

Chapter 1

Artificial Evolution Based Autonomous Robot Navigation

Modern robots are required to carry out work in unstructured dynamic human environments. In the recent decades, the application of artificial evolution to autonomous mobile robots to enable them to adapt their behaviors to changes of the environments has attracted much attention. As a result, an infant research field called evolutionary robotics has been rapidly developed that is primarily concerned with the use of artificial evolution techniques for the automatic design of adaptive robots. As an innovative and effective solution to autonomous robot controller design, it can derive adaptive robotic controllers capable of elegantly dealing with continuous changes in unstructured environments in real time. In the chapter, the basic concepts regarding artificial evolution and evolutionary robotics are introduced, and then a variety of successful applications of artificial evolution in autonomous robot navigation along the dimension of artificial evolution adopted are surveyed and discussed. Open issues and future research in this field are also presented.

1.1 Introduction

Early robots were nothing more than clever mechanical devices that performed simple pick-and-place operations. Nowadays robots are becoming more sophisticated and diversified so as to meet the ever-changing user requirements. The robots are developed to perform more precise industrial operations, such as welding, spray painting, and simple parts assembly. However, such operations do not really require the robot to have intelligence and behave like human beings since the robots are simply programmed to perform a series of repetitive tasks. If anything interferes with the pre-specified task, the robot cannot work properly anymore, since it is not

capable of sensing its external environment and figuring out what to do independently.

Robotics today has moved from the structured factory floors to the unpredictable human environment [Khatib, Brock, and Change, et al., 2002]. Therefore, traditional manipulator controlled robots are being replaced by the emerging autonomous intelligent mobile robots. Let's first look at what capabilities a robot should have:

Autonomous: having autonomy; not subject to control from outside; independent (Webster's New Universal Unabridged Dictionary, Barnes and Noble Books, New York, 1996).

Intelligent: pertaining to the ability to do data processing locally, smart (Webster's New Universal Unabridged Dictionary, Barnes and Noble Books, New York, 1996).

Such robots have the ability to adjust their behaviors autonomously in the ever-changing physical environment. Here a simple definition of a robot could be "a mechanism which is able to move and react to its environment". There are many types of robots and the robots discussed in this chapter are autonomous mobile robots. Autonomous robots refer to the agents capable of executing the specified tasks without human intervention by adjusting their behaviors based on the real environment. Mobile robots refer to those which navigate and perform tasks without external intervention. Thus, an autonomous mobile robot should be able to make decisions independently and adaptively in the real-world environments. As an example, complex robotic tasks such as trash collection using autonomous robots can be broadly applied to a variety of fields such as product transferring in manufacturing factory, rubbish cleaning in office, and bomb searching on battle field, etc. Such robots should be able to cope with the large amount of uncertainties existing in the physical environment. For instance, the robot may be required to achieve certain goals without colliding with obstacles in this motion. However, the physical environment is usually dynamic and unpredictable. Quite often the obstacles are not static and moreover, the sensor readings are imprecise and unreliable because of the noises. Therefore, the robot cannot operate properly in the real world anymore with a highly pre-programmed controller. For robots to become more efficient, maintenance free, and productive, they must be capable of making decisions independently according to the real situations [Staugaard, Jr., 1987]. The robot must be able to adjust its behavior by itself online and make appropriate decisions under various uncertainties encountered during its motion in an independent and smart fashion.

Therefore, a key challenge in autonomous robotics is to design control algorithms that allow robots to function adaptively in unstructured, dynamic, partially observable, and uncertain environments [Sukhatme and Mataric, 2002]. Humanoid robots [Brooks, 2001, 2002; Lund, Bjerre, and Nielsen, et al., 1999], self-reconfiguring robots [Rus, Kotay, and Vona, 2002], probabilistic robotics [Thrun, 2002], entertainment robotics [Velo, 2002] are some representatives in this emerging field these years.

In the last several decades, biologically inspired robotics [Taubes, 2000; Full, 2001] such as evolutionary robotics (ER) [Floreano, 1997; Nolfi, 1998b] is being developed as an innovative research field, which is an artificial evolution based methodology to obtaining self-adaptation and self-learning capabilities for robots to perform desired missions. For instance, investigations are being conducted on the use of robots in the imitation of life [Holland, 2001] and the integrated design using a co-evolutionary learning approach [Pollack, et al., 2001].

Several researchers have given overviews on evolutionary robotics and specific topics such as neural networks, fuzzy logic, and evolutionary algorithms several years ago [Nolfi, Floreano, and Miglino, et al., 1994; Gomi and Griffith, 1996; Walker and Oliver, 1997; Meyer, Husbands, and Harvey, 1998; Meeden and Kumar, 1998; Wang, 2002]. This review is intended to give a more state-of-the-art discussion, emphasizing the artificial evolution based approach to autonomous robot navigation in terms of methods and techniques currently used, and open issues and future trends, most of which are not covered in previous ER surveys. The remainder of the chapter is organized as follows. Section 1.2 presents the characteristics of evolutionary robotics. The adaptive autonomous robot navigation is discussed in Section 1.3, where online and offline evolutions are compared. In Section 1.4, some typical artificial evolution methods used in deriving robotic controllers are discussed one by one, which include neural networks, evolutionary algorithms, fuzzy logic, and other methods. Open issues and future prospects are detailed in Section 1.5, which include SAGA, combination of evolution and learning, inherent fault tolerance, hardware evolution, online evolution, and ubiquitous and collective robots. Finally, the conclusion is drawn in Section 1.6.

1.2 Evolutionary Robotics

Robots have developed along two paths, i.e., industrial and domestic. Industrial robots have been developed to perform a variety of manufacturing tasks such as simple product assembly. However, most industrial robots are unintelligent as they cannot hear, see, or feel. For instance, if assembly parts are not presented to the robot in a precise and repetitive fashion, the robot cannot perform its task properly. In autonomous robot navigation, the robot should be able to move purposefully and without human intervention in environments that have not been specifically engineered for it. Autonomous robot navigation requires the robot to execute elementary robotic actions, to react promptly to unexpected events, and to adapt to unforeseen changes in the environment.

