IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 23, 2020, accepted April 5, 2020, date of publication April 28, 2020, date of current version May 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2991003

Smooth Curve Fitting of Mobile Robot Trajectories Using Differential Evolution

VICTOR PARQUE[1,2], (Member, IEEE), AND TOMOYUKI MIYASHITA[1], (Member, IEEE)
[1]Department of Modern Mechanical Engineering, Waseda University, Tokyo 169-8555, Japan
[2]Department of Mechatronics and Robotics, Egypt-Japan University of Science and Technology, Borg-El Arab 21934, Egypt

Corresponding author: Victor Parque (parque@aoni.waseda.jp)

ABSTRACT Mobile robots have recently attracted the attention and applicability in field areas ubiquitously. Within the context of autonomous navigation, path planning is relevant for comfortability, safety, execution time and energy savings. In this paper, we propose an approach to suggest smooth paths from observed robot trajectories by optimizing fitting and smoothness criteria using Differential Evolution with distinct modes of initialization, selection pressure, exploration and exploitation. Our rigorous computational experiments using a relevant set of real-world robot trajectories from the Boe-Bot mobile robot architecture show the feasibility and efficiency of our approach in computing smooth curves, suggesting the superior performance of the greedy initialization scheme based on the triangular convex hull of the robot trajectory, and Differential Evolution with exploitative and parameter adaptation schemes such as Rank-Based Differential Evolution (RBDE), Adaptive Differential Evolution with External Archive (JADE) and Strategy Adaptation Differential Evolution (SADE). Our obtained results offer the building blocks to further advance towards developing data-driven curve fitting and path planning algorithms, which may find use in several real-world applications in Robotics and Operations Research.

INDEX TERMS Smooth curve fitting, optimization, differential evolution, mobile robots, path smoothing.

I. INTRODUCTION

In recent years, mobile robots have extended their application scope in field areas, including agriculture, forestry and manufacturing. In line of the above, to realize efficient navigation, path planning with smoothness considerations is relevant for comfortability, safety, minimum execution time, and energy savings.

Path planning has been extensively studied in the literature. Often, in the context of autonomous robot navigation, collision-free trajectories are generated considering safety and optimality. In line of these achievements, well-known methods such as Rapidly-exploring Random Tree (RRT) and Probabilistic Roadmaps (PRM) guarantee probabilistic completeness, while RRT* guarantees asymptotic optimality. However, challenges in scalability often arise, since approaches based on RRT and PRM have to extend over large maps to find optimal trajectories. To avoid covering the entire map, often approaches based on optimization using heuristic sampling and gradient of the cost function (length) are used. Examples of these approaches include CHOMP [1], STOMP [2], and TrajOpt [3]. However, these methods are sensitive to initial conditions (initial trajectory). Also, according to the presence or absence of knowledge of the environment, it is well-known the division between *global* and *local* methods in path planning. Also, the *incremental* methods such as Rapidly Exploring Random Trees (RRT), and their extensions, are well-known in the literature for its exploratory features. Furthermore, path planning algorithms using *graph-theory* and *search heuristics* are also well-known, e.g. Dijkstra and A*, visibility graphs combined with sampling heuristics [4]–[8].

Also, researchers have tackled the path planning problem by using geometric and graph-based approaches [9]–[13]. In line of these schemes, it has been shown that path planning in triangulated space is highly accurate and efficient. For a map having polygonal obstacles with n vertices, it is possible to use the Delaunay Triangulation of the free space to render a connected graph with $O(n)$ nodes, each of which represents the triangles conforming the free-space. Then, path planning can be efficiently achieved by graph search methods on the adjacency graph of the triangulation. If one uses the Dijkstra-based search [9], time complexity is bounded by $O(n.log(n))$.

The associate editor coordinating the review of this manuscript and approving it for publication was Ruofei Ma.

Complementarily, it is possible to use A* search [10], [11] along with the funneling algorithm [12], [13] to achieve a quasi-linear time complexity.

On the other hand, in order to consider safety and comfortability in mobile robot driving, the inclusion of *smoothness* factors in path planning has attracted the attention of the community [4], [14], [15]. In terms of safety, path planning methods usually maximize the distance between the vehicle and obstacles [16], and in terms of comfortability, path planning methods usually consider the change and continuity of curvature of the path. For instance, in [17] paths using Bézier curves are smoothed, in which theta* RRT is able to render feasible any-angle paths by using geometric information of the workspace and satisfying the non-holonomic constraints. Also, in [18], the Dijkstra algorithm is used to compute shortest paths, and then smooth trajectories are generated by cubic Bézier curves. In [19], clothoid curves are used to achieve paths which are shorter and have lower curvature derivatives. In [20], various types of Bézier curves are used to generate smooth road trajectories. Furthermore, in [21], vehicle and road constraints are considered in order to generate collision-free paths. In [14], Support Vector Machines are used to compute the critical points in obstacle-prone environments, and the control points of Bézier curves are optimized to compute smooth paths. In [22], Bézier curves were smoothed by Genetic Algorithm with diversity factors. In [23], the lattice-based motion planners were proposed to compute feasible paths, in which the ACADO Toolkit has the role of path smoothing and a Model Predictive Controller (MPC) has the role of optimizing control outputs to follow the trajectory. In [24], the global path planning was developed based on Visibility Graphs and Bézier curves were used to minimize the maximum curvature in order to generate a smooth path.

In line with the above, in [25], the path planning for corridors with curvature discontinuities is proposed by composing half-S-shaped paths of the Four Parameter Logistic (4PL) family achieving zero-end curvature and collision-avoidance with passages. In [26], quintic Bézier curves compute the smooth sequence of curves of A*-based paths by solving the constrained problem and globally-convergent moving-asymptotes (MMA) algorithm. In [27], the generation of smooth paths by gradient-informed post-smoothing algorithm (GRIPS) is proposed, in which locally placed vertices are optimized by considering the safe distance between the robot and the obstacles.

In order to generate smooth trajectories, the conventional methods basically have focused on the knot placement of polynomial curves to compute smooth paths and then perform trajectory following to ensure compliance with the generated path. In this paper, we propose an approach to generate smooth trajectories from given robot trajectory points. This problem is related to the well-known curve fitting and reconstruction problem, that is fitting data points to curves, which has attracted the attention of the non-linear optimization community due to the gradient-free and ability to deal with multimodal optimization approaches, e.g. Particle Swarm Optimization [28], Artificial Immune Systems [29], Lasso [30], Estimation of Distribution Algorithms [31], Invasive weed optimization [32], Differential Evolution [33], and Hybrids [34].

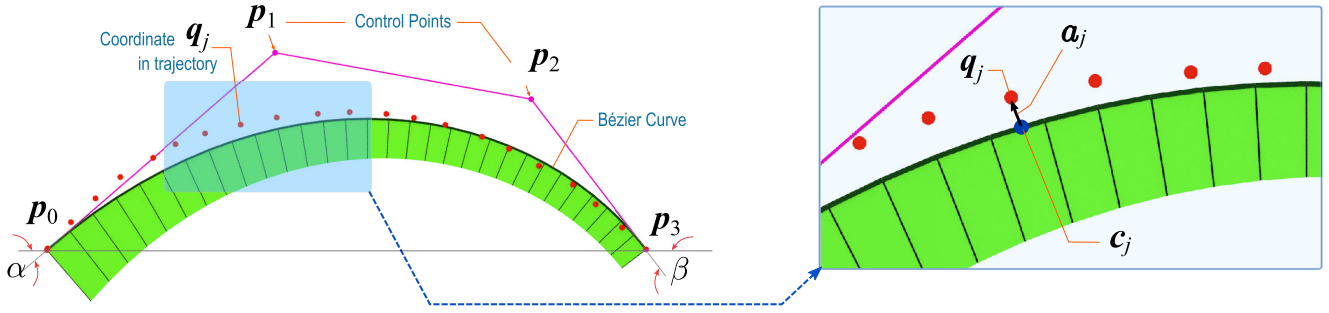
Although the above-mentioned schemes have rendered competitive curve fitting metaheuristics, it is unclear whether initialization, selection pressure, parameter adaptation, exploration or exploitation play key roles for the competitive performance to fit and compute fair curves to robot trajectory data. Thus, in this paper, to tackle the above-mentioned line of inquiry, our focus and contribution are as follows:

- We propose an approach to generate smooth paths from observed robot trajectories by optimizing the fitting and the smoothness performance with Differential Evolution with distinct modes of initialization, parameter adaptation, selection pressure, exploration and exploitation. In particular we used the following five relevant classes:
  - DERAND: DE/rand/1/bin strategy
  - RBDE: Rank-Based Differential Evolution
  - JADE: Adaptive Differential Evolution with External Archive
  - SADE: Strategy Adaptation Differential Evolution
  - DESIM: Differential Evolution with Similarity Based Mutation
- The exhaustive experiments using all feasible scenarios of torque control of Boe-Bot mobile robot show (1) the feasibility and efficiency of our approach to generate smooth curves from given trajectory data, (2) the improvement of the convergence rate by a greedy initialization scheme considering the triangular convex hull of trajectory data, and (3) the competitive convergence and curvature profiles being attainable by the class of Differential Evolution algorithms with exploitative and parameter adaptation schemes: RBDE, JADE and SADE.
- Our proposed approach offers insights on the performance of relevant Differential Evolution algorithms generating smooth paths for mobile robots complying with fitting and smoothness criteria, thus being potential to enable comfortability, energy-efficiency, and computationally efficient controllers.

