Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 5640562, 10 pages
https://doi.org/10.1155/2022/5640562

Research Article

Visual Analysis of Sports Actions Based on Machine Learning and Distributed Expectation Maximization Algorithm

Yan Luo

Jinhua Polytechnic, Jinhua 321000, China

Correspondence should be addressed to Yan Luo; 20070113@jhc.edu.cn

Received 17 April 2022; Revised 26 May 2022; Accepted 6 June 2022; Published 25 June 2022

Academic Editor: Kapil Sharma

In order to improve the scientificity of sports action analysis, this paper constructs a sports action analysis model based on machine learning based on the greedy algorithm and the bat algorithm. According to the structural characteristics of the model, the structure of the model is reflected in the form of face order, that is, the face neighborhood structure. Moreover, this paper judges the degree of similarity between model faces through the pros and cons of the order and applies it to the structural similarity matrix between models. In addition, this paper establishes corresponding mathematical models for the shape and structure of the model and constructs the shape similarity matrix, the surface neighborhood structure similarity matrix, and the structure similarity matrix between the source model and the target model. Finally, this paper designs and implements CAD model retrieval methods based on greedy algorithm and bat algorithm and designs experiments to compare the performance of the algorithm proposed in this paper with traditional algorithms. The result of the experiment shows that the algorithm proposed in this paper has obvious advantages in sports action analysis compared with the traditional algorithm.

1. Introduction

Scientific analysis of sports movements can effectively improve the effects of sports training, provide athletes with more scientific and effective training guidance, and reduce training injuries. Therefore, auxiliary sports training through intelligent sports movement recognition and analysis models is the direction of future sports training development. With the development of cost-effective sensors and the advancement of human body pose estimation algorithms, it has become easier to obtain three-dimensional and two-dimensional bone point data from human motion video data. The motion trajectory of the human skeleton points in the time sequence can represent the actions in the time sequence and is not affected by light, clothing, skin color, and so on. In recognition of the human body local limb movement with higher fine-grained requirements, the geometric characteristics between the bone points can be expressed naturally, and the movement has good interpretability. These characteristics make human action recognition based on skeletal point data an important direction in computer vision.

Action recognition based on bone points needs to design different classification algorithms according to different features of the extracted bone points. The relative motion of the human bones will produce the posture, so the relative position changes of the bone points can characterize the posture changes in the sequence. Moreover, the angle change of the connected bones indirectly reflects the change of the movement there. Therefore, the spatial structure used in this paper is the vector modulus ratio of the characteristic bone points and the vector angle of the bone points as the bone points.

Video retrieval is to search for useful or needed information from a large amount of video data. It is mainly based on the given example or the designed special diagnosis description to find out the video clips that meet the conditions in a large amount of video data. With the widespread use of video capture equipment, the amount of available video data is also increasing. If manual processing is used to process video data, it will cause a substantial increase in labor

costs. However, if a computer is used instead of labor to analyze the human motion behavior in the video, it can avoid excessive labor costs. Moreover, it can be faster, more comprehensive, and accurate in the labeling and indexing of videos, thereby helping people find important information of interest more conveniently.

2. Related Work

Most feature extraction models describe a certain part of the model, and it is difficult to extract and describe all functional models. Some research proposed a 3D shape descriptor and weight combination optimization scheme and used it to retrieve CAD models [1]. This scheme judges the similarity between models based on the weighted sum of the $L1$ distance on the distance histogram corresponding to the descriptor and uses the weight combination optimization scheme to improve the retrieval accuracy of the 3D model database. Some research proposed three combined feature descriptors and one class-based feature descriptor and applied them to model retrieval [2, 3]. Some research introduced coupled machining feature clusters to represent the three-dimensional CAD model as a structured model with machining features as a carrier and uses multilevel feature descriptors to establish a machining feature similarity evaluation model. Some research used fuzzy clustering of some physical descriptors to generate a multiscale index of the 3D model database for fast matching.

Some research proposed an ontology-based 3D CAD model retrieval algorithm. The CAD model is divided into several related subcomponents, and semantic description and annotations are added to complete the similarity evaluation between the models. Some research used hierarchical feature ontology and ontology mapping to generate semantic descriptors based on semantic descriptors for ontology reasoning so that the retrieval system can obtain better performance [4]. Some research used the ontology to describe the functional semantics of the CAD model and semiautomatically annotated the functional semantics of the CAD model according to the attribute adjacency graph, thereby establishing a CAD model library that supports functional semantic model retrieval according to the existing model retrieval technology and feature extraction method [5, 6]. Some research used deep learning technology to train model features and built a deep neural network classifier for the three-dimensional computer-aided design model. Some research used another way to extract the feature vector of the three-dimensional model and to predict and match the model, effectively assessing the difference between the two models [7]. In addition, many researchers began to convert three-dimensional models into two-dimensional graphics to extract the edge and shape information of the model to describe the characteristics of the model. Some research divided the surface adjacency graph into convex, concave, and flat areas, used the area attribute code to represent the surface area, and compared the area attribute codes to measure the similarity between the models. Some research used B-Rep information and label attribute adjacency graphs to represent 3D CAD models and used a two-stage filtering strategy combined with graphic code indexing to construct filters and verification frameworks to speed up the retrieval of 3D models [8, 9].

Many researchers have also begun to use this framework to optimize and improve traditional machine learning algorithms. Some research has studied clustering algorithms under the Hadoop cloud platform [10]. Some research used the MapReduce distributed framework to parallelize the improvement of the traditional ant colony algorithm, which makes the traditional ant colony algorithm faster and more effective when processing large-scale data sets [11, 12].

