Soft Computing (2021) 25:5073–5092
https://doi.org/10.1007/s00500-020-05510-6

METHODOLOGIES AND APPLICATION

A new evolving mechanism of genetic algorithm for multi-constraint intelligent camera path planning

Zeqiu Chen[1] · Jianghui Zhou[1] · Ruizhi Sun[1,2] · Li Kang[1]

Published online: 5 January 2021

Abstract
The main goal of intelligent camera path planning is to determine an optimal pathway that proceeds from the starting position to the target position under several constraint conditions in the given environment. Genetic algorithm-based method has found wide application in path optimization problem in the intelligent camera community recently. Because the roaming environments are very complex, the planning path of the intelligent camera should meet other constraint conditions in addition to the path length constraint and the obstacle-free constraint. In this study, a new fitness function was developed in the genetic algorithm, which can consider the constraint conditions in terms of free obstacle, path length, path smoothness, and the visibility of the objective of interest in advance during the camera roaming. In addition, a new evolving operator was introduced into the genetic algorithm, so that the number of iteration can be significantly reduced, and thus, the efficiency of the genetic algorithm can be improved. Experimental results show that the proposed genetic algorithm can obtain a high-quality path under multi-constraint conditions for intelligent camera with less numbers of iteration as compared with several conventional methods.

Keywords Path planning · Intelligent camera · Genetic algorithm · Multiple constraints · Fitness function · Evolving operator

1 Introduction

With the rapid development of artificial intelligence, scientific visualization has attracted increasing attentions in many fields, which include chemistry, medicine, astronomy, and agriculture, etc. Over the past decades, many researches have been conducted to improve scientific visualization (Bryson 1996; Liang and Li 2008; Pierre and Zakaria 2011; Wang and Tao 2017). Intelligent camera control, first introduced by Drucker and Zeltzer (1994), is the core issue of scientific visualization, which aims to plan an optimal pathway from the starting position to the target one in order to meet several constraint conditions. Although there exist a number of researches on the path planning problem in the literature, path planning is still a main subject of scientific visualization and deserves further investigation in the future.

It has been reported that path planning problem is a NP-hard optimization one (Srinivas and Patnaik 1994; Kuang et al. 2006; Manikas et al. 2007), which can be only solved by heuristic algorithms (Chaudhari et al. 2017; Liu et al. 2017; Contreras-Cruz et al. 2015; Sahoo et al. 2018; Tharwat et al. 2019). In recent years, the membrane algorithm area is focusing on developing new variants of heuristic algorithms for solving complex optimization problems, including the motion planning problem in robotics, by using either the hierarchical or network membrane structures, evolution rules and computational capabilities of membrane systems (Wang et al. 2015; Zhang et al. 2017; Perez-Hurtado et al. 2018, 2020). Among these heuristic algorithms, the genetic algorithm has been proved to be an effective and popular method to deal with the path planning problem. Nowadays, a variety of algorithm-based methods have been proposed to solve

Communicated by V. Loia.

✉ Li Kang
kangli@cau.edu.cn

1 College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China

2 Scientific Research Base for Integrated Technologies of Precision Agriculture (Animal Husbandry), the Ministry of Agriculture, Beijing 100083, China

the path planning problem (Hu and Yang 2004; Pol and Murugan 2015; Song et al. 2016; Elhoseny et al. 2018; Patle et al. 2018; Nazarahari et al. 2019). For example, Ali et al. (2005) employed the genetic algorithm to determine the collision-free path of manipulators and found that the genetic algorithm works better than the conventional A* method in terms of path length and computation time. Karami and Hasanzadeh (2015) designed a novel selection operator for a new adaptive genetic algorithm in order to maintain the diversity of individuals and escape from the local optima. And they confirmed that their algorithm outperforms the related methods in terms of solution quality and finding an optimum path based on the experimental results. Lee and Kim (2016) proposed an effective initialization method for genetic algorithm with variable-length chromosomes for robot to find a feasible path without intersecting any obstacles in the given environment. The results reveal that the proposed initialization method can effectively improve the performance of the genetic algorithm. Despite a lot of interesting achievements on the genetic algorithms for path planning, researches on the improvement of path planning by the genetic algorithm-based methods have never cease. One of the reasons lies in the fact that many existing genetic algorithms require huge execution time in order to find a high-quality path. Therefore, in order to promote the application of genetic algorithms on the path planning problem, it is of necessity to conduct studies on the genetic algorithms in terms of their efficiency.

The optimal solution of the path planning problem determined by genetic algorithms refers to the one that can satisfy certain constraint conditions, e.g., the planning path is the shortest among all the feasible cases or it won't pass through any obstacles in the working field, etc. Up to now, lots of genetic algorithms have been put forward for path planning problem, and most of them focused on the optimization of path planning under a single constraint condition. Researches on the path planning problem under multi-constraint condition are relatively limited (Ahuactzin et al. 1991; Xiao et al. 1997; Wang et al. 2004; Zhou et al. 2008; Mou et al. 2008). In fact, the roaming environments are very complex, so the planning path of the intelligent camera should meet other constraint conditions in addition to the path length constraint and the obstacle-free constraint. For example, a sharp turn must be avoided during the roaming process of the intelligent camera, and the planning path should be as smooth as possible in order to minimize the dizziness effect. Besides, the objective of interest should enter the vision field as soon as possible, so that the user can experience a pleasant interaction with the objective during the roaming process. Although several studies have been conducted to investigate the multi-constraint optimization of path panning (Brindle 1980; Vadakkepat et al. 2000; Ahmed and Deb 2013; Ramirez-Atencia et al. 2017; Nazarahari et al. 2019), this issue is far from being fully solved since the constraint conditions are complex and will vary with the working field (e.g., UAVs, vehicles, and robots). Therefore, in order to obtain the optimal path under the constraint conditions in real-life situation, it is important to study the path planning of intelligent camera under the multi-constraint condition.

The purpose of this study is to propose a new path planning method by introducing an evolving operator, with the aim to reduce the number of iteration and improve the efficiency of the genetic algorithm. In addition, a new fitness function is developed for the proposed method, so that the optimized path can satisfy the multiple constraint conditions, i.e., (1) the path is obstacle-free, (2) the objectives can enter the vision field of camera as soon as possible, (3) the path is relatively short, and (4) the path is smooth enough to reduce the dizziness effect during the roaming process. The rest of this paper is organized as follows: Section 2 presents a brief introduction about how to conduct path planning by genetic algorithm. The new fitness function to characterize the multiple constraint conditions and the new evolving operator to improve the efficiency of the genetic algorithm during the path planning are described in Sect. 3. In Sect. 4, we will introduce the experimental settings of the numerical tests and compare the simulation results by our algorithm with those by some of the state-of-the-art methods. Finally, the main findings are summarized in Sect. 5.

2 Path planning with genetic algorithm

As a natural-inspired algorithm, genetic algorithm was developed based on the concept of Darwinian evolution. Genetic algorithm is a kind of random search and optimization algorithm similar to the evolution of biological population, which involves an initialization method, fitness function, natural selection, crossover, and mutation operators. In genetic algorithm, an initial population is generated randomly. In order to obtain the optimal solution of the problem, the quality of each chromosome in the population is evaluated by a fitness function. Then, the parents will be selected to be subjected to the reproduction according to their fitness values. Crossover is applied to produce new offspring based on the selected parents in the previous step. The genetic structures of some chromosomes will be changed by the mutation operator in order to guarantee the diversity of the population. The genetic algorithm will be stopped until the optimal solution is determined, i.e., the stopping criteria are satisfied (Sugisaka and Fan 2001). The pseudo-code of genetic algorithm is shown in Table 1.

Table 1 Pseudo-code of genetic algorithm for path planning

Algorithm 1. Genetic algorithm for path planning

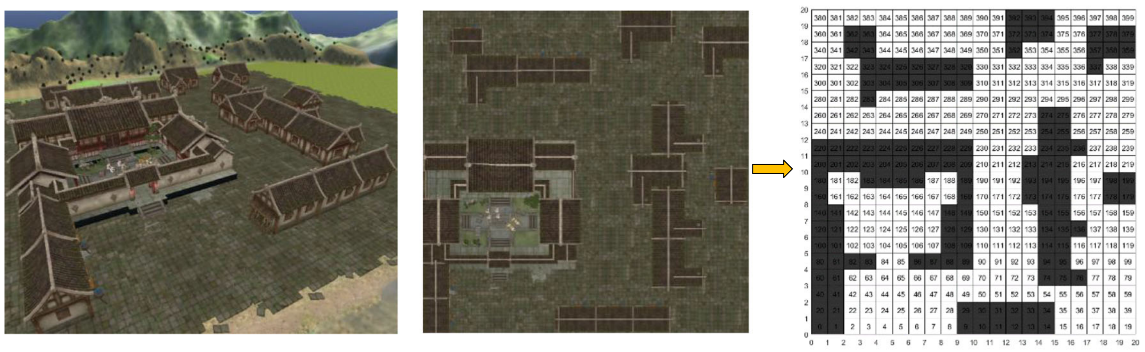
```
1:  INPUT:  map, starting point, target point, population, maximum number of iteration
2:  OUTPUT:  optimal path
3:  SEE ← See(map, target point)   // Traverse the set of visual points
4:  X ← Initial(map, starting point, target point, population)   // Initial population
5:  fx ← fitness(X)   // Calculate initial fitness
6:  gen ← 1
7:  while gen < maximum number of iteration do
8:        fit ← 1 ./ (fx+1)   // Calculate the average fitness.
9:        Xsel ← select(X, fit)   // Select operation.
10:       Xsel ← crossover(Xsel, fit)   // Crossover operation(Xsel).
10:       Xsel ← mutation(Xsel, fit, map)   // Mutation operation.
11:       X ← evolving(Xsel, SEE, map)   // Evolving operation.
12:       fx ← fitness(X)   // Calculate the new fitness
13:       gen ← gen + 1
14: end while
15: return optimal path
```

2.1 Representation of environment

During the process of path planning, the intelligent camera will interact with the virtual environment to determine the optimal pathway. Therefore, it is necessary to represent the environment by a model that can be identified by the intelligent camera. Recently, several models including grid-based model, polygon model, and the cell tree have been used to represent the environment (Lamini et al. 2018). In this study, the grid-based model, one of the most widely used models, was used to characterize the environment due to its good performance and easy implementation. Figure 1 shows a typical environment during the roaming of the intelligent camera and the corresponding grid-based model. As shown in Fig. 1c, the working space of the intelligent camera is represented by an even numbered grid. All the positions in the grid can be divided into two categories: One is the vacuum space that enables the intelligent camera to move freely, while the other is obstacle zone (black zone shown in Fig. 1c) where the intelligent camera cannot go through. Obviously, a feasible solution of the path planning problem is the pathway from the starting position to the target position that crosses a set of obstacle-free positions.

