Hindawi
Security and Communication Networks
Volume 2021, Article ID 6974757, 9 pages
https://doi.org/10.1155/2021/6974757

Research Article

Multicamera Calibration Optimization Method Based on Improved Seagull Algorithm

Shuai Du, Jianyu Wang, and Jia Guo

School of Automation, Nanjing University of Science and Technology, Nanjing 210094, China

Correspondence should be addressed to Shuai Du; dushuai@njust.edu.cn

Received 12 August 2021; Accepted 1 December 2021; Published 21 December 2021

Academic Editor: Xuyun Zhang

There are some problems in the process of camera calibration, such as insufficient accuracy and poor accuracy. Based on the seagull algorithm, the adaptive differential evolution algorithm is combined with the seagull algorithm to optimize the multicamera calibration. The seagull algorithm can achieve good results on multiparameter problems and effectively avoid falling into local optima. In this paper, the adaptive differential search algorithm is adopted to improve the local search ability and optimize the local search and global search ability. According to Zhang Zhengyou's method, the calibrated parameter is obtained, in which the parameter is used as the initial value. Then, taking the minimum mean error as the criterion, the improved seagull algorithm (SOA-SaDE) is used to establish the objective function, and the internal parameters and distortion coefficient of the camera are further solved. Verification experiments showed that the fusion algorithm has less reprojection error and higher calibration accuracy gull algorithm.

1. Introduction

Multiple cameras are widely used in various fields. Multicamera fusion can provide a wider range of real-time scene information. As an important tool of machine vision, improving camera calibration accuracy is the focus of its research. It has more accurate camera calibration and is more conducive to the camera image matching, recognition positioning, and other follow-up operation accuracy.

At present, the field of deep-sea exploration is developing in a deeper and farther direction. Underwater robots are widely used in underwater exploration due to their wide range of activities and strong autonomy. Compared with other underwater detection methods, robot visual observation is at a close range. Object detection has unparalleled advantages. It can not only track and detect underwater targets in real time but also record underwater video materials for researchers to study deep-sea objects after landing. Therefore, improving the accuracy of camera calibration and eliminating image distortion are of great significance to the research in the deep-sea field.

The camera calibration uses the calibration object in the space and obtains the calibration object different space positions through the position change. The mathematical model can be constructed by the relation between the object space coordinate system and the camera image coordinate system, Then, the internal parameters and distortion coefficients of the camera are calculated and solved.

Camera calibration is a multidimensional nonlinear problem. Each calibration image corresponds to different external parameters. Therefore, it is very difficult to find the concrete calculation equation in the process of back-projection to the calibration point. Common optimization algorithms, such as the pseudo-Newton method, cannot be applied to the optimization of camera calibration because they must depend on the specific functional form. Therefore, the researchers focus on using optimization algorithm to optimize the camera calibration results. Huang et al. [1] used the classical particle swarm optimization (PSO) algorithm to optimize the camera calibration results, took the absolute value of average relative error as the objective function, optimized the camera parameters, and improved the calibration accuracy. Qin et al. [2] proposed a full-parameter

adaptive mutation PSO. They used the algorithm to optimize the camera intrinsic parameters and improve the adaptive mutation rate of the particles according to the average particle distance of the particle swarm to optimize the camera calibration results. Xu et al. [3] introduced a diffusion mechanism to improve the local optimal solution problem of particle swarm optimization. Xiang et al. [4] proposed a calibration method based on depth learning. It can be calibrated by inputting the coordinates of the original image by improving the approximation ability of the DNN network. It can be used in large areas, multiple camera angles, and other complex environments. However, deep learning network training requires GPU acceleration and requires high computer configuration, and training takes time, so it is difficult to calibrate quickly. Lei et al. [5] combined PSO with simulated annealing (SA) algorithm, obtained the initial parameters of camera calibration by least squares, and optimized the camera parameters by the hybrid algorithm. This method improves the calibration accuracy of the camera. Based on the Levenberg–Marquardt algorithm, Liu Jiachen [6] corrected the reprojection error by using the improved beam adjustment method to reduce the error of 3D reconstruction. However, in this process, the formula is tedious and the computation is complex.

