Escher-like Tiling Design from Video Images Using Convolutional Variational Autoencoder *

Asuka Hisatomi[1], Tomofumi Matsuyama[1], Takahiro Kinoshita[1], Kazunori Mizuno[2], Satoshi Ono[1]

[1]*Department of Information Science and Biomedical Engineering, Graduate School of Science and Engineering, Kagoshima University 1-21-40, Korimoto, Kagoshima, 890-0065 Japan E-mail: {mc116029;sc114063;sc115015;ono;}@ibe.kagoshima-u.ac.jp*

[2]*Department of Computer Science, Faculty of Engineering, Takushoku University 815-1 Tatemachi, Hachioji-shi, Tokyo 193-0985 Japan E-mail: mizuno@cs.takushoku-u.ac.jp*

This paper proposes a method that deforms a prominent movie or animation character into a tileable shape. Tiling is the act of covering the plane with one or a very few types of figures without overlaps and/or gaps. Although some previous methods can transform a given shape into a tileable shape, they cannot easily move the character into a suitably tileable pose. The proposed method learns the latent feature space that abstracts the target character's silhouettes using a convolutional variational autoencoder, and looks for the poses suitable for tiling by optimization in the latent space. Experimental results showed that the proposed method successfully generated tileable figures of the tested character in various poses, some of which were not included in the training dataset.

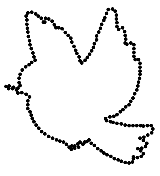
Keywords: Tiling, tessellations, hierarchical optimization, genetic algorithm

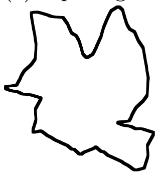
1. INTRODUCTION

Tiling, or tessellation, smoothly covers a plane with one or a very few types of geometric figures [1], as shown in Fig. 1. A figure is called a *tileable figure* (or *tile*) if its copies — duplicated only by translation, rotation, and reflection — can cover the entire plane with no gaps and/or overlaps. In many artworks, archetypically represented by M. C. Escher's works, titles are formed from complicated concavo-convex and irregular shapes.

Various artificial objects, such as textured walls, ceilings, and the floor surfaces of buildings, are tiled to improve their decorativeness. Moreover, tiles can cover areas of various sizes with a repeating pattern. Besides graphics and texture design, tiling is applied in modern designs of physical objects such as cushions and coasters. Fig. 2(a) shows some cushioning materials composed of sponges. An object with a meaningful shape is more likely to be retained and reused than an object of simple shape like a cube. Fewer discarded items in landfills reduce the environmental impact of modern living. As another example, the wooden coasters shown in Fig. 2(b) can be connected into a larger object, which can be used as a pot stand. Fig. 2(c) shows an application of tiling to confectioneries such as chocolates, cookies, snacks, and jellies.

Received March 11, 2014; revised June 20, 2014; accepted September 28, 2014.
Communicated by the editor.


(a) Input figure


(b) Tileable figure

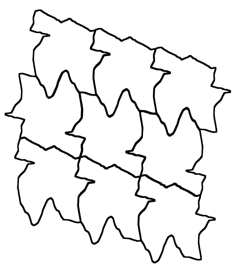

(c) Tiled figures

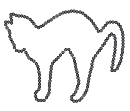
Fig. 1. Tileable shape design.

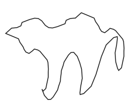
(a) Cushioning materials[1].

(b) Coaster and pot stand.

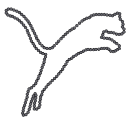
(c) Cookies [2]

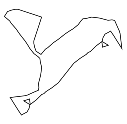
Fig. 2. Tiling application examples.


1) Input figure


2) Tileable figure

(a) Example 1 (good).


1) Input figure


2) Tileable figure

(b) Example 2 (bad).

Fig. 3. Example of the influence from input figure.

A major application of tiling is the conversion of prominent characters in movies and animations into tiles. These tiles are available for interior decorations such as wallpapers and floorings of amusement parks and their associated official hotels. In addition, cookies shaped as major characters are popularly sold at amusement parks. Tileable shapes bring added value to confectioneries, as they can be constructed like puzzles and can be packaged without breaking their shape. As mentioned above, technologies that create tileable shapes of given characters can increase the value and attractiveness of consumer products.

