


Article

RBFNN Design Based on Modified Nearest Neighbor Clustering Algorithm for Path Tracking Control

Dongxi Zheng ^{1,2}, Wonsuk Jung ^{3,*}  and Sunghoon Kim ^{1,4,*}

¹ Department of Electronics Convergence Engineering, Wonkwang University, Iksan 54538, Korea; zhengdongxi1979@wku.ac.kr

² School of Mechanical and Intelligent Manufacturing, Jiujiang University, Jiujiang 332005, China

³ School of Mechanical Engineering, Chungnam National University, Daejeon 34134, Korea

⁴ Wonkwang Institute of Material Science and Technology, Wonkwang University, Iksan 54538, Korea

* Correspondence: wonsuk81@cnu.ac.kr (W.J.); kshoon@wku.ac.kr (S.K.); Tel.: +82-63-850-6739 (S.K.)

Abstract: Radial basis function neural networks are a widely used type of artificial neural network. The number and centers of basis functions directly affect the accuracy and speed of radial basis function neural networks. Many studies use supervised learning algorithms to obtain these parameters, but this leads to more parameters that need to be determined, thereby making the system more complex. This study proposes a modified nearest neighbor-based clustering algorithm for training radial basis function neural networks. The calculation of this clustering algorithm is not large, and it can adapt to varying densities. Furthermore, it does not require researchers to set parameters based on experience. Simulation proves that the clustering algorithm can effectively cluster samples and optimize the abnormal samples. The radial basis function neural network based on modified nearest neighbor-based clustering has higher accuracy in curve fitting than the conventional radial basis function neural network. Finally, the path tracking control based on a radial basis function neural network of a magnetic microrobot is investigated, and its effectiveness is verified through simulation. The test accuracy and training accuracy of the radial basis function neural network was improved by 23.5% and 7.5%, respectively.



Citation: Zheng, D.; Jung, W.; Kim, S. RBFNN Design Based on Modified Nearest Neighbor Clustering Algorithm for Path Tracking Control. *Sensors* **2021**, *21*, 8349. <https://doi.org/10.3390/s21248349>

Keywords: radial basis function neural network (RBFNN); nearest neighbor-based clustering (MNNC); sample optimization; path tracking

Received: 29 November 2021
Accepted: 12 December 2021
Published: 14 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Path tracking control is a commonly used motion control method for vehicles and robots. Owing to its simple structure, easy operation and adjustment, and robustness, the proportion integral differential (PID, as shown in Appendix A) controller is often used for path tracking control [1,2]. However, the ability of PID in dealing with nonlinear systems is limited. Therefore, fuzzy PID control was developed. B.B. Ghosh et al. developed a fuzzy-PID-based controller to control the two degrees of freedom parallel manipulator. The control system has almost no overshoot based on the fuzzy-PID [3]. J.A. Algarin-Pinto et al. compared the fuzzy-PID with general PID for path tracking control of biomimetic autonomous underwater vehicles. The experiment results showed that path tracking control error with general PID was over 9%, but with fuzzy-PID was less than 2% [4]. T.A. Mai et al. applied fuzzy PID in path-following control of a nonholonomous mobile robot. Under the control system based on fuzzy-PID, the distance error of path-following control could be reduced from 0.172 m to 0.041 m [5]. Nonetheless, fuzzy rules require strong prior knowledge. Due to the time-varying dynamics, nonlinear uncertainty of the control object, and environmental interference, it is extremely difficult to control the high-precision path tracking for the linear state observer because it is difficult for the linear state observer to compensate errors of the nonlinear system [6]. The previous methods are incapable of addressing these issues. Although a sliding mode controller could control the trajectory

tracking of a nonlinear system [7,8], it occasionally caused a large lateral acceleration in the trajectory tracking using the sliding mode control method.

To deal with nonlinear systems, C. Liu et al. proposed a nonlinear adaptive controller based on PID [9]. B. Smeresky et al. discussed a deterministic artificial intelligence-instantiated method for a nonlinear system, which stems from a lineage of nonlinear adaptive control [10]. Compared with these methods, artificial neural networks have attracted increasing interest from researchers because they do not require complex modeling process or powerful processing, and have adaptive capabilities in constantly changing and noisy environments. Utilizing the learning ability of an artificial neural network facilitates improved flexibility of controller design, particularly when the dynamics of the controlled object are complex and highly non-linear [11]. Radial basis function neural networks (RBFNN) have the advantages of fast learning convergence speed and strong approximation ability; they have been used in finite-time trajectory tracking control of n-link robotic manipulators [12], longitudinal speed tracking of autonomous vehicles [13], trajectory tracking for a robotic helicopter [14], and tracking control of a nonholonomic wheel-legged robot in complex environments [15]. In these cases, the control systems based on RBFNN showed good accuracy and stability.

Before running an RBFNN, it is necessary to determine the relevant parameters, such as the type and number of basis functions, the center and the width of the basis functions, and the weight of each hidden layer neuron. These parameters affect not only the learning time, but also the controller performance [16,17]. To optimize the relevant parameters of an RBFNN, supervised learning or unsupervised learning methods can be used. In supervised learning, other intelligent algorithms are introduced to optimize the parameters of the RBFNN. F. Fernandez-Navarro et al. investigated performance of an RBFNN based on support vector machines (VSM). The parameters of VSM should be defined [18]. H.C. Huang et al. presented an evolutionary radial basis function neural network with genetic algorithm (GA) and artificial immune system (AIS) for tracking control of autonomous robots. Although the controller based on a GAAIS-RBFNN showed better performance than the controller based on an individual genetic algorithm and artificial immune system, GAAIS-RBFNN involved more variables to be decided [19]. Z.Y. Chen et al. trained the RBFNN by particle swarm optimization and genetic algorithm. The RBFNN showed good learning performance, but the algorithm was more complex [20]. When using unsupervised learning to design and optimize the parameters of an RBFNN, the clustering algorithm is a commonly used method that can speedily converge and avoid overfitting. A. Guillén et al. developed a clustering algorithm with a possibilistic partition to get the initial center of hidden layer neurons of an RBFNN. The algorithm showed better robustness than other general RBFNNs [21]. S.K. Oh et al. applied a k-means clustering algorithm in setting the center of hidden layer neurons of RBFNN; the algorithm showed good accuracy [22]. C.C. Liao et al. introduced an RBFNN-based control system for tracking the maximum power point of a photovoltaic system. The parameters of RBFNN were determined by the modified k-means clustering algorithm. The experiment results proved the tracking method was effective [23].

However, most clustering algorithms need to determine some parameters in advance; for example, k-means requires the number of clusters and initial center of cluster to cluster the samples; density-based spatial clustering of applications with noise (DBSCAN) requires the radius of the scan and the minimum number of samples of the cluster for clustering; clustering by fast search and find of density peaks clustering (DPC) requires the threshold of distance. These parameters are extremely important and affect the results of clustering significantly, but it is necessary for users to determine and adjust the parameters based on experience, which is difficult. Moreover, common clustering algorithms often require iteration or a large calculation that reduces the efficiency of the clustering algorithm. To avoid these issues, we propose a modified nearest neighbor-based clustering (MNNC) algorithm according to the characteristics of curve fitting and path following control datasets. Unlike other clustering algorithms, MNNC clusters the samples referring to the

distance between the sample and its nearest neighbor. It is easy to utilize this algorithm because it requires fewer parameters to be determined. Furthermore, MNNC improves the approach to searching for neighbors, thus it does not require iteration and requires less calculation which increases the efficiency of clustering. We evaluated the clustering results using accuracy (ACC) and adjusted Rand index (ARI); the simulation results show that ACC and ARI using MNNC are 20% and 10% higher than the common clustering algorithms, respectively. MNNC can also detect and optimize the outlier samples, and the simulation results show that the optimization of outlier samples can decrease the curve fitting errors by 10–50%. MNNC was used to set the initial parameters of RBFNN that can automatically adjust the number of hidden layer nodes according to the accuracy requirements. In particular, we applied the proposed method to a path tracking simulation of a spiral-type magnetic microrobot to generate a rotating magnetic field (RMF) to reach the desired position. Consequently, the proposed method featured 20 % lower error than conventional RBFNN.

The remainder of this study is organized as follows. Section 2 introduces the concept of RBFNN based on MNNC for path tracking. Section 3 describes a novel clustering algorithm that is applied in optimizing the samples in Section 4. Section 5 introduces the control system for the path tracking of magnetic microrobot and develops MNNC to train RBFNN for path tracking. Finally, the discussion and conclusion are presented in Sections 6 and 7, respectively.

2. Concept of RBFNN Algorithm Based on MNNC for Path Tracking

Figure 1 illustrates the proposed RBFNN algorithm based on MNNC for path tracking. The entire algorithm consists of three parts: MNNC, RBFNN, and path tracking. The MNNC is used to obtain the initial parameter of RBFNN and optimize the training samples of RBFNN. First, MNNC characterizes sample P_i by distance (d_i , more variables are shown in Appendix B) between P_i and its nearest neighbor distance and classifies samples with similar d_i into one class. The datasets of curve-fitting and path-following control have obvious temporal or spatial order characteristics; when searching for the nearest neighbor, the search range can be reduced by improving the searching direction in order to reduce the calculation of the clustering algorithm, as shown in Figure 1a, where d_i is the minimal value of d_{i1} and d_{i2} . Thereafter, the MNNC can detect and optimize the abnormal samples, as shown in Figure 1b. P_a is defined as an abnormal sample because P_a corresponds to the longest distance d_a , and there are no similar samples around P_a . We construct a triangle by P_a , P_{a1} , and P_{a2} , where P_{a1} and P_{a2} are the neighbors of P_a . Next, we obtain P_c , which is the center of the triangle, and replace P_a by P_c to optimize the training samples of RBFNN. MNNC can cluster datasets with different densities and shapes without specifying the number of clusters or scanning radius in advance, as shown in Figure 1c. Each cluster is displayed in a different color, and the cluster center is represented by blue circles. We construct the RBFNN based on MNNC. Each cluster corresponds to a hidden layer node, and the center of the cluster is the center of the node, as shown in Figure 1d. Thereafter, the RBFNN based on MNNC can be used to establish the relationship between the theoretical direction and reference direction of the magnetic microrobot's locomotion. Therefore, if we obtain the theoretical driving direction of each step and input the theoretical direction into the RBFNN, the RBFNN outputs the reference direction. We can calculate the coil currents according to the reference direction, and accordingly, the microrobot moves along the theoretical direction driven by the magnetism generated by the coils. Finally, we can realize path tracking of the magnetic microrobot through the proposed method shown in Figure 1e.

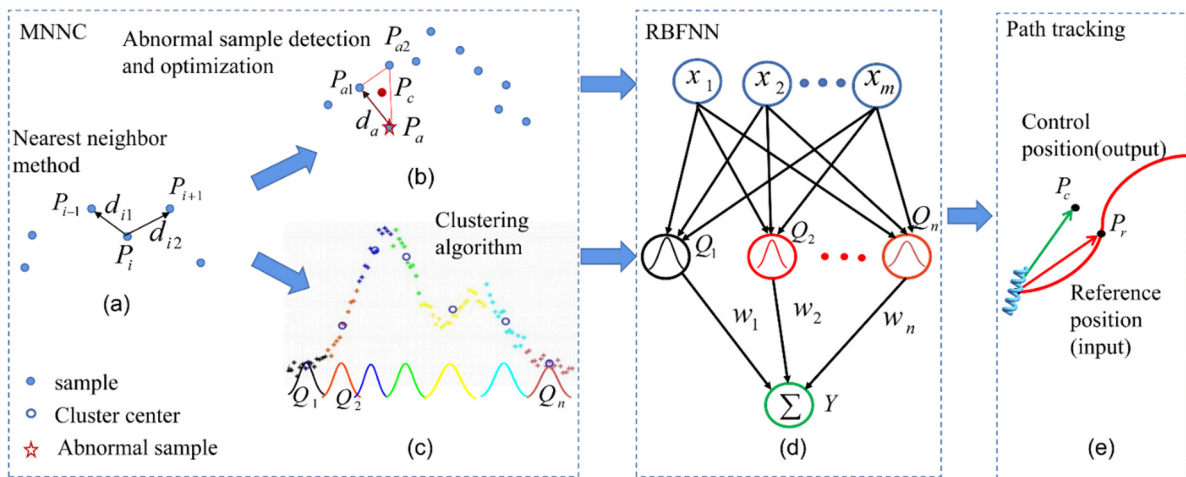


Figure 1. Concept of an RBFNN algorithm based on MNNC for path tracking: (a) calculation of the distance to obtain the nearest neighbor; (b) detection of the abnormal samples based on nearest neighbor method; (c) clustering of the samples based on the nearest neighbor; (d) construction of the RBFNN based on MNNC; (e) obtaining the reference position by the MNNC-based RBFNN in the path tracking system.