The seagull algorithm is a kind of metaheuristic, which is suitable for solving multiparameter optimization problems. When the traditional seagull algorithm is used, the initial seagull swarm has strong randomness, but the optimization process is inefficient and easy to fall into local optimization. A hybrid optimization algorithm is proposed in [7]. The algorithm is based on chaotic differential evolution and distribution estimation, which can obtain a high-precision solution. A hybrid algorithm of adaptive gravity search and differential evolution (DE) is proposed in [8]' which keeps the diversity of the population. The DE is used for local search and plays a big role. Inspired by the successful application of the above hybrid algorithm, seagull algorithm used in this paper, and seagulls algorithm compared to traditional optimization algorithm, the principle is simple and easy to implement, is suitable for multiobjective optimization, and does not coincide with the position of the population in an iterative process, reducing repetitive iterations and improving the effectiveness of iterations. To avoid the local optimization in the calibration calculation, this paper combines the seagull algorithm and the adaptive differential evolution algorithm and improves the seagull algorithm by absorbing the strong local searching ability of ADE, improving the accuracy and stability of camera calibration.

This article focuses on the key issues that need to be solved for camera calibration. Aiming at the problem of low calibration accuracy and obvious reprojection errors, a fusion algorithm (SOA-SaDE) is proposed based on the seagull algorithm and the adaptive differential algorithm. The internal parameters and distortion coefficients calibrated based on the pinhole camera model are optimized by the SOA-SaDE algorithm. Experiments show that the algorithm proposed in this paper effectively reduces the error of camera reprojection; the reprojection error is reduced by 63.03%, which is 16.75% higher than the effect of the seagull algorithm, and provides a feasible method for reducing the camera reprojection error.

2. Basic Principles of the Seagull Algorithm

In 2018, the seagull algorithm (SOA) proposed a new population-based intelligent optimization algorithm, which simulates the migrating and foraging behavior of seagulls to optimize the target [9].

The seagull algorithm is divided into two parts; they are migration and foraging. Migration is the behavior that is the movement of seagulls from their current position to a more livable position. Migration behavior affects the global exploration ability of the seagull algorithm. Foraging is the behavior of seagulls attacking the food in the current sea area during the flight. The foraging behavior affects the ability of the seagull algorithm for local exploitation.

There are three important points to be paid attention to during a Gull's migration from one place to another: avoiding collisions between individuals, the best orientation of its position, and its proximity to the best position. To avoid the collision with the seagulls, the algorithm uses the additional variable A to adjust the seagulls' position:

$$\overrightarrow{C_s} = A \times \overrightarrow{P_s(\mathbf{x})}, \tag{1}$$

where A represents the migration behavior of seagulls in each given search space The size of A is controlled by B:

$$A = f_c - \left(\left(\mathrm{t} \times \left(\frac{f_c}{\mathrm{Max}_{\mathrm{iteration}}}\right)\right)\right). \tag{2}$$

The final size of A decreases linearly from $2 \longrightarrow 0$ according to the number of iterations. After ensuring that individual gulls do not collide with each other, move all gulls closer to the best:

$$\overrightarrow{M_s} = B * \left(\overrightarrow{\mathbf{P_{best}}(\mathbf{x})} - \overrightarrow{P_s(\mathbf{x})}\right), \tag{3}$$

where $\mathbf{M}_s$ represents the convergence direction of the individual toward the optimal seagull and B is an important parameter for balancing the exploration and development capability of the algorithm. It changes according to

$$B = 2 * A^2 * \mathrm{rand}, \tag{4}$$

where rand is a random number in the range [0,1].

After calculating the direction of convergence of each gull, each gull began to move toward this position:

$$\overrightarrow{D_s} = \left|\overrightarrow{C_s} + \overrightarrow{M_s}\right|, \tag{5}$$

where $\overrightarrow{D_s}$ is the position of the seagull.

Seagulls can constantly change their angle and speed of attack during the migration, when attacking prey, seagulls will carry out the spiral movement. The position of the

seagull in the 3D is

$$x = r * \cos(k), \tag{6}$$

$$y = r * \sin(k), \tag{7}$$

$$z = r * k, \tag{8}$$

$$r = \mathrm{u} * e^{kv}, \tag{9}$$

where k is a random number at $[0, 2\pi]$. The algorithm controls the spiral radius r by u and v, and they are usually 1. According to the new position of seagull, the updated formula of the whole position of the seagull is as follows:

$$\overrightarrow{P_s(x)} = \left(\overrightarrow{D_s} \times x \times y \times z\right) + \overrightarrow{\mathbf{P}_{\mathbf{best}}(\mathbf{x})}, \tag{10}$$

where $\overrightarrow{\mathbf{P}_s(\mathbf{x})}$ is the attack position of the seagull.

