Digital contour approximation with NURBS and Differential Evolution

Diego Buchinger
Graduate Program in Applied Computing
Santa Catarina State University, Joinville, Brazil
Email: diego.buchinger@outlook.com

***Abstract*—Representing a digital object contour in a compressed and precise form is useful for spare computational resources and it is also an important step for many tasks of object recognition and image analysis. Compute a reliable simplification, minimizing error and size, however, is not an easy task. This paper present a new method for represent digital object contours using Non-Uniform Rational B-Splines (NURBS), computed with a Differential Evolution (DE) meta-heuristic. The proposed method was tested regarding its efficiency in approximating object contour in ten different benchmarks, using the canonical DE and the Self-adaptive DE (SaDE). The results shows that the SaDE perform significantly better than the canonical DE in almost every benchmark, and its results are close from those obtained in other research works. Moreover, the proposed method perform better for objects with a curvilinear shape, and worst for contours with many edges.**

I. Introduction

The representation of digital curves contour has been subject of extensive works in image analysis, pattern recognition and computer vision. Accurate and compact representation of digital curves are sought as a way of compress information, reducing the memory storage and processing time for subsequent procedures [1]. Moreover, image analysis tasks are facilitated, such as shape matching and video tracking and object recognition, to cite a few, since objects contour can compose important information about objects [2], [3].

Digital contours can be approximated by a polygonal based approach (e.g. [2], [3]) in which the contour is divided into a number of connected straight line segments, or by a curve based approach (e.g. [4]) in which implicit or parametric curves are used. Both approaches are based on keeping just a minimum set of representative points or dominant points (DP), while trying to preserve the main characteristics of a given contour [3]. Also, it is important to highlight that the order of the points are known.

Finding the optimal polygonal or curve approximation, however, is a challenging 2-objective combinatorial problem, specially when the contour is composed by many points. The objectives are: minimize the approximation error of the polygon or curve (min-ε) and the number of points used in their representation (min-#). Despite being a 2-objective problem, research works usually tackle this problem as a one-objective problem using a given fixed value for the number of dominant points to be used or the maximum error tolerance [2].

When using a polygonal based approach the dominant points are selected from the contour itself, constituting a discrete optimization problem, in which the optimum subset of dominant points should be found [3], [2]. The dominant points however, could also be represented in a continuum space, and thus, are not restricted to be exactly over an original contour point. The continuous version could be more effective for finding solutions of similar approximation error, but using less dominant points than a discrete solution. On the other hand, the discretization could simplify the problem, allowing the computation of the optimum solution (for small instances of the problem) and comparison with heuristic results.

When using curves for contour approximation, most works use a continuous space and parametric curves [4], where the dominant points become synonym for the curve control points, which are the prime element in this type of curves. There is also plenty of parametric curve models to use: Hermite, Bézier, B-spline and Non-Uniform Rational B-Splines (NURBS), to cite some. The last one is the most powerful representation between the given examples, since it could be used for draw a wider range of geometries. Yet, it demands the choice of extra parameter values (knot vector values and control points weight) [5].

Regardless the chosen approach, the problem of approximate a digital contour using a continuous representation, presents a vast search space that can not be searched in feasible time by an optimal algorithm. Hence, most works fall back on heuristics or meta-heuristics, trying to find the best or at least good solutions in an acceptable time. There are plenty of heuristics and meta-heuristics presented in the literature, but among them, one is particularly simple and usually efficient for continuous problems: the differential evolution.

In this paper a new method for contour approximation by curves using NURBS and differential evolution is presented, tested and compared with another methods. This paper was structured as follows: the second section presents some related works; the third section presents contour approximation measures of quality and the benchmark used; the fourth section presents the proposed algorithm; the fifth section presents the obtained results and comparison with other methods; and the sixth section present the conclusions.

II. Related Work

There are many research work published about this topic in the literature, mostly dealing with the discrete version. There exist algorithms for finding the optimal approximation polygon with a time complexity of $O(n^4)$ using dynamic programming, but they can be impractical in real world scenarios, since the number of points is usually large in real image analysis applications [1]. Therefore, local heuristics and meta-heuristics search based methods have been mainly proposed in the literature [2].

The techniques proposed in the literature could be classified into three main categories [6]: sequential approach, in which neighbor points are analyzed in sequential order; split-and-merge approach, in which segments are iteratively split or merged in order to improve the approximation error; and the dominant point-detection approach, in which important curvature regions are identified somehow and the points of those regions are analyzed as dominant points candidates.

To name a heuristic approach, Liu, Zhang and Rockwood [3] have proposed the Direction Change-based Polygonal Approximation (DCPA), which choose dominant points based on the direction change in neighboring points . Their approach do not ensure the optimal solution, but is fast to compute, presenting a linear time complexity of O(N), where N stands for the number of contour points.

Local optimal methods are usually fast to compute, but their results may vary according to initial choices and their results are often far from the optimal solution. Global search heuristics, in the other hand, are commonly slower but they can find solutions close to the global optimal in a yet acceptable time [6]. Thus, nature-inspired algorithms, including primarily genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO), also began to be used for solving discrete polygonal approximation problems [2].

Yin [6] has presented an adaptation of the ant colony optimization algorithm (ACS/Poly), adding the constraints of the problem and problem-specific measures. He has used an offline parameter tuning method for finding good parameter values. Similarly, Yin [1] has also presented an approach (HDPSO2) using an adaptation of the particle swarm optimization for a discrete domain (discrete binary PSO). In order to improve local search, Yin embedded a local optimizer in the algorithm, combination which surpassed the original version. Another adaptation of PSO was presented by Wang et al. [2] who have tried to tackle the problem with an integer particle swarm optimization (iPSO), which deals with the discrete constraints issues and also provides an extra local search optimizer. These methods have shown promising results.

There are many works in the literature about the curve-fitting problem, but most of them requires a parameterization of the data points on the target curve, process in which each contour point is associated with a given sample point of the curve. One problem is that the data parameterization process is unnatural for measuring geometric differences, and an optimal parameterization may be illusive and also difficult to compute. Therefore automatic placement of control points while satisfying a given error tolerance is an alternative approach[7].

Yang, Wang and Sun [7] make use of the squared distance minimization (SDM), that does not require data parameterization, for approximating contours compose of ordered data points. In their method, besides using the SDM to get an approximated initial curve, they add control points until the target curve is within a defined error tolerance, and then they remove redundant control points without making the error larger than the error tolerance. Besides being easier to approximate contours of ordered data points, some works have tackle the variant in which data points are unorganized, non-uniformly distributed and yet may contain noise. Xiuayng et al. [4], for example, propose a method of estimation of distribution algorithm (a novel stochastic optimization algorithm) using a Gaussian mixture model of distribution and an improved k-means algorithm for automatic knot adjustment.

III. Contour Approximation: Representation, Measures and Benchmark

In this section, the contour approximation problem is defined, then some measures of result quality are presented and the benchmark used in this work.

A. Problem Formulation

Given a digital object contour C with N points, represented by a ordered sequence of pixel points z_i, i.e. $C = \{z_i(x_i, y_i) \mid i = 1, 2, ..., N\}$, where z_i is the ith point with coordinates (x_i, y_i), find a new representation A with M points that best approximate its contour. The error in approximating the original contour is given by

$$e(C, A) = \sum_{i=0}^{N} d(z_i, A),$$

in which two objectives are sought: minimize the value of M and the value of $e(C, A)$. It is common, however, to set a fixed value for M or a maximum acceptable value for $e(C, A)$ and pursue just one objective [2], [8].

