Tiled EvoLisa Image Evolution With Blending Triangle Brushstrokes and Gene Compression DE

Aleš Zamuda and Uroš Mlakar
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova ul. 17, 2000 Maribor, Slovenia
Email: ales.zamuda@um.si, uros.mlakar@um.si

***Abstract*—This paper proposes a set of tiled image evolution approaches for the EvoLisa challenge, where an image is to be approximated using artistic elements, such as brushstrokes. Using differential evolution (DE) optimization algorithm, a lossy image representation with variable number of brushstrokes is evolved thereby. Several different methods to represent or combine a brushstroke on an image canvas are studied, including the control parameters of the proposed methods. An image is tiled and a DE is run on each tile separately. The proposed blending joins multiple brushstrokes over several pixels, while gene compression strives to select only the effective part of the potential full genome, the rendered codon with a limited subset number of brushstrokes. The results show that different proposed algorithms differ significantly in performance, however through a prolonged evolution they all obtain evolved images fairly closely resembling the reference images, which was not demonstrated yet at any previous experiments with EvoLisa.**

I. Introduction

This paper presents evolvable lossy image representation, where the image is represented using a variable number of triangular brushstrokes [1], each consisting of triangle position and color parameters. These parameters for each triangle brush are evolved using differential evolution (DE) [2], [3], which self-adapts the control parameters, including the utilized self-adaptation for number of triangles to draw and compare the generated evolved image to its counterpart reference image. Experimental results show the viability of the proposed encoding and evolution convergence for lossy compression of sample images.

The original challenge of EvoLisa was published by Robert Alsing [4], as a way to display evolutionary algorithm behavior. Later, Ding and Robb [5] have studied the original EvoLisa problem using a PSO, from the image matching perspective. Then, Poca [6] had implemented EvoLisa considering an evolutionary algorithm for image compression on this challenge. Later, Izadi and Ciesielski [1] presented an approach, which is built upon this challenge and uses triangles when trying to build an approximate model of an image. Zamuda and Mlakar [7] compared and extended this EvoLisa approach and applied DE in order to evolve the images. Also, their approach used a modified challenge from [1] where the reconstruction of the image model depended solely on the evolved model (without the need for a reference image like needed in the [1] that used it when drawing pixels to the canvas in deciding which pixels match the reference image to accept them into the evolved canvas). Also, none of the previous approaches demonstrated yielding evolved EvoLisa images that would much closely resemble the reference supplied image, when generally compared to the evolved approximation image. Therefore we conduct several large-scale experiments using a compute grid in order to show that such images can indeed be evolved. For such evolution, we now extend the approach [7] further, trying to encode the image morphology with the triangular brushstrokes more savvy. As the triangle is a basic brush shape for the EvoLisa challenge, we first allow these triangles to be savvily blended on the image, as they cover more pixels than a single point. Each pixel in the canvas also obtains information from more brushes at once. Compared to a line, a triangle allows overlaying and blending of colors over several regional surface pixels, which the simpler lines can not. Also, an arbitrary triangle shape is less constrained than any further point-approximated shape and also more complex shapes can be built further by combining several triangles.

In this paper, the triangle brushstroke encoding differs from [7] and is proposed especially designed for an efficient solution encoding. By possibly minimizing the bit-stream size for representing the image when encoded into a serial file, is a candidate for further binary compression, which might be very relevant in e.g. long-distance or limited-bandwidth channel communication transport, such as e.g. space Internet [8], [9]. Especially this might be useful in scenarios, such as when an autonomous robotic system (e.g. in underwater environments, outer space bodies exploration or other technical system [10]) has plenty of time and autonomous energy to compute, but no communication channel to send the results, e.g. such as during underwater glider stints communication silence [11], [12] or the recently lost Philae antenna comet shadowing.

In the following section, related work is presented, then the proposed approach is defined. In Section 4, experimental results are reported. Section 5 concludes the paper with propositions for future work.

II. Related Work

In this section, related work on evolutionary computer vision, evolutionary art, image representation, and evolutionary optimization using DE are presented. These topics are used in the proposed method, defined in the next section.

A. Evolutionary Optimization using Differential Evolution

DE [2] is a floating-point encoding evolutionary algorithm for continuous global optimization. It has been modified and extended several times with various versions proposed [13], [14]. DE has also been applied to remote sensing image sub-pixel mapping [15], image thresholding [16], [17], facial expression recognition [18], and for image-based modeling using evolutionary computer vision to reconstruct a spatial procedural tree model from a limited set of two dimensional images [19], [20]. In a survey by Neri and Tirronen on DE [21] they concluded that, compared to the other algorithms, a DE extension jDE [3], was superior to the compared algorithms in terms of robustness and versatility over a diverse benchmark set used in the survey. Also, the F and CR control parameters adaptation and self-adaptation of jDE were recently studied by Zamuda and Brest [22], showing that almost all configurations of this mechanism outperformed the same algorithm without this mechanism. Hence, we choose to apply jDE in this paper.

The original DE has a main evolution loop where a population of vectors is computed in each generation. For one generation, counted as g, each vector $\mathbf{x}_i$, $\forall i \in \{1, 2, ..., NP\}$ in the current population of size NP, undergoes DE evolutionary operators, namely the mutation, crossover, and selection. Using these operators, a trial vector (offspring) is produced and the vector with the best fitness value is selected for next generation. For each corresponding population vector, mutation creates a mutant vector $\mathbf{v}_{i,g+1}$ (*'rand/1'* [2]):

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}), \tag{1}$$

where the indexes r_1, r_2, and r_3 are random and mutually different integers generated from a set $\{1, 2, ..., NP\}$, which are also different from i. F is an amplification factor of the difference vector, mostly within the interval $[0, 1]$. The term $\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}$ denotes a difference vector, which is named amplified difference vector after multiplication with F. The mutant vector $\mathbf{v}_{i,g+1}$ is then used for recombination, where with the target vector $\mathbf{x}_{i,g}$ a trial vector of $u_{i,j,g+1}$ is created, e.g. using binary crossover:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1}, & \text{if } rand(0,1) \leq CR \text{ or } j = j_{\text{rand}}, \\ x_{i,j,g} & \text{otherwise,} \end{cases} \tag{2}$$

where CR denotes the crossover rate, $\forall j \in \{1, 2, ..., D\}$ is a j-th search parameter of D-dimensional search space, $rand(0, 1) \in [0, 1]$ is a uniformly distributed random number, and j_{rand} is a uniform randomly chosen index of the search parameter, which is always exchanged to prevent cloning of target vectors.