To become more intelligent in robot navigation, an autonomous robot should be able to sense its surroundings and respond to a changing environment promptly and properly. For instance, in the robotic task of target collection, target recognition and path planning are the two challenging tasks that a robot should execute. Target recognition under uncertainties is difficult to implement because the sensor information is usually too noisy to be directly mapped to a definitive target representation. Multi-sensor data fusion has turned out to be an effective approach to resolve this problem. Sensor data fusion is to combine and integrate inputs from different sources into one representational format. It yields more meaningful information as compared to the data obtained from any individual sources. Data fusion in robotics is very often used in applications such as pattern recognition and localization for autonomous robot navigation. The associated techniques include the least square method (LSM), Bayesian method, fuzzy logic, neural network, and so on. Each of these methods has their own merits. For instance, fuzzy logic has the advantage of mapping imprecise or noisy information as input into the output domain. It is particularly effective in autonomous robot navigation since the data is obtained under uncertain conditions. Their operation and inference rules can be represented by natural languages in form of linguistic variables. Traditional robot controllers are often designed to run in a well-defined environment to execute certain repetitive or fixed actions. However, actual autonomous robotic applications may require the mobile robots to be able to react quickly to unpredictable situations in a dynamic environment without any human intervention. For instance, personal and service robots are two important trends for the next generation of robotic applications [Schraft

and Schmierer, 2000]. In both cases, what matters most is the necessity to adapt to new and unpredictable situations, react quickly, display behavioral robustness, and operate in close interaction with human beings [Floreano and Urzelai, 2000; Hopgood, 2003]. However, for such mobile robot applications, it is difficult to model unstructured external environments perceived via sensors in sufficient detail, which are changing continuously. Therefore, traditional robotic controllers designed for factory automation are not suited for these types of applications anymore. The designed robot should be able to have the capability of self-organization for navigating in such dynamic environments. Self-organizing systems can operate in unforeseen situations and adapt to changing conditions. There are a variety of robotic architectures when building autonomous robot navigation systems. The most commonly used one is the hierarchical architecture, where functionalities of the robotic system are decomposed into high- and low-level layers [Nicolescu and Mataric, 2002]. The high-level layer is responsible for system modeling and planning and the low-level layer is in charge of sensing and execution. The second type of robotic architecture is called behavior-based architecture [Sridhar and Connell, 1992; Laue and Rfer, 2004; Aravintan and Nanayakkara, 2004]. The complicated robotic behavior is obtained by assembling some more solvable and simpler robotic behavior components. Another robotic architecture is the hybrid architecture [Connell, 1992; Secchi, et al., 1999]. It is the hybrid of layered organization and behavior-based decomposition of the execution layer. Behavior-based robotic system design is a commonly employed approach in building autonomous robot navigation systems. We need to enable the robot to autonomously coordinate behaviors in order to exhibit the complicated robotic behavior, e.g., coordinating target-tracking and anti-collision behaviors in order to reach a target position while avoiding unforeseen and moving obstacles during its motion. It offers a collection of fundamental robotic behaviors and the behavior is selected according to its response to a certain stimulus in the real situations. In behavior-based robotics, basic robotic behaviors are elementary building blocks for robot control, reasoning, and learning. The environment plays a central role in activating a certain basic robotic behavior throughout the execution of robotic systems. In general, both the robotic behavior modules and the coordination mechanisms are designed through a trial-and-error procedure. By doing so, the designer gradually adjusts them and examines corresponding behaviors until the satisfactory robotic behavior is derived. Inspired by the Darwinian thinking, evolutionary algorithms [Rechenberg, 1973; Holland, 1975] are being applied to the robotic control field [Gomi

and Griffith, 1996; Meyer, Husbands, and Harvey, 1998]. In this discipline, control algorithms inspired largely by biological evolution are used to automatically design sensorimotor control systems. Evolutionary robotics (ER) is a novel and active domain, where researchers apply the evolution mechanism to the robot design and control. The fundamental principles of evolution mechanism include a population of individuals competing for survival, inheritance of advantageous genes from the parent generation and the offspring variability possibly result in a higher chance of survival. ER enables robotic system to adapt to unknown or dynamic environments without human intervention. The robotic controllers derived by evolutionary methods have advantages of relatively fast adaptation time and carefree operations [Floreano and Mondada, 1996; Lund and Hallam, 1997; Nolfi and Floreano, 2000]. In evolutionary robotics, an initial population of chromosomes is randomly generated, where each chromosome encodes the controller or morphology of a robot. Each physical or simulated robot then acts guided by the robotic controller, and in the evolutionary process, the robotic performance in various situations is evaluated. The fittest robots have more chances to generate copies of their genotypes using genetic operators such as reproduction, crossover, and mutations. This process is iterated until an individual with satisfactory performance is attained. That is, artificial evolution starts with a population of strings or genotypes, which represents the control parameters of a robot. Each robot is evaluated in the problem domain and given a fitness value. Following the inspiration from Darwinian evolution, the fittest genotypes have a better chance to breed, which usually includes crossing two parents to produce a child and altering or mutating random values. As the evolution process proceeds, mean fitness within the task domain increases. Thus, evolutionary robotics is completely different from the conventional robotic controller design, where the desired robotic behavior is decomposed into some simpler basic behaviors. Under the framework of the evolutionary algorithm-based robotic controller, the sensory inputs are encoded into the genetic strings and sent to the evolutionary algorithm based robotic controller. The robotic controller makes appropriate decisions based on the real-time sensory values and sends its decision to the execution devices such as motors and actuators. Most commonly, the sensory signals are encoded into a binary string, where each bit represents corresponding sensory information. For instance, in the binary string, the first bit may indicate the status of the frontal sensor value of the mobile robot. “1” indicates it detects a front obstacle and “0” tells that there is no obstacle in front of the robot. This encoding mechanism