B. Measures of Evaluation

The new representation is often evaluated regarding the two objectives: how much the contour C was reduced and how precise the new contour A is with respect to the original contour C. The first one is measured with the Compression Ratio (CR) expression, whereas the second is evaluated with the Integral Square Error (ISE) defined as:

$$CR = \frac{N}{M} \qquad ISE(C, A) = e^2(C, A) \tag{1}$$

in which smaller values are better. Since the nearest the A arcs are from C the best the approximation is, the ISE measure (that uses the quadratic error) is used with the intent to benefit close arcs from C and to penalize distant arcs from C. From the combination of CR and ISE a third measure – Figure of Merit (FOM) – is defined as:

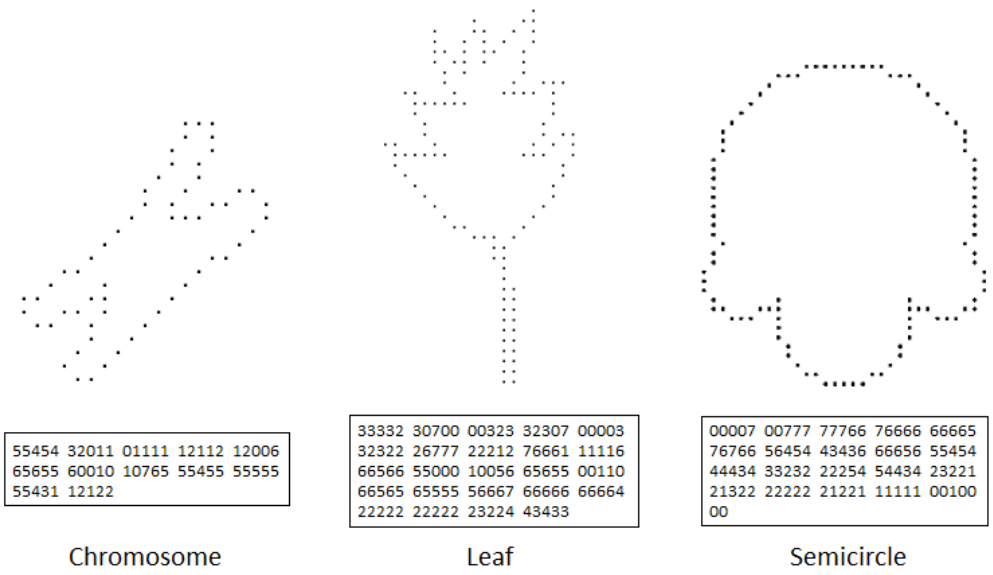


Fig. 1. Three popular synthesized shapes and their chain codes (adapted from [2], [9])

$$FOM = \frac{CR}{ISE} \tag{2}$$

in an attempt to put together the two objectives in a single measure. For this one, higher values are better.

There is also two other measures that are commonly used for the discrete case of the contour approximation problem: the fidelity and efficiency of an approximation. They require the optimum result to be computed, what is feasible for such cases. Their definition is:

$$Fidelity = \frac{E_{opt}}{E_{approx}} \qquad Efficiency = \frac{M_{opt}}{M_{approx}} \tag{3}$$

where E_{opt} and E_{approx} represent the error of the optimum and approximation algorithms, respectively, with the same number of dominant points. Similarly, M_{approx} represent the number of dominant points produced by the approximation algorithm, and M_{opt} is the minimum number of dominant points required by the optimum algorithm to produce the same error [3]. Again, a third composed measure – Merit – is derived for ranking the approximation algorithms:

$$Merit = \sqrt{Fidelity \times Efficiency} \tag{4}$$

As the continuous case of the contour approximation problem is being tackled in this research, and no efficient algorithm was found to compute the optimum solution, the fidelity, efficiency and merit measures were not used.

C. Benchmarks

When the order of points of a given contour is known, it is commonly represented by a Freeman chain code [10]. There are three popular synthesized benchmark for contour approximation of digital curves widely used. They are the chromosome, leaf, and semicircle [1], represented by 60, 120 and 102 points (N) respectively. Figure 1 shows these three shapes and their chain code.

Another benchmarks are presented and used by [2]. Their are composed by four lake shapes from famous lakes: Arlington (in USA), Como (in Italy), Managua (in Nicaragua) and Simcoe (in Canada), represented by 133, 124, 120 and 134 points (N) respectively. Figure 2 shows these four shapes and their chain code.

As these benchmarks are represented by a relatively small number of points (N), Yin [6], [1] also suggests the use of bigger shapes in order to demonstrate scalability for real-world applications. He has used a fish contour with 700 points and a plane contour with 682 points, however, their chain-codes were not given and therefore those shapes could not be replicated. Thus, in order to attain a similar objective, three new benchmarks were created inspired in three famous racing tracks: Laguna Seca (in California, USA), Nürburgring (in Nürburg, Germany) and Silverstone (in Northamptonshire, England), composed by 1.463, 1.887 and 1955 points respectively). Their shapes are presented in Figure 3 and their chain codes is presented in the supplementary materials.

IV. A Differential Evolution Approach using NURBS

Differential Evolution is one ramification of the Evolutionary Algorithms, therefore it is a stochastic, nature inspired population-based algorithm, specially powerful for continuous problems, i.e. use of real-valued parameters. DE uses a fixed population size (NP) – but it is not mandatory – where each individual represents a solution and is constituted by an array of D real values (number of problem dimensions). DE uses a

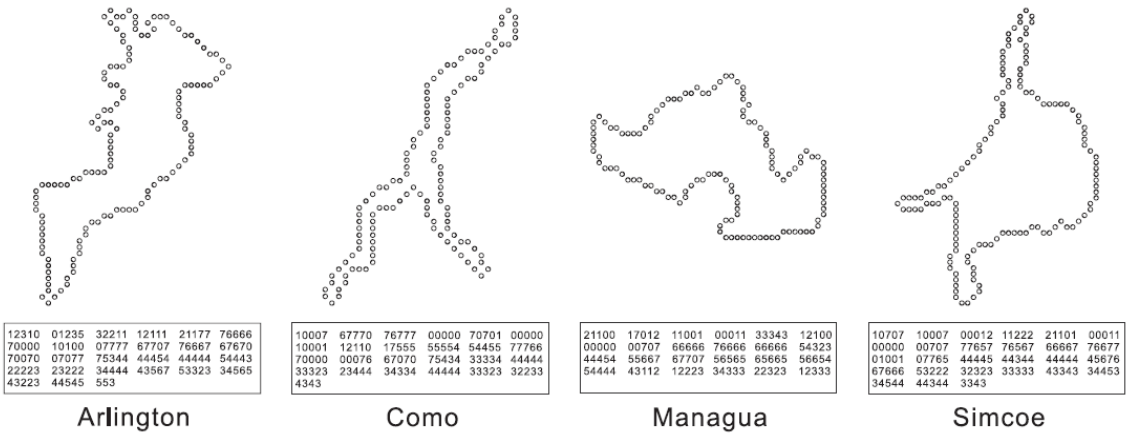


Fig. 2. Three popular synthesized shapes and their chain codes (adapted from [2], [9])

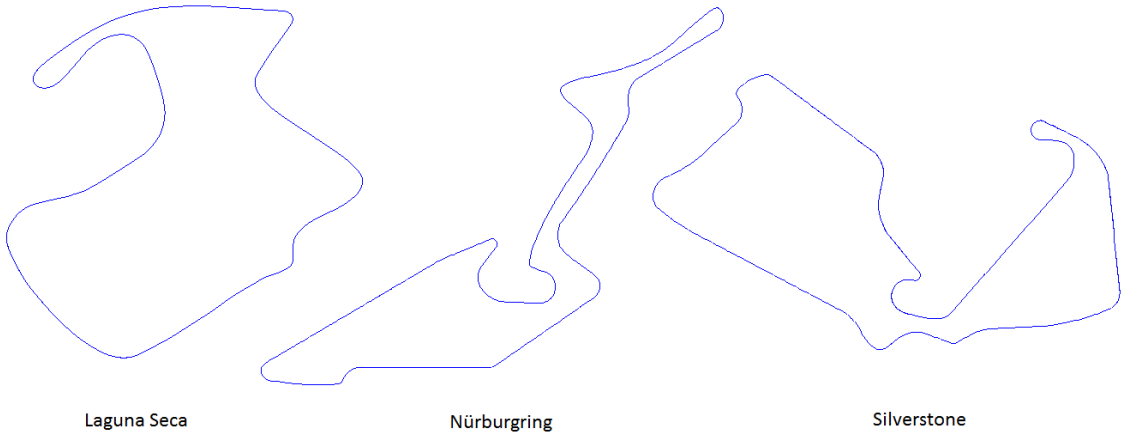


Fig. 3. Three famous racing tracks contour (their chain codes are presented in the Supplementary Materials

simple but effective mutation operator that uses the differences of randomly sampled pair of computed solutions [11].

DE is a robust meta-heurstic that is fast and efficient, using only linear complexity processes. Moreover, it is also inherently parallel, simple, easy to use and tune. Still, if problem-specific knowledge is not used, DE performance will likely be sub-par [11]. Given theses characteristics of DE, a new approach using DE for approximating objects contour with a single NURBS in a continuous space is presented. The proposed approach tries to find optimum location and weight for each control point of the NURBS.

