Nakane *et al. IPSJ Transactions on Computer Vision and Applications* (2020) 12:3
https://doi.org/10.1186/s41074-020-00065-9

IPSJ Transactions on Computer Vision and Applications

REVIEW PAPER **Open Access**

Application of evolutionary and swarm optimization in computer vision: a literature survey

Takumi Nakane[1†], Naranchimeg Bold[2†], Haitian Sun[2†], Xuequan Lu[3], Takuya Akashi[2] and Chao Zhang[1*]

Abstract

Evolutionary algorithms (EAs) and swarm algorithms (SAs) have shown their usefulness in solving combinatorial and NP-hard optimization problems in various research fields. However, in the field of computer vision, related surveys have not been updated during the last decade. In this study, inspired by the recent development of deep neural networks in computer vision, which embed large-scale optimization problems, we first describe a literature survey conducted to compensate for the lack of relevant research in this area. Specifically, applications related to the genetic algorithm and differential evolution from EAs, as well as particle swarm optimization and ant colony optimization from SAs and their variants, are mainly considered in this survey.

Keywords: Evolutionary algorithms, Swarm algorithms, Computer vision, Literature survey

1 Introduction

Many computer vision tasks can be regarded and formulated as a convex optimization, which allows a global optimum to be mathematically computed [110–112]. However, most of these tests can be highly non-convex and even ill-posed. As a result, there may exist numerous optima, with no solution, a non-unique solution, or an unstable solution, particularly under real-world settings that involve noisy or missing data. Regarding the non-convexity, for example, segmentation problems (Section 5) in computer vision can be cast as an energy minimization problem, which is applied to formulate an energy function over labels of pixels, such that the best solution can be obtained by minimizing the amount of energy. However, when the energy function given is complex, finding the exact energy minimum is NP-hard and the convex solvers are unable to explore the exponential number of local optima efficiently without adding additional constraints or hypotheses. Regarding the ill-posed problem, many tasks require optimizing the parameters of a certain mathematical model to reproduce the observations. For example, in face recognition problems (Section 9), there are various parameters that need to be tuned to model a "face likeness." Depending on the amount and quality of the training samples, finding a parameter setting that can reproduce the training labels could be extremely difficult.

By contrast, evolutionary algorithms (EAs) and swarm algorithms (SAs) are powerful metaheuristic tools used to search for solutions within a potentially huge solution space or provide approximate solutions for solving combinational constraints that may not hold stable solutions. To avoid being trapped in the local optima and provide a satisfactory solution, EAs and SAs have been successfully adopted to solve various computer vision tasks, which are listed and classified in this survey.

To the best of our knowledge, there have been no other studies specifically providing a comprehensive survey of EAs and SAs adopted to solving computer vision problems. Despite the many recent applications in computer

*Correspondence: zhang@u-fukui.ac.jp
†Takumi Nakane, Naranchimeg Bold and Haitian Sun contributed equally to this work.
[1]Department of Engineering, University of Fukui, Fukui, Japan
Full list of author information is available at the end of the article

vision combining deep neural networks with evolutionary optimization in recent years, we are interested in how the EAs and SAs for computer vision-related tasks have evolved. The main purpose of this paper is to present a comprehensive understanding of the existing research on EAs and SAs for solving computer vision tasks.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce the characteristics of the algorithms focused upon in this survey. In Section 3, we discuss why EAs and SAs are needed for computer vision applications based on a simple example. In Section 4 through Section 11, we explain how EAs and SAs have been applied to eight different computer vision tasks. For clarity, the summary of contents of this paper is shown in Table 1. Finally, we summarize the contents of this paper in Section 12.

2 Evolutionary and swarm algorithms

EAs and SAs are two important research fields belonging to the nature-inspired metaheuristics known as an evolutionary computation (EC). These metaheuristics share the following two characteristics: population-based presentation for the candidate solutions, and an iterative procedure with a stochastic exploration.

A significantly important factor in a population-based optimization method is a balance between exploration and exploitation capabilities. An exploration is the ability to search over a wide range of solution spaces by uniformly distributing the population (i.e., the population maintains its diversity). This brings robustness for a non-convex function landscape to a population. Even if some individuals fall into local optima, others may still be able to find a promising solution. By contrast, an exploitation is the ability to concentrate a population at a promising solution based on information that has been acquired thus far. An exploitation is necessary to obtain a converged population. The more valid and reliable the information shared within a population is, the faster a convergence can be achieved. The successes of EAs and SAs are derived from the nature-inspired operations potentially having a mechanism to adjust the above two abilities. These algorithms start from a state in which individuals are randomly distributed, i.e., in the most diverse state, and operations are designed to encourage convergence within the population and achieve balance between the two abilities automatically.

In this survey, we concentrate on studying approaches relevant to the following four representative algorithms: the genetic algorithm (GA) and differential evolution (DE) from the EAs, and particle swarm optimization (PSO) and ant colony optimization (ACO) from the SAs. For a simple comparison, brief flowcharts of these algorithms are shown in Fig. 1. More detailed procedures can be found in the pseudo-codes of Appendix A. In the following, we review the algorithms involved by analyzing their features and differences.

2.1 Evolutionary algorithms

EAs are optimization algorithms inspired by Darwin's evolutionary theory. This generic category mainly consists

Table 1 Summary of contents from Section 4 to Section 11. Background colors represent different categories of algorithms: 🟥 GA, 🟩 DE, 🟦 PSO, and 🟨 ACO

Section	Subsection	Literature										
Sec. 4 Neural Network	Sec. 4.1 Evolving DNNs for Image Classification	🟥 [1]	🟥 [2]	🟥 [3]	🟥 [4]	🟥 [5]	🟥 [6]	🟥 [7]	🟥 [8]	🟥 [9]	🟦 [10]	🟦 [11]
	Sec. 4.2 Evolving DNNs for Image Restoration	🟥 [12]	🟥 [13]									
	Sec. 4.3 Evolving DNNs for Other Tasks	🟥 [14]	🟥 [15]									
Sec. 5 Image Segmentation	Sec. 5.1 Thresholding Approaches	🟥 [16]	🟨 [17]	🟦 [18]	🟨 [19]	🟦 [20]	🟨 [21]	🟨 [22]	🟦 [23]			
	Sec. 5.2 Clustering Approaches	🟦 [24]	🟨 [25]	🟥 [26]	🟦 [27]	🟥 [28]	🟥 [29]	🟨 [30]	🟥 [31]	🟥 [32]		
	Sec. 5.3 Other Approaches	🟨 [33]	🟥 [34]	🟥 [35]	🟨 [36]	🟨 [37]						
Sec. 6 Feature Detection and Selection	Sec. 6.1 Feature Detection	🟨 [38]	🟨 [39]	🟩 [40]	🟥 [41]	🟦 [41]	🟥 [42]	🟥 [43]	🟥 [44]	🟥 [45]		
	Sec. 6.2 Feature Selection	🟥 [46]	🟥 [47]	🟩 [48]	🟩 [49]	🟩 [50]	🟦 [51]	🟦 [52]	🟨 [53]	🟨 [54]		
Sec. 7 Image Matching	Sec. 7.1 Template Matching	🟥 [55]	🟥 [56]	🟥 [57]	🟥 [58]	🟥 [59]	🟩 [60]	🟦 [60]				
	Sec. 7.2 Image Registration	🟩 [61]	🟩 [62]	🟦 [63]	🟦 [64]							
	Sec. 7.3 Jigsaw-puzzle-like Problems	🟥 [65]	🟥 [66]	🟥 [67]								
	Sec. 7.4 Feature Matching	🟥 [68]	🟥 [69]									
Sec. 8 Visual Tracking	Sec. 8.1 Single Object Tracking	🟥 [70]	🟩 [71]	🟦 [72]	🟦 [73]	🟩 [74]	🟩 [75]					
	Sec. 8.2 Multiple Object Tracking	🟥 [76]	🟦 [77]									
Sec. 9 Face Recognition	Sec. 9.1 Feature Selection/Weighting	🟥 [78]	🟥 [79]	🟥 [80]	🟨 [81]	🟦 [82]	🟦 [83]	🟥 [84]				
	Sec. 9.2 Fusion of Visible and IR Features	🟥 [85]	🟥 [86]	🟥 [87]								
	Sec. 9.3 Other Methods	🟥 [88]	🟥 [89]	🟦 [90]	🟦 [91]	🟨 [91]	🟩 [92]	🟥 [93]	🟥 [94]			
Sec. 10 Human Action Recognition	Sec. 10.1 Human Body	🟦 [95]	🟦 [96]	🟦 [97]	🟥 [98]	🟥 [99]	🟥 [100]	🟩 [101]				
	Sec. 10.2 Human Hands	🟦 [102]	🟦 [103]	🟦 [104]	🟥 [105]							
	Sec. 10.3 Human Head	🟦 [106]										
Sec. 11 Others		🟥 [107]	🟩 [108]	🟩 [109]								

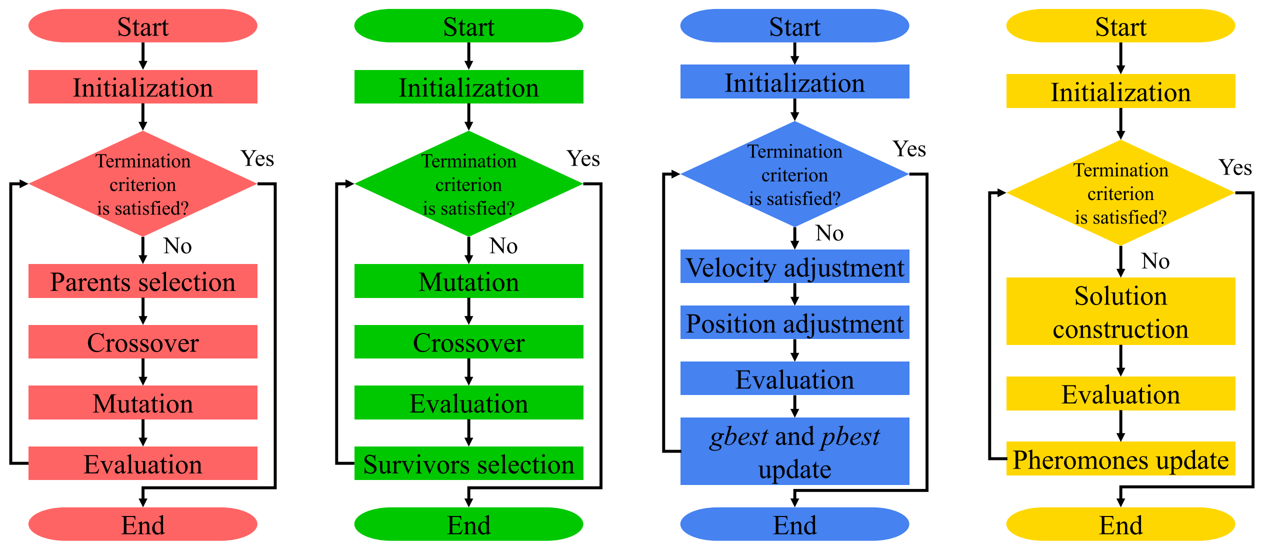


Fig. 1 Comparison of flowcharts between GA, DE, PSO, and ACO. Interested readers can refer to Appendix A for detail pseudo-codes and explanations

of the GA, genetic programming (GP), evolution strategy (ES), and evolutionary programming (EP). It also includes algorithms that have similar frameworks such as a DE algorithm in a broad sense. Each iteration in an EA (i.e., a generation) is composed of parents selection, recombination (i.e., a crossover), mutation, and survivors selection, as shown in Fig. 2. The two selections operate according to the evaluation values (i.e., fitness), which bring about a strong force of exploitation. However, a crossover and mutation are responsible for the exploration within, sometimes outside of, the distribution of the population. These operations together form a simulation of evolution for individuals, which lead the population to the desired solutions. Under the usual settings, an individual represents a single solution candidate.

The GA is the most well-known algorithm in both EAs and EC. An individual is a group of chromosomes, which are typically encoded by a binary code with a fixed length. The parent selection takes the fitness value of all individuals into account, which is implemented probabilistically. The selected parents produce the same number of offspring by a crossover and mutation. These two operations are a partial bit (i.e., gene) manipulation. A crossover produces new individuals by swapping the genes of the parent pairs. That is, a new individual is composed of partial blocks of genes of the parents, which implies the inheritance of the parental characteristics. The purpose of a mutation is to introduce an impact into a population that cannot be acquired by inheritance, which is achieved by changing genes in a completely independent and random manner. A mutation helps the individuals escape from the local optima. Alternation of generation (i.e., survivor selection) is realized by entirely replacing a population of parents with a population of offspring. Because the GA is designed for general purposes, it is often intuitive and simple to apply to real problems. In addition, numerous researchers have been working on developing a real-value coded GA with an improvement

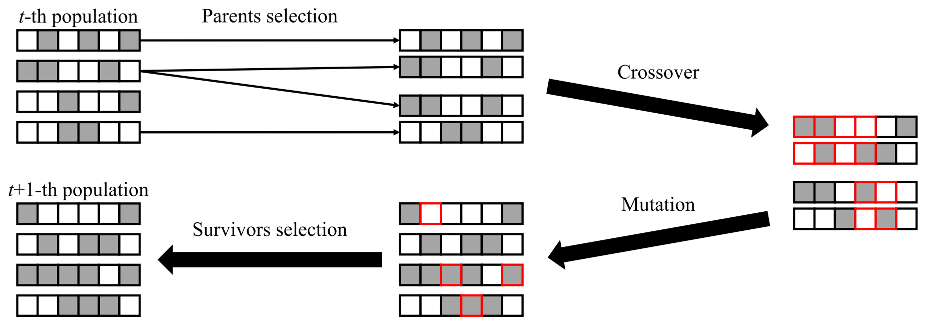


Fig. 2 Example of an one-generation cycle of GA. Modified genes in each step are shown in red

of the genetic operators, which enables the GA to be applied to not only combinatorial optimization but also continuous optimization problems.

A DE is one of the most popular EA optimization algorithms. An individual is termed a parameter vector and composed of real-valued parameters, which allows the algorithm to solve continuous optimization problems. The most significant characteristic of a DE algorithm is the existence of a donor vector constructed during the mutation step from a parameter vector (i.e., a base vector) and a difference vector of two parameter vectors. These three parameter vectors are randomly selected from the current population. The difference vector represents the direction and magnitude of the change caused by a mutation. In addition, selection from the population can reflect valid information from the distribution fitted into the functional landscape. That is, the donor vector is an indicator of the search with an automatic scaling adjustment, which improves the convergence of the algorithm. The survivor selection step in the DE algorithm is a competitive process between the target vector (i.e., parent) and trial vectors (i.e., offspring created from the target vector and donor vectors) based on the fitness values. Unlike the GA, which preserves all offspring until the next generation without exception, the offspring in the DE algorithm must be equal to or outperform the corresponding parent to survive. This strategy implies the preservation of best-so-far solutions individually, which can make the population maintain its diversity and improve its convergence over the long term.

GA and DE repeat the common steps, although the actual implementation of each step differs, as shown in Table 2. Note that this is an example of a simple implementation, and many variants exist.

2.2 Swarm algorithms

SAs, inspired by the collective behavior of social animals and insects, are optimization algorithms belonging to metaheuristics called swarm intelligence (SI). A swarm includes multiple agents, and the behavior of each agent is extremely simple, local, and stochastic. Despite a single swarm not having a centralized structure to control the rule of the agent behavior, interactions between agents introduce global swarms and intelligent behavior. The local behavior of each agent and the interactions shared within the swarm correspond to an exploitation and exploration respectively, and are combined as agent movements within a simple implementation.

PSO is a continuous optimization algorithm inspired by the collective behavior of flocking birds. All individuals (particles) composing the population (swarm) fly around the search space based on the corresponding velocity vector. The most attractive point of PSO is the preservation of two important elements: the global best (*gbest*) and the personal best (*pbest*). These are the memory of the positions where the best fitness values can be observed until the current iteration, with respect to the swarm and each particle, respectively. Here, *gbest* is an element that promotes the convergence of the swarm to the proper locations, whereas *pbest* contributes to the maintenance of the swarm diversity by generating unique behaviors for each particle. Both *gbest* and *pbest* are mainly used for a velocity update by considering the inertia. The velocity update function is similar to the target-to-best (type of base vector)/1 (number of difference vectors) scheme of the mutation step in the DE algorithm, which means that PSO also benefits from the difference vector. By contrast, PSO does not have a selection step like an EA, and an iteration only consists of a self-update of the velocity and position. The simple composition of this algorithm allows for an easy coding and efficient computations.

ACO is a metaheuristic mainly designed for combinatorial optimization problems, inspired by the behaviors of ants. The task of the artificial ants is to construct a candidate solution by adding unused solution components to the current partial solution iteratively. The ants probabilistically choose a solution component based on the pheromone intensity and heuristic information (if available). The pheromone intensity reveals the validity of the corresponding choice, which is updated after the artificial ants construct a candidate solution. The update of the pheromone consists of two mechanisms: deposit and evaporation. Artificial ants increase the pheromones on their own trail according to the evaluation value, and the pheromones will decrease over time. If the choice is optimal, it attracts more artificial ants because the deposit

Table 2 Differences between the implementations of GA and DE with respect to the four common steps of EAs

	GA	DE
Parents selection	Fitness-based selection	Random selection
Crossover	Genes swapping between two parents to generate two offspring	Components selection from parent (target vector) and donor vector to generate one offspring (trial vector)
Mutation	Bit inversion of each gene	Generation of one donor vector from three individuals
Survivors selection	Complete replacement	Fitness-based competition

exceeds the evaporation; otherwise, the choice will soon become uncompetitive. The update of the pheromone is a reflection of the experience accumulated by the artificial ant colony, which will improve the quality of the following candidate solutions. Algorithms that share the framework described above are generally referred to as ACO algorithms.

The key point of the SAs is the information shared within the swarm, which can directly influence the movement of each agent. The differences between PSO and ACO are summarized in Table 3.

2.3 Comparison of algorithm characteristics

Although the above algorithms have a common framework of population-based iterative processing, there are various differences in their specific implementations. In this subsection, we discuss the characteristics of each algorithm, which may provide an indicator to the question of which algorithm is appropriate to exploit.

One advantage of the GA is its flexible gene representation. Owing to its long history and popularity, various gene representations (e.g., binary, real-value, and graph) and corresponding genetic operators have been devised. Owing to the accumulation of these abundant implementations, the GA is widely used in various fields including computer vision.

The DE algorithm is simple to implement, but achieves a high optimization capability. This fact has been proven through numerous competitions on real parameter optimization [113]. Its effectiveness is expected to make it a powerful tool in computer vision as well.

PSO has attracted the interest of researchers owing to its simple implementation. The fast operators are effective for applications that require a high-speed performance. In addition, unlike the crossover operators in the GA and DE algorithm, the majority of PSO processing requires no interactions between particles. This fact shows that PSO is compatible with parallel processing.

A characteristic of ACO is a graph exploration for making probabilistic decisions. This unique process is extremely effective in problems that can be modeled using graphs.

Table 3 Differences between the information shared within the swarms of PSO and ACO

	PSO	ACO
Information shared within the swarm	Global best position (*gbest*)	Pheromone intensity
Information that affects agent movement	Best positions for global (*gbest*) and personal (*pbest*)	Pheromone intensity and heuristic information

3 Applications in computer vision

Computer vision aims to extract and understand meaningful information from images and videos. Various processes for performing such tasks are often interspersed with situations that require optimization, and the solution spaces usually constitute a vast and complex landscape. As a simple example, we demonstrated a simple object detection using a sliding window method, as shown Fig. 3. The reference image (Fig. 3a) is slide from the top left of the target image (Fig. 3b), and the sum of absolute differences (SAD) of the pixels at each position is calculated. That is, detection is achieved by finding the position where the SAD is 0. From the plot of the SAD at each position shown in Fig. 3c, we can observe a non-convex functional landscape. The presence of many small valleys makes optimization through a deterministic method difficult to achieve. In addition, the landscape becomes more complex if we must consider the rotation and scaling of the reference image. Therefore, EAs and SAs are expected to be powerful tools for solving the optimization problems occurring in computer vision.

