# Research on Convex Polyhedron Collision Detection Algorithm Based on Improved Particle Swarm Optimization

Wei Wang
School of Air and Missile Defense
Air Force Engineering University
Xi'an, China，710051
lhww11@tom.com

Shuiqing Gong
School of Air and Missile Defense
Air Force Engineering University
Xi'an, China，710051
415194375@qq.com

PeiLin Li
School of Air and Missile Defense
Air Force Engineering University
Xi'an, China，710051
LPL863@163.com

Mingwei Chui
School of Air and Missile Defense
Air Force Engineering University
Xi'an, China，710051
cmw1985@foxmail.com

***Abstract*—A convex polyhedron collision detection algorithm based on the shortest distance is proposed, which uses convex Bounding Volume Hierarchies to express convex polyhedron. Thus the distance problem of two convex polyhedrons is come down to a non-linear programming problem with constraints. The non-linear programming problem can be solved by improved particle swarm optimization. The strategy of self-adaptive parameters adjusting is applied, which have enhanced global searching ability of particle swarm optimization and optimized the time complexity. Results show that the efficiency of improved particle swarm optimization is higher and the computing speed is faster.**

***Keywords-Collision Detection, Convex Polyhedron, PSO, Distance***

## I. INTRODUCTION

Collision detection is a key technology of virtual reality, quick and accurate collision detection plays a vital role in improving the reality and real time and enhancing the immersion of virtual reality, and the complexity of virtual environment gives a higher request to the collision detection [1]. The solve of the shortest distance between different polyhedrons is one of the key techniques of collision detection[2], among which, the tracking method is used most commonly, it computes the shortest distance between two objects to judge whether the objects hit. The Lin-Candy[3] algorithm and the GJK[4] algorithm are the most famous algorithms in this kind of methods .The main idea of the Lin-Candy algorithm is to compute the nearest points according to the nearest feature between two polyhedrons, when the polyhedrons are moving , it will track and renew the nearest feature pairs. The GJK algorithm is a decent algorithm, it calculates Minkowski D-value instead of carrying real operation on the objects, and simplifies the calculation of distance between the two sets of convex hull to one[5] through the conversion of related problems.

The method put forward in this paper is to convert the problem of collision detection to a non-linear programming problem of calculating the nearest distance between two objects, and use improved particle swarm optimization to solve it. There are many methods to solve the non-linear programming problem[6], but each method has its adaptability and limitation. So far, there is not a very efficient method to solve the non-linear programming problems. On the contrary, the improved particle swarm optimization, which is put forward in this paper, has fast convergent speed, simple calculation and easy realization, and is suitable to optimize the complex non-linear functions. It strengthens the ability to search global solution through the strategy of self-adaption parameters.

## II. DISTANCE MODEL OF CONVEX POLYHEDRON

### A. Theoretical Basis

**Definition 1:** If $x_i \in R^n$ , $0 \le \lambda_i \le 1$ , $i = 1,...,k$ , and $\sum_{i=1}^{k} \lambda_i = 1$ , then call $x = \sum_{i=1}^{k} \lambda_i x_i$ the convex combination of $x_1,...,x_k$ . $R^n$ is a n-dimensional space.

**Definition 2:** Set $V \in R^n$ , then the convex combinations of any finite set of points in V is called the convex hull, booked as $C(s)$ . Namely

$$C(s) = \left\{ \sum_{i=1}^{k} \lambda_i x_i \mid x_i \in R^n, 0 \le \lambda_i \le 1, i = 1,...,k, \sum_{i=1}^{k} \lambda_i = 1 \right\} \quad (1)$$

The research work for this paper is sponsored by the National Natural Science Foundation of China (N0:51075395).

**Definition 3:** The $C(s)$, which is the convex hull of finite set of points $V=\{x_1,...,x_k\}$, is called the convex polyhedron generated by $x_1,...,x_k$.