The solutions are represented as a vector of DPs trios (x_i, y_i, w_i), i.e. the position of each dominant point in the x and y axis and its weight. As the most used fitness function is the ISE measure, we have also selected this function. As our approach we are using curves, however, the ISE value is also approximated by the smallest distance from a original contour point to a generated curve sample, as illustrated in Figure 4. The pentagons represent contour points of the original shape, the circles represent dominant points (control points), the green squares represents samples computed from the NURBS, and the red lines represent possible shorter distances.

Two important aspects about the ISE approximation should be highlighted. First, a naive computation of the shortest distance could be costly for many points or sample points, and the shorter distances found could be overestimated since there are limited samples from the curve. In the other hand, computing the shortest distance could underestimate error, since entire curve segments could be ignored for being away, like the tie near the dominant points x_{i+5} and x_{i+5} in Figure 4 that is totally inconsiderate in the ISE approximation, but should not be.

For our proposed method we have set two important properties: the number of control points (dominant points) M is defined before the algorithm's execution and it is not changed during execution; the curve degree is defined as 3, since [12] have pointed out that degree 3 to 4 are robust and are quickly computed, whereas high degree polynomials are likely to oscillate.

Using some problem-specific knowledge, we have also defined a new way of initialize the population. Since NURBS control points are usually close to the curve they represent,

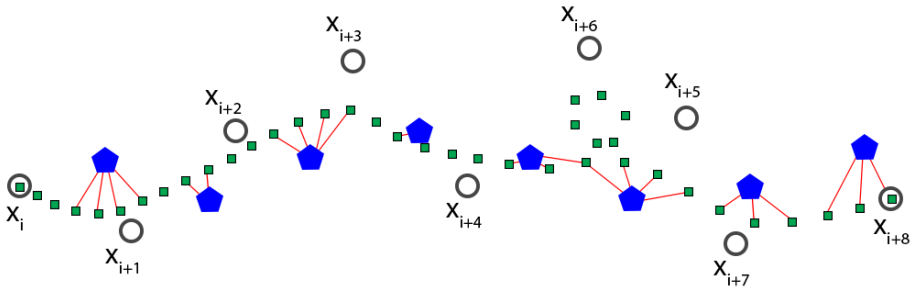


Fig. 4. Approximation used for the Integral Squared Error when using curves.

it is natural to initialize them near the target contour to approximate. In order to do that, we have established a lower and upper bound for the x and y axis, as well as for the weight value, and generate a random position near the original shape, as follows:

$$x_j = x_j^{lb} + rand_j[0,1] \times (x_j^{ub} - x_j^{lb}) \quad (5)$$

where $rand_j[0,1]$ denotes an uniformly distributed random value ranging from zero to one, and lb and ub denote a lower bound and an upper bound for the given parameter, respectively. The positional upper bound and lower bound are +5.0 and -5.0 respectively, and the weight upper bound and lower bound are 1.5 and 0.5 respectively.

Furthermore, as we are not able to anticipate which contour regions will require more or less control points for an optimal approximation (usually lines require less DPs and curves require more DPs), the alternative chosen was to split the contour in M sections, and randomly pick a point of each section from the original contour and then use equation 5 for compute the control point position and weight.

DE uses yet three more prime parameters: the crossover probability (CR), a scaling or differential factor (F) and the population size (NP). For these values, we choose to follow the values recommended in [11] with minor adaptation, using $CR = 0.8$, $F = 0.5$ and $NP = min(20D, 100)$.

Besides using a canonical DE (DE) we have also implemented and tested a Self-adaptive DE, as proposed by [13], in which CR, F and the mutation strategy are self-adaptive. We use the same initial parameters values, and for the mutation strategies we have used DE/rand/1/bin and DE/current-to-best/2/bin, same as [13]. According to [13], these two strategies have been commonly used in many DE presented in the literature, reporting to perform well on problems with distinct characteristics. The strategy "rand/1/bin" usually demonstrates good diversity whist the "current-to-best/2/bin" strategy shows good convergence.

V. Experimental Results and Discussion

In order to test the proposed method and compare with some literature values, we have executed the algorithm 30 times for each instance (benchmark + M + canonical/self-adaptive). The platform used in the experiments was a PC with an Intel Core^{TM} i7-4770 @ 3.40GHz (with 8 logical cores), equipped with 16 GB of RAM. The algorithms were coded in C# and, seeking for speedup gain, specially for the the fitness computation, both the canonical DE and the SaDE were implemented using multithreading.

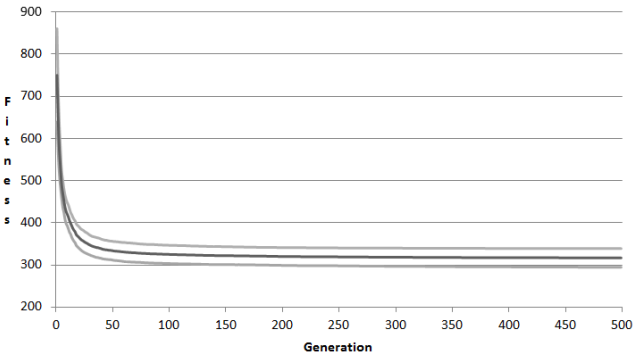


Fig. 5. The fitness convergence over generations in Arlington benchmark, with $DPs = 8$ and canonical DE. The stronger curve stands for the mean fitness and the thinner curves stands for the maximum and minimum solution fitness.

First, in order to obtain an estimation of how many generations are required for the algorithm to converge, we have executed the canonical DE and the SaDE 30 times with 500 generations each, for a random instance (arlington benchmark, with $M = 8$ and canonical DE), monitoring the population diversity with the [14] proposed measure. The Fitness convergence over the generations is presented in Figure 5 and the diversity change over generations is presented in Figure 6.

From Figures 5 and 6 one can see that the convergence is attained between the 50^{th} and 100^{th} generation. After that, just minor improvement is obtained. For this reason, we decided that we would execute the canonical DE and SaDE with 200 generations for each benchmark, giving enough time for the convergence and exploitation.

A. Classical Benchmarks

Firstly, the proposed method was evaluated with the classical benchmarks. When analyzing the results of similar research works we found different numbers of M used. We selected some of them to use in our tests and compare the literature

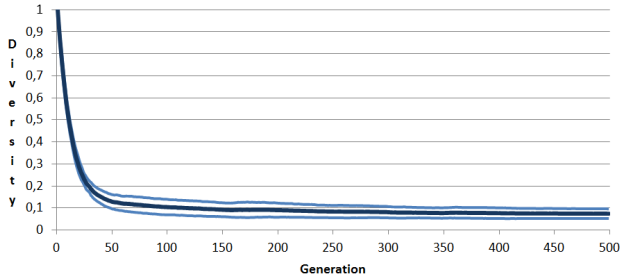


Fig. 6. The diversity change over generations in Arlington benchmark, with $DPs = 8$ and canonical DE. The stronger curve stands for the mean diversity and the thinner curves stands for the maximum and minimum diversity.

TABLE I
COMPARISON OF THE PROPOSED ALGORITHM AND OTHER METHODS ON THE CLASSICAL BENCHMARK

Benchmark	Alg.	M	ISE (σ)	MaxE (σ)	FOM
Chromosome (60 points)	DCPA[3]	12	4.93	0.417	1.013
	DE	12	6.2 (2.0)	0.6 (0.4)	0.805
	SaDE	12	5.0 (0.8)	0.4 (0.1)	1.009
Leaf (120 points)	iPSO[2]	9	136.6	-	0.098
	DE	9	219.1 (19.6)	12.5 (1.6)	0.061
	SaDE	9	190.1 (34.2)	11.1 (1.9)	0.070
	iPSO[2]	16	26.8	-	0.280
	DE	16	94.6 (17.3)	8.6 (1.5)	0.079
	SaDE	16	57.9 (17.0)	5.2 (2.4)	0.130
	DCPA[3]	25	10.20	0.626	0.471
	DE	25	28.1 (6.0)	2.6 (1.2)	0.171
	SaDE	25	20.8 (3.5)	1.6 (0.5)	0.231
Semicircle (102 points)	iPSO[2]	7	93.05	-	0.157
	DE	7	156.5 (8.8)	13.0 (1.0)	0.093
	SaDE	7	146.8 (18.2)	12.9 (0.8)	0.099
	iPSO[2]	12	26.00	-	0.327
	DE	12	31.3 (8.7)	3.8 (1.7)	0.271
	SaDE	12	22.9 (5.1)	2.5 (1.1)	0.372
	DCPA[3]	24	5.71	0.329	0.744
	DE	24	7.5 (1.2)	0.7 (0.3)	0.565
	SaDE	24	7.0 (1.0)	0.7 (0.3)	0.610

results with ours. To do this, we set up Table I listing the values of ISE, Maximal Error (MaxE) and FOM. The results presented for the iPSO algorithm are the best found, since no mean or standard deviation was presented, whereas the results from our canonical DE and SaDE represent the mean (and standard deviation of 30 executions).