can also be applied to the encoding of behavioral block. Each binary bit indicate whether or not its corresponding basic behavioral block is used in responding to the current real-world situations. The evolutionary algorithm performs a parallel search in the space of Boolean functions, searching for a set of functions that exhibits the required robotic behaviors in the real-world environment. It generates new Boolean functions by transforming and combining the existing functions using various operators in a genetically inspired way. Statistics related to their performance are used to steer the search by choosing appropriate functions for transformation and combination. For instance, a standard genetic algorithm with Roulette-wheel selection, single-point crossover, and random mutation can be used in the robotic controller evolution. When all the individuals of a population have been examined, the genetic operators are used to produce a new generation of individuals. Roulette-wheel selection includes a linear scaling of the fitness values and a probabilistic assignment for each offspring individual according to its fitness. All offspring are then randomly paired and a random single-point crossover is performed based on the pre-specified crossover probability. Then the newly obtained strings are mutated segment by segment with the assigned mutation probability. From the robotic point of view, the mutation operator is able to introduce some new robot motions for sensory input states and find appropriate motions. The mutation rate must be carefully chosen and a tradeoff between evolution stability and convergence should be reasonably made. For instance, if the mutation rate is too high, it will incur genetic noise and may even cause evolution instability. If it is too low, the robot may take a much longer time to acquire the expected behaviors. In general, the mutation rate used is slightly higher than the inverse of the length of the chromosomes. In the overall genetic evolution process, the elitist strategy is quite useful in the selective reproduction operation, which ensures that the best chromosome is reserved in the next generation. The elite strategy prevents the best chromosomes in each generation in case that the experimenter is not sure whether or not the specified mutation rate consistently brings noise into the evolution process.

Other main techniques for autonomous robotics include artificial life [Brooks, 1992], reinforcement learning [Tan, et al., 2002b], artificial immune system [Fukuda, et al., 1999], and plan-based control [Beetz, 2002; Beetz, et al., 2002], and so on. In the remainder of this chapter, we primarily discuss the evolutionary robotics and its related issues.

1.3 Adaptive Autonomous Robot Navigation

The perception-action cycles are too complicated to be systematically formulated using a unified theory. A perception-action cycle usually involves a large amount of highly interacting components (i.e., interrelated subsystems), that continuously interact with each other during the perception-action cycle. The interactions for generating the aggregate behavior are highly nonlinear, so that the aggregate behavior cannot be derived easily from the behaviors of isolated components using traditional mathematical formulism. The components in a perception-action cycle are highly diverse and their behaviors are closely associated with each other. For instance, malfunction of one component may cause a series of changes and adjustments on other components. And it is possible to compensate for the damaging effect caused by the failed component during the reception-action cycle via self-repairing capability. This type of self-reconfiguration in the reception-action process is not easy to be clearly represented by classical mathematical approaches. Furthermore, the internal models of components, the large amount of subtly interwoven information processing components, and their high inherent nonlinearities, make traditional approaches incapable of formulating the complex system dynamics systematically. Perception-action cycle will still remain an open problem unless we are able to take all of the aforementioned factors into consideration simultaneously. As a result, there is still no a unified theory for the perception-action cycle at the moment.

Generally speaking, the approaches to adaptive autonomous robotic navigation can be classified into two categories [Jakobi, 1997]. The first is to evolve the system in a pre-designed environment where various sensor readings needed for training the desired robot behavior are intentionally designed (a.k.a. off-line evolution) normally via a simulator [Seth, 1998]. The fitness evaluation can be performed by simulation where different controllers can be tested under the same environment conditions. A simulation is usually cheaper and faster than a real robot. The individuals might exploit simplifications or artifacts in the model. However, the evolved robotic controller might not work reliably on the real robot due to the simplified environment model used. The second scheme is to conduct evolution during the operational phase so that the robot behavior can be gradually improved by continuous learning from the external environment (a.k.a. on-line evolution) [Yao and Higuchi, 1999]. The fitness is evaluated using the real robot in the real world where fitness evaluation is a stochastic process. In

addition to the robot learning system, a changing environment must be taken into consideration. In addition, the robot may encounter unexpected situations which do not appear in simulations. Therefore, it is expected that the evolved robotic controller is more robust than that obtained by simulation.

However, both approaches are not well established as of today and there are still many unsolved problems. For instance, the adaptive behavior acquired from the first method is not strong enough to handle the unpredictable human environment. In essence, such behaviors can only work well in the predictable environment. Once a trained robot encounters an environment that it has not seen in its training phase, the robot may get stuck and take undesired actions. In real-world applications, it is not possible to take all the feasible events into account during the evolution process. Or even assuming it were possible, the training cost would be extremely high. The second method would require certain new opportunities for improving the robot behavior during the operational stage. In this method, the environment intended for the robot training should be carefully designed so that there are complete feasibilities for the new behaviors to be tried out. However, one of the main drawbacks of this training mode is that it is extremely time-consuming. The time taken to obtain a target controller in on-line evolution is primarily determined by fitness evaluation process for a particular task. It is difficult to evaluate individuals on-line due to the large number of interactions with the physical environments. In particular, since actions to be taken in the current sensor input state affect the future sensor state, it is difficult to present regular sensor input patterns to each variant controller in an efficient manner. If the problem cannot be well resolved, the fitness function approach applied to off-line hardware evolution cannot be used in the on-line evolution situations.

For instance, in the hardware based evolution system [Tan, et al., 2002a], various methodologies were investigated in the early stage of the study to find out the most suitable approach for the hardware evolution system, including the method of on-line intrinsic evolution. The obstacle blocks are densely distributed in the enclosed field so that the robot has higher chances to interact with the obstacles and different sensor input states could be presented to the robot in an efficient manner. For each chromosome, the reflected infrared is measured twice, i.e., one for sensor input states and the other for sensor-based fitness evaluations. Instead of assigning fitness value based on Field Programmable Gate Array (FPGA) outputs, the robot moves in the physical environment for 0.3 second and evaluates

the chromosome based on the new sensor states. The desired behaviors lead to a state where the robot faces fewer obstacles than in the previous state. The difference in the sensor states is used to calculate the fitness score for each chromosome. However, the on-line fitness evaluation turned out to be impractical because too much time is required for the evaluation process. Therefore, on-line evolution considerably decreases the efficiency of the evolutionary approach, due to the need of a huge number of interactions with the environment for learning an effective behavior.