We systematically summarize the studies in which four selected algorithms are involved with respect to different computer vision tasks: a neural network (Section 4), image segmentation (Section 5), feature detection and selection (Section 6), image matching (Section 7), visual tracking (Section 8), face recognition (Section 9), human action recognition (Section 10), and a few other studies (Section 11). The timeline of the literature summarized in this paper is listed in Table 4, and the statistics of the literature in terms of applications and algorithms are shown in Fig. 4. In addition, we also present a summary table at the end of each section to categorize the related studies.

4 Neural network

During the last two decades, deep neural networks (DNNs) have achieved a state-of-the-art performance on a variety of computer vision tasks, for instance, in object recognition, where problem-specific features can be automatically learned. However, designing and learning optimal network structures and their parameters are challenging tasks, requiring expert knowledge and significant trial and error. Therefore, the development of automated neural architecture search (NAS) methods is an attractive field of research.

There are numerous different strategies used by an NAS, including gradient-based methods, a random search, Bayesian optimization, and reinforcement learning. In particular, the strategy of using EAs and SAs, called NeuroEvolution, has received attention since the introduction of this field. Although gradient-based NAS methods (e.g., [115, 116]) are much faster than evolutionary-based NAS methods in many cases, the gradient-free exploration of EAs and SAs is useful for tasks for which

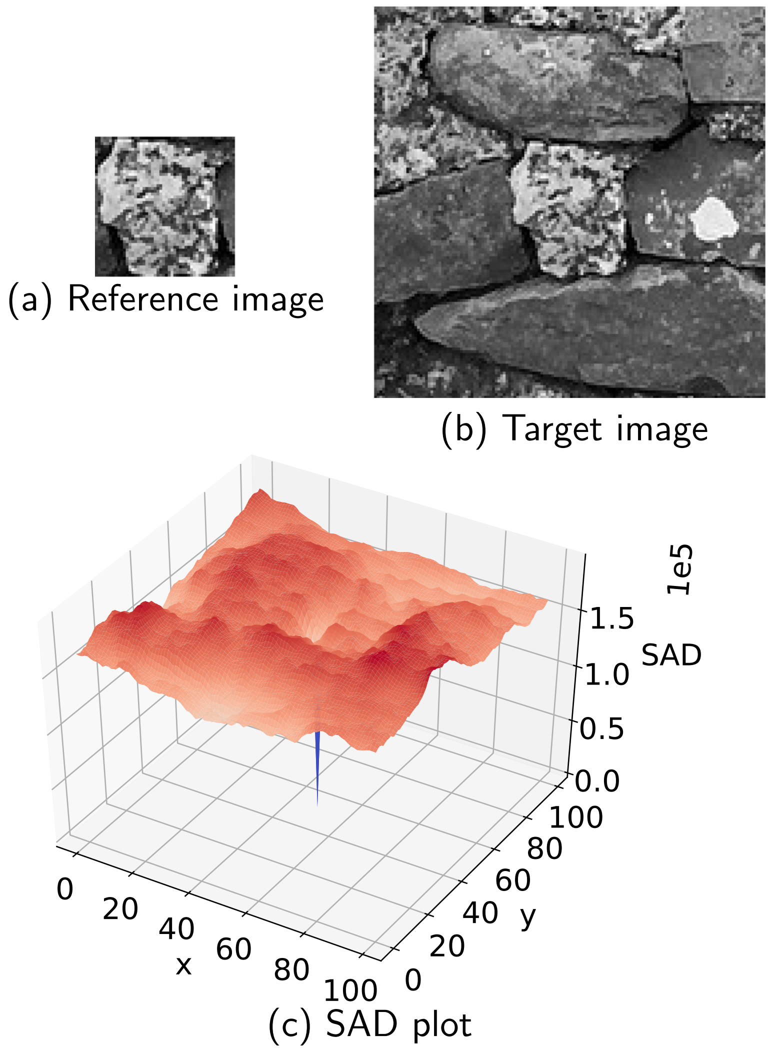


Fig. 3 Demonstration of a simple object detection by maximizing a SAD score. The plots of SAD at each detection window is shown by **c**, which can be observed that there exist many local optima

gradient-based methods are not typically applicable, such as learning building blocks and architectures of neural networks [117]. Interested readers are also referred to survey papers [118] and [117] for the details of NAS strategies and NeuroEvolution approaches, respectively.

Research on NeuroEvolution began in the 1990s with many interesting approaches [119], which were originally used to evolve the weights of a fixed architecture. In 2002, Stanley and Miikkulainen proposed the NEAT algorithm [120] to evolve the structure and connection weights of a small-scale neural network. After the NEAT algorithm, there has been a surging interest in using algorithms such as EAs to automatically design DNNs along with the connection weights and hyperparameters. However, with the dramatically increasing scale of DNNs, it has become difficult for even EAs and SAs to adjust the architectures and weights simultaneously. To address this issue, recent NeuroEvolution approaches again incorporate gradient-based methods to optimize weights [13, 118]. Through a series of efforts, DNNs designed by EAs and SAs achieve competitive performance for reinforcement learning [121] and image classification tasks [1]. Nonetheless, for supervised learning tasks, gradient-based optimization is by far the most common approach.

This section reviews NeuroEvolution approaches, which optimize the DNN structure, connection weights and hyperparameters with respect to computer vision tasks (particularly image classification tasks). We first describe several studies on discovering the structure of neural networks for large-scale image classification benchmarks using EAs and SAs in Section 4.1. Next, some studies on the evolving structure for image restoration are

Table 4 Timeline (2000∼2018) and categorization of research works summarized in this survey. Background colors represent algorithm categories 🟥GA, 🟩DE, 🟦PSO, and 🟨ACO

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Sec. 4					🟥[14]											🟥[15]	🟥[12]	🟥[1] 🟥[2] 🟥[3] 🟥[4] 🟥[5] 🟦[10]	🟥[6] 🟥[7] 🟥[8] 🟥[9] 🟦[11] 🟥[13]
Sec. 5		🟥[35]		🟥[16] 🟥[26] 🟨[33] 🟥[34]		🟦[24] 🟨[36]	🟨[19] 🟨[25] 🟦[27]	🟨[17] 🟥[28]		🟦[18] 🟥[29] 🟨[37]		🟨[21] 🟦[23] 🟨[30] 🟥[31]		🟨[22]	🟦[20]		🟥[32]		
Sec. 6			🟥[46]		🟥[47]	🟨[53]	🟨[38] 🟥[42]		🟥[43] 🟩[48]	🟥[44]	🟨[39]	🟩[40] 🟩[49]	🟥[41] 🟦[41]	🟥[45] 🟩[50] 🟨[54]		🟦[51]	🟦[52]		
Sec. 7		🟥[59] 🟥[68]		🟥[69]	🟦[63]			🟦[64]	🟩[61]					🟩[60] 🟦[60] 🟥[65]	🟩[62] 🟥[67]	🟥[55] 🟥[56]	🟥[57] 🟥[66] 🟥[114]		🟥[58]
Sec. 8						🟥[76]	🟥[70]		🟦[72]		🟦[77]			🟩[71]	🟦[73]				🟩[74] 🟩[75]
Sec. 9	🟥[78]	🟥[88]				🟥[79]	🟥[85]	🟨[81] 🟥[89]	🟥[86] 🟦[91] 🟨[91]	🟦[82] 🟦[83]	🟦[90]			🟥[80] 🟥[84]		🟥[87] 🟩[92]		🟥[93]	🟥[94]
Sec. 10							🟦[95]				🟦[96]		🟦[104] 🟦[106]		🟦[97] 🟥[98] 🟥[99] 🟥[105]		🟥[100] 🟦[102]	🟩[101] 🟦[103]	
Sec. 11							🟥[107]								🟩[108]				🟩[109]

elaborated in Section 4.2. Finally, EAs/SAs-based optimization of other aspects of neural networks is discussed in Section 4.3.

4.1 Evolving DNNs for image classification

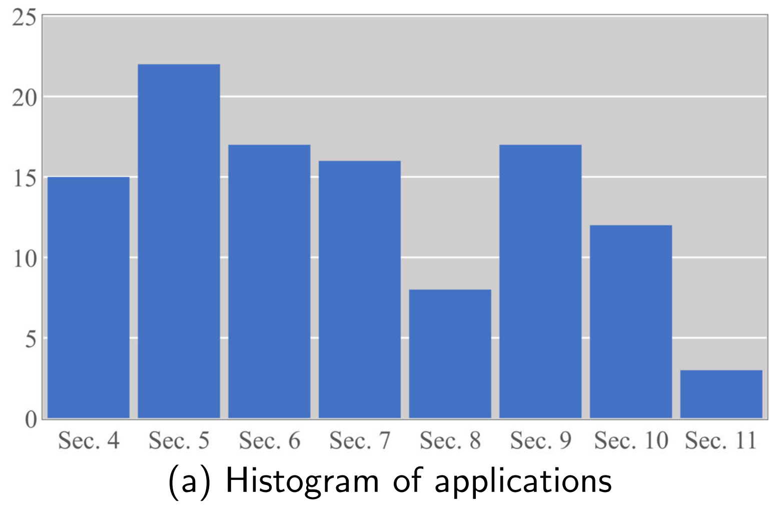
In recent years, image classification has become one of the most investigated tasks in computer vision, which has been brought about by the development of DNNs, particularly convolutional neural networks (CNNs). A typical CNN consists of multiple building blocks, and the order of placement affects the performance. This characteristic makes it difficult to adopt certain NAS methods that have been successfully applied to DNNs, such as a random search and Bayesian optimization [2]. In this subsection, we place emphasis on studies that aim to automatically design optimum DNNs for large-scale image classification benchmarks using the GA and PSO. The recent active development of NeuroEvolution for large-scale image classification began in 2017.

- LEIC

Real et al. [1] employ a GA at unprecedented scales to discover models for large-scale image classification benchmarks by using large computational resources (e.g., running on 250 GPUs for approximately 10 days), the result of which have demonstrated that NeuroEvolution can achieve a competitive performance as a handcrafted model. In their study, they developed CNN structures/models, where every individual (i.e., model) is evolved from scratch and encoded as a graph. Through the evolution process, different types of layers can be incorporated into the individuals through specific mutations (e.g., an insert convolution, remove convolution, or alter stride). A weight evolution is also considered in this study, they used pra backpropagation to allow the trained weights to be inherited by the children whenever possible. Specifically, if a layer has matching shapes, the weights are preserved. In addition, a binary tournament selection [122] is used to perform pairwise comparisons of random individuals, and the worst pair is immediately removed from the population. This study is important because it shows that NeuroEvolution can be used for large-scale image classification with a simple algorithm. However, such success requires an enormous amount of computational resources, which has been one of the challenges for later studies.

- EvoCNN

Sun et al. [2] proposed EvoCNN, a GA-based approach to automatically evolving the architecture and initial weights of a CNN. Because the optimal depth of a CNN is unknown, a variable-length gene encoding strategy is employed in EvoCNN. EvoCNN is composed of three different building blocks, a convolutional layer, a pooling layer, and a full connection layer, which are encoded in parallel into one chromosome for evolution. Therefore, each chromosome is separated into two parts. The first part includes a convolutional layer and a pooling layer, and



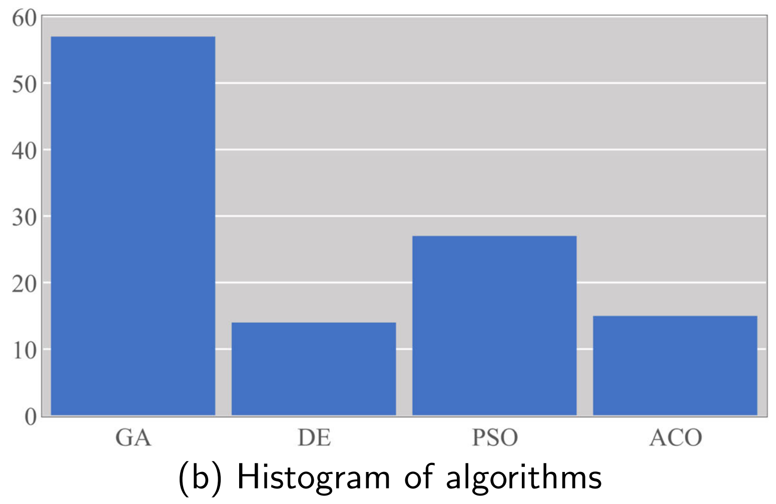


Fig. 4 Statistics of the literature in terms of applications and algorithms. The vertical axis indicates the number of related papers

the other part contains a full connection layer based on the convention of a CNN. Two statistical real numbers, the standard deviation and the mean value of the connection weights, are used to represent the numerous weight parameters, which eases the implementations of the GAs. When the optimum mean value and standard deviation are achieved, the weight values are then sampled from the corresponding Gaussian distribution. A slack version of a binary tournament selection is used to select the parent solutions for the crossover operations. The generated offspring conduct mutation by addition, deletion, and modification, with respect to the parent solution. In the fitness evaluation process, every individual is trained by a small number of epochs to speed up the training. Based on their structure and initialized weights, the mean value and standard derivation of the classification error are calculated on the validation set for the fitness of every individual. These elements indicate the performance tendency, which is sufficient information for evaluation. As a result, this process dramatically speeds up the evaluation by avoiding a thorough training as conducted in [1].

- CoDeepNEAT

CoDeepNEAT [3] is an extension of the NEAT algorithm, which is dedicated to the evolving network structure and hyperparameters of a DNN. The key idea of CoDeepNEAT is the coevolution of the modules and blueprints. The blueprint chromosome is a graph where each node contains a pointer to a particular module species, and each module chromosome is a graph that represents a small DNN. During a fitness evaluation, the modules and blueprints are combined to create a larger assembled network, which is further decoded into a phenotype (DNN) and then trained for a fixed number of epochs. This coevolution strategy allows efficiently acquiring an iterative modular structure, which is a common feature in many successful DNNs. Each node (layer) in the module chromosome contains a table of real and binary valued hyperparameters that are mutated through a uniform Gaussian distribution and random bit-flipping, respectively. Over the generations, a structure (i.e., a layer) is added to the graph incrementally through a mutation. To ensure that the parent layer's output is the same size as the current layer's input, the adjustment process is conducted through a concatenation or element-wise sum operation.

- CGP-CNN

In the study by Suganuma et al. [4], Cartesian genetic programming (CGP) is used in the evolution of a CNN architecture and connectivity, where the hyperparameters and connections of each layer along with the total number of layers are optimized. The architecture of a CNN is represented as a directed acyclic graph with a two-dimensional grid. The genotype consists of integers with a fixed length, and each gene has information regarding the type and connections of the node. Referring to the modern CNN architectures, highly functional modules such as ConvBlock, ResBlock (consisting of convolution processing, batch normalization, ReLU, and a summation), and pooling are selected as node functions. The $1 + \lambda$ evolutionary strategy is employed to conduct a search within the architecture solution space, which means that λ children are generated from a single parent at each generation by applying a mutation, and the best performing child compared to the parent is updated as the new parent for the next generation. The node type and connections of each node are randomly changed according to the mutation rate.

- Genetic CNN

Xie and Yuille [5] encoded each network structure into a fixed-length binary string and applied the GA to automatically learn the structure of a deep CNN. The search space is restricted by imposing constraints on the network structures such that a network is composed of a limited number of stages, and each stage is defined as a set of predefined building blocks (convolution and pooling). The Russian roulette process [123] is used for the selection. In each generation, the standard genetic operations, for example, a crossover and mutation, are conducted to generate competitive individuals.

- HREAS

Liu et al. [6] proposed a GA-based structure search method using multi-level hierarchical representations of DNNs, allowing flexible network structures (directed acyclic graphs) at each level of the hierarchy. The key idea of a hierarchical representation is to have several graphs (or motifs) at different levels of the hierarchy, and the lower-level graphs (such as a graph of primitive operations, e.g., convolution, pooling, etc.) are used as building blocks during the construction of the higher-level graphs. During the generation, a hierarchical genotype has mutated a sequence of actions that include selecting the hierarchy level, selecting the target graph at the target level, and modifying the target graph using add, alter, and remove operations. Similar to the approach by Real et al. [1], the evolutionary search algorithm is based on a queue-based tournament selection, which is implemented in an asynchronous distributed manner, consisting of a single controller responsible for performing mutations over the genotype and a set of workers responsible for their evaluations.

- DENSER

Assunccao et al. [7, 8] proposed a two-level representation. The outer level, i.e., GA-level, encodes the general structure of the network and is responsible for representing the sequence of layers. The inner level, i.e., dynamic structured grammatical evolution (DSGE), encodes the parameters associated with the layers. Because there is a one-to-one mapping between the layers and their parameters, the evolution of the networks keeps the genetic material of each layer together. This makes the manipulation of the solution easier. Two crossover operations are developed, acting on both levels of the genotype. A one-point crossover is used to exchange the layers within the same module. A module is a set of layers that belongs to the same GA structure index, such as the features (convolution or pooling) and classification (fully connected). A bit-mask crossover is used to exchange modules between two parents. In a mutation, they used two sets of mutation operations that act at the GA and DSGE levels, respectively. For example, the addition, replication, and removal are at the GA level, and the grammatical mutation and integer/float mutation are at the DSGE level.

In addition, Kramer [9] utilized a $(1 + 1)$-EA for optimization of the structure and hyperparameters of convolutional highway networks, which are methods for constructing networks with a large number (hundreds and even thousands) of layers. The convolutional highway network is represented as a bit string.

Several studies have adopted a PSO, taking advantage of its easy implementation and lower computational cost.

- PSOAO

The authors of EvoCNN [2] proposed a flexible convolutional auto-encoder (CAE). This flexible CAE aims to overcome the constraints of the classical CAE, which has only one convolutional layer and one pooling layer in the encoder. Its architecture optimization is achieved by a PSO consisting of variable-length particles, called PSOAO. A variable-length encoding strategy is applied to the PSOAO algorithm, where each particle contains different numbers of layers with different parameters (such as the filter width/height, stride width/height, convolutional type, number of feature maps, and pooling type). The main flow of the PSOAO algorithm follows the simple PSO algorithm. One challenge resulting from the adoption of variable-length particles is the need to calculate the *gbest*. To this end, the padding and truncation operations are used to keep the length of the layers unchanged in the global best and the reference (current) particle. In addition, the reconstruction error is taken as the fitness.

- IPPSO

In a study by Wang et al. [11], the PSO is utilized to search the optimal architectures of a CNN for image classification tasks. In their approach, a new encoding scheme is proposed, which defines a "network interface" containing the IP address and its corresponding subnet to carry the configurations of a CNN layer. The network IP address can be divided into numerous subsets, each of which can be used to define a specific type of CNN layer (convolution, pooling, or fully connected). This means that a high-dimensional particle vector (i.e., the entire IP address) can be divided into several parts (i.e., CNN layers), which facilitates the convergence of the PSO. To attain variable-length CNN architectures, a new layer called a disabled layer is defined to disable some of the layers in the fixed-length IP address encoding.

As a quantitative summary, the classification performances of the discovered models on large-scale image classification benchmarks such as MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, and ImageNet are listed in Table 5.