Some research used Newton's method to solve the beta distribution parameter algorithm and proposed a suitable initial value selection algorithm to enable the EM (Expectation Maximization) algorithm to effectively solve the parameters of the implicit regression model. Some research proposed an EM algorithm based on density detection, which selected the initial value based on density and distance to reduce the influence of the traditional EM algorithm's initial value selection on the convergence effect [13–15]. Some research proposed a fast and robust finite Gaussian mixture model clustering algorithm and applied an entropy penalty operator to the mixing coefficients of the model components and the probability coefficients of the components to which the samples belong to make the algorithm converge to a certain value in a few iterations. Some research used the greedy algorithm method and set appropriate thresholds for the hidden parameters so that the traditional EM algorithm can obtain the number of model components of the Gaussian mixture model in a few iterations without presetting the number of model components.

3. EM Algorithm

The EM algorithm is an iterative method and is used to solve the maximum likelihood estimation or maximum a posteriori estimation of parameters in the probability model, which can greatly reduce the computational complexity of solving the maximum likelihood estimation. The specific algorithm flow is as follows.

If we assume that the sample set is $X = \{x_1, x_2, x_3, \ldots, x_m\}$ and obeys the Gaussian mixture distribution, the probability density function $f_k(x)$ of the Gaussian mixture distribution is

$$f_k(x) = \sum_{j=1}^{k} w_j \Phi_j\left(x_i; \theta_j\right). \tag{1}$$

Among them, x_i is a P-dimensional vector, $\Phi_j(x_i; \theta_j)$ is the probability density of the Jth Gaussian model component, and θ_j is its parameter. w_j is the mixing coefficient of the jth component, describing the proportion of the sample covered by the jth Gaussian model component to the total sample, and k is the number of model components of the Gaussian mixture model.

$$\theta_j = \left(\mu_j, \sum_j\right). \tag{2}$$

μ_j is the mean value of Gaussian model components, and $\sum_j$ is the covariance matrix of Gaussian model components. Among them, the expression of Gaussian component probability density $\Phi_j(x_i;\theta_j)$ is

$$\Phi_j\left(x_i;\mu_j,\sum_j\right)=\frac{\exp\left\{-(1/2)\left(x_i-\mu_j\right)^T\sum_k^{-1}\left(x_i-\mu_j\right)\right\}}{(2\pi)^{p/2}\left|\sum_k\right|^{(1/2)}}. \tag{3}$$

The initial value $\mu_0, \sum_0, w_0$ is given, and steps E and M are repeated until the algorithm converges.

Step E: according to the initial value of the parameter θ or the parameter value obtained in the last iteration, the posterior probability of the recessive variable (i.e., the expectation of the recessive variable) is calculated as the current estimated value of the recessive variable:

$$Q_i\left(z^{(i)}\right)\coloneqq f\left(z^{(i)}|x^{(i)};\theta\right). \tag{4}$$

Step M: by calculating the maximum value of the likelihood function, new parameter values are obtained:

$$\theta\coloneqq \text{aeg}\max_\theta\sum_i\sum_{z^{(i)}}Q_i\left(z^{(i)}\right)\log\frac{f\left(x^{(i)},z^{(i)};\theta\right)}{Q_i\left(z^{(i)}\right)}. \tag{5}$$

Finally, the following results are obtained:

$$\begin{aligned}w_k^{t+1}&=\frac{\sum_{i=1}^n f\left(k|x_i,\theta^{t+1}\right)}{n},\\ \mu_k^{t+1}&=\frac{\sum_{i=1}^n f\left(k|x_i,\theta^{t+1}\right)x_i}{\sum_{i=1}^n f\left(k|x_i,\theta^{t+1}\right)},\\ \sum_k^{t+1}&=\frac{\sum_{i=1}^n f\left(k|x_i,\theta^{t+1}\right)\left(x_i-\mu_k^{t+1}\right)\left(x_i-\mu_k^{t+1}\right)^T}{\sum_{i=1}^n f\left(k|x_i,\theta^{t+1}\right)}.\end{aligned} \tag{6}$$

Among them, w_k^{t+1} is the k-th class weight, μ_k^{t+1} is the k-th class mean, $\theta t+1$ is the d-dimensional vector, and $\sum_k^{t+1}$ is the k-th class covariance matrix.

Finally, through calculation, the parameter values of the Gaussian mixture model are obtained, and each sample is found in the subordinate class to obtain the final clustering result.

4. Greedy EM Algorithm

Based on the original probability density function $f_k(x)$ of the Gaussian mixture distribution of the EM algorithm, when a new component $\delta(x;\theta)$ is brought into an existing k-component mixture density function $f_k(x)$, a new Gaussian mixture model density function is generated:

$$f_{k+1}(x)=(1-\alpha)f_k(x)+\alpha\delta(x;\theta). \tag{7}$$

Among them, α is the mixing coefficient of the newly added model components, $0<\alpha<1$.

Then, the newly generated log-likelihood function is

$$L_{k+1}=\sum_{i=1}^n\log\left[(1-\alpha)f_k(x)+\alpha\delta(x;\theta)\right]. \tag{8}$$

After the new Gaussian mixture model, including the mixture model components and the new components, is obtained, the mixture model $f_k(x)$ is set as unchanged. Therefore, the core of the greedy EM algorithm is to optimize the mixing coefficient α of the new model components and the parameters of the new components to calculate the highest value of the newly generated log-likelihood function L_{k+1}. Therefore, we first find a set of initial parameters μ_0, $\sum_0$, and α_0 of the new component through a global search. At α_0, the second-order Taylor formula of L_{k+1} is expanded, and the quadratic function of α is maximized to obtain an approximation of the likelihood function:

$$\widehat{L}_{k+1}=L_{k+1}(\alpha_0)-\frac{\left[L'_{k+1}(\alpha_0)\right]^2}{2L''_{k+1}(\alpha_0)}. \tag{9}$$