2.2 Encoding of paths (chromosomes)

In the genetic algorithm, each path corresponds to one chromosome. The choice of encoding of chromosomes is a very important step for the successful implementation of the genetic algorithm. It has been reported that different methods can be used to encode the pathway, which depends on the path planning problem to be solved (Lamini et al. 2018). The binary encoding method is one of the most



(a) top view of the roaming environment **(b)** plan view **(c)** grid-based model

Fig. 1 Typical roaming environment of intelligent camera and the corresponding grid-based model

widely used methods. However, we employed an orderly numbered method to encode the paths since it is simpler in form and easier to be operated by genetic operator. In the orderly numbered method, each position in the grid-based model can be represented by a unique number. For example, $P = \{p_1, p_2, p_3, \ldots, p_n\}$ denotes a path from the starting position (p_1) to the target position (p_n) in the grid-based model. p_i is the ith point on the pathway. More detailed descriptions about the orderly numbered method can refer to the reference (Adem and Mehmet 2012). It is noted that in this study, the length of paths is variable. The optimal path should be as short as possible in addition to satisfy other constraints of the path planning problem.

2.3 Initialization of population

The efficiency of the genetic algorithm is greatly influenced by the initial population. An efficient initial population method can improve the search effectiveness of the genetic algorithm. Recently, several methods such as random, key cells, potential field, and greedy approach method have been employed to obtain the initial paths (Lamini et al. 2018). In order to meet the diversity and randomness requirements of the initial population, random number method (Li et al. 2017) was used to generate the initial population in this study. Since the intelligent camera will pass through the grids between the starting position and the target position, one grid will be chosen randomly from each row of the grid-based model. By setting (x_0, y_0) as the starting point and (x_n, y_n) as the ending point, where $x_n \geq x_0$ and $y_n \geq y_0$, the position of current point is (x_i, y_i), then the position of the next point $((x_{i+1}, y_{i+1}))$ can be generated randomly by the following equation:

$$\begin{cases} x_{i+1} = x_i + (x_n - x_i) \\ y_{i+1} = y_0 + (y_n - y_0) \end{cases} \tag{1}$$

Then, these randomly selected grids will be connected into a continuous path based on the midpoint connection method (Wei and Long 2019). The random number method used for initialization of population in this study can ensure the continuity of path; however, the initialized path may go through the obstacles in the grid-based model, which will turn into an invalid path. In this study, the invalid path will be penalized in the fitness function; therefore, it will not be selected for the rest operations of the genetic algorithm. The specific process will be introduced in Sect. 3.2 in detail.

2.4 Fitness function

When the initial population is generated based on the random number method, the genetic algorithm needs to evaluate the performance of each individual in the initial population. Usually, the fitness function can be used to assess the quality of each individual, which should consider several constraints of path planning problem such as length, safety, and smoothness of path. The definition of fitness function is crucial to the genetic algorithm because it influences the convergence quality as well as the optimization solution of the genetic algorithm. In this study, a new fitness function that considers multi-constraints will be proposed, which will be introduced in Sect. 3.1 in detail.

2.5 Selection operator

The main objective of selection process of genetic algorithm is to choose the potentially better individuals to form a mating pool. Therefore, individuals with good performance can be obtained in the next generation. An appropriate selection operator can avoid the loss of useful information and improve the global convergence and calculation efficiency. Up to now, many selection operators including roulette wheel selection, rank selection, elitism selection, tournament selection, and deterministic sampling selection have been proposed for the path planning problem by genetic algorithm, which are summarized in Table 2.

In this study, the deterministic sampling selection method will be used to conduct the selection process of the genetic algorithm, which can ensure that some individuals with great fitness value can be retained in the next generation, while those with small fitness value can be removed from the next generation. The main procedures of the deterministic sampling selection are shown as follows:

1. The expectation of the survival number of individuals (N_i) in the next generation can be calculated according to Eq. 2.

   $$N_i = M_p \cdot f_i / \sum_{i=1}^{M_p} f_i \tag{2}$$

   where M_p is the number of individuals (chromosomes); f_i is the fitness value of the i th individual.
2. For the i th individual, the number of the survived individual in the next generation is $[N_i]$, where, $[N_i]$ represents the rounding down operator.
3. All the individuals are sorted in descending mode in terms of $N_i - [N_i]$; then, the first $M_p - \sum_{i=1}^{M_p} [N_i]$ individuals will be added based on the sorting results into the next generation.

2.6 Crossover operator

The main purpose of crossover operator is to combine the features of two parent chromosomes to form the offspring.

Table 2 Summary of typical selection operators for genetic algorithm in the literature

Approaches	Basic idea	Characteristic	References
Roulette wheel selection	The probability of the selected chromosome is proportional to the fitness value	The higher the fitness, the greater the probability to be selected; however, a great selection error will be introduced due to the random nature	Hung et al. (2007), Yao and Ma (2010), Cai et al. (2016), Song et al. (2016), Zhang and Ding (2016), Li et al. (2017)
Rank selection	It allocates the probability of each individual to be selected based on the order of individual fitness	It can reduce the negative effect caused by the significant difference between the fitness of individuals	Geisler and Manikas (2002), Tuncer and Yildirim (2012)
Elitism selection	It allows the best individual in the current generation to be passed into the next generation without any change	It can ensure that the best solutions must survive in the population; however, it is easy to fall into local optimum	Alajlan et al. (2013), Wang et al. (2017), Elhoseny et al. (2018)
Tournament selection	Every individual in the population is paired at random with another. The fitness values of each pair will be compared, and the fitter individual of the pair moves on to the next generation	It can reduce the probability of convergence to local optimum	Tsai et al. (2011), Hsu and Liu (2014), Yang et al. (2016)
Deterministic sampling selection	The selection is conducted based on the survival expectation of each individual in the next generation	It can keep diversity and avoid premature convergence	Yun and Xi (1996)

Therefore, the diversity of the population can be enriched, and the solution to the path planning problem won't be easy to fall into local optima. Currently, there exist many crossover schemes, e.g., single-point crossover, two-point crossover, uniform crossover, and arithmetic crossover, which are summarized in Table 3.

In this study, single-point crossover is employed due to its high computational efficiency. When the two paths have an intersection, the parts of the two paths will be swapped after the intersecting point. If there exists more than one intersecting point, only one point will be selected randomly to conduct the crossover operation. When there is no intersection for the two paths, the crossover operation won't be performed, which can avoid the discontinuity of the paths.

It is noted that in the original genetic algorithm, the crossover probability is fixed, which may lead to premature convergence and local optima. In order to solve these problems, a modified self-adaptive adjustment formula that uses a sigmoid function to adjust crossover probability was employed in this study, which is expressed as:

$$P_c = \begin{cases} \dfrac{p_{c\max} - p_{c\min}}{1 + \exp\left[(p_{c\max} - p_{c\min}) \left(\dfrac{f_c - f_{cave}}{f_{c\max} - f_{cave}} \right) \right]} + \dfrac{p_{c\max} + p_{c\min}}{2}, & f_c \geq f_{cave} \\ p_{c\max} p_{c\max}, & f_c < f_{cave} \end{cases} \tag{3}$$

where $P_{c\min}$ and $P_{c\max}$ are the lower and upper limit values of crossover probability, respectively; f_c is the greater fitness of the two individuals involved in the crossover operation, $f_{c\max}$ and f_{cave} denote the maximum and average fitness values in the crossover operation, respectively.

It is clear that from Eq. 3 that the crossover probability of the individuals with fitness value lower than f_{cave} will be set as $p_{c\max}$, which can promote the gene changes of such individuals. In addition, Eq. 3 possesses the characteristics of decreasing in $[f_{cave}, f_{c\max}]$, which can guarantee the continuity stability of the crossover probability. Moreover, when f_c approaches $f_{c\max}$, the crossover probability is $\frac{p_{c\max} - p_{c\min}}{1 + \exp(p_{c\max} - p_{c\min})} + p_{c\min}$, and the local convergence resulting from very small crossover probability can be avoided.

2.7 Mutation operator

Mutation process is also an important part of genetic algorithm, which refers to the process that candidate paths do an actual change at some points; therefore, the population diversity can be expanded and the global search for optimal path can be ensured, which can avoid the premature convergence. Recently, many mutation operators including simple mutation, uniform mutation, boundary mutation, non-uniform mutation, and Gaussian mutation have been used in genetic algorithm, which are summarized in Table 4.

In this study, the mutation operator is conducted like this: randomly select two points called mutation points from the candidate path (the starting and target points are not included), a segmented path is generated between the two mutation points, and finally a new path is created by

Table 3 Summary of typical crossover operators for genetic algorithm in the literature

Approaches	Description	Characteristic	References
Single-point crossover	It uses the single-point fragmentation of the parents and then combines the features of parents at the crossover point to create the offspring	It is easy to be implemented and has fast calculating speed	Shi and Cui (2010), Miao et al. (2011), Tuncer and Yildirim (2012), Song et al. (2016), Bakdi et al. (2017), Li et al. (2017), Ahmadi et al. (2018), Elhoseny et al. (2018), Patle et al. (2018)
Two-point crossover	It chooses two crossover points from two parents randomly and exchanges partial chromosomes of the two parents at the crossover points	It is less disruptive with respect to widely separated alleles	Yu et al. (2002), Qu et al. (2013)
Uniform crossover	It swaps bits in the parents to be included in the offspring by choosing a uniform random real number	It provides the uniformity in combining the bits of both parents	Mansouri et al. (2008), Wang and Hwang (2009), Trujillo et al. (2016)
Arithmetic crossover	It selects two chromosomes randomly for crossover and creates two offspring which are linear mixture of their parents	It is usually used for real-value encoding	Hung et al. (2007), Bai et al. (2011), Han et al. (2017)

combining the segmented path formed by the mutation points and the other segmented paths formed by the remaining points in the candidate path.