Two physical experiments were conducted to find solutions to this dilemma. In the first experiment, a random motion was carried out after each fitness evaluation via the FPGA output values (the wheel speeds). Such a random motion may generate a new world state, which is different from previous world states for the current chromosome. In the evolution, six classified input states need to be shown to the robot requesting for its response, and the fitness evaluation is then carried out. Therefore, many random motions are needed to search for different input states for each chromosome, and a lot of time is consumed for such an exhaustive state search that causes the evolution to be unpredictable.

The second experiment is similar to the first one in principle, but adopts a different strategy to search for the input states. In this approach, instead of looking for all the defined input states for consecutive chromosomes, algorithm is applied to search for the defined input states for all chromosomes at each generation. Once duplicated input state is presented to the robot, the next chromosome that has not encountered the current input state will take over the current chromosome for fitness evaluation. Such an approach seems to evaluate all the chromosomes in each generation at a much faster speed than the previous method. However, it is still not capable of coping with the massive data needed to process in our application, and a lot of time is wasted in searching for the sensor input states rather than in the intrinsic evolution.

As seen from the above attempts to conduct online evolutions, it is hard for the robot to achieve the adaptation in the operational phase, i.e., online evolution, though this method exhibits the real idea about adaptive robot evolutions.

In autonomous robot navigation applications, the reinforcement learning mechanism [Mitchell, 1997; Sutton, 1998; Sutton and Barto, 1998] is very often used for path planning (e.g., to figure out the shortest route to the designated destination via unsupervised learning). In doing so, the robot is capable of planning navigation path based on the acquired sensor

values coupled with certain appropriate self-learning algorithms instead of a predefined computational model, which is normally not adaptive. In reinforcement learning, situations are mapped to actions based on the reward mechanism. The autonomous robotic system takes an action after receiving the updated information collected from the environment. It then receives a reward as an immediate payoff. The system purposefully maintains the outputs that are able to maximize the received reward over time. In doing so, the cumulative reward is maximized. The four basic components in reinforcement learning are the policy, reward, value, and model. According to Sutton [Sutton and Barto, 1998], policy is the control strategy which determines how the agent chooses the appropriate actions to achieve its goals. Reward reflects the intrinsic desirability of the state and the reward accumulated over time should be maximized. The reward mechanism can be provided internally or externally. The state value is defined as the total reward it receives from that state onwards, which indicates what types of actions are desired in the long term. Finally, the real-world environment can always be seen as the collection of uncertain events. A predefined model is not capable of adapting to an unpredictable environment. Therefore, based on the real-time sensory values, the robot should be able to maintain and update the world model autonomously and adaptively during its movement. In the learning process of an autonomous mobile robot, it judges if it is desirable to take a certain action in a given state using trial and error search and delayed reward. Reinforcement learning enables an autonomous mobile robot to sense and act in its environment to select the optimal actions based on its self-learning mechanism. Two credit assignment problems should be addressed at the same time in reinforcement learning algorithms, i.e., structural and temporal assignment problems. The autonomous mobile robot should explore various combinations of state-action patterns to resolve these problems.

1.4 Artificial Evolution in Robot Navigation

It has shown in [Nolfi, 1998a] that the robot behaviors could be achieved by the simpler and more robust evolutionary approaches than the traditional decomposition/integration approach. This section gives an overall introduction of the artificial evolution mechanism. It presents the main strategies for robotic controller design. Various applications of artificial evolution in robotics are surveyed and classified. Furthermore, in this sec-

tion their specific merits and drawbacks in robotic controller design are discussed. At present, there is little consensus among researchers as to the most appropriate artificial evolution approach for heterogeneous evolutionary systems. It should be noted that the section only gives some basic ideas on what artificial evolution is due to the space restriction and the chapter theme. For more detailed information about these topics, readers are suggested to refer to specific references such as [Goldberg, 1989; Fogel, 2000a, 2000b; Driankov and Saffiotti, 2001; Tan, et al., 2001a, 2001b].

1.4.1 *Neural Networks*

Many evolutionary approaches have been applied to the field of evolvable robotic controller design in the recent decades [Meyer, 1998; Chocron and Bidaud, 1999; Pollack et al., 2000]. Some researchers used artificial Neural Networks (NN) as the basic building blocks for the control system due to their smooth search space. NNs can be envisaged as simple nodes connected together by directional interconnects along which signals flow. The nodes perform an input-output mapping that is usually some sort of sigmoid function. An artificial NN is a collection of neurons connected by weighted links used to transmit signals. Input and output neurons exchange information with the external environment by receiving and broadcasting signals. In essence, a neural network can be regarded as a parallel computational control system since signals in it travel independently on weighted channels and neuron states can be updated in parallel. NN advantages include its learning and adaptation through efficient knowledge acquisition, domain free modeling, robustness to noise, and fault tolerance, etc. [Huang, 2002]. The behaviors that evolutionary robotics is concerned with at present are low-level behaviors, tightly coupled with the environment through simple, precise feedback loops. Neural networks are suitable for this kind of applications so that the predominant class of systems for generating adaptive behaviors adopts neural networks [Jakobi, 1998]. The same encoding schemes can be used independently of the specific autonomous robot navigation system since different types of functions can be achieved with the same type of network structure by varying the properties and parameters of simple processing used. Other adaptive processes such as supervised and unsupervised learning can also be incorporated into NN to speed up the evolution process.

