Path-smoothing for a robot arm manipulator using a Gaussian process

So-Youn Park and Ju-Jang Lee
Dept. of Electrical Engineering
Korea Advanced Institute of Science and Technology
Daejeon, Korea
psy167@kaist.ac.kr, jjlee@ee.kaist.ac.kr

***Abstract*—In this paper, we present a path-smoothing algorithm for a robot arm manipulator that finds the path using a joint space-based rapidly-exploring random tree. Unlike other smoothing algorithms which require complex mathematical computation, the proposed path-smoothing algorithm is done using a Gaussian process. To find the optimal hyperparameters of the Gaussian process, we use differential evolution hybridized with opposition-based learning. The simulation result indicates that the Gaussian process whose hyperparameters were optimized by hybrid differential evolution successfully smoothed the path generated by the joint space-based rapidly-exploring random tree.**

***Index Terms*—Differential evolution, Gaussian process, opposition-based learning, path-smoothing, rapidly-exploring random tree**

I. Introduction

Path-planning is an important issue in robotics research that finds the optimal path between two points, and many techniques have been developed so far [1]–[10]. A rapidly-exploring random tree (RRT) is a randomized data structure designed for path-planning proposed by LaValle [11]. The RRT is suitable to solve problems that involve differential constraints and to apply further algorithms for smoothing [12], [13]. In our previous study [10], a RRT for a seven degree of freedom (DOF) manipulator based on joint space is built so that inverse kinematics is required only in the beginning to find initial and goal positions. The path generated by the joint space-based RRT has no sudden moves for joints. However, there is scattering in the path if obstacle exists, which needs path-smoothing. Path-smoothing is a post process for the path-planning, which smooths the ragged path. Usually, the path-smoothing is implemented using a continuous curvature [14]–[16], which needs complex mathematical calculation. Therefore, in this paper, we develop a path-smoothing algorithm for a robot arm manipulator using a Gaussian process (GP) [17]–[19].

The GP, which is one of the most widely used stochastic processes for modeling dependent data observed time and/or spaces, is completely determined by its mean and covariance functions and solving the prediction problem is relatively straight forward. To implement the path-smoothing for the manipulator using the GP, hyperparameters of the GP need to be optimized. Differential evolution (DE) [20], [21] is a class of optimization algorithm that is simple and easy to implement and performs effectively, which shows good performance in various real world situations [22]–[26]. Therefore, to optimize the hyperparameters of the GP for path-smoothing, we use DE in this paper. Furthermore, to enhance the search ability and accelerate the search process of DE, stochastic opposition-based learning (OBL) is hybridized with DE [27].

The organization of this paper is as follows. In Section II, we provide brief introductions to RRT, GP, DE, and OBL. In Section III, we propose a novel path-smoothing algorithm using a GP whose hyperparameters are optimized by hybrid DE. In Section IV, we address the simulation setup and analyze the results of the proposed algorithm. Finally, we provide some concluding remarks and discuss future work in Section V.

II. Fumdamentals

A. Rapidly-exploring random tree

For a given initial state p_{init}, an RRT T with H vertices is constructed as in Algorithm 1 [11]. The first vertex of T is p_{init}. In each iteration, an input s which minimizes the distance from p_{near} to p_{rand} is selected; a new state p_{new} is obtained by s; and the new state p_{new} and a connection between p_{near} and p_{new} are added as a vertex and an edge to T respectively.

Algorithm 1 GENERATE_RRT($p_{init}, H, \Delta t$)

```
T.init(p_init)
for h = 1 to H do
    p_rand ← RANDOM_STATE();
    p_near ← NEAREST_NEIGHBOR(p_rand, T)
    s ← SELECT_INPUT(p_rand, p_near)
    p_new ← NEW_STATE(p_near, s, Δt)
    T.add_vertex(p_new)
    T.add_edge(p_near, p_new, u)
end for
Return T
```

