Car Racing Line Optimization with Genetic Algorithm using Approximate Homeomorphism

Jaroslav Klapálek*, Antonín Novák*, Michal Sojka and Zdeněk Hanzálek

***Abstract*— In every timed car race, the goal is to drive through the racing track as fast as possible. The total time depends on selection of the racing line. Following a better racing line often decides who wins. In this paper, we solve the optimal racing line problem using a genetic algorithm. We propose a novel racing line encoding based on a homeomorphic transformation called Matryoshka mapping. We evaluate the fitness of racing lines by lap time estimation using a vehicle model suitable for F1/10 autonomous racing competition. By comparing to the former state-of-the-art, we show that our method is able to find racing lines with lower lap times. Specifically, on one of the testing tracks, we achieve 2.5% improvement.**

I. INTRODUCTION

The optimal racing line problem is of a common interest in an automotive racing environment. It resides in finding a trajectory, which allows the vehicle to drive through the track in the minimum possible time. Lap time minimization is a basic prerequisite for winning an F1/10 Autonomous Racing Competition [1], [2], an international event organized by the University of Pennsylvania. This competition is held at least twice a year, collocated with major conferences. The name comes from the vehicles used within the competition – 1:10 scaled-down car models, built from predefined hardware components, as shown in Fig. 1.

Our team won the F1/10 competition in Porto (2018) with a relatively simple reactive algorithm called Follow the Gap (FTG) [3], with our implementation publicly available at [4]. After that the FTG algorithm became quite popular, as it was used and adapted by many other teams [5], [6]. On the other hand, FTG suffers from a few limitations, namely it fails in complicated turns and dead ends, since it does not take advantage from the knowledge of the racing track map, despite it being known before the race. The objective of this paper is to propose and evaluate an innovative racing line planning algorithm, that benefits from the knowledge of the track.

Two common approaches to finding an optimal racing line are [7]: (i) minimize the path length so that the distance to be traveled is shorter; or (ii) find a path that minimizes the path curvature so that the speed allowed by the surface friction limits at any point is maximized. However, as noted in [8], [9], even the combination of these two approaches may not lead to the optimal racing line. Other approaches for finding an optimal racing line use, e.g., Euler spirals [10], Model Predictive Control [11], or genetic algorithms (GA) [7]. In this paper, we solve the optimal racing line problem using a genetic algorithm. Even though we focus mainly on scaled-down model racing, the proposed approach can be used for full-sized autonomous racing cars as well.

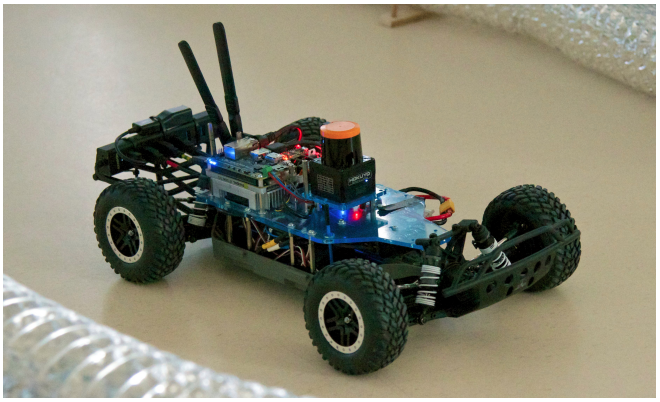


Fig. 1: Our F1/10 model car.

The main contributions of this paper are (i) reformulation of lap time minimization as an unconstrained problem via coordinate transformation by using proposed homeomorphic mapping, also called Matryoshka mapping algorithm, and (ii) reduction of lap time compared to the former state-of-the-art [7].

The paper is organized as follows: in Section II we survey previous research in the area of racing line optimization using genetic algorithms, in Section III we define terms and notation used within this paper. Section IV introduces our novel algorithm based on the Matryoshka coordinate transformation, in Section V we describe the evaluation procedure and discuss the results, and finally, we conclude this paper in Section VI.

II. RELATED WORK

Braghin et al. [7] describe the racing track by its discretization into n equidistantly spaced segments. Points P_i, $i \in \{1, 2, \ldots, n\}$ of the racing line are located at the boundaries of these segments (see magenta lines in Fig. 8). Their location is given by the index of the segment, and the position relative to the track borders. The resulting racing line is obtained by interpolating these points with a closed cubic spline. Authors try to find an optimal racing line as

*The authors are with Faculty of Electrical Engineering, Department of Control Engineering, Czech Technical University in Prague, Technická 1902/2, 166 27 Prague, Czech Republic

All authors are with Czech Institute of Informatics, Robotics and Cybernetics (CIIRC), Czech Technical University in Prague, Jugoslávských partyzánů 1580/3, 160 00 Prague, Czech Republic (email: klapajar@fel.cvut.cz, antonin.novak@cvut.cz, michal.sojka@cvut.cz, zdenek.hanzalek@cvut.cz)

a combination of the shortest path (SP) and the minimum curvature path (MCP) that minimizes the lap time. For the lap time estimation, they use a 14 DoFs vehicle model [12].

