DIFFERENTIAL EVOLUTION FOR SELF-ADAPTIVE TRIANGULAR BRUSHSTROKES

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

Abstract This paper proposes a lossy image representation where a reference image is approximated by an evolved image, constituted of variable number of triangular brushstrokes. The parameters of each triangle brush are evolved using differential evolution, which self-adapts the triangles to the reference image, and also self-adapts some of the control parameters of the optimization algorithm, including the number of triangles. Experimental results show the viability of the proposed encoding and optimization results on a few sample reference images.

Keywords: Differential Evolution, Evolutionary Computer Vision, Evolutionary Art, Image-based Modeling, Self-adaptation, Triangular Brushstrokes

1. Introduction

In this paper, evolvable lossy image representation utilizing an image compared to its evolved generated counterpart image, is proposed. The image is represented using a variable number of triangular brushstrokes [5], each consisting of triangle position and color parameters. These parameters for each triangle brush are evolved using differential evolution [10, 3], which self-adapts the control parameters, including the proposed self-adaptation for the number of triangles to be used. Experimental results show the viability of the proposed encoding and evolution convergence for lossy compression of sample images.

The approach presented is built upon and compared with [5], by addressing and also extending the original challenge. Namely, the challenge introduced in [5] uses triangles in trying to build an approximate model of an image [5]. The triangle is an efficient brush shape for this challenge, since it covers more pixels than a single point, and also allows overlaying and blending of colors over several regional surface pixels, which lines

can not. Also, an arbitrary triangle shape is less constrained than any further point-approximated shape, and also other shapes can be built by combining several triangles.

Instead of genetic programming in [5], in this paper differential evolution is used with a fixed size tree-like chromosome vector, which is cut-off self-adaptively to form codon and anti-codon parts of the chromosome. Also, our approach uses a modified challenge, where we can reconstruct the model for the reference image solely using the evolved model without using the reference image, whereas the [5] needs the reference image when drawing pixels to the canvas in deciding which pixels match the reference image for accepting them into the evolved canvas. Also, in this paper the triangle brushstroke encoding differes and is proposed especially designed for an efficient DE encoding.

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

2. Related Work

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

2.1 Image-based modeling, Evolutionary Computer Vision, and Evolutionary Art

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 [7]. For art drawings modeling, automatic evolutionary rendering has been applied [2, 9]. In [11] animated artwork is evolved using an evolutionary algorithm. Then, Izadi et al. [5] evolved triangular brushstrokes challenge using genetic programming for two-dimensional images, using unguided and guided searches on a three or four branch genetic program, where roughly 5% similarity with reference images was obtained on average per pixel. In this paper, we build upon and compare our new approach with [5], by addressing and also extending its challenge. After extending the challenge, we optimize it using DE, which is described in the next section.

2.2 Evolutionary Optimization using Differential Evolution

Differential evolution (DE) [10] is a floating-point encoding evolutionary algorithm for continuous global optimization. It has been modified and extended several times with various versions being proposed [4]. DE has also been applied to remote sensing image subpixel mapping [14], image thresholding [8], and for image-based modeling using evolutionary computer vision to reconstruct a spatial procedural tree model from a limited set of two dimensional images [13, 12]. Neri and Tirronen in their survey on DE [6] concluded that, compared to the other algorithms, a DE extension called jDE [3], is superior to the compared algorithms in terms of robustness and versatility over a diverse benchmark set used in the survey. Therefore, we choose to apply jDE in this approach.

The original DE has a main evolutionary loop where a population of vectors is computed within each generation. For one generation, counted as g, each vector $\mathbf{x}_i$, $\forall i \in \{1, ..., NP\}$ 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 the next generation. For each corresponding population vector, mutation creates a mutant vector $\mathbf{v}_{i,g+1}$ (*'rand/1'* [10]):