B. Gaussian process

The GP of a real process $f(\mathrm{x})$ is written as

$$f(\mathrm{x}) \sim GP(m(\mathrm{x}), k(\mathrm{x},\mathrm{x}')) \tag{1}$$

where $m(\mathrm{x})$ and $k(\mathrm{x},\mathrm{x}')$ are the mean function and the covariance function defined as follows.

$$m(\mathrm{x}) = E[f(\mathrm{x})] \tag{2}$$
$$k(\mathrm{x},\mathrm{x}') = E[(f(\mathrm{x}) - m(\mathrm{x}))(f(\mathrm{x}) - m(\mathrm{x}))'] \tag{3}$$

Usually, for notational simplicity, the mean function is set to be zero. Also, given training data $\mathrm{X} = \{\mathrm{x}_i\}$, the observed target values $\mathrm{y} = \{y_i\}$ contain noise: $\mathrm{y} = f(\mathrm{X}) + \epsilon$. Assuming additive independent identically distributed Gaussian noise ϵ with variance σ_n, the covariance becomes as follows.

$$\mathrm{cov}(\mathrm{y}) = K(\mathrm{X},\mathrm{X}) + \sigma_n^2 I \tag{4}$$

Then, the joint distribution of the observed target values and the function values $f(\mathrm{X}_*)$ of test points X_* will be

$$\begin{bmatrix} \mathrm{y} \\ f(\mathrm{X}_*) \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(\mathrm{X},\mathrm{X}) + \sigma_n^2 I & K(\mathrm{X},\mathrm{X}_*) \\ K(\mathrm{X}_*,\mathrm{X}) & K(\mathrm{X}_*,\mathrm{X}_*) \end{bmatrix}\right) \tag{5}$$

and the predictive equations for Gaussian process regression will be as follows.

$$f(\mathrm{X}_*) = K(\mathrm{X}_*,\mathrm{X})[K(\mathrm{X},\mathrm{X}) + \sigma_n^2 I]^{-1}\mathrm{y} \tag{6}$$
$$\begin{aligned}\mathrm{cov}(f(\mathrm{X}_*)) = {} & K(\mathrm{X}_*,\mathrm{X}_*) \\ & - K(\mathrm{X}_*,\mathrm{X})[K(\mathrm{X},\mathrm{X}) + \sigma_n^2 I]^{-1}K(\mathrm{X},\mathrm{X}_*)\end{aligned} \tag{7}$$

Also, the log marginal likelihood $\log p(\mathrm{y}|\mathrm{X})$ will be as follows.

$$\begin{aligned}\log p(\mathrm{y}|\mathrm{X}) = {} & -\frac{1}{2}\mathrm{y}^T(K(\mathrm{X},\mathrm{X}) + \sigma_n^2 I)^{-1}\mathrm{y} \\ & -\frac{1}{2}\log|K(\mathrm{X},\mathrm{X}) + \sigma_n^2 I| - \frac{n}{2}\log 2\pi\end{aligned} \tag{8}$$

C. Differential evolution

The combination of classical DE operators can be expressed as DE/BAS/NUM/CRO, in which BAS denotes a selected base vector in mutation, NUM denotes the number of pairs of difference vectors that affect the perturbation magnitude in mutation, and CRO denotes crossover. For example, DE/Rand/1/Bin indicates a randomly selected base vector in the mutation, one pair of difference vectors in the mutation, and binomial crossover. To control the exploration magnitude and direction for obtaining promising solutions, DE perturbs a base vector using the sum of the difference between the pair of individuals multiplied by the mutation factor F. In the early iterations of the algorithm, DE tends to explore a search space with the assistance of high population diversity and subsequently exploits the solutions that it discovers for fine tuning as iterations continue. The following operators are used extensively in DE reproduction:

- Mutation
  - Rand/1

$$v_{i,j}(t) = x_{r_1,j}(t) + F(x_{r_2,j}(t) - x_{r_3,j}(t)) \tag{9}$$

  - Best/1

$$v_{i,j}(t) = x_{best,j}(t) + F(x_{r_1,j}(t) - x_{r_2,j}(t)) \tag{10}$$

- Crossover
  - Binomial

$$u_{i,j}(t) = \begin{cases} v_{i,j}(t), & \text{if } rand(0,1) \le CR \\ & \text{or } j = j_{rand} \\ x_{i,j}(t), & \text{otherwise.} \end{cases} \tag{11}$$

