distance and its inaccurate version provided by the UWB system as the training/testing data. Here, for the reliability of entire procedure, each combination was gathered n times (lines 13–14). Particularly, if there are ℓ anchors and m combinations in a scenario, we collected $\ell \times m \times n$ pairs of distance information as the training/testing data. Finally, data set $DATA$ was separated into data set $TRAINING$ and $TESTING$ for the training and testing procedure (line 15).

Next, we appled the concept of ensemble learning to set the thresholds independently and automatically to remove the extreme unreasoned data. Namely, $TRAINING$ was filtered via an independent machine learning model first (line 16); then, we trained the refinement model via $TRAINING$ (line 17). Note that the former model was unsupervised learning, since we have no idea about the thresholds of unreasoned data in advance, and the latter one for the purpose of refinement was supervised learning. Lastly, in the evaluating procedure (lines 18–19), $TESTING$ was also filtered via an independent machine learning model first and then used to evaluate the performance via the existing trained model.

Figures 3 and 4 illustrate the main differences between [5] and our two improved versions. In Figure 3, the first refinement, k-means was introduced into the architecture for the purpose of setting the thresholds independently and automatically to remove the extreme unreasoned data. Next, as shown in Figure 4, was the second refinement; since MLP can be applied to complex nonlinear problems, this was another motivation for us to introduce ANN into the architecture to refine the distance information further.

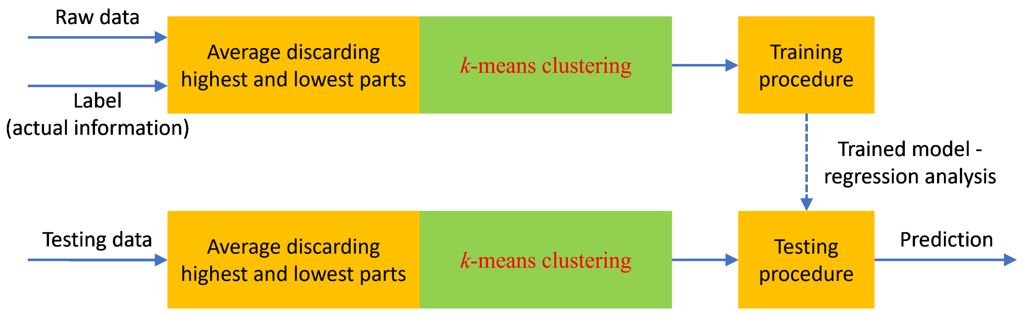


Figure 3. Our first refinement for UWB positioning.

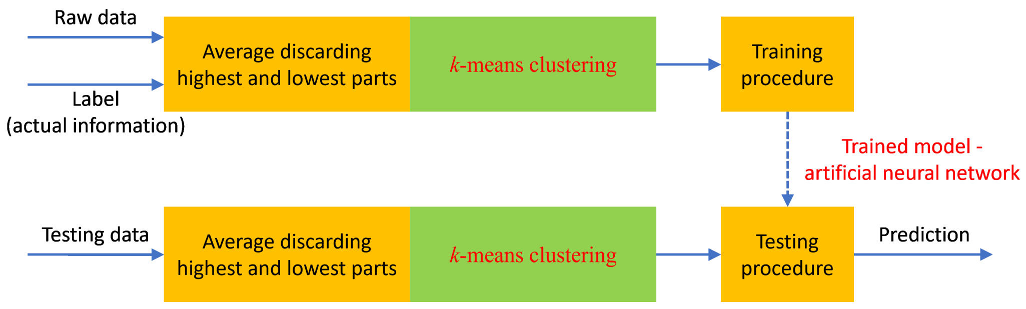

Raw data
Label
(actual information)
Average discarding highest and lowest parts
k-means clustering
Training procedure
Trained model - artificial neural network
Testing data
Average discarding highest and lowest parts
k-means clustering
Testing procedure
Prediction

Figure 4. Our second refinement for UWB positioning.