Similar to the crossover operator, a modified self-adaptive adjustment formula that uses a sigmoid function to adjust mutation probability is employed to avoid premature convergence and local optima, which is expressed as:

$$P_m = \begin{cases} \dfrac{p_{m\max} - p_{m\min}}{1 + \exp\left[(p_{m\max} - p_{m\min}) \left(\dfrac{f_m - f_{mave}}{f_{m\max} - f_{mave}} \right) \right]} + \dfrac{p_{m\max} + p_{m\min}}{2}, & f_m \geq f_{mave} \\ p_{m\max}, & f_m < f_{mave} \end{cases} \tag{4}$$

where $P_{m\min}$ and $P_{m\max}$ are the lower and upper limit values of mutation probability, respectively; f_m is the fitness value of mutation individuals, $f_{m\max}$ and f_{mave} denote the maximum and average fitness values of all the individuals, respectively.

3 Proposed methods for genetic algorithm

3.1 A new fitness function for path planning

The main goal of path planning problem is to find an optimal path between the starting point and the target point.

Table 4 Summary of typical mutation operators for genetic algorithm in the literature

Approaches	Description	Characteristic	References
Simple mutation	It generates randomly an integer within the range of the number of genes as a variation point	A small change on the chromosome results in a new gene fragment	Li et al. (2017), Yao and Ma (2010), Hsu and Liu (2014), Song et al. (2016)
Uniform mutation	It replaces the original gene of the individual with random numbers that are uniformly distributed in a certain range with a small probability	The diversity of the population can be increased by uniform mutation	Cheng et al. (2011), Zhang et al. (2013), Ramirez-Atencia et al. (2017)
Boundary mutation	It replaces the value of the original gene with the corresponding boundary value	It can be applied to a class of problems with the best point at or close to the boundary of the feasible solution	Xidias et al. (2012, 2016)
Non-uniform mutation	Each gene is randomly perturbed with the same probability, and the perturbed result is taken as the new gene	It can concentrate the search of optimal solution in the range of interest	Lin et al. (1994), Lee and Wu (2003)
Gaussian mutation	It refers to replace the original gene value with a random number which conforms to the normal distribution	It can focus on searching a local area of an individual due to the characteristics of normal distribution	Amiryan and Jamzad (2015), Menasri et al. (2015), Qi et al. (2019)

The optimal path refers to the path with best performance under the given constraint conditions. The performance of the candidate paths can be evaluated by the fitness function. It should be noted that the "optimal" here means that the planning must satisfy some criterions including the length of the planning path is the shortest, or the energy consumption of robot is the lowest, etc. However, most of the existing algorithms are developed to meet single constraint condition, e.g., the length of planning path should be the shortest. Actually, in many real-life situations, the path segment with a sharp turn small should be avoided when the intelligent camera is used in the virtual environment. If such path segment exists, it will bring dizziness effect to users when they are using the head-mounted display (HMD). Besides, the objective of interest should enter the vision field as soon as possible, so that the user can experience a pleasant interaction with the objective during the roaming process. Intelligent camera path planning can be regarded as a NP-hard problem with multiple optimal objects, which is difficult to find the precise solution. Consequently, a multi-constraint path planning algorithm for intelligent camera is proposed with the aim to the multiple constraint conditions. In order to meet the constraint conditions in terms of free obstacle, path length, path smoothness, and the visibility of the objective of interest in advance during the camera roaming, a new fitness function consisting of four components to consider the above constraint conditions is developed as Eq. 5. The greater the fitness function is, the better performance the path possesses, i.e., the path with a greater fitness value will have a higher probability of being selected to generate the optimal path during the genetic algorithm.

$$fit(p) = 1/\sum_{i=1}^{4} w_i f_i(p) \tag{5}$$

where $f_i(p)$ is the fitness component to characterize the i th constraint for path p; w_i is the weight of the i th fitness component.

3.1.1 Free obstacle constraint

In this study, the fundamental constraint for path planning problem is that the feasible path cannot pass through any obstacle during the camera roaming. And the fitness component is expressed as Eq. 6. Clearly, the fitness component of all feasible paths that do not pass through any obstacle will be zero.

$$f_1(p) = \begin{cases} 1, I_i \in \varepsilon(i = 1, \ldots, n) \\ 0, I_i \notin \varepsilon(i = 1, \ldots, n) \end{cases} \tag{6}$$

where I_i is the i th point in path p; n is the number of points in path p; ε denotes the point set of obstacles in the grid-based model.

3.1.2 Path length constraint

Finding the shortest path from the feasible solutions is one important goal of path planning problem. In this study, the fitness component to take the path length constraint into consideration is computed based on Euclidean distance, which can be written as Eq. 7. A smaller $f_2(p)$ means the candidate path exhibits a better performance.

$$f_2(p) = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{7}$$

3.1.3 Advance visibility constraint

During the roaming of intelligent camera, it is sometimes required that the objective of interest should enter the vision field of intelligent camera as soon as possible, so that the user can experience a pleasant interaction with the objective. Figure 2 illustrates the advance visibility constraint of two possible paths for virtual camera roaming. As shown in Fig. 2, the objective 1 of interest can be observed for the first time at Points A1 and B1 on the Paths A and B, respectively. By considering the path length of the two paths, objective 1 can be observed more early on Path A than on Path B. Therefore, Path A performs better than Path B in terms of advance visibility constraint.

In this study, the fitness component defined as Eq. 8 is employed to consider the advance visibility constraint. It can be inferred from Eq. 8 that a feasible path with a better performance will have a lower value of $f_3(p)$.

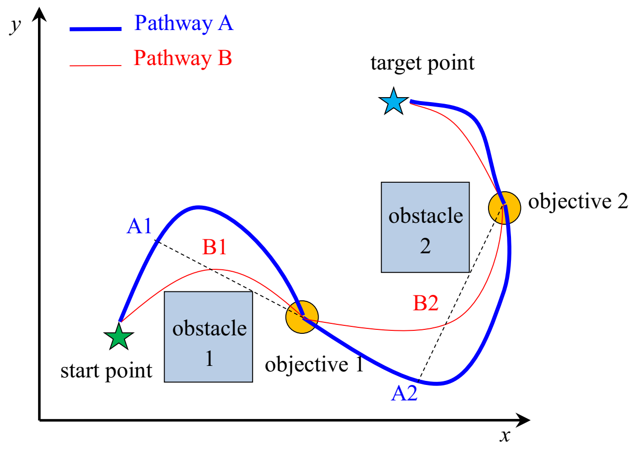


Fig. 2 Illustration of advance visibility of two possible paths for virtual camera roaming

$$f_3(p) = \frac{\sum_{i=1}^{m-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{|x_n - x_1|} \tag{8}$$

where m is the m th point on the possible path at which the objective of interest can be observed by the intelligent camera for the first time.

3.1.4 Path smoothness constraint

In order to minimize the dizziness effect caused by the sudden turn of intelligent camera during the roaming process, the optimal path determined by the genetic algorithm should be smooth as possible. In this study, the mean turning angle of intelligent camera is introduced to consider the path smoothness constraint. The mean turning angle denotes the change of angle per unit path length, which can be expressed as Eq. 9. It is evident that the path with better performance will possess a smaller mean turning angle.

$$f_4(p) = \frac{\sum_{i=1}^{n-2} |\theta_{i+1} - \theta_i|}{\sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \tag{9}$$

where $|\theta_{i+1} - \theta_i|$ is the angle between the path segments p_ip_{i+1} and $p_{i+1}p_{i+2}$, $\theta_i = \arctan\left(\frac{x_{i+1}-x_i}{y_{i+1}-y_i}\right)$ represents the angle between the path segment p_ip_{i+1} and the y coordinate.

3.2 A new evolving operator for path planning

It is widely accepted that one of the main drawbacks of original genetic algorithm is the premature problem, i.e., the result obtained by the genetic algorithm will prematurely converge to a local optima. During the process of finding the optimal solution by genetic algorithm, when certain individual performs much better than the other individuals, i.e., its fitness is much greater than those of the other individuals, this individual will be selected by the selection operator to generate the next generation with a high probability. As a result, most of the generated individuals from the next generation will possess similar features with the individual with a much greater fitness in the former generation. Therefore, the solution of the genetic algorithm will easily get trapped in a local optimum, and the selection operator and crossover operator usually fail to avoid such premature problem. Theoretically, the mutation operator can contribute to avoiding the premature problem. However, in order to ensure the stability of genetic algorithm, the mutation probability of the generated individual is usually very low (e.g., 0.01); therefore, it will require many numbers of iteration to generate a new individual that is quite different from those in the former generation, which will slow significantly the convergence speed of genetic algorithm. Furthermore, when the fitness of the new individual generated by the mutation operator is much smaller than the sum of the fitness of all the individuals, the new individual won't be probably be selected by the selection operator for the following operations. Thus, the premature problem cannot be effectively solved by the conventional mutation operator.

In this study, a new evolving operator was introduced to solve the premature problem in the path planning by genetic algorithm, which is similar to the conventional mutation operator. However, it is noted that the evolving operator will be performed for all the individuals to randomly and independently generate new individuals that may possess different features from those in the former generation. Moreover, in order to ensure that the individual with great fitness is not destroyed by the evolving operator, only the generated individual with a greater fitness by the evolving operator than that of the original one will be preserved for the following processes of genetic algorithm. Therefore, the diversity of the population can be increased significantly, which can extend the searching space of path planning problem and avoid falling into a local optima, and thus, the number of iterations of genetic algorithm can be effectively reduced.