SOA: the flow of the algorithm is as follows:

Step 1: initialization parameter

Step 2: calculate the fitness value for each seagull and the objective function value

Step 3: calculate $\overrightarrow{\mathbf{D}_s}$ according to formulas (1)–(5)

Step 4: calculate according to formulas (6)–(10)

Step 5: update position information and fitness values for the best seagull, internation = internation + 1

Step 6: if internation > $\mathrm{Max}_{\mathrm{interation}}$, skip to Step 7, or slip to Step 3

Step 7: output the optimal seagull position and fitness value

3. Adaptive Differential Evolution Algorithm

3.1. Differential Evolution. Differential evolution algorithm (DE) is a kind of evolutionary algorithm (EA), which is a search strategy for solving polynomial fitting problems put forward by R. Storn and K. Price. This algorithm is based on genetic algorithm and other evolutionary ideas; its essence is to optimize the multiobjective and multidimensional space to achieve the overall optimal solution of the goal. The differential evolution algorithm retains the crossover, mutation, and copy operations in the genetic algorithm. It differs from GA in which the variation vector is generated by the parent difference vector, which crosses with the incidental individuals to generate new individuals and then selects among them. Therefore, it has a better iterative approximation effect than GA. The differential algorithm is divided into two stages: population initialization and iteration [10].

3.1.1. Population Initialization. Suppose that $G = 0, 1, 2 \ldots G_{\max}$ stands for evolution algebra. Then, the 1st individual in the population under the current algebra is represented as

$$\overrightarrow{X}_{\mathrm{i,G}} = \left(\mathbf{x}^1_{\mathrm{i,G}}, \mathbf{x}^2_{\mathrm{i,G}}, \ldots, \mathbf{x}^j_{\mathrm{i,G}}\right), \quad j = 1, 2, \ldots, D, \tag{11}$$

where D is the dimension of the individuals of the population. In population initialization, the initial population is required to cover the entire search space R^D, and the initialization formula is shown in (8):

$$x_{j,i,0} = x_{j,\min} + \mathrm{rand}_{i,j}[0,1] * \left(x_{j,\max} - x_{j,\min}\right), \tag{12}$$

where $\mathrm{rand}_{i,j}[0,1]$ is a uniformly distributed random number in the interval $[0, 1]$ and $x_{j,\min}$ and $x_{j,\max}$ are the lower and upper bounds of the individual optimization variables $\overrightarrow{X}$, j, respectively.

3.1.2. Differential Mutation Operation. In each iteration, three individual vectors are randomly selected from the population $\overrightarrow{\mathbf{X}}_{\mathbf{r1,G}}$, $\overrightarrow{\mathbf{X}}_{\mathbf{r2,G}}$, and $\overrightarrow{\mathbf{X}}_{\mathbf{r3,G}}$, and $\mathrm{r1} \neq \mathrm{r2} \neq \mathrm{r3}$, according to (13), a new individual $\overrightarrow{V}_{i,G}$ can be created; this individual is a variation vector:

$$\overrightarrow{\mathbf{V}}_{\mathbf{i,G}} = \overrightarrow{\mathbf{X}}_{\mathbf{r1,G}} + \mathrm{F} * \left(\overrightarrow{\mathbf{X}}_{\mathbf{r1,G}} - \overrightarrow{\mathbf{X}}_{\mathbf{r3,G}}\right), \tag{13}$$

where F is the variation scale factor, which is used to scale the difference vector to control the search step. In general, the variation scale factor F is in the [0,2] interval.

3.1.3. Cross Operation. In the crossover step, the algorithm adopts discrete crossover. The test vector $\overrightarrow{U}_{i,G}$ is generated by crossing mutation vector $\overrightarrow{\mathbf{V}}_{\mathbf{i,G}}$ and target vector $\overrightarrow{\mathbf{X}}_{\mathbf{r1,G}}$ according to the binomial method. The specific operation is shown in

$$u^i_{i,G} = \begin{cases} v^j_{i,G}, & \text{if } \mathrm{rand}_{i,j}[0,1] \le C_r, \\ x^j_{i,G}, & \text{otherwise}, \end{cases} \tag{14}$$

where C_r is the crossover probability factor, and the crossover operator can enhance the diversity of the population. The value of C_r is generally in the interval [0,1], which is a random number uniformly distributed in the interval [0,1]. In the j dimension, if the random generating number is less than C_r, the test vector inherits the variation vector and vice versa.