Cardamone et al. [13] extend the previous work [7] by finding a locally optimal combination of SP and MCP instead of one global solution as in the original work. Each point P_i of the racing line is described by its lateral distance from a track border and also by a distance from the track's starting line, which is computed along the track centerline. When looking for an optimal racing line, SP and MCP are found first. Intersections of SP and MCP are used for track segmentation. Since each point is described by its distance from the start, for each point in the SP, a corresponding point in the MCP can be found. Points of the racing line are then found as a combination of SP and MCP using the segment's combination parameter. The authors find the best parameters by several methods, one of them being a genetic algorithm. The quality of the racing line is obtained by running a simulation in The Open Racing Car Simulator (TORCS) [14] and measuring lap time there.

In follow-up work, Botta et al. [15] use the same approach as Braghin [7]. They segment the track into segments. However, in contrast to the work by Cardamone at al. [13], the segment size varies according to the track curvature. Points located on the segments' ends represent control points of connected Bézier curves. A genetic algorithm is used to minimize the lap time by moving the control points along the segment edge. The quality of the racing line is calculated by two approaches: (i) by simulating vehicle in TORCS (as in [13]), or (ii) by a lap time estimator.

Vesel [8] builds upon the previous approach. Racing line points P_i are described using radial coordinates from track origin. The racing line is obtained by calculating a periodic smoothing spline. This type of spline ensures a smooth transition between the lines' start and endpoints. A genetic algorithm is used to minimize lap time by moving the points in their vicinity. The lap time of the racing line is estimated by calculating its speed profile, similarly to [7].

Research done in lap time optimization using genetic algorithms is concentrated on improving the representation of the racing line, track segmentation being the main point of interest. At first, equidistant segmentation was proposed in [7], followed by its adjustment based on track curvature in [15]. However, [15] showed curvature-based segmentation only with Bézier curves and achieved worse results than the work by Cardamone at el. [13], which focused only on finding the optimal solution as a combination of SP and MCP. In addition, calculating the track curvature might be difficult for a F1/10 competition track, as it is not as smooth as tracks from TORCS. Both tracks are shown in Fig. 5. Since Vesel's [8] approach leads to many invalid positions of the racing line points (by design), we have selected the approach by Braghin et al. [7] as a reference approach against which we compare our results.

III. Preliminaries

In this section, we define terms and notation used within this paper. We also define the problem solved in this paper.

A. Definitions

Racing track T is a connected subset of $\mathbb{R}^2$, i.e., $T \subseteq \mathbb{R}^2$. We assume the racing track to be shaped like a typical automotive racing track such as those depicted in Fig. 5. Specifically, there are no crossroads and obstacles on the road. In addition, to compensate for the size of the vehicle driving on the track, we assume that T is shrunk such that the vehicle does not collide with the track border when vehicle's center point is contained in T.

Racing line L is a curve in $\mathbb{R}^2$, i.e., $L : [0,1] \to \mathbb{R}^2$. Note that we parameterize the line by normalized length p, $0 \leq p \leq 1$, $p \in \mathbb{R}$. Racing line L is said to be *feasible* if it lies completely within the track, i.e., $\forall p \in [0,1] : L(p) \in T$. For our application, we assume L to be a closed curve, i.e., $L(0) = L(1)$.

We define racing line *waypoints* as a sequence $P = (P_1, \ldots, P_k)$ of k points $P_i = L(p_i)$, $i = 1, \ldots, k$, such that $p_1 = 0$, $p_k = 1$, and $\forall i \in \{1, \ldots, k-1\} : p_i < p_{i+1}$. Due to closed line assumption $P_k = P_1$ and the racing line can be described by just $n = k - 1$ waypoints. The racing line can be obtained from waypoints by interpolation function I such that $L = I(P)$. In this paper, we use periodic cubic splines as the interpolation function I since it guarantees that the resulting L is a smooth closed curve.

