2015 IEEE International Conference on Systems, Man, and Cybernetics

Applying an Extension of Estimation of Distribution Algorithm (EDA) for Mobile Robots to Learn Motion Patterns from Demonstration

Huan Tan
Software Sciences and Analytics
GE Global Research
Niskayuna, USA
huantan@ieee.org

Abstract— **This paper proposes a probabilistic evolutionary computing algorithm for robots to learn motion patterns. This algorithm is inspired from Estimation of Distribution Algorithms (EDA). The distribution of chromosomes (not the genes), which have higher fitness values in the configuration space, is estimated in a configuration space. A modified Probabilistic Rapidly-growing Random Tree (PRRT)-Connect algorithm is used for searching the configuration space to generate chromosomes which are represented as paths from the starting point to the goal point. Mutation is defined as searching with certain probability outside of the current distribution area (obstacle-free area). This algorithm is applied for robotic learning of motion trajectories through imitation. Simulation and practical experimental results are given in this paper to verify the effectiveness of this algorithm. The major contribution of this paper is proposing an extension of current EDAs, which could be applied for rapid robotic imitation learning.**

Keywords-Imitation Learning; Motion Pattern; Mobile Robotics, Motion Planning; Evolutionary Computing

I. INTRODUCTION

Evolutionary computing methods have been broadly applied for searching a solution, which is normally represented as a bit string, through an evolutionary process [1]. Candidate solutions gradually move to a target solution by crossing-over the parents from the previous generation of population. These parents are selected based on their fitness function values. Normally the chromosomes which have high fitness function values in the previous generation of population are selected with higher probabilities.

Traditionally, "Cross-Over" is implemented by taking parents from the previous generation of population as input, and using some operators to exchange some parts of the chromosomes. Recently, the Estimation of Distribution Algorithm (EDA) is proposed as a more robust method [2] [3] [4]. A general procedure of EDA is: Firstly, this probabilistic method computes the distribution of each chromosome in the chromosomes which have higher evaluation function values. Secondly, the corresponding chromosome is generated from the distribution by probabilistic sampling. The EDA provides a more robust evolutionary method for finding a solution [4].

In recent years, robotic imitation learning has been proposed for robots to rapidly learn knowledge and skills from human teachers [5] [6] [7] [8] [9] [10] [11] [12] [13] [14]. In most situations, the robotic imitation learning aims at finding a path in the configuration space, which is similar to the demonstrated motion path [8] [9]. Paths are normally represented as sequences of points which the end-effector or the robotic body should pass. There are several features (or requirements) of generating paths in robotic imitation learning: 1. A desirable path should be constrained by the mechanisms of robots [6]; 2. The evaluations of candidate paths are normally based on the weighted distance between the candidate paths and a demonstrated path [7] [10]. For the first feature, it means that the path should be smooth in a configuration space. Many 'Jumping-Points' are harmful to robots. For the second feature, it means that the evaluation of a candidate path is based on the evaluation of the whole path, not a single point [11] [12].

Currently, some robotic imitation learning algorithms try to train robots to generate a path or motion trajectory which is similar to a demonstration. We expect that robots could achieve such goals through an error-driven searching automatically by themselves. Some researchers already used Reinforcement Learning (RL) related algorithms and got success [13] [15] [16]. In this paper, we try to use evolutionary computing methods to train robots to learn the demonstrated skills.

The motivation of this paper is to propose a general method of applying EDA ideas for planning a path in robotic imitation learning. For the first feature, a path searching algorithm should be used to enable robots to find possible solutions to tasks by searching the task space and finding a path to satisfy task constraints [17]. Path searching algorithms are suitable for the robotic exploration and can be divided into three main types [18]: Roadmap [19], which aims at connecting points in the task space and generating a shortest path, Potential Field [20], which generates a path in the task space by moving a particle which is attracted by the goal point and repelled by obstacles, and Strategy Searching [21] [22] [23], which tries to find a possible solution by searching the policy or strategy database. We choose to use a modified RRT-Connect algorithm for robots to search the configuration space to find a reasonable path with less 'Jumping-Points'. The basic idea of RRT-

The work described in this paper was done when the author was a PhD student at Vanderbilt University

978-1-4799-8697-2/15 $31.00 © 2015 IEEE
DOI 10.1109/SMC.2015.493

Connect is to grow the trees in the configuration space gradually which avoids the 'jumping' after smoothing the generated path.