3.1.4. Select Operation. The selection process selects the more adaptable child from each iteration as the next generation by comparing the child with the corresponding parent based on the value of the fitness function; its selection method is shown as

$$\overrightarrow{\mathbf{X}}_{\mathbf{i,G}} = \begin{cases} \overrightarrow{\mathbf{U}}_{\mathrm{i,G}}, & \text{if } \left(\overrightarrow{\mathbf{U}}_{\mathbf{i,G}}\right) \le f\left(\overrightarrow{\mathbf{X}}_{\mathbf{i,G}}\right), \\ \overrightarrow{\mathbf{X}}_{\mathrm{i,G}}, & \text{otherwise}. \end{cases} \tag{15}$$

3.2. Adaptive Differential Evolution. According to formulas (13) and (14), F and C_r are two important parameters in DE, and the choice of their values will affect the optimization

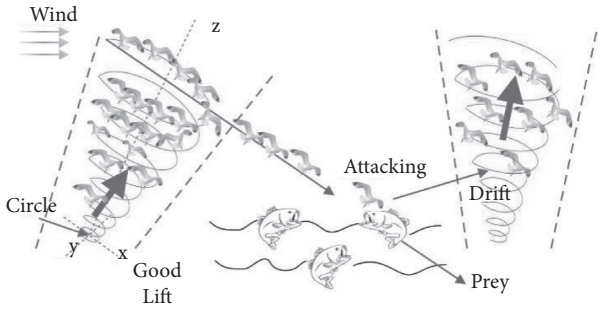

Figure 1: The biological principle of the seagull algorithm.

effect. However, in the DE algorithm, the values are all constant and cannot be well adapted to various problems, especially, for complex high-dimensional problems. Therefore, Janez Brest introduced adaptive control parameters in 2006. The improved algorithm is called the adaptive differential evolution algorithm (SaDE) [11]. The adaptive control parameters F and C_r are represented as

$$F_i^{t+1} = \begin{cases} F_l + \text{rand}_1 \times F_u, & \text{if rand}_2 \le \tau_1, \\ F_i^t, & \text{otherwise}, \end{cases}$$
$$CR_i^{t+1} = \begin{cases} \text{rand}_3, & \text{if rand}_4 < \tau_2, \\ CR_i^t, & \text{if rand}_4 \ge \tau_2, \end{cases} \tag{16}$$

where rand1, rand2, rand3, and rand4 is the random number in [0, 1], $\tau 1$ and $\tau 2$ indicate the probability of conversion, and F_l and Fu are the boundary scaling factor.

4. Camera Internal Parameter Optimization Design and Application Based on the Hybrid Algorithm

4.1. Design of Hybrid Algorithms. This paper proposes a combination of the SOA algorithm and the SaDE algorithm. It aims to improve the search precision, avoid the population falling into the local extremum, and maintain the population diversity in the later iteration. In the minimization problem, if the fitness of the iteration is greater than that of the previous generation, the location region of the iteration is not good, so the randomness of the population should be strengthened to improve the search range. If the fitness intelligence of the iterated individuals is less than that of the optimal individuals of the previous generation, then the region has the potential value, so we should continue searching along the region.

Combining the SOA algorithm with the SaDE algorithm to realize the internal parameter optimization, in each population iteration, the minimum fitness value of the population is calculated as $f_{\min}^i$. In the i+1 iteration, when $f_{\min}^{i+1} < f_{\min}^i$, SOA is used for optimization, when $f_{\min}^{i+1} > f_{\min}^i$, SaDE is used for optimization.

4.2. Establishment of the Objective Function. The objective function of the camera calibration problem is established as follows:

$$f = \sum_{i=1}^{N} \left\| p_{ij} - p\left(f_x, f_y, u_0, v_0, k_1, k_2, k_3, p_1, p_2, R, T\right) \right\|, \tag{17}$$

where p_{ij} is the actual pixel coordinates of the j corner, N is the number of corners, P is the calculated pixel coordinates, f_x, f_y, u_0, and v_0 are the camera's internal parameter, k_1, k_2, k_3, p_1, and p_2 are the radial distortion coefficient and the tangential distortion coefficient, and R and T are the rotation translation matrix of the image.

4.3. Parameter Initial Value Solution. The camera imaging relationship is

$$z_c \begin{bmatrix} \mathbf{x}_d \\ \mathbf{y}_d \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{f}_{c1}}{d_x} & \mathbf{0} & \mathbf{u}_0 & \mathbf{0} \\ \mathbf{0} & \frac{\mathbf{f}_{c2}}{\mathbf{d}_y} & \mathbf{v}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{T}_{3\times1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_w \\ \mathbf{y}_w \\ \mathbf{z}_w \\ \mathbf{1} \end{bmatrix}. \tag{18}$$