The quality of racing line L is given by *lap time*, which is calculated by function

$$F : (\mathcal{L}, A) \to \mathbb{R}^+, \tag{1}$$

where $\mathcal{L}$ is a set of all feasible racing lines and A are the parameters of the vehicle following the L. Typically, A consists of the initial velocity, maximum velocity, maximum longitudinal acceleration, maximum braking deceleration, maximum lateral acceleration, and maximum turning radius. The maximum turning radius of the vehicle is reflected by the function F. We describe the function F used for our evaluation in Section IV-D.

Optimal racing line L^* is a feasible racing line minimizing the lap time for the given vehicle parameters A:

$$L^* = \arg\min_{L \in \mathcal{L}} F(L, A). \tag{2}$$

B. Problem Statement

We solve the optimal racing line problem for a given track T and vehicle parameters A: initial velocity, maximum velocity, maximum forward acceleration, maximum braking deceleration, maximum lateral acceleration (related to track curvature and tire-road friction to avoid side slip, specifically, we use friction circle) , and maximum turning radius.

C. Genetic algorithm

Genetic Algorithm (GA) [16] is a biologically-inspired population-based metaheuristic. GA works with the so-called

population, which is a set of *candidate* solutions, sometimes also called individuals.

In general, every GA works as follows: (i) initial set of candidates is constructed, (ii) each candidate is evaluated by a *fitness value*, (iii) a subset of candidates is selected based on their fitness value, and (iv) genetic operators (e.g., recombination and mutation) are applied to the selected candidates. The whole procedure repeats from step (ii) until some candidate satisfies minimum criteria or allocated time/iteration *budget* is exhausted.

IV. Racing Line Optimization

In this section, we describe the GA-based algorithm to solve the optimal racing line problem (2). The algorithm consists of several steps and transformations which are described in the following subsections.

A. Track segmentation

The first step of our racing line optimization algorithm consists of partitioning the track into n segments as illustrated in Fig. 2. The *segments* S_i cover the whole track, i.e., $T = \bigcup_{i=1}^{n} S_i$ and do not overlap, i.e., $S_i \cap S_j = \emptyset$, $i \neq j$. To segment the track, we first find its centerline and place n seed points equidistantly along with it. Then we apply the *flood fill algorithm* (also known as tint fill [17]) to create the segments from the seed points. The order of segments is the same as the order of seed points along the centerline.

Note that our definition of segments differs from the one proposed in [7], [13], where they consider the segments to be 1D lines cutting the track, whereas we consider the segments to be 2D shapes.

The number of segments n can be chosen manually, depending on the complexity of the track (e.g., 2 segments per turn and 1 segment per straight section), or can even be treated as another optimization variable. In Section V, we show that our approach allows choosing lower values than previous approaches, leading to better solutions within the given optimization budgets.

B. Racing line encoding

To encode the racing line for the genetic algorithm, we represent it with n waypoints P_i, $i \in \{1, \dots, n\}$ such that $P_i \in S_i$. The sequence P of these waypoints forms a candidate solution for the genetic algorithm. The racing line is obtained from P by interpolating it with a cubic spline denoted as $L = I(P)$. We use Cubic splines because their curvature is smooth [8], as opposed to, e.g., Bézier curves. We use the lap time (1) as the fitness value f of candidate P, i.e., $f = F(I(P), A)$.

With this encoding, one would need to solve the following constrained optimization problem:

$$\min_{P_i \in \mathbb{R}^2} F(I(P_1, \dots, P_n), A), \tag{O1}$$

$$\text{subject to: } P_i \in S_i, \qquad \forall i \in \{1, \dots, n\}. \tag{O2}$$

However, the constraints (O2) would be difficult to represent in a form suitable for the genetic algorithm. If the constraints are not represented accurately, the genetic algorithm will produce a lot of infeasible candidates, leading to inefficient operation. To avoid these difficulties, we introduce mapping $H_i : [0,1]^2 \to S_i$, $i \in \{1, \dots, n\}$ that maps a unit square to the individual segments. By using the mapping H in the problem (O1)–(O2), we get the problem (Q) with simple box constraints:

$$\min_{P'_i \in [0,1]^2} F(I(H_1(P'_1), \dots, H_n(P'_n)), A), \tag{Q}$$

which is a more suitable formulation to be solved with a genetic algorithm under the conditions discussed below.

In order to make the problem (Q) equivalent to (O1)–(O2), the mapping H_i must be bijective so that all points of S_i are represented in $[0,1]^2$ and vice versa. Moreover, if the mapping is smooth, it helps with the convergence of the genetic algorithm. Note that smooth bijection is called *homeomorphism* [18].