In the following sections, we describe the basic idea of our proposed approach, and discuss our computational experiments.

II. PROPOSED APPROACH

Basically, we tackle the problem of fitting and computing fair curves to prescribed robot trajectory data. Smooth curves are computed by minimizing a cost function comprising the fitting error to trajectory inputs and the smoothness defined by the curvature profile of a curve. To show an example of our proposed scheme, Fig. 1 shows the main elements involved in our approach. Here, the curve $\boldsymbol{r}(u)$ is defined by the control points $\boldsymbol{p}_i$, $i = 0, 1, \ldots, n$, each of which is computed by minimizing the fitting error and the smoothness



(a) Curvature Profile (b) Vector a_j

FIGURE 1. Basic concept and main elements of the curvature profile in the Bézier curve. (a) The curve is defined by locations from both the control points and the Bernstein polynomials, (b) the vector a_j points from c_j in the curve towards the point q_j in the trajectory.

component of the curve. The fitting error aims at minimizing the shortest distances from prescribed input coordinates $\boldsymbol{q}_j$, $j = 1, 2, \ldots, m$ defined by the robot trajectory to the curve $\boldsymbol{r}(u)$, as shown by Fig. 1-(b). The smoothness of the curve is defined by aiming at finding the monotone variation in the curvature profile of $\boldsymbol{r}(u)$, as portrayed by the green-colored region of Fig. 1 showing the magnitude of the radius of curvature along the curve $\boldsymbol{r}(u)$ for $u \in [0, 1]$.

In the following section, we briefly describe the key components and dynamics of our proposed approach.

A. BASIC CONCEPT

Let a Bézier curve of degree n be defined by

$$\boldsymbol{r}(u) = \sum_{i=0}^{n} B_i^n(u)\boldsymbol{p}_i, \quad u \in [0, 1] \tag{1}$$

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i} \tag{2}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}, \tag{3}$$

where $\boldsymbol{p}_i$ is the i-th control point of the Bézier curve in the domain $\mathbb{R}^2$, and $B_i^n(u)$ is the i-th Berstein polynomial of degree n for $i \in \{0, 1, 2, \ldots, n\}$. For simplicity, and without loss of generality, we use the degree $n = 3$, in which Bernstein polynomials are represented by

$$B_0^3(u) = (1-u)^3, \tag{4}$$
$$B_1^3(u) = 3u(1-u)^2, \tag{5}$$
$$B_2^3(u) = 3u^2(1-u), \tag{6}$$
$$B_3^3(u) = u^3. \tag{7}$$

Basically, the curve $\boldsymbol{r}(u)$ lies within the *convex hull* of the control points $\boldsymbol{p}_i$, $i = 0, 1, \ldots, n$. Also, each point lying in $\boldsymbol{r}(u)$ is computed by the weighted-average of the control points $\boldsymbol{p}_i$, with weights defined by Bernstein polynomials, and fulfilling the condition $\sum_{i=0}^{n} B_i^n(u) = 1$.

Then, smooth paths are computed by minimizing the following cost function

$$\min_{\boldsymbol{p}_i} F = E + \lambda H, \tag{8}$$

where $\boldsymbol{p}_i$ is the i-th control point of the Bézier curve $\boldsymbol{r}(u)$, depicted by nodes connected by lines in pink color on Fig. 1-(a), E is the component related to the fitting error of the curve $\boldsymbol{r}(u)$ to given input trajectory coordinates $\boldsymbol{q}_j$, $j = 1, 2, \ldots, m$, H is the smoothness component (fairness metric) of the curve $\boldsymbol{r}(u)$, and λ is the user-defined parameter to balance the fitting error and the smoothness factors. Concretely speaking, the following relation computes the fitting error:

$$E = \sum_{j=1}^{m} \|\boldsymbol{a}_j\|^2 \tag{9}$$

$$\boldsymbol{a}_j = \boldsymbol{q}_j - \boldsymbol{c}_j, \tag{10}$$

where

- m is the number of input trajectory points,
- $\boldsymbol{a}_j$ is the vector pointing from $\boldsymbol{c}_j$ to $\boldsymbol{q}_j$,
- $\boldsymbol{q}_j \in \mathbb{R}^2$ is the j-th input trajectory coordinate obtained from the robot's trajectory measurements,
- $\boldsymbol{c}_j \in \mathbb{R}^2$ is the nearest point in the curve $\boldsymbol{r}(u)$ to the input coordinate $\boldsymbol{q}_j$, which is obtained by the following:

$$\min_{u} \|\boldsymbol{q}_j - \boldsymbol{r}(u)\|, \tag{11}$$

where $u \in [0, 1]$, and $\|\cdot\|$ denotes the Euclidean norm.

Furthermore, the smoothness component of the curve is computed by

$$H = \int \left(\frac{d^2\kappa}{ds^2}\right)^2 ds, \tag{12}$$

where κ is the curvature of the curve $\boldsymbol{r}(u)$ and s is the arc-length of the curve $\boldsymbol{r}(u)$.

In the above expression

$$\kappa = \frac{\|\boldsymbol{r}' \times \boldsymbol{r}''\|}{\|\boldsymbol{r}'\|^3}, \tag{13}$$

$$\frac{ds}{du} = \|\boldsymbol{r}'\| \tag{14}$$

By the chain rule,

$$\begin{aligned}\frac{d^2\kappa}{ds^2} = {} & \frac{\|\boldsymbol{r}' \times \boldsymbol{r}^{iv}\| + \|\boldsymbol{r}'' \times \boldsymbol{r}'''\|}{\|\boldsymbol{r}'\|^5} - 4\frac{\|\boldsymbol{r}' \times \boldsymbol{r}'''\|(\boldsymbol{r}' \times \boldsymbol{r}'')}{\|\boldsymbol{r}'\|^7} \\ & - 3\frac{\left(\|\boldsymbol{r}' \times \boldsymbol{r}''\|(\boldsymbol{r}' \cdot \boldsymbol{r}''' + \boldsymbol{r}'' \cdot \boldsymbol{r}'') + \|\boldsymbol{r}' \times \boldsymbol{r}'''\|(\boldsymbol{r}' \times \boldsymbol{r}'')\right)}{\|\boldsymbol{r}'\|^7} \\ & + 18\frac{\|\boldsymbol{r}' \times \boldsymbol{r}''\|(\boldsymbol{r}' \times \boldsymbol{r}'')^2}{\|\boldsymbol{r}'\|^9},\end{aligned} \tag{15}$$

where $\boldsymbol{r}' = \frac{d\boldsymbol{r}(u)}{du}$, $\boldsymbol{r}'' = \frac{d^2\boldsymbol{r}(u)}{du^2}$, $\boldsymbol{r}''' = \frac{d^3\boldsymbol{r}(u)}{du^3}$, $\boldsymbol{r}^{iv} = \frac{d^4\boldsymbol{r}(u)}{du^4}$.

To give a glimpse of the smoothness factor, Fig. 1 also shows an example of the curvature profile of a Bézier curve as well as the main components to compute the error in curve fitting. Here, the radius of curvature $\rho = 1/\kappa$ is depicted by green color in Fig. 1. The reader may note that the errors in curve fitting E as well as the smoothness component H (fairness metric) are contradicting terms, determining the optimal ratio λ is out of the scope of this paper.

Furthermore, we use the integration of $\left(\frac{d^2\kappa}{ds^2}\right)^2$ due to its relevance to compute curvature profiles with linear nature. The reader may note that using other fairness metrics, such as the minimum energy curvature (MEC) and the minimum variation curvature (MVC), is also possible. Also, once curves minimizing the criterion in Eq. 8 are computed, it becomes possible to generate compounded smooth paths with C^0 continuity by joining the curvature profiles of Bézier paths and by matching the tangent angles α and β between two consecutive curvature profiles. However, compounded paths will require further pruning to satisfy optimality of the cost function F (and smoothness component H) at the connecting node points of two path elements. The study of compounded fair curves is out of the scope of this paper, and left in our future agenda.

B. DIFFERENTIAL EVOLUTION

The cost function depicted by Eq. 8 is of non-linear landscape, and computing its gradient is of non-trivial nature. Thus, we use we use the class of gradient-free optimization algorithms considering features of balance between exploration and exploitation. In this paper, we use the relevant classes of *Differential Evolution (DE)* [35] algorithms due to the flexibility and the versatility to realize diversity of exploration and exploitation during search. In line of the above, we used five relevant classes of Differential Evolution as optimization heuristics, each of which denotes distinct modes of selection pressure and balance between *exploration* and *exploitation*. Considering other nature-inspired heuristics is straightforward, yet including a large number of such heuristics is out of the scope of this paper.