In the formula, L'_{k+1} and L''_{k+1} are the first and second differentials with respect to α. If we define

$$X(x;\theta)=\frac{f_k(x)-\delta(x;\theta)}{f_k(x)+\delta(x;\theta)}, \tag{10}$$

then the log-likelihood local optimum of L_{k+1} near $\alpha_0=0.5$ can be written as

$$\widehat{L}_{k+1}=\sum_{i=1}^n\log\frac{f_k(x_i)-\delta(x_i;\theta)}{2}+\frac{1}{2}\frac{\left[\sum_{i=1}^n\chi(x_i;\theta)\right]^2}{\sum_{i=1}^n\chi^2(x_i;\theta)}. \tag{11}$$

Thus, the formula for estimating $\widehat{\alpha}$ of the newly added model components can be obtained:

$$\widehat{\alpha}=\frac{1}{2}-\frac{1}{2}\frac{\sum_{i=1}^n\chi(x;\theta)}{\sum_{i=1}^n\chi^2(x;\theta)}. \tag{12}$$

Thus, the estimated value of the new model component is obtained, and then the optimal solution $\{\alpha_{k+1},\mu_{k+1},\sum_{k+1}\}$ of the new model parameter is extracted through the formula iteratively to extract the log-likelihood function value L_{k+1} of the new Gaussian mixture model.

The actual processing process of MapReduce is

$$\begin{aligned}&\text{Input}\longrightarrow\text{Map}\longrightarrow\text{Sort}\longrightarrow\text{Combine}\longrightarrow\text{Partition}\\ &\longrightarrow\text{Reduce}\longrightarrow\text{Output}.\end{aligned} \tag{13}$$

The specific workflow is shown in Figure 1.

For the Map phase, the algorithm begins to process data on the key-value pairs parsed in the Input phase. First, it initializes and then adds new model components to the single Gaussian mixture model initialized in each node. The new model component is a standard normal distribution with a mean value of 0 and a standard deviation of 1, and then the initialized mixing coefficient α_0 of the new model component is obtained. Among them, the initial parameter value of the lth node model is

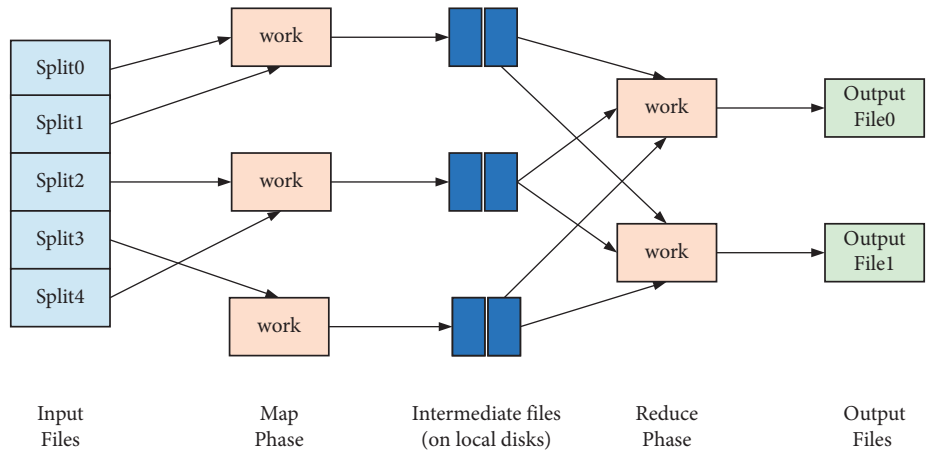

Figure 1: MapReduce workflow.

$$\begin{aligned} \mu_0^l &= E(x), \\ \sum_0^l &= \operatorname{cov}(x), \\ w_0^l &= 1, \quad 1 \le l \le h. \end{aligned} \tag{14}$$

Then, the new model component parameters μ_k^{t+1} and $\sum_k^{t+1}$ of each node are calculated to obtain the new model component, then the new model component mixing coefficient α_k^{t+1} is obtained, and the new Gaussian mixture model density function is obtained. The generated key value is used as the number of iterations, and the key-value pair with the Gaussian mixture model density as the value is output to the next stage.

Before output to the Reduce stage, there will be a series of intermediate processing. For the Sort stage, because the key value in the key-value pair is set to the number of iterations, this stage does not need to be considered too much. For the Partition stage, in this stage, the number of keys is the number of iterations, and the key-value pairs generated under the same iteration level have the same key value. Therefore, for each key-value pair generated in the Mapper phase, a Reduce job is assigned, the key and value values of the key-value pair are serialized into a byte array, and then the result is written into the buffer to wait for the call.

For the combine stage, the key-value pairs obtained by all nodes are integrated. Under the same number of iterations, the key values of the key-value pairs obtained from each node are the same, so the value values in these key-value pairs are integrated into a value set; namely,

$$\text{list}\langle \text{key}, \text{value} \rangle \longrightarrow \langle \text{key}, \text{list}\langle \text{value} \rangle \rangle. \tag{15}$$

Among them, key is the number of iterations, and value is the Gaussian mixture model density.

For the Reduce phase, the key-value pair generated in the combine phase is $\langle \text{key}, \text{list}\langle \text{value} \rangle \rangle$. That is, the Gaussian mixture model density function set in the key-value pair is summed, then the logarithm is taken, and the integrated log-likelihood function is obtained as

$$F_{k+1} = \sum_{i=1}^{n} \log \sum_{i=1}^{h} f_k^{t+1}(x). \tag{16}$$

Then, it is judged whether F_k satisfies the convergence condition, and the judgment factor is set:

$$\lambda = \left| \frac{F_k^{t+1}}{F_k^t} - 1 \right|. \tag{17}$$

When $\lambda < 10^{-6}$, if the algorithm satisfies the convergence condition, the algorithm proceeds to the next step of judgment. However, if it does not converge, the algorithm restarts the maximum likelihood estimation operation.