4. Experimental Results

Here, we detail our experiments, and all relevant settings are listed as follows:

- In the experiments, three scenarios were considered, i.e., classroom, parking garage, and parking lot (shown in Figure 5);
- Four anchors were set at fixed positions (red points);
- Twenty-seven points were utilized to collect the data: training data (blue points) and testing data (green points);
- For every point, 100 continuing distance values were collected;
- The complete location information is shown in Figure 6;
- The UWB transceiver device utilized in the experiments was DWM1000 [16].
- The DWM1000 integrated the antenna, the Radio Frequency (RF), clock circuitries, and the power management module. It used the Time Difference of Arrival (TDOA) method to obtain the raw distance data between each pair of UWB devices [17].
- Google Colab [18] allowed us to write and execute Python 3 in the browser with the power of a Graphics Processing Unit (GPU).
- The tools, scikit-learn [19] and Keras [20], were used to realize all the machine learning algorithms in this paper.

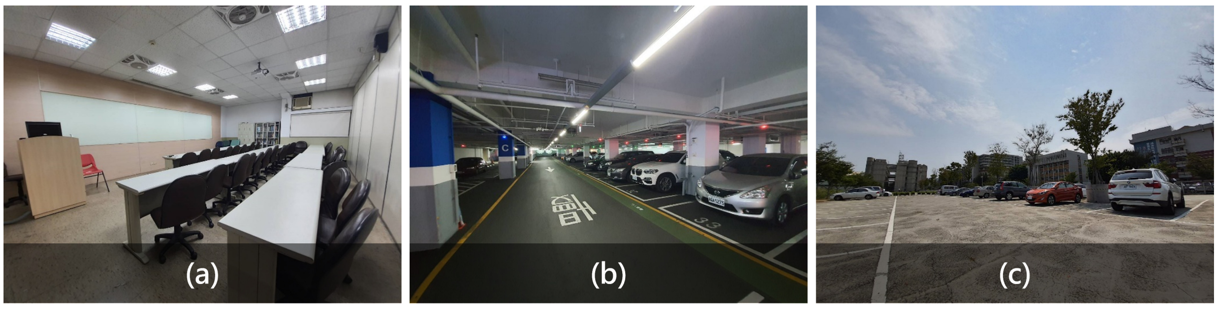


Figure 5. Three scenarios are considered in this paper: (**a**) classroom, (**b**) parking garage, and (**c**) parking lot.

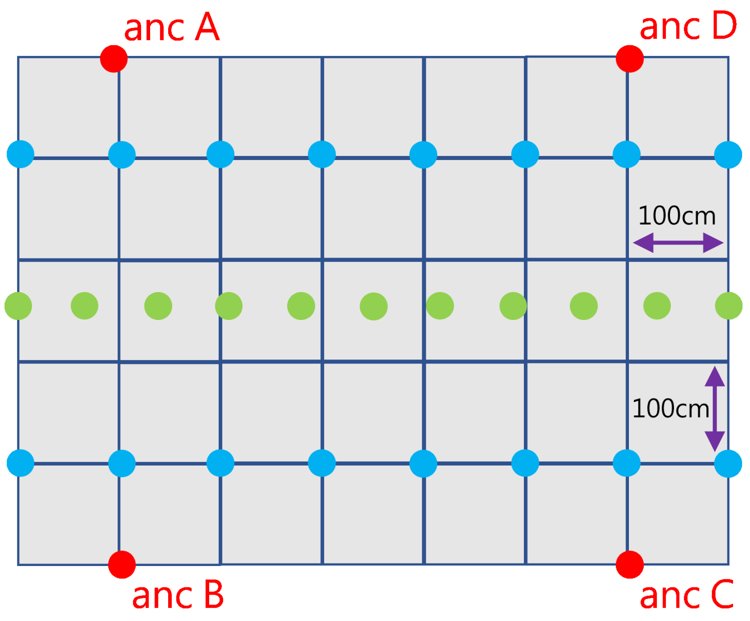


Figure 6. The complete location information.