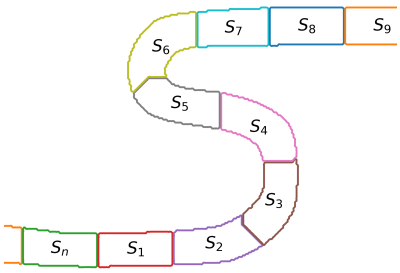


Fig. 2: Example of track segmentation.

C. Matryoshka mapping

Defining homeomorphism H_i for arbitrarily shaped segments would be difficult. Therefore, we propose to approximate it with *Matryoshka mapping*, denoted as $\hat{H}_i : [0,1]^2 \to S_i$. The mapping is constructed as shown in Algorithm 2: We cover both the track segment and the unit square with R points formed in so-called *rings* such that each ring point $u_j = (u_j^{(x)}, u_j^{(y)})$, $j \in \{1, \dots, R\}$, in the unit square has a corresponding ring point $s_j = (s_j^{(x)}, s_j^{(y)})$ in the segment, as illustrated in Fig. 3. The border of the mapped area represents the outermost ring. Inner rings are constructed by scaling the outermost ring around its center. Then, we create a smooth approximation of the mapping that maps points u_j to s_j. We find such a mapping by approximating it with two B-spline surfaces f_x and f_y found by BISPLREP function from SCIPY Python module. In other words, Matryoshka mapping for the segment S_i is defined $\forall u^{(x)}, u^{(y)} \in [0,1]$ as

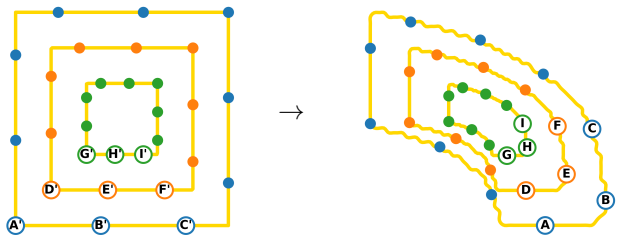
$$\hat{H}_i(u^{(x)}, u^{(y)}) = \left(f_x(u^{(x)}, u^{(y)}), f_y(u^{(x)}, u^{(y)}) \right), \tag{3}$$

where f_x and f_y are defined as:

$$f_x = \text{BISPLREP}(\{(u_j^{(x)}, u_j^{(y)}, s_j^{(x)}) \mid j \in \{1, ..., R\}\}), \tag{4}$$

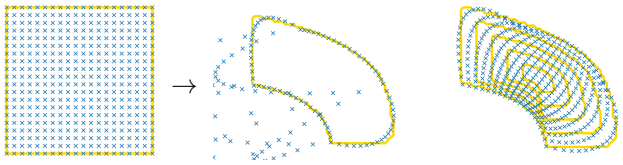
$$f_y = \text{BISPLREP}(\{(u_j^{(x)}, u_j^{(y)}, s_j^{(y)}) \mid j \in \{1, ..., R\}\}). \tag{5}$$

Matryoshka approximation $\hat{H}$ does not fulfill all properties of homeomorphism H, specifically: (i) target range of $\hat{H}_i$ is not exactly S_i due to its approximate character, and (ii)



(a) (b)

Fig. 3: Identifying corresponding pairs between (a) unit square and (b) segment during construction of the Matryoshka mapping (three rings with $R = 30$ points in total).



(a) (b) (c)

Fig. 4: Example of Matryoshka mapping $\hat{H}$: 400 evenly spaced blue points from (a) unit square are mapped by $\hat{H}$ constructed from: (b) 1 ring (only part is shown), and (c) 5 rings.

$\hat{H}_i$ may not be surjection, i.e., two distinct points from $[0,1]^2$ may map to the same point in S_i. These properties decrease the efficiency of the genetic algorithm. However, we limit the effect of both (i) and (ii) by using multiple rings during the $\hat{H}_i$ construction. This is shown in Fig. 4. In practice, Matryoshka mapping worked the best among the evaluated approximations of H_i. Other approaches, e.g., using morphological operators for the construction of the rings, led to worse results.

In summary, to solve the problem (Q), we introduce the 2D Matryoshka mapping algorithm (denoted as *2D-MM*), which is defined in Algorithm 1 and its subroutine in Algorithm 2.

D. Speed profile calculation

We estimated the lap time of a given racing line using a two-pass iterative algorithm introduced by Kapania et al. [19]. A similar approach was used in [7], [8].

This two-pass iterative algorithm calculates a speed profile of the racing line, given parameters of the vehicle A. For modeling the vehicle, we use a single-track model [20], as it is able to describe an F1/10 car.