For the second feature, we used an extension of EDA to compute the distribution of whole chromosomes, not the genes. This method is strongly related to the PRRT-Connect algorithm in this paper.

The rest of this paper is organized as follows: Section II introduces the specific algorithm design; Section III displays the simulation and experimental results on an E-Puck robot; Section IV discusses the results and the future study; Section V concludes the paper.

II. METHODOLOGY

Our algorithm can be divided into: initialization, evaluation, selection, estimation, generation, and mutation as shown in Fig.1.

At the "initialization" stage, there is no constraint in the configuration space. The particle moves from the required starting point to the ending point freely.

Chromosomes are evaluated using a fitness function and those with highest fitness function values are selected from the current population.

The distribution area of the selected chromosomes is estimated as the obstacle free area in the configuration space.

A new population of candidate solutions is generated by searching in the obstacle free area of the configuration space to find a path from the starting point to the required goal point.

Fig.1 displays the pseudo code of the designed algorithm.

Input: A demonstrated motion trajectory

Output: A generated motion trajectory

---

1 Generate the initial population P (initial) with N chromosomes (paths) and set it as the current population

2 Repeat until the termination criterion is achieved

  a. Evaluate the current population

  b. Select M chromosomes (M<N) from the current population based on some criterions

  c. Compute the distribution area of the selected chromosomes $p(\boldsymbol{x}|\boldsymbol{D^{se}})$ .in the configuration space

  d. Generate N chromosomes (paths) using PRRT-Connect algorithm from the obtained distribution

3 End

4 Return the best chromosome in the current population

Figure 1. Pseudo Code of the Proposed Algorithm

A. Initialization

Similar to traditional genetic algorithms, a population is initialized as a group of paths in an obstacle-free area of a configuration space. Using the PRRT-Connect algorithm, the paths are generated from the required starting point to the required goal point.

B. Evaluation

At the 'Evaluation' stage, a fitness function is used to calculate the distance between the demonstrated path and the generated path.

Assume that a demonstrated path is given as $\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \dots, \boldsymbol{d}_N\}^{\boldsymbol{T}}$, and $\boldsymbol{d}_i = \{d_i^1, d_i^2, \dots, d_i^m\}$. A generated path is $\boldsymbol{G} = \{\boldsymbol{g}_1, \boldsymbol{g}_2, \dots, \boldsymbol{g}_N\}^{\boldsymbol{T}}$ and $\boldsymbol{g}_i = \{g_i^1, g_i^2, \dots, g_i^m\}$.

$$Distance = \frac{\sum_{i=1}^{N} \sqrt{(\boldsymbol{d}_i - \boldsymbol{g}_i) W (\boldsymbol{d}_i - \boldsymbol{g}_i)^T}}{N} \quad (1)$$

where W is a diagonal weighted matrix.

$Distance$ is normalized to $(0,1)$ by the following equation:

$$Fitness = exp(-Distance) \quad (2)$$

From equation (2), if the distance between the demonstrated path and the generated path is around 0, the fitness value is around 1; if the distance increases, the fitness value decreases to zero.

M chromosomes are selected from the current population based the evaluation results. The selected M chromosomes have higher fitness values than others.

C. Estimation

In traditional GAs, 'cross-over' is used for generating off-springs from the parents. In EDA, this operator is replaced by estimating the distribution of the genes in chromosomes. The distribution is represented as:

$$p(\boldsymbol{x}|\boldsymbol{D^{se}}) = \prod_{i=1}^{N} p(x_i|\boldsymbol{D^{se}}) \quad (3)$$

where $\boldsymbol{x}$ is the chromosome in the population, x_i is a gene in an chromosome, and $\boldsymbol{D^{se}}$ is the obtained data from the selected chromosomes.