Table 5 Classification accuracy of discovered models by evolutionary approaches on different datasets

Dataset	Approach	Accuracy	OPT. time	#GPUs
CIFAR-10	LEIC [1]	94.60	10 days	250
	CoDeepNEAT [3]	92.70	–	–
	CGP-CNN [4]	94.34	10.4 days	2
	Genetic CNN [5]	92.90	2 days	10
	HREAS [6]	96.40	1.5 days	200
	DENSER [7, 8]	93.29	–	–
	PSOAO [10]	83.5	3.4 days	1
CIFAR-100	LEIC [1]	77.00	–	–
	Genetic CNN [5]	70.97	–	–
	DENSER [7, 8]	77.51	–	–
MNIST	EvoCNN [2]	98.82	2-3 days	2
	DENSER [7, 8]	99.70	–	–
	PSOAO [10]	99.51	5 days	1
	IPPSO [11]	98.87	–	–
Fashion MNIST	EvoCNN [2]	94.53	4 days	2
	DENSER [7, 8]	95.11	–	–
ImageNet	Genetic CNN [5]	72.13	–	–
	HREAS [6]	79.70	–	–

4.2 Evolving DNNs for image restoration

Image restoration, which recovers a given corrupted image to the original clean image, is an important task of computer vision along with image classification. There are several studies that have addressed this task for networks designed using NeuroEvolution.

- DPPN

Fernando et al. [12] proposed a differentiable pattern producing network (DPPN), which combines the evolution of a network structure and learned weights using a Lamarckian approach for an auto-encoder neural network. With DPPN, every individual is encoded using a connection matrix and a node list. During each generation, the auto-encoder is trained through a gradient descent approach, and the learned weights are inherited by the offspring. Two evolutionary algorithms (a microbial genetic algorithm (mGA) and an asynchronous binary tournament selection) are used to select the parent solutions, where two random individuals and random pairs (whenever more than two workers are working simultaneously) are chosen, respectively. The chosen individuals are then trained and their fitness is evaluated. The mutated copy of the winner overwrites the loser. Three types of mutations are applied to generate the network structure: the addition of a random node, the removal of a random edge, and the addition of a random edge. During a crossover operation, hidden units (nodes) of both parents are combined. The mean squared error is used as a fitness function.

- E-CAE

Suganuma et al. [13] introduced an evolutionary algorithm that searches the optimum architecture of the CAEs for an image restoration. The CAEs in this study are built using only standard ConvNet building blocks (i.e., convolutional layers with an optional downsampling and skip connections) that involve symmetric encoder-decoder structures. Nevertheless, the results show that the CAEs generated by an EA can achieve a competitive performance compared to hand-crafted models for image inpainting and denoising tasks. The representation and evolutionary strategy for the CAEs are the same as those described by Suganuma et al. [4]. At each generation, λ children are generated by applying mutations to the parent and are trained to minimize a standard l_2 loss. The fitness of every individual is measured using the peak signal-to-noise ratio (PSNR) between the restored and ground truth images on the validation set. The genotype is updated to maximize the fitness as the generation proceeds.

4.3 Evolving DNNs for other tasks

In [14] and [15], the GA is employed to evolve the weights of a fixed CNN and pass the local optimum, moving toward the global optimal during the training. A method presented in [15] shows that this can improve the performance of a pure CNN. To find the best weights for a CNN, the authors used a crossover operation exchanging the layer weights and threshold values between two chromosomes and a mutation operation changing the layer weights and threshold values. In [14], a standard GA is employed to train the weights of a CNN for crack detection on the image. However, the authors reported that the results are no better than when training the CNN through a backpropagation.

In short, Table 6 summarizes the information from the literature reviewed in this section.

5 Image segmentation

Image segmentation aims at partitioning a digital image into multiple segments according to the information extracted from the pixels. Many computer vision approaches employ segmentation for a pre-processing to easily understand the parts that construct the image. Informative segmentation such as semantic segmentation and instance segmentation is now active in the field of image segmentation, which is typically powered by high-level deep features with DNNs. On the other hand, most existing segmentation works using EAs and SAs focus on only classical tasks. That is, segmentation is achieved by dividing pixels based on low-level intensity information. The difficulty of an accurate segmentation typically increases as the number of segments increases. In addition, a determination of the optimal number of segments is also a challenging task.

In this section, we mainly describe the typical thresholding (in Section 5.1) and clustering (in Section 5.2) approaches used in image partitioning. Other approaches, such as contour-based methods, are described in Section 5.3.

Table 6 Brief information of the literature summarized in Section 4

Algorithm	Author	Section
GA	Real et al. [1]	Section 4.1
	Sun et al. [2]	Section 4.1
	Miikkulainen et al. [3]	Section 4.1
	Suganuma et al. [4]	Section 4.1
	Xie and Yuille [5]	Section 4.1
	Liu et al. [6]	Section 4.1
	Assunccao et al. [7, 8]	Section 4.1
	Kramer [9]	Section 4.1
	Fernando et al. [12]	Section 4.2
	Suganuma et al. [13]	Section 4.2
	Zhining and Yunming [15]	Section 4.3
	Ouellette et al. [14]	Section 4.3
PSO	Sun et al. [10]	Section 4.1
	Wang et al. [11]	Section 4.1

5.1 Thresholding approaches

Thresholding is a simple and popular technique used in image segmentation. This approach is typically divides a histogram of the pixel intensities. As a simple example, a demonstration of two-level segmentation is shown in Fig. 5. The pixel intensity, regarded as boundary, is determined according to the distribution of the histogram. There are two representative thresholding methods: a fuzzy partition and the Otsu method.

- Fuzzy partition

The fizzy partition is the probabilistic representation of the likelihood that each pixel intensity belongs to an class. The probability of belonging to each class is defined by the membership function, and the threshold between the two classes is set at the intersection of the membership functions, as shown in Fig. 6.

EAs and SAs are exploited in tuning the parameters of the membership functions. Tao et al. [16] optimized six integer parameters using the GA to segment a gray-level image into three clusters. Each parameter is encoded as a simple 8-bit string. These parameters are tuned such that the fuzzy entropy [124] is maximized. Later, Tao et al. [17] proposed a fuzzy entropy maximization method using ACO and applied it to the two-level segmentation of infrared images. The initial positions of the ants are randomly chosen from all possible solutions. The ants then search for more attractive solutions from the neighborhood according to their transition probability. Puranik et al. [18] presented a modified PSO to select the rules of the fuzzy logic for color the image segmentation. Each color class is described by several fuzzy sets in the HSL color space, specifically, ten sets for hue, five sets for saturation, and four sets for lightness. The task of PSO is to produce a smaller number of fuzzy roles while preserving a low error rate. In the velocity update, each dimension of the particle can be updated by learning the *pbest* of other particles,

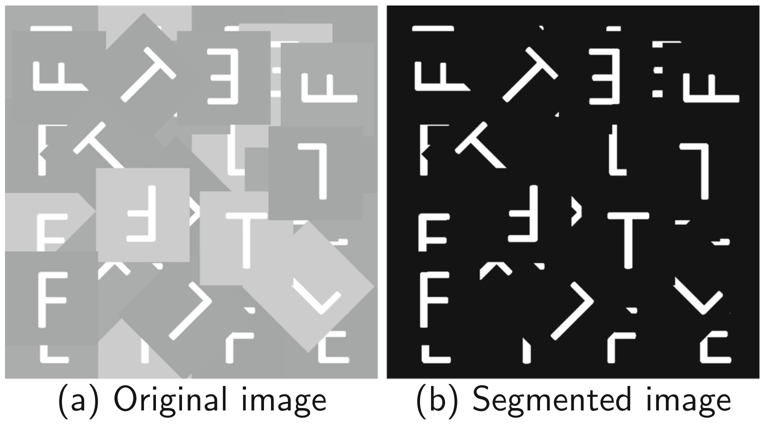


Fig. 5 Demonstration of a simple two-level segmentation. Pixels in the original image **a** is segmented into two levels with a intensity threshold of 200. As a consequence, alphabets are extracted in **b**

including particles in different generations. The algorithm is thus called comprehensive learning PSO (CLPSO).

- Otsu Method

The Otsu method selects the feasible thresholds only from gray-level histograms without any prior knowledge. It considers a threshold that maximizes the variance between classes to be reasonable. However, an exhaustive search conducted using to find the optimal threshold is a time-consuming procedure, and an extension to multi-level thresholds requires additional computational costs.

EAs and SAs have attracted attention as feasible search methods. Liang et al. [19] utilized a simple ACO in combination with Otsu thresholding (ACO-Otsu) for image segmentation. This is much faster in terms of 2∼4-level segmentation than the Otsu method with an exhaustive search. Ghamisi et al. [20] introduced an improved PSO, called fractional-order Darwinian PSO (FODPSO), to tackle hyperspectral image segmentation. As the two main changes from a traditional PSO, several swarms of traditional PSOs are treated in parallel to enhance the ability to escape from the local optima, and a fractional calculus controlling the convergence rate is added. Particles are encoded with the thresholds and the final optimal thresholds are combined with the results from other methods through a voting procedure. As with the Otsu method, the variance between classes is used for the adaptive degree function.

One of the challenging tasks in multi-level thresholding is to determine the number of thresholds. An automatic provisioning of the optimal number of thresholds can be applied to more practical situations. To improve the ACO-Otsu approach [19], Liang et al. [21] proposed an ant colony system (ACS) using the Otsu method (ACS-Otsu), introducing a hierarchical search range and uniformity measure to automatically determine the search

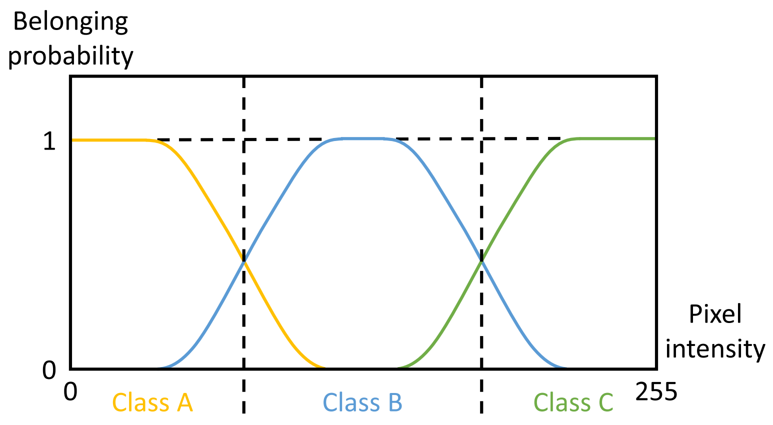


Fig. 6 Illustration of fuzzy partition in the case of three-level segmentation. The probability of each pixel intensity belonging to a class is defined by the corresponding membership functions (colored curves). Each threshold between two classes (vertical dotted line) is set at the intersection of two membership functions

ranges and number of thresholds, respectively. ACS-Otsu is also combined with a local search process for the best ant if the informative heuristics cannot be defined. The method is then combined with an expectation-maximization method [22], which initializes the ACS-Otsu method and obtains refined parameters from the approach. With the numbers of thresholds and positions determined through iterative Otsu thresholding, Chander et al. [23] introduced a "momentum" and "social" PSO to refine the thresholds. A particle encoded with such thresholds is placed in the "momentum" weight and "social" weight categorizes in the velocity update, which are variables depending on the fitness of the particle. The "momentum" weight emphasizes the influence of the previous iteration, whereas the "social" weight stresses the relationship with *gbest*. The two weights can favor each particle in moving toward the global optima.

5.2 Clustering approaches

Clustering is also an important technique in image segmentation. Thresholding-based segmentation determines the boundaries between classes, whereas clustering-based segmentation deals with the centroids of classes. The positions of the cluster centroids are adjusted by minimizing the distance defined between the pixels and centroids based on certain features. Omran et al. [24] proposed optimizing the cluster centroids with a fixed number of clusters to segment the image using the PSO. The evaluation of the particles is applied according to three principles: (1) minimizing the intra-distance between pixels and their cluster means, (2) maximizing the inter-distance between any pair of clusters, and (3) minimizing the quantization error. The fitness function is the sum of the weighted objective functions, which requires no effort to address the multi-objective problem for PSO. In addition, the pheromone matrix of ACO is useful for an image segmentation. Instead of segmenting an image using image primitives such as the intensity and color, Malisia et al. [25] proposed clustering the pheromone matrix of ACO into two clusters using a k-means approach. The ants move to the neighboring pixels and drop their pheromones there. After the ACO iteration is completed, the normalized pheromone matrix is combined with the original normalized gray-level image. K-means clustering classifies the values of the combined dataset as black or white.

Determining the optimal number of clusters is an important task in a clustering-based approach. Numerous studies have aimed at developing methods that automatically provide the optimal number of clusters. Maulik et al. [26] proposed a pixel classification method using a variable string length genetic algorithm (VGA), where each chromosome consists of a cluster of centroids encoded by real numbers, and the number of clusters (i.e., the length of the chromosome) is variable. The crossover guarantees that there are more than two clusters owing to the constraints of the range of crossover points. Omran et al. [27] proposed dynamic clustering using PSO (DCPSO). With this method, the position of each particle is a binary representation, and a value of 1 in the binary code means that the corresponding element in the cluster centroids pool is chosen. Then, the best set of centroids is refined using the k-means approach. The process is repeated a user-defined number of times with an updated centroid pool, which is the union of previous results and randomly chosen centroids. Awad et al. [28] proposed a hybrid GA (HGA), which includes a hill-climbing algorithm in the update to quickly find the local optima, for satellite image segmentation. The chromosomes are encoded with the features of self-organizing maps of the full image, avoiding the determination of the number of clusters and allowing the evolution to be the final result. After that, Awad et al. [29] presented a hybrid dynamic GA (HDGA), having the advantages of both HGA [28] and VGA [26], in solving the segmentation problem. On the one hand, HDGA employs the hill-climbing algorithm in the update, demonstrating the "hybrid" aspect of the approach. On the other hand, a chromosome, encoded with a cluster centroid and its pixel value, is set to a fixed length with an ending mark to confirm the actual flexibility of the chromosome, i.e., illustrating the "dynamic" aspect. A crossover occurs only at the cluster centroid bits, rather than at the pixel value bits or the bits after an ending mark. Bansal et al. [30] proposed an approach that focuses on the pheromone matrix, as in [25]. Each ant marks (updates the pheromone) and combines similar traveled pixels until all pixels have been marked. They amend the image to a fully connected graph, i.e., all pixels are connected to each other such that the ants travel toward unmarked pixels during every step. The number of clusters is automatically calculated based on the CMC distance, which is applied as a similarity measure. Halder et al. [31] proposed a GA-based clustering method for gray-level images. They first apply fuzzy c-means (FCM) and encode its result as an individual. This process is repeated until the population pool is filled, and a simple GA framework is then applied. To investigate the appropriate number of clusters, the GA is run multiple times, increasing the clusters to a predefined number. The results for each number of clusters are then evaluated using the validity index. The FCM-GA framework was applied to tumor detection in the brain [32].

Among the different methods available, there are major differences regarding whether the process of finding the optimal number of clusters is built into the EAs and SAs. In [26, 27, 29], EAs and SAs optimize the number of clusters and their centroids simultaneously. Subjected to this setting, the methods can be further categorized according to whether the length of the candidate solutions is fixed. Although a variable-length representation (e.g., [26]) is more natural, fixed-length representations (e.g., [27, 29]) have an advantage in that the traditional operations can be directly embedded. However, in [28, 30, 31], the EAs and SAs are not involved in the optimization of the number of clusters. [28, 30] applied other methods, and [31] adopted a simple approach in which the results of all situations are compared.

5.3 Other approaches

Ouadfel et al. [33] proposed a Markov random field (MRF)-based image segmentation using ACO. The ants trace over a solution space with pixel and label pairs as components and attempt to construct a solution that minimizes the posterior energy function. The search process adopts ACS, which is one of the implementations of ACO that incorporates two-step pheromone updating (local and global).

Pignalberi et al. [34] applied the GA to existing methods for parameter tuning. One hindrance to the adoption of the GA is the fact that some of the parameters are real numbers. Thus, they adopted an extended logical binary coding which uses the symbol set as {0, 1, dot}, allowing real numbers to be represented by a fixed precision. The fitness function is defined as a weighted sum of four components, which represents pixel- and cluster-level errors.

The studies described below are similar in that they accurately extract the contours of the objects. Jiang et al. [35] proposed a cell image segmentation using a parallel GA. The GA adjusts the parameters of the cell boundary model, which is designed based on prior knowledge about the cell shape. The parallel GA divides the population into multiple sub-populations, which self-evolves in parallel. It also includes an elite migration between the sub-populations randomly. Therefore, the diversity is preserved. Feng and Wang [36] derived a method for searching a space using ACO, given the active contour model. To reduce the computational cost of the pheromone updates, a finite grade ACO (FGACO) is proposed, which classifies the pheromones into finite grades. Pheromone updates are realized by changing the grades, which only requires addition and subtraction operations and allows independence from the objective function value. Ma et al. [37] proposed a texture segmentation and representation scheme based on ACO. They first proposed an ACO-based image processing framework and applied it to an image segmentation and texture representation. The difference between both methods can be seen in the design of the direction probability vector and the difficulty of movement, which affect the transition probability and pheromone update, respectively. With the ACO image segmentation algorithm (ACO-ISA), the direction probability vector considers two additional similarity factors, the gray-level between cells and the texture between sub-images. The difficulty of movement is designed to reduce the pheromone intensity at the edge cells. By contrast, an ACO-based texture representation algorithm (ACO-TRA) requires the ants to become sensitive to local changes in the gray-levels. For the direction probability vector, two elements added into ACO-ISA are changed to emphasize the difference in gray-level. The difficulty of movement is designed to increase the pheromone intensity at the edge cells according to changes in texture.

In summary, Table 7 shows a brief outline of the studies described in this section.

6 Feature detection and selection

Analyzing the content of the image for detecting an object or region of interest is highly dependent on the features, which provide rich information of the image. Extracting features from images is fundamental in many computer vision applications, e.g., recognition, detection, matching, and reconstruction. Detecting and selecting high-quality features are challenging tasks owing to the large search space. A variety of methods have been applied to solve the feature detection (Section 6.1) and selection (Section 6.2) problems, among which the EA and SA techniques have

Table 7 Brief information on the literature referenced in Section 5

Algorithm	Author	Section
GA	Tao et al. [16]	Section 5.1
	Maulik and Bandyopadhyay [26]	Section 5.2
	Awad et al. [28]	Section 5.2
	Awad et al. [29]	Section 5.2
	Halder et al. [31]	Section 5.2
	Halder et al. [32]	Section 5.2
	Pignalberi et al. [34]	Section 5.3
	Jiang et al. [35]	Section 5.3
PSO	Puranik et al. [18]	Section 5.1
	Ghamisi et al. [20]	Section 5.1
	Chander et al. [23]	Section 5.1
	Omran et al. [24]	Section 5.2
	Omran et al. [27]	Section 5.2
ACO	Tao et al. [17]	Section 5.1
	Liang et al. [19]	Section 5.1
	Liang and Yin [21]	Section 5.1
	Liang and Yin [22]	Section 5.1
	Malisia and Tizhoosh [25]	Section 5.2
	Bansal and Aggarwal [30]	Section 5.2
	Ouadfel and Batouche [33]	Section 5.3
	Wang et al. [36]	Section 5.3
	Ma et al. [37]	Section 5.3

received significant attention and achieved a remarkable success.

6.1 Feature detection

Feature detection aims to find or locate features (e.g., edges, shapes, and interest points). One of the main contributions of EAs and SAs is to reduce the computation time through a parallel and efficient search. Conventional methods typically involve high computational processing, such as linear filtering operations of a Canny edge detector for edge detection and applying a histogram to the transform space of a Hough transform for circle detection. By contrast, several studies have aimed at improving the interest point descriptors. The operators synthesized by EAs and SAs have shown desirable properties.