Here, $x_{i,j}(t)$ is the jth element of the ith individual in the population at iteration t; $v_{i,j}(t)$ is the jth element of the ith individual's offspring before crossover, called the donor vector, at iteration t; $u_{i,j}(t)$ is the jth element of the ith individual's offspring after crossover, called the trial vector, at iteration t; $r_1, r_2, r_3 \in [1\ NP]$ are randomly selected distinct integers that differ from i, $r_1 \neq r_2 \neq r_3 \neq i$; NP is the population size; F is the scaling factor controlling the exploration magnitude; $x_{best}(t)$ is the best individual at iteration t; K is the combination coefficient; $rand(0,1) \in [0\ 1]$ is a uniformly distributed random number; CR is the crossover rate; $j_{rand} \in [1\ D]$ is the randomly chosen index that ensures that at least one element from $v_i(t)$ constitutes $u_i(t)$; and D is the number of dimensions of the search space.

D. Opposition-based learning

In OBL, estimates and counter estimates are considered simultaneously to accelerate the search or process. The concept of an opposite number can be defined as follows [28]:

Definition 1. Let x be a real number in an interval $[a\ b]$. The opposite of x, denoted by $\breve{x}$, is defined as follows:

$$\breve{x} = a + b - x \tag{12}$$

Similarly, an opposite point, i.e., opposite numbers in the multidimensional case, can be defined as follows [28]:

Definition 2. Let $P(x_1, x_2, \ldots, x_D)$ be a point in D-dimensional space, where, $x_1, x_2, \ldots, x_D$ are real numbers and $x_j \in [a_j\ b_j],\ j = 1, 2, \ldots, D$. The opposite point of P is denoted by $\breve{P}(\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ where

$$\breve{x}_j = a_j + b_j - x_j \tag{13}$$

Finally, the opposition scheme for learning can be defined as follows [28]:

Definition 3. Let $f(x)$ be the function in focus and $g(\cdot)$ be a proper evaluation function. If $x \in [a\ b]$ is an initial (random) guess and $\breve{x}$ is its opposite value, then we calculate $f(x)$ and $f(\breve{x})$ in every iteration. The learning continues with x if $g(f(x)) \ge g(f(\breve{x}))$; otherwise, it continues with $\breve{x}$.

III. Proposed Algorithm

Fig. 1 shows a flowchart of the proposed algorithm. The details of the proposed algorithm are provided below.

A. Joint space-based RRT

As described in [10], a bidirectional RRT which grows single RRTs from both the initial state p_{init} and the goal state p_{goal} based on joint space is built. Collision and vision occlusion detections are performed in the bidirectional RRT.

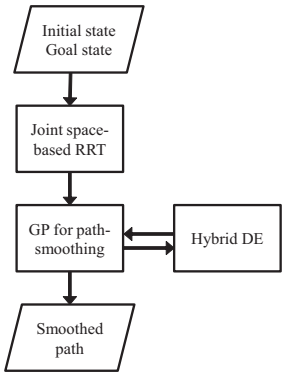


Fig. 1. A flowchart of the proposed algorithm

B. GP for path-smoothing

As mentioned in Section II-B, the observed target values y contain noise, whereas the function values $f(\mathrm{X}_*)$ of test points X_* do not contain noise. If we set the test points X_* as same as the training data X ($\mathrm{X} = \mathrm{X}_*$), $f(\mathrm{X}_*)$ will be the observed target values without noise. If we consider the scattering in the path generated by the joint space-based RRT as noise, the scattering can be removed by GP regression. Therefore, if we find the suitable hyperparameters of the GP, the path can be smoothed by GP regression. In this paper, the training data X will be time steps for the ragged path generated by the joint space-based RRT, the observed target values y_q will be the generated path for the qth joint, and the function values $f_q(\mathrm{X}) = K(\mathrm{X},\mathrm{X})[K(\mathrm{X},\mathrm{X}) + \sigma_n^2 I]^{-1}\mathrm{y}_q$ will be the smoothed path for the qth joint.