Solutions in a *minimization* problem are sampled by:

$$\boldsymbol{x}_{t+1} = \begin{cases} \boldsymbol{u}_t & f(\boldsymbol{u}_t) \leq f(\boldsymbol{x}_t) \\ \boldsymbol{x}_t & \text{otherwise} \end{cases} \tag{16}$$

$$\boldsymbol{u}_t = \boldsymbol{x}_t + \boldsymbol{b}_t \circ (\boldsymbol{v}_t - \boldsymbol{x}_t) \tag{17}$$

$$\boldsymbol{b}_t = [b_{t,1}, b_{t,2}, b_{t,3}, \ldots, b_{t,k}, \ldots, b_{t,D}] \tag{18}$$

$$b_{t,k} = \begin{cases} 1, & r_{t,k} \leq CR \text{ or } k = krand \\ 0, & \text{otherwise,} \end{cases} \tag{19}$$

where t denotes the iteration index, $\boldsymbol{x}_t$ is a D-dimensional (individual) vector ($\boldsymbol{x}_t \in \mathbb{R}^D$), $\boldsymbol{u}_t$ is the *trial* solution at iteration t, $\circ$ is the element-wise *Hadamard* product, $\boldsymbol{v}_t$ is the *mutant* vector at iteration t, $\boldsymbol{b}_t$ is a D-dimensional binary vector, $r_{t,j}$ is a random number with uniform distribution $U[0,1]$, *krand* is a random integer uniformly distributed in $U[1,D]$, and CR is the cross-over probability. Due to handling Bézier curves of degree $n = 3$ and due to control points laying

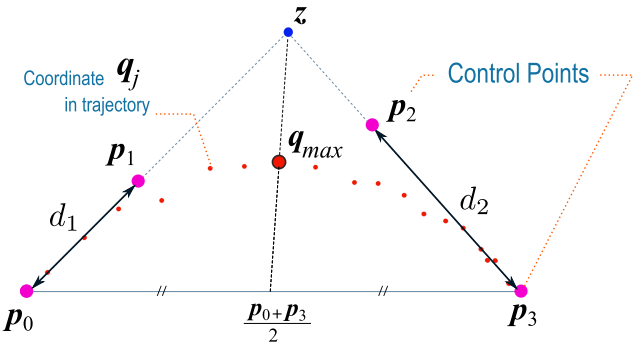


FIGURE 2. Basic idea for greedy initialization.

in the plane ($\boldsymbol{p}_i \in \mathbb{R}^2,\ i \in \{0,1,2,3\}$ in the formulation of Eq. 1 and Eq. 8), the (individual) vector $\boldsymbol{x}_t$ encodes the $x-y$ coordinates of the control points of the Bézier curve, thus $D = 2(n+1) = 8$.

Generally speaking, Differential Evolution considers the evolution of the set $\mathscr{P}$ of individuals, with $NP = |\mathscr{P}|$ denoting the population size, in which three relevant processes are performed:

- initialization to generate the population $\mathscr{P}$,
- mutation to generate the vector $\boldsymbol{v}_t$,
- crossover to generate the trial vector $\boldsymbol{u}_t$ (Eq. 17),
- selection to generate the vector $\boldsymbol{x}_{t+1}$ (Eq. 16).

As for initialization of the population, we used two different procedures, as follows:

- Random Initialization. Control points $\boldsymbol{p}_i \in \mathbb{R}^2$ are set arbitrarily within the convex hull defined by the $x-y$ coordinates of $-\epsilon\boldsymbol{q_m}$ and $\epsilon\boldsymbol{q_m}$, for a constant $\epsilon \geq 1$. This procedure is in line with the conventional initialization schemes in Differential Evolution, in which solutions are sampled from the hypercube defined by the lower and upper bound of the search space. Since the coordinate $\boldsymbol{q}_m$ represents the last input coordinate from the observed robot trajectory, new coordinates sampled within $-\epsilon\boldsymbol{q_m}$ and $\epsilon\boldsymbol{q_m}$ ensure the feasibility of the search space.
- Greedy Initialization. Here, the first (last) control point is set equal to the first (last) input trajectory; and $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ are computed from interpolating the vertices of the triangular convex hull of the coordinates $\boldsymbol{q}_j$ ($j \in [m]$), as portrayed by Fig. 2. Concretely speaking, this procedure is realized as follows:

$$\boldsymbol{p}_0 = \boldsymbol{q}_1 \tag{20}$$

$$\boldsymbol{p_1} = \boldsymbol{p}_0 + d_1(\boldsymbol{z} - \boldsymbol{p}_0) \tag{21}$$

$$\boldsymbol{p}_2 = \boldsymbol{p}_0 + d_2(\boldsymbol{z} - \boldsymbol{p}_3) \tag{22}$$

$$\boldsymbol{p}_3 = \boldsymbol{q}_m \tag{23}$$

$$\boldsymbol{z} = \epsilon\boldsymbol{q_{max}} - \frac{\boldsymbol{p}_0 + \boldsymbol{p}_3}{2}, \tag{24}$$

where $\boldsymbol{q_{max}}$ is the farthest coordinate from the segment $\overline{\boldsymbol{p}_0\boldsymbol{p}_3}$, $\epsilon \geq 1$ is a user-defined constant to estimate the triangular convex hull of the input coordinates $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m$, and $d_1,\ d_2 \in [0,1]$ are interpolation constants. This procedure enables to define the control

points of the Bézier curve such that the coordinates defined by the robot trajectory lie within the convex hull of the polygon defined by $\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2$ and $\boldsymbol{p}_3$.

After initialization, procedures related to mutation, crossover and selection are performed until the number of maximum number of evaluations of the cost function is attained. In line with our goal of evaluating distinct modes of selection pressure, exploration and exploitation, we describe the relevant heuristics used in our study in the following sections.

1) DE/rand/1/bin (DERAND)

This algorithm is one of the well-known strategies able to handle nonlinear cost functions [35], offering exploration abilities during search. Here, the mutant vector $\boldsymbol{v}_t$ are sampled as follows:

$$\boldsymbol{v}_t = \boldsymbol{x}_t^{\boldsymbol{r_1}} + F_d(\boldsymbol{x}_t^{\boldsymbol{r_2}} - \boldsymbol{x}_t^{\boldsymbol{r_3}}), \tag{25}$$

where $r_1, r_2, r_3 \in U[1, NP]$, with $r_1 \neq r_2 \neq r_3 \neq k$, NP is the population size, and F_d is the scaling factor. In the above, exploration is tunable by the factor F_d.

2) RANK-BASED DIFFERENTIAL EVOLUTION (RBDE)

Here, the mutant vector v_t is computed by [36]:

$$\boldsymbol{v}_t = \boldsymbol{x}_t^1 + F_d(\boldsymbol{x}_t^2 - \boldsymbol{x}_t^3), \tag{26}$$

where $\boldsymbol{x}_t^1, \boldsymbol{x}_t^2$ and $\boldsymbol{x}_t^3$ are individual vectors selected from the population by using the Whitley Distribution. That is, for $i_w = \{1, 2, 3\}$:

$$\boldsymbol{x}_t^{i_w} = \mathscr{P}_c^S \tag{27}$$

$$c = \left\lfloor \frac{NP}{2(\theta-1)} \left(\theta - \sqrt{\theta^2 - 4(\theta-1)r} \right) \right\rfloor, \tag{28}$$

where $\mathscr{P}_c^S$ is the c-th individual in the *population sorted* by fitness from best to worst, θ is a user-defined bias term, r is a random number uniformly distributed in $U[0, 1]$, and $\lfloor \cdot \rfloor$ denotes the floor function.

The above-mentioned sampling scheme focuses on the selective pressure to allow the improved *exploitation* by using a rank-based ordering of the population. The range of $\theta \in (1, 3]$ is found to be favorable to improve the convergence in non-separable problems [36].