In imitation learning, the criterion of judging similarity is based on the overall "distance" between the demonstration and the generated trajectory. Point-to-Point analysis cannot guarantee the overall learning results. Because of the feature of path generation, in our algorithm, we do not compute the distribution of each gene. A distribution area (obstacle-free area), which contains the selected paths, is computed as $p(\boldsymbol{x}|\boldsymbol{D^{se}})$.

Paths with highest fitness values are selected and a raw distribution area which contains these paths is computed. The edge of the distribution area is obtained by finding the envelope of the selected paths. In practical implementation, the final obtained distribution area is expanded to an $Obstacle\_Free\_area$ area by dilating the obtained raw distribution area. $p(\boldsymbol{x}|\boldsymbol{D^{se}})$ is described as the distribution of an area. The points which are within the area and near the edge have lower probability. In practical implementation, we

simply assume that the $p(\boldsymbol{x}|\boldsymbol{D^{se}})$ is a uniform distribution on the $Obstacle\_Free\_area$.

The distribution area is strongly related to our PRRT-Connect algorithm chosen and the mutation algorithm in this paper. The extending of a new node is completed with high probability within or near the distribution area and with low probability far away from the distribution area.

D. Generation

Normally, the generation in EDA and its related algorithms is implemented by resampling the distribution. In our algorithm, the generation is achieved by searching and generating paths in or around this distribution area in a configuration space.

The area which is outside of the distribution area is called impedance area. A point, which is outside of this area, has probabilities of impedances. If a point is father from the distribution area, its impedance probability is higher.

RRT-Connect algorithm is proposed by Kuffner for rapidly generating paths in a configuration space [24]. The pseudo code of our PRRT-Connect algorithm [25] is displayed in Fig.2.

At step 1, two trees are initiated: Γ_a and Γ_b. The root of Γ_a is the starting point, and the root of Γ_b is the goal point. From step 2, the two trees expand rapidly to each other by growing a certain length in the configuration space at each step.

Input: A starting point and a goal point

Output: A path (Γ_a, Γ_b) connects the two input points

---

1 $\Gamma_a.init(starting\ point)$ and $\Gamma_b.init(goal\ point)$
2 $For\ k = 1\ to \max_{interations}$
  a. $q_{rand} = Rand_{Config(\Gamma_a,\Gamma_b)}$
  b. $if\ Extend\ (\Gamma_a, q_{rand}) \neq Trapped$
    1) $\Gamma_a = Connect(\Gamma_a, q_{tentative})$
    2) $if\ Reach\ (\Gamma_a, \Gamma_b) == true$
      i. $Return\ Path\ (\Gamma_a, \Gamma_b)$
    3) end
  c. end
  d. $Swap(\Gamma_a, \Gamma_b)$
3 end

Figure 2. Pseudo Code of the PRRT-Connect Algorithm

A new point is generated around the goal point, which is the root of Γ_b. The generation is achieved by sampling a normal distribution. The mean of the distribution is the root of Γ_b, and the variance is the distance between Γ_a and the root of Γ_b.

Input: Two trees Γ_a and Γ_b from the starting point and the goal point respectively, one is specified as the growing tree

Output: A random point generated in the configuration space

$Rand\_Config(\Gamma_a, \Gamma_b)$
1 $\Sigma = distance(\Gamma_a, root(\Gamma_b))$
2 $q_{rand} = RandN(root(\Gamma_b), \Sigma)$
3 $return\ q_{rand}$

Figure 3. Pseudo Code of the Rand_Config Function

Then the node on Γ_a, which is nearest to Γ_b, tentatively generates a new node by growing a predefined length to the new point. If the expanded result is within the obstacle-free area, the new node is accepted; otherwise, it is rejected. Two trees are swapped after the expanding in order to grow the two trees alternately until they are connected.