Several ant-based algorithms have been proposed to solve the problem of edge detection. The method proposed by Nezamabadi-pour et al. [38] is one of the earliest approaches employing an ant algorithm to detect edges by formulating the image as a directed graph. In [39], Baterina and Oppus introduced the concept of applying a pheromone matrix that reflects the edge information at each pixel based on the routes formed by the ants. The movement of the ants is guided by the local variation of the pixel intensity values.

For a shape detection, Cuevas et al. [40] introduced a circle detection method based on the DE algorithm. This approach uses the encoding of three edge points to represent a candidate circle on the edge image of a scene. Guided by the value of an objective function for evaluating whether a candidate is presented within the edge image, the set of candidates is evolved using the discrete DE (DDE) algorithm. Dong et al. [41] introduced a combined evolutionary search method for circle detection, called chaotic hybrid algorithm (CHA). The authors combined the strengths of both PSO and the GA by including the standard velocity and position update rules of PSO with the ideas of selection, crossover, and mutation from the GA. Specifically, in each generation, after the fitness values of the individuals are calculated, the proportion of the bottom individuals undergoes breeding (selection, crossover, and mutation). The velocities of all individuals are updated and new information is acquired from the population for updating the position. During the mutation process, the chosen individual is reinitialized through the chaos initialization method.

Interest point detection can also be formulated as an optimization problem, and Trujillo and Olague [42] solved this problem using GP. In their study, GP was used to synthesize low-level image operators that detect interest points on digital images. In a newer version [43], the authors improved the performance of previously proposed detectors by considering the operator's geometric stability (by presenting 15 new operators) and the global separability of the detected points. Following the same philosophy, Perez and Olague presented several methods [44, 45] in which GP is used as a strategy to evolve the image descriptors for object detection. For example, in [44], the authors used GP to synthesize mathematical formulas to improve the scale-invariant feature transform (SIFT) image descriptor. They further extended their study in [45] by presenting an optimization-based approach using GP and a hill-climbing algorithm, which creates composite image operators for improving the SIFT descriptor.

6.2 Feature selection

Feature selection [125] is an important task in machine learning and computer vision to reduce the dimensionality of the data by removing irrelevant and redundant features. In the computer vision community, feature selection targets constructing/choosing important visual content (features, e.g., pixel, edges, color, texture, shape, and other problem-specific items) for the interpretation of the image content. Based on its importance, the problem of feature selection has been extensively investigated by researchers from both the machine learning and computer vision communities. To the best of our knowledge, almost all major EC paradigms have been applied to feature selection in the field of computer vision. Studies related to the GA and DE algorithm from EAs, and PSO and ACO from SAs, are mainly discussed in this section.

The GA is the earliest EC technique applied widely to feature selection problems. In [46], the GA with a binary representation is employed for feature selection to enhance the performance of hyperspectral data classification. The experiment results show that the number of features obtained can be decreased over the generations. Treptow and Zell [47] showed that the GA can be used within the Adaboost framework to find features, resulting in better classifiers for object detection such as faces and soccer balls. The chromosome encodes the parameters of the features using a string of up to 13 integer variables. The results demonstrate that, instead of an exhaustive search over all features, an evolutionary search can speed up the training and effectively find good features in a large feature pool within a reasonable time.

The DE algorithm was introduced to solve the feature selection problems in 2008, when Khushaba et al. [48] proposed a method called DE-based feature subset selection (DEFS) to utilize the DE optimization method for the feature selection problem. The improved version is [49]. A new feature distribution factor is introduced to aid in the replacement of the duplicated features by utilizing a roulette wheel weighting scheme. Experiments show that the proposed DEFS algorithm outperforms GA/PSO-based algorithms and other traditional feature selection algorithms on brain-computer-interface tasks. Gosh et al. [50] applied a self-adaptive DE algorithm for feature selection in a hyperspectral image. In their study, the self-adaptive DE algorithm outperforms the GA [46], ACO, DE algorithm, and combination of ACO and DE-based methods in terms of the classification accuracy and Kappa coefficient.

Ghamisi et al. [51] exploited FODPSO to solve the feature selection problems for hyperspectral data. Each particle uses a binary representation for the selection problem. The authors used the overall accuracy of a support vector machine (SVM) classifier on the validation set as the fitness function to evaluate the goodness of the selected features. Because SVM is capable of handling the curse of dimensionality, the proposed approach is capable of handling extremely high dimensional data even with a limited number of training samples. In the following year, Ghamisi et al. [52] proposed a PSO-based CNN method for the classification of hyperspectral data. To tackle the imbalance problem between the high spectral dimensionality and the limited number of training samples available for a CNN, a FODPSO-based feature selection method is employed to find the most informative bands from the hyperspectral data.

Al-Ani [53] applied the ACO algorithm for feature selection and claimed that it can perform better than the GA in the texture classification scenario. The algorithm utilizes both the local importance of the features and the overall performance of the feature subsets to search the feature space for optimal solutions. Chen et al. [54] proposed an efficient ACO-based feature selection algorithm for image classification by introducing a new representation scheme to reduce the size of the search space (i.e., a directed graph). Each node/feature is linked by two distinct edges showing whether a node/feature is selected. This representation scheme significantly reduces the total number of edges that the artificial ants need to traverse.

In summary, Table 8 shows a brief overview of the studies discussed in this section.

Table 8 Brief information on the literature summarized in Section 6

Algorithm	Author	Section
GA	Dong et al. [41]	Section 6.1
	Trujillo and Olague [42]	Section 6.1
	Trujillo and Olague [43]	Section 6.1
	Perez and Olague [44]	Section 6.1
	Perez and Olague [45]	Section 6.1
	Yu et al. [46]	Section 6.2
	Treptow and Zell [47]	Section 6.2
DE	Cuevas et al. [40]	Section 6.1
	Khushaba et al. [48]	Section 6.2
	Khushaba et al. [49]	Section 6.2
	Gosh et al. [50]	Section 6.2
PSO	Dong et al. [41]	Section 6.1
	Ghamisi et al. [51]	Section 6.2
	Ghamisi et al. [52]	Section 6.2
ACO	Nezamabadi-pour et al. [38]	Section 6.1
	Baterina and Oppus [39]	Section 6.1
	Al-Ani [53]	Section 6.2
	Chen et al. [54]	Section 6.2

7 Image matching

The purpose of image matching is to superimpose the common parts of multiple images. Matching is typically conducted by transforming the reference image into a coordinate system of the target image. Therefore, image matching is essentially an optimization problem used to find the transformation parameters that maximize the similarity. Template matching and image registration, which are typical applications of image matching, are described in Sections 7.1 and 7.2, respectively. In addition, this section deals with the jigsaw-puzzle-like problems of aligning the given parts to restore the original image (described in Section 7.3). In addition, methods for matching the features extracted from an image are described in Section 7.4.

7.1 Template matching

The purpose of template matching is to find the region that is most comparable to a template in the target image. An illustration of template matching is shown in Fig. 7. There are two main categories of methods used to search the target image: feature- and pixel-based approaches. In the former case, the transformation matrix between the template and target image is estimated from the feature descriptor such as SIFT. However, occasionally situations occur in which it is difficult to detect the key points, i.e., blurry and texture-less images [56]. The latter-category of methods such as SAD are robust to the above situation, although an efficient method to search the target image is required. In particular, as the degrees of freedom (DoF) of the template transformation increase, an exhaustive search on the target image becomes a more undesirable approach.

EAs and SAs are effective choices to explore in an extensive and complex solution space such as in the above situation. The three studies described below address matching at different DoFs (specifically, 5, 6, and 8, respectively). They commonly incorporate strategies for a further efficient exploration into the GA. Zhang and Akashi [55] proposed a simplified GA for template matching. In this case, the GA is simplified by replacing a crossover and mutation using global and local sampling, where global sampling controls the high-order of the chromosome,

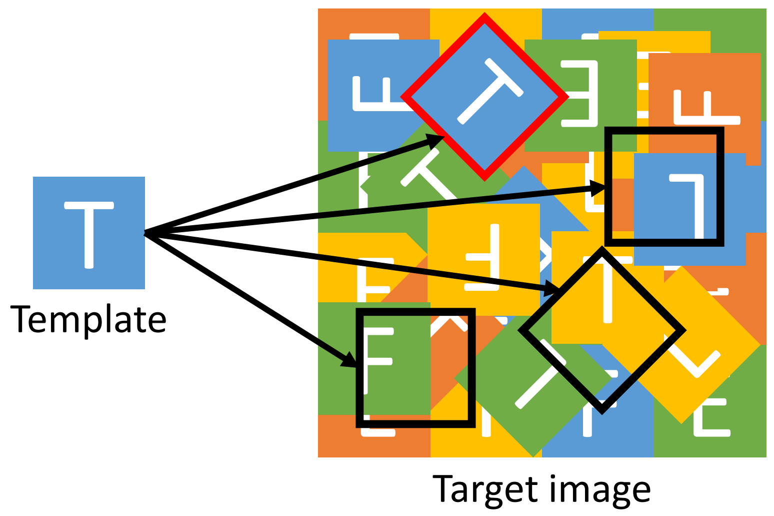


Fig. 7 Illustration of template matching. Similarity between template and candidate regions (black rectangles) are computed over the target image. Candidate region with the highest similarity is the matching result (red rectangle)

and local sampling controls the low-order chromosome. Although the simplified GA is more efficient and accurate in simulated template matching, its operation in real-world cases and in cases with large variations, and in finding the global optimization without a mutation, remain challenges. Zhang and Akashi [56] introduced a level-wise adaptive sampling (LAS) based on the GA to solve affine template matching over a Galois field. With an increase in the number of computations, Galois field can narrow the search range in the target image, and can finally locate the area. To reduce the number of computations, the researchers presented the LAS under the GA framework, which preserves the genetic variety through the selection of individuals from each fitness level uniformly, rules out the inferior individuals using learning thresholds, and simplifies the computational complexity of each individual by inspecting only a small fraction of pixels. The method have turned out to be robust and efficient, but a problem still remains regarding how effective it is in cases with large variations, and no theories exist that prove the method will not converge to the local optimum solutions without a mutation. After that, Zhang and Akashi [57] extended [56] to projective template matching using a binary finite field that can deal with a large DoF. Although the LAS under the GA framework saves considerable computational costs while retaining its accuracy, the algorithm is still far from achieving real-time capabilities for a large DoF. In addition, it may fail when the template image has large variations.

Considering a more practical situation, the approach is useful for matching when there are multiple detection targets on the target image. Sato and Akashi [58] proposed a method for distributing the population in deterministic crowding (DC), which is derived from the GA, to deal with multi-object template matching. The crossover in DC involves the interaction between parents and children (which can be mutated), thereby possibly leading to multi-local optimization, which can be solved using a method that loops the selection of the best-fit individual and a local search. This method has been successful in multi-object template matching with a simple background, the accuracy of which decreases with an increase in the background complexity. In addition, it is not necessarily the case that only materials that look perfectly the same (e.g., those produced in factories) are eligible for matching, and several studies use template models based on prior knowledge of the target. Lee et al. [59] proposed the application of GA-based template matching to lung nodule detection in computed tomography (CT) images. They employ GA-based template matching to detect the approximate location of nodules, and a conventional template matching to detect the nodules accurately. The template image used in lung detection is spherical/circular nodular models. Ugolotti et al. [60] compared PSO with the DE algorithm to solve the object detection problem, and validated the methods in two real-world computer vision problems—hippocampus localization in histological images, and human pose estimation in image sequences. Their method requires that the object follow certain models, which are defined to transform the problem into an optimization problem that can be searched using PSO and the DE algorithm individually. In addition, to accelerate the method, they take advantage of a GPU for parallel computations.

7.2 Image registration

The task of image registration is to convert multiple images into an unified coordinate system allowing the common parts to overlap. One of the most popular applications is an overlapping of multiple images taken from different viewpoints or at different times by remote sensors, as shown in Fig. 8. De Falco et al. [61] transformed a satellite image registration into the problem of optimizing the affine transformation according to the mutual information between images, and optimized the problem using the DE algorithm. Ma et al. [62] proposed an orthogonal learning DE (OLDE), which combines the orthogonal learning (OL) strategy with the DE algorithm, for remote sensing image registration. During the crossover step, multiple candidate vectors are generated from the parent vectors based on the OL strategy, and the vectors with higher fitness are selected as offspring. This incorporation of OL strategies enhances the ability to select promising search directions toward the global optimum. The two methods above were compared in experiments described in [62] using the Ottawa and Yellow River datasets, and the results demonstrated that OLDE outperforms a simple DE method.

In this subsection, we also cover the registration between 2D images and 3D objects. Wachowiak et al. [63] applied a modified PSO to a single-slice 3D-to-3D biomedical image registration. The authors assumed that the users of their proposed system are skilled clinical experts and can be given an accurate initial orientation, which is an important benefit to the complexity of medical image registration. Therefore, they added a term included in the initial orientation to the velocity update, which is expected to prevent a fall into the local optima. This modified velocity update is incorporated into three modified PSO approach (e.g., hybrid PSO with a crossover) selected through preliminary experiments. Liebelt and Schertler [64] addressed the registration of 3D models for use in images. The six parameters of the 3D model are optimized using a simple PSO. In addition, to accelerate the algorithm, the authors treat each inherently parallel optimization in different threads of the GPU. The similarity measure uses mutual information, which is a typical similarity metric in this field and represents the relative entropy of two images, with improved robustness owing to its fusion with edge-based measurements.

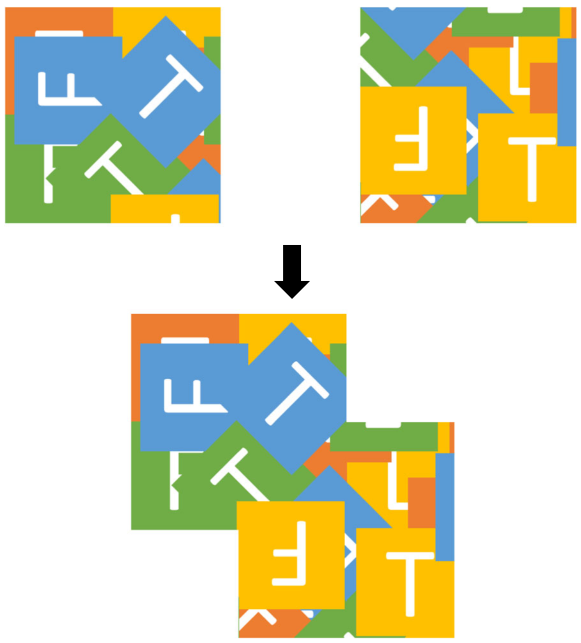


Fig. 8 Illustration of image registration. Two given images are aligned such that the common parts are overlapped

7.3 Jigsaw-puzzle-like problems

Jigsaw puzzles are popular all around the world. The player must reconstruct the original image from the given non-overlapping pieces, as shown in Fig. 9. Automatic jigsaw puzzle solvers with a computational aid can solve puzzles with an extremely large number of pieces, and such a technique can also be applied to a reconstruction, such as archeological artifacts and torn documents. Sholomon et al. [65] proposed a GA-based jigsaw puzzle solver for puzzles of known size and piece orientation. The pairwise compatibility of the adjacent pieces is evaluated based on color similarity along their abutting edges. A chromosome is represented by a matrix of the same size as the puzzle, and each element is assigned a piece number. This simple encoding causes a serious problem: offspring yielded from a traditional crossover may contain duplicate and/or missing pieces. Thus, the authors proposed a novel crossover operator based on a kernel-growing technique, which starts with a single piece and gradually joins other pieces at the available boundaries. The selection and assignment of the pieces to be joined are conducted using a three-phase process from a bank of available pieces, which ensures that every piece appears only once. The development of an applicable crossover operator enables the introduction of the GA into the jigsaw puzzle solver field and brings about significant improvements in the solving power. Specifically, the proposed method achieves an accurate reconstruction of 22,834 pieces, which is more than twice the existing results. After that, Sholomon et al. [66] confirmed the effectiveness of each phase in the crossover, as well as the robustness of the objective function experimentally. They also accelerated the crossover in [65] using multiple threads. In addition to type 1 puzzles (puzzles with pieces whose location is unknown), Sholomon et al. [67] extended the GA-based solver in [65] to solve type 2 puzzles (puzzles with pieces whose location and orientation are unknown) and type 4 puzzles

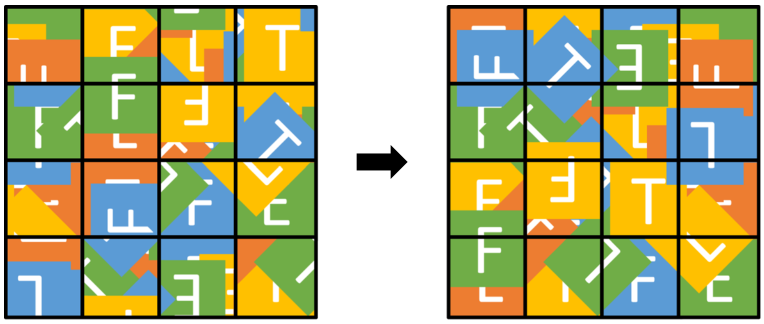


Fig. 9 Illustration of jigsaw puzzle problem. The given non-overlapping pieces are correctly rearranged to construct the original image

(two-sided puzzle with pieces whose location, orientation, and face are unknown). To consider the orientation, the authors adopt a graph representation where each node corresponds to a piece and each edge corresponds to a joint edge of two adjacent pieces. The crossover operator is similar to that in [65], i.e., it is applied based on a kernel-growing technique. In addition, for type 4 puzzles, a constraint is added to maintain the geometrical validity: the flipping side edge of an already jointed edge is not selected. This method outperforms the existing method in type 2 puzzles and was the first to successfully solve type 4 puzzles. The experiments in these studies were conducted in a common format. We summarize the results with the largest number of pieces regarding the neighbor comparison that measures the fraction of correct neighbors, in Table 9.

Wall painting reconstruction is similar to but more complex than jigsaw puzzles, it is not limited by a rectangular shape and can be eroded and lose some of its fragments. Sizikova and Funkhouser [114] proposed solving a wall painting reconstruction using a modified GA, modifying the selection in two steps, including fragment- and binary-based selection, and premodifying the crossover in two categories, a crossover by fragmentation and a crossover by matching. The GA framework starts with one or two fragments, grows to optimize the orientation and translation of the merges, and ends based on a set number of iterations or the completion of all fragments.

Table 9 A brief summary of the experimental results of the jigsaw puzzle solvers. The results for each image are the average of multiple runs using different random seeds, and the average best is the largest score among them

Approach	Type	Number of pieces	Average best
Sholomon et al. [65]	Type 1	22,834	96.28
Sholomon et al. [66]		30,745	93.40
Sholomon et al. [67]	Type 2	22,755	91.07
	Type 4	10,375	99.20

7.4 Feature matching

Graph representations are useful for representing local features in an image along with spatial relationships (i.e., nodes and edges represent local features and relationships, respectively) [126, 127], and hence graph matching, which aims at finding similarity between graphs, is used as one of the techniques for image matching. Myers and Hancock [68] proposed a multimodal GA for graph matching. They avoid the extra computations caused by a non-replacement during the selection through a biased selection without a replacement, thereby reducing the computational cost. Similarly, points are also important features in an image. Zhang et al. [69] presented a GA-based, incomplete (not one-to-one mapping), unlabeled (using no other information, e.g., color) point pattern matching method. They select several sets of triple points in two images, maximizing the partial bidirectional Hausdorff distance between the triple points sets in different images. The GA-based point pattern matching is more efficient than that of traditional optimization methods such as geometric hashing [128].