The above formula represents the transformation of point $[\mathbf{x}_w \ \mathbf{y}_w \ \mathbf{z}_w \ \mathbf{1}]^{\mathbf{T}}$ in the world coordinate system to point $[\mathbf{x}_d, \mathbf{y}_d, \mathbf{1}]^T$ in the pixel coordinate system in the linear model. M_A is the internal parameter matrix which represents the intrinsic geometry of the camera. The mathematical model is

$$\mathbf{M_A} = \begin{bmatrix} \mathbf{f}_x & \mathbf{0} & \mathbf{u_0} \\ \mathbf{0} & \mathbf{f}_y & \mathbf{v_0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}, \tag{19}$$

where $f_x = f_{c1}/d_x$ and $f_y = f_{c2}/d_y$, f_{c1} and f_{c2} are the focal length of the camera, d_x and d_y are the physical lengths of the pixels, and u_0 and v_0 are the intersections of the camera's optical axis and the image plane.

According to the above expression, the camera internal parameters can mainly solve 4 parameters. They are f_x, f_y, u_0, and v_0. We can obtain the initial value of f_x, f_y, u_0, and v_0 by the imaging relation. The initial value is obtained under the ideal condition, but the actual lens has distortion, so it needs to introduce distortion coefficient k_1, k_2, k_3, p_1, and p_2 to correct it.

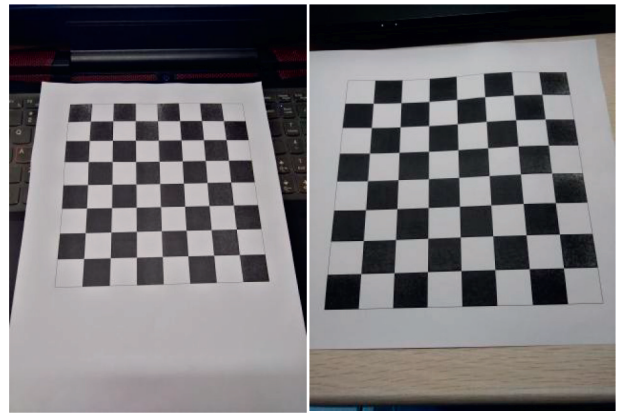

Figure 2: Calibration pictures.