NNs have been widely used in the evolutionary robotics due to the aforementioned merits. For instance, locomotion-control module based on

recurrent neural networks has been studied by Beer and Gallagher [1992] for an insect-like agent. Parisi, Nolfi, and Cecconi [1992] developed back propagation neural networks for agents collecting food in a simple cellular world. Cliff, Harvey, and Husbands [1993] have integrated the incremental evolution into arbitrary recurrent neural networks for robotic controller design. Floreano and Mondada [1996] presented an evolution system of a discrete-time recurrent neural network to create an emergent homing behavior. Using a dynamic evolution system, Steels [1995] proposed a coupled map latticed to control a robot. Lund and Miglino [1996] evolved neural network control system for the Khepera robot using a simulator. A modified back-propagation algorithm for development of autonomous robots was proposed by Murase et al. [1998] to control the real Khepera robot. Smith [1998] used feed-forward neural network to teach a robot to play football. A two stage incremental approach was used to simulate the evolution of neural controllers for robust obstacle avoidance in a Khepera robot by Chavas et al. [1998]. By analyzing the fitness landscape of neural network which controls a mobile robot, Hoshino, Mitsumoto, and Nagano [1998] have studied the issue on the loss of robustness of evolved controllers. In [Ishiguro, Tokura, and Kondo, et al., 1999], the concept of dynamic rearrangement function of biological neural networks is incorporated with the use of neuromodulators to reduce the robotic controller performance loss between the simulated and real environments. Hornby et al. [2000] built a recurrent neural network based robotic simulator that runs at over 11000 times faster than real time and used it to evolve a robust ball-chasing behavior successfully. Spiking neural controllers were evolved for a vision-based mobile robot [Floreano and Mattiussi, 2001]. Reil and Husbands [2002] used recurrent neural networks to deal with the control problem of bipedal walking.

However, neural networks also have certain drawbacks. For instance, a NN cannot explain its results explicitly and its training is usually time-consuming. Furthermore, the learning algorithm may not be able to guarantee the convergence to an optimal solution [Huang, 2002].

1.4.2 *Evolutionary Algorithms*

There are currently several flavors of evolutionary algorithms (EAs). Genetic Algorithms (GAs) [Holland, 1975] is the most commonly used one where genotypes typically are strings of binary. Genetic Programming (GP) [Koza, 1994] is an offshoot of GAs, where genotypes are normally computer programs. Other flavors such as Evolution Strategies (ES) are also used in

ER. Many concerns are shared among these approaches.

In principle, GA is a simple iterative procedure that consists of a constant-size population of individuals, each one represented by a finite string of symbols, known as the genome, encoding a possible solution in a given search space, which consists of all possible solutions to the target problem. Generally, the genetic algorithm is applied to space which is too large to be exhaustively searched. The symbol alphabet can be binary encodings, character-based encodings, real-valued encodings, and tree representations. The standard GA proceeds as follows: an initial population of individuals is created randomly or heuristically. At every evolutionary step (i.e., a generation), the individuals in the current population are decoded and evaluated based on some pre-specified fitness criterion (i.e., fitness function). To generate a new population, individuals are chosen according to their fitness. Many selection procedures can be used in deriving robotic controller. For instance, the fitness-proportionate selection is the simplest one, where individuals are selected based on their relative fitness. By doing so, the individuals with high fitness stand a better chance to reproduce while the ones with low fitness are prone to die.

The majority of GA work is search problems in a fixed-dimensional search space. The well-defined finite dimensionality of the search space allows a choice of genotype coding with fixed length. Typically, the GA starts with a population of random points effectively spanning the entire search space. Successive rounds of genetic operations lead to the optimum or near optimal population. In autonomous robot navigation, we assumed that the desired robot behaviors such as anti-collision could be accomplished with the evolved optimal chromosome. However, this assumption may not be true since we could not obtain the complete anti-collision behavior using the best chromosome achieved from evolution. The best solution may lie outside the boundaries that we defined in the experiments. Therefore, other new approaches need to be adopted in order to achieve complete obstacle avoidance behavior. There are two alternatives for solving this problem: incremental learning and species adaptation genetic algorithm (SAGA). Incremental evolution is able to derive a sequence of increasingly complex tasks. This is particularly useful when trying to resolve a complex and difficult problem. Genotype lengths are allowed to change in SAGA, which is discussed in the next section.

As a commonly used EA, GA has also been used in [Husbands, Harvey, and Cliff, 1995; Floreano and Mondada, 1996] for generating robotic behaviors. Based on GA techniques, Yamada [1998] trained the robot to

recognize uncertain environments. Gaits of a lagged robot were derived by the GA in [Gomi and Ide, 1998]. Barfoot and D'Eleuterio [1999] used GA to implement the multi-agent heap formation. Using perception-based GA, Kubota et al. [1999] realized collision avoidance of the mobile robot in a dynamic environment. Chaiyaratana and Zalzal [1999] presented the use of neural networks and GAs to time-optimal control of a closed-loop robotic system. Earon et al. [2000] developed walking gaits based on cellular automation for a legged robot where GA was used to search for cellular automata whose arbitration results in successful walking gaits. Inspired by the prior work using GA, Thompson [1995] adopts the conventional GA as the training tool to derive the robot controllers in the hardware level. The encouraging experimental results justify the effectiveness of GA as a robust search algorithm even in hardware evolution.

Most applications nowadays use the orthodox GA, however, Species Adaptation GAs (SAGA) proposed by [Harvey, 1992, 2001] would be more suitable for certain robot evolution applications such as evolvable hardware based robotic evolutions. In SAGA, different structures are encoded with genotypes of different lengths, which offers a search space of open-ended dimensionality. Cyclic Genetic Algorithm (CGA) was also introduced in [Parker, 1999] to evolve robotic controllers for cyclic behaviors. Also distributed genetic algorithms are introduced into the evolutionary robotics field these years. For instance, in the spatially distributed GA, for each iteration a robot is randomly selected from a population distributed across a square grid. The robot is bred with one of its fittest neighbors and their offspring replaces one of the least fit neighbors such that the selection pressure keeps successful genes in the population. The distributed GA is usually robust and efficient in evolving capable robots. GA exhibits its advantages in deriving robust robotic behavior in conditions where large numbers of constraints and/or huge amounts of training data are required [Walker and Oliver, 1997]. Furthermore, GA can be applied to a variety of research communities due to its gene representation. However, GA is computationally expensive [Walker and Oliver, 1997]. Though GA is now being widely used in ER field, a variety of issues are still keeping open in the GA-based ER. For instance, the fitness function design is an important issue in GA-based evolution schemes [Revello and McCartney, 2000]. The fitness function should be its measurement of its ability to perform under all of the operating conditions. In principle, all these objectives can be fulfilled by setting an appropriate fitness function so as to derive the desired robotic performance exhibited during autonomous navigation. Therefore, the fitness function