In summary, Table 10 shows a brief overview of the studies analyzed in this section.

8 Visual tracking

The purpose of visual tracking is to find a target object in each frame within a video sequence. Visual tracking can be regarded as a sequential detection problem when considering the variation in the object state. It is essentially equivalent to a dynamical optimization problem.

Most tracking algorithms can be classified into deterministic or stochastic methods. Deterministic methods, such as a mean shift, are computationally efficient, but suffer from the local optimum. By contrast, stochastic methods, such as condensation and a particle filter, can provide robust tracking, but require high computational costs. In addition, these methods may cause a degradation in the long-term tracking. EAs and SAs are the rational choice to alleviate these weaknesses. For instance, [72] showed that the iteration process of PSO is helpful

Table 10 Brief information on the literature referenced in Section 7

Algorithm	Author	Section
GA	Zhang and Akashi [55]	Section 7.1
	Zhang and Akashi [56]	Section 7.1
	Zhang and Akashi [57]	Section 7.1
	Sato and Akashi [58]	Section 7.1
	Lee et al. [59]	Section 7.1
	Sholomon et al. [65]	Section 7.3
	Sholomon et al. [66]	Section 7.3
	Sholomon et al. [67]	Section 7.3
	Sizikova and Funkhouser [114]	Section 7.3
	Myers and Hancock [68]	Section 7.4
	Zhang et al. [69]	Section 7.4
DE	Ugolotti et al. [60]	Section 7.1
	De Falco et al. [61]	Section 7.2
	Ma et al. [62]	Section 7.2
PSO	Ugolotti et al. [60]	Section 7.1
	Wachowiak et al. [63]	Section 7.2
	Liebelt and Schertler [64]	Section 7.2

for restoring particles sampled from inappropriate transition models to the appropriate (i.e., higher observable likelihood) region.

We categorize the visual tracking problem into single-object tracking and multiple-object tracking in Sections 8.1 and 8.2, respectively.

8.1 Single object tracking

Several studies have addressed block matching where the entire image is divided into non-overlapping blocks and the difference in position over successive frames for each block is computed. The difference in position is called the motion vector, and motion vectors of blocks containing the of interest are useful for tracking. Bhaskar et al. [70] proposed a motion estimation algorithm with variable-size block matching using the GA. Block-based motion estimation is accomplished by finding the same region in the next frame for blocks that represent segmented regions of the image. Variable-size block division is achieved through a quad-tree decomposition, which divides an entire image into four equivalent sized regions recursively. The GA is executed on all blocks, and moves the centroid of the block to match the block in the successive frame. With the genetic operator, only a mutation is executed. Individuals that have lower than the mean fitness are the targets of mutation, and others are taken into the next generation directly. Although experimental results show a better performance than that of other methods, the combination of a recursive division and the GA is a time-consuming process. Cuevas et al. [71] attempted to speed up the processing of fixed-size block matching using the DE algorithm by reducing the number of similarity evaluations. During a DE search, all fitness values are stored in the history array, and most individuals are evaluated through a nearest-neighbor-interpolation based estimation based on the stored fitness of the nearby individuals. This fitness estimation strategy substantially reduces the number of evaluations rather than exhaustive evaluations within a search area while maintaining the accuracy.

In the following tracking methods, the target is represented by a rectangle. The candidates have rectangular parameters, such as location, rotation angle, and scaling, and look for the most similar region based on the appearance model. Because the rectangular parameters are real values, PSO and the DE algorithm are preferred for optimization. Zhang et al. [72] proposed a sequential PSO that incorporates the sequential information into the traditional PSO. The attractive point of the sequential PSO is the introduction of a re-diversification mechanism using previous results and an adaptive parameter tuning. In addition, a spatially constrained Gaussian mixture model (GMM) for the appearance of the tracked object is used to evaluate each particle. From a Bayesian inference perspective, sequential PSO is a combination of multi-layer importance sampling and a particle filter, which can avoid the sample impoverishment problem in a particle filter. Cheng et al. [73] proposed a visual tracking technique to utilize a fragment-based appearance model, which can acquire the robustness of the target occlusion. The target state is divided by rectangle fragments, and the saliency based on the SIFT feature is assigned to each fragment. Particles of the PSO have affine transform parameters and are evaluated using the saliencies and an HSV color histogram. In addition, the initialization of the particles in each frame uses a Gaussian distribution constructed from the previous results to maintain diversity. Lin and Zhu [74] proposed an improved fast DE (IF-DE) algorithm to alleviate the evolution stagnation. The IF-DE algorithm focuses on inferior parents and trial individuals, which are discarded in the previous generation. These individuals are introduced as a difference vector during the mutation stage, which serves to extend the search space. Three scaling parameters during a mutation operation are changed dynamically based on the best individual or diversity information. The evaluation of each individual for the tracking process utilizes the GMM, similar to that described in [72]. Nenavath and Jatoth [75] introduced a hybrid SCA-DE, which is a combination of the sine-cosine algorithm (SCA) and the DE algorithm. The flow of the hybrid SCA-DE is simple: after every individual is updated by the SCA, DE operations including a mutation, crossover, and survivor selection are applied. Whereas the SCA conducts a global exploration with a large step size, the DE algorithm is in charge of the local search to encourage the population to reach the best solution, which enables the hybrid SCA-DE algorithm to balance between global and local searches. The tracking method using the hybrid SCA-DE approach optimizes a state vector consisting of the location, speed, and scaling with a kernel-based spatial color histogram as the observation model.

In the four studies above, the initialization of the candidates in each frame exploits the previous result. To achieve diversity, the authors adopt a Gaussian distribution [72–74] and random walk (RW) model [75]. This commonality is unique to dynamic optimization problems. Most tracking performance results are given as graphs plotting the accuracy per frame. In particular, the comparison results with [72] can be found in [73] and [74], respectively.

8.2 Multiple object tracking

As an extension of single-object tracking, multiple-object tracking requires managing multiple objects, which is a difficult task particularly owing to the occlusions occurring between objects within the same proximity. An occlusion may cause a finding of the foreground regions and a partial or full hiding of the objects. Therefore, it is essential to overcome the occlusion problem to achieve a stable tracking.

Huang and Essa [76] proposed a two-level approach, which consists of a region-level association process and an object-level localization process, for tracking with a stationary-camera. In the region-level tracking process, associations of the foreground regions from successive frames are characterized by a binary correspondence matrix. Rows and columns respectively correspond to existing and new foreground regions, and a non-zero element represents an association between the correspondence regions. Association events can be analyzed from a matrix, e.g., if a column has two or more non-zero entries, correspondence regions will be merged into one region in the current frame. Constructing reliable matrices is cast as an optimization problem based on the GA with the likelihood of associations. Initialization, crossover, and mutation operations are customized based on certain heuristics. During the object-level tracking process, objects are labeled for correct regions and localization based on the association events. The object model used by localization adopts an appearance model and a spatial distribution, as well as an occlusion relationship, to deal with a splitting event. Zhang et al. [77] proposed a species-based PSO, which divides a swarm into several species according to the objects. To overcome the occlusion between different objects, competition and repulsion models for species are introduced. A species with a higher competitive ability indicates that the corresponding object is more likely to occlude other objects, and its repulsive force, which is defined as the product of its competitive ability and velocity, is stronger. A species in which an occlusion has occurred encourages an opponent to be repelled by inserting a repulsion force term into its opponent's velocity update process. Moreover, the authors presented an annealed Gaussian-based PSO (AGPSO) approach, which enables a parameter reduction and fast convergence. AGPSO introduce zero-means Gaussian perturbation noise in velocity update procedure. Its covariance matrix elements decrease exponentially as the iteration progresses, which enables a fast convergence rate.

From these studies, we can find differences in the parts served by the EAs and SAs. In [76], the GA concentrates on reasoning the complex interrelationships between objects using a binary correspondence matrix. By contrast, in [77], PSO is in charge of the entire tracking framework, and an occlusion resolution is well integrated into the general procedure.

In short, Table 11 describes a brief overview of the studies summarized in this section.

9 Face recognition

Face recognition is an important technology in security systems and human-computer interaction. The goal of face recognition is to identify individuals in database who are identical to the input face image. This classification process is conducted using a model trained with feature sets extracted from face images in the database. An illustration of a face recognition procedure is shown in Fig. 10.

In many cases, EAs and SAs are exploited to select or weight the features, as described in Section 9.1. Moreover, fusion methods for features extracted from visible and infrared (IR) images are introduced in Section 9.2. In addition, other methods including the localization, detection, and tracking of the faces and eyes, are shown in

Table 11 Brief information on the literature summarized in Section 8

Algorithm	Author	Section
GA	Bhaskar et al. [70]	Section 8.1
	Huang and Essa [76]	Section 8.2
DE	Cuevas et al. [71]	Section 8.1
	Lin and Zhu [74]	Section 8.1
	Nenavath and Jatoth [75]	Section 8.1
PSO	Zhang et al. [72]	Section 8.1
	Cheng et al. [73]	Section 8.1
	Zhang et al. [77]	Section 8.2

Section 9.3, which are relevant to the pre-processing of an automatic face recognition framework.

9.1 Feature selection/weighting

- Feature selection

The quality of the feature set extracted from a face image has a significant impact on the performance of a face recognition. However, the feature set typically includes noisy, irrelevant, or redundant data. The task of the EAs and SAs is to reduce a feature set of size n to a subset of size m ($m < n$) to improve the face recognition accuracy. Owing to the large number of possible feature subsets, many existing feature selection methods employ a heuristic or random search strategy [82], including a sequential search, tabu search, and greedy algorithms. By contrast, population-based searches using EAs and SAs significantly contribute to high-quality subsets against an extensive search space.

Applying the GA for feature selection is a natural idea because it utilizes bit strings for representing chromosomes. A binary bit value can be used to indicate if a corresponding element assigned is selected or not. Liu and Wechsler [78] proposed an evolutionary pursuit to find the optimal basis by which faces can be projected.

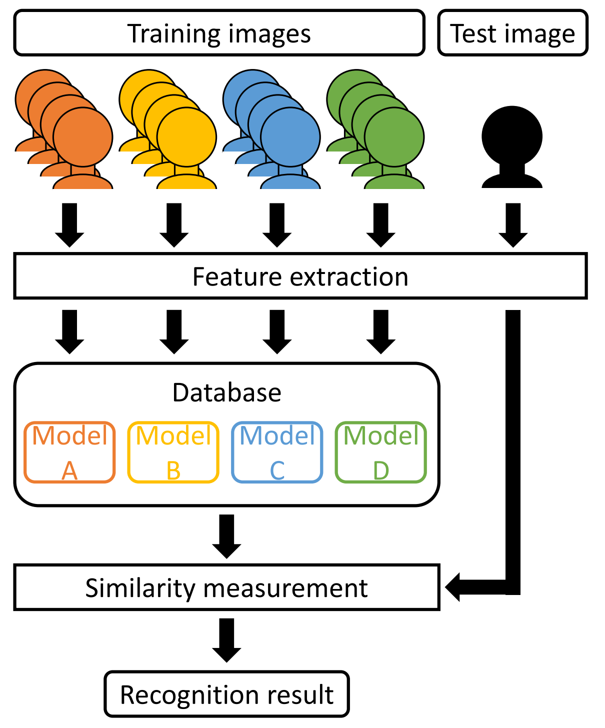


Fig. 10 Illustration of face recognition procedure. The models for each individual are created by extracting features from the training images and stored in database. The test image is identified by finding the model which is most similar to the test image

Specifically, the GA is applied to search the rotation angles for pairwise axes and the combination of basis vectors against the given whitened principal component analysis (PCA) space. The evaluation of every individual is based on the recognition rate and the scatter index. The GA can improve the results by balancing both the classification accuracy and generalization. Zheng et al. [79] proposed the GA-Fisher algorithm, which combines a GA-PCA for a dimension reduction with a linear discriminant analysis (LDA). The GA-PCA approach searches the optimal principal components based on the PCA dimension reduction theorem (PCA-DRT), which claims that some of the small principal components may have useful information. Crossover and mutation operators are improved to retain the number of selected principal components. Each chromosome is evaluated using a fitness function consisting of three terms based on the PCA-DRT. The GA-Fisher approach integrates the GA-PCA with a whitening transformation into the LDA. Vignolo et al. [80] applied the GA and multiple objective functions for feature selection. An aggregative fitness function integrates two types of evaluation functions with relevant parameters. In addition, the multi-objective GA is used to search the optimal Pareto front over three types of objective functions. The introduction of multiple objective functions provides a more flexible classification (e.g., considering not only the accuracy but also a class overlap), and the GA is an effective tool to accomplish this.

Regarding the methods using SAs, Kanan et al. [81] presented a feature selection method based on ACO. In this case, the ants travel on a complete graph consisting of nodes representing the features. Every time an ant chooses any node through a probabilistic transition rule, the current subset is evaluated based on the mean square error (MSE) of the classifier. If the MSE cannot be decreased within several steps, the exploration will terminate and the subset will be output as a candidate. A recognition is achieved using the nearest neighbor classifier, and the obtained MSE is further used by the update of the pheromone in the ACS or rank-based ant system (AS_{rank}). Ramadan and Abdel-Kader [82] utilized a binary PSO whose particles are represented using a binary string similar to a chromosome in the GA. A binary bit indicates whether the corresponding feature is selected. The velocity function is defined as a probability distribution for the update of the particle position. Each particle is evaluated by the class separation term. The term includes the scatter index and searches the optimal subset for discrete cosine transform (DCT) or discrete wavelet transform (DWT) features. Comparison experiments with the GA showed that the binary PSO can acquire smaller feature vectors at the expense of the training time.

Even with the common goal of face recognition, there are many choices of features to be targeted. EAs and SAs are useful methods because they only need to encode the target of the selection on the gene or graph directly. In addition, the GA and ACO, which use a two-optional representation, are preferred in terms of selection.

- Feature weighting

Feature weighting assigns a real value to each feature element. It allows determining the contribution of each feature element in a more detailed way, which can expect to improve the quality of the face recognition. Senaratne et al. [83] proposed EBGM$_{PSO}$, which is an extension of the elastic bunch graph matching (EBGM) [129] algorithm by including PSO during various phases. With the face graph matching procedure, the face graph, which consists of an interior-node grid and a head-boundary-node grid, is represented by particles. The location and size of each grid can vary (each particle has six parameters), and the PSO searches the optimal parameters to maximize the graph similarity. During the recognition phase, recognition-phase-landmark-weights (RPLWs) optimized by the PSO are used to compute a similarity score. In addition, Gabor wavelet features hybridized with eigenface features are also adopted. PSO plays a role in tuning the hybridization weights. The decision to adopt the PSO is based on its lack of limitations (e.g., non-linear and discontinuous). It also has the advantage of a fast convergence. Bhatt et al. [84] introduced a multi-objective evolutionary granular algorithm to recognize surgically altered face images. Each face image is divided into non-disjoint face granules based on three levels of granularity. Because the granules include diverse features, each granule should be assigned a suitable feature extractor and a weight. The GA achieves these two objectives simultaneously using two populations. One population consists of bit string chromosomes for selecting the feature extractors (values of 0 and 1 correspond to SIFT and extended uniform circular local binary patterns (EUCLBP) [130], respectively). The other population consists of chromosomes having real value genes representing weights. All populations evolve independently and are combined when computing the fitness function. To optimize the two objectives with different parameter types simultaneously, the GA is a choice reasonable owing to its encoding flexibility.

For a brief performance comparison, recognition rates of the studies described in this subsection are listed in Table 12.

9.2 Fusion of visible and IR features

Variations in the illumination are a significant problem in face recognition for visible images. IR images can be exploited to overcome this problem. However, IR images also have several drawbacks, such as sensitivity to temperature in the surrounding environment and occlusions by eyeglasses. Because the advantages and disadvantages of both are complementary, the combination of both visible and IR images is considered to allow more accurate recognition performance to be achieved.

Methods of fusing visible and IR images using the GA have been proposed in several studies. Typically, each gene of the GA represents the weight of the corresponding feature component, and the optimum fusion solution is exploited under the framework of the GA. Bebis et al. [85] used a bit string to represent a chromosome and select the feature components from either a visible or IR image. They used the GA for two different fusion schemes, i.e., pixel- and feature-based fusion. The first assigns wavelet coefficients to each gene to obtain a fused image. The second assigns eigenfeatures to each gene to obtain the fused eigenspace. The gene length corresponds to the number of wavelet coefficients or eigenfeatures, and the value of the

Table 12 Recognition rates of the studies summarized in Section 9.1 on different databases

Database	Approach	#Individuals	#Training images per individual	#Test images per individual	Recognition rate
FERET database	Liu and Wechsler [78]	369	2	1	92.14
	Zheng et al. [79]	255	3	1	89.37
	Senaratne et al. [83]	1195	1	1	97.1
CMU PIE database	Zheng et al. [79]	68	3	18	96.81
Essex face database	Vignolo et al. [80]	100	5	15	98.00
ORL database	Kanan et al. [81]	40	-	-	99.75
	Ramadan and Abdel-Kader [82]	40	4	6	96.8
Plastic surgery face database	Bhatt et al. [84]	540	-	-	87.32
Combined heterogeneous face database	Bhatt et al. [84]	1080	-	-	89.87

gene determines whether the corresponding wavelet coefficient is selected from the IR or visible spectrum. Desa and Hati [86] improved the feature-based fusion scheme [85], using kernel-based face subspaces and real number chromosomes. A chromosome represents a weighted vector against the extracted features. The weight given to the corresponding visible and IR features from a gene is complementary, i.e., if any IR features are weighted as α, the corresponding visible feature is weighted as $(1 - \alpha)$. The fused features consist of the sum of the weighted visible and IR features. Similarly, Hermosilla et al. [87] used real number chromosomes to represent weights for the descriptors. However, they assign the weights independently to visible and IR descriptors, and thus, the genetic coding consists of the two corresponding types of weights. The face images are divided into small regions, and the histogram of each region is obtained by the descriptor. The similarities between the probe image and the gallery image are then evaluated based on the sum of the histogram intersections weighted by the corresponding gene values in all regions. Despite requiring genes for both visible and IR features, this method using small regions allows the gene length to be reduced more than in the approaches by [85] and [86].

The three methods detailed in this subsection can be observed as natural extensions of the complexity of a gene representation along the timeline. Consequently, an experiment using the Equinox database in [87] confirms that these extensions directly contribute to an improved recognition rate.

9.3 Other methods

In this subsection, we discuss detection and tracking associated with the face, eyes, and facial expressions, which are important technologies for applications such as human-machine interfaces and surveillance systems.

Wong et al. [88] proposed a face detection and facial feature extraction method for gray-level images using the GA. As the key idea of this approach, the location of the face can be inferred from the location of both eyes. This is based on the fact that the size of a human face is proportional to the distance between both eyes. Therefore, the task of the GA is to select the appropriate pair from the detected eye candidates. This search space limitation and search capability of the GA allow to the high computational cost, which is a challenge to existing methods, to be overcome. A chromosome contains two indexes in a buffer, which stores the candidates of the eye regions, and the eigenfaces are used to evaluate the fitness. Akashi et al. [89] proposed a size- and orientation-invariant eye tracking method for real-time video processing through template matching with the GA. The GA optimizes the parameters of the geometric transformations for the template, consisting of the coordinates, scaling, and rotation. To achieve real-time tracking, they proposed the use of evolutionary video processing. To utilize information between video frames, genetic information in a previous frame is inherited by the current frame (i.e., the evolution continues throughout the entire video sequence), which enables an exploration within a small population size. Perez et al. [90] derived a template generation method using PSO for face localization. The position of the vector components of the PSO correspond to pixels of the template, which represents an angle for directional images. Only the vector components within the allowed range are used as the template, and the evaluation of each particle is defined as a liner integral value of the template over the face directional image. This method and the iris anthropometry template proposed by [131] are applied to a video sequence to localize the face and iris. The template generated through PSO improves the accuracy of the localization more than using an anthropometric template and reduces the computational time because of the decreased number of pixels in the template.