C. Hybrid DE

Usually, the hyperparameters of the GP are optimized by gradient method. However, in this paper, we use DE to find the hyperparameters of the GP. To enhance the search ability and accelerate the search process, OBL using a beta distribution with changes in the selection scheme and partial dimensions (BetaCOBL) is proposed and hybridized with DE (BetaCODE) [27].

IV. Simulation

A. Setup

In this paper, we used a seven DOF manipulator and 20 different paths generated by the joint space-based bidirectional RRT were smoothed by the GP. The manipulator and GP were implemented using robotics toolbox [29] and GPML [30]. The GP comprised a diagonal squared exponential covariance function

$$k(\mathrm{x},\mathrm{x}') = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(\mathrm{x}-\mathrm{x}')^{\mathrm{T}}(\mathrm{x}-\mathrm{x}')\right) \tag{14}$$

and a Gaussian likelihood function.

$$p(\mathrm{y}_q|\mathrm{f}_q) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(\mathrm{y}_q - \mathrm{f}_q)^2}{2\sigma_n^2}\right) \tag{15}$$

So, the hyperparameters σ_f and l for the covariance function and the hyperparameter σ_n for the likelihood function needed to be optimized. To reduce the number of variables in DE, the hyperparameters σ_f, l, and σ_n were unified for all the joints, i.e., the number of variables to be optimized in DE was set to 3. A Best/1/Bin reproduction strategy was chosen as the DE reproduction strategy, so BetaCODE was compared to DE/Best/1/Bin (DEB) and conjugate gradient method (CG). The fitness function was sum of the negative log marginal likelihood of each joint.

$$fitness(\sigma_f, l, \sigma_n) = -\sum_{i=q}^{7} \log p(\mathrm{y}_q|\mathrm{X}) \tag{16}$$

The population size was set to 30. The scaling factor and the crossover rate were set to 0.5 and 0.9 respectively. The number of fitness evaluation was set to 300 for each run. Each algorithm was run 25 times for each path. The parameter for partial dimensional change was set to [0.9, 0.1], so two opposite point candidates were generated for an original individual.

B. Performance metric

Because the real values for the hyperparameters of the GP are unknown, the following are employed as performance metrics:

- the fitness value: negative log marginal likelihood
- the number of successful runs: the smoothed path using the GP is regarded as successful if the initial and goal positions of the smoothed path are not far away from those of the original path

With regard to the fitness value, smaller fitness value, i.e., larger log marginal likelihood indicates better performance. On the other hand, with regard to the number of successful runs, more successful runs indicate better performance.

Also, we applied the Wilcoxon signed-rank test to determine whether the difference between the performances of the algorithms was significant [31].

C. Result

According to the fitness value, from Table I, BetaCODE and DEB performed better than CG. Furthermore, BetaCODE performed the best: the smallest mean and standard deviation. This is supported by the Wilcoxon test result in Table II: BetaCODE performed significantly better than DEB and CG and DEB performed significantly better than CG. According to the successful runs, from Table III, BetaCODE and DEB successfully generated the smoothed path using the GP all the time, whereas CG hardly generated the smoothed path using the GP. From these observations, it can be inferred that DE found the better solution than gradient method and BetaCOBL improved the performance of DE.

Figs. 2 and 3 show the joint angles and end-effector position before and after path-smoothing respectively. Fig. 4 shows the path in 3-D Cartesian space before and after path-smoothing. As seen, the scattering from 10 to 30 steps and the scattering from 40 to 70 steps in Figs. 2 and 4(a) disappeared in Figs.