For the Output stage, it is judged whether the convergence condition F_k satisfies $F_{k+1} \le F_k$. If the convergence conditions are not met, new model components are added again. However, if the convergence condition is met, the output condition is met, and the result is output in the form of key-value pairs according to the format of the output file. If the value is F_k that satisfies the output condition, then the k value corresponding to F_k at this time is the number of output ideal model components. The final algorithm flowchart is shown in Figure 2.

5. Implementation of Distributed Greedy EM Algorithm

According to the programming model of MapReduce, corresponding functions are designed for program operation. In the realization of the distributed greedy EM algorithm, because the main process of the distributed greedy EM algorithm includes the reading of global data, the update calculation of Gaussian mixture model parameters, and the three aspects of iteration, the function design and implementation of these three main steps are carried out.

5.1. Reading of Global Data. In the algorithm of this paper, the data needs to be globally searched and initialized, and all the data in the distributed file system can be obtained through the setup function in the Mapper class. Since the

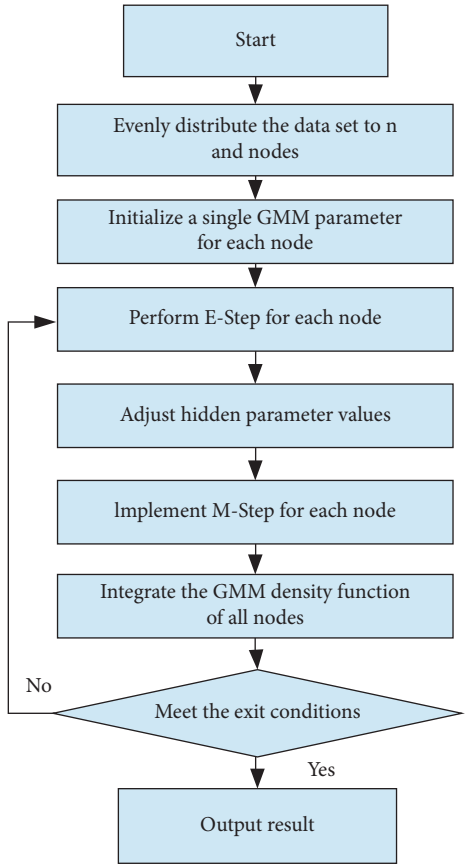

Figure 2: Flowchart of distributed greedy EM algorithm.

distributed greedy EM algorithm needs to obtain global data variables after each iteration, it can be implemented through the setup function in the MapReduce programming model. For the distributed greedy EM clustering algorithm, the Map task is completed through each data node, and the parameter values of the Gaussian mixture model are obtained. The setup function can obtain data files stored in HDFS while ensuring that each data node shares the same variables. Because the global variables mentioned are sometimes required to be acquired and used, the Map function can be defined after the setup function is defined. First, the path of the model parameters of the distributed greedy EM algorithm is obtained and stored in the Configuration object. For the first iteration, the initial parameter values of the model are obtained from the path stored in the Configuration object. Then, when the subsequent iterations are performed, the output path is obtained from the Reduce function in the previous iteration.

Then in the Mapper class, the setup function is reassigned, and the corresponding data is obtained from the global file from the Configuration object. After the reading is completed, the parameter values are read into the cluster object through the Buffered Reader function, thus completing the reading of the global data.

5.2. Update of Gaussian Mixture Model Parameters. When the parameters are updated through the maximum likelihood estimation of the Gaussian mixture model, the parameters include the mixing coefficient, mean value, and covariance matrix of the new model components. Among them, the covariance matrix needs to obtain the mixing coefficient and mean value of the newly added model components. Therefore, two functions need to be designed to update them separately. The constant MapReduce function is defined to obtain the mixing coefficient and mean value of the new model components, and the matrix MapReduce function is defined to obtain the covariance matrix and output all parameter values.

For the constant MapReduce function, every time the mixing coefficient and mean value of the new model component are performed, all the parameter values obtained from the previous iteration (the mixing coefficient, mean value, and covariance matrix of the new model component) are required. Therefore, the data in the cluster global object is read in the step function. However, for the matrix MapReduce function, the mixing coefficient and mean value of the new model components obtained by the constant MapReduce function update this time are required. Moreover, the covariance matrix obtained this time will be used as the input of the constant MapReduce function of the next iteration, so it is necessary to use the step function to read the mixing coefficient, mean, and covariance matrix of the new model component obtained this time into the cluster_new object.

5.3. Iteration of Distributed Greedy EM Algorithm. When the distributed greedy EM algorithm is iterative, it needs to store the output result of the previous MapReduce in HDFS and use it as the input of the subsequent iterative MapReduce. Moreover, after an iteration is completed, the corresponding median data needs to be deleted. Therefore, the iterative process when designing the distributed greedy EM algorithm is shown in Figure 3.