Considering an application operating in a real environment, the front of the target face is not always clearly shown. A head in a 3D space may have an arbitrary pose, which has a significant adverse effect on a face recognition system. Several studies have thus addressed the treatment of 3D face models. Mpiperis et al. [91] proposed a 3D facial expression recognition approach that classifies expressions based on the rule discovered by the PSO or the Ant-Miner (a variant of ACO) framework. ACO explores a graph whose nodes represent the attributes, whereas PSO controls those particles having two parameters (the lower and upper bound per attribute). The difference between the two swarm intelligence approaches is the representation of the attributes. PSO can apply continuous attributes, whereas ACO requires a discretization of the attributes for an assignment of each node. A facial surface is represented as a deformation of a generic 3D mesh, and its facial expression is classified based on a certain rule. Chandar and Savithri [92] introduced an algorithm for estimating a 3D face model from a face with a non-frontal view. This algorithm regards an estimation as an optimization problem when searching for the pose parameters for a face a non-frontal view consisting of angles around the x-, y-, and z-axes, and for depth values of the facial feature points of a frontal-view 3D face model. They used a two-step DE optimization, abbreviated as DE2. In the first step, the pose parameters are optimized, and in the second step, the results of the first step are applied to update the depth values. The two-step optimization enhances the accuracy of the estimated depth values of the facial feature points.

In addition, considering the real-time processing, the application suffers from a constraint in terms of the computational cost. As one solution to this issue, several studies have used 2D approximate multiview face models. Sato and Akashi [93] introduced a high-speed multiview face localization and tracking method using template matching and the GA. This method approximates a human head using a cylinder, which allows a multiview face to be represented through the development of the lateral surface of a cylinder. Although this approximation avoids the computational cost from using a 3D head model, an increase in the parameters of the template matching is induced to create the template from the cylinder head model. The GA provides a significant contribution to the matching (i.e., optimization) with a feasible speed from this extended search space. You and Akashi [94] put forward a multiview face detection algorithm using an existing frontal face detector that requires a training process for frontal faces only. The main idea with this algorithm is a flipping scheme that utilizes a mirror reversal. A proper horizontal reversal for the candidate regions makes it possible to generate frontal faces from multiview faces. The GA is applied to search for the candidate region parameters, which consist of a center point, and scale factors for x- and y-axes and angles. Because speeded-up robust features (SURF) cascade is adopted for the frontal face detection, the fitness function for the GA comprises the number of stages passed and the probability output at the exit stage. When this algorithm is applied to video frames, it can further use the evolutionary video processing proposed by [89]. The two multiview face detection methods described above are designed to operate in real-time. The GA provides a sufficient quality solution with high-speed, thereby contributing to the real-time processing.

In summary, Table 13 provides a brief overview of the studies discussed in this section.

Table 13 Brief information on the literature referenced in Section 9

Algorithm	Author	Section
GA	Liu and Wechsler [78]	Section 9.1
	Zheng et al. [79]	Section 9.1
	Vignolo et al. [80]	Section 9.1
	Bhatt et al. [84]	Section 9.1
	Bebis et al. [85]	Section 9.2
	Desa and Hati [86]	Section 9.2
	Hermosilla et al. [87]	Section 9.2
	Wong et al. [88]	Section 9.3
	Akashi et al. [89]	Section 9.3
	Sato and Akashi [93]	Section 9.3
	You and Akashi [94]	Section 9.3
DE	Chandar and Savithri [92]	Section 9.3
PSO	Ramadan and Abdel-Kader [82]	Section 9.1
	Senaratne et al. [83]	Section 9.1
	Perez et al. [90]	Section 9.3
	Mpiperis et al. [91]	Section 9.3
ACO	Kanan et al. [81]	Section 9.1
	Mpiperis et al. [91]	Section 9.3

10 Human action recognition

Vision-based human action recognition is an essential part of the development of human-computer interaction technologies. To accurately analyze a complex human body structure, many studies use 3D data with depth information as input. In particular, Kinect has made it possible to easily handle RGB-D information. EAs and SAs are effective for processing 3D images involving a high computational cost. Moreover, the affinity for parallel processing is often useful for implementations with practical processing times. We categorize the parts of interest, and provide discussions of each, as follows: body (Section 10.1), hands (Section 10.2), and head (Section 10.3).

10.1 Human body

A human body posture estimation generally aims to fit a human body model, e.g., a skeleton as shown in Fig. 11, to the observed data. This is important for many computer vision applications and is also an input for action recognition, which is described in the second half of this subsection. Because human models usually have numerous DoFs, the fitting problem constructs a high-dimensional and nonlinear solution space. Many of the existing methods employ Bayesian approaches, such as Kalman filtering and particle filtering, but have difficulties in terms of incorporating anatomical constraints and devising realistic body movement models [95].

EAs and SAs can be effective at overcoming the above weaknesses without prior knowledge. Robertson and Trucco [95] proposed an upper-body posture estimation system from multi-view markerless point cloud sequences scanned using a laser scanner. The posture estimation is accomplished by fitting a 24-DoF skeleton model to the given point cloud. The system exploits PSO by comparing the input point cloud with the positions of pairs describing each limb of the skeleton with the absolute distances. They also applied a hierarchical fitting to PSO, which predicts some simply predictable parts on the human body. Furthermore, they proposed a parallel version of PSO across multiple CPUs to increase the efficiency. Their method is highly accurate and can be applied in real-time. Zhang et al. [96] presented a search strategy for 3D human body tracking using an annealed PSO based particle filter (APSOPF). Compared to normal PSO, they applied the annealing strategy into the velocity updating equation of the PSO, which includes a

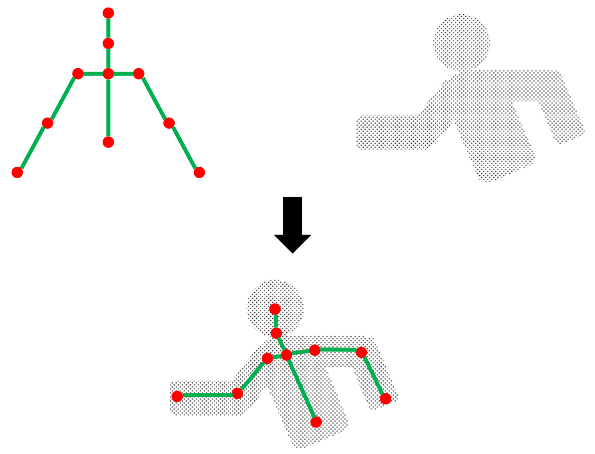


Fig. 11 Illustration of fitting a upper-body skeleton model to the observation data. The fitting procedure is achieved by the adjustment of joint positions indicated by the red circles

sampling covariance and annealing factors. The annealing strategy gradually confines the search area. Experimental results show that this strategy can alleviate the inconsistencies between the observation model and ground truth caused by a self-occlusion. The tracking results are represented as a 3D 31-DoF kinematic tree. Panteleris and Argyros [97] proposed a moving object tracking system with a RGB-D camera in a static environment through simultaneous localization and mapping. The RGB-D camera can provide a dense point cloud of the environment and is registered to another 3D point cloud representing a frame (the cloud contains the object). The registration according to the structure and color information is considered as an optimization, which is optimized using the PSO. The system achieves a real-time capability owing to the efficiency of the PSO. Their tracking results are not accompanied with a human model and are used for a cognitive navigation prosthesis, i.e., a safe navigation of cognitively impaired people in public spaces.

An action recognition is a fundamental technology for applications such as sports analysis and video surveillance. In many cases, the classification of the input data is achieved using machine learning algorithms. EAs and SAs are engaged in supporting such algorithms, and their domains can be categorized into training data [98, 99] and parameter tuning [100, 101]. Although machine learning can solve the problem of vision-based human body action recognition, Chaaraoui and Flórez-Revuelta [98] believe that the learning is insufficient until the training data are complete. Thus, based on the GA, the authors proposed an evolving bag of key poses. When a new pose is input, it can be added to the training dataset for learning. When the new pose belongs to a present pose class, the new pose is operated by a crossover and can go through a mutation under a certain probability. The results of an evolving bag of key poses with both RGB-D and RGB images reveal the availability of incremental learning. Chaaraoui et al. [99] utilized an evolutionary algorithm to determine the optimal subset of human joints in a bag of key poses based on human action recognition with RGB-D cameras. The proposed evolutionary algorithm is comparable to, and based on, the GA but different in terms of crossover. Because a human pose can be regarded as a tree topology, their crossover is aware of such a topology. Their method achieves better results and higher efficiency than other competitive methods at the same time. Ijjina and Chalavadi [100] combined the GA with a CNN to deal with human action recognition. They trained the initial weights of the CNN using global and local optimization by applying the GA and gradient descent algorithm, respectively. During the fitness evaluation step, the CNN classifier is trained using the decoded weights and the gradient descent algorithm, and its classification accuracy is regarded as the fitness value. That is, the GA identifies several local basins, and the gradient descent algorithm quickly finds the optimum within a basin. The human action recognition framework introduced by Nunes et al. [101] involves the DE algorithm. Their framework consists of a feature extraction from the input skeleton data and classification using random forest (RF). The DE algorithm is employed to find the splitting node with the best condition in each decision tree. RF using the DE algorithm has no thresholds to tune, and the DE has certain parameters that are extremely well adjusted based on past studies (i.e., they are controllable, independent of other parameters, and input and output data.) For these studies for action recognition, the performance on different datasets is listed in Table 14.

10.2 Human hands

Similar to the body models, hand models also have the problem of a complex configuration owing to the existence of numerous DoFs. In addition, the presence of

Table 14 The performance of the action recognition approaches on different datasets

Dataset	Approach	Evaluation index	Performance
MSR-Action3D dataset	Chaaraoui and Flórez-Revuelta [98]	Recognition rate	93.10
	Chaaraoui et al. [99]		93.23
UCF50 dataset	Ijjina and Chalavadi [100]	Recognition rate	99.98
CAD-60	Nunes et al. [101]	Precision	81.83
		Recall	80.02

similar parts and severe self-occlusions make the hand model more difficult to handle. Ye et al. [102] put forward a hand pose estimation method for depth images by incorporating a CNN with PSO in each layer. The results predicted by a CNN can often incur a kinematic error, which is solved by the PSO in their research. Each hierarchy of a hand structure is predicted by a refined CNN, in which the refinement is achieved using PSO. The refinement is treated as another optimization problem that takes kinematic constraints into account. Because the next layer is based on the current layer, refining each layer increases the accuracy, but decreases the efficiency. Panteleris and Argyros [103] introduced stereo RGB images (using two monocular cameras) into a hand tracking method. Unlike a naive method that tracks after recovering the depth information and achieves a low accuracy, they transform the hand tracking to maximize the color consistency between the stereo RGB images through the PSO. Particles take the parameters of the observed or tracked hands and are initialized around the center upon the first iteration. Although their method must process multiple images, the method still achieves a real-time implementation.

As a more challenging situation, studies have attempted to track both hands at the same time. More severe occlusions from complex hand interactions have shown that simple extensions of single-hand tracking are insufficient to ensuring the accuracy [104]. Oikonomidis et al. [104] proposed a two-hand tracking method using PSO and RGB-D data. PSO searches in a 54-dimensional solution space to construct an articulation hypothesis. The penalty function is regarded as an objective function consisting of two terms: a prior term that penalizes an invalid articulation hypothesis and a data term that quantifies the incompatibility of the observation with an articulation hypothesis. The proposed method achieves a frame rate of 4 Hz through a parallel implementation using a GPU. Since then, Oikonomidis et al. [105] proposed a novel evolutionary quasi-random search method to achieve a faster processing. The key to this method is the use of a Sobel sequence [132], which allows for a more uniform coverage of the sample space. The method defines a center position at each step of the iteration and generates candidates around it based on the Sobel sequence. All candidates are recorded, and a new center position is determined according to the best candidates and their fitness. This method applies eight parameters, which are tuned using PSO. For two hand tracking, the new method achieves a speedup of 8-times that of the existing method while maintaining the level of accuracy.

10.3 Human head

Head pose estimation is a partial problem of human posture recognition. Padeleris et al. [106] formulated a head pose estimation for depth images as an optimization problem, which is solved using PSO in their research. As the targets of the search, six pose parameters that represent a particular view are applied. The surface model obtained from the depth camera is rendered from the candidate views, and its similarity to the reference range image (the frontal face range image obtained at initialization) is measured. By combining the parallel structure of the PSO with the GPU, the method achieves a frame rate of 10 fps.

Table 15 displays a brief overview of the studies analyzed in this section.

11 Others

This section introduces some studies that are outside the above categories.

Rodehorst and Hellwich [107] proposed GASAC, which is a robust parameter estimation approach using the GA. The task of GASAC is to estimate correct projective transformation parameters by avoiding outliers, which are undesirable correspondences. A chromosome is a tuple of a homologous point index, the length of which is defined as the minimum number of points required to construct the transformation model. Genetic operations are considered to avoid a duplication of the index on the chromosome. The parallel evaluation using the GA makes it possible to improve the estimation accuracy. Moreover, the non-linear optimization method can provides more desirable results because small measurement errors are eliminated. However, the computational cost will increase.

Ghosh et al. [108] introduced a moving object detection method that solves a task by integrating spatial and temporal segmentation. With spatial segmentation, a segmentation is regarded as a pixel labeling problem, which is solved by the maximum a posteriori (MAP) estimation

Table 15 Brief information on the literature referenced in Section 10

Algorithm	Author	Section
GA	Chaaraoui and Flórez-Revuelta [98]	Section 10.1
	Chaaraoui et al. [99]	Section 10.1
	Ijjina and Chalavadi [100]	Section 10.1
	Oikonomidis et al. [105]	10.2
DE	Nunes et al. [101]	Section 10.1
PSO	Robertson and Trucco [95]	Section 10.1
	Zhang et al. [96]	Section 10.1
	Panteleris and Argyros [97]	Section 10.1
	Ye et al. [102]	Section 10.2
	Panteleris and Argyros [103]	Section 10.2
	Oikonomidis et al. [104]	Section 10.2
	Padeleris et al. [106]	Section 10.3

of a multi-layer compound MRF. The authors proposed a distributed DE (DDE) algorithm for a MAP estimation. A parameter vector in the DDE algorithm corresponds to each pixel in the targeted video frame and consists of a segmented output for each RGB channel. Therefore, the size of a population is equal to the number of pixels in the video frame, and the result of the evolution is output as the complete population. To increase the convergence speed, the authors adopted a neighborhood-based mutation. They defined a neighborhood using a small window centered at the target vector to maintain the spatial regularity. A donor vector is generated by three parameter vectors chosen in the window. Moreover, they use a randomly chosen index in a crossover operation, which ensures that a trial vector including at least one parameter forms a donor vector. The segmented frames are used by the temporal segmentation, which classifies whether the region has changed (i.e., is a moving object).

Kumar et al. [109] derived the DE for image enhancement (DE-IE) algorithm. They first design a 2D histogram, which reveals the existence of homogeneous regions based on diagonal values. Because larger diagonal values require a higher intensity enhancement, a color enhancement is achieved by smoothing the 2D histogram. The authors consider a probability distribution, in which each probability density is the pixel length of the gray-level image, as the population for the DE algorithm. The DE algorithm minimizes the difference between the probability distribution of the input image and a satisfactory output image. They endow the mutant factor *F* a different role than a conventional DE algorithm; the *F* value changes based on the difference in the probability distribution between the input image and the output image. An adaptive scheme can be adopted during a mutation operation.

Table 16 provides a brief overview of the studies summarized in this section.

12 Conclusion

This literature survey extensively summarized various computer vision applications employing EAs and SAs developed since 2000. First, we briefly introduced EAs and SAs, focusing particularly on their characteristics and differences. Next, we analyzed and discussed studies applying EAs and SAs to solve computer vision tasks. Different computer vision tasks are described in Section 4 through Section 11, and each general task was classified into subsections according to the problem setting and types of solutions. The vast number of references considered in this paper demonstrate that EAs and SAs are powerful tools for computer vision applications.

Among the four algorithms focused in this paper, the GA and PSO are more popular optimization tools in computer vision applications. Because the GA and PSO are the most representative algorithms of EAs and SAs, respectively. Many characteristics and related researches of GA and PSO have been well verified and studied, such as parameter tuning, improvement of the efficiency, and applications to practical applications. Such accumulated experience in the community make the adoption of GA and PSO to computer vision applications easier. Since the DE is a relatively new algorithm, the application of the DE is not as active as the GA. However, the powerful optimization capability of the DE algorithm may have the potential to attract more researchers in the future. ACO is employed in limited (especially graph-based) topics, such as Sections 5 and 6.

Based on the taxonomy applied in this paper, it can be seen that the solution to whether EAs and SAs should be directly or indirectly adopted varies based on the tasks. For instance, because the tasks described in Sections 7 and 8 are essentially similarity maximization problems between images or models, EAs and SAs play a role as fundamental tools in this process. As described in Section 5 and Section 6, EAs and SAs have frequently been directly applied. However, for recognition problems, such as those described in Sections 9 and 10, classification using machine learning methods is the basis of processing, and EAs and SAs are mainly engaged in boosting the performance. In particular, it can be seen from Table 4 that the development of NAS methods using EAs and SAs described in Section 4 has been an extremely attractive field in recent years.

There exist many foreseeable challenges when applying EAs and SAs to computer vision tasks. First, due to the variety of algorithms, the optimal choice of algorithm for a particular problem remains an open question. Also, the tuning of hyperparameters and the combination of appropriate operators require human experience [30, 60, 87, 114]. Second, many methods include time-consuming processes especially with expensive fitness function, which makes real-time applications difficult to implement [32, 51, 57, 106]. Finally, but not limited to, many real-world problems embed multi-objective optimization, which requires the EAs and SAs to find solutions on the Pareto front [80, 98, 133, 134]. Such challenges need to be faced in order to achieve a breakthrough in applying EAs and SAs to computer vision tasks. Moreover, since DNNs have drawn much attention

Table 16 Brief information on the literature referenced in Section 11

Algorithm	Author	Section
GA	Rodehorst and Hellwich [107]	Section 11
DE	Ghosh et al. [108]	Section 11
	Kumar et al. [109]	Section 11

in computer vision in recent years, the combination of EAs/SAs and DNNs, such as NAS described in Section 4, is one of the promising research directions in this field. Our study can provide a comprehensive reference to the use of EAs and SAs in helping solve various computer vision problems. In addition, we expect that this paper will help broaden the perspective and motivate new insight and research in the relevant fields in the future.

Appendix A: Pseudo-codes

In this section, the pseudo-codes of GA, DE, PSO, and ACO for solving a minimization problem are introduced. Before we get into the description of each algorithm, the notation of the common variables are defined. A population or swarm, i.e., a pool of NP candidate solutions is denoted by $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_{NP}\}$ where $\boldsymbol{x}_i$ is the ith candidate solution called individual, particle, ant, etc. in each algorithm. Each candidate solution also consists of D elements, denoted as $\boldsymbol{x}_i = \{x_{i,1}, ..., x_{i,D}\}$ where $x_{i,j}$ is the jth element of ith candidate solution. The domain of elements depends on each algorithm. The evaluation value (fitness) of $\boldsymbol{x}_i$ obtained by the fitness function is denoted as $f(\boldsymbol{x}_i)$, and the function *evaluate*() calculates the fitness of all input candidate solutions. In addition, considering that these algorithms require iteration processes, the current number of iterations is denoted as t and given as superscript (i.e., tth $\boldsymbol{X}$ is $\boldsymbol{X}^t$). In the following subsections, the pools of intermediate solution candidates generated during iteration processes use the same notation rule for $\boldsymbol{X}$. Also, the alphabets assigned to the variables are only valid within each subsection. Note that the following pseudo-codes are traditional processes, and there are various improved versions.