### B. Distance Model

Set A and B as are convex object in three dimension space, $k_A$ and $k_B$ are the vertex numbers of *A* and *B*. $x_i, y_i \in R^3$ are the vertex coordinates of *A* and *B*. Based on the above theory, the point set of *A* can be expressed as

$$A=\left\{x \mid x=\sum_{i=1}^{k_A}\lambda_i x_i, x_i \in R^3, \sum_{i=1}^{k_A}\lambda_i=1, 0\le\lambda_i\le 1\right\}$$

, the point set of A can be expressed as

$$B=\left\{y \mid y=\sum_{i=1}^{k_B}\delta_i y_i, y_i \in R^3, \sum_{i=1}^{k_B}\delta_i=1, 0\le\delta_i\le 1\right\} .$$

Therefore, the distance model between A and B can be expressed as the following non-linear programming problem.

$$\begin{aligned} \min \quad & O_{A,B}=\| \sum_{i=1}^{k_B}\delta_i y_i-\sum_{i=1}^{k_A}\lambda_i x_i \| \\ s.t. \quad & \sum_{i=1}^{k_A}\lambda_i=1, \quad \sum_{i=1}^{k_B}\delta_i=1 \\ & 0\le\lambda_i\le 1, \quad 0\le\delta_i\le 1 \end{aligned} \tag{2}$$

Where, $O_{A,B}$ denotes the Euclidean distance between *A* and *B*.

In this way, the problem of collision detection between *A* and *B* is converted to the problem of solving the above non-linear programming problem. We can obtain the nearest points under restriction conditions. If $O_{A,B}=0$, the two objects collide; if $O_{A,B}>0$, the two objects separate. Due to the vertex information of *A* and *B* is known, the key is to optimizing collision time, this paper will adopt improved particle swarm optimization to seek the answer.

## III. IMPROVED PARTICLE SWAM OPTIMIZATION

### A. Particle Swam Optimization Fundamentals

Particle Swam Optimization(PSO) is a method used to solve optimization problems, which is enlightened from biotic population. In PSO, every potential solution of optimization problem which can be thought of a point in d dimension searching space is called a particle. Every particle flies in the searching space with a speed which can be dynamically adjusted by the flying experience of its accompaniers and itself. Every particle has its own fit value determined by object function, and knows its current place and the best place (shortened by $p_{best}$) by now, which can be regarded as the particle's own flying experience. In addition to, every particle knows the global best place (shortened by $g_{best}$) found by all particles, which can be considered as the partners' flying experience. The process of optimization problem can be regarded as a process in which particles are continuous renewal. Depending on *Eq.3* and *Eq.4*, every particle adjusts its flying speed and direction based on its current speed, $p_{best}$ and $g_{best}$. After iterating n times, the $g_{best}$ is the answer of problem.

$$\begin{aligned} v_{id}(t+1)=w\times v_{id}(t)+c_1\times r_1\times(p_{best}b_{id}(t)-x_{id}(t)) \\ +c_2\times r_2\times(g_{best}b(t)-x_{id}(t)) \end{aligned} \tag{3}$$

$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1) \tag{4}$$

Where, $x_i(t)$ is the location of particle $i$ on $t$ time, $v_i(t)$ is the speed of particle $i$ on $t$ time, $x_i(t+1)$ is the location of particle $i$ on $t+1$ time, $v_i(t+1)$ is the speed of particle $i$ on $t+1$ time, $r_1$、$r_2$ is random numbers between 0 and 1; $p_i(t)$ is the best place of particle $i$ till $t$ time, $p_g(t)$ is the global best place of all particles till $t$ time, constant $c_1$、$c_2$ are learning factors, which express how deep the particle is affected by individual and social canonization, and to adjust the flying step of particles along the $p_i(t)$ and $p_g(t)$ directions; $w$ is the inertia weight, which makes the particles free fall on their own speeds.

### B. Design of the Fitness Function

The fitness represents the adjustability of particle x in different environments. The purpose of intelligent optimization algorithms is to seek the global best solution. For the non-linear problem in this paper, we separate the fitness function into two kinds: One is objective fitness for objective function, the other is constraint fitness for constraint function.

Hence, we give the following fitness functions:

$$F_{obj}(\lambda_1,...,\lambda_{k_A},\delta_1,...,\delta_{k_B})=O_{A,B}. \tag{5}$$

$$F_{res}(\lambda_1,...,\lambda_{k_A},\delta_1,...,\delta_{k_B})=\left(\sum_{i=1}^{k_A}\lambda_i-1\right)^2+\left(\sum_{i=1}^{k_B}\delta_i-1\right)^2. \tag{6}$$

Eq. 5 is the objective function, Eq. 6 is the constraint function. When particle x obtain feasible solution, the fitness function is Eq. 5, when it obtain infeasible solution, the fitness function is Eq. 6.