design needs to be investigated more carefully to make the robot evolve in a more effective way. Several experiments have also been performed where the robotic controllers were evolved through Genetic Programming (GP) [Koza, 1994; Dittich, Burgel, and Banzhaf, 1998]. Brooks [1992] has outlined a GP-based evolutionary approach applied to lisp-like programming languages. Reynolds [1994] used GP to create robot controllers that enabled a simple simulated vehicle to avoid obstacles. Nordin and Banzhaf [1997] reported experiments using GP to control a real robot trained in real environments with actual sensors. Later they proposed a new technique allowing learning from past experiences that are stored in memory. The new method shows its advantage when perfect behavior emerges in experiments quickly and reliably [Nordine, Banzhaf, and Brameier, 1998]. Liu, Pok, and Keung [1999] implemented a dual agent system capable of learning eye-body-coordinated maneuvers in playing a sumo contest using GP techniques. An example-based approach to evolve robot controllers using GP was implemented in [Lee and Lai, 1999]. A GP paradigm using Lindenmayer system re-writing grammars was proposed as a mean of specifying robot behaviors in the uncertain environment [Schaefer, 1999]. Kikuchi, Hara, and Kobayashi [2001] implemented a robotic system featuring reconfigurable morphology and intelligence using GP.

1.4.3 *Fuzzy Logic*

As mentioned earlier, fuzzy logic provides a flexible means to model the nonlinear relationship between input information and control output [Hoffmann, Koo, and Shakernia, 1998]. It incorporates heuristic control knowledge in the form of if-then rules, and is a convenient alternative when the system to be controlled cannot be precisely modeled [Paraskevopoulos, 1996; Driankov and Saffiotti, 2001]. They have also shown a good degree of robustness in face of large variability and uncertainty in the parameters. These characteristics make fuzzy control particularly suited to the needs of autonomous robot navigation [Saffiotti, 1997]. Fuzzy logic has remarkable features that are particularly attractive to the hard problems posed by autonomous robot navigation. It allows us to model uncertainty and imprecision, to build robust controllers based on the heuristic and qualitative models, and to combine symbolic reasoning and numeric computation. Thus, fuzzy logic is an effective tool to represent real world environments. In evolutionary robotics, fuzzy logic has been used to design sensor interpretation systems since it is good at describing uncertain and imprecise

information.

All the specific methods have their own strengths and drawbacks. Actually they are deeply connected to one another and in many applications some of them were combined together to derive the desired robotic controller in the most effective and efficient manner. For instance, Fuzzy-genetic system [Hagras, Callaghan, and Colley, 2001] is a typical evolution mechanism in evolving adaptive robot controller. Arsene and Zalzal [1999] controlled the autonomous robots by using fuzzy logic controllers tuned by GA. Pratihari, Deb, and Ghosh [1999] used fuzzy-GA to find obstacle-free paths for a mobile robot. Driscoll and Peters II [2000] implemented a robotic evolution platform supporting both GA and NN, Xiao, et al. [2002] designed autonomous robotic controller using DNA coded GA for fuzzy logic optimization.

1.4.4 *Other Methods*

Apart for the above commonly used methodologies, several other evolutionary approaches were also tried out in the ER field in recent years. For example, classifier systems have been used as an evolution mechanism to shape the robotic controllers [Dorigo and Schnepf, 1993; Dorigo and Colombetti, 1994]. Grefenstette and Schultz used the SAMUEL classifier system to evolve anti-collision navigation [Grefenstette, 1989; Grefenstette and Schultz, 1994]. Katagami and Yamada [2000] proposed a learning method based on interactive classifier system for mobile robots which acquires autonomous behaviors from the interaction experiences with a human. Gruau and Quatramaran [1997] evolved robotic controllers for walking in the OCT-1 robot using cellular encoding. In the work of Berlanga et al. [1999], the ES is adopted to learn high-performance reactive behavior for navigation and collisions avoidance. Embodied evolution was proposed as a methodology for the automatic design of robotic controllers [Watson, Ficici, and Pollack, 1999], which avoids the pitfalls of the simulate-and-transfer method.

Most of the ER approaches aforementioned are essentially software-based. More recently, hardware-based robotic controllers using artificial evolution as training tools are also being exploited. The development of evolvable hardware (EHW) has attracted much attention from the ER domain, which is a new set of integrated circuits able to reconfigure their architectures using artificial evolution techniques unlimited times. Higuchi, Iba, and Manderick [1994] used off-line model-free and on-line model-based methods to derive robot controllers on the logic programmable device. At-

tempting to exploit the intrinsic properties of the hardware, Thompson [1995] used a Dynamic State Machine (DSM) to control a Khepera robot to avoid obstacles in a simple environment. The hardware evolution issues will be detailed in the next section.

1.5 Open Issues and Future Prospects

Although a variety of successes have been achieved in ER, there are still a bunch of issues to be addressed and there also exists plenty of room to further explore the relevant research areas. For instance, Lund, Miglino, and Pagliarini, et al. [1998] intended to make evolutionary robotics an easy game, and Mondada, and Legon [2001] explored interactions between art and mobile robotic system engineering. In this section, open issues as well as future prospects for evolutionary robotics are presented.

1.5.1 SAGA

As discussed earlier, it is highly difficult to develop a fitness evaluation strategy in standard GA to progressively improve the robot behavior. The EAs such as conventional GA and GP are not suitable for such incremental evolutions. SAGA [Harvey, 1992, 2001] extends the traditional GA to attack this issue, where mutation is the primary genetic operator and crossover does not play the driving force as it does in conventional GAs. SAGA is more appropriate than orthodox GA for certain ER applications such as hardware-based robot evolution.