3. Clustering Algorithm Based on Nearest Neighbor

3.1. Typical Clustering Algorithm

A clustering algorithm is a typical unsupervised learning algorithm that is mainly used to automatically classify similar samples into a specific category. The main clustering methods can be divided into five methods: the partitioning method, hierarchical method, grid-based method, model-based method, and density-based method [24]. The partitioning method decomposes the data into n clusters, such that the items in each cluster are closely related to each other, for example, K-means algorithm. The calculation procedure in this algorithm is simple, but it is necessary to know the number of clusters of data in advance [25].

Balanced iterative reducing and clustering using hierarchies (BIRCH) is a typical representative of the hierarchical method that decomposes a given dataset hierarchically until a certain condition is met. Specifically, it can be categorized into “bottom-up” and “top-down” schemes [26]. This algorithm is not particularly suitable for non-convex datasets, and owing to the limit on the number of each node, the clustering result may deviate from the actual classification.

Clustering in QUES (CLIQUE) is a clustering algorithm based on the grid method. In this algorithm, the data space is first divided into a grid structure of finite units, and all processing is based on a single unit. This algorithm is highly sensitive to parameters and cannot handle irregularly distributed data [27]. There is no iteration required in this method, but it is difficult to determine the density threshold, an important parameter in the algorithm. Model-based methods set a model for each cluster, and subsequently detect a dataset that satisfies this model adequately. Such a model may be the density distribution function of data points in space or other. The efficiency of this algorithm also needs to be improved [28]. The model-based method incorporates the probability and statistics approach and the neural network approach.

Density-based methods attempt to determine the high-density clusters separated by sparse regions. The size and shape of these clusters may be different. The most commonly used clustering algorithm based on density is DBSCAN. Although this algorithm does not necessitate knowledge of the number of classes the data is divided into in advance, knowledge of the radius and the minimum number of points is required [29].

3.2. Modified Nearest Neighbor-Based Clustering Algorithm for Training RBFNN

When we train the RBFNN for curve fitting and path tracking to obtain the structure parameters, clustering the training data to determine the center and number of basis functions of the hidden layer is an effective approach. The dataset in this case features obvious time or space characteristics. Here, we propose a simple clustering algorithm MNNC that clusters the samples according to the distance (d_i) between the sample and its nearest neighbor, as shown in Figure 2 and Definition 1. Adjacent samples with similar d_i are categorized into the same cluster, and the method of searching for the nearest neighbor is modified. Only the distance between the sample and the preceding and the following samples needs to be calculated according to the property of the RBFNN training dataset. Thus, the calculation is significantly less than the other clustering algorithms, and only a single MNNC parameter requires to be determined.

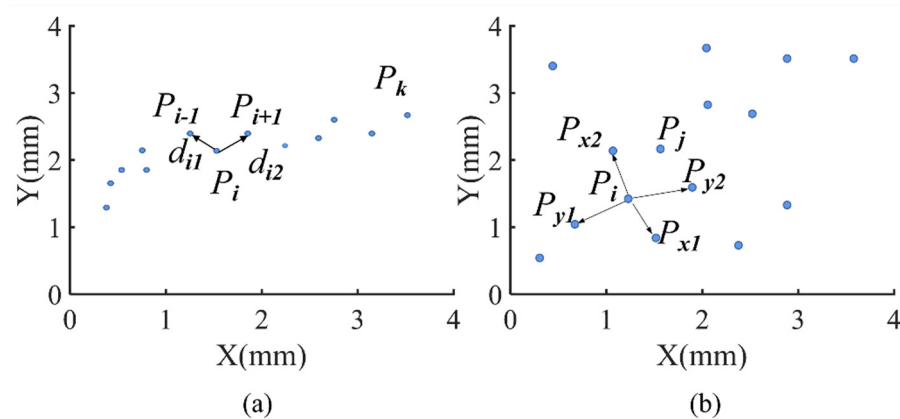


Figure 2. Sample distribution: (a) the training samples are dispersed with spatial sequence; (b) the training samples are dispersed without spatial sequence.

The basic principle of the clustering algorithm is that similar samples are placed in the same cluster, where the similarity of two samples is described by the Euclidean distance of the two samples. Since the sample has the characteristics of time series, the nearest samples to sample P_i are the adjacent samples P_{i-1} and P_{i+1} , as shown in Figure 2a. The distances between the samples are d_{i1} and d_{i2} , respectively, thus the nearest distance, d_i , of P_i is the smaller of $[d_{i1}, d_{i2}]$. The calculation of this method is simpler than that of the other clustering algorithms. Thereafter, d_i is divided into different levels, and the samples with the same d_i level belong to the same cluster (Figure 3a).

If the samples in one cluster are not adjacent, as shown in Figure 3a (where P_8 and P_{10} are not adjacent samples), the cluster is divided at the breakpoint (Figure 3b). Finally, the small clusters are merged with the adjacent clusters, as shown in Figure 3c,d. The merging criterion is that the total distance change (ΔD) between all samples and the centroid should be the least, such that the adjacent samples with the similar distance characteristic form a cluster. The related definitions are executed in Algorithm 1 for the distance (d_i), distance step (d_{step}), and distance changes (ΔD).

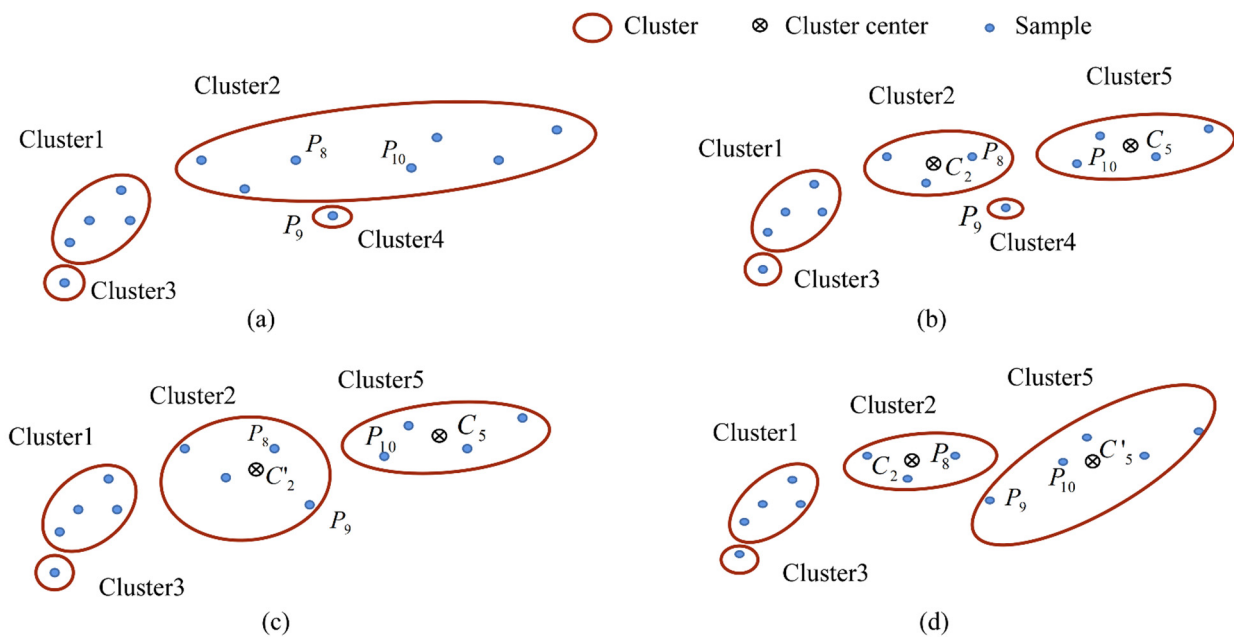


Figure 3. Clustering process of MNNC: (a) clustering result of training samples; (b) division of the cluster with discontinuous sample label into independent clusters; (c) Cluster 4 is merged into Cluster 2; (d) Cluster 4 is merged into Cluster 5.

Algorithm 1. MNNC.

Input: training samples (P_1, P_2, \dots, P_k), minimum samples number (Nmin) of cluster.

Output: clustering result

1. **for** each sample P_i do
 2. Calculate the d_i of P_i ; // referring to **Definition 1**.
 3. **end for**
 4. **for** each sample P_i do // arrange the samples into clusters referring to the distance.
 5. **if** $d_i < d_{min} + d_{step}$ then $P_i \in \text{cluster1}$; // **Definition 2**
 6. **else if** $d_i < d_{min} + 2 \times d_{step}$ then $P_i \in \text{cluster2}$;
 7. **else if** $d_i < d_{min} + 3 \times d_{step}$ then $P_i \in \text{cluster3}$;
 8. **else** $P_i \in \text{cluster4}$;
 9. **end if**; **end for**
 10. **for** cluster(i) do // The clusters with discontinuous sample numbers are divided into two clusters at the discontinuity.
 11. **if** the samples label of cluster(i) is not continuous then
 12. Divide the cluster(i) into cluster(i1) and cluster(i2) whose samples label is continuous.
 13. **end if**; **end for**
 14. **for** cluster(i) do // merge the small cluster into the adjacent clusters referring to **Definition 3**.
 15. **if** samples number of cluster(i) $< N_{min}$ **then**
 16. **if** $\Delta D_{i-1} < \Delta D_{i+1}$ then cluster(i - 1) = cluster(i - 1) + cluster(i);
 17. **else** cluster(i + 1) = cluster(i + 1) + cluster(i);
 18. **end if**; **end for**
 19. **Return** clusters
-

Definition 1. The distance d_i attribute of the sample, (P_1, P_2, \dots, P_k) is the dataset of path tracking control system that includes k samples, and $P[x_1, x_2, \dots, x_m]^T$ is a single sample of the dataset that consists of m dimension components. The distance between P_i and P_{i-1} is d_{i1} that can be expressed as [17]

$$d_{i1} = \sqrt{(x_{i1} - x_{i-11})^2 + (x_{i2} - x_{i-12})^2 + \dots + (x_{im} - x_{i-1m})^2} \quad (1)$$

The distance between P_i and P_{i+1} is d_{i2} that can be expressed as

$$d_{i2} = \sqrt{(x_{i1} - x_{i+11})^2 + (x_{i2} - x_{i+12})^2 + \dots + (x_{im} - x_{i+1m})^2} \quad (2)$$

The minimal distance d_i among (d_{i1}, d_{i2}) is expressed as

$$d_i = \min(d_{i1}, d_{i2}) \quad (3)$$

Definition 2. Distance step (d_{step}) (d_1, d_2, \dots, d_k) is the distance of the sample (P_1, P_2, \dots, P_k) .

$$d_{max} = \max(d_1, d_2, \dots, d_k) \quad (4)$$

$$d_{min} = \min(d_1, d_2, \dots, d_k) \quad (5)$$

The distance step is calculated as follows, where H is the number of d_i level determined by the user.

$$d_{step} = \frac{d_{max} - d_{min}}{H} \quad (6)$$

Definition 3. Distance change (ΔD)