GA

The pseudo-code of GA is described in Algorithm 1. Each individual $\boldsymbol{x}_i$ has a chromosome encoded by a binary string, i.e., $x_{i,j} \in \{0, 1\}$. $\boldsymbol{O}$ represents the pool of offspring.

Algorithm 1: GA

```
   input  : $t = 0, \boldsymbol{X}^t = \varnothing$
   output: Estimated global optimum $\boldsymbol{x}^*$
   // initialization
1  $NP$ individuals in $\boldsymbol{X}^t$ are randomly initialized;
2  $evaluate(\boldsymbol{X}^t)$;
3  while termination criterion is not satisfied do
4      $\boldsymbol{O} = \varnothing$;
5      while $\boldsymbol{O}$ do not get $NP$ individuals do
           // parents selection
6          $\boldsymbol{x}_{p1}, \boldsymbol{x}_{p2} = selectParents(\boldsymbol{X}^t)$;
           // crossover
7          $\boldsymbol{O} = \boldsymbol{O} \cup crossover(\boldsymbol{x}_{p1}, \boldsymbol{x}_{p2})$;
8      end
9      for $i = 1, ..., NP$ do
           // mutation
10         $\boldsymbol{o}_i = mutate(\boldsymbol{o}_i)$;
           // fitness evaluation
11         $evaluate(\boldsymbol{o}_i)$;
           // survivors selection
12         $\boldsymbol{x}_i^{t+1} = \boldsymbol{o}_i$;
13     end
14     $t = t + 1$;
15 end
16 return $\boldsymbol{x}^* = \min_{\boldsymbol{x} \in \boldsymbol{X}^t} f(\boldsymbol{x})$;
```

- *selectParents*() (Algorithm 1, line 6)

Two individuals as parents are selected from the input population according to their fitness. For instance, the roulette wheel selection defines the probability of each individual being selected:

$$p_i = \frac{f(\boldsymbol{x}_i)}{\sum_{j=0}^{NP} f(\boldsymbol{x}_j)}, \tag{1}$$

where p_i is the probability that ith individual is selected.

- *crossover*() (Algorithm 1, line 7)

The elements (i.e., genes) of the two input parents $\boldsymbol{x}_{p1}$ and $\boldsymbol{x}_{p2}$ are probabilistically exchanged. The two individuals after the operation become part of $\boldsymbol{O}$ as offspring. This operation is performed according to the constant probability called crossover rate p_c, and the parents are directly regarded as offspring if crossover is not performed.

- *mutate*() (Algorithm 1, line 10)

A bit swapping is executed for each gene in each individual with the constant probability called mutation rate p_m. The p_m is generally set to a small value to prevent the information inherited from the parent from being destroyed excessively.

DE

The pseudo-code of DE is described in Algorithm 2. DE generally assumes a continuous optimization problem, i.e., $\boldsymbol{x}_i \in \mathbb{R}^D$. During the iteration, the corresponding donor vector $\boldsymbol{v}_i$ and trial vector $\boldsymbol{u}_i$ are generated for each parent (target vector) $\boldsymbol{x}_i$.

- *mutate*() (Algorithm 2, line 5)

The ith donor vector is created using three randomly chosen individuals $\boldsymbol{x}_{r1}, \boldsymbol{x}_{r2}$, and $\boldsymbol{x}_{r3}$ from the input population as follows:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}), \tag{2}$$

Algorithm 2: DE

```
input  : $t = 0, \boldsymbol{X}^t = \varnothing$
output : Estimated global optimum $\boldsymbol{x}^*$
// initialization
1  $NP$ individuals in $\boldsymbol{X}^t$ are randomly initialized;
2  $evaluate(\boldsymbol{X}^t)$;
3  while termination criterion is not satisfied do
4      for $i = 1, ..., NP$ do
           // mutation
5          $\boldsymbol{v}_i = mutate(\boldsymbol{X}^t)$;
           // crossover
6          $\boldsymbol{u}_i = crossover(\boldsymbol{v}_i, \boldsymbol{x}_i^t)$;
           // fitness evaluation
7          $evaluate(\boldsymbol{u}_i)$;
           // survivors selection
8          if $f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i^t)$ then
9              $\boldsymbol{x}_i^{t+1} = \boldsymbol{u}_i$;
10         else
11             $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t$;
12         end
13     end
14     $t = t + 1$;
15 end
16 return $\boldsymbol{x}^* = \min_{\boldsymbol{x} \in \boldsymbol{X}^t} f(\boldsymbol{x})$;
```

where F is a scaling factor. The indices $i, r1, r2$, and $r3$ are non-duplicated integers in the range $[1, NP]$.

- *crossover*() (Algorithm 2, line 6)

Similar to the crossover of GA, the input target vector and donor vector probabilistically exchange their own components. Only one trial vector is generated, which is a competitor to the corresponding target vector.

PSO

The pseudo-code of PSO is described in Algorithm 3. PSO also assumes a continuous optimization problem similarly to DE. A candidate solution $\boldsymbol{x}_i$ represents the position in a solution space of the corresponding particle, which is modified by velocity $\boldsymbol{v}_i$ of the particle. The *pbest* $\boldsymbol{x}_p$ and *gbest* $\boldsymbol{x}_g$ are the best positions so far for personal particles and the swarm, respectively.

- *adjustVelocity*() (Algorithm 3, line 8)

Velocity is updated using *pbest* and *gbest* as follow:

$$\boldsymbol{v}_i^{t+1} = \omega \boldsymbol{v}_i^t + c_1 r_1 (\boldsymbol{x}_{pi} - \boldsymbol{x}_i^t) + c_2 r_2 (\boldsymbol{x}_g - \boldsymbol{x}_i^t), \tag{3}$$

where ω is the inertia weight, c_1 and c_2 are the acceleration coefficients, and r_1 and r_2 are uniformly random values in the range $[0.0, 1.0]$.

Algorithm 3: PSO

```
input  : $t = 0, \boldsymbol{X}^t = \varnothing, \boldsymbol{V}^t = \varnothing$
output : Estimated global optimum $\boldsymbol{x}^*$
// initialization
1  $NP$ positions in $\boldsymbol{X}^t$ are randomly initialized;
2  $NP$ velocity in $\boldsymbol{V}^t$ are randomly initialized;
3  $evaluate(\boldsymbol{X}^t)$;
4  Each $\boldsymbol{x}_{pi}$ is initialized by the corresponding initial
   position;
5  $\boldsymbol{x}_g$ is initialized by the best position in the swarm;
6  while termination criterion is not satisfied do
7      for $i = 1, ..., NP$ do
           // velocity adjustment
8          $\boldsymbol{v}_i^{t+1} = adjustVelocity(\boldsymbol{v}_i^t, \boldsymbol{x}_i^t, \boldsymbol{x}_{pi}, \boldsymbol{x}_g)$;
           // position adjustment
9          $\boldsymbol{x}_i^{t+1} = adjustPosition(\boldsymbol{v}_i^{t+1}, \boldsymbol{x}_i^t)$;
           // fitness evaluation
10         $evaluate(\boldsymbol{x}_i^{t+1})$;
           // pbest update
11         if $f(\boldsymbol{x}_i^{t+1}) < f(\boldsymbol{x}_{pi})$ then
12             $\boldsymbol{x}_{pi} = \boldsymbol{x}_i^{t+1}$;
13         end
           // gbest update
14         if $f(\boldsymbol{x}_i^{t+1}) < f(\boldsymbol{x}_g)$ then
15             $\boldsymbol{x}_g = \boldsymbol{x}_i^{t+1}$;
16         end
17     end
18     $t = t + 1$;
19 end
20 return $\boldsymbol{x}^* = \boldsymbol{x}_g$;
```

- *adjustPosition*() (Algorithm 3, line 9)

Based on updated velocity, a position update (in other words, generation of the new candidate solution) is executed as follow:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}. \tag{4}$$

ACO

The pseudo-code of ACO is described in Algorithm 4. ACO assumes a combinatorial optimization problem, which is achieved by a feasible walk on a graph whose nodes are solution components (denotes a solution component as c_k) and edges are connections between components. For instance, if the ith ant is on the c_k as initial position and moves to c_l (denotes this movement as c_k^l), elements of the corresponding candidate solution are constructed as $x_{i,0} = c_k$ and $x_{i,1} = c_l$. Every edge is assigned a pheromone (denotes the pheromone on the edge between

```
Algorithm 4: ACO
   input  : $t = 0, X^t = \varnothing, T^t = \varnothing$
   output: Estimated global optimum $x^*$
   // initialization
 1 Every pheromone in $T^t$ is initialized to a value $\tau_0$;
 2 while termination criterion is not satisfied do
 3     for $i = 1, ..., NP$ do
           // solution construction
 4         $x_i^t = constructSolution(T^t)$;
 5     end
       // local search (optional)
 6     $daemonActions(X^t)$;
       // fitness evaluation
 7     $evaluate(X^t)$;
       // pheromone update
 8     $T^{t+1} = updatePheromones(T^t, X^t)$;
 9     $t = t + 1$;
10 end
11 return $x^* = \min_{x \in X^t} f(x)$;
```

c_k and c_l as τ_{kl} and set of all pheromone as T) that indicates the validity of selecting the corresponding component.

- *constructSolution*() (Algorithm 4, line 4)

A feasible solution is constructed by a probabilistically walk of the ant on the graph. Let the current partial solution of x_i be s_p and the set of solution components with feasibility be $\mathcal{N}(s_p)$, then a probability of a solution components is chosen is defined as follow:

$$p(c_k^l \mid s_p) = \frac{\tau_{kl}^{\alpha}\eta_{kl}^{\beta}}{\sum_{c_k^m \in \mathcal{N}(s_p)} \tau_{km}^{\alpha}\eta_{km}^{\beta}}, \forall c_k^l \in \mathcal{N}(s_p), \quad (5)$$

where η is heuristic information, and α and β are parameters that adjust the influences of pheromone and heuristic information, respectively. The choice according to Eq. (5) is repeated until the construction of x_i is complete.

- *daemonActions*() (Algorithm 4, line 6)

Optional local search operations, called daemon actions, can be applied to constructed solutions. These operations are generally centralized actions that cannot be performed by individual ants.

- *updatePheromones*() (Algorithm 4, line 8)

The pheromone update is performed by two mechanisms: evaporation and deposit. While the former gives an equal change to all pheromones, the latter gives a change which depends on the fitness of the ants including the corresponding path. The implementation of these mechanisms is achieved as follows:

$$\tau_{kl}^{t+1} = (1-\rho)\tau_{kl}^{t} + \sum_{x \in X^t | c_k^l \in x} \frac{1}{f(x)}, \quad (6)$$

where ρ is a parameter in the range (0.0, 1.0] called evaporation rate.

Abbreviations
EA: Evolutionary algorithm; SA: Swarm algorithm; EC: Evolutionary computation; GA: Genetic algorithm; DE: Differential evolution; PSO: Particle swarm optimization; ACO: Ant colony optimization; GP: Genetic programming; ES: Evolution strategy; EP: Evolutionary programming; SI: Swarm intelligence; SAD: Sum of absolute differences; DNN: Deep neural network; NAS: Neural architecture search; CNN: Convolutional neural network; CGP: Cartesian genetic programming; CAE: Convolutional auto-encoder; PSNR: Peak signal-to-noise ratio; ACS: Ant colony system; FCM: Fuzzy c-means; MRF: Markov random field; SIFT: Scale-invariant feature transform; SVM: Support vector machine; DoF: Degrees of freedom; DC: Deterministic crowding; CT: Computed tomography; OL: Orthogonal learning; GMM: Gaussian mixture model; SCA: Sine-cosine algorithm; RW: Random walk; IR: Infrared; PCA: Principal component analysis; LDA: Linear discriminant analysis; MSE: Mean square error; DCT: Discrete cosine transform; DWT: Discrete wavelet transform; SURF: Speeded-up robust features; RF: Random forest; MAP: Maximum a posteriori;

Acknowledgements
Not applicable.

Authors' contributions
Takumi Nakane, Naranchimeg Bold, and Haitian Sun contributed equally to this work.

Funding
This work is supported by the Foundation for the Fusion of Science and Technology, JSPS KAKENHI Grant Number [JP20K19568], and Tateisi Science and Technology Foundation.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Engineering, University of Fukui, Fukui, Japan. [2]Department of Electrical Engineering and Computer Science, Iwate University, Iwate, Japan. [3]Department of School of Info Technology, Deakin University, Waurn Ponds, Australia.

Received: 10 February 2020 Accepted: 5 August 2020
Published online: 31 August 2020

References
1. Real E, Moore S, Selle A, Saxena S, Suematsu YL, Tan J, Le QV, Kurakin A (2017) Large-scale evolution of image classifiers. In: International Conference on Machine Learning (ICML). JMLR.org. pp 2902–2911. https://dl.acm.org/doi/10.5555/3305890.3305981
2. Sun Y, Xue B, Zhang M, Yen GG (2020) Evolving Deep Convolutional Neural Networks for Image Classification. IEEE Trans Evol Comput 24(2):394-407. https://doi.org/10.1109/TEVC.2019.2916183
3. Miikkulainen R, Liang J, Meyerson E, Rawal A, Fink D, Francon O, Raju B, Shahrzad H, Navruzyan A, Duffy N, Hodjat B (2019) Chapter 15 - Evolving Deep Neural Networks. In: Robert Kozma, Cesare Alippi, Yoonsuck Choe, Francesco Carlo Morabito (eds). Artificial Intelligence in the Age of Neural Networks and Brain Computing. Academic Press. pp 293–312. https://doi.org/10.1016/B978-0-12-815480-9.00015-3

4. Suganuma M, Shirakawa S, Nagao T (2017) A genetic programming approach to designing convolutional neural network architectures. In: Genetic and Evolutionary Computation Conference (GECCO). ACM. pp 497–504. https://doi.org/10.1145/3071178.3071229
5. Xie L, Yuille A (2017) Genetic cnn. In: International Conference on Computer Vision (ICCV). IEEE. https://doi.org/10.1109/ICCV.2017.154
6. Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K (2018) Hierarchical representations for efficient architecture search. In: International Conference on Learning Representations (ICLR). https://openreview.net/forum?id=BJQRKzbA-
7. Assunção F, Lourenço N, Machado P, Ribeiro B (2018) Evolving the topology of large scale deep neural networks. In: European Conference on Genetic Programming (EuroGP). Springer. pp 19–34. https://doi.org/10.1007/978-3-319-77553-1_2
8. Assunçao F, Lourenço N, Machado P, Ribeiro B (2019) DENSER: deep evolutionary network structured representation. Genet Program Evolvable Mach 20(1):5–35. https://doi.org/10.1007/s10710-018-9339-y
9. Kramer O (2018) Evolution of convolutional highway networks. In: International Conference on the Applications of Evolutionary Computation. Springer. pp 395–404. https://doi.org/10.1007/978-3-319-77538-8_27
10. Sun Y, Xue B, Zhang M, Yen GG (2019) A Particle Swarm Optimization-Based Flexible Convolutional Autoencoder for Image Classification. IEEE Trans Neural Netw Learn Syst 30(8):2295–2309. https://doi.org/10.1109/TNNLS.2018.2881143
11. Wang B, Sun Y, Xue B, Zhang M (2018) Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp 1–8. https://doi.org/10.1109/CEC.2018.8477735
12. Fernando C, Banarse D, Reynolds M, Besse F, Pfau D, Jaderberg M, Lanctot M, Wierstra D (2016) Convolution by evolution: differentiable pattern producing networks. In: Genetic and Evolutionary Computation Conference (GECCO). ACM. pp 109–116. https://doi.org/10.1145/2908812.2908890
13. Suganuma M, Ozay M, Okatani T (2018) Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. In: International Conference on Machine Learning (ICML). PMLR. http://proceedings.mlr.press/v80/suganuma18a.html
14. Oullette R, Browne M, Hirasawa K (2004) Genetic algorithm optimization of a convolutional neural network for autonomous crack detection. In: IEEE Congress on Evolutionary Computation (CEC), vol 1. IEEE. pp 516–521. https://doi.org/10.1109/CEC.2004.1330900
15. Zhining Y, Yunming P (2015) The genetic convolutional neural network model based on random sample. Int J U- E-Serv Sci Technol (UNESST) 8(11):317–326
16. Tao W-B, Tian J-W, Liu J (2003) Image segmentation by three-level thresholding based on maximum fuzzy entropy and genetic algorithm. Pattern Recogn Lett 24(16):3069–3078
17. Tao W, Jin H, Liu L (2007) Object segmentation using ant colony optimization algorithm and fuzzy entropy. Pattern Recogn Lett 28(7):788–796
18. Puranik P, Bajaj P, Abraham A, Palsodkar P, Deshmukh A (2009) Human perception-based color image segmentation using comprehensive learning particle swarm optimization. In: International Conference on Emerging Trends in Engineering and Technology (ICETET). IEEE. pp 630–635. https://doi.org/10.1109/ICETET.2009.116
19. Liang Y-C, Chen AH-L, Chyu C-C (2006) Application of a hybrid ant colony optimization for the multilevel thresholding in image processing. In: International Conference on Neural Information Processing (ICONIP). Springer. pp 1183–1192. https://doi.org/10.1007/11893257_129
20. Ghamisi P, Couceiro MS, Martins FM, Benediktsson JA (2014) Multilevel image segmentation based on fractional-order darwinian particle swarm optimization. IEEE Trans Geosci Remote Sens (TGRS) 52(5):2382–2394
21. Liang Y-C, Yin Y-C (2011) Optimal multilevel thresholding using a hybrid ant colony system. J Chin Inst Ind Eng 28(1):20–33
22. Liang Y, Yin Y (2013) Int J Innov Comput Inf Control (IJICIC) 9(1):319–337
23. Chander A, Chatterjee A, Siarry P (2011) A new social and momentum component adaptive pso algorithm for image segmentation. Expert Syst Appl 38(5):4998–5004
24. Omran M, Engelbrecht AP, Salman A (2005) Particle swarm optimization method for image clustering. Int J Patt Recog Artif Intell (IJPRAI) 19(03):297–321
25. Malisia AR, Tizhoosh HR (2006) Image thresholding using ant colony optimization. In: Conference on Computer and Robot Vision (CRV). IEEE. pp 26–26. https://doi.org/10.1109/CRV.2006.42
26. Maulik U, Bandyopadhyay S (2003) Fuzzy partitioning using real coded variable length genetic algorithm for pixel classiocation. IEEE Trans Geosci Remote Sens (TGRS) 41(5):1075–1081
27. Omran MG, Salman A, Engelbrecht AP (2006) Dynamic clustering using particle swarm optimization with application in image segmentation. Pattern Anal Appl 8(4):332
28. Awad M, Chehdi K, Nasri A (2007) Multicomponent image segmentation using a genetic algorithm and artificial neural network. IEEE Geosci Remote Sens Lett (GRSL) 4(4):571–575
29. Awad M, Chehdi K, Nasri A (2009) Multi-component image segmentation using a hybrid dynamic genetic algorithm and fuzzy c-means. IET Image Process 3(2):52–62
30. Bansal S, Aggarwal D (2011) Color image segmentation using cielab color space using ant colony optimization. Int J Comput Appl (IJCA) 29(9):28–34
31. Halder A, Pramanik S, Kar A (2011) Dynamic image segmentation using fuzzy c-means based genetic algorithm. Int J Comput Appl (IJCA) 28(6):15–20
32. Halder A, Pradhan A, Dutta SK, Bhattacharya P (2016) Tumor extraction from mri images using dynamic genetic algorithm based image segmentation and morphological operation. In: International Conference on Communication and Signal Processing (ICCSP). IEEE. pp 1845–1849. https://doi.org/10.1109/ICCSP.2016.7754489
33. Ouadfel S, Batouche M (2003) MRF-based image segmentation using ant colony system. Electronic Letters on Computer Vision and Image Analysis (ELCVIA) 2(1):12–24
34. Pignalberi G, Cucchiara R, Cinque L, Levialdi S (2003) Tuning range image segmentation by genetic algorithm. EURASIP J Adv Signal Process 2003(8):683043
35. Tianzi J, Faguo Y, Yong F, David JE (2001) A parallel genetic algorithm for cell image segmentation. Electron Notes Theor Comput Sci (ENTCS) 46:214–224
36. Wang X-N, Feng Y.-j., Feng Z-R (2005) Ant colony optimization for image segmentation. In: International Conference on Machine Learning and Cybernetics (ICMLC), vol 9. IEEE. pp 5355–5360. https://doi.org/10.1109/ICMLC.2005.1527890
37. Ma L, Wang K, Zhang D (2009) A universal texture segmentation and representation scheme based on ant colony optimization for iris image processing. Comput Math Appl 57(11-12):1862–1868
38. Nezamabadi-Pour H, Saryazdi S, Rashedi E (2006) Edge detection using ant algorithms. Soft Comput 10(7):623–628
39. Baterina AV, Oppus C (2010) Image edge detection using ant colony optimization. WSEAS Trans Signal Process 6(2):58–67
40. Cuevas E, Zaldivar D, Pérez-Cisneros M, Ramírez-Ortegón M (2011) Circle detection using discrete differential evolution optimization. Pattern Anal Appl 14(1):93–107
41. Dong N, Wu C-H, Ip W-H, Chen Z-Q, Chan C-Y, Yung K-L (2012) An opposition-based chaotic GA/PSO hybrid algorithm and its application in circle detection. Comput Math Appl 64(6):1886–1902
42. Trujillo L, Olague G (2006) Using evolution to learn how to perform interest point detection. Int Conf Pattern Recog (ICPR) 1:211–214
43. Trujillo L, Olague G (2008) Automated Design of Image Operators That Detect Interest Points. In: Evolutionary Computation, vol. 16. MIT Press. pp 483–507. https://doi.org/10.1162/evco.2008.16.4.483
44. Perez CB, Olague G (2009) Evolutionary learning of local descriptor operators for object recognition. In: Genetic and Evolutionary Computation Conference (GECCO). ACM. pp 1051–1058. https://doi.org/10.1145/1569901.1570043
45. Perez CB, Olague G (2013) Genetic programming as strategy for learning image descriptor operators. Intell Data Anal 17(4):561–583
46. Yu S, De Backer S, Scheunders P (2002) Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. Pattern Recogn Lett 23(1-3):183–190
47. Treptow A, Zell A (2004) Combining adaboost learning and evolutionary search to select features for real-time object detection. In: IEEE Congress on Evolutionary Computation (CEC), vol 2. IEEE. pp 2107–2113. https://doi.org/10.1109/CEC.2004.1331156