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

where the indexes r_1, r_2, and r_3 are random and mutually different integers generated in from set $\{1, ..., NP\}$, 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 the 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 $u_{i,j,g+1}$ 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, $\forall j \in \{1, ..., D\}$ 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. Since the jDE self-adapts the F and CR control parameters to generate the vectors $\mathbf{v}_{i,g+1}$ and $\mathbf{u}_{i,g+1}$, corresponding

values F_i and CR_i, $\forall i \in \{1, ..., NP\}$ are updated prior to their use in the mutation and crossover mechanisms:

$$F_{i,g+1} = \begin{cases} F_{\mathrm{l}} + rand_1 \times F_{\mathrm{u}} & \text{if } rand_2 < \tau_1, \\ F_{i,g} & \text{otherwise,} \end{cases} \tag{3}$$

$$CR_{i,g+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2, \\ CR_{i,g} & \text{otherwise,} \end{cases} \tag{4}$$

where $\{rand_1, ..., rand_4\} \in [0, 1]$ are uniform random floating-point numbers and $\tau_1 = \tau_2 = 0.1$. Finally, the selection operator evaluates and compares the trial to current vector and propagates the fittest:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{if } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{5}$$

3. Differential Evolution for Self-adaptive Triangular Brushstrokes

In this section, the encoding aspect, genotype-phenotpye rendering, and evaluation mechanisms of the proposed approach are defined.

3.1 Encoding Aspect

We encode an individual compressed image into a DE vector as follows. A DE vector $\mathbf{x} = (x_1, x_2, ..., x_{8T^{\max}}, F, CR, T^{\mathrm{L}}, T^{\mathrm{U}})$ 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 $D = 8T^{\max}$, and the rest are the self-adaptive control parameters of the vector to be used during the DE. 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^{L}, $T_i^{\mathrm{U}} \in \{1, .., T^{\max}\}$ control parameters follow.

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 $\{x_j : \forall j \in \{T^{\mathrm{L}}, ..., T^{\mathrm{U}}\}\}$.

In this paper, we propose to have the whole vector represent a triangle set, organized similar to serializing a tree as a linear vector in visiting nodes by depth-first search. However, the leaf nodes are mostly 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 in being retained, since they are more anchored due to cut-offs mostly around the codon edges. After being included into a new trial vector, all nodes have an equal probability of having their triangle data changed.

In this way, the T^{L} and T^{U} allow us to render only a sub-portion of the triangles set, similarly to taking an inseparable portion of a GP tree traversal as in [5]. This gives us an arbitrary length render set, and keeps the crossover of anti-codon to help us find the number of triangles $T_i \in \{1, ..., T^{\max}\}$, which is more suitable for image approximation:

$$T_i = \begin{cases} T_i^{\mathrm{U}} - T_i^{\mathrm{L}} + 1 & \text{if } T_i^{\mathrm{L}} < T_i^{\mathrm{U}} \\ (T^{\max} - T_i^{\mathrm{L}}) + T_i^{\mathrm{U}} & \text{otherwise.} \end{cases} \tag{6}$$

The T_i^{L} and T_i^{U} are updated similarly to the F_i control parameter:

$$T_{i,g+1}^{\mathrm{L}} = \begin{cases} \lfloor rand_1^{\mathrm{L}} \times T^{\max} \rfloor & \text{if } rand_2^{\mathrm{L}} < \tau^{\mathrm{L}}, \\ T_{i,g}^{\mathrm{L}} & \text{otherwise,} \end{cases} \tag{7}$$

$$T_{i,g+1}^{\mathrm{U}} = \begin{cases} \lfloor rand_1^{\mathrm{U}} \times T^{\max} \rfloor & \text{if } rand_2^{\mathrm{U}} < \tau^{\mathrm{U}}, \\ T_{i,g}^{\mathrm{U}} & \text{otherwise,} \end{cases} \tag{8}$$

where $\tau^{\mathrm{L}} = \tau^{\mathrm{U}} = \tau_1 = 0.1$ of the jDE.

3.2 Genotype-Phenotype Rendering

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

The triangle brushstrokes (Figure 1) are represented as $(c_x, c_y, r, \alpha_1, \alpha_2, b^{\mathrm{Y}}, b^{\mathrm{Cb}}, b^{\mathrm{Cr}})$, where $c_x \in [0, ..., R_{\mathrm{x}})$, $c_y \in [0, ..., R_{\mathrm{y}})$, and $r \in [0, R_{\mathrm{x}}/\sqrt{T_{\max}}]$ define the circumscribed circle center and radius for the triangle to be rendered; $\alpha_1 \in [1°, 360°)$ and $\alpha_2 \in [1°, 180°)$ define the points of this triangle on its circumscribed circle; and $b^{\mathrm{Y}} \in [16, 236)$, $b^{\mathrm{Cb}} \in [16, 241)$, and $b^{\mathrm{Cr}} \in [16, 241)$ are the color components of the brush for the triangle contained pixels.