Next, different prefilters and machine learning models were applied in our experiment accordingly:

- Prefilters:
  - MaxAve: We removed the largest 10% and smallest 10% and then only considered the average of the cluster with the most elements for k-means clustering ($k = 3$) as the training data;
  - MaxCen: We removed the largest 10% and smallest 10% and then only considered the median of the cluster with the most elements for k-means clustering ($k = 3$) as the training data;
  - CenAve: We removed the largest 10% and smallest 10% and then only considered the average of the central cluster for k-means clustering ($k = 3$) as the training data;
  - CenCen: We removed the largest 10% and smallest 10% and then only considered the median of the central cluster for k-means clustering ($k = 3$) as the training data;
  - 10%: We removed the largest 10% and smallest 10% and then only considered the average of the residual part as the training data;
- Machine learning approaches:
  - LR: Linear Regression;
  - Third order PR: Third order Polynomial Regression;
  - Fifth order PR: Fifth order Polynomial Regression;
  - Tenth order PR: Tenth order Polynomial Regression;
  - ANN:
    - Model: sequential model;
    - Activation function: rectified linear unit;
    - Optimizer: adam;
    - Loss function: mean absolute error;
    - The architecture is shown in Figure 7.

Next, the experimental results are shown in the following tables. First, the higher order polynomial regression, i.e., the tenth order polynomial regression, caused overfitting for all the experiments, which was consistent with that in [5]. Hence, we will not discuss this in this paper.

According to Tables 1 and 2, we considered our methods with these new prefilters to work well. More specifically, we observed that if the experimental space had a higher density of obstacles (classroom), using the ANN refinement with the MaxCen prefilter was better for accuracy (error) and stability (standard deviation). Then, for the space with a lower density of obstacles (parking lot), using the ANN refinement was also better. Note that since all standard deviation values were lower in the parking lot, we merely considered the viewpoint of error.

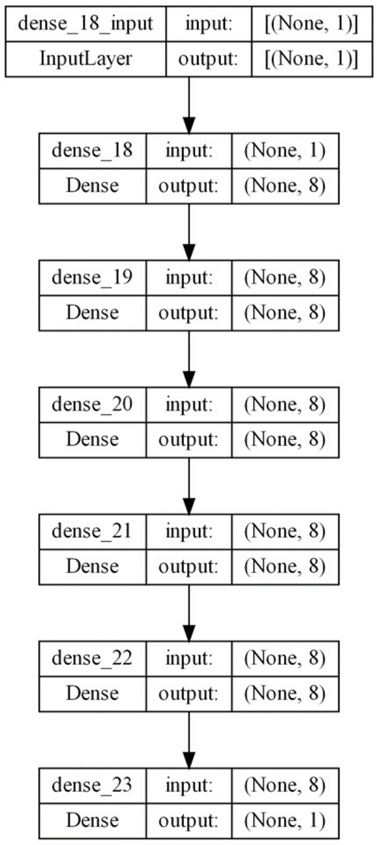


Figure 7. The ANN architecture for the refinement in the experiments.

These results met all three declarations in the beginning of this paper. Namely, in [5], all training data were refined first by sorting the data from large to small and then removing the largest and smallest 10% of data; the authors claimed that this filtering method helped them to remove some extreme unreasoned data and worked well. However, they were not the best threshold values. We applied the *k*-means prefilters to improve the performance, which can be observed via the differences among the same refinement algorithm with various prefilters.

Second, it is obvious that the refinement of the UWB positioning topic is not a linear or simple polynomial issue due to the effects of signal attenuation and multipath propagation. Since MLP can be applied to complex nonlinear problems, the introduction of ANN helps us improve the performance further for the viewpoints of accuracy (error) and stability (standard deviation), which can be observed via the differences among the distinct refinement algorithms. This conforms with the characteristics of ANN as well.

For the last declaration in the beginning of this paper: the authors in [5] utilized the positioning error to conduct their evaluation. However, the positioning information and error was calculated by a trilateration, which may implicitly include some error-tolerant or optimization design. Here, some results about the distance and positioning error are provided in Table 3 to address this viewpoint, where the trilateration algorithm was realized via invoking the *solve* function provided by SymPy [21,22].

Table 1. The experimental results about accuracy.