The results in Table I shows that the proposed method has an efficacy very similar to the DCPA and iPSO in some cases (chromosome and semicircle with M=12). In the other hand, the results for the leaf benchmark were far worst from DCPA and iPSO. These results suggests that the proposed method is better when approximating curvilinear contours, what would be expected since NURBS are a parametric curve representation.

One last important thing that should be highlighted is the better performance of the SaDE regarding the canonical DE. In order to better evaluate their differences several comparative box plots are presented. Figure 7 shows the result for the chromosome benchmark, Figure 8 shows the result for the leaf benchmark and Figure 7 shows the result for the semicircle benchmark. From the box plots presented, one can see that SaDE got better results than the canonical DE. In order to test the statistical significance of this difference, the Wilcoxon signed-rank test was applied for every benchmark. The Wilcoxon test shows that in all cases the SaDE was statistically better than DE with confidence level of 95%, except in the semicircle with M=24.

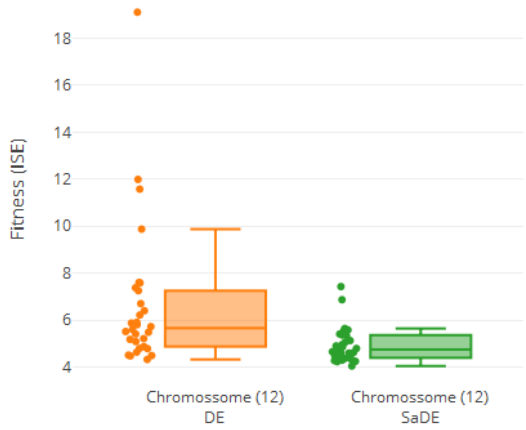


Fig. 7. Results when using the chromosome benchmark (M=12 → Wilcoxon p-value=0.0054)

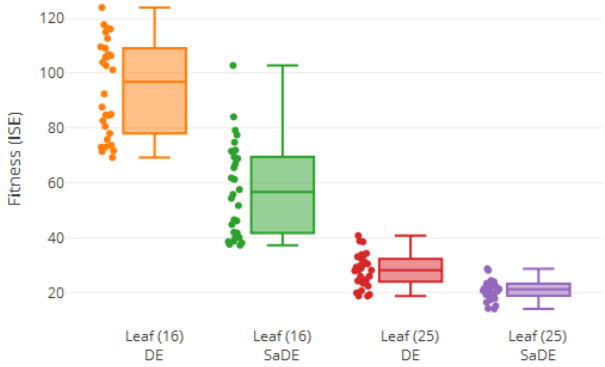


Fig. 8. Results when using the leaf benchmark. (M=16 → Wilcoxon p-value=9.313e-09 / M=25 → Wilcoxon p-value=1.684e-06

Some visual results of the classical benchmarks are presented in the Supplemental Material, in Figure 14, Figure 15 and Figure 16.

B. Lake Benchmarks

The proposed method was also evaluated with the lake benchmarks presented by [2]. Again, we have selected some numbers of M used by [2] in their experiments in order to compare the efficacy of the algorithm. The obtained results are presented in Table II listing the values of ISE, Maximal Error

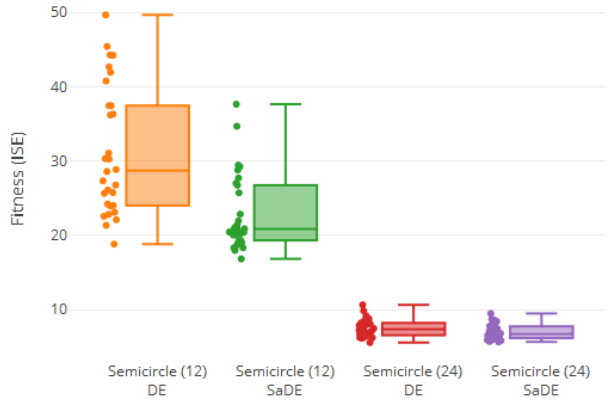


Fig. 9. Results when using the semicircle benchmark (M=12 → Wilcoxon p-value=3.79e-06 / M=24 → Wilcoxon p-value=0.08794

(MaxE) and FOM. Here again, the results presented for the iPSO algorithm are the best found, since no mean or standard deviation was presented, whereas the results of our algorithms represent the mean (and standard deviation of 30 executions).

For this benchmark one can see the results of the proposed method are worst than the iPSO. As the iPSO has only its better results a statistical analysis comparing if this algorithm is significantly better than ours can not be done. Nevertheless, the results of SaDE are again better than the canonical DE. In order to evaluate their differences, several comparative box plots are presented, where the instances of lowest M were omitted. Figure 10 shows the result for the arlington benchmark, Figure 11 shows the result for the como benchmark, Figure 12 shows the result for the managua benchmark and Figure 13 shows the result for the simcoe benchmark. We have used again the Wilcoxon signed-rank, which pointed out that in all cases the SaDE was statistically better than DE with confidence level of 95%.

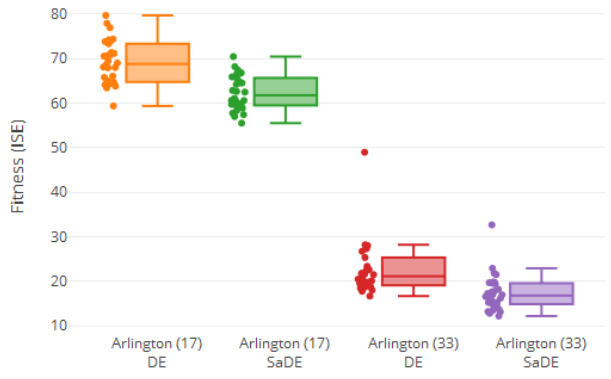


Fig. 10. Results when using the arlington benchmark (M=17 → Wilcoxon p-value=9.22e-06 / M=33 → Wilcoxon p-value=3.904e-05

TABLE II
COMPARISON OF THE PROPOSED ALGORITHM AND OTHER METHODS ON THE LAKE CONTOUR BENCHMARK [2]

Benchmark	Algorithm	M	ISE (σ)	MaxE (σ)	FOM
Arlington (133 points)	iPSO[2]	8	145.70	-	0.114
	DE	8	319.5 (21.0)	14.3 (1.0)	0.052
	SaDE	8	288.5 (22.4)	14.1 (1.1)	0.058
	iPSO[2]	17	29.45	-	0.266
	DE	17	69.2 (4.9)	5.9 (0.9)	0.113
	SaDE	17	62.3 (3.8)	5.8 (0.6)	0.126
	iPSO[2]	33	9.88	-	0.408
	DE	33	22.7 (6.0)	1.8 (1.0)	0.177
	SaDE	33	17.3 (4.0)	1.3 (0.6)	0.233
Como (124 points)	iPSO[2]	6	121.13	-	0.171
	DE	6	272.4 (13.8)	11.4 (1.3)	0.076
	SaDE	6	250.6 (15.8)	10.5 (1.1)	0.082
	iPSO[2]	11	55.11	-	0.205
	DE	11	101.0 (22.7)	5.8 (2.0)	0.112
	NDE	11	71.1 (14.0)	3.9 (1.6)	0.159
	iPSO[2]	24	9.78	-	0.528
	DE	24	21.8 (2.8)	1.3 (0.4)	0.237
	NDE	24	19.3 (2.7)	1.1 (0.3)	0.268
Managua (120 points)	iPSO[2]	8	148.65	-	0.101
	DE	8	322.6 (53.0)	18.8 (7.9)	0.046
	NDE	8	245.0 (35.0)	12.4 (2.9)	0.061
	iPSO[2]	14	29.00	-	0.296
	DE	14	58.0 (5.4)	4.2 (0.9)	0.148
	NDE	14	47.7 (4.8)	3.4 (0.7)	0.180
	iPSO[2]	25	9.23	-	0.520
	DE	25	17.1 (2.2)	1.6 (0.5)	0.292
	NDE	25	13.5 (2.5)	1.1 (0.5)	0.370
Simcoe (134 points)	iPSO[2]	9	108.46	-	0.137
	DE	9	229.6 (51.0)	11.8 (3.3)	0.065
	SaDE	9	159.8 (32.6)	7.7 (2.4)	0.093
	iPSO[2]	16	28.24	-	0.297
	DE	16	38.0 (5.6)	2.0 (0.5)	0.221
	SaDE	16	32.4 (4.2)	1.7 (0.4)	0.259
	iPSO[2]	31	9.72	-	0.445
	DE	31	19.2 (1.3)	1.3 (0.2)	0.225
	SaDE	31	17.3 (1.3)	1.3 (0.2)	0.250

Some visual results of the lake benchmarks are presented in the Supplemental Material, in Figure 17, Figure 18, Figure 19 and Figure 20.