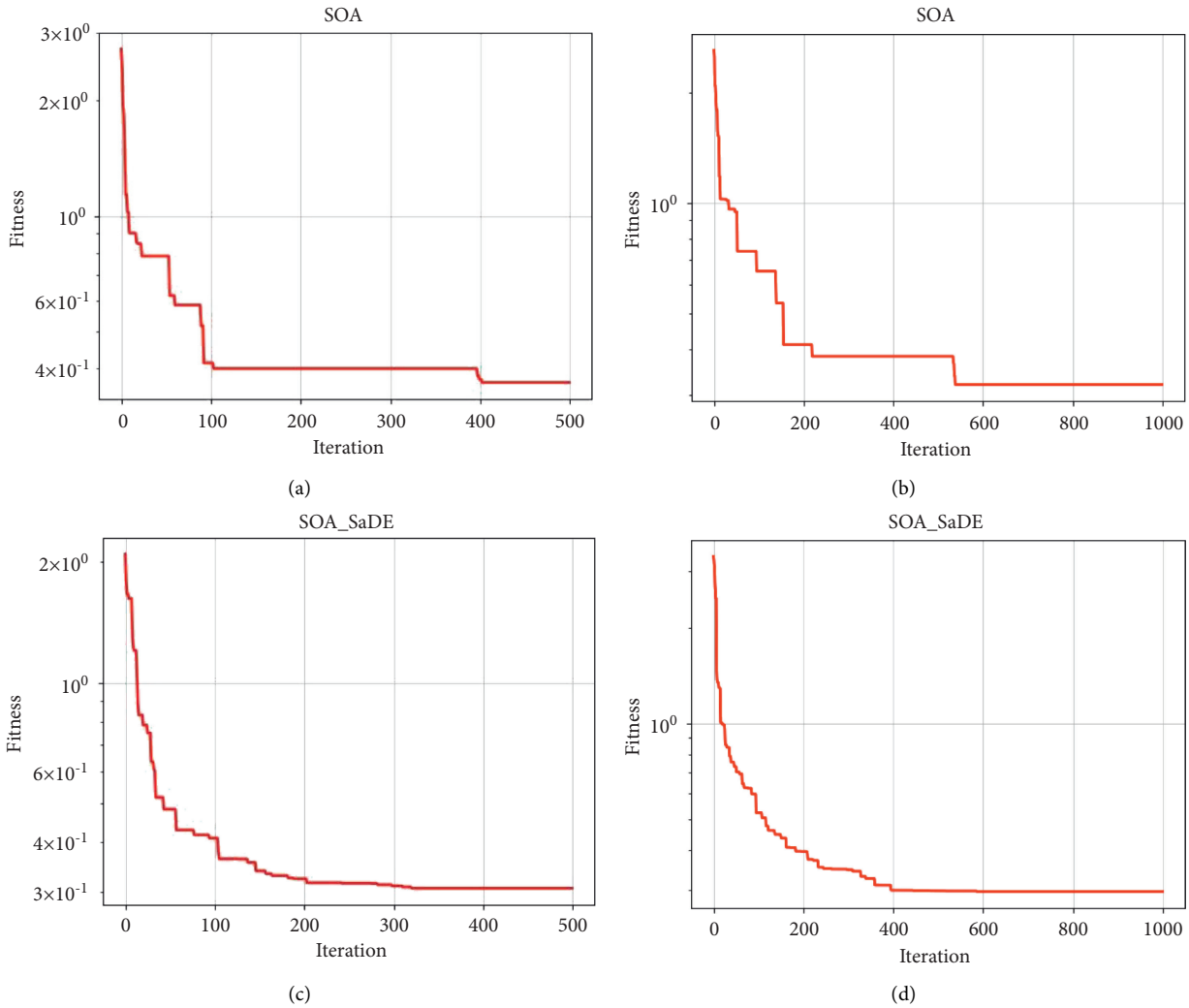

Figure 3: Object function curve. (a) SOA iterates 500 objective function curves. (b) SOA iterates 1000 objective function curves. (c) SOA-SaDE iterates 500 objective function curves. (d) SOA-SaDE iterates 1000 objective function curves.

Table 1: Calibration results of the Zhang Zhengyou calibration method.

Parameter	Zhang Zhengyou calibration method
f_x	3575.57253
f_y	3558.46976
u_0	1541.29510
v_0	1885.83655
k_1	0.14037782
k_2	−1.12437053
p_1	0.0079399
p_2	−0.00243523
k_3	1.45905723
Error	0.828057702

Table 2: Calibration results of camera internal parameters for 500 iterations.

Parameter	Method	
	SOA	SOA-SaDE
f_x	3572.85291	3640.19021
f_y	3556.14008	3614.61743
u_0	1546.41000	1540.28375
v_0	1886.40299	1888.24897
k_1	−0.00130089	−0.25517776
k_2	0.02047008	0.96152329
p_1	−0.00114037	0.00163706
p_2	−0.00115161	−0.00013975
k_3	−2.72448436	1.34834219

Table 3: Iterative calibration results of 1000 camera internal parameters.

Parameter	Method	
	SOA	SOA-SaDE
f_x	3581.17874	3623.59017
f_x	3564.41244	3601.84834
u_0	1544.83354	1538.79344
v_0	1886.52839	1888.07675
k_1	0.00972433	−0.23023645
k_2	−0.00264705	1.33036931
p_1	0.00050738	0.00148013
p_2	−1.09968486	0.00344624
k_3	0.00202380	−0.75344467

Table 4: Reprojection errors.

Number of iterations	Method	
	SOA	SOA-SaDE
500	0.36774533	0.30615074
1000	0.32056968	0.29615753

The mathematical model of radial distortion is

$$\begin{cases} x_d = x_u\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right), \\ y_d = y_u\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right). \end{cases} \tag{20}$$

The mathematical model of tangential distortion is

$$\begin{cases} x_d = x_u + \left[2p_1 y_u + p_2\left(r^2 + 2x_u^2\right)\right], \\ y_d = y_u + \left(2p_2 x_u + p_1\left(r^2 + 2y_u^2\right)\right). \end{cases} \tag{21}$$