Scene	Classroom					Parking Garage					Parking Lot				
Filtering algorithm	MaxAve	MaxCen	CenAve	CenCen	10%	MaxAve	MaxCen	CenAve	CenCen	10%	MaxAve	MaxCen	CenAve	CenCen	10%
Average error (cm) for LR refinement	13.4840	13.5016	12.9833	12.9509	13.1802	13.1247	13.1195	13.2806	13.3082	13.2374	5.5774	5.5771	5.2287	5.2522	5.2791
Average error (cm) for 3rd order PR refinement	14.7483	14.6830	14.1591	14.1242	14.4526	13.7075	13.7177	14.3688	14.3690	14.3312	7.0026	6.9436	6.7687	6.7979	6.8600
Average error (cm) for 5th order PR refinement	16.2367	16.2464	16.0984	16.0911	16.1965	14.1433	14.0594	14.1278	14.1456	14.1356	7.7564	7.7698	7.6098	7.6510	7.7056
Average error (cm) for ANN refinement	13.9084	12.7475	14.0147	13.3212	14.1257	12.7099	12.5266	12.8038	11.9679	12.6255	6.2953	6.2102	5.3152	4.9925	6.0157

Table 2. The experimental results about stability.

Scene	Classroom					Parking Garage					Parking Lot				
Filtering algorithm	MaxAve	MaxCen	CenAve	CenCen	10%	MaxAve	MaxCen	CenAve	CenCen	10%	MaxAve	MaxCen	CenAve	CenCen	10%
Standard deviation (cm) for LR refinement	4.9911	5.0357	4.4573	4.4348	4.4121	11.2675	11.3058	11.1558	11.1844	11.2332	1.7559	1.7165	1.6711	1.6744	1.6475
Standard deviation (cm) for 3rd order PR refinement	7.4616	7.3711	6.5284	6.5000	6.6460	10.4364	10.4098	11.8883	11.8629	11.7947	1.4641	1.4836	1.7100	1.7309	1.6900
Standard deviation (cm) for 5th order PR refinement	6.4702	6.4784	6.1868	6.2237	6.3537	11.4380	11.4833	11.8216	11.7886	11.6871	1.4185	1.5199	1.6037	1.6316	1.5720
Standard deviation (cm) for ANN refinement	4.2665	4.4225	3.1471	3.2531	3.6112	9.8709	10.0507	9.3053	9.7154	9.2710	1.4387	0.6929	1.4117	1.5468	1.1412

Table 3. The example to show the ill-advised point of error estimation.

Example	Distance Error between Tag and Anchor A (cm)	Distance Error between Tag and Anchor B (cm)	Distance Error between Tag and Anchor C (cm)	Distance Error between Tag and Anchor D (cm)	Positioning Error (cm)
LR refinement with 10% prefilter	8.5619	14.7145	13.1708	3.1046	19.2614
Raw data	38.7411	33.6501	36.0461	54.3153	16.7028

According to the results shown in the example in Table 3, since the trilateration algorithm had some error-tolerant design, and even though all the distance errors between the tag and anchors in the raw data row were higher, it still possessed a lower positioning error. Apparently, this may cause the misjudgment of the algorithm efficiency. Consequently, a fairer evaluation mechanism is to evaluate the distance error instead of the positioning error.

5. Conclusions and Future Work

In this paper, we reconsidered the UWB positioning problem by incorporating ensemble learning in the real world. According to the experimental results, our method performed better than the existing one. More specifically, *k*-means was introduced into the architecture for the purpose of setting the thresholds independently and automatically to remove the extreme unreasoned data, and ANN was introduced into the architecture to refine the distance information further.

Of note, in this work, the density of obstacles was applied to assess what refinement and filtering algorithms may be better options for a specific scenario. However, the system needs to be retrained for a new scenario, and there may be more environment conditions that should be considered in practice. Hence, for future work, we intend to integrate more heterogeneous machine learning algorithms and the corresponding hyperparameters to automatically refine the UWB raw data to obtain more stable and accurate information according to various and objective environment conditions.

Likewise, solving the positioning issues with regard to various kinds of technologies is also significant, e.g., Global Positioning System (GPS), Wi-Fi, Dedicated Short Range Communications (DSRC), cellular network, and so forth.