C. Racing Tracks Benchmark

Finally we have tested our method in the proposed racing tracks benchmarks, which are composed by much more points (N) than other benchmarks. For these benchmarks we set two different values for M. The first M value used is the square root of N which we have believed to be sufficient to create at least a primitive contour, and the second M value is given by the expression: $M = N^{0.65}$ which we have believed to be a fair value for creating a good approximation and provide a good compression rate.

As the naive approach used for approximating the ISE value for parametric curves is $O(n * m)$, where n is the number of contour points and m is the value of NURBS curve samples, our algorithm do not scale well for these benchmarks, taking between 40 to 50 minutes to compute 200 generations. Because of this, we have executed our algorithm just 5 times for each of these benchmarks. The obtained results are presented in Table III listing the values of ISE, Maximal

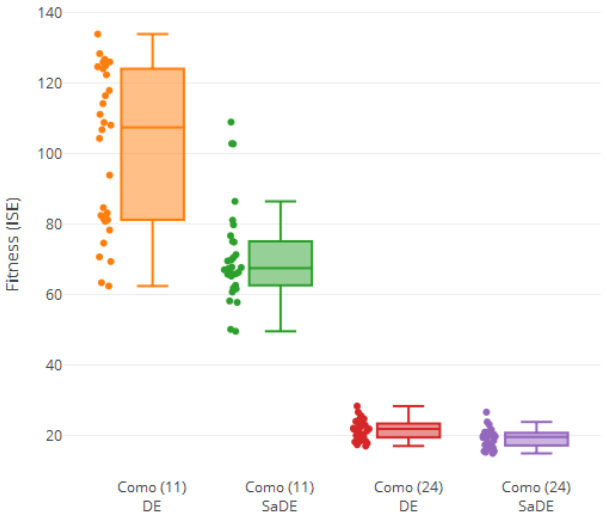


Fig. 11. Results when using the como benchmark (M=11 → Wilcoxon p-value=6.918e-06 / M=24 → Wilcoxon p-value=0.001583

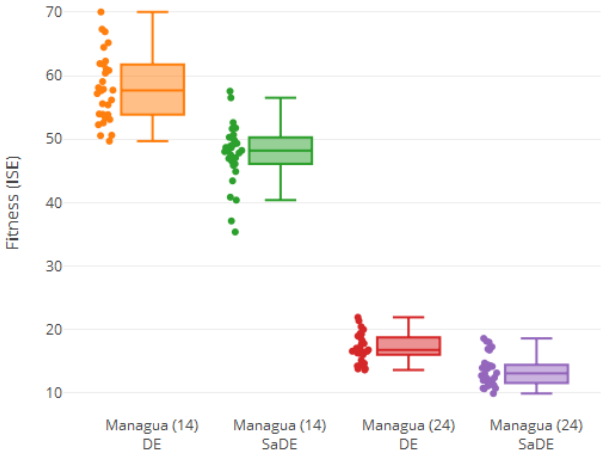


Fig. 12. Results when using the managua benchmark (M=14 → Wilcoxon p-value=1.863e-09 / M=25 → Wilcoxon p-value=5.974e-06

Error (MaxE) and FOM. whereas the results of our algorithms represent the mean (and standard deviation of 30 executions).

Despite the scalability issue, circuits benchmarks may demonstrate the potential of this new method of tackle the problem. The proposed method has presented considerable good visual results that are presented in the supplementary materials, in Figure 21, Figure 22 and Figure 23, and further comparisons are require in order to evaluate the efficacy of the presented results.

VI. Conclusion

Digital contour approximation is an important task for information compression, object recognition and image analysis,

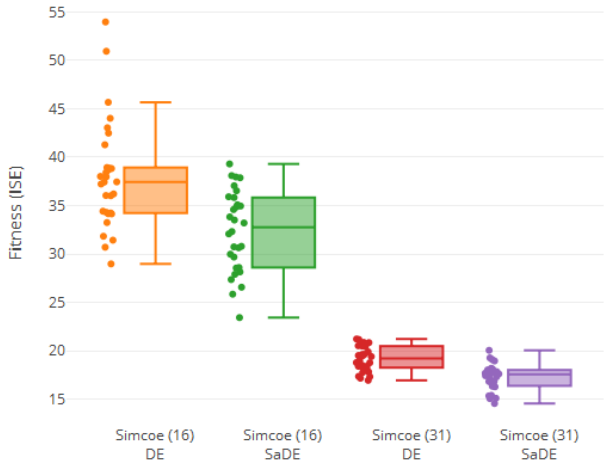


Fig. 13. Results when using the simcoe benchmark (M=16 → Wilcoxon p-value=3.904e-05 / M=31 → Wilcoxon p-value=1.218e-05

TABLE III
Results of the proposed algorithm on the racing tracks benchmark

Benchmark	Alg.	M	ISE (σ))	MaxE (σ))	FOM
Laguna Seca (1463 points)	DE	38	3021.2 (250)	24 (9.3)	0.013
	SaDE	38	1934.7 (146)	9 (2.5)	0.020
	DE	114	370.5 (25)	3 (0.2)	0.027
	SaDE	114	279.2 (20)	2 (0.3)	0.138
Nürburgring (1887 points)	DE	43	7680.8 (1316)	64 (27.5)	0.006
	SaDE	43	4341.8 (320)	27 (3.1)	0.010
	DE	135	826.7 (58)	5 (2.2)	0.017
	SaDE	135	552.5 (42)	4 (1.6)	0.025
Silverstone (1955 points)	DE	44	12060 (1165)	96 (17.6)	0.004
	SaDE	44	5057.6 (593)	50 (38.2)	0.009
	DE	138	773.1 (118)	7 (3.8)	0.019
	SaDE	138	604.7 (50)	4 (1.9)	0.024

to cite some. The problem is often tackle through a discrete approach in which the contour points should be exactly over original points and the resulting approximation is a polygon. We have presented in this paper a new continuous approach using the Non-Uniform Rational B-Spline parametric curve for approximating a contour, where the control points may lay in any position. In order to approximate a given contour using NURBS, we have used the differential evolution meta-heuristic.

Through experiments on ten benchmarks the proposed method showed promising results, very close to the ones presented in the literature. The method was robust, specially for contours with smooth curves (e.g. semicircle benchmark), but presented difficulties when dealing with many sharp edges (e.g. leaf benchmark). Thus, this method can be considered as an option for digital contour approximation when dealing with curved shapes. Moreover, the results pointed out that a Self-adaptive Differential Evolution performed significantly better than a canonical Differential Evolution.

Despite using the Integral Square Error (ISE) as other

research works, this measurement is tricky and not entirely adequate when using a continuous space, since it may disregard contour errors that occur far away from the original contour. Anyway, for discrete or continuous space, since the original shape is discretized it seems more adequate to use a discrete measure of approximation as well, like the sum of the differences between pixels of the original contour and approximated contour.

In the presented approach we have used a single NURBS for approximating a contour. Further works could split the original shape and approximate each fragment, individually, with a NURBS, ensuring that the first and last points are the same as the original contour. Also, the proposed method should be compared with other, more similar, methods. In this research work we have compared our continuous approach with discrete approaches, but it should be tested with other continuous approach as well.

REFERENCES

[1] P.-Y. Yin, "A discrete particle swarm algorithm for optimal polygonal approximation of digital curves," *Journal of Visual Communication & Image Representation*, vol. 15, pp. 241–260, 2004.

[2] B. Wang, D. Brown, X. Zhang, H. Li, Y. Gao, and J. Cao, "Polygonal approximation using integer particle swarm optimization," *Information Sciences*, vol. 278, pp. 311–326, 2014.

[3] H. Liu, X. Zhang, and A. Rockwood, "A direction change-based algorithm for polygonal approximation," in *International Conference on Pattern Recognition (ICPR 2012)*, vol. 21, nov. 2012, pp. 3586–3589.