The total distance of Cluster 2 and Cluster 5 (Figure 3b) is calculated before the merge operation.

$$D_2 = \sum_{i=1}^{Quan_2} \|P_i - C_2\|^2 \quad (7)$$

$$D_5 = \sum_{i=1}^{Quan_5} \|P_i - C_5\|^2 \quad (8)$$

where $Quan_2$ and $Quan_5$ are the quantity of samples in Cluster 2 and Cluster 5, respectively, and C_2 and C_5 are the centers of Cluster 2 and Cluster 5 as shown in Figure 3b, respectively; the center of Cluster 2 can be determined using Equation (9), and we can obtain the centers of the other clusters similarly.

$$C_{2m} = \frac{x_{1m} + x_{2m} + \dots + x_{Um}}{U} \quad (9)$$

where C_{2m} is the component m of the center of Cluster 2, and x_{1m} is the component m of sample 1 of Cluster 2. The total distances of Cluster 2 and Cluster 5 are calculated after the merge operation. If Cluster 4 is merged into Cluster 2, the center of Cluster 2 becomes C'_2 , as shown in Figure 3c.

$$D'_2 = \sum_{i=1}^{U'} \|P_i - C'_2\|^2 \quad (10)$$

If Cluster 4 is merged into Cluster 5, the center of Cluster 5 becomes C'_5 , as shown in Figure 3d.

$$D'_5 = \sum_{i=1}^{V'} \|P_i - C'_5\|^2 \quad (11)$$

Therefore, the distance change (ΔD) is

$$\Delta D_2 = |D'_2 - D_2| \quad (12)$$

$$\Delta D_5 = |D'_5 - D_5| \quad (13)$$

Figure 4 shows the results of Algorithm 1 for clustering. To test the algorithm, we selected 101 points from the curve $y = 1.1(1 - x + 2x^2)e^{-x^2/2}$ and combined them with random noise. We set the number of distance level $H = 4$, and tests were conducted twice with different datasets. Under these conditions, Algorithm 1 automatically generated seven and five clusters according to the distance properties (d_i , d_{step} , and ΔD), implying that MNNC is adaptive to the different density and can tune the number of clusters automatically. Merging clusters with a small number of samples (the number is not limited to 1) into other clusters can reduce the number of clusters. In this manner, when the

clustering algorithm is applied along with other intelligent algorithms, the speed of the intelligent algorithm can be improved.

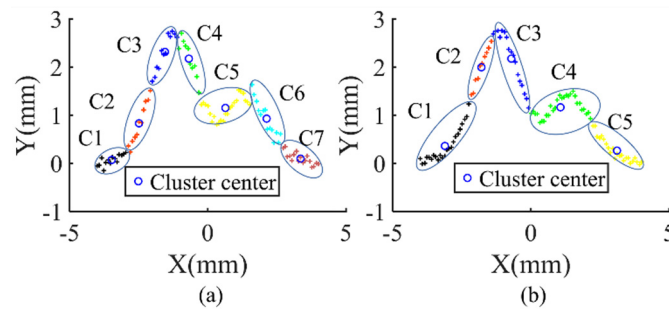


Figure 4. Clustering results: (a) the training samples are divided into seven clusters; (b) the training samples are divided into five clusters.

3.3. Enhancement of MNNC Performance

The clustering algorithm can use the samples with a time sequence or spatial sequence, as shown in Figure 2a; the nearest neighbor of sample P_i is either P_{i-1} or P_{i+1} . This clustering algorithm is suitable for curve fitting and path tracking control. Furthermore, the samples are randomly distributed without the time and spatial sequences, as shown in Figure 2b. In this case, the nearest neighbor of P_i may be in any direction, and we need to calculate the distances from P_i to its neighbors for determining d_i . The detailed calculation is shown in Definition 4. After we obtain the distances (d_1, d_2, \dots, d_k) , that is, the d_i of the samples, we set $d_{\max} = \min(d_1, d_2, \dots, d_k)$. Thereafter, the sample P_m is determined, whose distance to P_i is less than d_{\max} . These samples and P_i form a neighbor cluster of P_i . Similarly, the neighbor clusters of the other samples can be established. If the sample P_{\max} owns the distance attribute of d_{\max} , and its neighbor cluster contains only two samples, we define P_{\max} as an abnormal sample and delete this sample. Next, we set the updated maximum distance, and subsequently establish the neighbor cluster of each sample again. If there are several neighbor clusters containing the same samples, then these neighbor clusters merge into one cluster. The execution process proceeds based on Algorithm 2.

Definition 4. Distance attribute of the sample (d_i): To reduce the calculation, the samples are sorted by x and y , respectively. P_{x1} and P_{x2} are the preceding and following samples relative to sample P_i , sorted by x . P_{y1} and P_{y2} are the preceding and following samples relative to sample P_i sorted by y . d_{x1} , d_{x2} , d_{y1} , and d_{y2} are the distances between P_i and P_{x1} , P_{x2} , P_{y1} , and P_{y2} , respectively.

$$d_{x1} = \sqrt{(x_{x1} - x_i)^2 + (y_{x1} - y_i)^2} \quad (14)$$

$$d_{x2} = \sqrt{(x_{x2} - x_i)^2 + (y_{x2} - y_i)^2} \quad (15)$$

$$d_{y1} = \sqrt{(x_{y1} - x_i)^2 + (y_{y1} - y_i)^2} \quad (16)$$

$$d_{y2} = \sqrt{(x_{y2} - x_i)^2 + (y_{y2} - y_i)^2} \quad (17)$$

d_{i0} can be expressed as

$$d_{i0} = \min(d_{x1}, d_{x2}, d_{y1}, d_{y2}) \quad (18)$$

The sample P_j ; x_j and y_j of P_j satisfy

$$|x_j - x_i| \leq d_{i0} \quad (19)$$

$$|y_j - y_i| \leq d_{i0} \quad (20)$$

The distance between P_j and P_i : can be calculated using

$$d_j = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad (21)$$

Finally, the distance attribute of P_i is

$$d_i = \min(d_1, d_2, \dots, d_j) \quad (22)$$

Algorithm 2. MNNC for the path tracking of magnetic microrobot.

Input: training samples (P_1, P_2, \dots, P_k).

Output: clustering result.

1. **for** each sample P_i do
 2. Calculate the d_i of P_i ; // referring to **Definition 4**.
 3. **end for**
 4. $d_{max} = \max(d_1, d_2, \dots, d_k)$; $d_{min} = \min(d_1, d_2, \dots, d_k)$;
 5. $d_{step} = (d_{max} + d_{min})/H$; // H is distance level number that is determined by user.
 6. **for** each distance level;
 7. **for** each sample P_i do // find the nearest neighbors of P_i and establish the neighbor clusters.
 8. Find the sample P_m which $\|P_m - P_i\| \leq d_{min} + H \times d_{step}$;
 9. Construct cluster(i) = (P_i, P_m);
 10. **end for**
 11. **for** each cluster(i) do // if clusters contain same sample, then merge these clusters into one cluster.
 12. **If** cluster(i) \cap cluster(j) $\neq \emptyset$ then cluster(i) = cluster(i) + cluster(j);
 13. **end if**; **end for**
 14. **end for**
 15. merge the small cluster into nearest cluster
 16. **Return** clusters
-

It can be observed from the previous steps that MNNC for path tracking does not require us to pre-select important parameters based on experience. Because there is no iterative process, the calculation is not large in the algorithm. Furthermore, this clustering algorithm is also suitable for multi-dimensional samples. To verify the effectiveness of the clustering algorithm, we used MNNC, K-means, and DBSCAN to perform clustering analysis on the same samples, as shown in Figure 5.

We generated three synthetic datasets called Data 1 (Figure 5a), Data 2 (Figure 5b), and Data 3 (Figure 5c) containing 600, 1001, and 1650 samples, respectively. Data 1 consists of three clusters that are marked with 'blue +', 'red +', and 'black +', respectively. Data 2 also consists of three clusters that are marked with 'blue +', 'red +', and 'black +', respectively. Data3 consists of four clusters that are marked with 'blue +', 'red +', 'black +', and 'green +', respectively.

Figure 5d–f are the clustering results of Data 1, Data 2, and Data 3 obtained by k-means, respectively. Prior to the cluster analysis of Data 1, Data 2, and Data 3 by k-means, we set the parameter K to 3, 3, and 4, respectively, but the clustering results still remain incorrect.

Data 1 was divided into three clusters that were marked with 'blue +', 'green +', and 'red +', as shown in Figure 5d. As the reference result in Figure 5a shows, the samples of every cluster form a spiral. However, the cluster formations were changed when Data 1 was clustered by k-means. The changed cluster formed around each center of the clusters (C1, C2, and C3); blue cluster is the area around C1, green cluster is the area around C2, the red cluster is the area around C3. This is because of the principle of k-means that clusters the dataset based on the distance between samples and cluster centers. For example, we assume that C1, C2, and C3 are the centers of blue, green, and red clusters, respectively. P_n is any sample of the blue cluster, as shown in Figure 5d. For the sample P_n , C1 is the nearest cluster center among C1, C2, and C3. Therefore, sample P_n becomes one of the samples of the blue cluster. Similar results are shown in Figure 5e. The cluster marked with

'red +' consists of some samples indicated by circles and some samples indicated by "N" because C3 is the nearest cluster center for these samples. The k-means algorithm divides the samples in "N" into three different clusters that are marked with 'blue +', 'green +', and 'red +', respectively. However, in the reference result (Figure 5b), the samples in "N" form a single cluster (red +), and the samples in one circle form a single cluster as well. In Figure 5f, Data 3 is divided into four clusters by k-means algorithm, but the green cluster contains samples into two arcs because C2 becomes the nearest cluster center for the samples in this case. The blue cluster includes the samples of a circle and two arcs, and the samples in "I" are divided into two clusters. However, the reference result (Figure 5b) shows that the samples in one arc should form a single cluster, and the samples in circle and "I" should also form a single cluster, respectively. According to Figure 5d–f we can conclude that k-means was unsuccessful in clustering Data 1, Data 2, and Data 3.