Finally, the selection operator evaluates and compares the trial to current vector and propagates the fittest one:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{if } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{3}$$

B. Evolutionary Computer Vision: Evolutionary Art and Image-based Modeling

Image-based approaches to modeling include processing of images, e.g. two-dimensional, from which after segmentation certain features are extracted and used to represent a geometrical model [23]. For art drawings modeling, automatic evolutionary rendering has been applied [24], [25]. In [26] animated artwork is evolved using an evolutionary algorithm. Heijer and Eiben evolved pop-art two-dimensional scalable vector graphics (SVG) images [27]. They defined genetic operators for SVG to evolve representational images using SVG to evolve new images, different from source images, leading to new and surprising images for pop-art. Bergen and Ross [28] interactively evolved vector graphics images using a genetic algorithm, where solid-colored opaque or translucent geometric objects or mosaic tile effects with bitmap textures were utilized; they considered the art aspect of the evolved image and multiple possible outcomes due to evolution stochastics and concluded to investigate vector animation of the vectorized image. Coia and Ross [29] also considered three-dimensional morphology evolution, in order to evolve building-inspired three dimensional models using fractal procedures. Then, Bergen and Ross also evolved aesthetically pleasing three dimensional images utilizing marching cubes algorithm on voxels rendered from a short string, one production per symbol, context-free L-systems [30], and multi-objective optimization. Pospichal et al. [31] also used graphics processing units to accelerate grammatical evolution. Beaumont and Stepney [32] presented an approach for reconstruction of a procedural model, similar to a work of Ashlock et al. [33]. An extended reconstruction approach to the domain of three-dimensional procedural models suitable to approximate woody plants without user interaction is presented in [19], vector-encoded in [20], and evolved multi-objectively in [34], where the morphology of a 3D tree as a huge state-of-the-art challenge in computer graphics, is reconstructed being guided by computer vision in large-scale DE by serializing the tree nodes into a fixed-size DE vector.

Izadi et al. [1] evolved triangular brushstrokes using genetic programming for two-dimensional images, using unguided and guided search on a three or four branch genetic program, where roughly 5% similarity with reference images was obtained on average per pixel. The presented approach in this paper is built based after [1], [7], by addressing and also extending the EvoLisa challenge. After extending the challenge with tiling and blending, we optimize it using DE, which is described in the next section.

All the DE approaches to tree programming however, require to set the vector limit size. In the approach in this paper, we want to have a variable number of triangular brushstrokes to represent an image, and want to have these recombined not depending on the initial index position within the vector. We use DE with a vectorized tree encoding and a fixed size tree-like chromosome vector formed from breadth-first search upon an imaginary full tree. A question arises, how

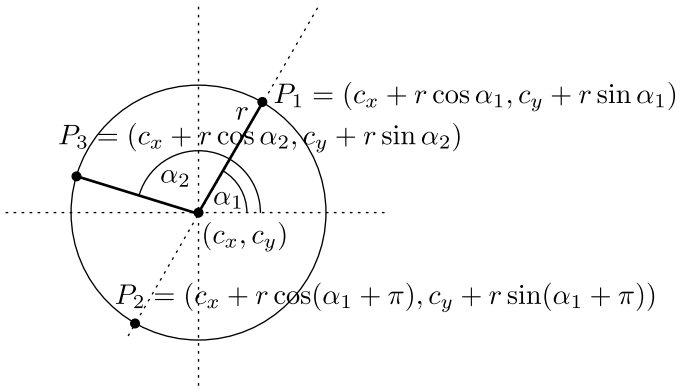


Figure 1. The triangle brush definition and the circumscribed circle.

to set the DE vector dimension size, dependent on this tree size, representing a variable number of triangular brushstrokes and then exchange information among individual vectors with different numbers of triangular brushstrokes being rendered? For variable lengths genotypes, genetic programming seems a viable solution [35]. In [1], namely genetic programming is used to represent and recombine the evolutionary material of triangular brushstrokes within a tree-like genetic programming algorithm. For DE, grammar evolution is introduced by Neill and Brabazon [36] and later, an initial study on shape grammars is introduced to evolve binary images of shapes [37]. Evolutionary design using grammatical evolution and shape grammars to designing a shelter by evolution of three-dimensional morphology shapes was then presented [38]. Moraglio and Silva [39] introduce a geometric DE for genetic programs by acting on a tree represented within DE. Tree structures evolution is also addressed by Veenhuis [40], where TreeDE by Neill and Brabazon [36] is improved in the performance of grammatical DE, which requires an additional low-level parameter (the tree depth of solutions) that has to be set beforehand. Moreover, evolved programs of TreeDE do not include random constants and Fonlupt et al. [41] extend grammar evolution by linear sequences of genetic programs [42] as continuous schemes that are evolved by DE. In [41] evolution of trees among DE and CMA-ES mechanisms is compared and suspected on the suggestions by Das and Suganthan, that CMA-ES is competitive up to 100 variables, but it is difficult to extend it to higher dimensional problems due mainly to the cost of computing and updating the covariance matrix [13]; Fonlupt et al. [41] conclude that for both regression and artificial ants, CMA-based algorithm performs poorly, compared with a DE-based algorithm, with the same representation of solutions; they attribute the better DE performance to its fundamental differences in behavior and robustness, and CMA-ES lacking elitism. DE variants, CMA-ES, and other similar evolutionary algorithms were recently also compared on a real application problem domain of underwater robotics, where a jDE-like variant outperformed other algorithms [11]. These DE performances give clear implications to consider DE as an optimizer in this paper.

III. Differential Evolution for Self-adaptive Triangular Brushstrokes With Tiling and Blending

In this section, the encoding aspect, genotype-phenotype rendering, and evaluation mechanisms of the proposed approach are defined, which is greatly similar to and, at matching parts marked with [7]; it differs mainly in the newly proposed tiling, blending, and genotype compression.

A. Encoding Aspect

To tile the image, we divide it to a map of same sized tiles, so that each tile is evolved by a separately initialized DE population, effectively dividing the image dimensions $R_x \times R_y$ by the number of tiles. Therefore, the number of fitness evaluations denotes the sum of individual tiles DE evaluations, i.e. for comparing as whole image evaluation, one needs to divide the MAXFES by the number of tiles, same as in [7].

We encode an individual image tile into a DE vector as follows. A DE vector $\mathbf{x} = (x_1, x_2, ..., x_{7T^{\max}}, F, CR, T^{\mathrm{L}}, T^{\mathrm{U}})$ is composed of floating-point scalar values packed sequentially as $\{x_j : \forall j \in \{1, ..., D+4\}\}$, starting with a triangles-coding part of length $D = 7T^{\max}$ ($D = 8T^{\max}$ was used in [7] as for the uncompressed genes), and the rest are the self-adaptive control parameters of the vector to be used during the evolution. The self-adaptive control parameters part of the $\mathbf{x}$ vector encodes and uses the scaling factor F and crossover rate CR as in the jDE [3]; then the $T_i^{\mathrm{L}}, T_i^{\mathrm{U}} \in \{1, 2, ..., T^{\max}\}$ control parameters follow. Therefore the full length of $\mathbf{x}$ is $|\mathbf{x}| = 7T^{\max} + 4$.