### C. Self-adaptive Parameters Adjusted Strategy

In order to enhance the ability to skip local optimization, after researching the function of inertia weight, we find that if inertia weigh $w$ is bigger, arithmetic ability of whole search is more powerful, if $w$ is smaller, arithmetic ability of local search is more powerful. Consequently, some researchers adopt self-adaptive inertia weigh strategy to balance search and exploitation. For example, Eberhart put forward the linearity descending strategy of inertia weigh, the inertia weigh on t time can be expressed[7].

$$w(t)=(w_{start}-w_{end})(T_{max}-t)/T_{max}+w_{end}. \tag{7}$$

The improved particle swarm arithmetic put forward in this paper is not based on iteration times to adjust inertia weigh parameter, but adopt different parameters in each iteration to dynamically balance arithmetic searching ability.

The idea of self-adaptive parameters adjusted Strategy is: rank the particles from excellent to inferior according to their $p_{best}$, the inertia weigh and accelerating coefficient of particle ranked i is expressed as follow:

$$w_i = w_{\min} + (w_{\max} - w_{\min})(N_p - i)/(N_p - 1). \quad (8)$$

$$c_{i1} = c_{i2} = (w_i + 1 + 2\sqrt{w_i})/2. \quad (9)$$

Where, $w_{\max}$, $w_{\min}$ is defined as the maximum and minimum inertia weigh value, $N_p$ is the size of swarm, accelerating coefficient $c_{i1}$, $c_{i2}$ is self-adjusted by inertia weigh $w_i$. Through distributing different tasks to different particles, the strategy can keep a better balance on exploitation and searching ability in every iteration and is easy to be achieved.

Therefore, speed updating expression of particle $i$ in improved particle swarm optimization can be expressed:

$$\begin{aligned} v_i(t+1) = {} & w_i \times v_i(t) + c_{i1} \times r_1 \times (p_i(t) - x_i(t)) \\ & + c_{i2} \times r_2 \times (p_g(t) - x_i(t)) \end{aligned}. \quad (10)$$

Particle position expression can be expressed:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (11)$$

*D. Improve Particle Swarm Optimization Process*

**Step 1:** Setting initialization parameter, including: the particle swarm $N_p$, the maximum iteration time $G_m$, the minimum inertia weigh $w_{\min}$, the maximum inertia weigh $w_{\max}$, and randomly initialize the position of particles and their speed.

**Step 2:** Updating particles position. The positions of particles in swarm are updated based on Eq.10 and Eq.11. In the process of iteration, $p_g(t)$ is determined by the global best point under restrictions. Particle updating rules are:

Condition 1: If $x_i(t)$, $x_i(t+1)$ are both feasible, select the particle which make Eq.5 smaller as the new position of $x_i(t+1)$.

Condition 2: If only one between $x_i(t)$ and $x_i(t+1)$ is feasible, select the feasible position as the new position of $x_i(t+1)$.

Condition 3: If $x_i(t)$, $x_i(t+1)$ are both infeasible, select the particle which make Eq.6 smaller as the new position of $x_i(t+1)$.

**Step3:** when iteration times exceed the maximum time, or content the precision, arithmetic is finished, and export $p_g$, otherwise, turn to the second step.

## IV. CALCULATION EXAMPLE

As for improved particle swarm optimization, set $N_P = 40$, $G_m = 200$, $w_{\max} = 0.9$, $w_{\min} = 0.4$, and select two convex tetrahedrons (A and B) in 3d space, then calculate two kinds of cases, that is collision takes place and not.

*A. Two Convex Tetrahedrons Collide*

When the two convex tetrahedrons A and B collide, assume the vertex coordinates of A is:

$x_1(0, 1, 0)$, $x_2(1, 0, 0)$, $x_3(0, 0, 1)$, $x_4(0, 0, 0)$

and the vertex coordinates of B is:

$y_1(1, 0, 0)$, $y_2(2, -2, 0)$, $y_3(0, -2, 2)$, $y_4(0, -2, 0)$

So the procedure to calculate the distance between A and B is :

$$\begin{aligned} \min \quad & O_{A,B} = \left\| \sum_{j=1}^{4} \delta_j y_j - \sum_{i=1}^{4} \lambda_i x_i \right\| \\ s.t. \quad & \sum_{i=1}^{4} \lambda_i = 1, \quad \sum_{j=1}^{4} \delta_j = 1 \\ & 0 \le \lambda_i \le 1, \quad 0 \le \delta_j \le 1, \quad i, j = 1, 2, 3, 4 \end{aligned} \quad (12)$$

The optimization problem of nonlinear programming is calculated respectively with penalty function algorithm and improved particle swarm algorithm, and its results are shown in table 1.