The evolving operator is conducted after the crossover operation and the mutation operation. The pseudo-code of the evolving operator is shown in Table 5. The main procedures of the evolving operator are shown as follows: For each feasible path in the population, two points except the starting and target points are randomly selected. Then, the path segment between these two points is removed. A new path segment using these two points of the original path before being evolved is generated. Grids will be chosen randomly from each row of the grid-based model between the two points. Then, these randomly selected grids will be connected into a continuous path based on the midpoint connection method.

In the midpoint connection method, the following equation can used to judge whether the two adjacent points are continuous:

$$\Delta = \max\{|x_{i+1} - x_i|, |y_{i+1} - y_i|\} \tag{10}$$

where (x_i, y_i) and (x_{i+1}, y_{i+1}) are the coordinates of Points p_i and p_{i+1}, respectively.

It is obvious that $\Delta = 1$ denotes that Points p_i and p_{i+1} are two adjacent points. Otherwise, more points need to be inserted between Points p_i and p_{i+1}. The inserted point p'_i can be expressed as:

$$\begin{cases} x'_i = (x_{i+1} + x_i)/2 \\ y'_i = (y_{i+1} + y_i)/2 \\ n_i = x'_i + l \times y'_i \end{cases} \tag{11}$$

Table 5 Pseudo-code of evolving operator

Algorithm 2. Evolving operator for path planning

```
1:  INPUT:  $N_{itermax}$, $XSel$   // Maximum number of iteration, Selected individuals
2:  OUTPUT:  $XSel$   // Individuals after evolving
3:  $k$=0
4:  while $k < N_{itermax}$ do
5:      for $i \leftarrow 1$ to len($XSel$) do
6:          $path \leftarrow Xsel[i]$   // Calculate the fitness of the path
7:          $fx \leftarrow$ fitness($path$)   // Calculate the fitness of the path
8:          $p1, p2 \leftarrow$ random($path$ [1:-1])   // Randomly select two points
9:          $path$ [p1:p2] $\leftarrow$ []   // Remove the path between the two points
10:         $path \leftarrow$ generateContinuousRoute($path$)   // Generate a continuous path
11:         $fnew \leftarrow$ fitness($path$)   // Recalculate the fitness of the new path
12:         if $fnew < fx$ then
13:             $Xsel[i] \leftarrow path$   // Replace the old path with the new path
14:         end if
15:     end for
16:     $k \leftarrow k+1$
17: end while
```

where (x_i', y_i') is the coordinate of Points p_i'; n_i' is the grid number of the inserted point; l is the column number of the inserted point.

It should be noted that when the inserted point is located in the obstacle, the free grid nearest to the initial point located in the obstacle will be selected as the inserted point. If there exist duplicate grids in the generated path, the segmented path between the duplicate grids will be removed from the generated path, which is shown in Fig. 3.

The midpoint connection method used in this study can ensure the continuity of the randomly generated path; however, the generated path by connecting all the points may still go through the obstacles in the grid-based model, which will turn into an invalid path. In this study, the invalid path that cannot meet the multiple constraint conditions will be penalized in the fitness function; therefore, the invalid path that go through the obstacles will not be selected for the rest operations of the genetic algorithm.

Then, the fitness value of the new path obtained from the evolving operation will be evaluated according to the new fitness function proposed in Sect. 3.1, which will be compared with that of the original path. If the fitness value of the new path is greater than that of the original path, the original path will be replaced by the new path, which will be restored for the next operations, otherwise, the original path will be restored without any operation.

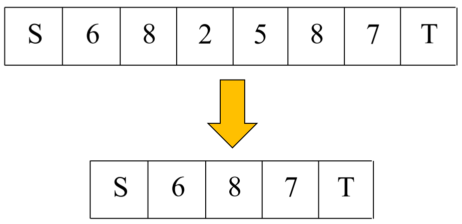


Fig. 3 Illustration of elimination duplicate grids in the initialized path

In this study, the evolving operator is controlled by the maximum number of iteration, i.e., when the specified maximum number of iteration is reached, the evolving process will be terminated and the optimal solution will be output. It is noted that the maximum number of iteration will vary with the size of grid-based model to represent the environment. Although it will increase the computing time of genetic algorithm to a certain extent, the evolving operation can effectively improve the searching ability of genetic algorithm to determine the optimal solution and avoid being trapped in a local optimum. Therefore, the number of iteration can be significantly reduced, and thus, the efficiency of the genetic algorithm can be improved.

4 Experimental results

4.1 Experimental settings

In this section, the performance of the genetic algorithm proposed in the study will be discussed based on the experimental results. Some of the state-of-the-art methods were employed in the experiments for comparison. The experiments were conducted in the two-dimensional grid-based model with different numbers of obstacles of different sizes and shapes. All the experimental simulations, coded in MATLAB language, were performed on a computer equipped with Intel Xeon E5 12 core processor (2.20 GHz clock rate) and 32 GB of random access memory. The parameters of the proposed method were summarized as follows: The population size was set as 200, the crossover and mutation probability were adaptive, and the penalty coefficients in fitness function were set as 0.38, 0.04, 0.56, 0.02 for w_i ($i = 1, \ldots, 4$), respectively. The comparison of the results obtained by our proposed genetic algorithm and those in the literature in terms of iteration number, visibility in advance, path length, and path smoothness will be presented in the following section. All the simulation experiments were conducted for 50 independent runs. Due to the fact that the nature of meta-heuristic algorithms is stochastic and different results may be obtained in different runs, the statistical analysis on the simulated results was performed to support the conclusions drawn in this study. First, the well-known nonparametric test, i.e., Kruskal–Wallis test, was used to analyze the data to validate the performances of all the algorithms used for comparison and to find out whether there are significant differences among these algorithms. If there exist significant differences among the experimental data of these algorithms, the multiple comparison procedure using the analysis of variance (ANOVA) test will be carried out to analyze the results of Kruskal–Wallis test, and the estimated value of the mean ranks for each group (algorithm) can be achieved, which can be used to evaluate the performance of the algorithms (Kruskal and Wallis 1952; Vargha and Delaney 1998; McKight and Najab 2010). A lower mean rank indicates that the algorithm has a better performance.

4.2 Comparison on iteration number

The iteration number affects the performance and the computational time of genetic algorithms. The iteration number of our proposed method was evaluated in the environment represented by the 2D grid-based model shown in Fig. 4. The genetic algorithms by Ahuactzin et al. (1991) and Alajlan et al. (2013) were also employed in the experimental simulations for comparison. The main features of these three genetic algorithms are summarized in Table 6. And the values for the parameters of the algorithms selected for comparison are summarized in Table 7.

Figure 5 displays the comparison of iteration number of different genetic algorithms to find the shortest path in the environment shown in Fig. 4. Three scenarios with different starting and target points shown in Table 8 are considered. For each scenario, the typical optimal paths determined by the three genetic algorithms are also shown in Table 8.

It can be seen from Fig. 5 that the iteration number of the genetic algorithms by Ahuactzin et al. (1991) and Alajlan et al. (2013), and our proposed one decrease in turn. For example, the iteration number of our proposed algorithm is about 8–82% of that of the genetic algorithm by Ahuactzin et al. (1991).

In order to examine whether there exist significant differences among the performances of all the algorithms used for comparison, Kruskal–Wallis test was conducted on the simulated results, which are shown in Table 9. It can be seen that all p-values for three scenarios are less than 0.05, so the null hypothesis are false, which indicates there exist significant differences among the experimental data of these algorithms. Thus, multiple comparison procedures were carried out to analyze the results of Kruskal–Wallis tests, and the estimated value of the mean ranks for each group is achieved, as shown in Table 10. It can be seen that the estimated values of the mean ranks of iteration number of our proposed algorithm are the smallest one for the three scenarios, which means the computational efficiency of our proposed algorithm is the best among these three algorithms. The main reason is related to the evolving operator proposed in our study.

To further illustrate why our proposed algorithm performs better than the other two algorithms, Fig. 6 shows the comparison of iteration number of our proposed genetic algorithm with and without the evolving operator. It is interest to observe from Fig. 6 that the genetic algorithm

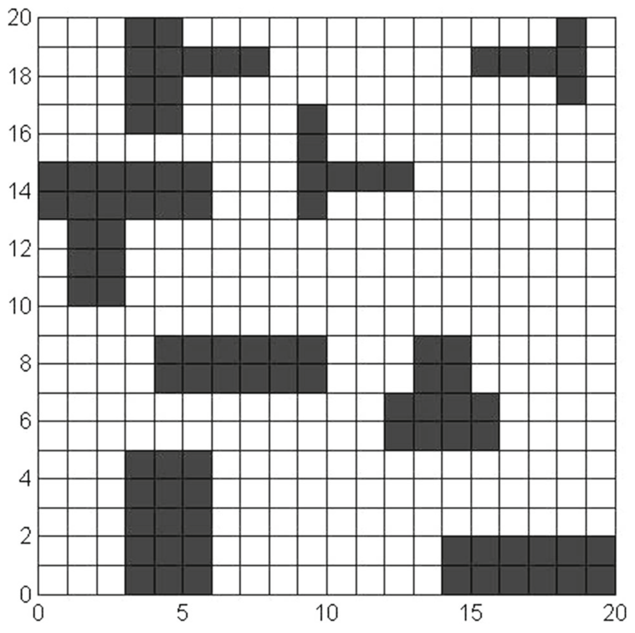


Fig. 4 2D grid-based model used in the experiments to evaluate the iteration number

Table 6 Summary of genetic algorithms used in the experimental simulations

Index	Algorithms			
	GA by Ahuactzin et al. (1991)	GA by Ahmed and Deb (2013)	GA by Alajlan et al. (2013)	GA in this study
Initialization of population	Random number method	Random number method	Greedy approach based on Euclidean distance	Random number method
Fitness function	Single constraint fitness function	Multi-constraint fitness function	Single constraint fitness function	Multi-constraint fitness function
Selection operation	Roulette wheel selection	Modified NSGA-II's selection scheme	Combined elitist selection and rank selection	Deterministic sampling selection
Crossover operation	Two-point crossover	Two-point crossover	Modified crossover operator	Single-point crossover
Mutation operation	Simple mutation	Bit-wise mutation	Single-point mutation	Two-point mutation
Evolving operation	–	–	–	Randomly evolving algorithm