The self-adaptive T_i^{L} and T_i^{U} control parameters determine index-wise triangles encoded in the vector $\mathbf{x}$ to be used for rendering the evolved image, i.e. the portion of $\mathbf{x}$ to render an image is $\{x_j : \forall j \in \{T^{\mathrm{L}}, ..., T^{\mathrm{U}}\}\}$. [7]

As in the approach in [7], we utilize to have the whole vector represented as a triangle set, organized similar to serializing a tree in a linear vector when visiting nodes by depth-first search. By this representation, the leaf nodes are most exposed to being cut-off, whereas the root node is encoded in the middle of the vector and the near-root nodes are therefore more protected and being retained, since they are more anchored due to cut-offs mostly around the codon edges T^{L} and T^{U}. After being included in a new trial vector, all nodes have an equal probability to have their triangle data changed [7]. In this way, the T^{L} and T^{U} allow us to render only a sub-portion of the triangles set, similar as if to take an inseparable portion of a GP tree traversal as in [1]. This gives us an arbitrary length render set, and keeps the crossover of anti-codon to help us find the number of triangles $T_i \in \{1, 2, ..., T^{\max}\}$ which is most suitable for the image approximation:

$$T_i = \begin{cases} T_i^{\mathrm{U}} - T_i^{\mathrm{L}} + 1 & \text{if } T_i^{\mathrm{L}} < T_i^{\mathrm{U}} \\ (T^{\max} - T_i^{\mathrm{L}}) + T_i^{\mathrm{U}} & \text{otherwise.} \end{cases} \quad (4)$$

The T_i^{L} and T_i^{U} are updated similar to the F_i control parameter, i.e. using the jDE self-adaptation mechanism [7].

Table I

EXPERIMENTS CLASS 1 (BASE), METHOD 0: OBTAINED FITNESS OVER $T_{\max}$ AND R_x (R_y): TEST INSTANCES BABOON (3) AND LENA (4). THE BEST SETTINGS ARE HIGHLIGHTED FOR EACH IMAGE (BOLD FONT).

		Baboon (3)				Lena (4)			
$T_{\max}$	R_x (R_y)	Best	Worst	Average	STD	Best	Worst	Average	STD
10	8	14.92	15.3	15.10	0.11	14.72	15.12	14.96	0.10
10	16	10.18	10.73	10.49	0.14	9.02	9.89	9.42	0.19
10	24	9.19	9.65	9.41	0.13	7.96	8.68	8.21	0.16
10	32	9.24	9.95	9.58	0.17	8.17	8.72	8.46	0.16
10	40	9.13	9.75	9.42	0.16	7.92	8.68	8.22	0.20
10	50	9.54	10.59	10.10	0.27	8.78	9.91	9.31	0.27
10	100	12.4	13.98	13.20	0.52	10.69	12.62	11.67	0.45
20	8	14.21	14.63	14.40	0.09	14.17	14.53	14.36	0.10
20	16	10.94	11.38	11.16	0.11	10.11	10.55	10.33	0.11
20	24	9.7	10.46	10.08	0.15	8.64	9.49	9.07	0.17
20	32	9.51	10.51	9.97	0.26	8.15	9.41	8.66	0.31
20	40	8.61	9.44	9.07	0.21	**7.1**	**8.01**	**7.54**	0.27
20	50	**8.59**	**9.49**	**8.92**	0.20	7.57	8.46	7.94	0.23
20	100	10.29	12.04	11.05	0.44	9.17	11.2	9.88	0.52
30	8	13.71	14.05	13.91	0.09	13.77	14.1	13.96	0.08
30	16	11.21	11.68	11.45	0.11	10.46	10.82	10.66	0.10
30	24	9.89	10.62	10.39	0.14	9.23	10.01	9.62	0.19
30	32	9.85	10.81	10.32	0.22	8.81	9.63	9.24	0.21
30	40	8.98	9.79	9.41	0.18	7.58	8.56	8.15	0.26
30	50	8.59	9.44	9.09	0.22	7.4	8.64	7.99	0.29
30	100	9.54	11.41	10.28	0.49	8.41	10.59	9.26	0.56
40	8	13.43	13.79	13.61	0.09	13.54	13.82	13.71	0.08
40	16	11.27	11.65	11.49	0.08	10.67	11.01	10.87	0.09
40	24	10.08	10.73	10.49	0.14	9.55	10.08	9.82	0.14
40	32	10.02	10.78	10.48	0.18	9.15	9.89	9.56	0.20
40	40	9.29	10.06	9.62	0.19	7.96	8.7	8.36	0.19
40	50	8.76	9.81	9.32	0.27	7.59	9.05	8.44	0.33
40	100	9.42	12.04	10.44	0.62	8.12	10.5	9.28	0.54
50	8	13.32	13.56	13.41	0.06	13.35	13.72	13.53	0.09
50	16	11.38	11.77	11.60	0.09	10.82	11.2	11.07	0.09
50	24	10.26	10.86	10.64	0.17	9.77	10.28	10.03	0.13
50	32	10.28	10.89	10.61	0.15	9.47	10.2	9.81	0.20
50	40	9.6	10.25	9.83	0.13	8.36	9.07	8.67	0.20
50	50	9.15	10.05	9.60	0.22	8.08	9.18	8.68	0.29
50	100	9.51	11.92	10.61	0.55	8.29	10.3	9.28	0.57
60	8	13.12	13.5	13.28	0.09	13.23	13.64	13.44	0.08
60	16	11.45	11.86	11.62	0.09	10.93	11.31	11.16	0.09
60	24	10.52	10.9	10.72	0.10	10	10.45	10.21	0.12
60	32	10.32	10.89	10.64	0.14	9.53	10.24	9.92	0.20
60	40	9.58	10.4	9.94	0.18	8.46	9.02	8.77	0.16
60	50	9.08	10	9.69	0.22	8.49	9.4	8.95	0.27
60	100	9.57	11.32	10.66	0.41	8.34	10.25	9.53	0.53
70	8	13.05	13.32	13.19	0.07	13.18	13.49	13.34	0.09
70	16	11.49	11.86	11.68	0.09	11.01	11.44	11.22	0.10
70	24	10.62	11	10.80	0.11	10.13	10.63	10.34	0.12
70	32	10.46	11.06	10.76	0.17	9.69	10.42	10.11	0.19
70	40	9.84	10.25	10.03	0.10	8.76	9.21	8.98	0.13
70	50	9.37	10.06	9.81	0.18	8.41	9.47	9.09	0.24
70	100	9.69	11.55	10.82	0.43	9.04	10.81	9.88	0.44
80	8	13	13.2	13.11	0.06	13.19	13.43	13.31	0.07
80	16	11.54	11.91	11.68	0.09	11.09	11.44	11.27	0.10
80	24	10.63	11.26	10.92	0.13	10.22	10.66	10.39	0.11
80	32	10.55	11.04	10.81	0.15	9.79	10.42	10.17	0.15
80	40	9.81	10.46	10.11	0.15	8.86	9.36	9.07	0.11
80	50	9.61	10.15	9.91	0.15	8.64	9.7	9.19	0.22
80	100	9.97	11.6	10.94	0.38	8.49	10.61	9.89	0.54
90	8	12.94	13.13	13.05	0.06	13.13	13.39	13.27	0.07
90	16	11.55	11.92	11.70	0.10	11.12	11.6	11.35	0.11
90	24	10.8	11.12	10.93	0.08	10.32	10.73	10.53	0.10
90	32	10.58	11.1	10.87	0.13	9.95	10.48	10.24	0.14
90	40	9.78	10.32	10.06	0.14	8.9	9.4	9.17	0.14
90	50	9.66	10.33	9.93	0.15	8.94	9.77	9.29	0.22
90	100	10.27	11.39	10.91	0.27	9.63	10.78	10.25	0.27
100	8	12.87	13.12	13.02	0.06	13.09	13.33	13.21	0.07
100	16	11.6	12	11.73	0.08	11.22	11.61	11.39	0.10
100	24	10.77	11.21	10.96	0.11	10.35	10.71	10.57	0.09
100	32	10.64	11.04	10.88	0.10	10.02	10.57	10.30	0.13
100	40	9.95	10.47	10.18	0.13	9.05	9.67	9.31	0.15
100	50	9.65	10.31	10.01	0.17	9.08	9.68	9.44	0.16
100	100	10.49	11.58	11.03	0.26	9.4	10.77	10.22	0.34