In the above formula,

$$r^2 = x_u^2 + y_u^2. \tag{22}$$

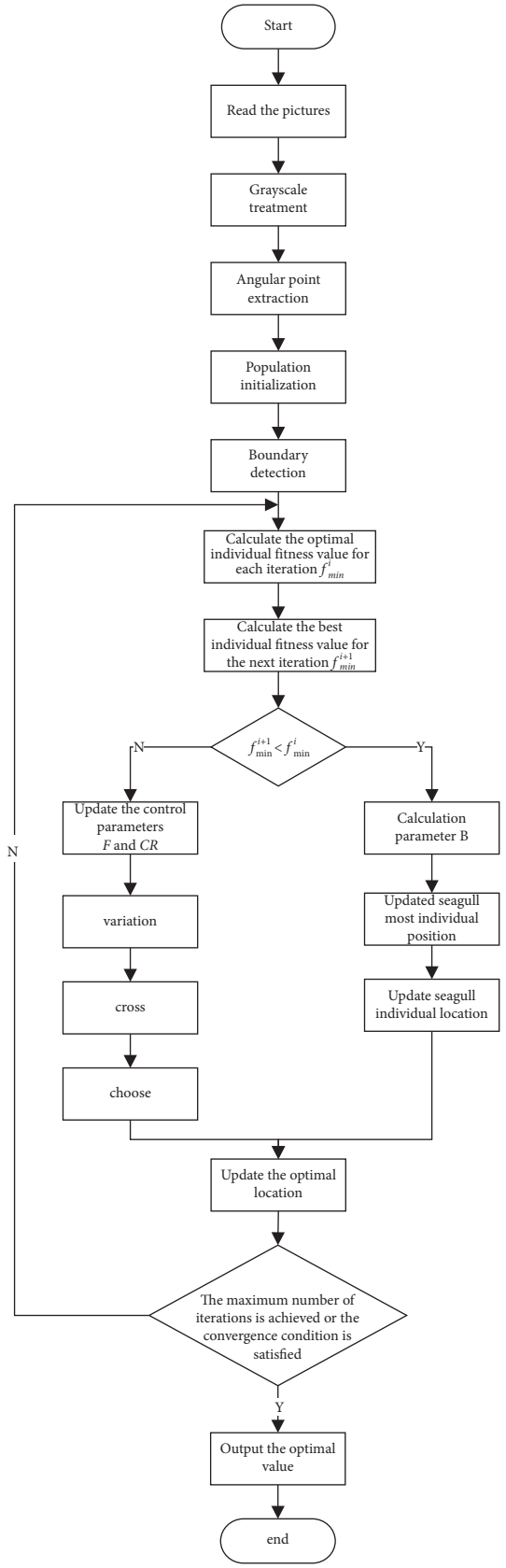

Figure 4: Algorithm flow of internal parameter optimization.

Contacting formulas (20)–(22), we can obtain

$$\begin{cases} x_d = x_u\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_1 x_u y_u + p_2\left(3x_u^2 + y_u^2\right), \\ y_d = y_u\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_2 x_u y_u + p_1\left(3x_u^2 + y_u^2\right), \end{cases} \tag{23}$$

where (x_d, y_d) is the image coordinate of the ideal camera model and (x_u, y_u) is the real image coordinate of the nonlinear camera model. The initial value under the distortion is obtained by formula (23).

4.4. Hybrid Algorithm Application. First, the initial value of the internal parameter $f_x, f_y, u_0, v_0, k_1, k_2, k_3, p_1$, and p_2 then initializes the seagull population to generate N different seagull individuals. Initializing population position, parameters A, B, and $\text{Max}_{\text{internation}}$, set the appropriate parameters $f_c = 2$, $u = 1, v = 1$, and initialize the current iteration number $t = 0$. The fitness function is obtained from the objective function and is defined as

$$\text{fitness} = \min \sum_{i=1}^{m} \sqrt{(x-u)^2 + (y-v)^2}, \tag{24}$$

where (x, y) is the actual pixel coordinates obtained by the corner extraction algorithm, (u, v) is the pixel coordinates calculated by camera imaging relations, and m is the total number of corners.

The algorithm flow of camera internal parameter optimization using the hybrid algorithm is shown in Figure 1.

5. Experiment

5.1. Experimental Design. The experiment uses camera RealSense D435i as the hardware platform and uses *Python* as the software development platform. In this experiment, 16 images were taken, and the camera internal parameters and distortion coefficients were calibrated based on these images. The pictures are shown in Figure 2.

5.1.1. Specific Calibration Steps

Step 1: the camera calibration is realized based on OpenCV-Python

Step 2: according to the calibration parameters obtained by the traditional method, the upper and lower interval of the parameters are set, the scope is limited, and the parameters are initialized

Step 3: the results of 300, 500, and 1000 iterations in the seagull algorithm are brought in

Step 4: the initial parameters are brought into the SOA-SaDE fusion algorithm to calculate the iterative results of 300 times, 500 times, and 1000 times, respectively

Step 5: compare the results of the traditional method and the improved algorithm

5.2. Analysis of Experimental Results. Figures 3(a)–3(d) show the results of 500 and 1000 iterations of the general SOA algorithm and the SOA-SaDE algorithm. As can be seen from the graph that the SOA algorithm converges fast, but it is easy to fall into the local optimum. However, the SOA-SaDE algorithm is effective in dealing with the local optimum and can jump out of the local optimum and continue to converge toward the global optimum. When the image curve is flat, it is close to the optimal target value.