Our work used the bat algorithm to find the optimal face matching sequence. In the process of bats searching for the structural similarity matrix FC of the source model and the target model, in order to make the bat move to the best position, a corresponding fitness value is set for each bat. The position solution sequence of the ith bat in the structural similarity matrix FC is $(1, j(1)), (2, j(2)), \ldots, (i, j(i)), \ldots, (m, j(m))$. Among them, $j(i)$ represents the target face number $i = 1, 2, \ldots, m$ that matches the ith face of the source model. The calculation process of the fitness function of the ith bat is given, as shown in the following formula:

$$f(x_i) = \sum_{t=1}^{m} S[t, j(t)]. \tag{18}$$

In the process of searching for the optimal solution, the points acquired by the bat are discrete. In a *d-dimensional* search space, the size of the population is N. x_i represents the position of the ith bat, that is, a two-dimensional sequence pair composed of the row and column indices of the structural similarity matrix, and v_i represents the flight speed of the ith bat. At time t, the speed of bat i is updated as follows:

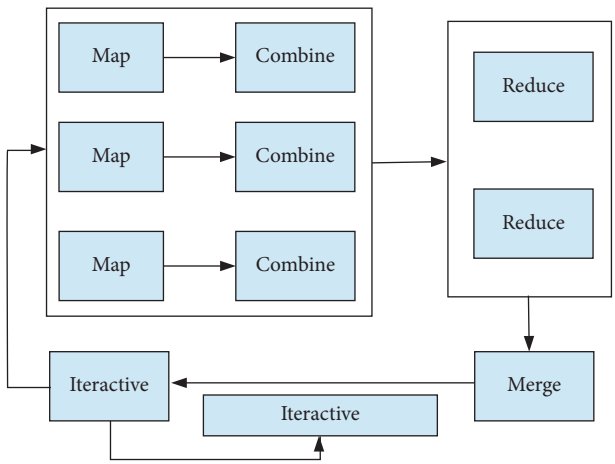

Figure 3: Iterative process of distributed greedy EM algorithm.

$$v_i^t = v_i^{t-1} + \text{round}\left[\left(x_i^{t-1} - x^*\right)Q(i)\right]. \tag{19}$$

Among them, round represents the rounding operation, and $Q(i)$ is the frequency of the ith bat, as shown in the following formula:

$$Q(i) = Q_{\min} + \left(Q_{\min} - Q_{\max}\right) * \text{rand}. \tag{20}$$

rand is a uniformly distributed random number between $[0, 1]$, and x^* is the current optimal position solution. Taking source model A and target model B as examples, using the above formula, a set of velocity solutions about the current ith bat can be obtained. From the sequence of these solutions, a set of sequence solutions is obtained to simulate the whole process of bats searching for the optimal position sequence.

A set of solution sequences are randomly obtained to represent the current optimal position solution x^*:

$$(1,2), (2,3), (3,4), (4,5), (5,1), (6,9), (7,8), (8,6), (9,7). \tag{21}$$

The velocity solution sequence of the ith bat at $t-1$ is

$$(1,6), (2,-2), (3,8), (4,0), (5,4), (6,-16), (7,-10), (8,6), (9,6). \tag{22}$$

The velocity solution sequence of the current ith bat at time t can be obtained:

$$(1,8), (2,-3), (3,10), (4,0), (5,4), (6,-21), (7,-12), (8,8), (9,8). \tag{23}$$

The speed of the ith bat at time t is v_i^t, and its position at time $t-1$ is x_i^{t-1}. The calculation process of x_i^t is as follows:

$$x_i^t = \text{round}\left(x_i^{t-1} + x_i^t\right). \tag{24}$$

Among them, round represents the rounding operation.

The position solution sequence of the ith bat at time $(t-1)$ is

$$(1,2), (2,3), (3,4), (4,5), (5,1), (6,9), (7,8), (8,6), (9,7). \tag{25}$$

The position solution sequence of the ith bat at time t can be calculated:

$$(1,10), (2,0), (3,14), (4,5), (5,5), (6,-12), (7,-4), (8,14), (9,15). \tag{26}$$

The obtained bat position solution sequence is processed, and the processing process is as follows:

Step 1. The remainder operation of the surface sequence (mod) is used to solve the situation that the second component of the bat position solution is out of bounds, and the position solution is obtained again.

The algorithm uses the following formula to take the remainder and reacquire the current bat position solution. Among them, mod is the remainder function.

$$x_i^t = \text{mod}\left[x_i^t, (m+1)\right]. \tag{27}$$

The mod operation is used to update the position solution sequence of the ith bat at time t, and the result is

$$(1,0), (2,0), (3,4), (4,5), (5,5), (6,8), (7,6), (8,4), (9,5). \tag{28}$$

After the mod operation, the second component of the bat position solution may appear 0. If 0 appears, the algorithm executes Step 2; otherwise, the algorithm executes Step 3.

Step 2. The second component of the processed position solution is 0 (replace operation). The random sequence of the face number m of the source model is obtained, and the value of the second component that does not appear in the position solution after the remainder is found. After that, the found value is used to fill in the 0 in the second component of the position solution after the remainder.

The replace operation is shown in Figure 4. The black solid point on the left is the position solution sequence after taking the remainder:

$$(1,0), (2,0), (3,4), (4,5), (5,5), (6,8), (7,6), (8,4), (9,5). \tag{29}$$

The white solid point on the right is the position solution sequence obtained after the replacement operation. At this time, the position solution sequence of the bat is

$$(1,1), (2,2), (3,4), (4,5), (5,5), (6,8), (7,6), (8,4), (9,5). \tag{30}$$

Step 3. Delete the repeated position solution of the second component (unique operation). After the remainder operation, the second component of the position solution will have repeated values. At this time, it is time to delete the duplicate position solution.