The algorithm works as follows. Its input is a racing line and vehicle parameters. First, the algorithm processes the racing line in a backward direction from the end to the start, calculating maximum permissible vehicle speed (according to the racing line curvature), limiting it by friction and backward acceleration limit. Second, the algorithm goes in a forward direction from the start to the end of the racing line, lowering the vehicle speed even more to obey the vehicle's maximum forward acceleration.

Note that the F1/10 competition ranks the cars based on the shortest lap time and the number of completed laps during the fixed time slot. Therefore, we are interested in minimizing the "steady state" lap time, rather than the lap time of the first lap. Since the speed profile calculated by this two-pass iterative algorithm depends on the initial speed, we concatenate three identical racing lines (i.e., three laps) together, calculate the speed profile of the concatenated line, and use only the speed profile of the middle part. This way, the effect of the initial speed is suppressed.

Algorithm 1 2D-MM

Input: T, n, A, ringCount, ringLength, budget
Output: L^*

```
S ← segmentateTrack(T, n)
for i = 1 to n do
    Ĥ_i ← constructMatryoshka(S_i, ringCount, ringLength)
end for
L* ← ∅, f* ← +∞
P ← initializeCandidate()
repeat
    L ← I(Ĥ_1(P_1), ..., Ĥ_n(P_n))
    if ∀s ∈ [0,1] : L(s) ∈ T then
        v ← calculateSpeedProfile(L, A)
        f ← getLapTime(v)
    else
        f ← 100× countInfeasiblePoints(L, T)
    end if
    if f < f* then
        L* ← L, f* ← f
    end if
    P ← applyGeneticOperators(P, f)
until budget exhausted
```

Algorithm 2 constructMatryoshka

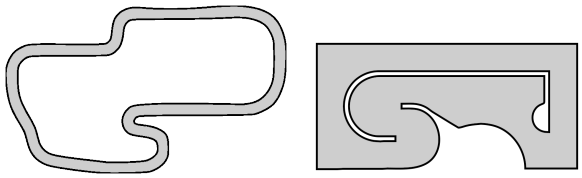
Input: S_i, ringCount, ringLength
Output: $\hat{H}_i$ mapping

```
B_seq ← border(S_i)
B_usq ← border([0,1]^2)
pairs ← ∅
for j = 0 to ringCount − 1 do
    ringScale ← 1 − (j/ringCount)
    B_segScaled ← scale(B_seg, ringScale)
    B_usqScaled ← scale(B_usq, ringScale)
    P_seg ← sample(B_segScaled, ringLength)
    P_usq ← sample(B_usqScaled, ringLength)
    pairs ← pairs ∪ (P_seg, P_usq)
end for
Ĥ_i ← findSurfaceRepresentation(pairs)
```

E. Implementation

We have implemented the 2D-MM algorithm from Section IV in Python 3. Path interpolation I was implemented with *CubicSpline* function from SCIPY module.

The genetic algorithm is implemented using Nevergrad [21], a toolbox for performing gradient-free optimiza-



(a) (b)

Fig. 5: Tracks used for evaluation: (a) Ruudskogen track, and (b) Torino track.

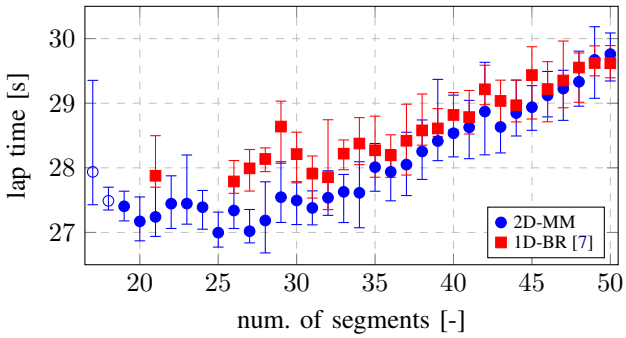


Fig. 6: Results on the Ruudskogen track (mean, min/max).

tion. Specifically, we use its *DoubleFastGADiscreteOnePlusOne* optimizer, which is a *Discrete One Plus One* genetic optimizer [22] with heavy-tailed mutation operator [23].

V. Experimental evaluation

We compare our 2D-MM algorithm with Braghin's [7] approach denoted as 1D-BR. We run a set of experiments on two tracks depicted in Fig. 5. The track in Fig. 5a is the same as the track used for comparison in [7]. The track from Fig. 5b is a race track built during F1/10 Autonomous Racing Competition in Torino, Italy, 2018.

The reference 1D-BR approach [7] works as follows. First, n lines (called cuts) perpendicular to the centerline are placed equidistantly along the track (see magenta lines in Fig. 8). Note that the imperfect perpendicularity of the cuts is caused by low resolution of the map used for centerline generation. These cuts are defined by their two endpoints, and each waypoint P_i is thus defined by its position between those endpoints, represented as a number between 0 and 1. Hence, 1D name stands in its name.