As mentioned in the earlier sections, it would be difficult to develop a fitness evaluation technique that allows a progressive path of improvements starting from an initial random population and finally reaching a complex target behavior. The EAs such as conventional GA and GP are not suitable for this long-term open-ended incremental evolution of complex systems. A slightly more complex problem might not be solved properly using the same experimental parameters since it requires several more generations. The final population suited for the original problem could not be used as the starting population of the next complex task to evolve in an incremental manner. Harvey's SAGA [1992] was developed as an extension of the GA to cope with this issue. In SAGA, mutation is the primary genetic operator while crossover is not as crucial as it is in conventional GAs in the evolution process.

SAGA should be more appropriate than standard GA for the autonomous robotic application. There is no pre-specified chromosome length in the structure to be designed in autonomous robot navigation. In SAGA, different structures are encoded with genotypes of different lengths. This makes the evolutionary search operate in the space of varying dimensionality. Progress through such a genotype space is accomplished by adjusting the genotype length. Meanwhile, it should be noted that except for the genotype length, many other factors should also be carefully considered to attain the optimal string configuration for the desired robotic performance.

1.5.2 *Combination of Evolution and Learning*

Evolution and learning are two forms of biological adaptation that differ in space and time. Evolution is a strategy for a population of robots (physical or simulated) to adapt to the environment through genetic operations in a distributed population of individuals. However, learning is an adaptation strategy for a robot to adapt to its environment by conducting a set of modifications in each individual during its own lifetime. Evolution is a form of adaptation capable of capturing relatively slow environmental changes that might encompass several generations. Learning enables an individual to adapt to dynamic and unpredictable environmental changes at the generation level. Learning includes a variety of adaptation mechanisms, which are able to produce adaptive changes in an individual level during its lifetime. Learning and evolution are highly related to each other. Learning allows individuals to adapt to changes in the task environment that occur in the lifespan of an individual or across few generations and enables evolution to use information extracted from the environment thereby channeling evolutionary search. It is a replenishment of evolution and can help and guide evolution. However, it may involve high computational cost. For instance, sometimes robots based on reinforcement learning can be very slow in adapting to environmental changes and thereby may incur delays in the ability to acquire fitness and even system unreliability. For instance, reinforcement learning allows an autonomous robot that has no knowledge of a task or an environment to learn its behavior by gradually improving its performance based on given rewards in performing the learning task.

1.5.3 *Inherent Fault Tolerance*

It is also observed that not all device properties are constant over time as many factors may change as time goes on during system operations, which may incur the internal failure of the robot. A practical and useful robot controller should be able to operate properly in face of temperature changes and supply voltage fluctuations. Also it should be able to work in the presence of medium instability. A fit robotic behavior must be capable of handling the full variety of conditions that it may encounter in the practical environment. In the robotic applications, the entire autonomous robot can be viewed as an evolvable system since it is able to adjust its own structure and strategy. In light of the variations in operating conditions, the robot should have the ability to perform the desired tasks as before.

Meanwhile, the ability to embrace the inevitability of the external robot failure is also crucial to guarantee proper autonomous robot navigation in harsh environments. The evolved robot should also be able to detect any fault that occurs at any time in the sensor-motor system and respond fast enough to cover up the fault. That is also the purpose of implementing the on-line evolution in ER evolutions. Embedding fine-grained Micro-Electro-Mechanical Systems (MEMS) components into robotic systems can bring stronger processing capacities. However, sensor and actuator failures may often lead to performance deterioration or dysfunction of robotic systems. In most cases, it is not known when and how much a component fails. To guarantee the proper operations of the overall robotic system, the remaining components should be reconstructed to compensate for the damaging effects caused by the failed components in a timely manner. As a result, an effective fault tolerance based robotic controller for such component failure compensation in a real-time fashion is highly desirable.

Another issue in fault tolerance based autonomous robot navigation is the sensor differences even in the same robot. Even the physical sensors and actuators from the same batch may perform differently as they may be slightly different in their electronics or mechanics. This should also be taken into account in the fault-tolerant control design.

One approach to fault tolerance in the evolutionary robotics is morphology evolution [Mautner and Belew, 1999a; 1999b]. Except for the robotic controller evolution, it is also possible to evolve robots whose size and morphology are under the control of an evolutionary process. Sometimes, morphology can play an essential role since certain tasks may become much easier to perform for robots with certain particular shapes. Mostly im-

portantly, morphology evolution provides the robot with the self-repairing capability such that the remaining components can be reconstructed via the evolutionary process. Thus the robotic system becomes more robust and reliable in various harsh conditions.

1.5.4 *Hardware Evolution*

The emerging application of artificial evolution to autonomous mobile robots enables robots to adapt their behaviors to changes of the environments in a continuous and autonomous fashion. For example, personal robots used for entertainment, education, and assistance interact with humans and service robots are being made to carry out work in unstructured environments [Keramas, 2000]. Meanwhile, Evolvable Hardware (EHW) has attracted much attention [Yao and Higuchi, 1999], which is a set of integrated circuits capable of adapting their hardware autonomously and in real time with a changing environment. Consequently, we have seen the fusion of these two technologies in these years. Artificial evolution can operate on reconfigurable electronic circuits to produce efficient and powerful control systems for autonomous mobile robots which are able to make their own decisions in complex and dynamic environment [Tan, et al., 2004]. Reconfigurable electronic circuit such as Field Programmable Gate Array (FPGA) can be viewed as a complex multi-input multi-output digital device, which is made up of a large number of Boolean functions. The input variables defined by the sensors on vehicle are fed into the FPGA and the output variables from the device are encoded into the different motor speeds. The multi-input multi-output function implemented on the FPGA should provide the desired reactive behaviors in response to the new changes in the environment.