The triangles' vertices encoded by i-th DE vector construct T_i triangles, each triangle $\mathbf{T}_k = (c_{x,k}, c_{y,k}, r_k, \alpha_{1,k}, \alpha_{2,k}), \forall k \in \{1, ..., T_i\}$ ($\mathbf{T}_k$ being packed as $\mathbf{x}_i = \{x_{i,j}\}$, $j = 8k + m$, $m \in \{1, ..., 8\}$), defining the vertices of a triangle $P_{1,k}$, $P_{2,k}$, and $P_{3,k}$:

$$P_{1,k} = \lfloor (c_{x,k} + r_k \cos \alpha_{1,k}, c_{y,k} + r_k \sin \alpha_{1,k}) \rfloor , \tag{9}$$

$$P_{2,k} = \lfloor (c_{x,k} + r_k \cos(\alpha_{1,k} + \pi), c_{y,k} + r_k \sin(\alpha_{1,k} + \pi)) \rfloor , \tag{10}$$

$$P_{3,k} = \lfloor (c_{x,k} + r_k \cos \alpha_{2,k}, c_{y,k} + r_k \sin \alpha_{2,k}) \rfloor . \tag{11}$$

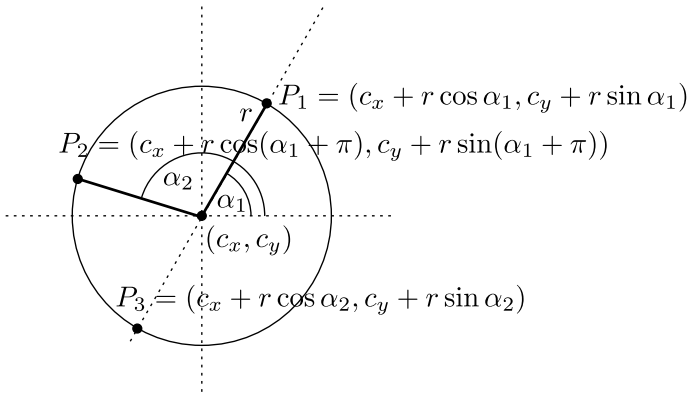

Figure 1. The triangle brush definition and the circumscribed circle.

The brush color $\mathbf{b}_k^{\mathrm{YCbCr}} = (b_k^{\mathrm{Y}}, b_k^{\mathrm{Cb}}, b_k^{\mathrm{Cr}})$ is first transformed into RGB color model as $\mathbf{b}_k^{\mathrm{RGB}} = (b_k^{\mathrm{R}}, b_k^{\mathrm{G}}, b_k^{\mathrm{B}})$ $(b_k^{\mathrm{R}}, b_k^{\mathrm{G}}, b_k^{\mathrm{B}} \in [0, 255])$, where:

$$b_k^{\mathrm{R}} = \left\lfloor 1.164(b_k^{\mathrm{Y}} - 16) + 1.596(b_k^{\mathrm{Cr}} - 128) \right\rfloor \tag{12}$$

$$b_k^{\mathrm{G}} = \left\lfloor 1.164(b_k^{\mathrm{Y}} - 16) - 0.813(b_k^{\mathrm{Cr}} - 128) - 0.391(b_k^{\mathrm{Cb}} - 128) \right\rfloor \tag{13}$$

$$b_k^{\mathrm{B}} = \left\lfloor 1.164(b_k^{\mathrm{Y}} - 16) + 2.018(b_k^{\mathrm{Cb}} - 128) \right\rfloor \tag{14}$$

For each triangle T_k, a solid color is rendered without antialiasing over the triangle brush area rasterizing [1] with a transparency factor of $1/T_i$:

$$\mathbf{b}_k = \left\lfloor \frac{255}{T_i} \mathbf{b}_k^{\mathrm{RGB}} \right\rfloor . \tag{15}$$

This is analogous to blending the triangle as a part-transparent layer within the evolved image $\boldsymbol{Z}_i = \sum_k \mathbf{z}_{k,x,y}$ and computes R, G, and B color layers for the pixels of the i-th individual:

$$\mathbf{z}_{k,x,y} = \sum_{\mathbf{T}_k \text{ over } (x,y)} \mathbf{b}_{k,x,y} = \sum_{\mathbf{T}_k \text{ over } (x,y)} \left\lfloor \frac{255}{T_i} \mathbf{b}_{k,x,y}^{\mathrm{RGB}} \right\rfloor , \tag{16}$$

where $\mathbf{T}_k$ over (x, y) denotes each triangle being rendered over the pixel (x, y) such that $\mathbf{b}_{k,x,y}$ contains the rendered pixels of a brushstroke. Triangles defined possibly over the edges of image canvas are drawn by clipping away pixels outside of the canvas area.