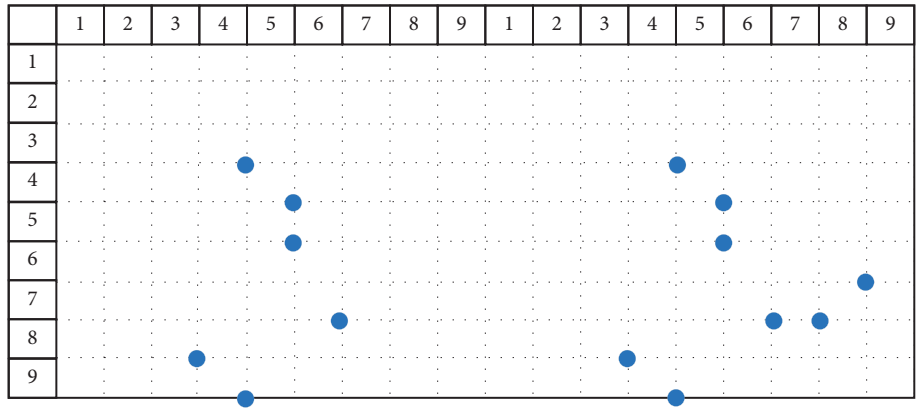

Figure 4: Replace operation.

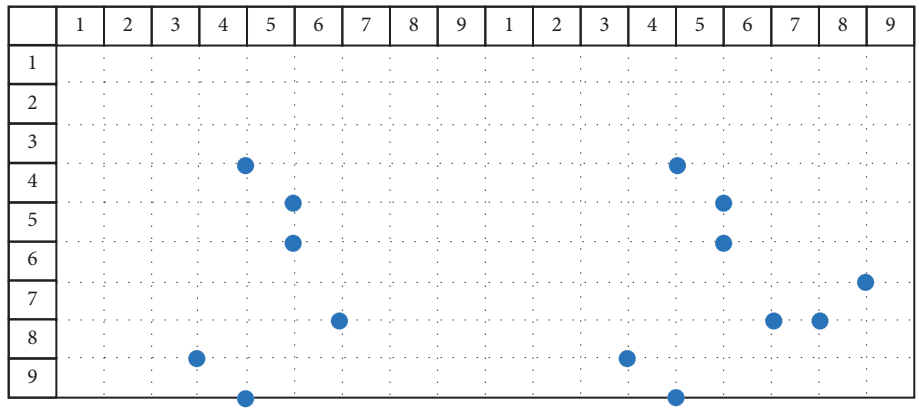

Figure 5: Unique operation.

Using the following formula, the position solution of the deleted second component is repeated, where unique is the deletion function:

$$x_i^t = \text{unique}\left(x_i^t\right). \tag{31}$$

The unique operation is shown in Figure 5. The left side is the current position solution sequence:

$$(1,1),(2,2),(3,4),(4,5),(5,5),(6,8),(7,6),(8,4),(9,5). \tag{32}$$

After executing the unique operation, the position solution sequence becomes

$$(1,1),(2,2),(3,4),(4,5)(6,8),(7,6). \tag{33}$$

The deleted location solution is

$$(5,5),(8,4),(9,5). \tag{34}$$

That is, the position solution where the second component is repeated, as shown in Figure 5.

Step 4. Fill position solution sequence (fill operation). From the position solution sequence after a unique operation, the algorithm obtains all the first components to form a set FSet. The algorithm obtains all the second components to form the set SSet.

The algorithm obtains a set of random number sequences of the source model face number m and finds all the values that do not appear in FS to form the first component set FS_1. After that, the algorithm obtains a set of random number sequences of the target model face number n and finds all the values that do not appear in SS to form the second component set SS_1. From FS_1 and SS_1, the algorithm randomly selects a numerical value to form a position solution and adds it to the position solution sequence until FS_1 and SS_1 are empty.

The fill operation process is shown in Figure 6, and the left side represents the position solution sequence to be processed:

$$(1,1),(2,2),(3,4),(4,5),(6,8),(7,6). \tag{35}$$

The white solid dot on the right represents the new position solution. The transformed position sequence is

$$(1,1),(2,2),(3,4),(4,5),(5,3),(6,8),(7,6),(8,7),(9,9). \tag{36}$$

Since the points in the structural similarity matrix between the source model and the target model are all discrete

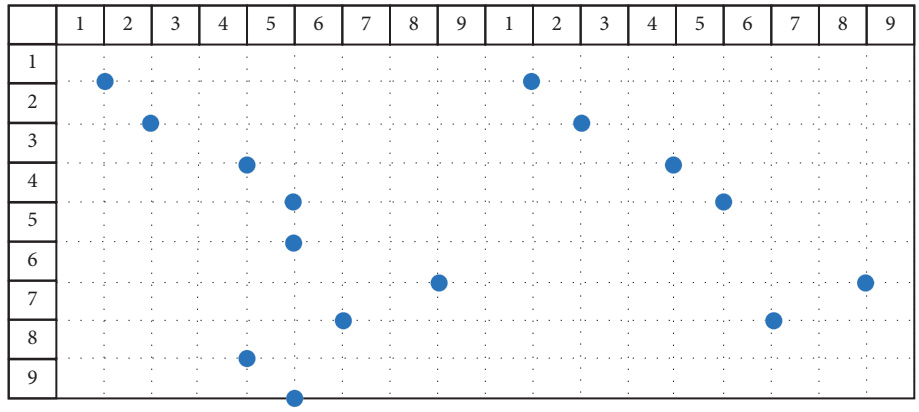

Figure 6: Fill operation.

points, when the bat algorithm searches this matrix, the position solution obtained is also discrete. This requires processing these position solutions to reacquire the current optimal position solution sequence. The face matching process based on the bat algorithm is as follows:

Step 1. The algorithm calculates the shape similarity and neighborhood structure similarity between the source model surface and the target model surface. Moreover, the algorithm constructs a structural similarity matrix FC between the source model and the target model.

Step 2. Initialize the population. The algorithm sets the population size N, the number of iterations count, the maximum number of iterations Maxk, the initial loudness A_i^0 of the ith bat at the moment $t = 0$, the pulse rate r_i^0, and the maximum value $Q_{\max}$ and minimum value $Q_{\min}$ of the pulse frequency.