Table II

EXPERIMENTS CLASS 1 (BASE), METHOD 0: OBTAINED FITNESS OVER $T_{\max}$ AND R_x (R_y): TEST INSTANCES PLANE (5) AND PAPRICA (7). THE BEST SETTINGS ARE HIGHLIGHTED FOR EACH IMAGE (BOLD FONT).

		Plane (5)				Paprica (7)			
$T_{\max}$	R_x (R_y)	Best	Worst	Average	STD	Best	Worst	Average	STD
10	8	14.99	15.89	15.45	0.20	14.42	14.89	14.70	0.12
10	16	8.85	9.67	9.33	0.21	9.89	10.43	10.14	0.14
10	24	7.56	8.28	7.94	0.18	8.79	9.54	9.17	0.17
10	32	7.96	8.85	8.26	0.18	9.19	9.95	9.59	0.19
10	40	7.29	7.91	7.56	0.17	9.02	9.53	9.25	0.13
10	50	7.25	8.3	7.77	0.21	10.23	11.4	10.80	0.29
10	100	8.69	10.26	9.19	0.37	12.98	15.01	14.07	0.45
20	8	14.6	15.09	14.83	0.11	14.08	14.47	14.30	0.10
20	16	10.23	10.81	10.52	0.16	10.64	11.17	11.02	0.12
20	24	8.53	9.54	8.93	0.25	9.64	10.47	9.99	0.21
20	32	8.27	8.94	8.62	0.20	9.41	10.2	9.82	0.21
20	40	7.04	8.13	7.52	0.25	**8.35**	**9.12**	**8.73**	0.20
20	50	**6.78**	**7.8**	**7.30**	0.27	8.95	9.74	9.37	0.19
20	100	7.61	8.77	8.25	0.26	11.37	12.81	11.94	0.45
30	8	14.08	14.49	14.30	0.11	13.79	14.21	14.03	0.10
30	16	10.57	11.19	10.87	0.13	11.08	11.59	11.36	0.14
30	24	9.1	9.82	9.55	0.18	10.15	10.86	10.56	0.16
30	32	8.63	9.62	9.25	0.25	9.83	10.59	10.30	0.16
30	40	7.44	8.57	7.98	0.25	8.54	9.66	9.04	0.25
30	50	6.8	8.18	7.53	0.40	8.89	10.04	9.41	0.28
30	100	7.72	9.17	8.25	0.37	10.74	12.09	11.17	0.31
40	8	13.63	14.12	13.90	0.09	13.65	14	13.84	0.09
40	16	10.78	11.24	10.97	0.13	11.39	11.81	11.61	0.10
40	24	9.33	10.04	9.75	0.16	10.58	11.34	10.80	0.17
40	32	9.13	9.96	9.54	0.20	10.2	11.09	10.68	0.20
40	40	7.87	8.71	8.33	0.21	9.15	9.75	9.49	0.16
40	50	7.37	8.47	7.90	0.29	8.9	10.39	9.72	0.31
40	100	7.44	8.96	8.36	0.45	9.99	12	11.11	0.55
50	8	13.39	13.79	13.61	0.10	13.6	13.9	13.75	0.08
50	16	10.83	11.28	10.99	0.10	11.54	12	11.78	0.12
50	24	9.58	10.2	9.89	0.14	10.73	11.62	11.13	0.20
50	32	9.34	10.2	9.81	0.21	10.33	11.31	10.91	0.20
50	40	8.16	8.79	8.46	0.17	9.2	9.97	9.70	0.21
50	50	7.46	8.6	8.10	0.25	9.26	10.61	10.04	0.30
50	100	7.51	9.33	8.50	0.45	10.26	11.88	11.23	0.46
60	8	13.26	13.65	13.39	0.08	13.5	13.91	13.69	0.10
60	16	10.8	11.3	11.04	0.11	11.67	12.07	11.90	0.10
60	24	9.62	10.3	10.01	0.18	11.03	11.64	11.36	0.15
60	32	9.46	10.19	9.85	0.18	10.91	11.47	11.18	0.13
60	40	8.22	8.89	8.54	0.16	9.52	10.18	9.87	0.17
60	50	7.85	8.52	8.27	0.19	9.84	10.9	10.37	0.31
60	100	7.86	9.24	8.77	0.30	10.91	12.16	11.64	0.38
70	8	13.06	13.4	13.23	0.08	13.46	13.78	13.62	0.08
70	16	10.81	11.28	11.00	0.10	11.7	12.23	11.97	0.12
70	24	9.76	10.28	10.04	0.13	11.15	11.73	11.47	0.14
70	32	9.61	10.14	9.82	0.14	11.02	11.55	11.28	0.12
70	40	8.28	8.9	8.63	0.15	9.78	10.5	10.07	0.19
70	50	8.06	8.72	8.36	0.18	10.08	10.74	10.44	0.17
70	100	8.25	9.28	8.82	0.27	10.93	12.32	11.73	0.32
80	8	12.94	13.22	13.11	0.07	13.46	13.76	13.60	0.07
80	16	10.61	11.17	10.98	0.12	11.88	12.34	12.05	0.10
80	24	9.71	10.32	10.04	0.14	11.28	11.87	11.61	0.14
80	32	9.69	10.3	9.92	0.16	11.07	11.74	11.41	0.15
80	40	8.32	9.05	8.78	0.16	9.8	10.45	10.14	0.16
80	50	7.83	8.57	8.36	0.17	9.9	11.17	10.64	0.26
80	100	8.4	9.32	8.86	0.20	11.17	12.54	11.94	0.33
90	8	12.85	13.14	12.99	0.07	13.48	13.67	13.59	0.05
90	16	10.92	11.22	11.04	0.08	11.92	12.32	12.13	0.11
90	24	9.91	10.37	10.13	0.11	11.4	11.98	11.71	0.15
90	32	9.6	10.18	9.91	0.14	11.02	11.67	11.44	0.14
90	40	8.57	9.12	8.80	0.13	10.05	10.68	10.36	0.15
90	50	8.1	8.68	8.39	0.16	9.89	11.03	10.73	0.24
90	100	8.41	9.14	8.91	0.21	10.71	12.58	11.97	0.44
100	8	12.79	13.06	12.95	0.08	13.41	13.71	13.58	0.07
100	16	10.79	11.18	10.98	0.08	11.92	12.34	12.15	0.11
100	24	9.84	10.24	10.07	0.12	11.58	12.08	11.84	0.12
100	32	9.57	10.39	10.00	0.15	11.29	11.82	11.51	0.13
100	40	8.59	9.23	8.81	0.15	10.18	10.78	10.44	0.16
100	50	8.11	8.8	8.46	0.17	10.47	11.22	10.79	0.19
100	100	8.22	9.42	8.92	0.24	11.37	12.57	12.08	0.32