The initialization of a genotype is such that the $c_x, c_y, \alpha_1, \alpha_2, b^{\mathrm{Y}}, b^{\mathrm{Cb}}$, b^{Cr}, T_i^{L}, and T_i^{U} are initialized uniform randomly to integer values within their respective definition intervals, while r is kept as a floating-point. All parameters are however evolved as floating-point scalar values in DE.

3.3 Evaluation

Evaluation of the phenotype image $\boldsymbol{Z}_i$ to be compared against a reference image $\boldsymbol{Z}^*$ is as follows. A reference image $\boldsymbol{Z}^*$ is represented as RGB-encoded colored pixels integer values in layers $\boldsymbol{Z}^* = \{(z^{\mathrm{R}}_{x,y}, z^{\mathrm{G}}_{x,y}, z^{\mathrm{B}}_{x,y})\}$.

To obtain a difference assessment value, the following comparison metric is used for comparing an evolved image $\boldsymbol{Z} = \boldsymbol{Z}_i$ to $\boldsymbol{Z}^*$:

$$f(\boldsymbol{Z}) = 100 \times \frac{\sum_{y=0}^{R_{\mathrm{y}}-1} \sum_{x=0}^{R_{\mathrm{x}}-1} \mid z^{*\mathrm{R}}_{x,y} - z^{\mathrm{R}}_{x,y} \mid + \mid z^{*\mathrm{G}}_{x,y} - z^{\mathrm{G}}_{x,y} \mid + \mid z^{*\mathrm{B}}_{x,y} - z^{\mathrm{B}}_{x,y} \mid}{3 \times 255 \times R_{\mathrm{x}} R_{\mathrm{y}}}. \tag{17}$$

4. Experiments

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 $NP = \{25, 50, 100\}$ and $T_{\max} = \{10, 20, ..., 150\}$, thereby for each run RNi={0,1,...,51} this counts for total of 45 parameter sets, i.e. 2340 independent runs. The maximum number of function evaluations (MAXFES) used is same as with [5], MAXFES is 1e+5. For image rendering, basic GDI+ is used.

4.1 Obtained Results

The obtained fitness values at the MAXFES termination of 1e+5, over different parameters of $T_{\max}$ and NP, are seen in Tables 1 and 2. The best values obtained overall for an image are marked in bold text font. The fitness convergence graphs for these best runs are seen in Figure 2, where after the initialization, the fitness is roughly below 40 (i.e. 40% similarity with reference), then drops below 15 for all test images and even further to slightly above 6 for two of them.

The convergent obtained results depend on the MAXFES used being same as with [5], but also NP and $T_{\max}$, as reported below. From Tables 1 and 2, we choose to report further evolved images upto MAXFES of 1e+6 with all images. The best approximated images after MAXFES of 1e+6 are shown in the Figure 3 which shows the evolution of the four images. In each line of Figure 3, the best fitting vectors upto MAXFES of 1e+6 in generations $g = \{0, 100, 200, 400, 700, 1200, 2000\}$, and the final generation, are shown, then the rightmost the corresponding reference image. Figure 4 shows for each test image, dynamics of the number of triangle brushes in current best vector during generations, displaying varying convergent best T_i values across images.

Table 1. Obtained fitness over $T_{\max}$ and NP: test instances `Liberty` and `Palace`.