3) ADAPTIVE DIFFERENTIAL EVOLUTION WITH EXTERNAL ARCHIVE (JADE)

This algorithm uses the successful historical references to update the learning parameters CR and F_a [37]. The mutation is based on DE/current-to-$p$best/1 with an *archive of inferior solutions*, as follows:

$$\boldsymbol{v}_t = \boldsymbol{x}_t + F_{x_t}(\boldsymbol{x}^{\boldsymbol{pbest}} - \boldsymbol{x}_t) + F_{x_t}(\boldsymbol{x}_t^{\boldsymbol{r_1}} - \breve{\boldsymbol{x}}_t^{r_2}), \tag{29}$$

where F_{x_t} is the mutation scaling factor associated to vector $\boldsymbol{x}_t$, $\boldsymbol{x}^{\boldsymbol{pbest}}$ is a random individual from the $100p\%$ best of the population $\mathscr{P}$ for a constant$p \in (0, 1]$, r_1 is an (integer) index chosen randomly from $[1, NP]$, $\breve{\boldsymbol{x}}_t^{r_2}$ is a random individual from $\mathscr{P} \cup \mathcal{A}$ ($r_2 \in [1, NP + |\mathcal{A}|]$, $r_1 \neq r_2 \neq k$), in which:

- $\mathcal{A}$ denotes the set of inferior solutions,
- $\mathcal{A}$ is empty during initialization,
- The vector $\boldsymbol{x}_t$ is added to the set $\mathcal{A}$ if the selection of u_t succeeds in (Eq. 16), and
- Elements of the set $\mathcal{A}$ are deleted arbitrarily if $|\mathcal{A}| > NP$.

Also, the scaling factor and crossover rate are updated by the following rules:

$$CR_{x_t} \sim \mathbf{N}(\mu_{CR}, \sigma_{CR}) \tag{30}$$

$$F_{x_t} \sim Cauchy(\mu_F, \sigma_F), \tag{31}$$

where $\mathbf{N}(\cdot, \cdot)$ is a random number with normal distribution, *Cauchy* is a random number with Cauchy distribution (truncated at $[0, 1]$), and CR_{x_t} is the crossover rate associated to vector $\boldsymbol{x}_t$. During initialization, $\mu_{CR} = \mu_F = 0.5$; $\sigma_{CR} = \sigma_F = 0.1$. Then, after selection (Eq. 16), the following is computed:

$$\mu_{CR} = (1-c)\mu_{CR} + c\overline{S}_{CR} \tag{32}$$

$$\mu_F = (1-c)\mu_F + c\overline{S}_F, \tag{33}$$

where $c \in [0, 1]$ is a user-defined constant for averaging, S_{CR} and S_F are sets of successful parameters corresponding to CR_{x_t} and F_{x_t}, respectively, $\overline{S}_{CR}$ is the arithmetic mean of S_{CR}, and $\overline{S}_F$ is the Lehmer mean of S_F as follows:

$$\overline{S}_F = \frac{\sum_{F_s \in S_F} {F_s}^2}{\sum_{F_s \in S_F} F_s} \tag{34}$$

4) STRATEGY ADAPTATION DIFFERENTIAL EVOLUTION (SADE)

SADE allows a pool of strategies learn selection probabilities over a number of generations, wherein probabilities are updated in line with successful mutation [38]. Thus, rather than implementing one mutation strategy, SADE uses a set of strategies. Here, $\boldsymbol{v}_{g,t}$ denotes the *vector* generated by the gth strategy at iteration t. We use the original pool of strategies as those used in [38], that is the DE/rand/1/bin strategy defined by Eq. 25, and four additional strategies, as follows:

DE/rand-to-best/2/bin

$$\boldsymbol{v}_t = \boldsymbol{x}_t + F_d(\boldsymbol{x}_t^{best} - \boldsymbol{x}_t) + F_d(\boldsymbol{x}_t^{r_1} - \boldsymbol{x}_t^{r_2}) \tag{35}$$

DE/current-to-best/2/bin

$$\boldsymbol{v}_t = \boldsymbol{x}_t + F_d(\boldsymbol{x}_t^{best} - \boldsymbol{x}_t) + F_d(\boldsymbol{x}_t^{r_1} - \boldsymbol{x}_t^{r_2}) + F_d(\boldsymbol{x}_t^{r_3} - \boldsymbol{x}_t^{r_4}) \tag{36}$$

DE/rand/2/bin

$$\boldsymbol{v}_t = \boldsymbol{x}_t^{r_1} + F_d(\boldsymbol{x}_t^{r_2} - \boldsymbol{x}_t^{r_3}) + F_d(\boldsymbol{x}_t^{r_4} - \boldsymbol{x}_t^{r_5}) \tag{37}$$

DE/current-to-rand/1/bin (with $CR = 1$)

$$\boldsymbol{v}_t = \boldsymbol{x}_t + F_d(\boldsymbol{x}_t^{r_1} - \boldsymbol{x}_t) + F_d(\boldsymbol{x}_t^{r_2} - \boldsymbol{x}_t^{r_3}), \tag{38}$$

where r_1, r_2, r_3, r_4, r_5 are mutually exclusive random integers in the uniform range $[1, NP]$.

Furthermore, the probability to select strategies, the crossover probability CR and the scaling factor F_d are computed adaptively following the same learning rules proposed in SADE [38].

5) DIFFERENTIAL EVOLUTION WITH SIMILARITY BASED MUTATION (DESIM)

DESIM considers the similarity-based mutation strategy, and was shown to outperform explorative and exploitative variants of DE [39]. The mutation is as follows:

$$\boldsymbol{v}_t = \boldsymbol{x}_t + F_d(\boldsymbol{x}^{si} - \boldsymbol{x}_t) + F_d(\boldsymbol{x}_t^{r_1} - \boldsymbol{x}_t^{r_2}) \tag{39}$$

$$si \sim U[si_l,\ si_u] \tag{40}$$

$$si_u = \frac{(NP-\delta)}{\Omega}\omega + \delta \tag{41}$$

$$si_l = si_u - \delta, \tag{42}$$

where $\boldsymbol{x}^{si}$ is the si-th vector from the *population sorted by similarity* in *descending order* with respect to the best (fittest) individual. Here, similarity is computed in terms of the Euclidean distance. Also, in the above, si is an integer uniformly sampled from $U[si_l, si_u]$, Ω is the maximum number of function evaluations, ω is the number of function evaluations at iteration t, δ is the intensification parameter ($\delta = \{5, 10, 15\}$ are found to be favorable [39]). Furthermore, the parameters F_d and CR are computed by the adaptation principles from JADE. Initialization of the population is realized by the *Opposition Based Learning*.

III. COMPUTATIONAL EXPERIMENTS

In order to evaluate the performance and the feasibility of our proposed approach, we performed a set of experiments in which our goal is to evaluate the convergence ability when minimizing the cost function, and the degree of the smoothness and curvature profiles of the obtained paths.

A. SETTINGS

The robot trajectories are realized by the Boe-Bot hardware [40] with a marker-based localization through an RGB camera @30FPS located on top. Our algorithms were implemented in Matlab 2018a, in which our computing environment consisted of i7-4930K @3.40GHz. The evaluated algorithms are labeled as follows:

- DERAND: DE/rand/1/bin strategy
- RBDE: Rank-Based Differential Evolution
- JADE: Adaptive Differential Evolution with External Archive
- SADE: Strategy Adaptation Differential Evolution
- DESIM: Differential Evolution with Similarity Based Mutation

Our main rationale in using the above described heuristics is due to our aim in evaluating whether initialization, selection pressure, parameter adaptation, exploration and exploitation play key roles in the competitive performance to fit and fair curves to robot trajectory data. Basically, our approach extends the conventional nonlinear least squares curve fitting problem to ordered data [41] by allowing the inclusion of a functional fairness to obtain monotonically varying robot trajectories. Although it is possible to extend the local algorithms for curve fairing [42], the conjugate gradient-based methods [43], and the nonlinear constrained optimization approaches such as Sequential Quadratic Programming [44], their performance is limited by the ability to approximate a local minimum, the differentiability of the cost function, the sequential nature of sampling solutions and the computational cost involved in computing gradients with respect to all degrees of freedom of the curve. Thus, our focus of interest span metaheuristics due to the nonlinearity of the cost function [41], [43], the inherent parallelization scheme to sample solutions through a population-based approach, and the potential to scape local optima, thus enabling to compute fit and fair curves in the global perspective. The evaluation of gradient-based and local heuristics is out of the scope of this paper, whose study and integration to metaheuristic approaches is left for future work in our agenda.

As for parameters in the above-mentioned metaheuristic algorithms, we used probability of crossover $CR = 0.5$, scaling factor $F_d = 0.7$, population size $NP = 10$ individuals, the bias term $\theta = 3$, initialization constant $\eta = 2$, interpolation constants for initial solutions $d_1 = d_2 = 2/3$, and the termination criterion is set to 2000 and 10000 function evaluations. Also, due to the stochastic nature of Differential Evolution, 20 independent runs were evaluated for each configuration. Other parameters followed RBDE [36], JADE [37], SADE [38], DESIM [39]. The key motivations of using the above parameters are as follows:

- Crossover probability with $CR = 0.5$ enables to consider equal importance to historical search directions.
- Small population size $NP = 10$ and number of evaluations up to 2000 and 10000 are used in order to evaluate the (competitive) performance of the gradient-free classes of Differential Evolution algorithms under tight evaluation budgets. This setting allows to evaluate the feasibility on the fast convergence of Differential Evolution during distinct modes of selection pressure.
- Also, as for initialization of the population, in order to allow the relevant initialization schemes with reasonable geometry of the search space, we use $\eta = 2$ and interpolation constants $d_1 = d_2 = 2/3$. The optimal selection of initialization parameters is out of the scope of the paper.