TABLE I
MEAN AND STANDARD DEVIATION OF THE FITNESS VALUE

Path no.	BetaCODE		DEB		CG	
	Mean	SD	Mean	SD	Mean	SD
1	-1286.79	4.41	-1282.51	7.61	623.65	1002.87
2	-1218.02	2.49	-1216.34	4.90	659.02	1205.75
3	-1354.22	7.41	-1350.67	8.94	739.71	1351.93
4	-1351.82	4.18	-1343.90	18.04	813.12	1709.25
5	-1292.48	9.99	-1287.16	17.14	829.07	1726.11
6	-1297.81	14.13	-1290.68	18.73	847.16	1820.44
7	-1164.84	3.25	-1162.43	5.95	689.87	1293.72
8	-1306.56	2.49	-1304.25	6.35	656.61	1149.83
9	-1294.77	1.55	-1293.54	3.86	749.13	1424.36
10	-1332.45	15.10	-1325.28	15.18	735.35	1342.76
11	-1268.40	4.29	-1264.84	5.37	664.67	1173.15
12	-1359.29	1.67	-1355.02	12.27	805.28	1720.75
13	-1403.13	17.85	-1402.85	10.87	801.62	1650.02
14	-1306.93	0.72	-1303.11	7.80	652.57	1112.47
15	-1409.78	18.07	-1410.75	9.06	879.07	1836.11
16	-1335.20	13.81	-1327.94	21.58	687.51	1210.01
17	-1345.66	3.46	-1344.34	6.45	724.52	1341.03
18	-1382.59	14.28	-1373.52	22.46	630.41	978.30
19	-1187.28	13.62	-1182.61	16.70	897.06	2176.69
20	-1218.61	7.78	-1213.82	11.85	908.46	2160.25

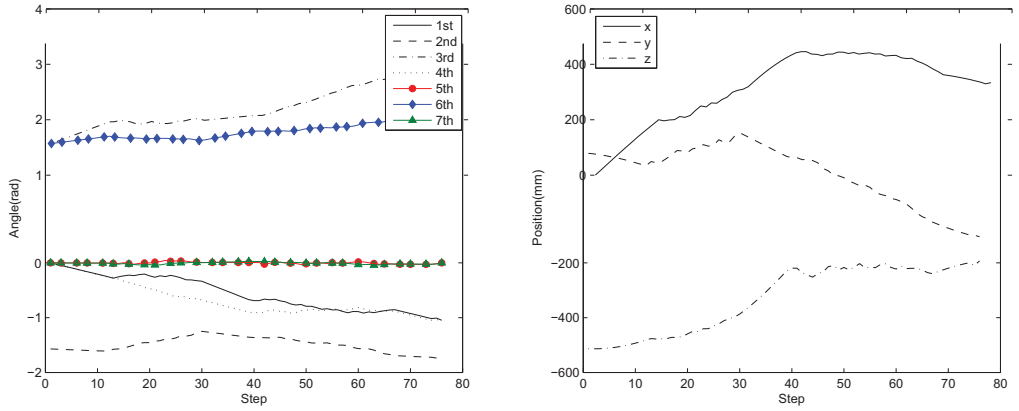
TABLE II
SUMMARY OF THE WILCOXON TEST

	BetaCODE	DEB	CG
BetaCODE	-	●	●
DEB	○	-	●
CG	○	○	-

● denotes the method in the row improves the method of the column.
○ denotes the method in the column improves the method of the row.
Upper diagonal of level significance $\alpha = 0.9$.
Lower diagonal level of significance $\alpha = 0.95$.