[4] Z. Xiuyang, Z. Caiming, Y. Bo, and L. Pingping, "Adaptive knot placement using a gmm-based continuous optimization algorithm in b-spline curve approximation," *Computer-Aided Design*, vol. 43, pp. 598–604, 2011.

[5] M. E. Mortenson, *Geometric Modeling*, 3rd ed. New York: Industrial Press Inc., 2006.

[6] P.-Y. Yin, "Ant colony search algorithms for optimal polygonal approximation of plane curves," *Pattern Recognition*, vol. 36, pp. 1783–1797, 2003.

[7] H. Yang, W. Wang, and J. Sun, "Control point adjustment for b-spline curve approximation," *Computer-Aided Design*, vol. 36, pp. 639–652, 2004.

[8] T. Pavlidis, "Approximation of curves," in *Algorithms for Graphics and Image Processing*. Berlin, Heidelberg: Springer, 1982, ch. 12, pp. 77–108.

[9] C.-H. Teh and R. T. Chin, "On the detection of dominant points on digital curves," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, pp. 859–872, 1989.

[10] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers*, vol. EC-10, pp. 241–260, 1961.

[11] K. Price, "An introduction to differential evolution," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. Berkshire: McGraw-Hill, 1999, ch. 10, pp. 77–108.

[12] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, pp. 3–15, 2003.

[13] A. Qin and P. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *The 2005 IEEE Congress on Evolutionary Computation (IEEE CEC 2005)*, vol. 3, sep. 2005, pp. 1785–1791.

[14] G. Corriveaua, R. Guilbault, A. Tahana, and R. Sabourinb, "Review of phenotypic diversity formulations for diagnostic tool," *Applied Soft Computing*, vol. 13, pp. 9–26, 2013.

SUPPLEMENTARY MATERIALS

TABLE IV
LAGUNA SECA CHAIN CODE

00000 01001 00101 01010 10101 01010 10101 10101 01010 10110
10110 10110 11011 01011 01101 10101 10110 11010 11011 01101
10111 01101 11011 10110 11011 01101 01101 01101 01011 01010
11011 01010 10100 10010 01001 00101 01010 10110 11212 12222
22222 22222 22222 22122 22121 21121 11111 11110 11101 01101
01010 10101 01010 10100 10101 01101 01011 01110 11110 11111
21121 21221 22222 22232 32332 33333 33333 33334 33334 33433
34334 33433 43343 34334 33434 33343 34334 33434 33433 43343
34334 33433 43343 33433 34333 43333 33333 33332 33233 23232
23222 22222 22121 21212 11211 21121 21121 12112 11211 21112
11121 12111 21121 11211 21121 12212 22323 23433 44344 44444
43444 44444 44444 44434 44444 44444 44344 44444 44444 43444
44444 44444 43444 44444 44444 44443 44444 44444 44444 44444
44444 44444 44544 44444 44444 44445 44444 44444 54444 44544
44544 44544 45444 54445 44544 45445 44544 54454 45454 45454
54545 45454 54545 45454 55454 55454 55455 45455 45554 55455
45554 55455 54555 45554 55555 45555 55505 55656 56666 66767
77707 00070 00010 00100 10110 11011 11111 11111 21111 11121
11111 21111 11211 11011 11011 10110 11010 01010 01000 10000
01000 00070 00007 07007 70770 77777 76776 76767 67676 67676
67676 67667 66766 76676 67667 66676 67666 76667 66676 66667
66666 66766 65666 66665 66665 66566 56656 56556 55655 56555
55555 55545 54554 55455 45545 54545 54554 55454 55455 45455
45545 45545 45545 45454 55454 54544 54454 45444 54544 54445
44445 44445 44454 44454 44544 45444 54445 44544 54545 45545
55455 55655 56556 55656 56656 65666 66666 66667 66606 66767
66767 67676 76767 67676 76067 67677 67677 67767 77677 77677
77767 77776 06777 77776 77777 77777 77777 77777 70777 77770
77770 77707 70770 77077 07070 77070 70707 00707 00070 00700
00006 00000 00

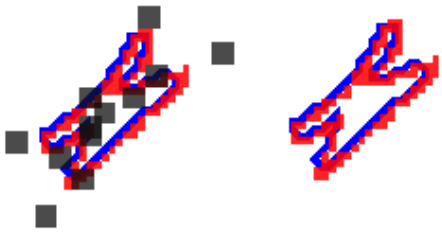


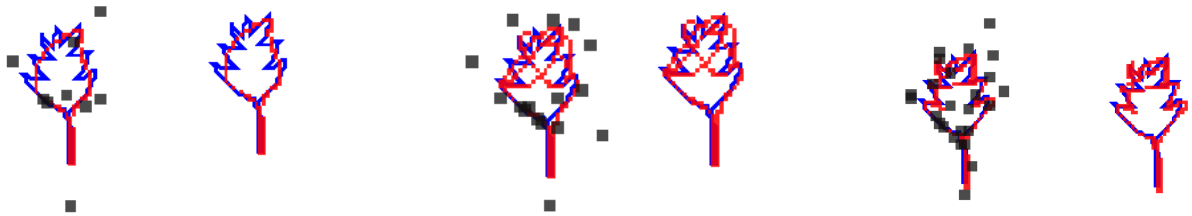
Fig. 14. Chromossome (DPs=12)

TABLE V
NÜRBURGRING CHAIN CODE

00000 01011 21212 12121 11110 11010 10010 00000 00000 00000
00000 00000 00000 00000 00000 00000 00000 00000 00000 00000
00000 00000 00000 00000 00000 00000 00000 00000 00000 00000
00000 00000 00000 00000 00000 00000 00000 00000 00000 00000
00000 00000 00000 10010 11011 10110 11011 01101 11011 01101
10111 01101 10110 11011 01101 11011 01101 10111 01101 10110
11011 01110 11011 01101 10111 01101 10110 11101 10110 11011
01111 11211 22122 22222 22232 23323 33334 33343 34333 43334
33343 33433 43334 33343 33333 23332 32322 32223 22222 22222
22122 21221 21121 12111 21121 12111 21112 11211 21112 12111
21121 12112 11211 12121 12112 11211 21211 21121 12112 12112
11212 11212 11212 11212 11212 11211 21212 11212 12112 12121
22122 21222 22222 22232 22222 22222 22221 22212 21212 12112
11111 01101 10110 10110 10110 10110 10110 10110 10110 10110
11010 11010 11010 11010 11011 01010 11011 01010 11011 01011
01011 01011 01011 11211 21221 22222 22223 22323 33434 44544
55454 55455 54555 55455 55545 55555 54555 55555 55554 55555
55555 45554 55545 54554 54554 54554 54545 45454 45454 54544
54454 54454 45445 44544 54454 44544 54445 44544 44544 45444
54444 54444 54444 54444 54445 44454 44544 54454 45454 54545
54556 56666 76676 77677 77777 07777 77077 77777 07777 77077
67767 76766 76666 76666 65666 66656 65666 56656 56565 65655
65565 55655 65565 56556 55655 65565 56556 55655 65565 65565
56565 56565 56565 56565 65565 65656 56556 56566 55665 65656
56566 56565 66565 66565 66566 56665 66566 66566 66666 67770
70070 70070 07000 70070 77077 77676 76766 76666 66665 66656
56555 64555 45454 45444 44444 44444 44444 44444 43444 44444
44444 44444 44444 44434 44434 44343 44343 34343 33433 33323
33232 23223 22222 22222 21221 21212 11121 11121 11211 12111
12111 21212 23233 33444 45454 45454 45454 54454 54454 54454
54454 54544 54544 54545 44544 54545 44545 45454 54545 54545
45545 45545 45545 45455 45455 45455 45455 45454 55454 54554
54554 54554 54554 54545 54545 45545 54545 45545 45545 45545
45455 45455 45455 45454 55454 55454 54554 54554 54554 54554
54545 54545 54545 45545 54545 45545 45455 45455 45455 45454
55454 55454 54554 54554 54555 55656 66666 66676 77777 07000
70000 07000 00070 00000 70000 00007 00000 00000 00070 00000
00000 00000 00000 00000 00000 00000 00000 0