Step 3. The algorithm calculates the fitness function value $f(i)\,(i = 1, 2, \ldots, n)$ of the ith bat and obtains the smallest fitness function value $f_{\min}$ and its corresponding optimal position solution x^*.

Step 4. If

$$\text{count} = \text{Maxk}, \tag{37}$$

then the algorithm ends, and the current optimal position solution x^* is output; otherwise, count + +, and the algorithm goes to Step 5.

Step 5. The algorithm redefines various values and rounds the speed v_i^t and position x_i^t of the ith bat at time t.

Step 6. The algorithm performs mod processing on x_i^t and judges whether there is a 0 solution in the surface sequence position solution. If there is 0 solution, the algorithm goes to Step 7; otherwise, the algorithm executes Step 8.

Step 7. Through the replace operation, the algorithm handles the 0 solution problem in the position solution sequence.

Step 8. The algorithm performs unique processing to delete the repeated position solutions in the current surface sequence.

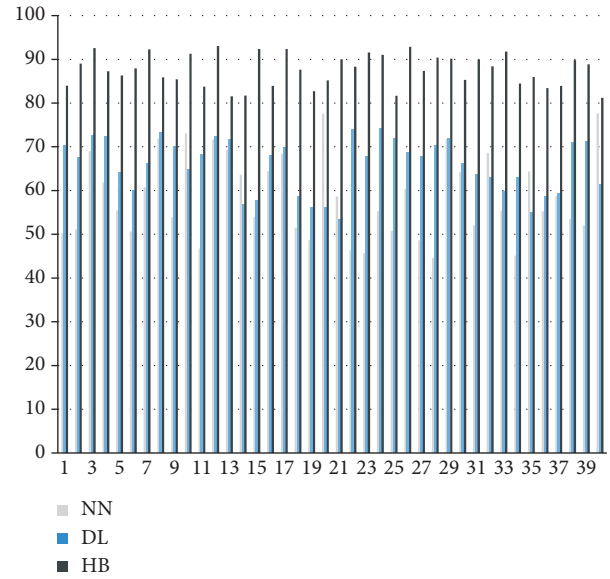

Figure 7: Comparison diagram of the accuracy rate of sports action feature recognition.

Step 9. Through the fill operation, the algorithm handles the empty solution problem in the surface sequence position solution and obtains a new undetermined solution x_{new}.

Step 10. The algorithm generates a random number rand1. If r and $1 > r_i^0$, the position of the best bat in the current group shifts to the next position, the algorithm goes to Step 6; otherwise, the algorithm goes to Step 11.

Step 11. The algorithm generates a random number rand2. If rand2 $< A_i^0$ and $f(x_{\text{new}}) < f(i)$, then $f(i) = f(x_{\text{new}})$ and $x_i^{t-1} = x_{\text{new}}$; otherwise, $f(i)$ and x_i^{t-1} do not change.

Step 12. If $f(x_{new}) < f_{\min}$, then $x^* = x_{\text{new}}$; otherwise, x^* does not make any changes, and the algorithm returns to Step 4.

The bat algorithm is used to search the structural similarity matrix, and the final optimal position solution vector

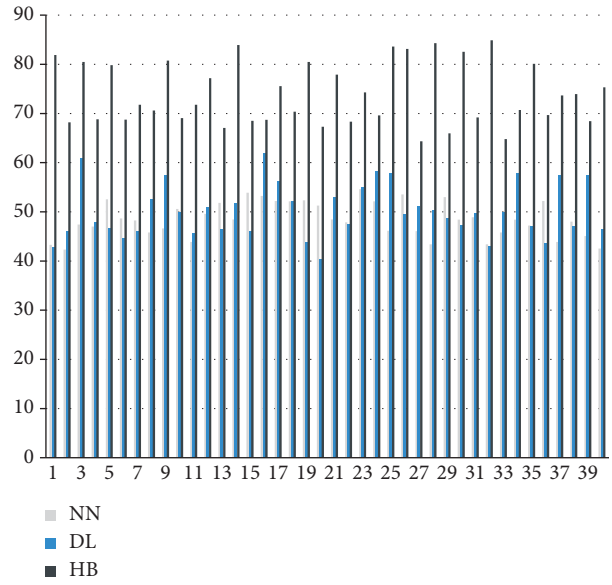

Figure 8: Comparison diagram of standard evaluation of actions.

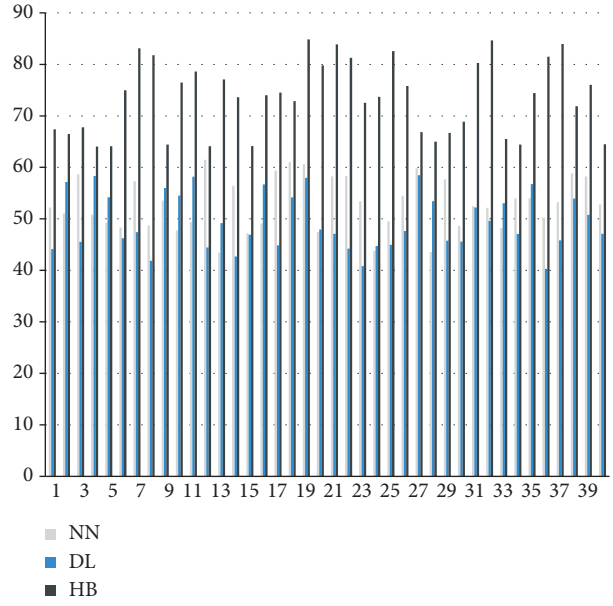

Figure 9: Comparison diagram of scores of action correction opinions.

is $(j(1), j(2), \ldots, j(m))$. It can be known that the matching sequence between the source model and the target model is $(1, j(1)), (2, j(2)), \ldots, (m, j(m))$.