		`Liberty`				`Palace`			
NP	$T_{\max}$	Best	Worst	Average	STD	Best	Worst	Average	STD
25	10	8.29	11.99	9.93096	0.8233	8.69	13.69	10.1362	0.9655
25	20	8.03	13.14	10.0935	1.0845	7.83	11.5	9.12173	0.8092
25	30	8.41	13.74	10.0525	1.1712	7.52	11.1	8.97942	0.7992
25	40	8.13	12.81	10.4408	1.1416	7.34	11.36	8.91788	0.8922
25	50	8.49	13.37	10.6767	1.1768	7.65	12.53	8.87442	0.9788
25	60	7.95	14.65	10.9858	1.4284	7.9	11.88	8.99673	0.8761
25	70	8.28	14.21	11.4075	1.3630	7.79	13.17	9.50327	1.0482
25	80	8.72	15.89	11.7554	1.6330	7.97	12.34	9.43558	0.9765
25	90	8.84	16.24	12.1342	1.6608	8.41	13.54	9.82	1.2756
25	100	9.01	16.74	12.4798	1.7521	8.62	12.96	9.83635	0.8869
25	110	8.07	16.78	12.7412	1.7849	9.01	14.42	10.4119	1.2468
25	120	9.67	16.14	12.8467	1.7359	8.93	15.13	10.3858	1.3149
25	130	10.16	17.96	13.2692	1.7193	9.02	14.2	10.2858	1.0292
25	140	9.29	17.99	13.7029	1.7886	8.29	13.51	10.7779	1.0299
25	150	10.82	18.56	14.0373	1.6573	9.89	14.91	11.1206	1.0586
50	10	7.51	9.69	8.45077	0.4198	7.43	11.84	8.68058	0.8825
50	20	6.78	8.99	7.80173	0.4987	**7.1**	11.39	8.79173	0.9592
50	30	6.89	9.17	7.81788	0.5119	7.53	12.58	9.75654	1.1186
50	40	**6.77**	9.87	8.0375	0.6578	8.27	12.24	10.0575	0.9537
50	50	7.08	10.61	8.39923	0.7056	7.97	13.14	10.3338	1.1009
50	60	7.15	10.4	8.67115	0.7472	8.59	12.49	10.7817	1.0754
50	70	7.46	10.9	9.1025	0.8666	7.58	12.8	10.7744	1.1086
50	80	7.6	11.4	9.47981	0.8689	9.15	13.11	11.3802	1.0178
50	90	8.05	12.65	9.67346	0.9115	9.97	13.41	11.5227	0.9315
50	100	8.75	11.75	10.0152	0.7824	8.55	13.62	11.4356	0.9923
50	110	8.93	13.63	10.6356	0.9682	9.32	13.77	12.0712	0.9579
50	120	9.22	13.01	10.7502	0.9840	9.77	14.21	12.429	0.8972
50	130	9.42	12.59	11.0527	0.7707	11.37	14.07	12.7387	0.6134
50	140	9.99	13.39	11.5719	0.7815	9.69	15.5	12.9317	0.9708
50	150	10.2	14.56	12.2633	1.0702	9.58	15.36	12.8092	1.1717
100	10	7.1	9.12	7.98596	0.4241	7.91	13.88	10.9573	1.8019
100	20	6.85	9.77	7.83962	0.5360	8.86	14.59	12.1117	1.2862
100	30	7.15	11.8	8.49077	1.1563	9.59	16.15	12.9098	1.0589
100	40	7.22	13	8.86327	1.1092	9.65	14.97	13.2477	1.1543
100	50	7.41	12.75	9.34846	1.3939	11.01	15.52	13.8606	0.9750
100	60	8.06	12.97	9.77731	1.1539	11.5	16.14	14.1856	1.1234
100	70	8.67	13.28	10.1954	1.3722	10.77	16.32	14.3629	1.1713
100	80	8.73	14.48	11.0929	1.4093	10.98	17.06	14.9348	1.1679
100	90	9.04	14.92	11.3594	1.3483	11.1	16.8	15.104	1.2586
100	100	9.4	16.13	11.6604	1.4952	10.8	17.62	15.36	1.2330
100	110	10.17	15.68	12.3365	1.5685	13.01	17.86	16.0202	0.9744
100	120	10.26	15.45	12.3358	1.5076	11.07	17.99	15.6113	1.6455
100	130	10.22	16.19	13.2212	1.6108	12.33	18.37	16.4085	1.3168
100	140	11.42	16.65	13.7808	1.5502	11.64	18.35	16.1229	1.4990
100	150	11.35	18.68	14.6113	1.9726	10.11	18.34	16.2929	2.0056

Table 2. Obtained fitness over $T_{\max}$ and NP: test instances `Vegetables` and `Baboon`.