B. Genotype-Phenotype Rendering

A DE vector $\mathbf{x}_i, \forall i \in \{1, 2, ..., NP\}$ encoded using floating-point numbers $x_{i,j}, \forall j \in \{1, 2, ..., D+4\}$ constituting a genotype is rendered into a phenotype image tile $\mathbf{Z}_i = \{\mathbf{Z}_{i,x,y}\}$ of R_{x} width and R_{y} height in pixels, to be compared against a matching position reference image tile $\mathbf{Z}^*$ as follows.

The triangle brushstrokes (Figure 1) are represented as $(c_x, c_y, r, \alpha_1, \alpha_2, b_f, b_k)$, where $c_x \in \{0, 1, ..., R_{\mathrm{x}}\}$, $c_y \in \{0, 1, ..., R_{\mathrm{y}}\}$, and $r \in [0, R_{\mathrm{x}}/\sqrt{T_{\max}}]$ define the circumscribed circle center and radius for the triangle to be rendered; mutually different $\alpha_1 \in \{0°, 22.5°, 45°, ..., 337.5°\}$ and $\alpha_2 \in \{0°, 22.5°, ..., 157.5°\}$ (in [7] these were $\alpha_1 \in [1°, 360°)$ and $\alpha_2 \in [1°, 180°)$) define the points of this triangle on its circumscribed circle. The b_f is a lookup into a color frequency defined in the spectrum as $\mathbf{b}^{RGB} = \{b^R, b^G, b^B\}$, $b^{\mathrm{R}}, b^{\mathrm{G}}, b^{\mathrm{B}} \in [0, 255]$ which are the indices of evenly-spaced color components (using a color mapping model, with Euclidean difference value of 51 per component) of the brush for the triangle contained pixels (in [7] the genotype was laid as $(c_x, c_y, r, \alpha_1, \alpha_2,\ b^{\mathrm{Y}}, b^{\mathrm{Cb}},\ b^{\mathrm{Cr}})$, where $b^{\mathrm{Y}} \in [16, 236)$, $b^{\mathrm{Cb}} \in [16, 241)$, and $b^{\mathrm{Cr}} \in [16, 241)$ were these color components). The b_k is the evolved transparency factor of a brush, multiplying the brush color components.

The triangles vertices encoded by i-th DE vector construct T_i triangles, each triangle $\mathbf{T}_k = (c_{x,k}, c_{y,k}, r_k, \alpha_{1,k}, \alpha_{2,k}), \forall k \in \{1, 2, ..., T_i\}$ ($\mathbf{T}_k$ being packed as $\mathbf{x}_i = \{x_{i,j}\}$, $j = 7k + m$, $m \in \{1, 2, ..., 7\}$), defining vertices of a triangle $P_{1,k}$, $P_{2,k}$, and $P_{3,k}$:

$$P_{1,k} = (\lfloor c_{x,k} + r_k \cos \alpha_{1,k} \rfloor, \lfloor c_{y,k} + r_k \sin \alpha_{1,k} \rfloor), \quad (5)$$

$$P_{2,k} = (\lfloor c_{x,k} + r_k \cos(\alpha_{1,k} + \pi \rfloor, \lfloor c_{y,k} + r_k \sin(\alpha_{1,k} + \pi \rfloor), \quad (6)$$

$$P_{3,k} = (\lfloor c_{x,k} + r_k \cos \alpha_{2,k} \rfloor, \lfloor c_{y,k} + r_k \sin \alpha_{2,k} \rfloor). \quad (7)$$

For each triangle T_k, a solid color is rendered without anti-aliasing over the triangle brush area rasterizing [43] with diminishing it by transparency factor of b_k. This is analogous to blending the triangle as part-transparent layer within the evolved image $\mathbf{Z}_i = \sum_k \mathbf{z}_{k,x,y}$ and computes R, G, and B color layers for the pixels of the i-th individual:

$$\mathbf{z}_{k,x,y} = \sum_{\mathbf{T}_k \text{ over } (x,y)} \mathbf{b}_{k,x,y} = \sum_{\mathbf{T}_k \text{ over } (x,y)} \left\lfloor b_k \mathbf{b}^{\mathrm{RGB}}_{k,x,y} \right\rfloor, \quad (8)$$

where $\mathbf{T}_k$ over (x, y) denotes each triangle being rendered adding to the pixel (x, y) such that $\mathbf{b}_{k,x,y}$ contains rendered pixels of a brushstroke. The blended EvoLisa is a different from basic EvoLisa and also different from e.g. triangulation or mosaics, because the brushstrokes share the final canvas color (inseparably). Triangles defined possibly over the edges of an image canvas are drawn by clipping away pixels outside of the canvas area. As all parameters are evolved as floating-point scalar values in DE, they are initialized uniform randomly.

C. Evaluation

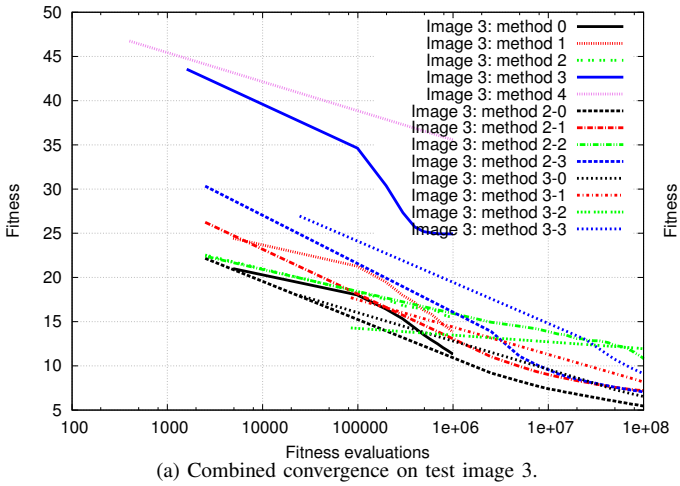
Evaluation of the phenotype image $\mathbf{Z}_i$ to be compared against a reference image $\mathbf{Z}^*$ is as follows. A reference image $\mathbf{Z}^*$ is represented as RGB-encoded colored pixels integer values in layers $\mathbf{Z}^* = \{(z^{\mathrm{R}}_{x,y}, z^{\mathrm{G}}_{x,y}, z^{\mathrm{B}}_{x,y})\}$. To obtain a difference assessment value, the comparison metric for comparing an evolved image is the degree of residual noise (RN), which is used as the percentage of the changed pixels in the resulting evolved image that are incorrectly represented when compared to the original reference image (here, the $\boldsymbol{Z}$ is an evolved image $\boldsymbol{Z}_i$, which is being compared to the reference image $\boldsymbol{Z}^*$ [7]):