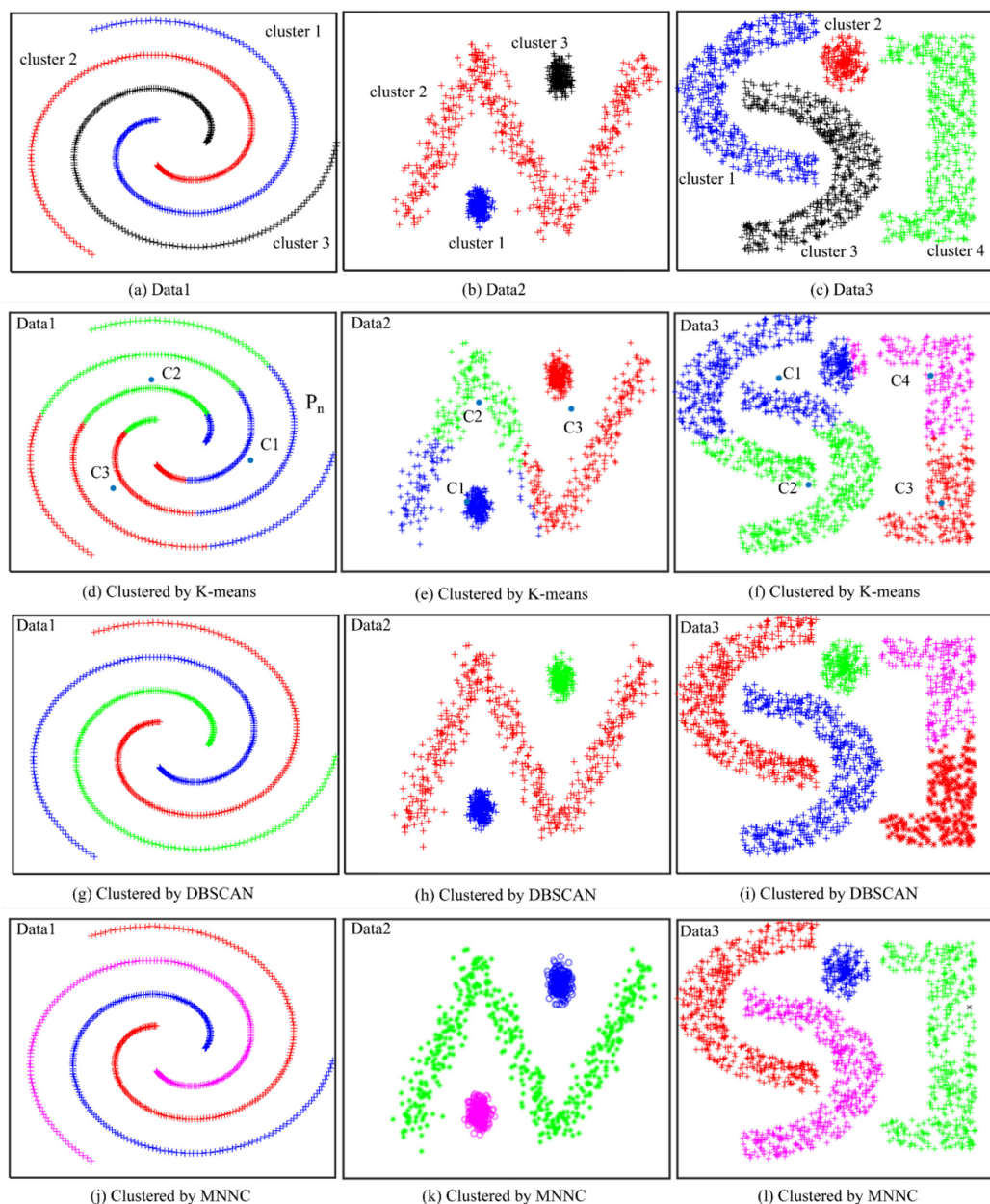


Figure 5. Clustering results of different clustering algorithms: (a–c) are the reference results; (d–f) are clustered by k-means; (g–i) are clustered by DBSCAN; (j–l) are clustered by MNNC.

During the simulation, we set R (scanning radius) of DBSCAN equal to the maximum distance (d_{max}) of MNNC, and set the same M_p (minimal points number) for Data 1, Data 2, and Data 3. The clustering results of Data 1, Data 2, and Data 3 by DBSCAN are shown in Figure 5g–i respectively. As shown in Figure 5g, DBSCAN divided Data 1 into three clusters precisely, as shown in Figure 5g; it placed the samples of one spiral in an independent cluster. DBSCAN generated three clusters for Data 2, as shown in Figure 5h; the three clusters were marked with ‘blue +’, ‘green +’, and ‘red +’, respectively. However, as shown in Figure 5i DBSCAN divided the samples in “I” into two clusters that were marked with ‘pink +’, and ‘black +’. The clustering result of Data 3 by DBSCAN is not equivalent to the reference result for Data 3 that shows that the samples in “I” belong to a single cluster. This is because DBSCAN not only depends on the parameter R , but also on the parameter M_p . However, at this time, M_p is not suitable for Data 3 anymore, implying that we should define two correct parameters of DBSCAN for different cases based on experience.

Figure 5j–l are the clustering results of Data 1, Data 2, and Data 3 using MNNC, respectively. Figure 5j,k show that MNNC generated equivalent clusters for Data 1 and Data 2. MNNC divided the samples of Data 1 and Data 2 into three clusters that were marked with ‘blue +’, ‘green +’, and ‘red +’, respectively. MNNC almost clustered Data 3 correctly, except it regarded one sample in “I” as an abnormal sample (marked with “black ×”), as shown in Figure 5l. This is because the sample, marked with “black ×”, features the largest d_i , and there are no similar samples around it.

Therefore, we can conclude that MNNC features the best clustering function for these datasets. Simultaneously, MNNC does not require users to decide the parameters, whereas both DBSCAN and k-mean require users to define two parameters. To clearly describe the clustering results, we used ACC and ARI to evaluate the clustering results [30]. The results are shown in Table 1.

As shown in Table 1, the ACC and ARI of three datasets of k-means clustering are much smaller than those of DBSCAN and MNNC, implying that the clustering results by k-means are lower than those obtained by DBSCAN and MNNC. The ACC and ARI of the spiral and zigzag of DBSCAN and MNNC are all 1, indicating that both DBSCAN and MNNC cluster those two datasets precisely. The ACC index of C4 of DBSCAN and MNNC are 0.8558 and 0.9994, respectively. The ARI index of C4 by DBSCAN and MNNC are 0.9019 and 0.9994, respectively. Both indexes of DBSCAN are smaller than those of MNNC which implies that the clustering results of MNNC are better than those of DBSCAN. The result of clustering index is similar to the clustering result, as shown in Figure 5.

Table 1. Clustering index of the clustering results.

Clustering Algorithm	Performance Index	Dataset		
		Data 1	Data 2	Data 3
K-means	ACC	0.3333	0.6683	0.4891
	ARI	0.0233	0.3121	0.3010
DBSCAN	ACC	1	1	0.8558
	ARI	1	1	0.9019
MNNC	ACC	1	1	0.9994
	ARI	1	1	0.9994

4. Adjustment of Training Samples Based on MNNC

The abnormal samples can be optimized by unsupervised methods before the intelligent algorithm parameters are defined. Therefore, the unsupervised methods can significantly improve the predictive ability of intelligent algorithm models [31]. Abnormal training samples always affect the operating efficiency of neural networks [32]; it is necessary to detect the abnormal samples and treat them. Training samples for the path tracking system with abnormal samples are shown in Figure 6a. The abnormal samples are marked as stars that reduce the learning effect of the algorithm for path tracking such as RBFNN. We can detect the abnormal samples by MNNC that requires fewer parameters than the

other algorithms. After defining the abnormal samples, we can delete them directly, but it is not particularly effective for curve fitting or path tracking control.

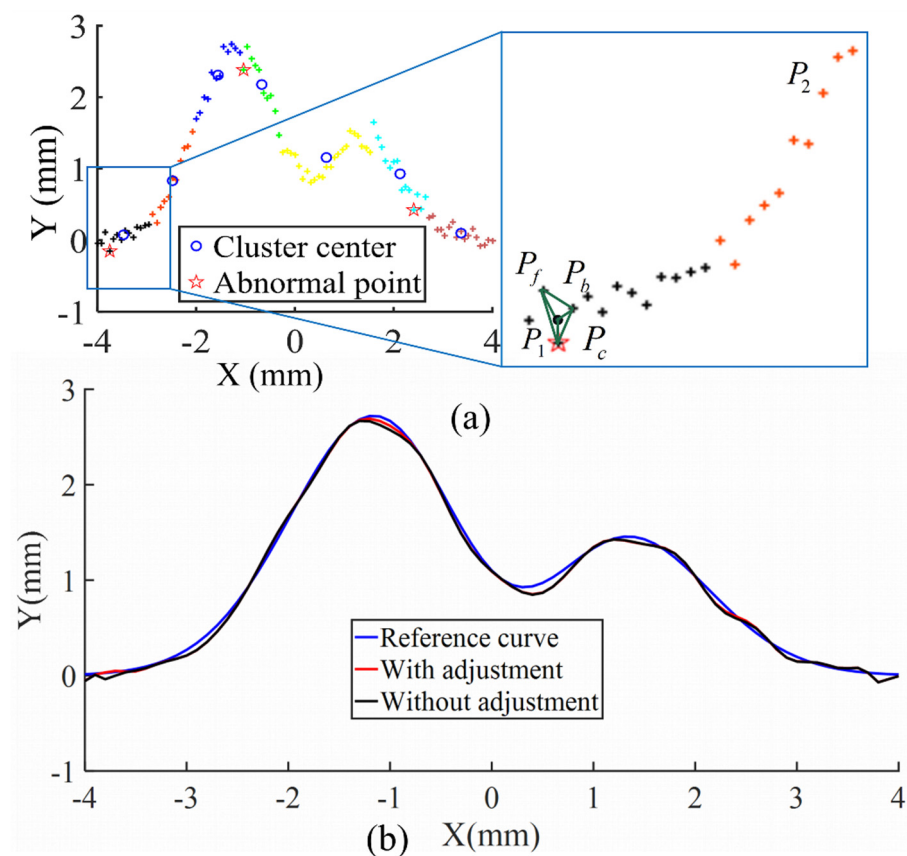


Figure 6. (a) Scheme of analysis of abnormal sample; (b) curve fitting results.

Particularly, when the number of training samples is not large, insufficient training samples also reduce the accuracy of RBFNN. It is effective to adjust the abnormal samples to normal samples. The process of detecting and adjusting abnormal samples is performed by MNNC. First, the training samples are clustered by MNNC. There are some independent samples because they are far away from the neighbors, such as P_1 and P_2 that are the abnormal and normal samples, respectively. We should distinguish between P_1 and P_2 . Therefore, a triangle is formed by the samples of P_1 , P_f , and P_b . The samples P_f and P_b are neighbors of P_1 , as shown in Figure 6a. Thereafter, we calculate the distance from the center of the triangle (P_c) to P_1 , P_f , and P_b , respectively. On one hand, if the distance (d_{c-1}) between P_c and P_1 is not larger than that of P_f and P_b , we define P_1 as a normal sample. On the other hand, when the d_{c-1} is larger than that of P_f and P_b , the sample P_1 becomes the abnormal sample and moves to P_c .

We tested the effect of training sample adjustment. The training samples were obtained from different curves that were combined with random noise or some specific noise. The results are shown in Figure 6b and summarized in Table 2. From Figure 6b and Table 2, we can observe that the fitting errors of the 2D curve mixed with random noise are 3.0793 when the abnormal samples are not adjusted; but the fitting errors of the same dataset are only 2.8145 after the abnormal samples are adjusted. Furthermore, the fitting errors of the 2D curve mixed with six noise points without and with abnormal sample adjustment are 2.7281 and 1.3292, respectively. The modified effect of the 3D curve is not comparable to that of the 2D curve; the fitting errors decrease from 516.6542 to 485.3374. Therefore, we can conclude that the adjustment of abnormal samples can improve the curve fitting accuracy. Particularly, the accuracy is improved by approximately 50% when there are only six abnormal samples. Because these six abnormal samples deviate far from the

normal samples, these six samples change considerably after they are adjusted to normal samples. Therefore, the accuracy of the entire curve fitting is significantly improved. All the simulations were performed in Matlab.

Table 2. Errors of curve fitting with/without the adjustment of abnormal samples.

Curve Type	2D Curve		3D Curve
Noise style	Randomly distributed	Six points with noise	Randomly distributed
Errors without adjustment	3.0793	2.7281	516.6542
Errors with adjustment	2.8145	1.3292	485.3374

5. Application of RBFNN in Path Tracking for a Spiral-Type Magnetic Microrobot

Figure 7a shows the control method of a spiral-type magnetic microrobot using rotating magnetic field (RMF) control. The robot is synchronized by the applied RMF and driven by magnetic torque. A rotation of the robot generates propulsive force via the screw mechanism. The driving magnetic torque T_m can be expressed as follows [33]:

$$T_m = VM \times B \quad (23)$$

where V is the volume of microrobot, M is the magnetization, and B is the external magnetic flux density. The magnetized direction of the robot is the radial direction. The external magnetic field is a uniform RMF and is generated by a three-axis Helmholtz coil. We assume that a magnetic field B generated by 3D Helmholtz coils rotates in plane P . Thus, the normal vector (nB) of plane P represents the movement direction of the robot. In addition, because the control angles of γ and α determine the position of plane P , the control of two angles determines the steering of the robot. The normal vector n_B and magnetic field B can be described as follows:

$$n_B = [\sin(\gamma) \cos(\alpha), \sin(\gamma) \sin(\alpha), \cos(\gamma)]^T \quad (24)$$

$$B = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = B_0 \begin{bmatrix} \cos(\gamma) \cos(\alpha) \sin(\omega t) + \sin(\alpha) \cos(\omega t) \\ \cos(\gamma) \sin(\alpha) \sin(\omega t) - \cos(\alpha) \cos(\omega t) \\ -\sin(\gamma) \sin(\omega t) \end{bmatrix} \quad (25)$$

where B_0 is the norm of B ; γ is the polar angle, and α is the azimuthal angle.