We run both 2D-MM and 1D-BR algorithms on both tracks, varying only the number of segments. The number of segments starts at the lowest number for which one of the approaches was able to find a feasible racing line and was determined experimentally. The budget parameter of the genetic algorithm was set to 4000 iterations. Each run of the genetic algorithm for a given number of segments was repeated 10 times.

A. Results

The results in Figs. 6 and 7 show an average lap time as well as minimum and maximum from 10 runs. Situations in which the algorithm was not able to find a feasible racing line in all 10 runs are marked by an empty marker. If the algorithm did not find any solution in 10 runs, no marker is shown at all.

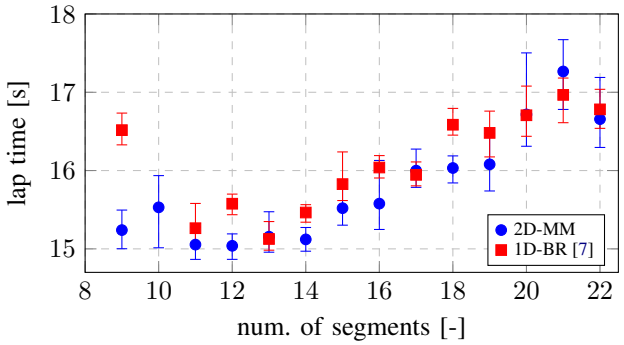


Fig. 7: Results on the Torino track (mean, min/max).

For the Ruudskogen track (Fig. 6), our 2D-MM algorithm was able to find a solution for any number of segments. The 1D-BR algorithm did not find a single solution for $17, 18, 19, 20, 22, 23, 24, 25$ segments. With 35 segments or more, both algorithms tend to produce solutions with larger lap time as the provided budget is not sufficient for finding a better one. Indeed, the choice of the number of segments leads to interesting trade-offs between the solution quality and the computational difficulty. Introducing more segments increases the dimensionality of the problem, thus it gets more computationally difficult. At the same time, as it gets more degrees of freedom, it also may lead to a solution of better quality. However, when the budget for the optimization is not large enough, the resulting solution with more segments may be actually worse.

Fig. 6 also shows that our 2D-MM was also able to find a solution with a lower lap time than the 1D-BR algorithm. The difference between the best solutions (2D-MM with $n = 28$ and 1D-BR with $n = 32$) was $0.671\,\mathrm{s}$ (2.5%). The most notable difference for the same number of segments was $1.259\,\mathrm{s}$ between the best solutions with 28 segments, and $1.093\,\mathrm{s}$ between the averages with 29 segments.

On the Torino track (Fig. 7), our 2D-MM was able to find a solution in all runs, whereas 1D-BR did not find any solution for 10 segments. Similar to the previous track, our 2D-MM found a solution with a lower lap time. The difference between the best solutions (2D-MM with $n = 11$ and 1D-BR with $n = 13$) was $0.118\,\mathrm{s}$. When looking at the differences per number of segments, both the largest differences between best solutions and the average lap time are for 9 segments, $1.327\,\mathrm{s}$ and $1.276\,\mathrm{s}$, consecutively.

B. Discussion

As notable from the results, our 2D-MM algorithm is able to find a racing line with a lower lap time than the 1D-BR algorithm. Moreover, our 2D-MM algorithm is more successful at finding feasible solutions for a low number of segments, whereas the 1D-BR algorithm sometimes fails to do so. The reason can be seen in Fig. 8, where we show the solution space for the 1D-BR algorithm with 20 cuts. The

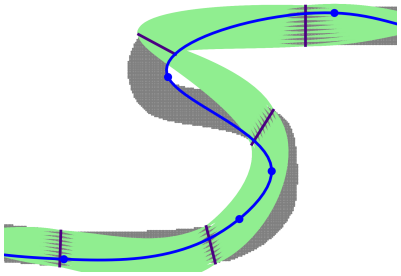


Fig. 8: Problematic part of the 1D-BR solution space on Ruudskogen track (gray) with 20 segments (magenta lines). The set of the possible solutions is show with green lines (all of them infeasible). For comparison, a feasible solution found by 2D-MM is shown in blue.

algorithm did not find any feasible racing line because the positions of the cuts (magenta lines in the figure) always lead to infeasible interpolations (green).

An example of how the 2D segmentation can help in finding a racing line, especially in turns, is shown in Figures 9a and 9b, where the blue points produced by 2D-MM lie further away from the cuts of 1D-BR. This is most visible in the bottom S-shaped section of the track.