Therefore, the evolution task can be clearly viewed as a searching process for the best system configuration whenever the environment changes. Intrinsic evolvable hardware best suits this application. Thompson used a reconfigurable hardware device (FPGA) to evolve a binary frequency discriminator that classifies its input signal into one of two frequencies [Thompson, Layzell, and Zebulum, 1999]. His search was GA-based and obtained its fitness measures in real time from the FPGA. Thompson identified the phenomenon of evolution exploiting the device's analog nature in finding solution circuits. The tone-discriminator experiment of Thompson clearly demonstrated intrinsic evolution's ability to explore rich structures and dynamical behaviors that are obviously radically different to those pro-

duced by conventional design, but yet which achieve the desired behavior perfectly. Evolving physical hardware directly instead of control systems simulated in software results in both higher speed and richer dynamics.

1.5.5 *On-Line Evolution*

One of the most urgent concerns is how evolved controllers can be efficiently evaluated. If they are tested using physical robots in the real world, then this should be conducted in real time. However, the evolution of complex behavior will take a prohibitively long time to evolve the desired behavior. If controllers are tested using simulation, then the amount of modeling needed to ensure that evolved controllers work on real robots may make the simulation too complex and computationally expensive. As a result, the potential speed advantage over the real-world evaluation is lost.

Up until now, several possible solutions have been proposed to solve the current problems in the evolution of adaptation tasks. For instance, Jakobi [1997] proposed the methodology based on minimal simulations which has been corroborated by a variety of experiments. The reliable transition from simulation to reality without performance loss is turned out to be feasible by modeling only a small subset of the robot/environmental properties [Jakobi, 1998; Jakobi and Quinn, 1998; Husbands, 1998; Smith, 1998]. However, as discussed previously, future work should pay more attention to the on-line learning schemes since they are more promising and useful in real-world applications in complex and dynamic environments to handle uncertainties. In the long sight, this might bring great advantages by accomplishing missions beyond human abilities. Grefenstette [1996] proposed that while operating in a real environment, an agent should maintain a simulated environmental model updated continuously whenever new features occur from the environment and the agent itself such that the learning continues throughout the operational phase. The work of Thompson [1995a] and Keymeulen et al. [1999] has already shown that the reliable transfer of behavior from simulation to reality is feasible without loss of robotic behaviors, though some issues still remain open. These ideas would definitely open up new opportunities in on-line robot evolution. Keymeulen et al. [1999] evaluated two approaches to the implementation of EHW to a robot navigation system: off-line model-free and on-line model-based evolution.

1.5.6 *Ubiquitous and Collective Robots*

In the recent decades, the fusion of information processing with physical processes has significantly changed the physical world around us and it also represents one of the most important opportunities for ER. The wide use of novel technologies such as System On a Chip (SOC), smart Microelectromechanical Systems (MEMS) transducers, Commercial-Off-The-Shelf (COTS) components, Internet connectivity, and high-dependability systems have brought great revolutions to the traditional real-time control systems [Kopetz, 2000; Prahlad, et al., 2002]. For example, inexpensive MEMS-based sensors and actuators have made it fairly feasible to integrate the physical world and information processing systems in various domains. Sensor networks are one of the most crucial applications of such embedded and distributed systems. Since wireless communication is used to reduce uncertainty in robotics [Sukhatme and Mataric, 2000] and ER is an effective way to deal with uncertainties, the fusion of these two innovative technologies is a natural and intuitive solution to handling uncertainty more efficiently. By introducing ER into distributed and embedded systems, there are many promising research fields ahead such as distributed and embedded robotic system based on wireless networks [Sukhatme and Mataric, 2000]. Here the robotics is used for network communication through physical mobility [Sukhatme and Mataric, 2000]. It can also be viewed as an application of ubiquitous computing (ubicom) concept [Weiser, 1993] in robotics and the integration of distributed ER and embedded real-time system would incur new compelling and interesting topics to be resolved for this research community.

The distributed ER is also closely related to collective robotics. The term collective behavior generically refers to any behavior of a system having more than one robot. The use of multiple robots is often suggested to have many advantages over single-robot systems. Cooperating robots have the potential to accomplish the desired tasks more efficiently than a single robot [Xu, et al., 2002]. Furthermore, using several low-cost robots introduces redundancy and therefore is more fault-tolerant than having only one powerful and expensive robot. Therefore, collective behaviors offer the possibility of enhanced task performance, increased task reliability, and decreased cost over traditional robotic systems. In developing a multi-robot system, one of the primary concerns is how to enable individual robots to automatically generate task-handling behaviors adaptive to the dynamic changes in their task environments. The interactions among robots impli-

cate development since they are not obvious in a particular robot but only emerge in an operating group.

1.6 Summary

Free-navigating mobile robotic systems can be used to perform service tasks for a variety of applications such as transport, surveillance, fire fighting, and so forth. For such robotic application systems, it is crucial to derive simple robotic behaviors that guarantee robust operation despite of the limited knowledge prior to system execution, e.g., designing an anti-collision behavior that is effective in the presence of unknown obstacle shapes. In recent years, autonomous mobile service robots have been introduced into various non-industrial application domains including entertainment, security, surveillance, and healthcare. They can carry out the cumbersome work due to their high availability, fast task execution, and cost-effectiveness. An autonomous mobile robot is essentially a computational system that acquires and analyzes sensory data or exterior stimulus and executes behaviors that may affect the external environment. It decides independently how to associate sensory data with its behaviors to achieve certain objectives. Such an autonomous system is able to handle uncertain problems as well as dynamically changing situations. Evolutionary robotics turns out to be an effective approach to realizing this purpose. The main theme of this chapter is to look into the applications of evolutionary approach in autonomous robotics. A general survey is reported regarding the effectiveness of a variety of artificial evolution based strategies in robotics. The open issues and future prospects in evolutionary robotics are also discussed. Some questions need to be answered if evolutionary robotics is to progress beyond the proof-of-concept stage. Furthermore, future prospects including SAGA, combination of learning and evolution, inherent fault tolerance, hardware evolution, on-line evolution, and ubiquitous and collective robots are suggested. With the infusion of more and more innovative ideas and technologies, we can see the bright future in this field, though there is a long way to go.