Table 7 Summary of the parameters in the genetic algorithms used for comparison

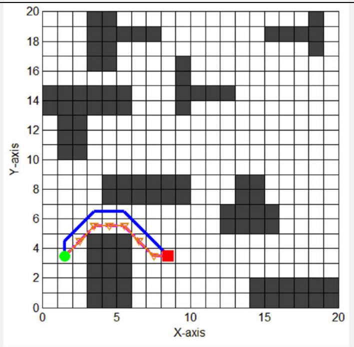
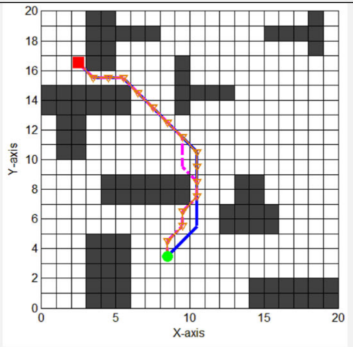
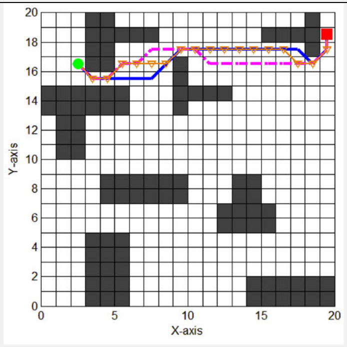
Parameter	Algorithms			
	GA by Ahuactzin et al. (1991)	GA by Ahmed and Deb (2013)	GA by Alajlan et al. (2013)	GA in this study
Number of tests for each case	50			
Population size	200			
Maximum number of iteration	100, 1000, 4000, and 8000 for the environment with grids of 20×20, 16×16, 32×32, and 64×64, respectively			
Crossover probability	0.8			Self-adaptive
Mutation probability	0.2			Self-adaptive
Penalty coefficients in fitness function	$1 \times$ length	$0.4 \times$ length $+ 0.4 \times$ enhancing safety $+0.2 \times$ smoothness	$1 \times$ length	$0.38 \times$ free obstacle $+ 0.04 \times$ length $+ 0.56 \times$ advance visibility $+0.02 \times$ smoothness

with the evolving operator requires less iteration number in all scenarios, which is only 4–76% of that by the genetic algorithm without the evolving operator. Through analyzing these data with Kruskal–Wallis tests (Table 11), we can find that the algorithm with the evolving operator is different from that without the evolving operator since all the *p*-values are less than 0.05. Table 12 summarizes the estimated values of the mean ranks of iteration number for our proposed genetic algorithm with and without the evolving operator. It can be seen from Table 12 that the estimated value of the mean ranks of iteration number of our proposed genetic algorithm with the evolving operator is the smallest, which verifies again that the evolving operator proposed in this study can contribute greatly to the reduction of iteration number of genetic algorithm for path planning problem.

4.3 Comparison on visibility in advance

In order to evaluate the advance visibility of objective of interest on the path, this section conducted experimental simulation in different scenarios shown in Table 13. The position to observe the objective of interest on typical paths obtained by different genetic algorithms is also marked in Table 13. Figure 7 shows the comparison of the advance visibility of objective by the genetic algorithms by Ahuactzin et al. (1991) and Alajlan et al. (2013), and our proposed genetic algorithm. A smaller advance visibility means that the genetic algorithm performs better in terms

Table 8 Typical paths obtained by different genetic algorithms for Scenario 1, Scenario 2, and Scenario 3

Scenarios	Scenario 1	Scenario 2	Scenario 3
Typical path			
Iteration number-GA by Ahuactzin et al. (1991)	3	86	15
Iteration number-GA by Alajlan et al. (2013)	5	12	5
Iteration number-GA in this study	**2**	**7**	**5**

Note: ● denotes the starting point, ■ denotes the ending point.
-·-·- is the path generated by GA by Ahuactzin et al. (1991).
—▽— is the path generated by GA by Alajlan et al. (2013).
—— is the path generated by GA in this study.

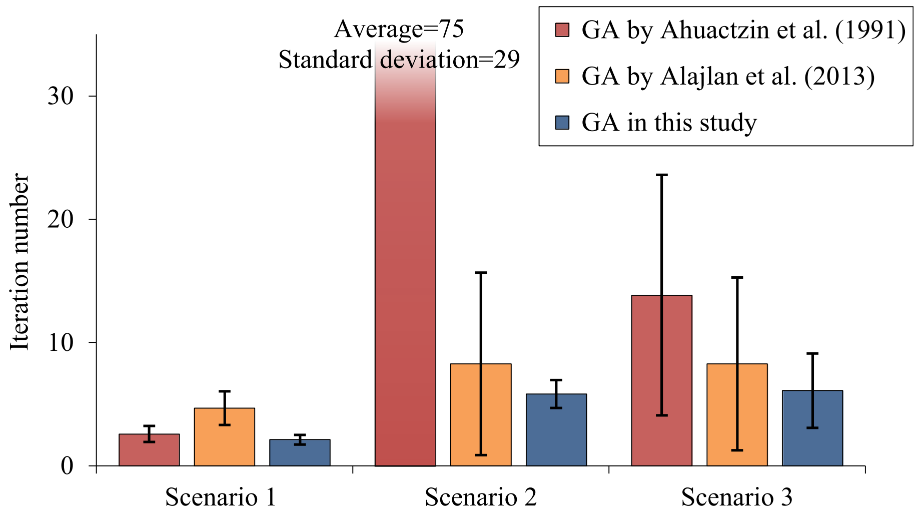


Fig. 5 Comparison of iteration number of different genetic algorithms to find the shortest path in the environment shown in Fig. 4

of advance visibility constraint. It can be seen from Fig. 7 that the advance visibility determined by the genetic algorithms by Ahuactzin et al. (1991) and Alajlan et al. (2013) and our proposed one decrease in turn. For example, the advance visibility of our proposed algorithm is about 57–80% (79–92%) of that of the genetic algorithm by Ahuactzin et al. (1991) (the genetic algorithm by Alajlan et al. (2013)).

Then, both Kruskal–Wallis test and ANOVA test were performed on the simulated results in terms of advance visibility by different algorithms; the results are shown in Tables 14 and 15. It can be seen from Table 14 that all *p*-values are less than 0.05, which means at least one algorithm is significantly different from others. Based on the estimated value of the mean ranks of advance visibility shown in Table 15, it can be seen that the estimated value of the mean ranks of advance visibility by our proposed method is the smallest. The reason lies in that only our proposed algorithm considers advance visibility constraint in the fitness function.

Table 9 Kruskal–Wallis ANOVA Table of iteration number for different genetic algorithms to find the shortest path in the environment shown in Fig. 4

Scenarios	Source	SS	*df*	MS	Chi-sq	Prob > Chi-sq
Scenario 1	Groups	168758.760	2	84379.380	101.566	8.817E−23
	Error	78815.740	147	536.161		
	Total	247574.500	149			
Scenario 2	Groups	186954.520	2	93477.260	99.749	2.187E−22
	Error	92308.480	147	627.949		
	Total	279263.000	149			
Scenario 3	Groups	56602.330	2	28301.165	30.271	2.744E−7
	Error	222499.170	147	1513.600		
	Total	279101.500	149			

Table 10 The estimated values of the mean ranks of iteration number for different genetic algorithms to find the shortest path in the environment shown in Fig. 4

Scenarios	Source	Iteration number-GA by Ahuactzin et al. (1991)	Iteration number-GA by Alajlan et al. (2013)	Iteration number-GA in this study
Scenario 1	Estimated value of the mean ranks	63.66000	121.20000	41.64000
	Standard errors	4.631032	4.631032	4.631032
Scenario 2	Estimated value of the mean ranks	125.38000	52.44000	48.68000
	Standard errors	5.011781	5.011781	5.011781
Scenario 3	Estimated value of the mean ranks	102.14000	67.99000	56.37000
	Standard errors	7.781002	7.781002	7.781002

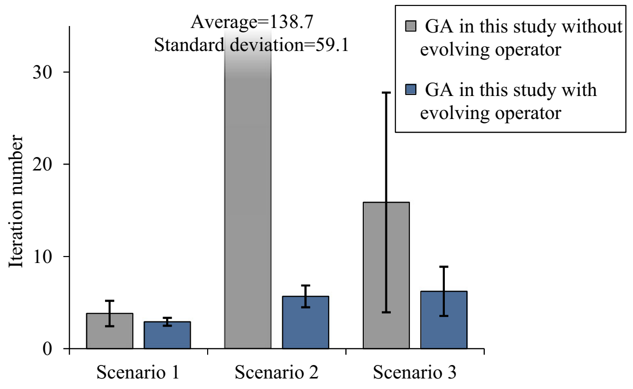


Fig. 6 Comparison of iteration number of our proposed genetic algorithms with and without the evolving operator

Based on the above results, it can be concluded that the visibility constraint of objective of interest in advance during the roaming of intelligent camera can be realized by our proposed genetic algorithm.

4.4 Comparison on path length

The length of the optimal path is an important index to assess the performance of the genetic algorithm in the path planning problem. In this study, three environments shown in Table 16, namely the 16 × 16-gird environment (Scenario 7), the 32 × 32-gird environment (Scenario 8), and the 64 × 64-gird environment (Scenario 9), were used in the experimental simulations. It is noted that the minimum unit of path length shown in this section is one pixel. In addition to our proposed genetic algorithm, the genetic algorithms by Ahuactzin et al. (1991), Ahmed and Deb (2013), and Alajlan et al. (2013) were used in the experimental simulation. The typical optimal path determined by these four genetic algorithms in the three environments is also plotted in Table 16. It can be seen that the optimal path varies with the employed genetic algorithm.

Figure 8 shows the comparison of length of the optimal paths determined by different genetic algorithms. As the environment becomes more complex, i.e., the grid size of the environment becomes larger, the length difference between the original and modified genetic algorithms is more significant. For example, the length of the optimal path can be reduced by up to 39–97% by our proposed genetic algorithm as compared with the one by Ahuactzin et al. (1991).