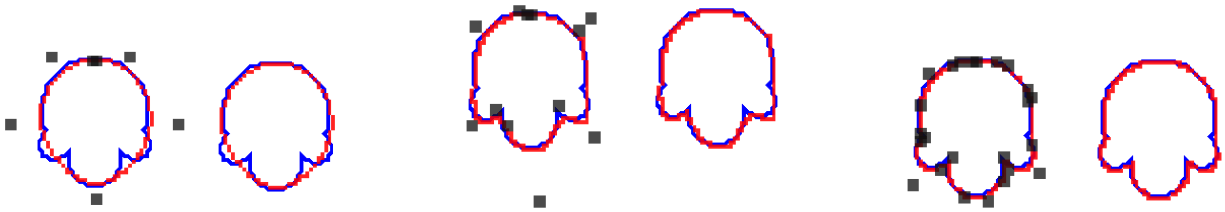
TABLE VI
SILVERSTONE CHAIN CODE

00010 01000 10101 01010 11011 01110 11112 12122 22323 23323
23232 22322 22222 12221 22121 12111 11110 11111 11111 10111
11111 11101 11111 11110 11111 11101 11110 11101 11011 01011
01011 01010 10101 01010 01010 10110 11111 12112 12121 22122
22222 22322 32323 23334 33333 43333 43334 33433 34334 33434
33433 43343 43433 43434 34334 34343 43434 34343 43434 33434
34343 43433 43434 34343 43434 34343 43433 43434 34343 43433
43434 34343 43434 34343 43433 43434 34343 43434 33434 34343
43434 34334 34343 43434 34343 43433 43434 34343 43434 34334
34343 43434 34343 43343 33433 34234 32333 33233 23323 23232
32323 32333 32333 33433 43434 44444 44454 45545 55450 54555
54554 55454 54545 44544 45444 44444 44444 44344 43443 44344
34344 34434 43434 43443 44344 34444 44454 55454 55454 54545
45454 54545 44544 54454 44454 44444 44444 44454 44444 44444
44444 44444 45444 44444 44444 44444 44445 44444 44444 44544
44444 45444 44454 44445 44444 54444 54444 54445 44445 44454
45444 54454 45444 54544 54454 54454 54454 54545 45554 50555
50556 65666 65056 66660 66666 66666 06666 66666 66066 66666
66606 66666 66666 06666 66666 05066 66666 66606 66666 66660
50666 66666 66066 66666 66605 06666 66666 60666 66666 66066
66666 66605 06666 66666 05066 66666 66606 66666 76667 66606
67660 66767 67606 76776 77760 67777 77777 77060 77060 77077
07070 70707 07070 70707 07070 70700 60060 06006 00707 07070
70700 06001 00011 01112 12122 22223 22333 33344 34444 44444
44444 44444 34444 44434 44343 43433 33333 23232 32222 32222
22222 22122 22122 12212 12112 11211 11121 11111 11211 11111
21111 11112 11111 11121 11111 11211 11111 21111 11121 11111
11211 11111 12111 11111 12111 11112 11111 12111 11111 21111
11112 11111 11211 11111 12111 11111 21111 11121 11111 11211
11111 11110 10100 10000 00000 00000 00000 07000 00700 00060
00070 00600 07000 60006 00700 60060 77060 60677 67660 66660
66666 65056 66565 65655 55545 45454 45444 44444 43443 44545
55656 66767 77767 77760 60677 77677 77606 77776 06777 60606
77776 06777 60677 76067 76760 66067 67660 66766 76676 66766
66660 66666 66666 66666 50566 66656 66650 56666 56666 56666
50566 66666 66666 06666 76606 67676 06776 06777 70770 60777
07707 77060 77060 77077 70777 07770 77707 70777 06077 07770
77707 70607 70607 77077 06077 06077 06077 07770 77060 77060
77060 77060 77060 77077 70777 07770 77707 70607 70607 07000
0000



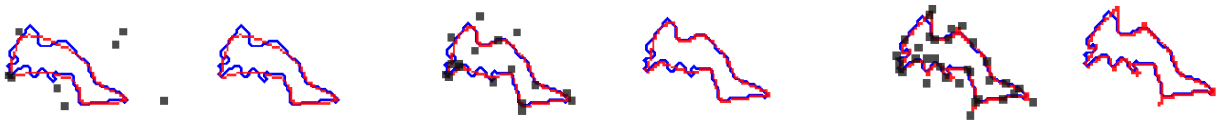
(a) Leaf (DPs=9) (b) Leaf (DPs=16) (c) Leaf (DPs=25)

Fig. 15. Approximation results for Leaf benchmark – original contour in blue, best approximation found in red and control points as gray squares



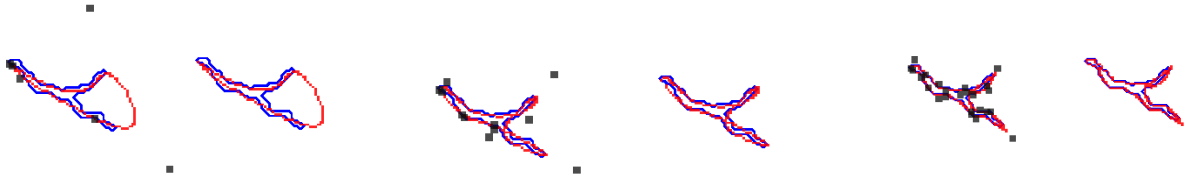
(a) Semicircle (DPs=7) (b) Semicircle (DPs=12) (c) Semicircle (DPs=24)

Fig. 16. Approximation results for Semicircle benchmark – original contour in blue, best approximation found in red and control points as gray squares



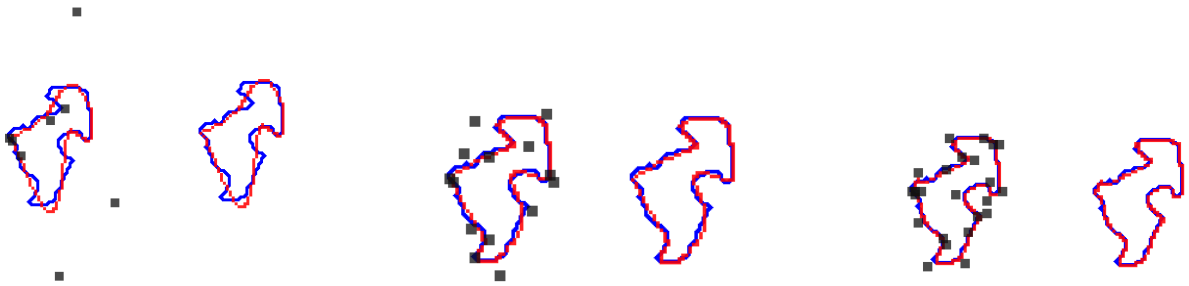
(a) Arlington (DPs=8) (b) Arlington (DPs=17) (c) Arlington (DPs=33)

Fig. 17. Approximation results for Arlington benchmark – original contour in blue, best approximation found in red and control points as gray squares



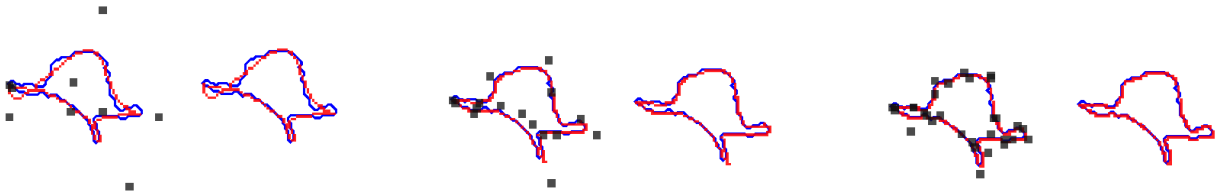
(a) Como (DPs=6) (b) Como (DPs=11) (c) Como (DPs=24)

Fig. 18. Approximation results for Como benchmark – original contour in blue, best approximation found in red and control points as gray squares