The parameter for smoothness preference in Eq. 8 was set at $\lambda = 0.01$, which showed the reasonable results, after a number of trials, in balancing the fitting error and the smoothness of curves. Fine-tuning of the above-described parameters is out of the scope of this paper.

Furthermore, the mobile robot (Boe-Bot hardware [40]) is able to generate trajectories as byproduct of navigation. Basically, by fixing the ratio of Pulse Width Modulation (PWM) on the servo motors, the expected trajectory is circumference; yet due to inherent noises in current, the mechanical configuration of the gearboxes and the interactions between the wheel and the floor, the realized trajectories are rather noisy. Thus, in order to collect real-world robot trajectories, we employed the following setup:

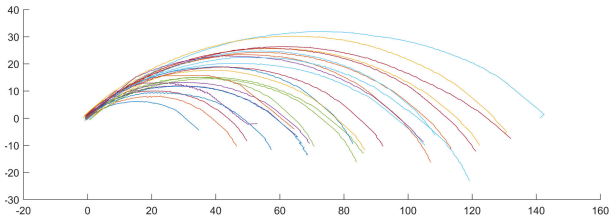


FIGURE 3. Overview of real-world trajectories from Boe-Bot robot hardware.

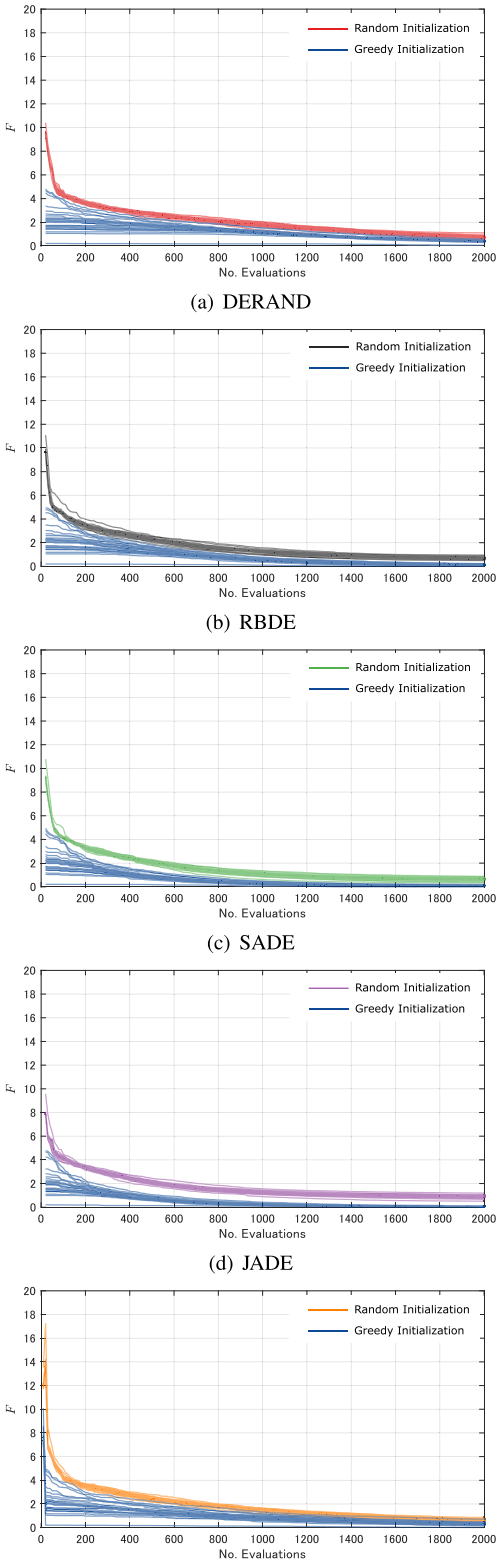
- Boe-Bot [40] uses discretized combinations of torque through Pulse-Width Modulation ratio of the servo motors of the robot.
- Pulse-Width Modulations are discretized by using a microcontroller (Arduino UNO) attached to the robot hardware.
- Trajectories are tracked by an RGB camera located on the ceiling, and markers located on top of the robot.

For simplicity and without loss of generality, we collected trajectories by using feasible values of Pulse-Width Modulations in one side of the servo motor. In order to show the kind of collected trajectories, Fig. 3 shows the set of coordinates and trajectories for all discretized Pulse-Width Modulations of the torque in the left side of the servo motor (inducing a curved motion to the right). Overall, 28 trajectories were collected, in which the origin was set at locations close to $(0, 0)$ in Fig. 3, and the end locations are positioned at the right side of the plot. The reader may note that the accurate positioning of the robot at $(0, 0)$ is irrelevant since curve modeling considers the relative coordinates of the point $\boldsymbol{q}_1$ (initial point in the curve $\boldsymbol{r}(u)$. We observed that the number of points m (Eq. 9) were variable in each trajectory, and the following range was obtained $m \in [311, 766]$, which is mainly due to the inherent nature of the noisy and curved paths.

B. CONVERGENCE PERFORMANCE

To evaluate the convergence ability of our proposed approach, Fig. 4 and Fig. 5 show the convergence of the minimization of the cost function (Eq. 8) under $\Omega = 2000$ and $\Omega = 10000$ function evaluations, respectively. In these figures, the x-axis denotes the number of function evaluations, and the y-axis denotes the value of the cost function F. Both Fig. 4 and Fig. 5 report the mean of the convergence of each of the 28 trajectories cases over 20 independent runs.

- Furthermore, in order to portray the effect of the greedy initialization on the convergence performance within the first $\Omega = 2000$ function evaluations, Fig. 4 shows the convergence of the cost function when minimizing Eq. 8 for 28 trajectories. Here, greedy initialization is depicted by blue color, whereas the convergence under random initialization is portrayed in red (DERAND), black (RBDE), green (SADE), purple (JADE) and orange (DESIM) colors.



(a) DERAND

(b) RBDE

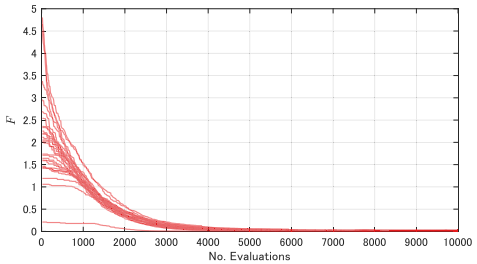
(c) SADE

(d) JADE

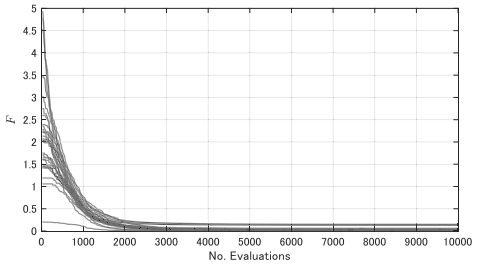
(e) DESIM

FIGURE 4. Convergence of the function F under $\Omega = 2000$ function evaluations.

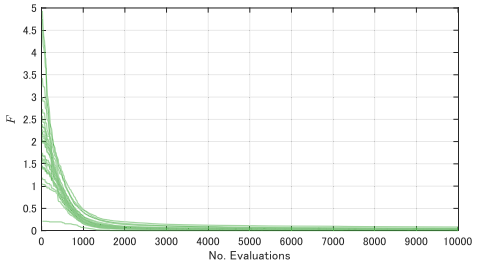
- Also, in order to show the convergence behaviour under large number of function evaluations, Fig. 5 shows the convergence under $\Omega = 10000$ function evaluations.



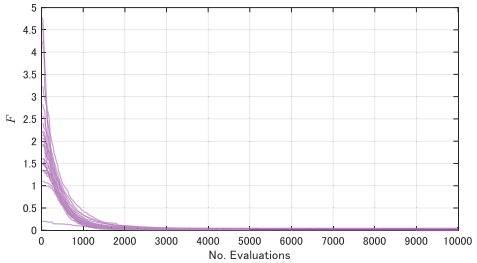
(a) DERAND



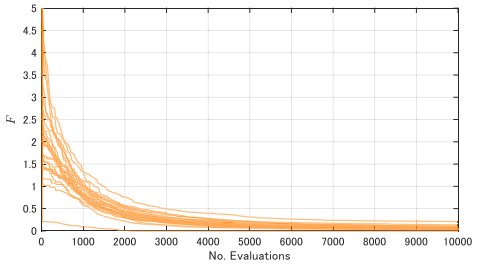
(b) RBDE



(c) SADE



(d) JADE



(e) DESIM

FIGURE 5. **Convergence of the function F under $\Omega = 10000$ function evaluations.**

By observing Fig. 4 and Fig. 5, we note the following facts:

- The number of iterations required for convergence is around 2000 in the best scenarios, in which RBDE, JADE and SADE show the best convergence speed.
- Reasonable convergence fulfilling the termination criteria occurs in all studied cases when minimizing the cost function over all collected trajectories and classes of Differential Evolution algorithms.
- As shown by Fig. 4, the improved convergence performance is observed within the stages of the search process when the population uses the greedy initialization scheme (Eq. 20 - Eq. 24).