$$f(\boldsymbol{Z}) = 100 \times \frac{\sum_{y=0}^{R_{\mathrm{y}}-1} \sum_{x=0}^{R_{\mathrm{x}}-1} \mid z^{*\mathrm{R}}_{x,y} - z^{\mathrm{R}}_{x,y} \mid + \mid z^{*\mathrm{G}}_{x,y} - z^{\mathrm{G}}_{x,y} \mid + \mid z^{*\mathrm{B}}_{x,y} - z^{\mathrm{B}}_{x,y} \mid}{3 \times 255 \times R_{\mathrm{x}} R_{\mathrm{y}}}. \quad (9)$$

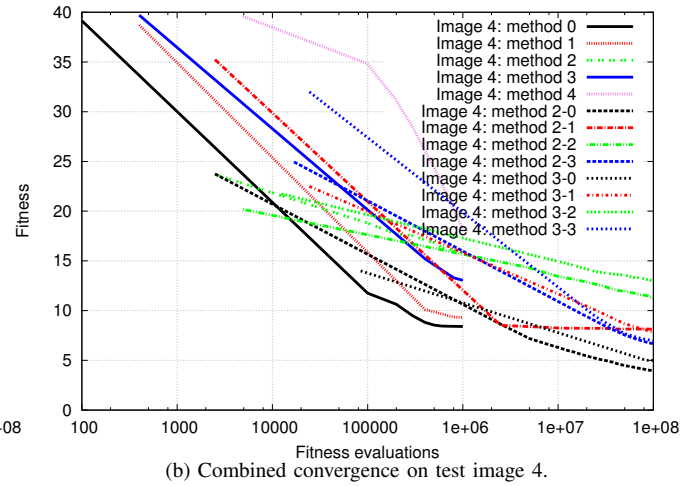
IV. Experiments

We have tested the proposed approach on four different images (`Baboon (3)`, `Lena (4)`, `Plane (5)`, and `Vegatables (7)` – we chose their numberings arbitrarily). The following experiments assess the viability of the approach on different control parameters, each with several independent runs. The parameter sets are as follows: the DE population size NP was set at 100, $T_{\max} \in \{10, 20, ..., 100\}$, and $R_x = R_y \in \{8, 16, 24, 32, 40, 50, 100\}$; thereby for each run $RNi \in$\{0,1,...,10\} this counted total of 70 parameter sets, i.e. 700 independent runs per one method for each of 4 images. We have used the following 5 methods: 0) blending filled triangles and jDE, 1) blending filled triangles and canonical DE with $F = 0.5$, $CR = 0.9$, 2) filled triangles without blending, jDE, 3) empty triangles without blending, jDE, 4) lines between first two encoded points instead of a triangle, jDE. We have then experimented with these 5 different methods on different classes of experiments, i.e.: base class 1) setting MAXFES to 1e+6; class 2) setting MAXFES to 1e+8 and only running with the best settings $T_{\max}, R_x = R_y$ where a best setting was found at the base class (1) experiment, and also $R_x = R_y = 100$ for that $T_{\max}$; and class 3) setting parameter NP initially at 500 and halving it through 4 population reductions [44], while keeping everything else same as for class 2 (in order to also study the parameter NP). For image rendering, C# language and basic GDI+ implementation was used (the [7] studied only one approach, the class 1, method 2, with different settings). For speeding up the experiments execution, a compute grid from the SLAIS GRID initiative under runtime environment "APPS/FERI/MONO" using Scientific Linux was utilized to parallelize the independent runs, i.e. several ten thousand processing task jobs were submitted and successfully retrieved from the SLURM scheduler.

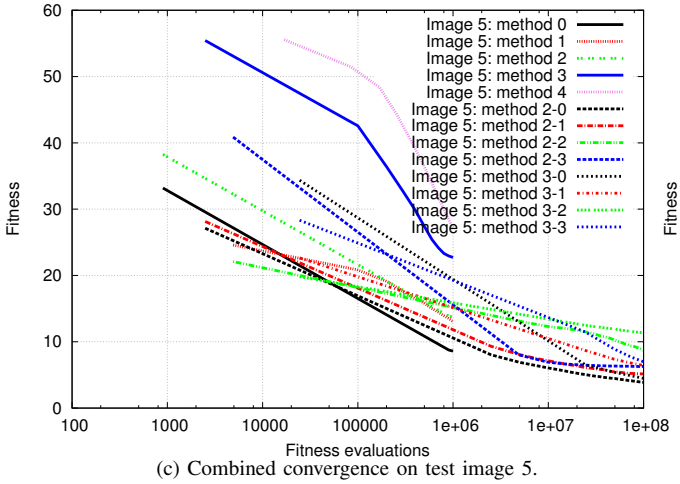
The obtained runs final fitness values over $T_{\max}$ and NP are collected in Tables I and II for the basic experiments class (1) with method 0. In this class the method 0 obtained the best settings as marked (rank matching also for the best/worst/average) for all images, i.e. overall best fitness best/worst/average values obtained were 8.59/9.49/8.9, 7.1/8.01/7.54, 6.78/7.8/7.3, and 8.35/9.12/8.73 for the 4 images, respectively. As can generally be observed from these results, increasing $T_{\max}$ improved image fitness (i.e. best $T_{\max}$ was seen at 40 or 50, with increasing performance from 10 on),



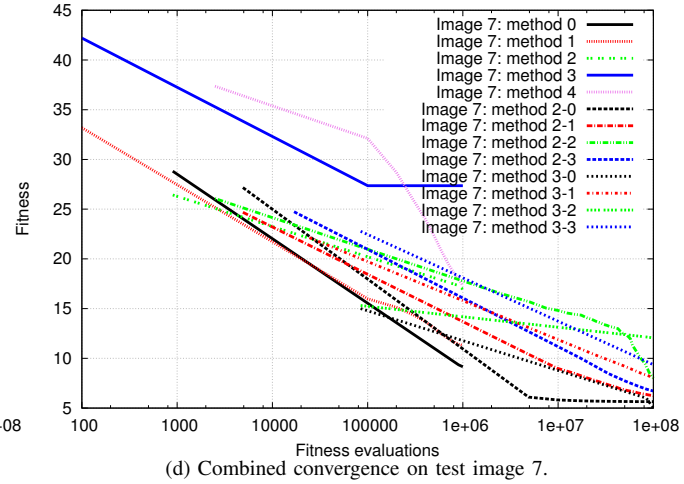
(a) Combined convergence on test image 3.