Table 1 is the calibration result of Zhang Dingyou's method; Tables 2–4 are the optimization results of the SOA algorithm and SOA-SaDE fusion algorithm after 500 and 1000 iterations, respectively. According to the reprojection error after optimization is calculated, the results of Zhang's method can be well optimized by these two optimization algorithms. Moreover, the SOA-SaDE algorithm proposed in this paper is generally superior to the SOA algorithm. The fusion algorithm proposed in this paper is robust and reusable and can improve the local convergence of the SOA algorithm and get better results, as shown in Figure 4.

6. Conclusion

In order to solve the local convergence problem, SaDE algorithm is not easy to fall into local convergence. When the optimal individual fitness of each iteration is less than the value of the last iteration, local convergence may have occurred. SaDE was used to optimize population parameters and increase population diversity. This paper presents a new optimization method for camera internal parameters. The algorithm is based on seagull algorithm and adaptive parametric differential evolution algorithm. In this paper, the two are integrated into a framework according to certain mechanisms. By comparing the reprojection errors of Zhang Zhengyou's calibration algorithm, SOA algorithm, and SOA-SADE fusion algorithm, it can be seen that the gull differential evolution algorithm can get smaller errors. The calibration accuracy is improved to a certain extent. The experimental results show that the gull difference algorithm has good accuracy and feasibility for camera internal parameters optimization. The algorithm can be combined with practical engineering cases to solve multidimensional nonlinear optimization problems accurately and effectively. There are many similar bionic algorithms, such as monarch butterfly optimization (MBO), earthworm optimization algorithm (EWA), elephant herding optimization (EHO), moth search (MS) algorithm, slime mould algorithm (SMA), and Harris hawks optimization (HHO). [12, 13] These bionic algorithms have their own unique characteristics and can play a very good role in specific engineering fields. We believe that, in the follow-up research, we can comprehensively consider the advantages and disadvantages of each algorithm, merge different algorithms, learn from each other, apply it to the vision of underwater robots for deep-sea exploration, and provide a solution for improving the accuracy of underwater robot vision detection. The underwater environment is complicated. How to solve the problem of underwater imaging should consider the refraction of light brought by water, and the problem of

floating objects in the water affecting image clarity is the next issue we need to consider.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] X. Xu, Z. Fang, J. Zhang et al., "Edge content caching with deep spatiotemporal residual network for IoV in smart city," *ACM Transactions on Sensor Networks*, vol. 17, no. 3, pp. 1–33, 2021.

[2] R. Qin, Y. Yang, and F. Li, "Calibration of monocular camera based on full-parameter adaptive mutation particle swarm optimization algorithm," *Journal of Southeast University (Natural Science Edition)*, vol. 47, pp. 193–198, 2017.

[3] C. Xu, Y. Liu, and Yi Xiao, "etc. Optimization methods of camera internal parameters based on improved particle swarm algorithm," *Progress in Laser and Optoelectronics*, vol. 57, no. 6, Article ID 061501, 2020.

[4] X. Xu, Z. Fang, J. Zhang et al., 2021.

[5] L. Yang, H. Zhang, and C. Wang, "Hybrid particle swarm optimization method for accurate camera calibration," *Progress in Laser and Optoelectronics*, vol. 56, no. 21, pp. 171–179, 2019.

[6] J. Liu, *Research and Implementation of Multi-Camera Calibration Algorithm Based on One-Dimensional Calibration Rod*, Hefei University of Technology, Hefei, China, 2020.

[7] F. Zhao, F. Xue, Y. Zhang, W. Ma, C. Zhang, and H. Song, "A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution," *Expert Systems with Applications*, vol. 113, pp. 515–530, 2018.

[8] M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Systems*, vol. 169, pp. 39–52, 2019.

[9] G. Dhiman and V. Kumar, "Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems," *Knowledge-Based Systems*, vol. 165, pp. 169–196, 2019.

[10] L. Chen, *Improved Adaptive Differential Evolution Algorithm and its Application Research*, Donghua University, Shanghai, China, 2012.

[11] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 646–657, 2006.

[12] W. Li, G.-G. Wang, and A. H. Gandomi, "A survey of learning-based intelligent optimization algorithms," *Archives of Computational Methods in Engineering*, vol. 28, no. 5, pp. 3781–3799, 2021.

[13] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and D. Gong, "A comprehensive review of krill herd algorithm: variants, hybrids and applications," *Artificial Intelligence Review*, vol. 51, no. 1, pp. 119–148, 2019.