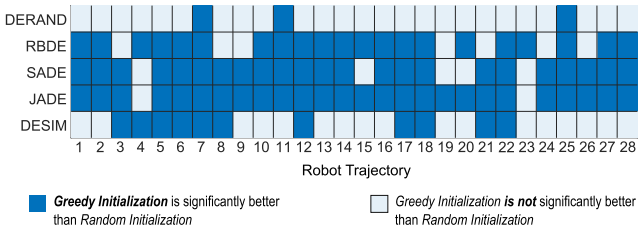


FIGURE 6. **Test to evaluate the statistical significance of using the greedy initialization scheme. Wilcoxon tests were performed at 5% significance level over 20 independent runs and overall robot trajectories.**

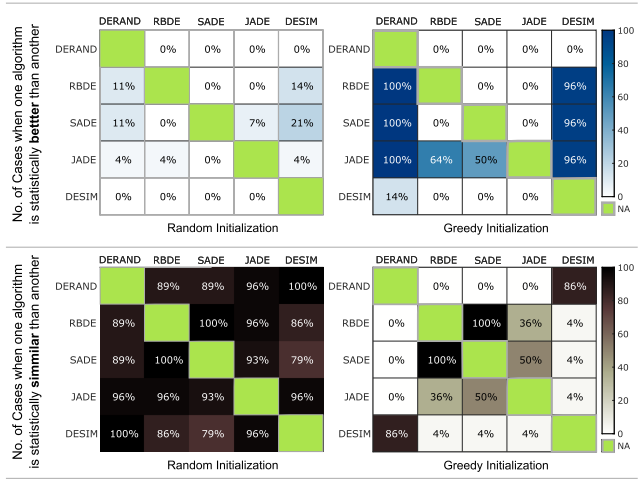
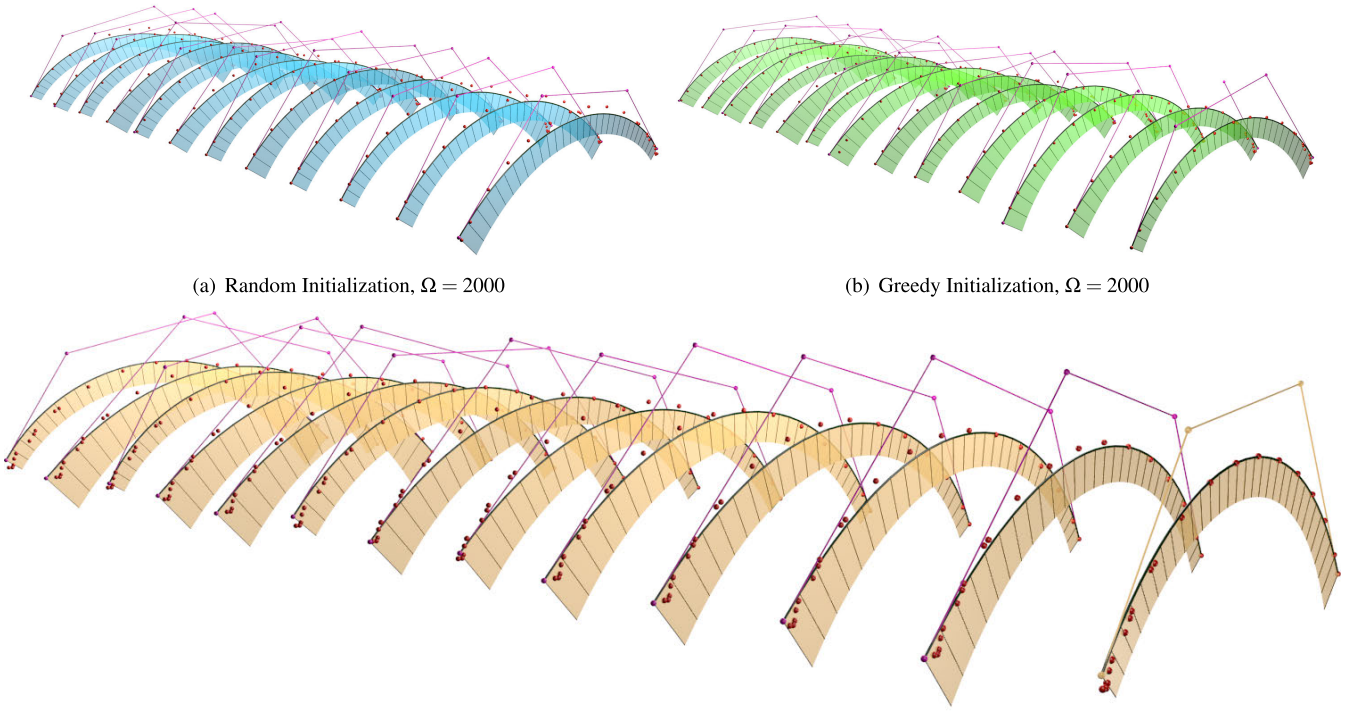


FIGURE 7. **Statistical test to evaluate the performance of Differential Evolution classes. Wilcoxon tests were performed at the 5% significance level over 20 independent runs and overall robot trajectories. On top (bottom): the number of cases in which an algorithm in the row is statistically better (similar) than an algorithm in the column. In the left (right) side: situations using random (greedy) initialization.**

The above-mentioned observations show the amenability of the Differential Evolution classes to find solutions within the basins of the cost function F, suggesting the ability to compute smooth curve configurations within small number of function evaluations.

In order to evaluate the statistical significance of the improvement of the greedy initialization scheme, Fig. 6 shows the Wilcoxon rank test at 5% significance level over the converged cost function F under $\Omega = 2000$ function evaluations. Here, the x-axis shows the robot trajectory scenario, and the y-axis shows the Differential Evolution class. As shown by Fig. 6, dark blue colors denote that the greedy initialization scheme (Eq. 20 - Eq. 24) is significantly better than the random initialization at 5% significance level. Also, the greedy initialization improves the performance significantly in RBDE (21 out of 28 cases), SADE (23 out of 28 cases) and JADE (26 out of 28 cases). However, no significant improvement was observed in DERAND



(a) Random Initialization, $\Omega = 2000$

(b) Greedy Initialization, $\Omega = 2000$

(c) Greedy Initialization, $\Omega = 10000$

FIGURE 8. Examples of smooth curves extracted from robot trajectories.

and DESIM. We believe that the greedy initialization is advantageous for better convergence in RBDE, SADE and JADE due to the exploitation abilities of these schemes compared to DERAND and DESIM. Likewise, the exploration mechanism of DERAND and DESIM in early generations enables to counter-balance the lack of greedy solutions.

Furthermore, to show the performance comparison among the Differential Evolution algorithms, Fig. 7 shows the statistical comparison of the converged cost function after $\Omega = 2000$ function evaluations. Here, Wilcoxon tests were performed at the 5% significance level over 20 independent runs and overall robot trajectories. By observing the results in Fig. 7 we observe the following facts:

- All studied Differential Evolution algorithms achieve similar performance when using random initialization.
- SADE was shown to outperform DESIM in 21% of the trajectory cases, yet found to achieve similar results in 79% of the trajectory cases.
- In greedy initialization, RBDE, SADE and JADE attained the best results, outperforming DERAND and DESIM. Also, DERAND and DESIM were found to attain similar results in 86% of the trajectory cases.
- When using greedy initialization, SADE and JADE achieved similar results in all trajectory cases. However, JADE was shown to outperform RBDE and SADE in 64% and 50% of the trajectory cases, respectively.

The above-mentioned facts confirm the feasibility and the efficiency for competitive convergence. Also, the above results show the advantageous properties of the greedy initialization to improve the convergence of Differential Evolution algorithms with exploitative and parameter adaptation mechanism such as RBDE, SADE and JADE.

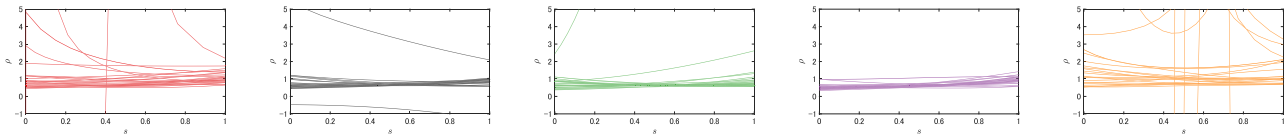
C. CURVATURE PROFILES

In order to show the kind of smooth curves which our proposed approach is able to compute, Fig. 8 shows examples of curves and curvature profiles rendered as a result of the minimization problem in Eq. 8. In these figures, the following elements are portrayed:

- A selected number of points in the robot trajectory are denoted by nodes colored in red.
- The rendered smooth curves approximating the robot trajectories are shown in dark color.
- The control points $\boldsymbol{p}_i$ of the curve $\boldsymbol{r}(u)$ are denoted by vertices and lines colored in purple.
- The radius of curvature are shown as curved shapes in blue, green and orange color, respectively.