(b) Combined convergence on test image 4.

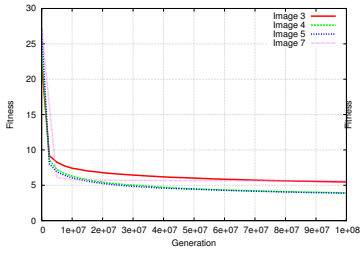


(c) Combined convergence on test image 5.

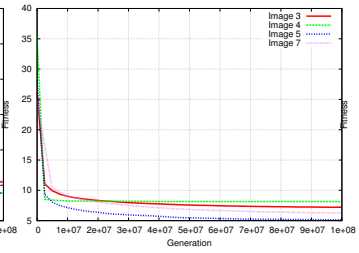


(d) Combined convergence on test image 7.

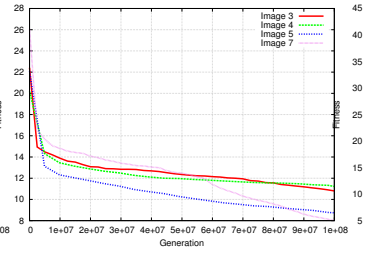
Figure 2. Combined convergences on test images for different algorithms, aligned on matching FES (plotted each 10 generations, log-scale; note that due to the different NP settings, algorithm's first output is taken at different initial FES). When fitness value does not change (stagnation), a certain fitness convergence line is not drawn further. For all methods, with lowering the RN degree to attain, the difficulty increases significantly exponentially with FES.



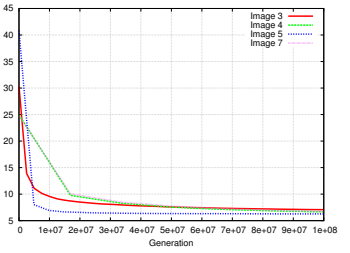
(a) Convergence: method 0.



(b) Convergence: method 1.



(c) Convergence: method 2.



(d) Convergence: method 3.

Figure 3. Convergences on test images for different methods (experiments class 2).

however when $T_{\max}$ was increased too much, the lack of fitness count however could not optimize all triangles fast enough (i.e. values of $T_{\max}$ at 100 were not as good, signaling that we have successfully obtained the coarse $T_{\max}$ limits for the experiment settings at hand). The tables are shown only for this method, because it has yielded best results, as can be seen from respective method's best runs fitness convergences displayed in Figure 2; the graphs comparatively display best performing runs convergences, joined for all experiments classes and methods, separately for each image. As can be observed from these convergence graphs, the different proposed algorithms differ significantly in performance, however through a prolonged evolution (up to 1e+6 or 1e+8 FES) the proposed best methods obtain fairly good results (in log-scale time/FES).

Also, as the method 0 for class 2 can be seen as the best from Figure 2 (this is the proposed method, i.e. tiling with blending filled triangles, using basic self-adaptive DE; the population size reduction as parameterized, did not yet yield better results and would perhaps need more configurations experimentation at future work), the fitness convergences of its best obtained runs are further displayed in detail shown for class 2 of experiments for each method for the different

Figure 4. The evolved and the reference images (experiments class 2, method 0: self-adaptive F and CR jDE with blending through 1e+8 FES, best setting).

Table III
EXPERIMENTS CLASS 2, METHOD 0 (PROPOSED): FINAL FITNESS.

Image	Baboon (3)	Lena (4)	Plane (5)	Paprica (7)
Fitness	**4.3**	**3.46**	**3.18**	**3.58**

tested images in Figure 3. For the best settings as marked for the class 2, Figure 3 shows the convergence graphs for each method, generally showing that the first two images are harder to optimize than the second two, while the algorithms converge to considerably different levels (note the vertical axes labels).

Finally, in Figure 4 some evolved images are shown at their best runs corresponding for each image, sampled at generations 1, 1000, 2000, 3000, 4000, and 5917 (as NP was set at 100 and number of tiles was 169, generation 5917 resembles FES at 99,997,300, i.e. below limit 1e+8). As can be observed from these evolved images, through a prolonged evolution, the method 0 (proposed one, class 2) obtained evolved images fairly closely resembling the reference images. The experiments like [1], [7] report the RN degree at quite above 5%, even 10%, and we now show that for all images, we have now obtained considerably lower values. The obtained fitness values (i.e. the degree of lost detail, fitness, for which the convergence difficulty increases exponentially), was now namely down to values as seen in Table III (i.e. averagely 3.63%); such significantly better values were not demonstrated yet at any previous experiments with EvoLisa.

V. CONCLUSION

In this paper, the proposed evolvable lossy image representation utilizing an image compared to its evolved generated counterpart image, was presented. The image was represented using variable number of triangular brushstrokes, each consisting of triangle position and color parameters. These parameters for each triangle brush were evolved using DE, which self-adapted the control parameters for mutation and crossover. The proposed DE extension splits the DE vector in codon and anti-codon part, where the triangles material was used only from the codon part, adjusting the genetic tree center and its borders, together with the number of triangle brushstrokes to render. Experimental results have shown the viability of the proposed encoding and evolution convergence for lossy representation of reference images, where fitness was displayed for different input reference images, dependent on the brush style and optimization algorithm method, maximal number of triangles used in image representation and population size, and maximal number of function evaluations allowed. Future work can include a deeper study on how much the images might get compressed using the proposed encoding, or addressing different encoding aspects, evolutionary operators, control-parameters update, Euclidean distance for color comparison, detailed pixels blending impact study, and more case studies on input images with different properties. Further option is also a case study of the utilized DE mechanisms configuration.

ACKNOWLEDGMENT
This work was supported in part by Slovenian Research Agency (ARRS) under program P2-0041.
Workload was also run on SLING.SI infrastructure under European Grid Infrastructure.

REFERENCES

[1] A. Izadi, V. Ciesielski, and M. Berry, "Evolutionary non photo–realistic animations with triangular brushstrokes," in *AI 2010: Advances in Artificial Intelligence*. Springer, 2011, pp. 283–292.

[2] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[3] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[4] R. Alsing, "Genetic programming: Evolution of mona lisa, http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/," 2008.

[5] J. Ding and D. Robb, "Particle swarm optimization," 2009.

[6] I. Poca, "Gplis – an evolutionary image compression tool, http://genetic-image-compression.googlecode.com/svn-history/r93/trunk/Dissertation/l4proj.pdf, 2012.

[7] A. Zamuda and U. Mlakar, "Differential Evolution Control Parameters Study for Self-Adaptive Triangular Brushstrokes," *Informatica - An International Journal of Computing and Informatics*, vol. 39, pp. 105–113, 2015.