VI. Conclusion

We introduced and implemented a racing line optimization algorithm 2D-MM that represents the racing line using a homeomorphic transformation. Using the genetic algorithm, we showed that the algorithm is able to overcome the prior 1D-BR algorithm [7] by (i) finding a racing line with lower lap time by $0.671\,\mathrm{s}$ resp. $0.118\,\mathrm{s}$ on the tested tracks, (ii) finding a feasible racing line with a lower number of segments (17 vs. 21 segments on the Ruudskogen track), (iii) being less prone to the placement of segments (e.g., 1D-BR is unable to find a feasible racing line for 22 segments on Ruudskogen track).

While the lap time difference might seem small, in the F1/10 Competition even a difference like $0.07\,\mathrm{s}$ can decide between two places in the ranking.

In future work, we will explore the possibilities of automatically estimating the number of segments, as starting with a properly selected number of segments speeds up the optimization process.

Also, we will enhance the segmentation to exploit the track layout, similarly to [15]. While increasing the number of segments increases the average lap time, moving the segments into the turns should: (i) increase the ratio of candidates that form a feasible racing line (when focused on the turns), (ii) generally decrease the lap time given iteration budget.

Acknowledgement

Research leading to these results has received funding from the EU ECSEL Joint Undertaking and the Ministry of Education of the Czech Republic under grant agreement 826452 and 8A19011 (project Arrowhead Tools).

References

[1] F1TENTH Foundation, "F1tenth." https://f1tenth.org. Accessed on: March 5, 2021.

[2] A. Agnihotri, M. O'Kelly, R. Mangharam, and H. Abbas, "Teaching Autonomous Systems at 1/10th-scale: Design of the F1/10 Racecar, Simulators and Curriculum," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, (New York, NY, USA), pp. 657–663, Association for Computing Machinery, Feb. 2020.

[3] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: "Follow the Gap Method"," *Robotics and Autonomous Systems*, vol. 60, pp. 1123–1134, Sept. 2012.

[4] A. S. Pedersen and J. Klapálek, "Follow The Gap." https://github.com/CTU-IIG/f1t-ftg, 2020. Accessed on: March 5, 2021.

[5] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, and M. Bertogna, "F1/10: An Open-Source Autonomous Cyber-Physical Platform," *arXiv:1901.08567 [cs]*, Jan. 2019. arXiv: 1901.08567.

[6] V. S. Babu and M. Behl, "f1tenth.dev - An Open-source ROS based F1/10 Autonomous Racing Simulator," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 1614–1620, Aug. 2020. ISSN: 2161-8089.

[7] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni, "Race driver model," *Computers & Structures*, vol. 86, pp. 1503–1516, July 2008.

[8] R. Vesel, "Racing line optimization @ race optimal," *SIGEVOlution*, vol. 7, pp. 12–20, Aug. 2015.

[9] N. R. Kapania, *Trajectory planning and control for an autonomous race vehicle*. PhD Thesis, Standford University, Department of Mechanical Engineering, 2016.

[10] Y. Xiong, "Racing Line Optimization," Master's thesis, Massachusetts Institute of Technology, Sept. 2010.

[11] R. Verschueren, S. Bruyne, M. Zanon, J. Frasch, and M. Diehl, "Towards Time-Optimal Race Car Driving Using Nonlinear MPC in Real-Time," in *53rd IEEE Conference on Decision and Control*, vol. 2015, Dec. 2014.

[12] F. Cheli, E. Leo, F. Mancosu, and S. Melzi, "A 14 dof Model for the Evaluation of Vehicle's Dynamics, Numerical-Experimental Comparison," *Meccanica*, vol. 41, pp. 35–43, Jan. 2006.

[13] L. Cardamone, D. Loiacono, P. L. Lanzi, and A. P. Bardelli, "Searching for the optimal racing line using genetic algorithms," in *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pp. 388–394, Aug. 2010. ISSN: 2325-4289.

[14] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "TORCS, The Open Racing Car Simulator." http://www.torcs.org, 2014. Accessed on: March 5, 2021.

[15] M. Botta, V. Gautieri, D. Loiacono, and P. L. Lanzi, "Evolving the optimal racing line in a high-end racing game," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 108–115, Sept. 2012. ISSN: 2325-4289.

[16] A. Meyer-Baese and V. Schmid, "Chapter 5 - Genetic Algorithms," in *Pattern Recognition and Signal Analysis in Medical Imaging (Second Edition)* (A. Meyer-Baese and V. Schmid, eds.), pp. 135–149, Oxford: Academic Press, Jan. 2014.