*B. Two Convex Tetrahedrons Separate*

When the two convex tetrahedrons A and B separate, assume the vertex coordinates of A is:

$x_1(0, 1, 0)$, $x_2(1, 0, 0)$, $x_3(0, 0, 1)$, $x_4(0, 0, 0)$

and the vertex coordinates of B is:

$y_1(1, -1, 0)$, $y_2(2, -2, 0)$, $y_3(0, -2, 2)$, $y_4(0, -2, 0)$

Calculate in the same way, and the results are shown in table 2.

TABLE 1. THE TWO ALGORITHMS' COMPARISON UNDER COLLISION CASE

| | $\lambda_1$ | $\lambda_2$ | $\lambda_3$ | $\lambda_4$ | $\delta_1$ | $\delta_2$ | $\delta_3$ | $\delta_4$ | $O_{A,B}$ | Time /s |
|---|---|---|---|---|---|---|---|---|---|---|
| penalty function algorithm | 0.0004 | 1.0281 | -0.0005 | 0.0401 | 0.9882 | -0.0003 | 0.0022 | 0.0005 | 2.54e-033 | 5.532 |
| improved particle swarm algorithm | 0.0001 | 0.9999 | -0.0001 | 0.0001 | 0.9999 | -0.0001 | 0.0001 | 0.0002 | 1.33e-035 | 0.256 |

TABLE 2. THE TWO ALGORITHMS' COMPARISON UNDER SEPARATION CASE

| | $\lambda_1$ | $\lambda_2$ | $\lambda_3$ | $\lambda_4$ | $\delta_1$ | $\delta_2$ | $\delta_3$ | $\delta_4$ | $O_{A,B}$ | Time /s |
|---|---|---|---|---|---|---|---|---|---|---|
| Penalty Function Algorithm | 0.0002 | 1.042 | -0.0004 | 0.0205 | 0.9890 | -0.0007 | 0.0032 | 0.0006 | 0.9855 | 4.832 |
| Improved Particle Swarm Algorithm | 0.0001 | 0.9999 | -0.0001 | 0.0001 | 0.9999 | -0.0001 | 0.0001 | 0.0001 | 1.0000 | 0.356 |

The results show that the improved particle swarm algorithm is faster and more accurate than the penalty function algorithm.

## V. CONCLUSIONS

This paper has mainly discussed a fast approach for collision detection between polyhedrons, and converted the nearest distance calculation to a non-linear programming problem with obligation conditions, and solved the problem by Particle Swam Optimization after the obligation conditions were managed. The conclusions were showed as follows through experiment and simulation analysis, Particle Swam Optimization has a higher speed and precision, which proved that Particle Swam Optimization is feasible and efficient for this series problems. This paper has just discussed with two objects, if we dissolved the more complicated object modality or more objects with Particle Swam Optimization, we would get favorable advantages.

## REFERENCES


[1] W. W. Gu, Y. M. Chen, "Application of Collision Detection in Virtual Simulation System", Computer Engineering, 2005, pp.186-188.

[2] Cameron S, "A comparison of two fast algorithms for computing the distance between convex polyhedron", IEEE Transactions on Robotics and Automation, 1997, pp.915-920.

[3] Lin M C, Manocha D, "Fast interference detection between geometric models", The Visual Computer, 1995 , pp.542-561.

[4] Gilbert E G, Johnson D W, Keerthi S, "A fast procedure for computing the distance between complex objects in three-dimensional space", IEEE Trans on Robotics and Automation, 1988 , pp.193-203.

[5] Christer Ericson, Real-Time Collision Detection, Beijing, 2010.

[6] Bai Jiangbi, Jin Weigang, Zhang Jianhua, "A Fuzzy Neuron Network Based on Particle Swam", Northeast Electricity Technology, 2007 , pp.16-19.

[7] H. Song, J. Z. Zuo, Z. J. Wang, "Intelligent Control System of Guided Bomb Based on ANFIS Evolved", Systems Engineering and Electronics, 2005 , pp.1795-1799.