5.4. The Analysis Effect of the Model on Sports Movements. The scientific analysis of sports actions is carried out through the model constructed above, the algorithm in this paper is named HB algorithm, and the algorithm performance of the algorithm proposed in this paper is compared with neural network algorithm (NN) and deep learning algorithm (DL). First, this paper compares the effects of sports action feature recognition and compares 40 sets of actions. The results are shown in Figure 7.

It can be seen from the above chart that the hybrid model constructed in this paper performs well in recognition of sports action features, while the recognition rates of traditional neural network algorithms and deep learning algorithms are low. Next, this paper analyzes the scientific evaluation of the action by the algorithm love, the results are expressed by the scoring method, and the action correction opinions are scored. The results are shown in Figures 8 and 9.

It can be seen from the above figure and table that the algorithm in this paper performs very well in action evaluation and action correction, which is much higher than the traditional algorithm. It can be seen that the algorithm in this paper has a certain effect.

6. Conclusion

This paper combines the greedy algorithm and the bat algorithm to construct an intelligent model that can be applied to sports action analysis. Moreover, this paper designs and implements CAD model retrieval methods based on greedy algorithm and bat algorithm. In addition, this paper focuses on the retrieval process based on the greedy algorithm and the bat algorithm and compares and analyzes the advantages and disadvantages of the two algorithms in model retrieval based on experimental data. The results show that the bat algorithm is feasible in model retrieval, and the bat algorithm is better than the greedy algorithm when measuring the subtle differences between CAD models. The pros and cons of the order are used to judge the degree of similarity between the model faces and apply it to the structural similarity matrix between the models. At the same time, this paper establishes the corresponding mathematical model for the shape and structure of the model and constructs the shape similarity matrix, the surface neighborhood structure similarity matrix, and the structure similarity matrix between the source model and the target model. Finally, this paper designs experiments to verify the performance of the model. From the research results, it can be seen that the model constructed in this paper has a certain effect.

Data Availability

The data used to support the findings of this study are available from the author upon request.

Conflicts of Interest

The author declared no conflicts of interest.

Acknowledgments

This work in this paper was supported by Jinhua Polytechnic.

References

[1] Y. Chai, "In-sensor computing for machine vision," *Nature*, vol. 579, no. 7797, pp. 32-33, 2020.

[2] B. Tippetts, S. Fowers, K. Lillywhite, D.-J. Lee, and J. Archibald, "FPGA implementation of a feature detection and tracking algorithm for real-time applications," in *Proceedings of the 3rd International Conference on Advances in Visual Computing (ISVC'07)*, Lake Tahoe, Nev, USA, November 2007.

[3] H. C. Garcia, J. R. Villalobos, and G. C. Runger, "An automated feature selection method for visual inspection systems," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no. 4, pp. 394–406, 2006.

[4] Y. Zhang, J. Zhao, and H. Han, "A 3D machine vision-enabled intelligent robot architecture," *Mobile Information Systems*, vol. 2021, Article ID 6617286, 11 pages, 2021.

[5] B. Tippetts, K. Lillywhite, S. Fowers, A. Dennis, D.-J. Lee, and J. Archibald, "A simple, inexpensive, and effective implementation of a vision-guided autonomous robot," in *Proceedings of the Intelligent Robots and Computer Vision XXIV: Algorithms, Techniques, and Active Vision, 63840P*, Boston, Mass, USA, October 2006.

[6] V. Bonato, E. Marques, and G. A. Constantinides, "A parallel hardware architecture for scale and rotation invariant feature detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 12, pp. 1703–1712, 2008.

[7] L. Chang and J. Hernández-Palancar, "A hardware architecture for SIFT candidate keypoints detection," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Lecture Notes in Computer Science*, pp. 95–102, Springer, Berlin, Germany, 2009.

[8] W. Gu and Y. Liu, "Description and reasoning of object-orientation-based direction relation in three-dimensional space," *Journal of Computational Information Systems*, vol. 7, no. 12, pp. 4433–4440, 2011.

[9] H. X. Hao, P. L. Zhang, and L. I. Songb, "3DR44 direction relation representation model in three dimensional space," *Computer Engineering*, vol. 37, no. 1, pp. 74–80, 2011.

[10] P. Ramya and C. Sundar, "SecDedoop: secure deduplication with access control of big data in the HDFS/hadoop environment," *Big Data*, vol. 8, no. 2, pp. 147–163, 2020.

[11] A. Erraissi, A. Belangour, and A. Tragha, "Digging into hadoop-based big data architectures," *International Journal of Computer Science Issues (IJCSI)*, vol. 14, no. 6, pp. 52–59, 2017.

[12] S. Dolev, P. Florissi, E. Gudes, S. Sharma, and I. Singer, "A survey on geographically distributed big-data processing using MapReduce," *IEEE Transactions on Big Data*, vol. 5, no. 1, pp. 60–80, 2019.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via theEMAlgorithm," *Journal of the Royal Statistical Society: Series B*, vol. 39, no. 1, pp. 1–22, 1977.

[14] K. Davies, S. Pal, and J. A. Siddiqua, "Stochastic EM algorithm for generalized exponential cure rate model and an empirical study," *Journal of Applied Statistics*, vol. 48, no. 12, pp. 2112–2135, 2021.

[15] H. Okamura and T. Dohi, "Application of EM algorithm to NHPP-based software reliability assessment with generalized failure count data," *Mathematics*, vol. 9, no. 9, p. 985, 2021.