Based on the Kruskal–Wallis ANOVA Table of length of the optimal paths determined by different genetic algorithms shown in Table 17, it can be seen that all *p*-values are less than 0.05, which means at least one algorithm is significantly different from others. Table 18 shows the estimated value of the mean ranks of path length by

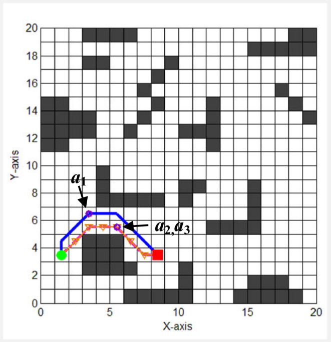
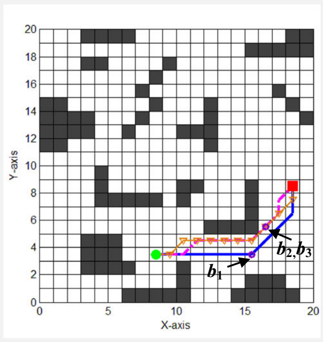
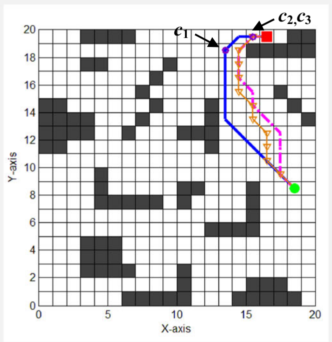
Table 11 Kruskal–Wallis ANOVA Table of iteration number for our proposed genetic algorithms with and without the evolving operator

Scenarios	Source	SS	*df*	MS	Chi-sq	Prob > Chi-sq
Scenario 1	Groups	36290.250	1	36290.250	51.739	6.339E−13
	Error	33149.250	98	338.258		
	Total	69439.500	99			
Scenario 2	Groups	62500.000	1	62500.000	74.901	4.948E−18
	Error	20108.500	98	205.189		
	Total	82608.500	99			
Scenario 3	Groups	23012.890	1	23012.890	27.487	1.582E−7
	Error	59874.110	98	610.960		
	Total	82887.000	99			

Table 12 The estimated values of the mean ranks of iteration number for our proposed genetic algorithms with and without the evolving operator

Scenarios	Source	Iteration number-GA in this study without the evolving operator	Iteration number-GA in this study with the evolving operator
Scenario 1	Estimated value of the mean ranks	69.55000	31.45000
	Standard errors	3.678356	3.678356
Scenario 2	Estimated value of the mean ranks	75.50000	25.50000
	Standard errors	2.864882	2.864882
Scenario 3	Estimated value of the mean ranks	65.67000	35.33000
	Standard errors	4.943522	4.943522

Table 13 The position to observe the objective of interest on typical paths obtained by different genetic algorithms for Scenario 4, Scenario 5, and Scenario 6

Scenarios	Scenario 4	Scenario 5	Scenario 6
Typical path			
Advance visibility-GA by Ahuactzin et al. (1991)	0.69	0.88	6.54
Advance visibility-GA by Alajlan et al. (2013)	0.69	0.88	6.54
Advance visibility-GA in this study	**0.55**	**0.70**	**6.04**

Note: ● denotes the starting point, 🟥 denotes the ending point, ○ denotes the position to observe the objective of interest.
-·-·- is the path generated by GA by Ahuactzin et al. (1991).
—▽— is the path generated by GA by Alajlan et al. (2013).
—— is the path generated by GA in this study.

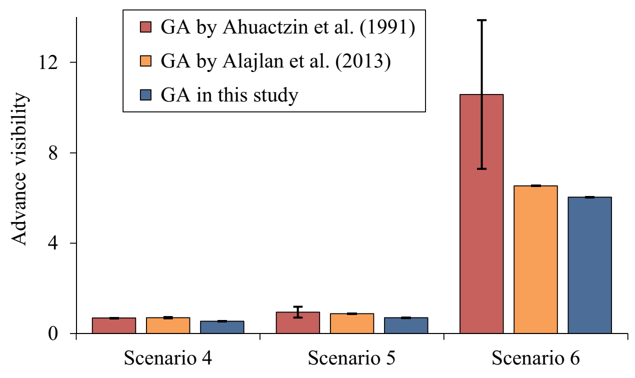


Fig. 7 Comparison of advance visibility of different genetic algorithms to find the shortest path in the environment shown in Table 13

different algorithms, which reveals that the estimated value of the mean ranks of path length by our proposed algorithm is the smallest. What's more, the length of the optimal path determined by our proposed genetic algorithm is slightly shorter than that of the other two genetic algorithms in the environments shown in Table 16, confirming that our proposed genetic algorithm can perform well in the path planning problem in terms of the path length constraint.

4.5 Comparison on path smoothness

Path smoothness is also a key factor that needs to be considered during the path planning for intelligent camera, since the dizziness effect caused by the sharp turn of the intelligent camera will lead to a bad user experience. This section is dedicated to the assessment of the smoothness of the optimal path determined by our proposed genetic algorithm. For better illustration, the other three genetic algorithms used in Sect. 4.4 were employed in the experimental simulations.

Figure 9 shows the comparison of smoothness of the optimal paths determined by different genetic algorithms in the environments shown in Table 16. The smoothness is evaluated by the mean turning angle shown in Eq. 9. Similar to path length results, the genetic algorithm by Ahuactzin et al. (1991) gives an optimal path with less smoothness; the smoothness difference between the genetic algorithm by Ahuactzin et al. (1991) and the other ones tends to increase with increasing grid size of the environment.

In this section, both Kruskal–Wallis test and ANOVA test were performed on the simulated results in terms of

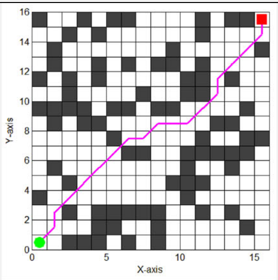
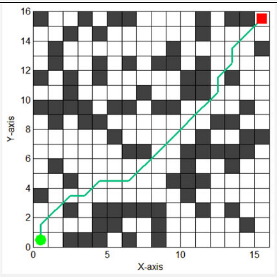
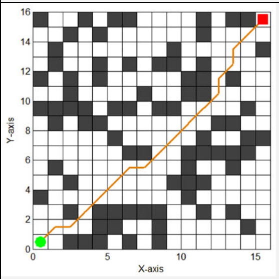
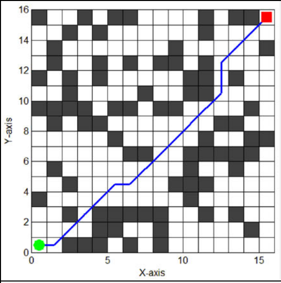
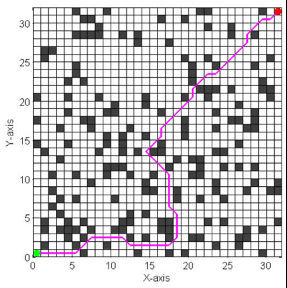
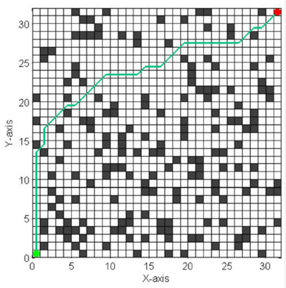
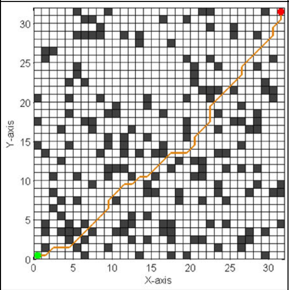
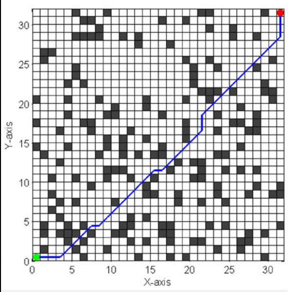
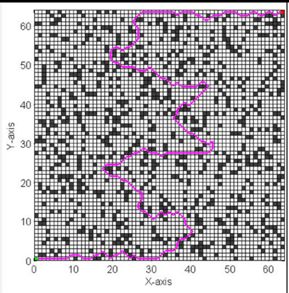
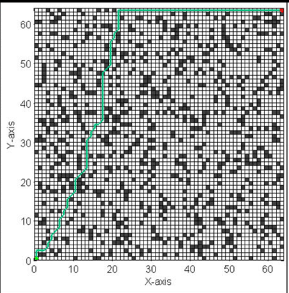
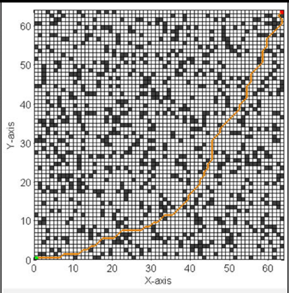
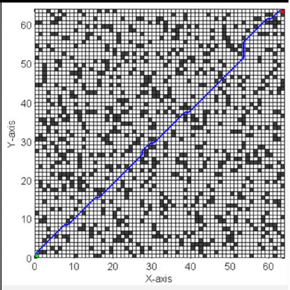
Table 14 Kruskal–Wallis ANOVA Table of advance visibility for different genetic algorithms to find the shortest path in the environment shown in Table 13

Scenarios	Source	SS	*df*	MS	Chi-sq	Prob > Chi-sq
Scenario 4	Groups	187600.000	2	93800.000	145.283	2.833E−32
	Error	4800.000	147	32.653		
	Total	192400.000	149			
Scenario 5	Groups	177855.310	2	88927.655	115.354	8.935E−26
	Error	51875.190	147	352.892		
	Total	229730.500	149			
Scenario 6	Groups	225525.000	2	112762.500	132.449	1.734E−29
	Error	28181.500	147	191.711		
	Total	253706.500	149			

Table 15 The estimated values of the mean ranks of advance visibility for different genetic algorithms to find the shortest path in the environment shown in Table 13