		`Vegetables`				`Baboon`			
NP	$T_{\max}$	Best	Worst	Average	STD	Best	Worst	Average	STD
25	10	14.13	17.21	15.7269	0.7148	15.02	18.59	16.38	0.7128
25	20	12.56	18.03	14.5658	0.9850	13.44	17.12	15.3815	0.8129
25	30	12.33	15.98	13.9215	0.8475	12.99	19.03	15.0204	1.1150
25	40	11.62	16.21	13.674	1.0436	11.99	16.85	14.4342	1.0135
25	50	12.16	17.08	13.88	1.0726	11.39	17.62	14.4573	1.2299
25	60	11.64	17.88	13.6438	1.2155	11.74	17.51	14.8038	1.2229
25	70	11.29	17.15	13.9056	1.3790	11.88	17.9	14.6267	1.3495
25	80	11.61	16.6	14.0871	1.3881	12.11	17.13	14.3606	1.2815
25	90	11.63	17.96	14.1062	1.4428	11.93	19.41	14.6644	1.5269
25	100	11.34	17	14.4533	1.4694	11.7	18.77	14.7642	1.7438
25	110	11.74	19.66	14.6085	1.7664	12.02	19.11	15.0046	1.7605
25	120	12.26	17.91	14.7737	1.5726	12.2	18.5	15.6467	1.6086
25	130	12.1	19.75	14.6338	1.9283	13.01	19.5	15.4254	1.5505
25	140	11.94	19.01	14.7635	1.6282	12.64	19.37	15.8235	1.8458
25	150	12.82	18.7	14.6487	1.3015	13.13	20.17	15.7952	1.6923
50	10	13.03	15	14.0723	0.4674	13.86	16.52	14.9192	0.5494
50	20	11.66	13.26	12.4644	0.3184	11.8	14.54	13.271	0.5569
50	30	11.12	13.59	12.2425	0.6528	11.59	13.62	12.5506	0.5732
50	40	**10.94**	14.1	12.1848	0.6656	11.1	13.84	12.3137	0.6090
50	50	11.04	13.92	12.2946	0.7609	11.34	14.36	12.4075	0.6304
50	60	11.29	15.86	12.5506	0.9222	11.25	14.1	12.3662	0.6161
50	70	11.18	15.21	12.6104	0.8682	11.54	14.57	12.5437	0.6510
50	80	11.32	15.26	12.8619	0.7658	**11.07**	15.56	12.9473	0.8087
50	90	11.84	15.28	13.0077	0.8038	11.32	16.2	12.857	1.0291
50	100	11.72	15.8	13.5058	0.9565	11.85	15.72	13.2658	0.7972
50	110	12.02	15.92	13.5204	0.8750	11.98	15.56	13.4275	0.7805
50	120	11.9	16.87	13.829	1.1151	12.43	15.66	13.5106	0.7265
50	130	12.51	15.97	14.094	0.8855	12.64	16.32	14.085	0.8259
50	140	12.16	17.07	14.8198	1.2154	12.54	16.31	14.15	0.8865
50	150	13.11	17.98	14.9838	1.2072	13.08	18	14.8765	1.0178
100	10	12.56	16.19	13.9815	0.8083	13.49	16.19	14.5367	0.5672
100	20	11.84	16.45	13.4704	1.0483	12.02	15.87	13.8244	0.8747
100	30	11.83	17.64	13.9133	1.3335	12	15.76	13.7206	0.9727
100	40	12.01	17.95	14.6354	1.3660	11.63	17.01	13.6467	1.3582
100	50	11.87	17.35	14.9156	1.4272	11.99	17.48	14.1658	1.5554
100	60	12.32	18	15.21	1.5119	12.12	17.46	14.5021	1.4517
100	70	12.13	18.05	15.6513	1.2457	12.12	17.16	14.3881	1.3782
100	80	12.9	18.86	16.2008	1.4121	12.13	17.56	14.8656	1.4214
100	90	12.32	20.04	16.3233	1.7789	12.25	18.66	15.2558	1.5144
100	100	12.98	20.55	16.7275	1.7119	13.09	18.42	15.5398	1.5064
100	110	13.76	20.18	17.2896	1.5242	13	19.62	15.84	1.6164
100	120	13.12	20.62	17.626	1.5807	13.34	19.58	16.4725	1.5223
100	130	13.52	20.12	17.9052	1.3516	13.84	19.6	16.9367	1.7362
100	140	14.08	20.52	18.216	1.6975	14.3	21	17.4387	1.7372
100	150	14.97	21.19	19.1221	1.2128	14.75	21.13	17.9488	1.6872