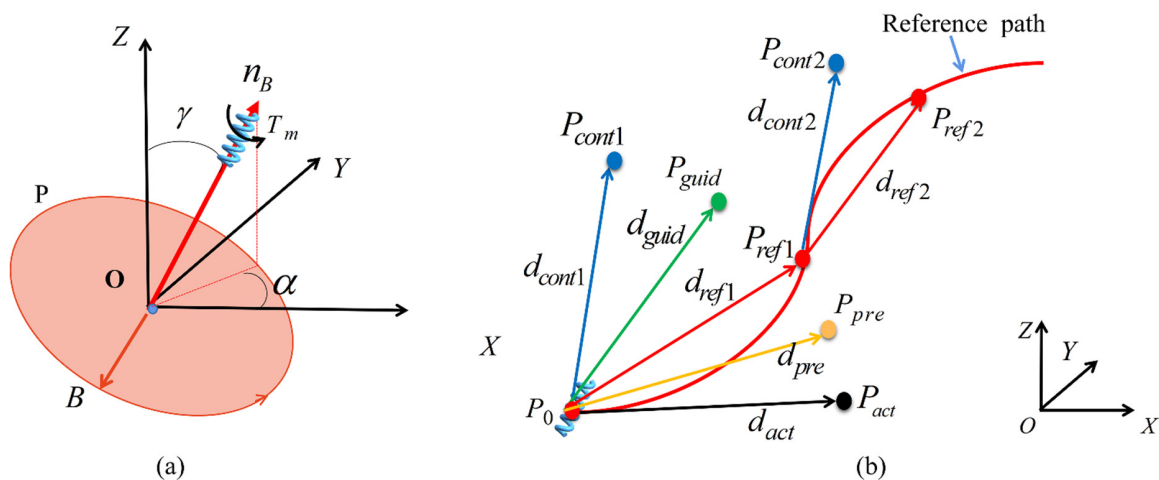


Figure 7. (a) Magnetic field vector; (b) scheme of path tracking: the green arrow is the guidance direction, the red arrow is the desired direction, the yellow arrow is the predicted direction, the blue arrow is the control direction, and the black arrow is the actual directions.

We assume that there are control errors resulting from various environmental factors. When we plan to drive the robot from the present position P_o to the reference target position P_{ref1} along the reference direction d_{ref1} , the robot may arrive at the actual position P_{act} because of locomotion error between the actual and reference positions. Therefore, when we drive the robot to move along the control direction d_{act1} to compensate for the locomotion error, the robot may reach the position P_{ref1} , as shown in Figure 7b. If there is no error between the actual and reference positions, the control direction d_{c1} is matched to the reference direction d_{ref1} by training RBFNN. Next, when the robot arrives the position P_{ref1} , we can obtain the next reference target position P_{ref2} and the reference direction d_{ref2} . Because the locomotion error is different in each locomotion step, we can obtain the corresponding compensation by driving the robot along the control direction. Therefore, driving the robot to move along the control direction d_{ci} , the robot can reach each reference position P_r along the reference path. To decide the steering direction of the robot along the reference direction, the two angles of γ_{ref} and α_{ref} are input to the RBFNN, and we obtain the actual control angles of γ_{cont} and α_{cont} by RBFNN for the controlling plane of RMF. To achieve this aim, it is necessary to develop a locomotion control system for the robot that is a nonlinear system. For nonlinear locomotion control systems, some researchers use RBFNN to simulate dynamic models [34,35]. However, the large number of parameters of these methods make the control system highly complicated. The neural network controller is a nonlinear mapping system; it has been proved that any smooth function can be represented by a three-layer neural network with sufficient hidden neurons [36]. Finally, RBFNN can be used to establish the relationship between the reference (γ_{ref} and α_{ref}) and control angles (γ_{cont} and α_{cont}) after it is trained.

Generally, RBFNN includes three layers: input, hidden, and output layers, as shown in Figure 8a. In this study, we used MNNC to train the RBFNN to develop its structure.

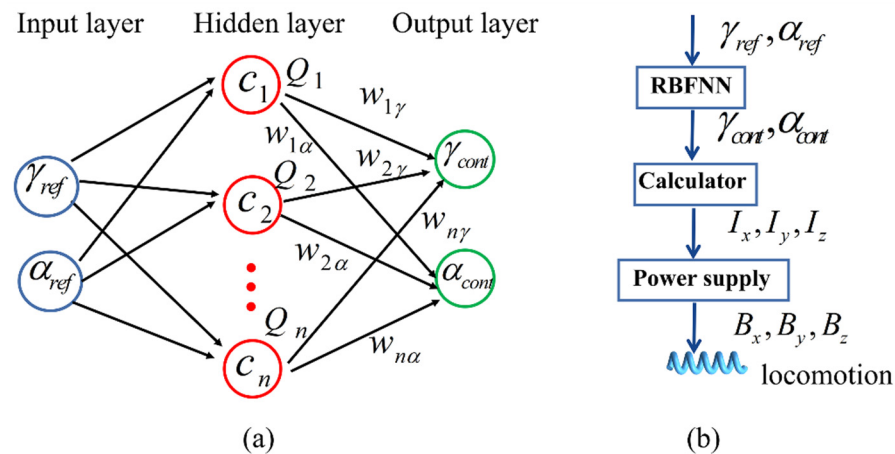


Figure 8. Scheme of RBFNN and control system: (a) structure of RBFNN for path tracking; (b) flow chart of control system.

The input layer of RBFNN includes two neurons: reference angle γ_{ref} and α_{ref} . The number of neuron and center of hidden layer are determined by MNNC after the RBFNN is trained. Accordingly, the output of the hidden layer can be obtained as

$$Q_n = e \left(-\frac{\|(\gamma_{ref}, \alpha_{ref}) - c_n\|^2}{2\sigma^2} \right) \quad (26)$$

where c_n is the center of hidden layer neurons that is decided by the MNNC. σ is the width of basis function that can be expressed as [37].

$$\sigma = \frac{d_c}{\sqrt{2n}} \quad (27)$$

where d_c is the maximum distance among the neuron centers of hidden layer; n is the quantity of neuron units of the hidden layer, and both of them can be obtained by MNNC.

The output layer includes two neurons: control angles of γ_{cont} and α_{cont} . They can be obtained by RBFNN according to

$$\begin{cases} \gamma_{cont} = \sum_{i=1}^n Q_i w_{i\gamma} \\ \alpha_{cont} = \sum_{i=1}^n Q_i w_{i\alpha} \end{cases} \quad (28)$$

where $w_{i\gamma}$ and $w_{i\alpha}$ is the weight of γ_{cont} and α_{cont} , respectively, that can be obtained by training the RBFNN based on the training samples.

Thus, when we input the reference angle into RBFNN, we can obtain the control angle from the output layer. Next, we calculate the driving current to generate RMF, as shown in Figure 8b. Using Equations (24) and (25), we obtain the driving magnetic field B that is the uniform magnetic field generated in the Helmholtz coils; the relationship between the magnetic field and coil current can be expressed as follows [33]:

$$B = \mu_0 N K_B I \quad (29)$$

where μ_0 is the permeability of vacuum, N is number of turns of coil, K_B is the magnetic field coefficient of Helmholtz coil, and I is the coil current.

To verify the ability of the proposed method for path tracking, we performed simulation using RBFNN with MNNC for path tracking. We generated 600 training samples to train the RBFNN to compare the performance of the clustering using MNNC, DBSCAN, and k-means applied to the path tracking simulation, as shown in Figure 9. The 600 samples were composed of 59 clusters for comparison under the same conditions. Thus, we could determine the neuron number and center of the hidden layer of the RBFNN, and obtain the width of each basis function of the RBFNN hidden layer.

Although the suitable k and accuracy of k -means are set, there are still some problems in the clustering result obtained by k -means algorithm. For example, there are many clusters (marked with circles) included for only one sample, as shown in Figure 9a. These clusters are closely spaced and could be merged into a large cluster. Although we adjusted the scanning radius and minimal sample number of DBSCAN for a long time, the clustering result was still not satisfactory. For example, there is a sample (marked with a red circle) far away from another sample in Cluster 1 that should be categorized as a neighboring cluster, as shown in Figure 9b. There are also samples far away from the other samples in Cluster 2 and Cluster 3, as shown in Figure 9b. Nonetheless, the similar data were placed in the same cluster by MNNC, as shown in Figure 9c.

As described above, after the training data were clustered, the neuron number was set as the cluster number of training data, and the cluster center was set as the neuron center of the hidden layer. We calculated and adjusted the width of basis function and weights between the hidden layer neurons and output layer neurons while training the RBFNN. Hence, the relationship between reference direction and control direction are established by RBFNN.

After the relationship between the control and reference angles are established, if we place any reference angle into RBFNN, we can obtain the corresponding control angle. To obtain the reference angle, we should obtain the reference target position first. We selected 30 points as the target points $P_t(x_t, y_t, z_t)$ in the reference path of the robot, and the path equation can be expressed as follows:

$$\begin{cases} x = 4 \cos(t) \\ y = 4 \sin(t) \\ z = 3t/\pi \end{cases} \quad (30)$$

$$\gamma_{ref} = \arccos \left(\frac{z_t - z_i}{\sqrt{(x_t - x_i)^2 + (y_t - y_i)^2 + (z_t - z_i)^2}} \right) \quad (31)$$

$$\alpha_{ref} = \arccos \left(\frac{x_t - x_i}{\sqrt{(x_t - x_i)^2 + (y_t - y_i)^2}} \right) \quad (32)$$