Scenarios	Source	Advance visibility-GA by Ahuactzin et al. (1991)	Advance visibility-GA by Alajlan et al. (2013)	Advance visibility-GA in this study
Scenario 4	Estimated value of the mean ranks	99.50000	101.50000	25.50000
	Standard errors	1.142857	1.142857	1.142857
Scenario 5	Estimated value of the mean ranks	102.59000	97.00000	26.91000
	Standard errors	3.757086	3.757086	3.757086
Scenario 6	Estimated value of the mean ranks	120.00000	81.00000	25.50000
	Standard errors	2.769194	2.769194	2.769194

Table 16 Comparison of path length and path smoothness of the optimal path determined by different algorithms in 2D grid-based environments

Algorithms		GA by Ahuactzin et al. (1991)	GA by Ahmed and Deb (2013)	GA by Alajlan et al. (2013)	GA in this study
16×16-grid environment	Typical path				
	length	23	23	**22**	**22**
	smoothness	0.31	0.31	0.28	**0.18**
32×32-grid environment	Typical path				
	Length	59	53	48	**47**
	Smoothness	0.34	0.19	0.33	**0.13**
64×64-grid environment	Typical path				
	length	237	115	104	**93**
	smoothness	0.61	0.22	0.32	**0.14**

Note: ● denotes the starting point, 🟥 denotes the ending point.

path smoothness by different algorithms; the results are shown in Tables 19 and 20, respectively. It can be seen from Table 19 that all p-values are less than 0.05, which means at least one algorithm is significantly different from others. Based on the estimated value of the mean ranks of path smoothness shown in Table 20, it can be seen that the estimated value of the mean ranks of path smoothness by our proposed algorithm is the smallest, which indicates that our proposed algorithm performs better than other algorithms in terms of path smoothness. What's more, the mean turning angle of the optimal path determined by our proposed genetic algorithm is smaller than that of the other two genetic algorithms in the environments shown in Table 16, which indicates that the dizziness effect during the roaming of intelligent camera can be minimized by our proposed genetic algorithm.

The above experimental results reveal that our proposed genetic algorithm is capable of reducing the dizziness effect during the path planning of intelligent camera.

5 Conclusions

In this study, a modified genetic algorithm for the path planning problem of intelligent camera was proposed. In the proposed method, a new fitness function was developed with the goal to meet the multi-constraint requirements during the roaming of intelligent camera. In addition, an evolving operator was also introduced into the modified

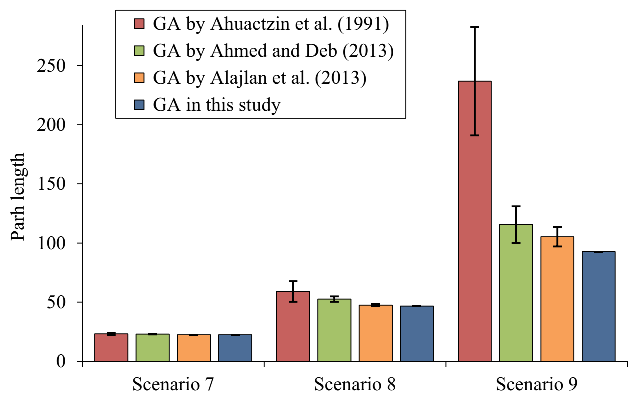


Fig. 8 Comparison of length of the optimal paths determined by different genetic algorithms in the environments shown in Table 16

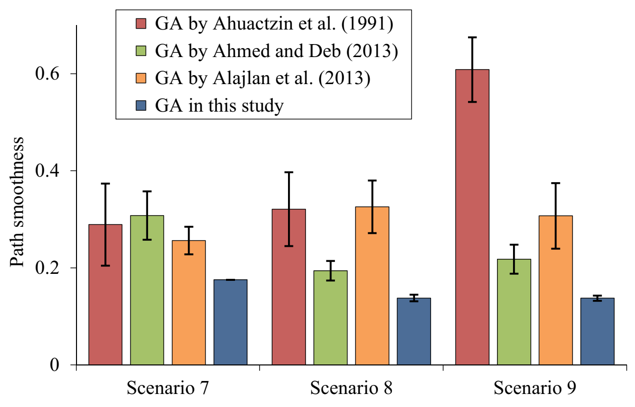


Fig. 9 Comparison of smoothness of the optimal paths determined by different genetic algorithms in the environments shown in Table 16

genetic algorithm, which is designed to improve the efficiency of the algorithm.

Based on the experimental results conducted in different environments, our proposed genetic algorithms perform well in terms of iteration number, visibility in advance, path length, and path smoothness as compared with some state-of-the-art genetic algorithms. This confirms the feasibility and efficiency of our proposed method in the path planning problem of intelligent camera under multiple constraint conditions.

For the future work, different experiments in a real large-scale dynamic environment will be conducted in order to further verify and improve our proposed genetic algorithm. In addition, we would like to extend this work to

Table 17 Kruskal–Wallis ANOVA Table of length of the optimal paths determined by different genetic algorithms in the environments shown in Table 16

Scenarios	Source	SS	*df*	MS	Chi-sq	Prob > Chi-sq
Scenario 7	Groups	182313.610	3	60771.203	83.788	4.723E−18
	Error	250687.890	196	1279.020		
	Total	433001.500	199			
Scenario 8	Groups	360185.840	3	120061.947	115.404	7.534E−25
	Error	260907.660	196	1331.162		
	Total		199			
Scenario 9	Groups	569040.010	3	189680.003	173.157	2.648E−37
	Error	84925.490	196	433.293		
	Total	653965.500	199			

Table 18 The estimated values of the mean ranks of length of the optimal paths determined by different genetic algorithms in the environments shown in Table 16

Scenarios	Source	Path length-GA by Ahuactzin et al. (1991)	Path length-GA by Ahmed and Deb (2013)	Path length-GA by Alajlan et al. (2013)	Path length-GA in this study
Scenario 7	Estimated value of the mean ranks	125.69000	135.31000	70.50000	70.50000
	Standard errors	7.152677	7.152677	7.152677	7.152677
Scenario 8	Estimated value of the mean ranks	163.40000	111.08000	78.52000	49.00000
	Standard errors	7.297017	7.297017	7.297017	7.297017
Scenario 9	Estimated value of the mean ranks	175.26000	109.75000	91.49000	25.50000
	Standard errors	4.163140	4.163140	4.163140	4.163140

Table 19 Kruskal–Wallis ANOVA Table of smoothness of the optimal paths determined by different genetic algorithms in the environments shown in Table 16

Scenarios	Source	SS	*df*	MS	Chi-sq	Prob > Chi-sq
Scenario 7	Groups	378420.910	3	126140.303	116.605	4.155E−25
	Error	267400.090	196	1364.286		
	Total	645821.000	199			
Scenario 8	Groups	432582.190	3	144194.063	133.067	1.181E−28
	Error	214339.310	196	1093.568		
	Total	646921.500	199			
Scenario 9	Groups	592691.560	3	197563.853	179.561	1.097E−38
	Error	64164.440	196	327.370		
	Total	656856.000	199			

Table 20 The estimated values of the mean ranks of smoothness of the optimal paths determined by different genetic algorithms in the environments shown in Table 16

Scenarios	Source	Path smoothness-GA by Ahuactzin et al. (1991)	Path smoothness-GA by Ahmed and Deb (2013)	Path smoothness-GA by Alajlan et al. (2013)	Path smoothness-GA in this study
Scenario 7	Estimated value of the mean ranks	122.71000	141.00000	110.79000	27.50000
	Standard errors	7.387249	7.387249	7.387249	7.387249
Scenario 8	Estimated value of the mean ranks	143.25000	73.33000	147.32000	38.10000
	Standard errors	6.613828	6.613828	6.613828	6.613828
Scenario 9	Estimated value of the mean ranks	175.40000	82.96000	118.14000	25.50000
	Standard errors	3.618672	3.618672	3.618672	3.618672

dynamic environments with moving obstacles and goal positions.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Adem T, Mehmet Y (2012) Chromosome coding methods in genetic algorithm for path planning of mobile robots. Computer and information sciences II. Springer London, London

Ahmadi SM, Kebriaei H, Moradi H (2018) Constrained coverage path planning: evolutionary and classical approaches. Robotica 36(6):904–924

Ahmed F, Deb K (2013) Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. Soft Comput 17(7):1283–1299

Ahuactzin JM, Talbi EG, Bessiere P, Mazer E (1991) Using genetic algorithms for robot motion planning. In: Workshop on geometric reasoning for perception and action, pp 84–93

Alajlan M, Koubaa A, Chaari I, Bennaceur H, Ammar A (2013) Global path planning for mobile robots in large-scale grid environments using genetic algorithms. In: 2013 International conference on individual and collective behaviors in robotics (ICBR). IEEE, pp 1–8

Ali MS, Babu NR, Varghese K (2005) Collision free path planning of cooperative crane manipulators using genetic algorithm. J Comput Civ Eng 19(2):182–193

Amiryan J, Jamzad M (2015) Adaptive motion planning with artificial potential fields using a prior path. In: 2015 3rd RSI international conference on robotics and mechatronics (ICROM). IEEE, pp 731–736

Bai W, Xue B, Sun Y (2011) Research on path planning for soccer robot based on improved genenic algorithm. In: 2011 international conference on mechatronic science, electric engineering and computer (MEC). IEEE, pp 1687–1690

Bakdi A, Hentout A, Boutami H, Maoudj A, Hachour O, Bouzouia B (2017) Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. Robot Auton Syst 89:95–109

Brindle A (1980) Genetic algorithms for function optimization. University of Alberta

Bryson S (1996) Virtual reality in scientific visualization. Commun ACM 39(5):62–71

Cai P, Cai Y, Chandrasekaran I, Zheng J (2016) Parallel genetic algorithm based automatic path planning for crane lifting in complex environments. Automat Constr 62:133–147

Chaudhari AM, Apsangi MR, Kudale AB (2017) Improved a-star algorithm with least turn for robotic rescue operations. In: International conference on computational intelligence, communications, and business analytics, pp 614–627

Cheng Z, Sun Y, Liu Y (2011) Path planning based on immune genetic algorithm for UAV. In: 2011 International conference on electric information and control engineering. IEEE, pp 590–593