48. Khushaba RN, Al-Ani A, Al-Jumaily A (2008) Differential evolution based feature subset selection. In: International Conference on Pattern Recognition (ICPR). IEEE. pp 1–4. https://doi.org/10.1109/ICPR.2008.4761255
49. Khushaba RN, Al-Ani A, Al-Jumaily A (2011) Feature subset selection using differential evolution and a statistical repair mechanism. Expert Syst Appl 38(9):11515–11526
50. Ghosh A, Datta A, Ghosh S (2013) Self-adaptive differential evolution for feature selection in hyperspectral image data. Appl Soft Comput 13(4):1969–1977
51. Ghamisi P, Couceiro MS, Benediktsson JA (2015) A novel feature selection approach based on FODPSO and SVM. IEEE Trans Geosci Remote Sens Lett (TGRS) 53(5):2935–2947
52. Ghamisi P, Chen Y, Zhu XX (2016) A self-improving convolution neural network for the classification of hyperspectral data. IEEE Geosci Remote Sens Lett (GRSL) 13(10):1537–1541
53. Al-Ani A (2005) Feature subset selection using ant colony optimization. Int J Comput Intell (IJCI) 2(1):53–58
54. Chen B, Chen L, Chen Y (2013) Efficient ant colony optimization for image feature selection. Signal Process 93(6):1566–1576
55. Zhang C, Akashi T (2015) Simplifying genetic algorithm: a bit order determined sampling method for adaptive template matching. In: Irish Machine Vision and Image Processing Conference (IMVIP). Irish Pattern Recognition & Classification Society. http://www.tara.tcd.ie/handle/2262/74714
56. Zhang C, Akashi T (2015) Fast affine template matching over Galois field. In: British Machine Vision Conference (BMVC), BMVA Press. pp 121–112111. https://dx.doi.org/10.5244/C.29.121
57. Zhang C, Akashi T (2016) Inst Electron Inf Commun Eng (IEICE) 99(9):2341–2350
58. Sato J, Akashi T (2018) Deterministic crowding introducing the distribution of population for template matching. IEEJ Trans Electr Electron Eng 13(3):480–488
59. Lee Y, Hara T, Fujita H, Itoh S, Ishigaki T (2001) Automated detection of pulmonary nodules in helical CT images based on an improved template-matching technique. IEEE Trans Med Imaging (T-MI) 20(7):595–604
60. Ugolotti R, Nashed YS, Mesejo P, Ivekovič Š, Mussi L, Cagnoni S (2013) Particle swarm optimization and differential evolution for model-based object detection. Appl Soft Comput 13(6):3092–3105
61. De Falco I, Della Cioppa A, Maisto D, Tarantino E (2008) Differential evolution as a viable tool for satellite image registration. Appl Soft Comput 8(4):1453–1462
62. Ma W, Fan X, Wu Y, Jiao L (2014) An orthogonal learning differential evolution algorithm for remote sensing image registration. Math Probl Eng 2014:1–11
63. Wachowiak MP, Smolíková R, Zheng Y, Zurada JM, Elmaghraby AS (2004) An approach to multimodal biomedical image registration utilizing particle swarm optimization. IEEE Trans Evol Comput (TEVC) 8(3):289–301
64. Liebelt J, Schertler K (2007) Precise registration of 3D models to images by swarming particles. In: Computer Vision and Pattern Recognition (CVPR). IEEE. pp 1–8. https://doi.org/10.1109/CVPR.2007.383167
65. Sholomon D, David O, Netanyahu NS (2013) A genetic algorithm-based solver for very large jigsaw puzzles. In: Computer Vision and Pattern Recognition (CVPR). IEEE. pp 1767–1774. https://doi.org/10.1109/CVPR.2013.231
66. Sholomon D, David OE, Netanyahu NS (2016) An automatic solver for very large jigsaw puzzles using genetic algorithms. Genet Program Evolvable Mach 17(3):291–313
67. Sholomon D, David OE, Netanyahu NS (2014) A generalized genetic algorithm-based solver for very large jigsaw puzzles of complex types. In: Association for the Advancement of Artificial Intelligence (AAAI). pp 2839–2845. https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8650
68. Myers R, Hancock ER (2001) Least-commitment graph matching with genetic algorithms. Patt Recogn 34(2):375–394
69. Zhang L, Xu W, Chang C (2003) Genetic algorithm for affine point pattern matching. Pattern Recogn Lett 24(1-3):9–19
70. Bhaskar H, Kingsland R, Singh S (2006) Multi-resolution based motion estimation for object tracking using genetic algorithm. In: 2006 IET International Conference on Visual Information Engineering. IET. pp 583–588. https://doi.org/10.1049/cp:20060596
71. Cuevas E, Zaldivar D, Pérez-Cisneros M, Oliva D (2013) Block matching algorithm based on differential evolution for motion estimation. Eng Appl Artif Intell 26(1):488–498
72. Zhang X, Hu W, Maybank S, Li X, Zhu M (2008) Sequential particle swarm optimization for visual tracking. In: Computer Vision and Pattern Recognition (CVPR). IEEE. pp 1–8. https://doi.org/10.1109/CVPR.2008.4587512
73. Cheng X, Li N, Zhang S, Wu Z (2014) Robust visual tracking with sift features and fragments based on particle swarm optimization. Circ Syst Signal Process 33(5):1507–1526
74. Lin L, Zhu M (2018) Efficient tracking of moving target based on an improved fast differential evolution algorithm. IEEE Access 6:6820–6828
75. Nenavath H, Jatoth RK (2018) Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking. Appl Soft Comput 62:1019–1043
76. Huang Y, Essa I (2005) Tracking multiple objects through occlusions. In: Computer Vision and Pattern Recognition (CVPR), vol 2. IEEE. pp 1051–1058. https://doi.org/10.1109/CVPR.2005.350
77. Zhang X, Hu W, Qu W, Maybank S (2010) Multiple object tracking via species-based particle swarm optimization. IEEE Trans Circ Syst Video Technol 20(11):1590–1602
78. Liu C, Wechsler H (2000) Evolutionary pursuit and its application to face recognition. IEEE Trans Patt Anal Mach Intell (TPAMI) 22(6):570–582
79. Zheng W-S, Lai J-H, Yuen PC (2005) IEEE Trans Syst Man Cybern B (Cybernet) 35(5):1065–1078
80. Vignolo LD, Milone DH, Scharcanski J (2013) Feature selection for face recognition based on multi-objective evolutionary wrappers. Expert Syst Appl 40(13):5077–5084
81. Kanan HR, Faez K, Hosseinzadeh M (2007) Face recognition system using ant colony optimization-based selected features. In: IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). IEEE. pp 57–62
82. Ramadan RM, Abdel-Kader RF (2009) Face recognition using particle swarm optimization-based selected features. Int J Signal Process Image Process Patt Recogn (IJSIP) 2(2):51–65
83. Senaratne R, Halgamuge S, Hsu A (2009) Face recognition by extending elastic bunch graph matching with particle swarm optimization. J Multimed 4(4):204–214
84. Bhatt HS, Bharadwaj S, Singh R, Vatsa M (2013) Recognizing surgically altered face images using multiobjective evolutionary algorithm. IEEE Trans Inf Forensics Secur (TIFS) 8(1):89–100
85. Bebis G, Gyaourova A, Singh S, Pavlidis I (2006) Face recognition by fusing thermal infrared and visible imagery. Image Vis Comput 24(7):727–742
86. Desa SM, Hati S (2008) IR and visible face recognition using fusion of kernel based features. In: International Conference on Pattern Recognition (ICPR). Citeseer. pp 1–4. https://doi.org/10.1109/ICPR.2008.4761862
87. Hermosilla G, Gallardo F, Farias G, Martin CS (2015) Fusion of visible and thermal descriptors using genetic algorithms for face recognition systems. Sensors 15(8):17944–17962
88. Wong K-W, Lam K-M, Siu W-C (2001) An efficient algorithm for human face detection and facial feature extraction under different conditions. Patt Recogn 34(10):1993–2004
89. Akashi T, Wakasa Y, Tanaka K, Karungaru S, Fukumi M (2007) Using genetic algorithm for eye detection and tracking in video sequence. J System Cybern Inform (JSCI) 5(2):72–78
90. Perez CA, Aravena CM, Vallejos JI, Estevez PA, Held CM (2010) Face and iris localization using templates designed by particle swarm optimization. Patt Recogn Lett 31(9):857–868
91. Mpiperis I, Malassiotis S, Petridis V, Strintzis MG (2008) 3D facial expression recognition using swarm intelligence. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE. pp 2133–2136
92. Chandar KP, Savithri TS (2015) 3D face model estimation based on similarity transform using differential evolution optimization. Procedia Comput Sci 54:621–630
93. Sato J, Akashi T (2017) High-speed multiview face localization and tracking with a minimum bounding box using genetic algorithm. IEEJ Trans Electr Electron Eng (TEEE) 12(5):736–743

94. You M, Akashi T (2018) Multi-view face detection using frontal face detector. IEEJ Trans Electr Electron Eng (TEEE) 13(7):1011–1019
95. Robertson C, Trucco E (2006) Human body posture via hierarchical evolutionary optimization. In: Br Mach Vis Conf (BMVC). p 999. http://www.macs.hw.ac.uk/bmvc2006/volume3.html
96. Zhang X, Hu W, Wang X, Kong Y, Xie N, Wang H, Ling H, Maybank S (2010) A swarm intelligence based searching strategy for articulated 3D human body tracking. In: Comput Vis Patt Recogn Workshops (CVPRW). IEEE. pp 45–50. https://doi.org/10.1109/CVPRW.2010.5543804
97. Panteleris P, Argyros AA (2014) Vision-based slam and moving objects tracking for the perceptual support of a smart walker platform. In: European Conference on Computer Vision (ECCV). Springer. pp 407–423. https://doi.org/10.1007/978-3-319-16199-0_29
98. Chaaraoui AA, Florez-Revuelta F (2014) Adaptive human action recognition with an evolving bag of key poses. IEEE Trans Auton Mental Dev (TAMD) 6(2):139–152
99. Chaaraoui AA, Padilla-López JR, Climent-Pérez P, Flórez-Revuelta F (2014) Evolutionary joint selection to improve human action recognition with RGB-D devices. Expert Syst Appl 41(3):786–794
100. Ijjina EP, Chalavadi KM (2016) Human action recognition using genetic algorithms and convolutional neural networks. Patt Recogn 59:199–212
101. Nunes UM, Faria DR, Peixoto P (2017) A human activity recognition framework using max-min features and key poses with differential evolution random forests classifier. Patt Recogn Lett 99:21–31
102. Ye Q, Yuan S, Kim T-K (2016) Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation. In: European Conference on Computer Vision (ECCV). Springer. pp 346–361. https://doi.org/10.1007/978-3-319-46484-8_21
103. Panteleris P, Argyros A (2017) Back to RGB: 3D tracking of hands and hand-object interactions based on short-baseline stereo. IEEE Int Conf Comput Vis Workshops (ICCVW) 2(63):39
104. Oikonomidis I, Kyriazis N, Argyros AA (2012) Tracking the articulated motion of two strongly interacting hands. In: Computer Vision and Pattern Recognition (CVPR). IEEE. https://doi.org/10.1109/CVPR.2012.6247885
105. Oikonomidis I, Lourakis MI, Argyros AA (2014) Evolutionary quasi-random search for hand articulations tracking. In: Computer Vision and Pattern Recognition (CVPR). IEEE. pp 3422–3429. https://doi.org/10.1109/CVPR.2014.437
106. Padeleris P, Zabulis X, Argyros AA (2012) Head pose estimation on depth data based on Particle Swarm Optimization. In: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. IEEE. pp 42–49. https://doi.org/10.1109/CVPRW.2012.6239236
107. Rodehorst V, Hellwich O (2006) Genetic Algorithm SAmple Consensus (GASAC) - A Parallel Strategy for Robust Parameter Estimation. In: 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06). IEEE. pp 103–103. https://doi.org/10.1109/CVPRW.2006.88
108. Ghosh A, Mondal A, Ghosh S (2014) Moving object detection using Markov random field and distributed differential evolution. Appl Soft Comput 15:121–136
109. Kumar S, Pant M, Ray AK (2018) DE-IE: differential evolution for color image enhancement. Int J Syst Assur Eng Manag 9(3):577–588
110. Chen Q, Koltun V (2015) Robust nonrigid registration by convex optimization. In: Proceedings of the IEEE International Conference on Computer Vision. IEEE. pp 2039–2047. https://doi.org/10.1109/ICCV.2015.236
111. Zhou X, Leonardos S, Hu X, Daniilidis K (2015) 3D shape estimation from 2D landmarks: a convex relaxation approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE. pp 4447–4455. https://doi.org/10.1109/CVPR.2015.7299074
112. Cheng Y, Lopez JA, Camps O, Sznaier M (2015) A convex optimization approach to robust fundamental matrix estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE. pp 2170–2178. https://doi.org/10.1109/CVPR.2015.7298829
113. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31
114. Sizikova E, Funkhouser T (2016) Wall painting reconstruction using a genetic algorithm. EUROGRAPHICS Work Graph Cult Herit (GCH) 11(1):3
115. Nayman N, Noy A, Ridnik T, Friedman I, Jin R, Zelnik L (2019) Xnas: neural architecture search with expert advice. In: Advances in Neural Information Processing Systems. Curran Associates, Inc. pp 1977–1987. https://doi.org/http://papers.nips.cc/paper/8472-xnas-neural-architecture-search-with-expert-advice.pdf
116. Cai H, Zhu L, Han S (2019) ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In: International Conference on Learning Representations
117. Stanley KO, Clune J, Lehman J, Miikkulainen R (2019) Designing neural networks through neuroevolution. Nat Mach Intell 1(1):24–35
118. Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: a survey. J Mach Learn Res 20(55):1–21
119. Xin Y (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447
120. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127
121. Chrabaszcz P, Loshchilov I, Hutter F (2018) Back to basics: benchmarking canonical evolution strategies for playing atari. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. AAAI Press. pp 1419–1426. https://dl.acm.org/doi/10.5555/3304415.3304617
122. Goldberg DE, Deb K (1991) A comparative analysis of selection schemes used in genetic algorithms. Found Genet Algoritm 1:69–93
123. Carlson SE (1996) Genetic algorithm attributes for component selection. Res Eng Des 8(1):33–51
124. Zhao M, Fu AM, Yan H (2001) A technique of three-level thresholding based on probability partition and fuzzy 3-partition. IEEE Trans Fuzzy Syst 9(3):469–479
125. Xue B, Zhang M, Browne WN, Yao X (2016) A survey on evolutionary computation approaches to feature selection. IEEE Trans Evol Comput 20(4):606–626
126. Zhang D-Q, Chang S-F (2004) Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In: Proceedings of the 12th Annual ACM International Conference on Multimedia. Association for Computing Machinery. pp 877–884. https://doi.org/10.1145/1027527.1027730
127. Dasigi P, Jawahar CV (2008) Efficient graph-based image matching for recognition and retrieval. In: Proceedings of National Conference on Computer Vision, Pattern Recognition. http://web2py.iiit.ac.in/publications/default/download/inproceedings.pdf.f4613d92-8ea2-4905-b7fb-08702c4b301d.pdf
128. Lamdan Y, Schwartz JT, Wolfson HJ (1988) Object recognition by affine invariant matching. In: Computer Vision and Pattern Recognition (CVPR). IEEE. pp 335–344. https://doi.org/10.1109/CVPR.1988.196257
129. Wiskott L, Fellous J-M, Kruger N, Von Der Malsburg C (1997) Face recognition by elastic bunch graph matching. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 19. IEEE. pp 775–779. https://doi.org/10.1109/34.598235
130. Bhatt HS, Bharadwaj S, Singh R, Vatsa M (2010) On matching sketches with digital face images. In: 2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS). IEEE. pp 1–7. https://doi.org/10.1109/BTAS.2010.5634507
131. Perez CA, Lazcano VA, Estevez PA (2007) Real-time iris detection on coronal-axis-rotated faces. IEEE Trans Syst Man Cybern C (Appl Rev) 37(5):971–978
132. Sobol' IM (1967) On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput Math Math Phys 7(4):86–112
133. Sarkar S, Das S (2013) Multi-level image thresholding based on two-dimensional histogram and maximum tsallis entropy - a differential evolution approach. IEEE Trans Image Process 22(12):4788–4797
134. Maulik U, Saha I (2010) Automatic fuzzy clustering using modified differential evolution for image classification. IEEE Trans Geosci Remote Sens (TGRS) 48(9):3503–3510

Publisher's Note