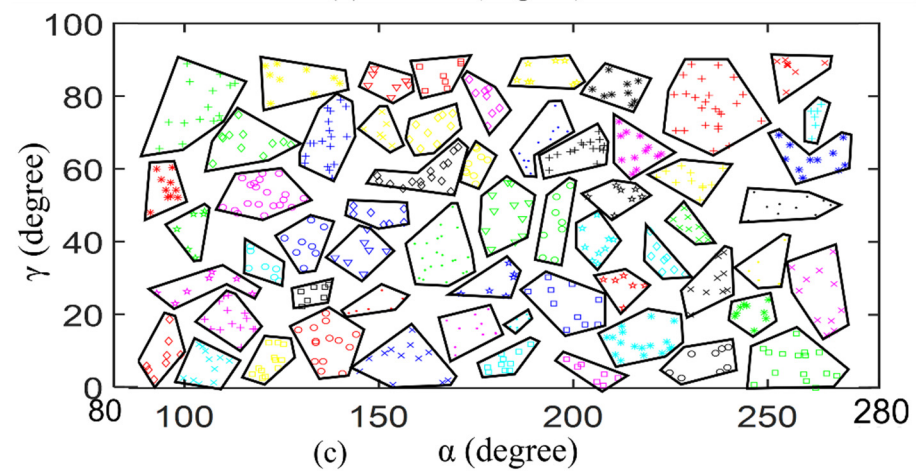
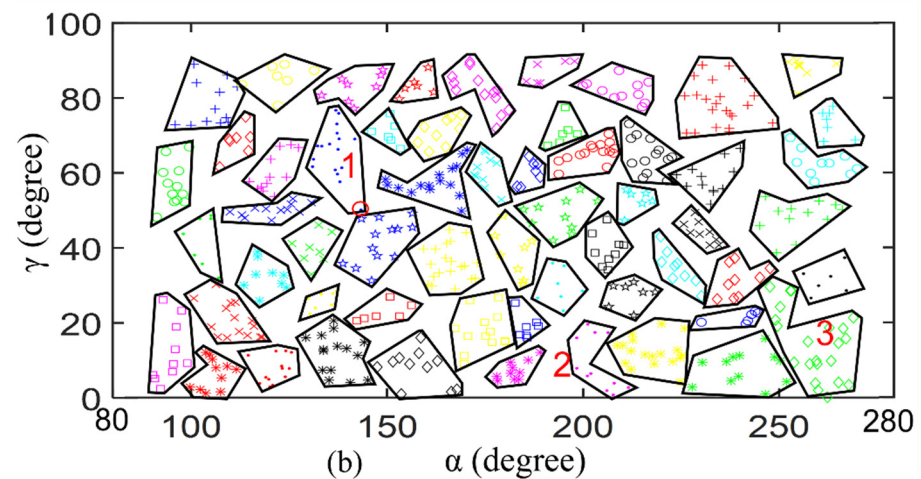
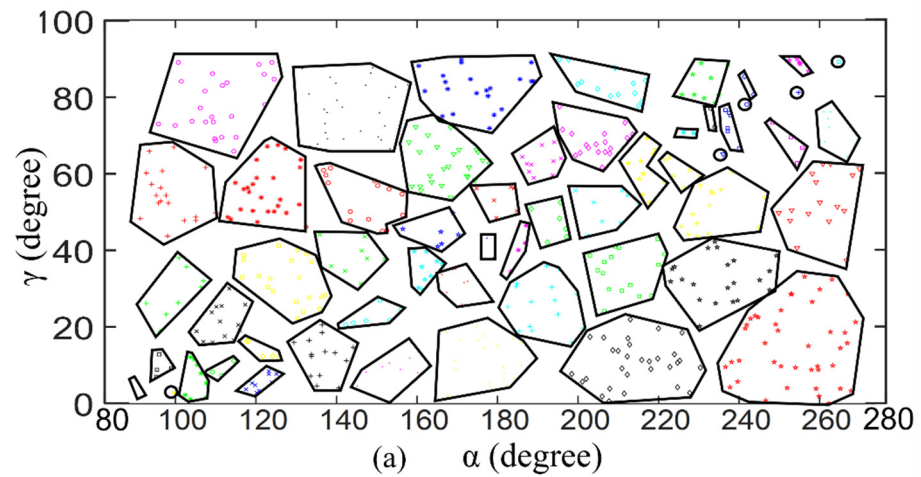


Figure 9. Clustering results of training data: (a) clustered by K-means; (b) clustered by DBSCAN; (c) clustered by MNNC.

These reference angles were mixed with the input data of training samples, and subsequently, they were combined with the compensation angle ($\alpha_{comp}, \gamma_{comp}$) to obtain the control directions d_{cont} with the components of α_{cont} and γ_{cont} , as shown in Figure 7b. Upon inputting γ_{ref} and α_{ref} to RBFNN, the control angles were obtained. Nonetheless, because of the fitting error of RBFNN, there was some deviation between the output data and control directions. The output of RBFNN at this time acts as the guidance direction d_{guid} that includes the components of α_{guid} and γ_{guid} , as shown in Figure 7b. α_{guid} and γ_{guid} can be obtained from RBFNN according to

$$\gamma_{guid} = \sum_{i=1}^n e^{\left(-\frac{\|(\alpha_{ref}, \gamma_{ref}) - c_i\|^2}{2\sigma_i^2}\right)} w_{i\gamma} \quad (33)$$

$$\alpha_{guid} = \sum_{i=1}^n e^{\left(-\frac{\|(\alpha_{ref}, \gamma_{ref}) - c_i\|^2}{2\sigma_i^2}\right)} w_{i\alpha} \quad (34)$$

where n , c_i , σ_i , $w_{i\gamma}$, and $w_{i\alpha}$ are the neuron number of the hidden layer, the neuron center of the hidden layer, the width of the basis function of the hidden layer, the weight of γ_{guid} , and the weight of α_{guid} , respectively. The parameters can be obtained after training RBFNN. When the reference angles of γ_{ref} and α_{ref} are input to RBFNN, the control angles γ_{cont} and α_{cont} are obtained for the path tracking of the spiral-type magnetic microrobot. Comparing the guidance and control directions, the test errors of radial basis function neural network, representing its accuracy, can be obtained. We trained and tested the RBFNN based on k-means, DBSCAN, and MNNC, respectively. These tests were based on the same learning rate, iteration number, the momentum factor, training samples, and test samples. The iteration number, learning rate, and momentum factor of the training process are 5000, 0.09, and 0.03, respectively. The results are shown in Table 3.

Table 3. Test results of RBFNN based on different clustering algorithms.

Algorithm	Cluster Number	Training Error	Test Error
K-means	59	2.27°	2.89°
DBSCAN	59	2.22°	2.24°
MNNC	59	2.10°	2.21°

As can be observed from Table 3, the cluster numbers of all clustering algorithms are 59, and the training parameters are the same, but the training and test errors are different. The training errors of radial basis function neural network based on k-means, DBSCAN, and MNNC are 2.27°, 2.22°, and 2.10°, respectively, and the test errors of the radial basis function neural networks based on k-means, DBSCAN, and MNNC are 2.89°, 2.24°, and 2.21°, respectively. Therefore, the radial basis function neural network based on MNNC provides the best test result. Accordingly, we can conclude that MNNC is the best algorithm for training the RBFNN for establishing the relationship between the control angle and the reference angle.

Based on the guidance direction angles, the coil current can be obtained from the calculator of the control system, as shown in Figure 8b. We selected seven positions of the test samples evenly, the coil current equations of which are shown in Table 4.

After current is input into the coils, the coils generate the magnetic field B_x , B_y , and B_z . These magnetic fields are combined into a rotating magnetic field that drive the spring-type robot to the predicted target P_{pre} along the predicted direction d_{pre} , as shown in Figure 7b. Here, we calculate the predicted angles α_{pre} and γ_{pre} based on

$$\begin{cases} \alpha_{pre} = \alpha_{guid} - \alpha_{comp} \\ \gamma_{pre} = \gamma_{guid} - \gamma_{comp} \end{cases} \quad (35)$$

where α_{comp} and γ_{comp} are the compensation angles that are generated during the sample generation process.

Table 4. Coil current equation corresponding to Figure 10b–h.

Position	$I_x(\text{A})$	$I_y(\text{A})$	$I_z(\text{A})$
b	$I_x = 0.999 \sin(360t + 90.69^\circ)$	$I_y = 0.238 \sin(360t + 12.72^\circ)$	$I_z = -0.973 \sin(360t)$
c	$I_x = 0.909 \sin(360t + 95.96^\circ)$	$I_y = 0.471 \sin(360t + 64.89^\circ)$	$I_z = -0.975 \sin(360t)$
d	$I_x = 0.592 \sin(360t + 109.57^\circ)$	$I_y = 0.84 \sin(360t + 80.86^\circ)$	$I_z = -0.971 \sin(360t)$
e	$I_x = 0.234 \sin(360t + 160.18^\circ)$	$I_y = 0.997 \sin(360t + 89^\circ)$	$I_z = -0.975 \sin(360t)$
f	$I_x = 0.481 \sin(360t + 245.64^\circ)$	$I_y = 0.904 \sin(360t + 96.15^\circ)$	$I_z = -0.975 \sin(360t)$
g	$I_x = 0.838 \sin(360t + 261.24^\circ)$	$I_y = 0.592 \sin(360t + 108.55^\circ)$	$I_z = -0.974 \sin(360t)$
h	$I_x = 0.997 \sin(360t + 268.84^\circ)$	$I_y = 0.272 \sin(360t + 163.4^\circ)$	$I_z = -0.965 \sin(360t)$

t is time (s).

We can obtain the angle error ratio of α and γ using

$$\begin{cases} err_{alpha} = \frac{\alpha_{pre} - \alpha_{ref}}{\alpha_{ref}} \times 100\% \\ err_{gamma} = \frac{\gamma_{pre} - \gamma_{ref}}{\gamma_{ref}} \times 100\% \end{cases} \quad (36)$$

The coordinate of the predicted target P_{pre} can be obtained from

$$\begin{cases} x_{pre} = \|P_{ref} - P_0\| \sin(\gamma_{pre}) \cos(\alpha_{pre}) \\ y_{pre} = \|P_{ref} - P_0\| \sin(\gamma_{pre}) \sin(\alpha_{pre}) \\ z_{pre} = \|P_{ref} - P_0\| \cos(\gamma_{pre}) \end{cases} \quad (37)$$

Accordingly, the position error is calculated based on

$$err_{position} = \frac{\|P_{pre} - P_{ref}\|}{\|P_{ref} - P_0\|} \times 100\% \quad (38)$$

The simulation result of seven positions are shown in Table 5 and Figure 10.

Because the control angles determine the steering direction, the two control angles automatically generate three current signals to produce an RMF and determine the position of the plane of RMF, as shown in Figure 10. Figure 10a shows the reference path and the simulation result of the path tracking on the reference path using RBFNN along with MNNC. Figure 10b–h shows the seven positions of the robot and their control conditions according to the changes in the control angles. There are the coordinates of position, reference path (green curve), plane of RMF (blue circle plan), movement direction (red arrow), or the direction of the normal vector of the plane of the rotating magnetic field, the control angles, and the generated currents in one cycle of the three-axis Helmholtz coils. At the seven positions on the path, the generated current signals for the RMFs are summarized in Table 4. We assumed that the frequency of RMF was 1 Hz and the coefficients $\mu_0 N K_B$ of the coil were normalized as 1. In addition, the robot has a right-handed screw mechanism, and the rotating direction of the magnetic field is clockwise. In this case, the direction of normal vector becomes the movement direction of the robot, and the control angles become the steering direction of the robot.