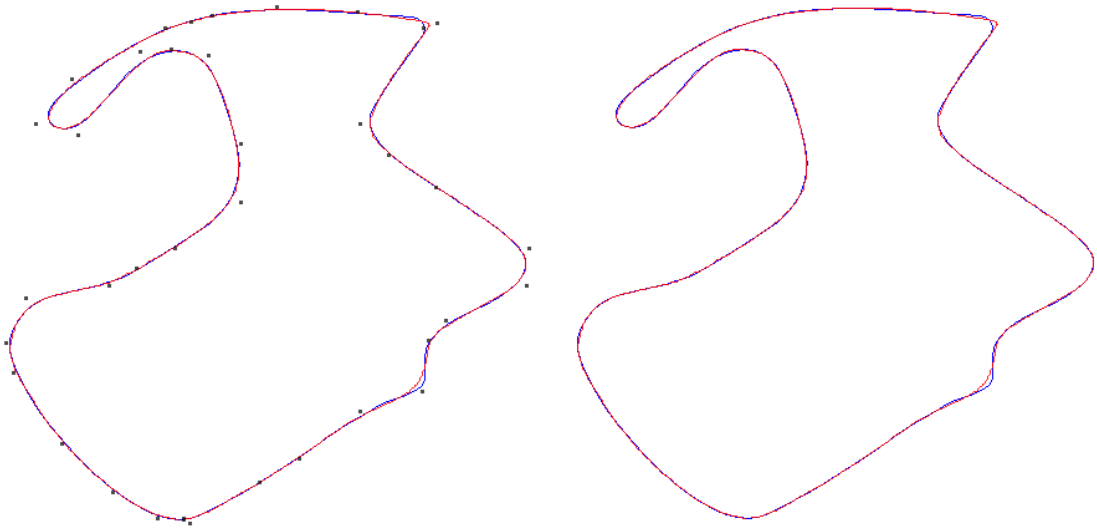
(a) Managua (DPs=8) (b) Managua (DPs=14) (c) Managua (DPs=24)

Fig. 19. Approximation results for Managua benchmark – original contour in blue, best approximation found in red and control points as gray squares

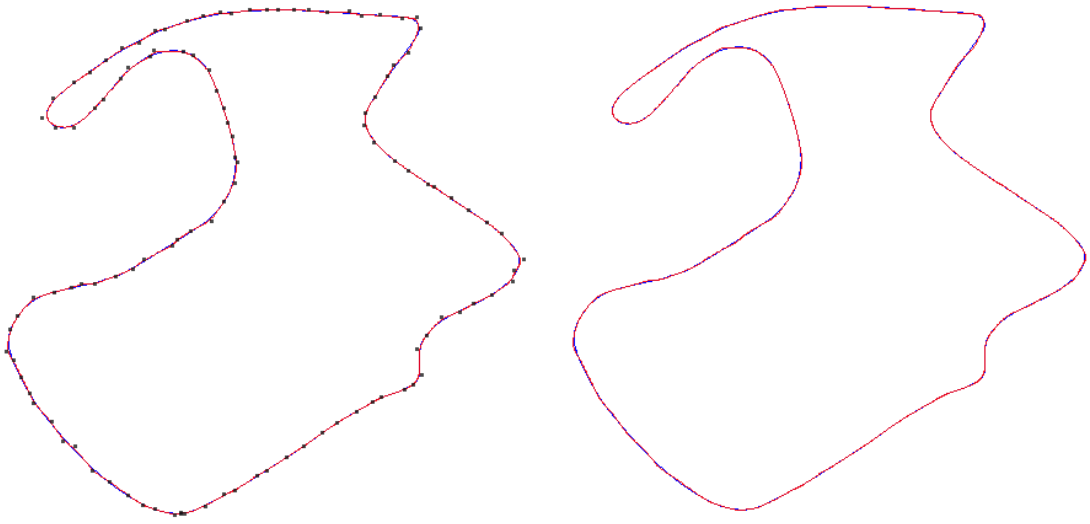


(a) Simcoe (DPs=9) (b) Simcoe (DPs=16) (c) Simcoe (DPs=31)

Fig. 20. Approximation results for Simcoe benchmark – original contour in blue, best approximation found in red and control points as gray squares

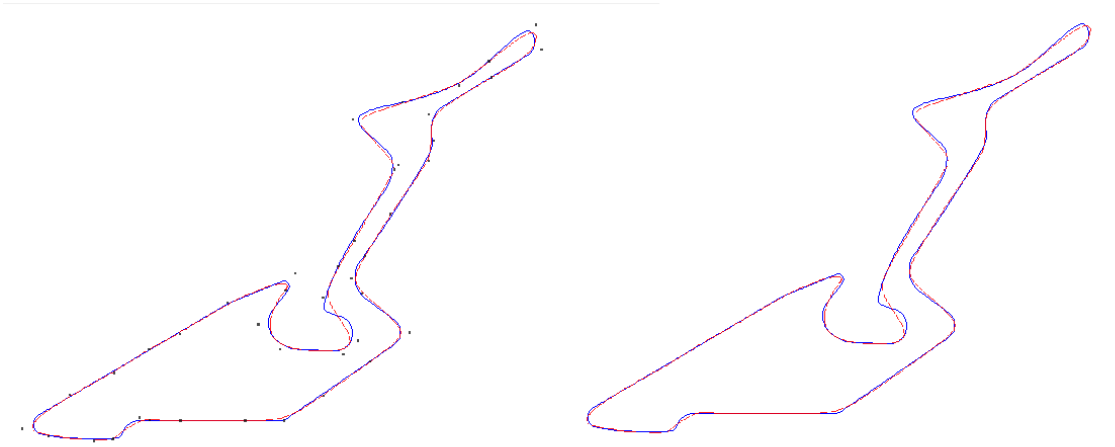


(a) Laguna Seca (DPs=38)

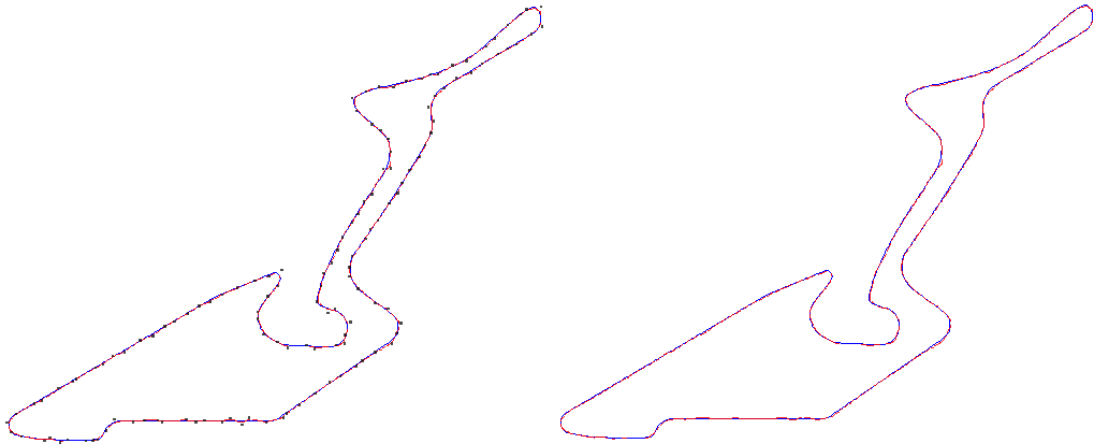


(b) Laguna Seca (DPs=144)

Fig. 21. Approximation results for Laguna Seca benchmark – original contour in blue, best approximation found in red and control points as gray squares



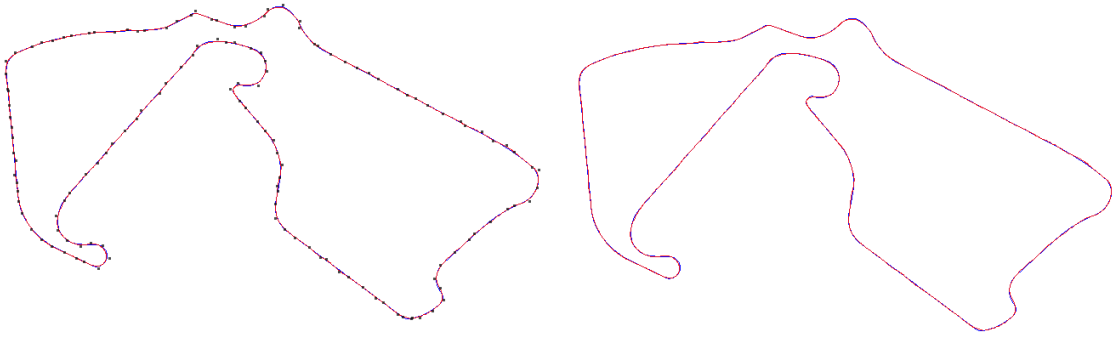
(a) Nürburgring (DPs=43)



(b) Nürburgring (DPs=135)

Fig. 22. Approximation results for Nürburgring benchmark – original contour in blue, best approximation found in red and control points as gray squares

(a) Silverstone (DPs=44)



(b) Silverstone (DPs=138)

Fig. 23. Approximation results for Silverstone benchmark – original contour in blue, best approximation found in red and control points as gray squares