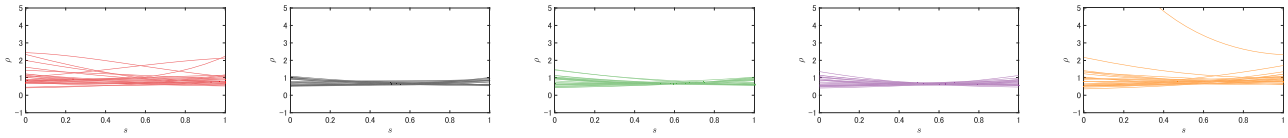
By observing Fig. 8, we can note that smooth curves reasonably fit the collected robot trajectories. Also, in order to show the properties of the curvature profiles and to portray the (variation of) curvature in all cases of smooth robot trajectory scenarios, Fig. 9 - Fig. 11 show the profiles of the attained smooth curves. Here, the x-axis denotes the arc-length of the smooth curve, and the y-axis denotes the value of the radius of curvature $\rho = \frac{1}{\kappa}$. By observing Fig. 9 - Fig. 11, we note the following facts:

- Under small number of evaluations ($\Omega = 2000$), RBDE, SADE and JADE attain a linear-like variation of radius



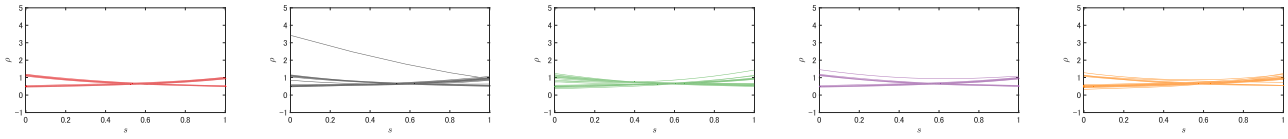
(a) DERAND (b) RBDE (c) SADE (d) JADE (e) DESIM

FIGURE 9. Radius of curvature profiles as a function of curve length using Random Initialization, $\Omega = 2000$.



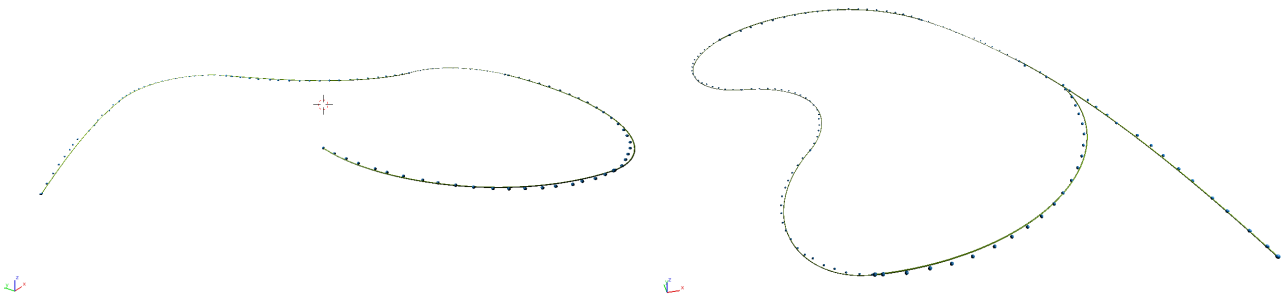
(a) DERAND (b) RBDE (c) SADE (d) JADE (e) DESIM

FIGURE 10. Radius of curvature profiles as a function of curve length using Greedy Initialization, $\Omega = 2000$.



(a) DERAND (b) RBDE (c) SADE (d) JADE (e) DESIM

FIGURE 11. Radius of curvature profiles as a function of curve length using Greedy Initialization, $\Omega = 10000$.



(a) Scenario 1 (b) Scenario 2

FIGURE 12. Examples of compounded curves from smooth robot trajectories.

of curvature over all studied cases of smooth robot trajectories.

- In particular, the linear-like behaviour is attined when the greedy initialization scheme is used. The above suggests a monotonically varying curvature, which is due to the presence of the smoothness factor H (Eq. 12).
- Under large number of evaluations ($\Omega = 2000$), all algorithms attain the linear-like variation of the radius of curvature over all studied cases.

The above observations confirm the feasibility of generating curves which not only are able to approximate robot trajectories reasonably, but also are able to show linear-like variation of curvature. These features are useful to suggest alternative robot trajectories which comply with linear variation of torque, and whose control laws are computationally efficient.

Furthermore, in order to exemplify the ability of compounding trajectories generated by smooth curves within the context of mobile robot navigation (Boe-Bot robot architecture), Fig. 12 shows examples of compounded trajectories by a plural number of fitted smooth curves. Here, curves are compounded considering C^0 continuity and by matching the gradient of the curve $\dot{\boldsymbol{r}}8(u)$ at the connecting node points. Fig. 12 shows the compounded path segments in green color and the prescribed input coordinates from the robot trajectory in blue-colored nodes. In scenario 1, 5 arbitrary path segments were used to generate an arbitrary compounded trajectory, whereas in scenario 2, 6 arbitrary path segments were used. By observing the profile in Fig. 12, we can note that it becomes possible to compute compounded smooth paths by using the fitted smooth trajectories. However, as expressed in section II, the compounded paths will require further pruning to satisfy the minimal cost function F at the node joints. Also, the compounded paths are unable to satisfy collision-free navigation. Our results offer the scheme to generate alternative navigation trajectories satisfying not only the closeness

to real-world achieved trajectories, which are byproduct of the interaction of the robot dynamics and the environment, but also showing the linear-varying curvature profile, which is relevant for comfortability and safety in navigation.

In future work, we aim at deriving efficient methods for collision-free path planning of smooth paths, and integrating with the proposed aesthetic B-spline Curves [45]–[48]. Also, in our agenda is the development of compact trajectories in the plane and its application in resource distribution and interaction of Multi-Agent Sytems. Our obtained results offer the building blocks to further advance towards developing data-driven path planning algorithms for field areas, which may find use in several real-world applications in Robotics, Operations Research and Multi-Agent Systems.

IV. CONCLUSION

In this paper, we have proposed an approach to generate smooth paths from observed robot trajectories by optimizing criteria for fitting and smoothness using Differential Evolution under distinct modes of initialization of the population, selection pressure, exploration and exploitation during sampling. In particular, we the DE/rand/1/bin strategy (DERAND), the Rank-Based Differential Evolution (RBDE), the Adaptive Differential Evolution with External Archive (JADE), the Strategy Adaptation Differential Evolution (SADE), and the Differential Evolution with Similarity Based Mutation (DESIM).

Our rigorous experiments using all feasible cases of pulse width modulation on one side of the (servo) motor of the Boet-Bot mobile robot architecture showed the feasibility of our approach to generate smooth curves efficiently, in which fast convergence to the basins of the search space occurs with the greedy initialization scheme and Differential Evolution with exploitative and parameter adaptation schemes, such as RBDE, SADE and JADE.

Our proposed approach is useful to suggest alternative trajectories for mobile robots complying with linear variation of the radius of curvature, and whose control realization would be computationally efficient, offering the data-driven planning mechanisms for comfortability in riding.

REFERENCES

[1] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 489–494.

[2] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4569–4574.

[3] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, Aug. 2014.

[4] L. Yu, D. Kong, X. Shao, and X. Yan, "A path planning and navigation control system design for driverless electric bus," *IEEE Access*, vol. 6, pp. 53960–53975, 2018.

[5] V. Parque and T. Miyashita, "Path planning on hierarchical bundles with differential evolution," in *Advances in Swarm Intelligence*, Y. Tan, Y. Shi, and Q. Tang, Eds. Cham, Switzerland: Springer, 2018, pp. 251–260.

[6] V. Parque, S. Miura, and T. Miyashita, "Route bundling in polygonal domains using differential evolution," *Robot. Biomimetics*, vol. 4, no. 1, p. 22, Dec. 2017.

[7] V. Parque, S. Miura, and T. Miyashita, "Optimization of route bundling via differential evolution with a convex representation," in *Proc. IEEE Int. Conf. Real-time Comput. Robot. (RCAR)*, Okinawa, Japan, Jul. 2017, pp. 727–732.

[8] V. Parque and T. Miyashita, "Towards bundling minimal trees in polygonal maps," in *Proc. Genetic Evol. Comput. Conf. Companion (GECCO)*. New York, NY, USA: ACM, Jul. 2018, pp. 1813–1820.

[9] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[10] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[11] M. Kallman, "Path planning in triangulations," in *Proc. Workshop Reasoning, Represent., Learn. Comput. Games (IJCAI)*, 2005, pp. 49–54.