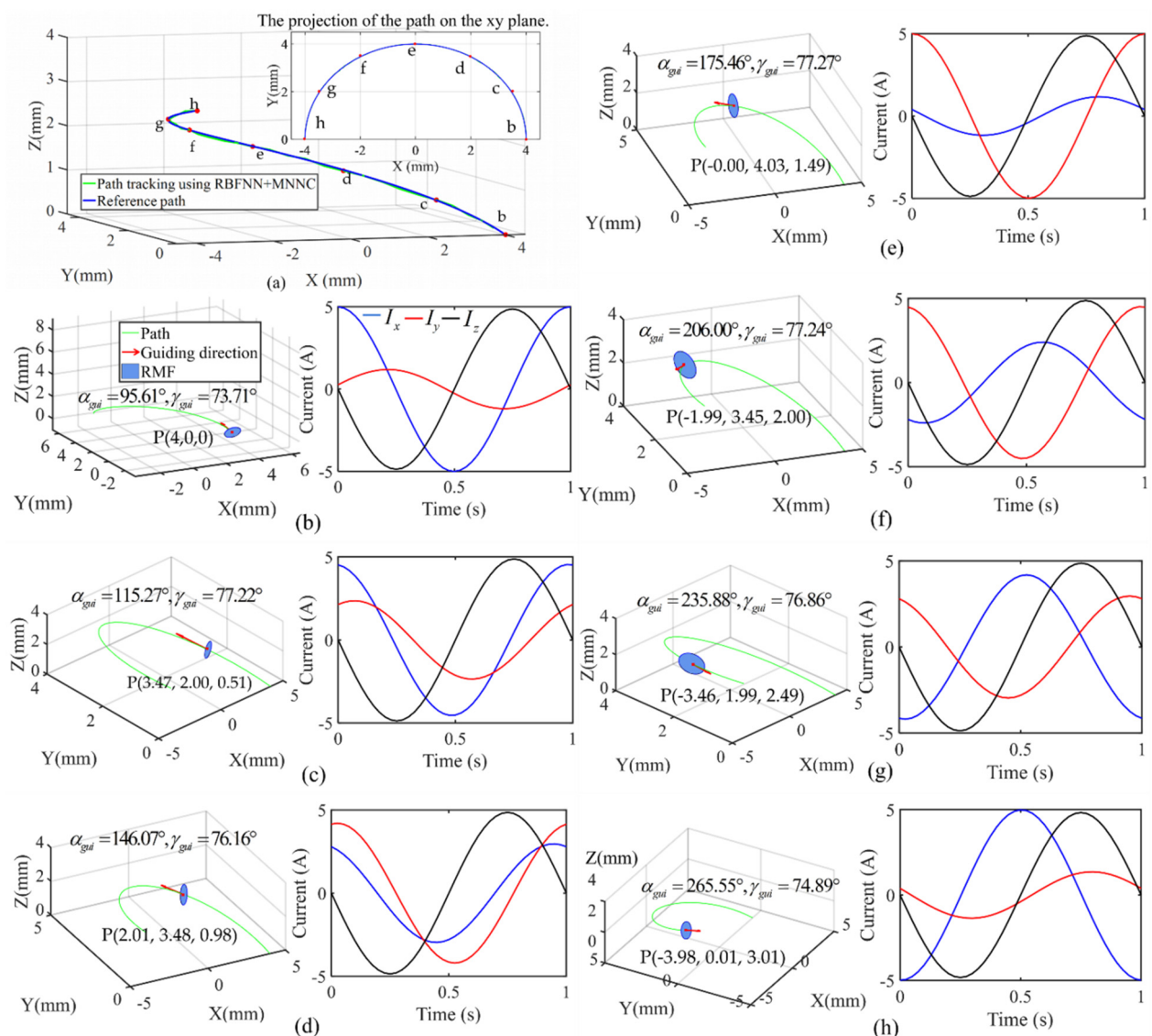


Figure 10. Path tracking of magnetic microrobot by RBFNN: (a) simulation results of path tracking by MNNC-based RBFNN; (b–h) show the rotating plan of the magnetic field and the movement direction at different positions of the path, and the coil current of X, Y, and Z-axis in one cycle; the rotating frequency of magnetic field is 1 Hz. For improved visualization of the parameters, the view angle for each figure is different.

Table 5. Path tracking error ratio of position b–h.

Position	Reference Angle (°)		Predicted Angle (°)		Angle Error Ratio (%)		Reference Coordinate (mm)	Predicted Coordinate (mm)	Position Error Ratio (%)
	α_{ref}	γ_{ref}	α_{pre}	γ_{pre}	γ	α			
b	93	76.57	93	76.57			(4, 0, 0)	(4, 0, 0)	0
c	117	76.57	115.78	75.79	1.04%	1.02%	(3.46, 2.0, 0.5)	(3.47, 2.00, 0.51)	2.48%
d	147	76.57	144.40	79.89	1.77%	4.34%	(2.0, 3.46, 1.0)	(2.01, 3.48, 0.98)	7.31%
e	177	76.57	172.34	77.90	2.63%	1.74%	(0, 4.0, 1.5)	(-0.00, 4.03, 1.49)	8.26%
f	207	76.56	208.69	76.30	0.81%	0.35%	(-2.0, 3.46, 2.0)	(-1.99, 3.45, 2.00)	2.90%
g	237	76.57	237.56	76.97	0.24%	0.53%	(-3.46, 2.0, 2.5)	(-3.46, 1.99, 2.49)	1.19%
h	267	76.56	269.61	76.39	0.98%	0.23%	(-4, 0, 3)	(-3.98, 0.01, 3.01)	4.44%

When the starting position of the robot was at point (4,0,0), the driving angles (γ_{ref} and α_{ref}) and guidance angles (γ_{guid} and α_{guid}) were calculated as $\gamma_{ref} = 76.57^\circ$, $\alpha_{ref} = 93^\circ$, $\gamma_{guid} = 73.71^\circ$, and $\alpha_{guid} = 95.61^\circ$, respectively, as shown in Figure 10b. Under the conditions, the three generated currents are $I_x = 0.999 \sin(360t + 90.69^\circ)$, $I_y = 0.238 \sin(360t + 12.72^\circ)$, and $I_z = -0.973 \sin(360t)$, respectively. When the guid-

ance angles of γ_{guid} and α_{guid} are 77.22° and 115.27° , the robot reaches Position c, as shown in Figure 10c. Moving from Position b to c, we can confirm that the current profiles are changed by the variation of the guidance angles (steering angles). α_{guid} allows the RMF plane to rotate around the Z-axis and the changes in the γ_{guid} cause the RMF plane to rotate around X-axis or Y-axis or both (Figure 7a). Therefore, when there is no angular change in the moving path of the robot, the generated current profiles are constant, while the current profiles are changed when there is an angular change in the moving path. Through the phase difference and amplitude of the currents, the movement direction of the robot is determined.

In Figure 10c, the present position of the microrobot is (3.47, 2.00, 0.51). The guidance angle γ_{guid} and α_{guid} are 77.22° and 115.27° , respectively, that were obtained by RBFNN. The control system calculated the corresponding coils current along the x-axis, y-axis, and z-axis indicated by the blue, red, and green curves, respectively, as shown in Figure 10c. The amplitudes of I_x , I_y , and I_z are 0.909, 0.471, and 0.975, respectively, as shown in Table 4. From Table 4, we can observe that the phase of I_x , I_y , and I_z are 95.96° , 64.89° , and 0 , respectively. Comparing Figure 10b,c the current in the z-axis coils of these two cases are similar because the angle γ_{guid} changes negligibly. However, there are large changes in the curve corresponding to the current in the x-axis and y-axis coils because the angle α_{guid} changes significantly; therefore, we can obtain the results using Equations (24) and (25). The similar control process was implemented for the other positions, and the corresponding results are shown in Figure 10c–g and Table 5. The error ratios of path tracking are shown in Table 5. When the microrobot is at Position c, the reference target position and actual position coordinates are (3.5, 2.0, 0.5) and (3.47, 2.00, 0.51), respectively. We calculated the reference distance from the starting position to the reference target position for each step, and calculated the deviation between reference target and predicted position. Accordingly, the path tracking error ratios at Positions c, d, e, f, g, and h, are obtained as shown in Table 5. Because Position b is the initial position of the entire path tracking, there is no error at this time. The error ratios are primarily less than 5%. Finally, the microrobot realized the locomotion along the reference path as shown in Figure 10h. The standard deviation of position is 0.0145 mm. According to the result, we can conclude that the control system based on RBFNN can provide the control direction of each position. Subsequently, the corresponding coil currents can be calculated to generate the rotating magnetic field for driving the robot to move along the reference direction.

In the actual experiment, it is necessary to obtain some training samples for RBFNN learning, to establish the relationship between the reference angle and control angle. First, we can obtain the present position P_0 of robot. We set the control direction with angle γ_{cont} and α_{cont} , and calculate the currents of the Helmholtz coils. Thereafter, the Helmholtz coils generate the rotating magnetic field and drive the spring-type robot to the position P_{ref} . The simulation result for this case shows that if we want to drive the robot from P_0 to P_{ref} , we can set the control angle γ_{cont} and α_{cont} to generate a rotating magnetic field for the movement of the robot. The direction from P_0 to P_{ref} is the reference direction. We can calculate the angle γ_{ref} and α_{ref} of the direction from P_0 to P_{ref} using Equations (31) and (32). Thus, a training sample with components of $\gamma_{ref}, \alpha_{ref}, \gamma_{cont}$, and α_{cont} is obtained. In this manner, we can obtain many training samples and train the RBFNN.

After the RBFNN is trained, we can apply the control system based on RBFNN to path tracking control. We can obtain the reference target position and present position of each step, and subsequently calculate the reference angle γ_{ref} and α_{ref} to provide as input to the RBFNN. RBFNN outputs the control angle γ_{cont} and α_{cont} . Next, the control system can derive the current of Helmholtz coils, and subsequently generate the RMF to drive the robot to the reference target position.

6. Discussion

Clustering algorithms can classify similar samples into the same cluster, but the conventional clustering algorithms often require the determination of several important pa-

parameters based on experience in advance, thus leading to inconvenience. Moreover, when conventional clustering algorithms are applied in some specific situations, the clustering algorithms can be improved to increase efficiency and accuracy. MNNC determines the samples with the highest similarity based on the determination of the nearest neighbors. Because the data of curve fitting and path following control have the characteristics of obvious time or spatial sequence, the performance of the MNNC is considerably improved for this type of data. MNNC reduces the range of determining the nearest neighbor that reduces the computation cost, thereby requiring few parameters to be set. Furthermore, it can adaptively adjust the number of clusters. Moreover, MNNC can determine the abnormal samples in the dataset and adjust them. After the adjusted data is used for curve fitting, the fitting accuracy can be improved by 50%; particularly, the adjustment effect is more prominent when there are not many outliers because the adjustment is performed on the samples with the largest outliers, and the sample adjustment is a gradual process. The abnormal sample adjustment of MNNC can avoid misjudgment and over-adjustment of outliers. For cases with a large number of outliers, the adjustment effect can be enhanced by increasing the number of optimizations.

RBFNN is commonly used in nonlinear systems for curve fitting and path tracking control. Using a clustering algorithm to obtain the initial parameters of RBFNN is a relatively simple method. When MNNC is used to train RBFNN, the number of hidden nodes of RBFNN can be changed by automatically adjusting the number of clusters according to the accuracy requirements of RBFNN, to improve the accuracy of RBFNN. The simulation results show that the curve fitting accuracy of RBFNN trained by MNNC is up to 60% higher than that of other RBFNNs.

When the magnetic microrobot is moving, it is difficult to reach the target position accurately due to the interference of various factors. In this study, the motion mechanism of the magnetic robot is analyzed, and a locomotion control system based on RBFNN is proposed. The system uses RBFNN to determine the reference target of the theoretical locomotion target, and controls the magnetic microrobot to reach the theoretical motion target by moving to the reference target. In order to use MNNC to establish the parameters of RBFNN better, this study enhanced the function of MNNC on the basis of the previous analysis. The simulation results show that the enhanced MNNC demonstrates an improved performance over traditional clustering algorithms in clustering analysis, and there are fewer parameters to be determined in advance. The simulation results show that a better control effect can be obtained after applying RBFNN based on MNNC in the path tracking control of the magnetic robot.

Although MNNC has only been applied in clustering 2D data in this study, it can adapt to multidimensional datasets. This will be verified in future research, and the algorithm will be improved to increase the clustering accuracy and generalization ability.

7. Conclusions

A modified nearest neighbor-based clustering algorithm is proposed in this study that does not necessitate the setting of important parameters relying on past experience, and can perform cluster analysis on samples of different densities and shapes. The abnormal samples can be found, and the adjustment of the sample can be realized by this clustering algorithm. The simulation results show that the curve fitting accuracy of the samples optimized by the clustering algorithm is increased by 50%. The number and center of basis functions can be automatically determined by applying this clustering algorithm on the training samples of RBFNN. The simulation results proved that the RBFNN trained in this manner has a higher operating accuracy than the conventional RBFNN; the accuracy in curve fitting is improved by 60%. The simulation result showed that the RBFNN based on the clustering algorithm could improve the accuracy in the path tracking simulation by 20%.

Author Contributions: Conceptualization, D.Z. and S.K.; methodology, D.Z.; validation, W.J. and S.K.; investigation, D.Z.; writing—original draft preparation, D.Z.; writing—review and editing, W.J. and S.K.; supervision, S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Korea Medical Device Development Fund grant funded by the Korea government (the Ministry of Science and ICT, the Ministry of Trade, Industry and Energy, the Ministry of Health & Welfare, the Ministry of Food and Drug Safety) (Project Number: KMDF_PR_20200901_0130, 9991006803), and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1A4A3079595 and 2018R1C1B6003491).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data can be requested from the corresponding authors.

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

Appendix A

Table A1. List of acronyms.