D. Mutation

Input: A growing tree Γ_a and a generated random point q_{rand}

Output: Acceptance of q_{rand}

$Extend\ (\Gamma_a, q_{rand})$
1 $q_{tentative} = \Gamma_{a_{Node_{Nearest(\Gamma_b)}}} + length * \nabla\left(q_{rand} - \Gamma_{a_{Node_{Nearest(\Gamma_b)}}}\right)$
2 $if\ q_{tentative}\ is within\ the\ Distribution_{Area}$
  a. $return\ ACCEPTED$
3 $else$
  a. $Impedance = distance(q_{tentative}, Distribution)$
  b. $Acceptance = exp(-Impedance)$
  c. $Acceptace_D = RANDN(0, _{Acceptance})$
  d. $if\ Acceptace_D > P_m$
    1) $return\ ACCEPTED$
  e. $else$
    1) $return\ TRAPPED$
  f. end
4 end

Figure 4. Pseudo Code of the Mutation

As stated before, the distribution area and the impedance area describe the probabilities of accepting or rejecting a new

node. In our algorithm, mutation is defined as that accepting a new node in the impedance area using PRRT-Connect algorithm. Therefore, the function of $Extend\ (\Gamma_a, q_{rand})$ should contain the operation of accepting a new node in the impedance area with certain probabilities.

As shown in Fig.4, one of the trees tentatively generates a new node by expanding to the new points generated in the configuration space. If the new node is within the distribution area, it is accepted and the trees add the node to it. If the new node is not within the distribution area, a probability is computed through step 5 to step 7. Step 5 and step 6 compute an Acceptance value. If the new node is around the distribution area, the value of the Acceptance is near zero. As the distance between the node and the distribution area increases, the value of Acceptance decreases to zero. A random number is generated from a uniform distribution on [0,1]. If this number is larger than the mutation probability, this node is accepted; if it is smaller than the mutation probability, this node is rejected. Obviously, if the value of Acceptance is smaller than the mutation probability, it is impossible to accept this new node. If the value of Acceptance is larger than the mutation probability, it is possible to accept this new node with some probabilities. The larger the Acceptance is, the larger the acceptance probability is.

For each generation of population, the above operations are carried until a predefined termination criterion is satisfied. The generation and the mutation step are integrated in the PRRT-Connect algorithm

III. Experimental Results

An E-Puck mobile robot trained to learn three motion patterns. Three paths are generated manually by human operators, and the robot needs to learn how to navigate in the environment using similar motion patterns. The proposed algorithm is applied to generate similar paths through iterative learning and computing.

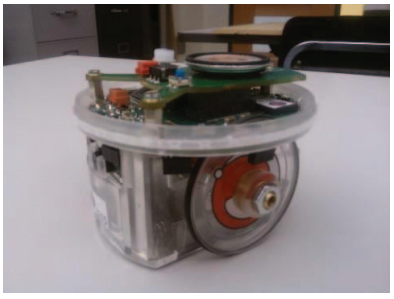

Figure 5. E-Puck Robot

A. Sine Curve Learning

In this experiment, the target is to train the robot to learn a sine curve. The desired three paths are generated on a 2-dimensional plane as shown in in upper-left picture of Fig.6. The lower-left picture of Fig.6 displays the final distribution area of the last population, and the lower-right picture of Fig.6 displays the generated paths in the last population.

The upper-right picture of Fig.6 displays the final selected path from learning results.

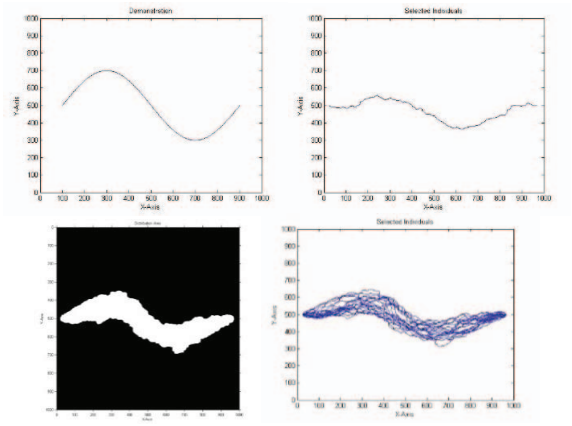

Figure 6. Results of Learning a Sine Curve

The learning process is shown in Fig.7. We only have 36 learning iterations, and the learning speed is really fast.