Author Contributions: Formal analysis, C.-C.C., Y.-M.O. and S.-T.T.; Methodology, C.-C.C., Y.-M.O., S.-T.T., T.-H.C. and M.-H.T.; Software, Y.-M.O. and S.-T.T.; Writing—original draft, C.-C.C.; Writing—review & editing, C.-C.C., T.-H.C. and M.-H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by National Science and Technology Council, Taiwan, R.O.C. under grant 109-2221-E-035-067-MY3 and 109-2622-H-035-001-.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chang, C.-C.; Tsai, J.; Lin, J.-H.; Ooi, Y.-M. Autonomous Driving Control Using the DDPG and RDPG Algorithms. *Appl. Sci.* **2021**, *11*, 10659. [CrossRef]
2. Chang, C.-C.; Lin, W.-M.; Lai, C.-A. Using Real-Time Dynamic Prediction to Implement IoV-Based Collision Avoidance. *Appl. Sci.* **2019**, *9*, 5370. [CrossRef]
3. Xu, Q.; Wang, B.; Zhang, F.; Regani, D.S.; Wang, F.; Liu, K.J.R. Wireless AI in Smart Car: How Smart a Car Can Be? *IEEE Access* **2020**, *8*, 55091–55112. [CrossRef]
4. Chang, C.-C.; Shih, K.-C.; Ting, H.-C.; Su, Y.-S. Utilizing Machine Learning to Improve the Distance Information from Depth Camera. In Proceedings of the 2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 29–31 October 2021.
5. Chang, C.-C.; Wang, H.-W.; Zeng, Y.-X.; Huang, J.-D. Using Machine Learning Approaches to Improve Ultra-Wideband Positioning. *J. Internet Technol.* **2021**, *22*, 1021–1031. [CrossRef]
6. Opitz, D.; Maclin, R. Popular Ensemble Methods: An Empirical Study. *J. Artif. Intell. Res.* **1999**, *11*, 169–198. [CrossRef]
7. Zhu, Z.; Wang, Z.; Li, D.; Zhu, Y.; Du, W. Geometric Structural Ensemble Learning for Imbalanced Problems. *IEEE Trans. Cybern.* **2020**, *50*, 1617–1629. [CrossRef] [PubMed]
8. Hwang, K. *Cloud Computing for Machine Learning and Cognitive Applications*; The MIT Press: Cambridge, UK, 2017.
9. Hwang, K.; Chen, M. *Big-Data Analytics for Cloud, IoT and Cognitive Computing*; Wiley: Hoboken, NJ, USA, 2017.

10. K-Means Clustering-Introduction to Machine Learning Algorithms | by Rohith Gandhi | Towards Data Science. Available online: https://towardsdatascience.com/k-means-clustering-introduction-to-machine-learning-algorithms-c96bf0d5d57a (accessed on 1 August 2022).
11. Illukkumbura, A. *Introduction to Regression Analysis*; Amazon Digital Services: Seattle, WA, USA, 2020.
12. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; The MIT Press: Cambridge, UK, 2018.
13. Geron, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*; O'Reilly Media: Sebastopol, CA, USA, 2019.
14. Dionisio-Ortega, S.; Rojas-Perez, L.O.; Martinez-Carranza, J.; Cruz-Vega, I. A Deep Learning Approach towards Autonomous Flight in Forest Environments. In Proceedings of the 2018 International Conference on Electronics, Communications and Computers (CONIELECOMP), Cholula, Mexico, 21–23 February 2018; pp. 139–144.
15. Maximov, V.; Tabarovsky, O. Survey of Accuracy Improvement Approaches for Tightly Coupled ToA/IMU Personal Indoor Navigation System. In Proceedings of the International Conference on Indoor Positioning and Indoor Navigation, Montbeliard, France, 28–31 October 2013.
16. DWM1000 Module-Decawave. Available online: https://www.decawave.com/product/dwm1000-module/ (accessed on 1 August 2022).
17. DWM1000 Datasheet. Available online: https://datasheet.lcsc.com/lcsc/1809291612_DecaWave-DWM1000_C95238.pdf (accessed on 1 August 2022).
18. Colaboratory. Available online: https://colab.research.google.com/ (accessed on 1 August 2022).
19. scikit-learn: Machine Learning in Python-scikit-learn 1.0 Documentation. Available online: https://scikit-learn.org/stable/ (accessed on 1 August 2022).
20. Keras Documentation. Available online: https://keras.io/ (accessed on 1 August 2022).
21. Solvers-SymPy 1.10.1 Documentation. Available online: https://docs.sympy.org/latest/modules/solvers/solvers.html (accessed on 1 August 2022).
22. Rocklin, M.; Terrel, A.R. Symbolic Statistics with SymPy. *Comput. Sci. Eng.* **2012**, *14*, 88–93. [CrossRef]