RBFNN	Radial basis function neural network
MNNC	Modified nearest neighbor-based clustering
PID	Proportion integral differential
SVM	Support vector machines
GA	Genetic algorithm
AI	Artificial immune
DBSCAN	Density-based spatial clustering of applications with noise
DPC	Density peaks clustering
ACC	Accuracy
ADI	Adjusted Rand index
RMF	Rotating magnetic field
BIRCH	Balanced iterative reducing and clustering using hierarchies
CLIQUE	Clustering in QUES

Appendix B

Table A2. List of variables.

d_i	Distance between sample and its nearest neighbor
H	The number of d_i level
P_i	Sample
ΔD	Distance change
d_{step}	Distance step
D	Total distances between samples and centroid of cluster before merged
D'	Total distances between samples and centroid of cluster after merged
$Quan_i$	Quantity of samples of cluster i
C	Center of cluster
R	Scanning radius of DBSCAN
T_m	Magnetic torque
V	Volume of microrobot
M	Magnetization of microrobot
B	External magnetic field
γ	Polar angle
α	Azimuthal angle

Table A2. Cont.

n_B	Normal vector of plan P
P_{cont}	Control position
P_{ref}	Reference position
P_{act}	Actual position
P_{guid}	Guidance position
P_{pre}	Predicted position
γ_{ref}	Reference polar angle
α_{ref}	Reference azimuthal angle
γ_{cont}	Control polar angle
α_{cont}	Control azimuthal angle
γ_{guid}	Guidance polar angle
α_{guid}	Guidance azimuthal angle
γ_{pre}	Predicted polar angle
α_{pre}	Predicted azimuthal angle
γ_{comp}	Compensation of polar angle
α_{comp}	Compensation of azimuthal angle
d_{cont}	Control direction
d_{guid}	Guidance direction
d_{ref}	Reference direction
d_{pre}	Predicted direction
d_{act}	Actual direction
Q_i	Output of hidden layer neuron of RBFNN
c_i	Center of hidden layer neuron of RBFNN
w_i	Weight between hidden layer and output layer of RBFNN
σ	Width of hidden layer neuron of RBFNN
μ_0	Permeability of free space
N	Turns number of coil
K_B	Magnetic field coefficient of Helmholtz coil
I	Coil current
I_x	Coil current of x -axis
I_y	Coil current of y -axis
I_z	Coil current of z -axis
err_{γ}	Error of polar angle
err_{α}	Error of azimuthal angle
$err_{position}$	Error of position

References

- Rout, R.; Subudhi, B. Inverse optimal self-tuning PID control design for an autonomous underwater vehicle. *Int. J. Syst. Sci.* **2017**, *48*, 367–375. [\[CrossRef\]](#)
- Xiang, H.B.; Li, M.W.; Zhang, T.L.; Wang, S.J.; Zhang, M.; Song, Y.; Huo, W.X.; Huang, X. Motion characteristics of untethered swimmer with magnetoelastic material. *Smart Mater. Struct.* **2021**, *30*, 075030. [\[CrossRef\]](#)
- Ghosh, B.B.; Sarkar, B.K.; Saha, R. Realtime performance analysis of different combinations of fuzzy-PID and bias controllers for a two degree of freedom electrohydraulic parallel manipulator. *Robot. Comput.-Integr. Manuf.* **2015**, *34*, 62–69. [\[CrossRef\]](#)
- Algarin-Pinto, J.A.; Garza-Castanon, L.E.; Vargas-Martinez, A.; Minchala-Avila, L.I. Dynamic modeling and control of a parallel mechanism used in the propulsion system of a biomimetic underwater vehicle. *Appl. Sci.* **2021**, *11*, 4909. [\[CrossRef\]](#)
- Mai, T.A.; Dang, T.S.; Duong, D.T.; Le, V.C.; Banerjee, S. A combined backstepping and adaptive fuzzy PID approach for trajectory tracking of autonomous mobile robots. *J. Braz. Soc. Mech. Sci. Eng.* **2021**, *43*, 156. [\[CrossRef\]](#)
- Xu, B.J.; Ji, S.; Zhang, C.R.; Chen, C.; Ni, H.P.; Wu, X.J. Linear-extended-state-observer-based prescribed performance control for trajectory tracking of a robotic manipulator. *Ind. Robot.* **2021**, *44*, 544–555. [\[CrossRef\]](#)
- Li, Q.X.; Zhou, Y.S. Precise trajectory tracking control of ship towing systems via a dynamical tracking target. *Mathematics* **2021**, *9*, 974. [\[CrossRef\]](#)
- Wu, Y.; Wang, L.; Zhang, J.; Li, F. Path following control of autonomous ground vehicle based on nonsingular terminal sliding mode and active disturbance rejection control. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6379–6390. [\[CrossRef\]](#)
- Liu, C.; Cheah, C.C.; Slotine, J. Adaptive Jacobian PID regulation for robots with uncertain kinematics and actuator model. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 3044–3049. [\[CrossRef\]](#)
- Smeresky, B.; Rizzo, A.; Sands, T. Optimal learning and self-awareness versus PDI. *Algorithms* **2020**, *13*, 23. [\[CrossRef\]](#)
- Juang, J.G. Intelligent trajectory control using recurrent averaging learning. *Appl. Artif. Intell.* **2001**, *15*, 277–296. [\[CrossRef\]](#)

12. Chen, Z.; Yang, X.; Liu, X. RBFNN-based nonsingular fast terminal sliding mode control for robotic manipulators including actuator dynamics. *Neurocomputing* **2019**, *362*, 72–82. [[CrossRef](#)]
13. Nie, L.; Guan, J.; Lu, C. Longitudinal speed control of autonomous vehicle based on a self-adaptive PID of radial basis function neural network. *IET Intell. Transp. Syst.* **2018**, *12*, 485–494. [[CrossRef](#)]
14. Lee, C.T.; Tsai, C.C. Improved nonlinear trajectory tracking using RBFNN for a robotic helicopter. *Int. J. Robust Nonlinear Control* **2010**, *20*, 1079–1096. [[CrossRef](#)]
15. Li, J.; Wang, J.; Wang, S. Neural approximation-based model predictive tracking control of non-holonomic wheel-legged robots. *Int. J. Control Autom. Syst.* **2020**, *19*, 372–381. [[CrossRef](#)]
16. Guillen, A.; Rojas, I.; Gonzalez, J. Output value-based initialization for radial basis function neural networks. *Neural Process. Lett.* **2007**, *25*, 209–225. [[CrossRef](#)]
17. Juang, J.G. Effects of Using Different Neural Network Structures and Cost Functions in Locomotion Control. In *Lecture Notes in Computer Science, Proceedings of the 2nd International Conference on Natural Computation (ICNC 2006), Xian, China, 24–28, September 2006*; Springer: Berlin/Heidelberg, Germany, 2006. [[CrossRef](#)]
18. Fernandez-Navarro, F.; Hervás-Martínez, C.; Gutiérrez, P.A. Generalised gaussian radial basis function neural networks. *Soft Comput.* **2013**, *17*, 519–533. [[CrossRef](#)]
19. Huang, H.C.; Chiang, C.H. An evolutionary radial basis function neural network with robust genetic-based immunecomputing for online tracking control of autonomous Robots. *Neural Process. Lett.* **2016**, *44*, 19–35. [[CrossRef](#)]
20. Chen, Z.Y.; Kuo, R.J. Combining SOM and evolutionary computation algorithms for RBF neural network training. *J. Intell. Manuf.* **2019**, *30*, 1137–1154. [[CrossRef](#)]
21. Guillén, A.; Rojas, I.; Gonzalez, J. A possibilistic approach to RBFN centers initialization. In *Lecture Notes in Computer Science, Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC 2005), Regina, SK, Canada, 31 August 2005*; Springer: Berlin/Heidelberg, Germany, 2005. [[CrossRef](#)]
22. Oh, S.K.; Kim, W.D.; Pedrycz, W. Design of k-means clustering-based polynomial radial basis function neural networks (pRBF-NNs) realized with the aid of particle swarm optimization and differential evolution. *Neurocomputing* **2012**, *78*, 121–132. [[CrossRef](#)]
23. Liao, C.C. Genetic k-means algorithm based RBF network for photovoltaic MPP prediction. *Energy* **2010**, *35*, 529–536. [[CrossRef](#)]
24. Dogan, A.; Birant, D. Machine learning and data mining in manufacturing. *Expert Syst. Appl.* **2021**, *166*, 114060. [[CrossRef](#)]
25. Fan, Y.K.; Bai, J.R.; Lei, X.; Lin, W.G.; Hu, Q.; Wu, G.D.; Guo, J.M.; Tan, G. PPMCK: Privacy-preserving multi-party computing for K-means clustering. *J. Parallel Distrib. Comput.* **2020**, *154*, 54–63. [[CrossRef](#)]
26. Khalid, A.; Hammed, A. Genetic divergence in wheat genotypes based on seed biochemical profiles appraised through agglomerative hierarchical clustering and association analysis among traits. *Pak. J. Bot.* **2021**, *53*, 1281–1286. [[CrossRef](#)]
27. Fang, X.J.; Wu, Y.J.; Liao, S.S.; Xue, L.Z.; Chen, Z.; Yang, J.N.; Lu, Y.M.; Ling, K.; Hu, S.Y.; Kong, S.Y. Division of crustal units in China using grid-based clustering and a zircon U-Pb geochronology database. *Comput. Geosci.* **2020**, *145*, 104570. [[CrossRef](#)]
28. Montanari, G.E.; Doretti, M.; Marino, M.F. Model-based two-way clustering of second-level units in ordinal multilevel latent Markov models. *Adv. Data Anal. Classif.* **2021**, 1–29. [[CrossRef](#)]
29. Luo, S.; Liu, H.W.; Qi, E. Recognition and labeling of faults in wind turbines with a density-based clustering algorithm. *Data Technol. Appl.* **2021**. [[CrossRef](#)]
30. Li, C.Z.; Zhang, Y.Z. Density peak clustering based on relative density optimization. *Math. Probl. Eng.* **2020**, *2020*, 2816102. [[CrossRef](#)]
31. Syarrudin, M.; Alfian, G.; Fitriyani, N.L. Performance analysis of IoT-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing. *Sensors* **2018**, *18*, 2946. [[CrossRef](#)]
32. Minimol, P.V.; Mishra, D.; Gorthi, R.K.S.S. Guided MDNet tracker with guided samples. *Visual Comput.* **2021**, 1–15. [[CrossRef](#)]
33. Jeon, S.; Kim, S.; Ha, S.; Lee, S.; Kim, E.; Kim, S.Y.; Park, S.H.; Jeon, J.H.; Kim, S.W.; Moon, C. Magnetically actuated microrobots as a platform for stem cell transplantation. *Sci. Robot.* **2019**, *4*, eaav4317. [[CrossRef](#)]
34. Chen, X.Z.; Hoop, M.; Mushtaq, S.F.; Hu, E.C.Z.; Nelson, B.J.; Pane, S. Recent developments in magnetically driven micro- and nanorobots. *Appl. Mater. Today* **2017**, *9*, 37–46. [[CrossRef](#)]
35. Li, J.; Li, X. Development of a magnetic microrobot for carrying and delivering targeted cells. *Sci. Robot.* **2018**, *3*, eaat8829. [[CrossRef](#)] [[PubMed](#)]
36. Yan, X.H.; Zhou, Q.; Vincent, M.; Deng, Y.; Yu, J.F.; Xu, J.B.; Xu, T.T.; Tang, T.; Bian, L.M.; Wang, Y.X.J.; et al. Multifunctional biohybrid magnetite microrobots for imaging-guided therapy. *Sci. Robot.* **2017**, *2*, eaaq1155. [[CrossRef](#)] [[PubMed](#)]
37. Yang, C.; Wang, X.; Cheng, L. Neural-learning-based telerobot control with guaranteed performance. *IEEE Trans. Cybern.* **2017**, *47*, 3148–3159. [[CrossRef](#)] [[PubMed](#)]