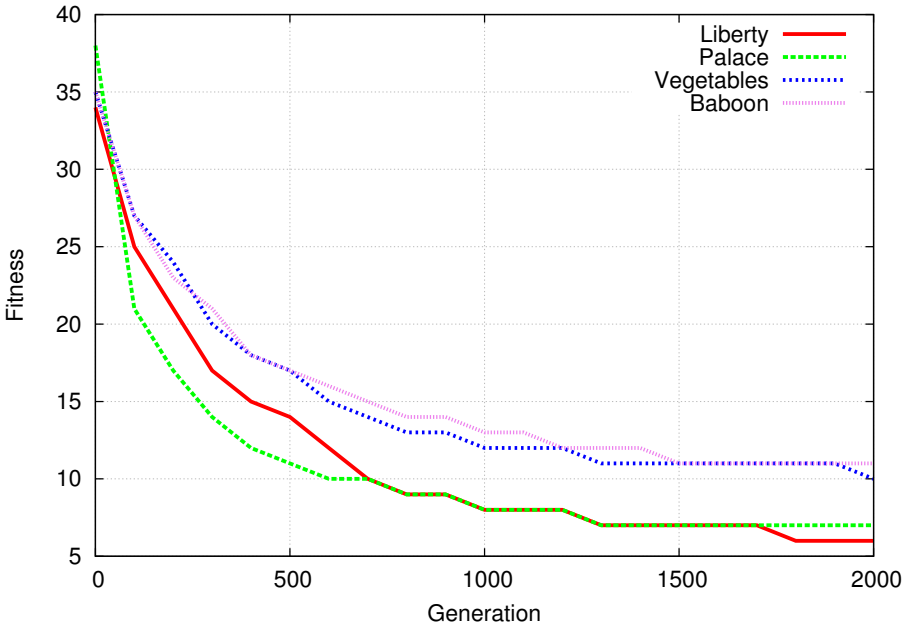

Figure 2. Fitness convergence during optimization, for best runs of each test image.

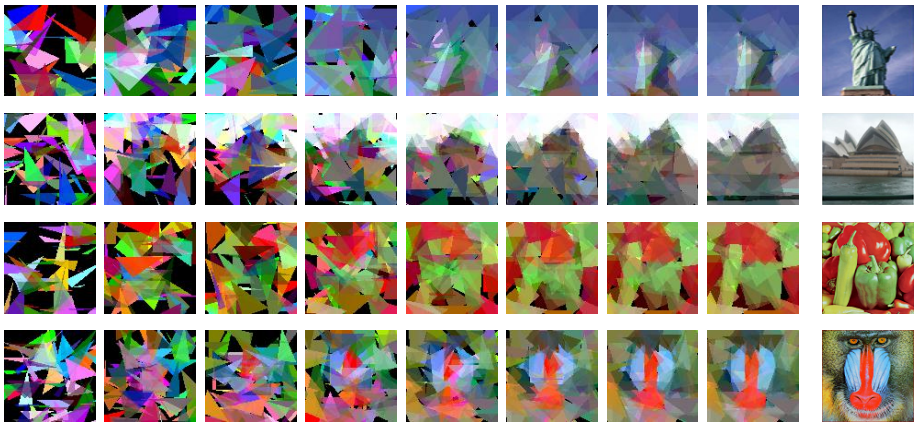

Figure 3. The evolved and the reference images.

Our approach searches for a representative image model and the values obtained such as 6.77, can roughly be compared to the 4.83 of [5]. Such representation of the problem also makes our *NP* parameter have higher value, since we have no guided search and the problem is therefore more general. Also, our approach does not use a dynamically re-allocatable morphable variable-size tree structure as in genetic programming encod-