[12] B. Chazelle, "A theorem on polygon cutting with applications," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*, Nov. 1982, pp. 339–349.

[13] D. T. Lee and F. P. Preparata, "Euclidean shortest paths in the presence of rectilinear barriers," *Networks*, vol. 14, no. 3, pp. 393–410, 1984.

[14] Q. H. Do, S. Mita, H. T. N. Nejad, and L. Han, "Dynamic and safe path planning based on support vector machine among multi moving obstacles for autonomous vehicles," *IEICE Trans. Inf. Syst.*, vol. E96.D, no. 2, pp. 314–328, 2013.

[15] L. Han, H. Yashiro, H. T. N. Nejad, Q. H. Do, and S. Mita, "Bézier curve based path planning for autonomous vehicle in urban environment," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 1036–1042.

[16] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[17] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 2775–2781.

[18] R. Cimurs, J. Hwang, and I. H. Suh, "Bezier curve-based smoothing for path planner with curvature constraint," in *Proc. 1st IEEE Int. Conf. Robotic Comput. (IRC)*, Apr. 2017, pp. 241–248.

[19] J. A. R. Silva and V. Grassi, "Clothoid-based global path planning for autonomous vehicles in urban scenarios," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4312–4318.

[20] A. Artunedo, J. Godoy, and J. Villagra, "A primitive comparison for traffic-free path planning," *IEEE Access*, vol. 6, pp. 28801–28817, 2018.

[21] D. Gonzalez, J. Perez, R. Lattarulo, V. Milanes, and F. Nashashibi, "Continuous curvature planning with obstacle avoidance capabilities in urban scenarios," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 1430–1435.

[22] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bezier curve based path planning in a dynamic field using modified genetic algorithm," *J. Comput. Sci.*, vol. 25, pp. 339–350, Mar. 2018.

[23] H. Andreasson, J. Saarinen, M. Cirillo, T. Stoyanov, and A. J. Lilienthal, "Fast, continuous state path smoothing to improve navigation accuracy," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 662–669.

[24] K. R. Simba, N. Uchiyama, and S. Sano, "Real-time obstacle-avoidance motion planning for autonomous mobile robots," in *Proc. 4th Austral. Control Conf. (AUCC)*, Nov. 2014, pp. 267–272.

[25] S. Upadhyay and A. Ratnoo, "Smooth path planning for passages with heading and curvature discontinuities," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 2672–2677.

[26] J. Choi and K. Huhtala, "Constrained path optimization with bézier curve primitives," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 246–251.

[27] E. Heiden, L. Palmieri, S. Koenig, K. O. Arras, and G. S. Sukhatme, "Gradient-informed path smoothing for wheeled mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1710–1717.

[28] A. Hashemian and S. F. Hosseini, "An integrated fitting and fairing approach for object reconstruction using smooth NURBS curves and surfaces," *Comput. Math. with Appl.*, vol. 76, no. 7, pp. 1555–1575, Oct. 2018.

[29] A. Gálvez, A. Iglesias, A. Avila, C. Otero, R. Arias, and C. Manchado, "Elitist clonal selection algorithm for optimal choice of free knots in B-spline data fitting," *Appl. Soft Comput.*, vol. 26, pp. 90–106, Jan. 2015.

[30] Y. Yuan, N. Chen, and S. Zhou, "Adaptive B-spline knot selection using multi-resolution basis set," *IIE Trans.*, vol. 45, no. 12, pp. 1263–1277, Dec. 2013.

[31] X. Zhao, C. Zhang, B. Yang, and P. Li, "Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation," *Comput.-Aided Design*, vol. 43, no. 6, pp. 598–604, Jun. 2011.
[32] K. Uyar and E. Ülker, "B-spline curve fitting with invasive weed optimization," *Appl. Math. Model.*, vol. 52, pp. 320–340, Dec. 2017.
[33] R. Interian, J. M. Otero, C. C. Ribeiro, and A. A. Montenegro, "Curve and surface fitting by implicit polynomials: Optimum degree finding and heuristic refinement," *Comput. Graph.*, vol. 67, pp. 14–23, Oct. 2017.
[34] A. Gálvez and A. Iglesias, "A new iterative mutually coupled hybrid GA–PSO approach for curve fitting in manufacturing," *Appl. Soft Comput.*, vol. 13, no. 3, pp. 1491–1504, Mar. 2013.
[35] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[36] A. M. Sutton, M. Lunacek, and L. D. Whitley, "Differential evolution and non-separability: Using selective pressure to focus search," in *Proc. 9th Annu. Conf. Genetic Evol. Comput. (GECCO)*. New York, NY, USA: ACM, 2007, pp. 1428–1435.
[37] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.
[38] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[39] E. Segredo, E. Lalla-Ruiz, and E. Hart, "A novel similarity-based mutant vector generation strategy for differential evolution," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*. New York, NY, USA: ACM, Jul. 2018, pp. 881–888.
[40] Parallax. (2019). *Boe-Bot*. Accessed: Mar. 12, 2019. [Online]. Available: https://www.parallax.com/product/boe-bot-robot
[41] C. F. Borges and T. Pastva, "Total least squares fitting of Bézier and B-spline curves to ordered data," *Comput. Aided Geometric Design*, vol. 19, no. 4, pp. 275–289, Apr. 2002.
[42] N. Sapidis and G. Farin, "Automatic fairing algorithm for B-spline curves," *Comput.-Aided Design*, vol. 22, no. 2, pp. 121–129, Mar. 1990.
[43] H. P. Moreton and C. H. Séquin, "Minimum variation curves and surfaces for computer-aided geometric design," *Designing Fair Curves Surfaces, Shape Qual. Geometric Model. Comput.-Aided Design*, vol. 6, Jan. 1994, pp. 123–159.
[44] Y. Wang, B. Zhao, L. Zhang, J. Xu, K. Wang, and S. Wang, "Designing fair curves using monotone curvature pieces," *Comput. Aided Geometric Design*, vol. 21, no. 5, pp. 515–527, May 2004.
[45] M. Higashi, V. Parque, M. Kobayashi, and T. Oya, "Aesthetic B-spline curves and surfaces: Formulation and experiments for curves," *Proc. JSPE Semestrial Meeting*, vol. 2014, pp. 1125–1126, Mar. 2014, doi: 10.11522/pscjspe.2014S.0_1125.
[46] M. Higashi, V. Parque, M. Kobayashi, and S. Tsuchie, "Aesthetic b-spline curves and surfaces (2nd report): Generation of space curves and surfaces," *Proc. JSPE Semestrial Meeting*, vol. 2014, pp. 519–520, Mar. 2014.
[47] M. Higashi, M. Kobayashi, V. Parque, and S. Tsuchie, "Aesthetic B-spline curves and surfaces (3rd report): Curvature of surface and application to compound curves and surfaces," *Proc. JSPE Semestrial Meeting*, vol. 2015, pp. 453–454, Mar. 2015, doi: 10.11522/pscjspe.2015S.0_453.
[48] V. Parque, M. Kobayashi, and M. Higashi, "Comparative analysis of aesthetic surfaces," *Proc. JSPE Semestrial Meeting*, vol. 2016, pp. 191–192, Mar. 2016, doi: 10.11522/pscjspe.2016S.0_191.

VICTOR PARQUE (Member, IEEE) received the B.Sc. degree in systems engineering from National Central University, in 2004, the M.B.A. degree from the Graduate School of Business Administration, Esan University, in 2009, and the Ph.D. from the Graduate School of Information, Production and Systems, Waseda University, in 2011. He was a Postdoctoral Fellow with the Department of Mechanical Engineering, Toyota Technological Institute, from 2012 to 2014, and an Assistant Professor at Waseda University, from 2014 to 2018. He is currently an Associate Professor with the Department of Modern Mechanical Engineering, Waseda University, and the JSUC Tokunin Professor with the Egypt-Japan University of Science and Technology. He is the first author of more than 70 articles in journals, conferences, and book chapters, and is actively involved in research collaborations with both industry and academia. His research interests span the principles of Learning and Intelligent Systems and its applications to Design Engineering, Planning and Control. He was honored as a Finalist in the Hummies Awards for Human-Competitive Results, in 2018. He is a member of the IEEE (RAS, IES, and SMC), ACM (SIGAI and SIGEVO), Robotics Society of Japan (RSJ), and Japan Society for Precision Engineering (JSPE).

TOMOYUKI MIYASHITA (Member, IEEE) received the Ph.D. degree from Waseda University, in 2000. He was with Nippon Steel Company, Ltd., in 1992. He is currently a Professor with the Department of Modern Mechanical Engineering, Waseda University. Since 2000, he has been a Research Associate with Waseda University, and with Ibaraki University, since 2002. In 2005, he became an Associate Professor at Waseda University, where he also became a Professor, in 2007. His research interests are design process, design optimization, organ deformation.