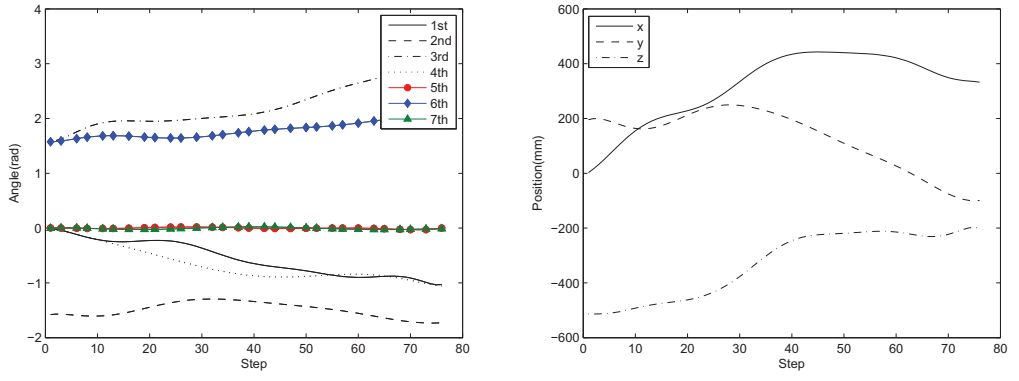
TABLE III
THE NUMBER OF SUCCESSFUL RUNS OF PATH-SMOOTHING

Path no.	BetaCODE	DEB	CG	Path no.	BetaCODE	DEB	CG
1	25	25	1	11	25	25	3
2	25	25	2	12	25	25	2
3	25	25	1	13	25	25	3
4	25	25	3	14	25	25	2
5	25	25	1	15	25	25	3
6	25	25	1	16	25	25	1
7	25	25	1	17	25	25	2
8	25	25	1	18	25	25	1
9	25	25	1	19	25	25	1
10	25	25	2	20	25	25	2



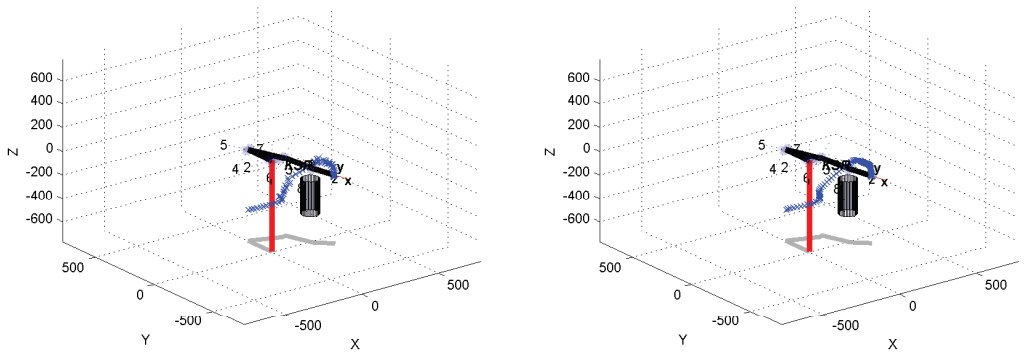
(a) Joint angles (b) End-effector position

Fig. 2. Joint angles and end-effector position before path-smoothing



(a) Joint angles (b) End-effector position

Fig. 3. Joint angles and end-effector position after path-smoothing



(a) Before (b) After

Fig. 4. Path before and after path-smoothing

3 and 4(b), which shows that BetaCODE successfully finds the hyperparameters of the GP which smooths the scattering in the path.

V. Conclusion

In this paper, we proposed a path-smoothing algorithm for a robot arm manipulator using a GP whose hyperparameters are optimized by DE. To accelerate the search process and enhance the search ability of DE, DE was hybridized with BetaCOBL. The result showed that BetaCODE found the hyperparameters of the GP and the GP smoothed the ragged path generated by the RRT successfully.

For future work, we will optimize other hyperparameters of the GP and the hyperparameters of the GP for each joint will be treated separately.

References

[1] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 1, pp. 132–141, 2013.

[2] N. Ganganath, C. Cheng, and C. Tse, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 1, pp. 132–141, 2013.

[3] L. D. Xu, C. Wang, Z. Bi, and J. Yu, "AutoAssem: An automated assembly planning system for complex products," *Industrial Informatics, IEEE Transactions on*, vol. 8, no. 3, pp. 669–678, 2012.

[4] N. Sudha, and A.R. Mohan, "Hardware-efficient image-based robotic path planning in a dynamic environment and its FPGA implementation," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 5, pp. 1907–1920, 2011.

[5] L. M. Capisani, and A. Ferrara, "Trajectory planning and second-order sliding mode motion/interaction control for robot manipulators in unknown environments," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 8, pp. 3189–3198, 2012.