[17] A. Smith, "Tint fill," *ACM SIGGRAPH Computer Graphics*, vol. 13, pp. 276–283, Aug. 1979.

[18] T. W. Gamelin and R. E. Greene, *Introduction to topology*. Courier Corporation, 1999. ISBN: 978-0486406800.

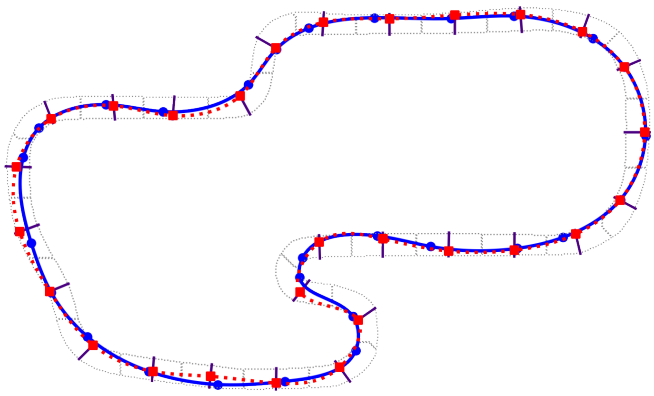
[19] N. R. Kapania, J. Subosits, and J. Christian Gerdes, "A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, p. 091005, Sept. 2016.

[20] R. Rajamani, *Vehicle Dynamics and Control*. Mechanical Engineering Series, Springer US, 2 ed., 2012.

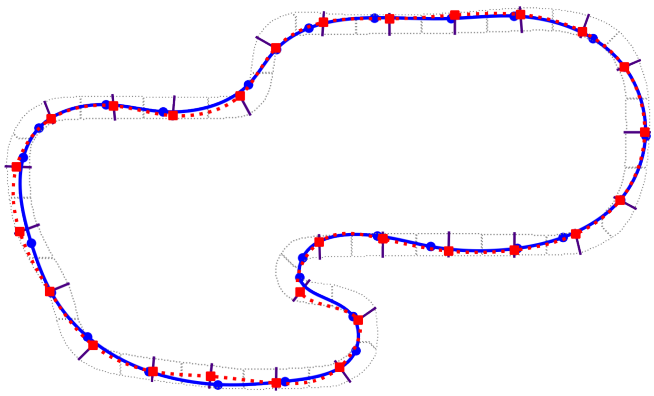
[21] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform." https://GitHub.com/FacebookResearch/Nevergrad, 2018. Accessed on: March 5, 2021.

[22] M. Schumer and K. Steiglitz, "Adaptive step size random search," *IEEE Transactions on Automatic Control*, vol. 13, no. 3, pp. 270–276, 1968.

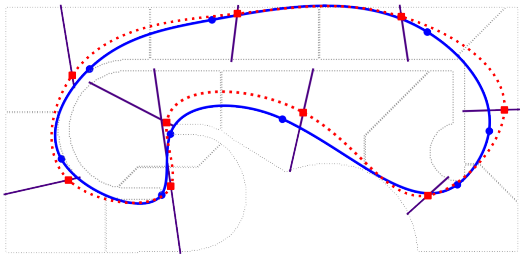
[23] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen, "Fast genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 777–784, ACM, 2017.



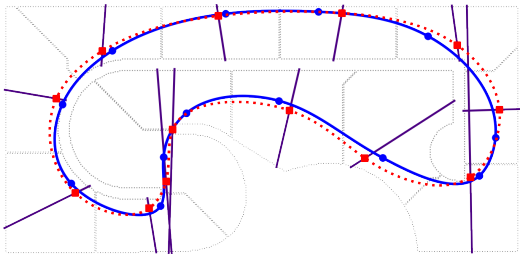
(a) Ruudskogen track, 21 segments. Lap times: 26.94 s vs. 27.70 s.



(b) Ruudskogen track, 28 segments. lap times: 26.68 s vs. 27.94 s.



(c) Torino track, 9 segments. Lap times: 15.00 s vs. 16.33 s.



(d) Torino track, 13 segments. lap times: 14.96 s vs. 14.98 s.

Fig. 9: Comparison of solutions of both approaches with the lowest lap time for a various number of segments on both tracks. The best racing line of 2D-MM is plot in blue color, best racing line of 1D-BR is depicted with red color. Note that we have placed the centers of our 2D segments (their borders are shown using gray color) at the centers of 1D segments for 1D-BR (magenta lines). Also, we tried to avoid intersecting of 1D-BR segments on track (d) by cropping them to the corresponding 2D-MM segments, but this led to longer lap times.