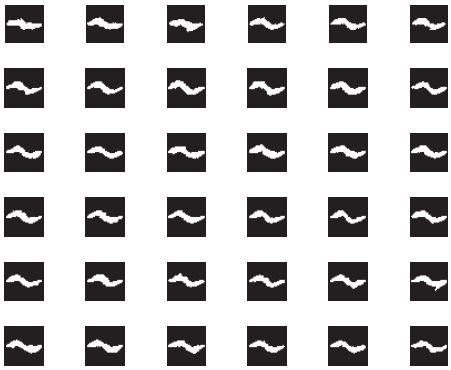

Figure 7. Evolution Process of Learning a Sine Curve

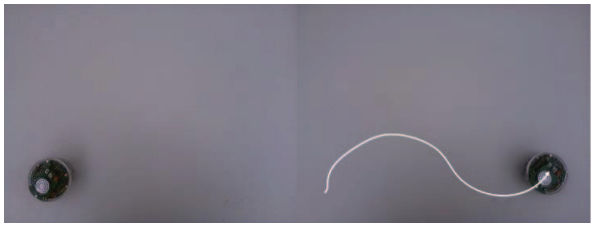

Figure 8. Experimental Results of Learning a Sine Curve

Fig.8 displays the experimental results of using an E-Puck robot to learn the sine curve.

In experiment B and C, the figures of the experimental results are organized as the same as in experiment A.

B. Inverse W-Curve Learning

Fig.9, Fig.10, and Fig. 11 display the results of learning the inverse W-curve.

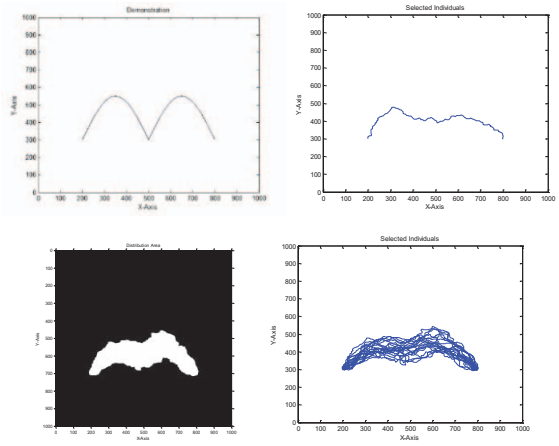

Figure 9. Results of Learning a W-Curve

Figure 10. Evolution Process of Learning a W-Curve

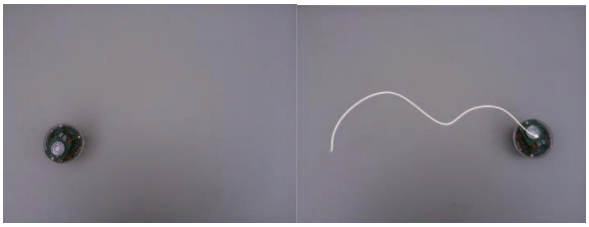

Figure 11. Experimental Results of Learning a W-Curve

C. *Inverse L-Curve Learning*

Fig.12, Fig.13, and Fig. 14 display the results of learning the inverse L-curve.

D. *Discussion*

From the comparisons of Fig.6, Fig.9 and Fig. 12, the generated paths are similar to the required path. From Fig. 7, Fig.10, and Fig.13, we can see that the learning speed is very fast. Only after several learning iterations, the estimated distribution areas are similar to the demonstrated areas.

Fig.15 (next page) displays the average fitness values of the selected chromosomes of every generation for learning the sine curve. The maximum fitness value is 0.5. From Fig.15, we can see that the fitness values increases very quickly in several initial generations. This is suitable for the robotic imitation learning, because it is expected that robots can learn the skills very fast. It robots could learn the skills from several iterations, it is better for robot to make decisions or complete other tasks much faster.