[6] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 10, pp. 4813–4821, 2011.

[7] T. Sato, S. Sakaino, E. Ohashi, and K. Ohnishi, "Walking trajectory planning on stairs using virtual slope for biped robots," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 10, pp. 1385–1396, 2011.

[8] K. Yang, S. K. Gan, and S. Sukkarieh, "A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV," *Advanced Robotics*, vol. 27, no. 6, pp. 431–443, 2013.

[9] C.-B. Moon and W. Chung, "Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree," *Industrial Electronics, IEEE Transactions on*, vol. 62, no. 2, pp. 1080–1090, 2015.

[10] Y.-J. Kim, S.-Y. Park, J.-J. Kim, J.-H. Wang, J.-Y. Lee, and J.-J. Lee, "A RRT-based collision-free and occlusion-free path planning method for a 7DOF manipulator," in *Mechatronics and Automation (ICMA), 2014 IEEE International Conference on*, 2014, pp. 1017–1021.

[11] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Comput. Sci. Dept., Iowa State Univ., Ames, IA, TR 98-11, Tech. Rep., 1998.

[12] S. M. LaValle and J. J. Kuffner Jr., "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000, pp. 293–308.

[13] L. Knispel and R. Matousek, "A performance comparison of rapidly-exploring random tree and dijkstras algorithm for holonomic robot path planning," in *Recent Advances in Applied & Theoretical Mathematics*, D. Anderson, Ed. WSEAS - World Scientific and Engineering Academy and Society, 2013, pp. 154–162.

[14] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 561–568, 2010.

[15] J. Villagra, V. Milanes, J. Perez, and J. Godoy, "Smooth path and speed planning for an automated public transport vehicle," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 252–265, 2012.

[16] J. Pan, L. Zhang, and D. Manocha, "Collision-free and smooth trajectory computation in cluttered environments," *The International Journal of Robotics Research*, 2012.

[17] C. M. Bishop, *Pattern recognition and machine learning*. Springer New York, 2006.

[18] R. A. Davis, "Gaussian process," in *Encyclopedia of Environmetrics*, Wiley, New York, 2006.

[19] C. K. I. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. The MIT Press, 2005.

[20] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2011.

[21] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: A practical approach to global optimization*. Springer-Verlag Berlin Heidelberg, 2005.

[22] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "An improved self-adaptive differential evolution algorithm for optimization problems," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 1, pp. 89-99, 2013.

[23] J.-T. Tsai, K.-M. Lee, and J.-H. Chou, "Robust evolutionary optimal tolerance design for machining variables of surface grinding process," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 1, pp. 301–312, 2014.

[24] R. B. Godoy, J. O. P. Pinto, C. A. Canesin, E. A. Alves Coelho, and A. M. A. C. Pinto, "Differential-evolution-based optimization of the dynamic response for parallel operation of inverters with no controller interconnection," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 7, pp. 2859–2866, 2012.

[25] T. Marcic, B. Stumberger, and G. Stumberger, "Differential-evolution-based parameter identification of a line-start IPM synchronous motor," *Industrial Electronics, IEEE Transactions on*, vol. 61, no. 11, pp. 5921–5929, 2014.

[26] C.-F. Juang, Y.-H. Chen, and Y.-H. Jhan, "Wall-following control of a hexapod robot using a data-driven fuzzy controller learned through differential evolution," *Industrial Electronics, IEEE Transactions on*, vol. 62, no. 1, pp. 611–619, 2015.

[27] S.-Y. Park and J.-J. Lee, "Stochastic opposition-based learning using a beta distribution in differential evolution," *Cybernetics, IEEE Transactions on*, Article in press.

[28] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, 2005, pp. 695–701.

[29] P. I. Corke, *Robotics, vision & control: Fundamental algorithms in matlab*. Springer, 2011.

[30] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (GPML) toolbox," *Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, 2010.

[31] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesús, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "KEEL: A software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

(접수:2015.09.23., 수정:2015.10.21., 게재확정:2015.10.28.)