Contreras-Cruz MA, Ayala-Ramirez V, Hernandez-Belmonte UH (2015) Mobile robot path planning using artificial bee colony and evolutionary programming. Appl Soft Comput 30:319–328

Drucker SM, Zeltzer D (1994) Intelligent camera control in a virtual environment. In: Graphics interface, CIPS, pp 190–190

Elhoseny M, Tharwat A, Hassanien AE (2018) Bezier curve based path planning in a dynamic field using modified genetic algorithm. J Comput Sci-NETH 25:339–350

Geisler T, Manikas TW (2002) Autonomous robot navigation system using a novel value encoded genetic algorithm. In: The 45th Midwest symposium on circuits and systems. IEEE, pp 45–48

Han Z, Wang D, Liu F, Zhao Z (2017) Multi-AGV path planning with double-path constraints by using an improved genetic algorithm. PLoS ONE 12(7):e0181747

Hsu CC, Liu YC (2014) Path planning for robot navigation based on Cooperative Genetic Optimization. In: Proceedings of the 11th IEEE international conference on networking, sensing and control. IEEE, pp 316–321

Hu Y, Yang SX (2004) A knowledge based genetic algorithm for path planning of a mobile robot. In: IEEE international conference on robotics and automation, pp 4350–4355

Hung KT, Liu JS, Chang YZ (2007) A comparative study of smooth path planning for a mobile robot by evolutionary multi-objective optimization. In: International symposium on computational intelligence in robotics and automation. IEEE, pp 254–259

Karami AH, Hasanzadeh M (2015) An adaptive genetic algorithm for robot motion planning in 2D complex environments. Comput Electr Eng 43:317–329

Kruskal WH, Wallis WA (1952) Use of ranks in one-criterion variance analysis. J Am Stat Assoc 47(260):583–621

Kuang Y, Jin J, Su Y (2006) Improving crossover and mutation for adaptive genetic algorithm. Comput Eng Appl 12:93–96+99 **(in Chinese)**

Lamini C, Benhlima S, Elbekri A (2018) Genetic algorithm based approach for autonomous mobile robot path planning. Procedia Comput Sci 127:180–189

Lee J, Kim DW (2016) An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph. Inform Sci 332:1–18

Lee TL, Wu CJ (2003) Fuzzy motion planning of mobile robots in unknown environments. J Intell Robot Syst 37(2):177–191

Li M, Wang C, Chen Z, Lu X, Wu M Hou P (2017) Path planning of mobile robot based on genetic algorithm and gene rearrangement. In: 2017 Chinese automation congress (CAC), IEEE, pp 6999–7004

Liang W, Li X (2008) 3D trajectory planning and visualization simulation for helicopter. Comput Aided Eng 2(17):73–76 **(in Chinese)**

Lin HS, Xiao J, Michalewicz Z (1994) Evolutionary navigator for a mobile robot. In: Proceedings of the 1994 IEEE international conference on robotics and automation, IEEE, pp 2199–2204

Liu J, Yang J, Liu H, Tian X, Gao M (2017) An improved ant colony algorithm for robot path planning. Soft Comput 21(19):5829–5839

Manikas TW, Ashenayi K, Wainwright RL (2007) Genetic algorithms for autonomous robot navigation. IEEE Instrum Meas Mag 10(6):26–31

Mansouri M, Shoorehdeli MA, Teshnehlab M (2008) Path planning of mobile robot using integer GA with considering terrain conditions. In: 2008 IEEE international conference on systems, man and cybernetics, pp 208–213

McKight PE, Najab J (2010) Kruskal–Wallis Test. In: The Corsini Encyclopedia of Psychology

Menasri R, Nakib A, Daachi B, Oulhadj H, Siarry P (2015) A trajectory planning of redundant manipulators based on bilevel optimization. Appl Math Comput 250:934–947

Miao YQ, Khamis AM, Karray F, Kamel M (2011) A novel approach to path planning for autonomous mobile robots. Control Intell Sys 39(4):235–244

Mou C, Qing-xian W, Chang-sheng J (2008) A modified ant optimization algorithm for path planning of UCAV. Appl Soft Comput 8(4):1712–1718

Nazarahari M, Khanmirza E, Doostie S (2019) Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. Expert Syst Appl 115:106–120

Patle BK, Parhi DRK, Jagadeesh A, Kashyap S (2018) Matrix-binary codes based genetic algorithm for path planning of mobile robot. Comput Electr Eng 67:708–728

Perez-Hurtado I, Perez-Jimenez MJ, Zhang G, Orellana-Martin D (2018) Simulation of rapidly-exploring random trees in membrane computing with P-Lingua and automatic programming. Int J Comput Commun 13(6):1007–1031

Perez-Hurtado I, Martınez-del-Amor MA, Zhang G, Neri F, Perez-Jimenez MJ (2020) A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning. Integr Comput-Aid E 27(2):121–138

Pierre DM, Zakaria N (2011) Genetic algorithm approach to path planning for intelligent camera control for scientific visualization. In: International conference on software engineering and computer systems, pp 205–213

Pol RS, Murugan M (2015) A review on indoor human aware autonomous mobile robot navigation through a dynamic environment survey of different path planning algorithm and methods. In: 2015 International Conference on Industrial Instrumentation and Control (ICIC), IEEE, pp 1339–1344

Qi C, Min Z, Yanhua J, Min Y (2019) Multi-objective cooperative paths planning for multiple parafoils system using a genetic algorithm. J Aerosp Technol Manag 11:1–12 **(in Chinese)**

Qu H, Xing K, Alexander T (2013) An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots. Neurocomputing 120:509–517

Ramirez-Atencia C, Bello-Orgaz G, R-Moreno MD, Camacho D (2017) Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. Soft Comput 21(17):4883–4900

Sahoo B, Parhi DR, Priyadarshi BK (2018) Analysis of path planning of humanoid robots using neural network methods and study of possible use of other AI techniques. In: Emerging trends in engineering, science and manufacturing, pp 1–16

Shi P, Cui Y (2010) Dynamic path planning for mobile robot based on genetic algorithm in unknown environment. In: 2010 Chinese control and decision conference. IEEE, pp 4325–4329

Song B, Wang Z, Sheng L (2016) A new genetic algorithm approach to smooth path planning for mobile robots. Assembly Autom 36(2):138–145

Srinivas M, Patnaik L (1994) Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE T Syst M Cy-S 24(4):656–667

Sugisaka M, Fan X (2001) Adaptive genetic algorithm with a cooperative mode. In: 2001 IEEE International symposium on industrial electronics proceedings, pp 941–1945

Tharwat A, Elhoseny M, Hassanien AE, Gabel T, Kumar A (2019) Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm. Cluster Comput 22(2):4745–4766

Trujillo MM, Darrah M, Speransky K, DeRoos B, Wathen M (2016) Optimized flight path for 3D mapping of an area with structures using a multirotor. In: 2016 International conference on unmanned aircraft systems. IEEE, pp 905–910

Tsai C, Huang H, Chan C (2011) Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation. IEEE Trans Ind Electron 58(10):4813–4821

Tuncer A, Yildirim M (2012) Dynamic path planning of mobile robots with improved genetic algorithm. Comput Electr Eng 38(6):1564–1572

Vadakkepat P, Tan KC, Ming-Liang W (2000) Evolutionary artificial potential fields and their application in real time robot path planning. In: Proceedings of the 2000 congress on evolutionary computation. IEEE, pp 256–263

Vargha A, Delaney HD (1998) The Kruskal–Wallis test and stochastic homogeneity. J Educ Behav Stat 23(2):170–192

Wang C, Hwang R (2009) Context-aware path planning in ubiquitous network. In: International conference on ubiquitous intelligence and computing, pp 54–67

Wang C, Tao J (2017) Graphs in scientific visualization: a survey. In: Computer graphics forum, pp 263–287

Wang Q, Yao J, Wang J (2004) A path planning approach to moving robot based on genetic algorithms. J Harbin Inst Technol 7:867–870 **(in Chinese)**

Wang X, Zhang G, Zhao J, Rong H, Ipate F, Lefticaru R (2015) A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning. Int J Comput Commun 10(5):732–745

Wang L, Luo C, Li M, Cai J (2017) Trajectory planning of an autonomous mobile robot by evolving ant colony system. Int J Robot Autom 32(4):406–413

Wei T, Long C (2019) Path planning for mobile robot based on improved genetic algorithm. J Beijing Univ Aeronaut Astronaut (in Chinese)

Xiao J, Michalewicz Z, Zhang L, Trojanowski K (1997) Adaptive evolutionary planner/navigator for mobile robots. IEEE Trans Evol Comput 1(1):18–28

Xidias EK, Nearchou AC, Aspragathos NA (2012) Integrating path planning, routing, and scheduling for logistics operations in manufacturing facilities. Cybernet Syst 43(3):143–162

Xidias E, Paliotta C, Aspragathos N, Pettersen K (2016) Path planning for formation control of autonomous vehicles. In: International conference on robotics in Alpe-Adria Danube Region, pp 302–309

Yang X, Cai M, Li J (2016) Path planning for unmanned aerial vehicles based on genetic programming. In: 2016 Chinese control and decision conference. IEEE, pp 717–722

Yao Z, Ma L (2010) A static environment-based path planning method by using genetic algorithm. In: 2010 International conference on computing, control and industrial engineering. IEEE, pp 405–407

Yu Z, Liang J, Gu G, Zhang R, Yang H (2002) An implementation of evolutionary computation for path planning of cooperative mobile robots. In: Proceedings of the 4th world congress on intelligent control and automation. IEEE, pp 1798–1802

Yun W, Xi Y (1996) Optimum motion planning in joint space for robots using genetic algorithms. Robot Auton Syst 18(4):373–393

Zhang Q, Ding L (2016) A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems. Expert Syst Appl 60:183–189

Zhang Y, Gong DW, Zhang JH (2013) Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing 103:172–185

Zhang G, Perez-Jimenez MJ, Gheorghe M (2017) Real-life applications with membrane computing. Springer, Berlin, pp 33–115

Zhou W, Yi Z, Ruimin Y (2008) Mobile robot path planning based on genetic algorithm. Microcom Inform 24(26):187–189