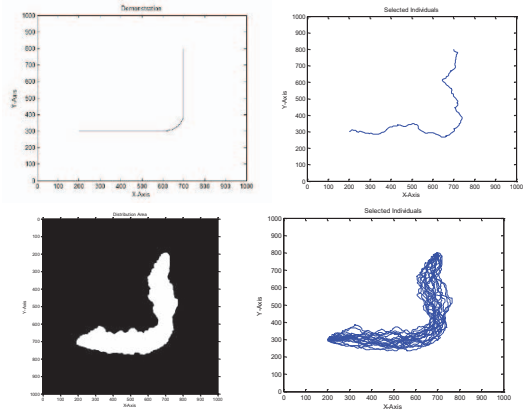

Figure 12. Results of Learning a L-Curve

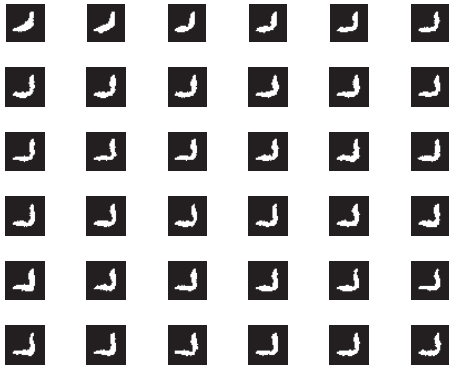

Figure 13. Evolution Process of Learning an L-Curve

Figure 14. Experimental Results of Learning an L-Curve

IV. Discussion and Future Study

This paper proposes a novel evolutionary computation method for robotic imitation learning. From the results, we can see that this algorithm can successfully generate similar trajectories. Someone may find that this algorithm is different from the conventional GAs. However, we believe that the

essence of GA or other evolutional computing methods is selecting the chromosomes from a current population and generating the new chromosomes by utilizing the information from the selected chromosomes. The proposed algorithm has the same principle as the conventional EDA. The advantage of this algorithm is that it can generate similar trajectories very quickly in fewer generations. This feature is suitable for fast robotic motion planning and decision making. However, unlike traditional EDAs, it only estimates the distribution of the chromosomes not the genes. Therefore, the fitness values of the generated populations in this algorithm are not as high as the fitness values in other evolutionary algorithms.

The next step is to find relationship between the evaluation of genes and estimation of chromosomes. If there exists a mapping between them, the fitness values could increases and be similar to the results from traditional evolutionary algorithms.

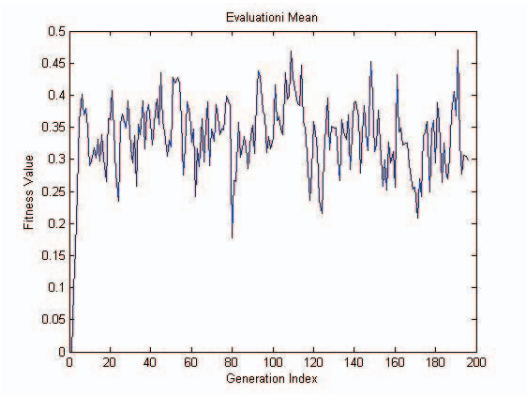

Figure 15. Fitness Values

V. CONCLUSION

This paper proposes an extension of current EDA algorithm to train robots to learn skills from human teachers through imitation. This algorithm is based on a searching algorithm and an evolutional process. The estimation of chromosomes is served as the basis of the generation step. The simulation and experimental results demonstrated that this algorithm is effective for robots to quickly learn skills from human teachers. The main contribution of this paper is proposing an extension of current EDAs by estimating the probabilistic distribution of chromosomes not the genes.

REFERENCES

[1] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing,* vol. 4, pp. 65-85, 1994.

[2] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation,* 2011.

[3] P. Larranaga and J. A. Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation vol. 2: Springer Netherlands, 2002.

[4] J. A. Lozano, Towards a new evolutionary computation: advances in the estimation of distribution algorithms vol. 192: Springer-Verlag New York Inc, 2006.