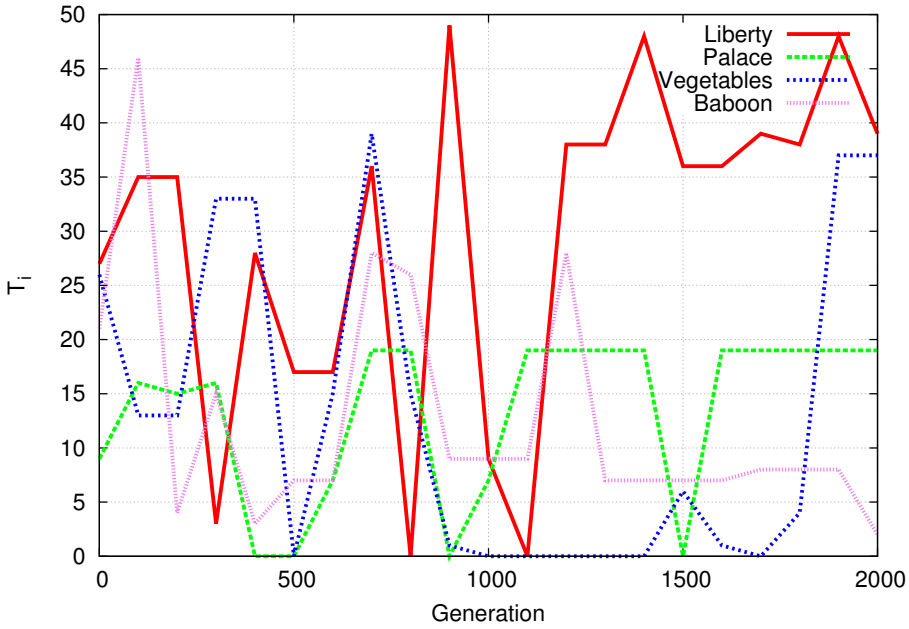

Figure 4. Number of best vector brushstrokes, for best runs of each test image.

ing, inspite it rather uses a fixed size vector and limits its brushstrokes set by two simple bounds, making the approach faster for execution.

5. Conclusion

This paper presents an evolvable lossy image representation, approximating an image by comparing it to its evolved generated counterpart image. The image is represented using a variable number of triangular brushstrokes, each consisting of a triangle position and color parameters. These parameters for each triangle brush are evolved using differential evolution, which self-adapts the control parameters for mutation and crossover. Also, the proposed DE extension splits the DE vector in the codon and anticodon parts, where the triangles material is used only from the codon part, adjusting the genetic tree center and its borders, together with the number of triangle brushstrokes to be rendered. Experimental results show the viability of the proposed encoding and evolution convergence for the lossy representation of reference images, where fitness is displayed dependent on the population size, maximal number of function evaluations allowed, maximal number of triangles used in image representation, and different input reference images. Future work can include addressing different encoding aspects, evolutionary operators,

control-parameters update, Euclidean distance for colors comparison, and more case studies on input images with different properties.

References

[1] Ackland, B. D. and Weste, Neil H. The edge flag algorithm — a fill method for raster scan displays. *IEEE Transactions on Computers*, 100, 1 (1981) 41–48.

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

[3] Brest, J., Greiner, S., Bošković, B., Mernik, M., and Žumer, V. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, 10, 6 (2006) 646–657.

[4] Das, S. and Suganthan, P. N. Differential Evolution: A Survey of the State-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15, 1 (2011) 4–31.

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

[6] Neri, F. and Tirronen, V. Recent Advances in Differential Evolution: A Survey and Experimental Analysis. *Artificial Intelligence Review*, 33, 1–2 (2010) 61–106.

[7] Quan, L. *Image-Based Modeling*. Springer Publishing Company, Incorporated, 1st edition (2010).

[8] Rahnamayan, S. and Tizhoosh, H.R. Image thresholding using micro opposition-based Differential Evolution (Micro-ODE). In *IEEE World Congress on Computational Intelligence* (2008) 1409–1416.

[9] Riley, Jeff and Ciesielski, Victor. Fitness landscape analysis for evolutionary non-photorealistic rendering. In *2010 IEEE Congress on Evolutionary Computation (CEC)*, IEEE (2010) 1–9.

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

[11] Trist, K., Ciesielski, V., and Barile, P. Can't see the forest: Using an evolutionary algorithm to produce an animated artwork. In *Arts and Technology*, Springer Berlin Heidelberg (2010) 255–262.

[12] Zamuda, A. and Brest, J. Vectorized procedural models for animated trees reconstruction using differential evolution. *Information Sciences*, 278 (2014) 1–21.

[13] Zamuda, A., Brest, J., Bošković, B., and Žumer, V. Differential Evolution for Parameterized Procedural Woody Plant Models Reconstruction. *Applied Soft Computing*, 11, 8 (2011) 4904–4912.

[14] Zhong, Y. and Zhang, L. Remote sensing image subpixel mapping based on adaptive differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42, 5 (2012) 1306–1329.