[8] D. H. Rogstad, A. Mileant, and T. T. Pham, *Antenna arraying techniques in the deep space network*. John Wiley & Sons, 2005, vol. 6.

[9] Y. Qianli and R. De Gaudenzi, "Space communications and internet," *Communications, China*, vol. 10, no. 10, pp. ix–x, Oct 2013.

[10] H. Hamann, Y. Khaluf, J. Botev, M. D. Soorati, E. Ferrante, O. Kosak, J.-M. Montanier, S. Mostaghim, R. Redpath, J. Timmis, F. Veenstra, M. Wahby, and A. Zamuda, "Hybrid Societies: Challenges and Perspectives in the Design of Collective Behavior in Self-organizing Systems," *Frontiers in Robotics and AI*,vol. 3, 2016. DOI 10.3389/frobt.2016.00014.

[11] A. Zamuda and J. D. Hernández Sosa, "Differential Evolution and Underwater Glider Path Planning Applied to the Short-Term Opportunistic Sampling of Dynamic Mesoscale Ocean Structures," *Applied Soft Computing*, vol. 24, pp. 95–108, November 2014.

[12] A. Zamuda, J. D. H. Sosa, and L. Adler, "Constrained Differential Evolution Optimization for Underwater Glider Path Planning in Sub-mesoscale Eddy Sampling," *Applied Soft Computing*, vol. 42, pp. 93–118, 2016.

[13] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[14] S. Das, S. S. Mullick, and P. Suganthan, "Recent advances in differential evolution – An updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[15] Y. Zhong and L. Zhang, "Remote sensing image subpixel mapping based on adaptive differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 5, pp. 1306–1329, 2012.

[16] M. Ali, C. W. Ahn, and M. Pant, "Multi-level image thresholding by synergetic differential evolution," *Applied Soft Computing*, vol. 17, p. 111, 2014.

[17] S. Sarkar and S. Das, "Multi-level Image Thresholding based on Two-dimensional Histogram and Maximum Tsallis Entropy – A Differential Evolution Approach," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4788–4797, 2013.

[18] U. Mlakar and Potočnik, "Automated facial expression recognition based on histograps of oriented gradient feature vector differences," *Signal, Image and Video Processing*, vol. 9, no. 1, pp. 245–253, 2015.

[19] A. Zamuda, J. Brest, B. Bošković, and V. Žumer, "Differential Evolution for Parameterized Procedural Woody Plant Models Reconstruction," *Applied Soft Computing*, vol. 11, no. 8, pp. 4904–4912, 2011.

[20] A. Zamuda and J. Brest, "Vectorized procedural models for animated trees reconstruction using differential evolution," *Information Sciences*, vol. 278, pp. 1–21, 2014.

[21] F. Neri and V. Tirronen, "Recent Advances in Differential Evolution: A Survey and Experimental Analysis," *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 61–106, 2010.

[22] A. Zamuda and J. Brest, "Self-adaptive control parameters' randomization frequency and propagations in differential evolution," *Swarm and Evolutionary Computation*, vol. 25, pp. 72–99, 2015, (Special Issue on Recent Advances in Modern Nature-Inspired Algorithms, RAMONA).

[23] L. Quan, *Image-Based Modeling*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[24] P. Barile, V. Ciesielski, M. Berry, and K. Trist, "Animated drawings rendered by genetic programming," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 939–946.

[25] J. Riley and V. Ciesielski, "Fitness landscape analysis for evolutionary non-photorealistic rendering," in *2010 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2010, pp. 1–9.

[26] K. Trist, V. Ciesielski, and P. Barile, "Can't see the forest: Using an evolutionary algorithm to produce an animated artwork," in *Arts and Technology*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2010, vol. 30, pp. 255–262.

[27] E. den Heijer and A. E. Eiben, "Evolving pop art using scalable vector graphics," in *Evolutionary and Biologically Inspired Music, Sound, Art and Design*. Springer, 2012, pp. 48–59.

[28] S. Bergen and B. J. Ross, "Automatic and interactive evolution of vector graphics images with genetic algorithms," *The Visual Computer*, vol. 28, no. 1, pp. 35–45, 2012.

[29] C. Coia and B. J. Ross, "Automatic evolution of conceptual building architectures," in *2011 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2011, pp. 1140–1147.

[30] S. Bergen and B. J. Ross, "Aesthetic 3D Model Evolution," in *Evolutionary and Biologically Inspired Music, Sound, Art and Design*. Springer, 2012, pp. 11–22.

[31] P. Pospichal, E. Murphy, M. O'Neill, J. Schwarz, and J. Jaros, "Acceleration of grammatical evolution using graphics processing units: computational intelligence on consumer games and graphics hardware," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*. ACM, 2011, pp. 431–438.

[32] D. Beaumont and S. Stepney, "Grammatical Evolution of L-systems," in *The 2009 IEEE Congress on Evolutionary Computation CEC 2009*. IEEE Press, 2009, pp. 2446–2453.

[33] D. Ashlock, K. M. Bryden, and S. Gent, "Simultaneous evolution of bracketed L-system rules and interpretation," in *IEEE Congress on Evolutionary Computation, 2006. CEC 2006*, 2006, pp. 2050–2057.

[34] A. Zamuda and J. Brest, "Tree Model Reconstruction Innovization Using Multi-objective Differential Evolution," in *2012 IEEE World Congress on Computational Intelligence (IEEE WCCI 2012)*. Brisbane, Australia: IEEE Press, 2012, pp. 2827–2834.

[35] R. I. Mckay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. ONeill, "Grammar-based genetic programming: a survey," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 365–396, 2010.

[36] M. O'Neill and A. Brabazon, "Grammatical differential evolution," in *IC-AI*, 2006, pp. 231–236.

[37] M. O'Neill, J. M. Swafford, J. McDermott, J. Byrne, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg, "Shape grammars and grammatical evolution for evolutionary design," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 1035–1042.

[38] M. O'Neill, J. McDermott, J. M. Swafford, J. Byrne, E. Hemberg, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg, "Evolutionary design using grammatical evolution and shape grammars: designing a shelter," *Int. J. Design Engineering*, vol. 3, no. 1, pp. 4–24, 2010.

[39] A. Moraglio and S. Silva, "Geometric differential evolution on the space of genetic programs," in *Genetic Programming*. Springer, 2010, pp. 171–183.

[40] C. B. Veenhuis, "Tree Based Differential Evolution," in *Genetic Programming*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5481, pp. 208–219.

[41] C. Fonlupt, D. Robilliard, V. Marion-Poty, and S. Ventura, "Continuous schemes for program evolution," *Genetic Programming-New Approaches and Successful*, pp. 27–48, 2012.

[42] M. F. Brameier and W. Banzhaf, *Linear genetic programming*. Springer, 2007.

[43] B. D. Ackland and N. H. Weste, "The edge flag algorithm — a fill method for raster scan displays," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 41–48, 1981.

[44] J. Brest and M. S. Maučec, "Population Size Reduction for the Differential Evolution Algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.