[5] C. Atkeson and S. Schaal, "Robot Learning from Demonstration," in *the Fourteenth International Conference on Machine Learning*, Morgan Kaufmann, 1997, pp. 11-73.

[6] S. Schaal, "Is imitation learning the route to humanoid robots," *Trends in Cognitive Sciences,* vol. 3, pp. 233-242, 1999.

[7] A. Billard*, et al.*, "Robot programming by demonstration," in *Handbook of robotics*, B. Siciliano and O. Khatib, Eds., ed New York, NY, USA: Springer, 2007.

[8] S. Calinon*, et al.*, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B,* vol. 37, pp. 286-298, 2007.

[9] Huan Tan, Erdem Erdemir, Kazuhiko Kawamura, and Qian Du, "A potential field method-based extension of the dynamic movement primitive algorithm for imitation learning with obstacle avoidance", *Proceedings of 2011 IEEE International Conference on Mechatronics and Automation*, pp.525-530, 2011

[10] Huan Tan, "Implementation of a Framework for Imitation Learning on a Humanoid Robot Using a Cognitive Architecture", in *The Future of Humanoid Robots - Research and Applications*, Dr. Riadh Zaier (Ed.), InTech.

[11] Huan Tan, Qian Du, and Na Wu, "Robots Learn Writing", *Journal of Robotics*, vol. 2012, Article ID 505191, 2012.

[12] Huan Tan; Qu Zhang, "Learning Behaviors from Human Teachers by Generalizing Task-Relevant Features", *Proceedings of 2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp.4391-4396, 2013.

[13] J. Peters and S. Schaal, "Reinforcement learning for parameterized motor primitives," in *International Joint Conference on Neural Networks*, 2006, pp. 73-80.

[14] A. Billard, "Learning motor skills by imitation: a biologically inspired robotic model," *Cybernetics and Systems,* vol. 32, pp. 155-193, 2001.

[15] E. Theodorou*, et al.*, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *the 2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2397-2403.

[16] J. Latombe, *Robot motion planning*: Springer Verlag, 1990.

[17] Huan Tan, Baljajee Kannan, and Lynn DeRose, "Integration of evolutionary computing and reinforcement learning for robotic imitation learning", *Proceedings of 2014 IEEE International Conference on Systems, Man and Cybernetics*, pp.407-412, 2014.

[18] E. Masehian and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning–a chronological review," in *World Academy of Science, Engineering and Technology*, 2007, pp. 101-106.

[19] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," TR 98-11, Computer Science Dept., Iowa State University1998.

[20] O. Khatib, "The potential field approach and operational space formulation in robot control," *Adaptive and Learning Systems: Theory and Applications,* pp. 367–377, 1986.

[21] G. Vachtsevanos and H. Hexmoor, "A fuzzy logic approach to robotic path planning with obstacle avoidance," in *the 25th Conference on Decision and Control*, Athens, Greece, 1986, pp. 1262-1264.

[22] R. Glasius*, et al.*, "Neural network dynamics for path planning and obstacle avoidance," *Neural Networks,* vol. 8, pp. 125-133, 1995.

[23] K. Lee and B. Zhang, "Learning robot behaviors by evolving genetic programs," in *the 26th Annual Conference of IEEE Industrial Electronics Society*, Nagoya, Japan, 2000, pp. 2867-2872.

J. J. Kuffner Jr and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *the 2000 IEEE Conference on Robotics and Automation*, San Francisco, CA, USA, 2000, pp. 995-1001.

[24] J. Kuffner*, et al.*, "Motion planning for humanoid robots," in *11th Internationa Symposium on Robotics Research*, Siena, Italy, 2003, pp. 365-374.

[25] H. Tan and K. Kawamura, "A Computational Framework for Integrating Robotic Exploration and Human Demonstration in Imitation Learning," Proceedings of *2011 IEEE International Conference on System, Man and Cybernetics*, Anchorage, AK, USA, 2011, pp. 2501-2506.