However, transforming a character with complex contours into a tileable shape is a difficult task. Kaplan formulated the problem of designing a tileable shape [3] as an Escherization problem. The Escherization problem has been solved by various methods, which are broadly categorized into analytical optimization approaches [4,5] and meta-heuristics-based approaches [6,7]. These methods aim to obtain a tile similar to the given shape, but shapes with imbalanced convex and concave parts are not easily transformed into a tileable shape. This difficulty stems not from the methods but from the problem itself; for example, the tileable transformation of a circle is a hexagon, which does not resemble the original shape. Therefore, when translating a character into a tileable shape, some poses defy transformation into a tileable figure by any method, as shown in Fig. 3. Finding the poses of a target character suitable for tiling then becomes vitally important.

This paper attempts to generate tile-transformable character images from video recordings. The proposed method combines a Convolutional Variational

[1]http://www.sanodesign.jp/cushion/Cushion_san.html

Auto-encoder (Conv-VAE) [8] with optimization in the latent space obtained by the Conv-VAE. After learning low-dimensional feature space of the target character's pose from the target character image set by Conv-VAE, the method can generate character images that are not included in the given image set. The proposed method first trains Conv-VAE on images extracted from videos, and then optimizes the latent variables to generate pose images suitable for tiling. Experimental results showed that the proposed method generated posed images that were absent from the training set, identified images suitable for tiling, and generated tileable figures of the target character.

2. RELATED WORK

2.1 Escher-like tiling problem

This study focuses on isohedral tilings, in which all tiles are congruent, i.e., a figure covers the entire two-dimensional (2D) space (or plane). The tileable-shape design problem is formulated as the following *Escherization problem* [3]:

Problems ("ESCHERIZATION"): Given a closed plane figure $\boldsymbol{W}$ (the 'goal shape'), find a new closed figure $\boldsymbol{U}$ such that:

1. $\boldsymbol{U}$ is as close as possible to $\boldsymbol{W}$, and
2. the copies of $\boldsymbol{U}$ fit together to tile the plane.

Only three types of regular polygons can fill a plane: regular triangles, squares, and regular hexagons. Moreover, there are a finite number of patterns for copying the tile to cover the plane, if the operation for copying the tile is limited to three types of transformations: translation, rotation and reflection. Such patterns are called *tiling patterns* (TPs), and isohedral tiling admits 93 TPs [9]. Each TP encodes information on the constraint of a tile shape by its adjacencies.

Fig. 4 shows examples of TPs. Pattern IH01, shown in Fig. 4(a), is composed of hexagons with the three pairs of parallel sides that have the same shape. Note that each hexagon side is not necessarily a straight line and can be deformed under the constraints, i.e., IH01 contains three deformable sides and the remaining are copies of them. Pattern IH16, shown in Fig. 4(b), is composed of hexagons with four equal-shape adjacent sides, two equal-shape adjacent sides, and three of the six angles equal to 120 degree.

2.2 Escher-like tiling methods

Various methods have been proposed to deform a given image into a tileable shape, which can be categorized into analytical optimization approaches and meta-heuristic-based approaches.

The analytical optimization approach was originated by Koizumi and Sugihara [4,5]. They formulated the Escherization problem as an eigenvalue problem to be solved by analytical optimization. This method calculates the vertex positions, and constructs a tileable shape by minimizing the sum of square distances between the corresponding vertices of an input polygon and a generated tile. The solution is guaranteed to satisfy the *edge-matching constraint*, which stipulates that all corresponding edges shared between two adjacent tiles have the same shape. However, this method is sensitive to the given vertices of the input figure. Minute differences

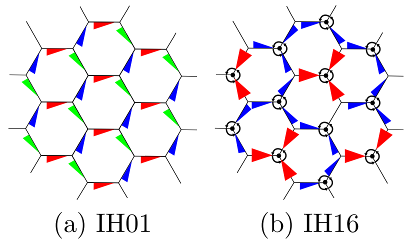


Fig. 4. Example tiling pattern: IH01 and IH16.

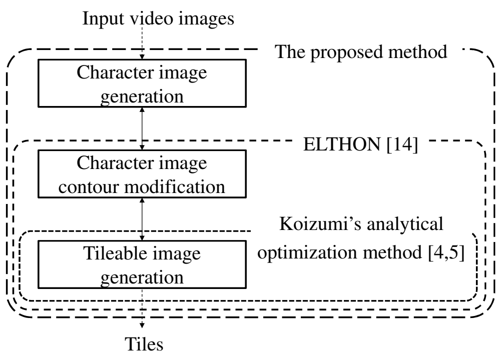

Input video images
The proposed method
Character image generation
ELTHON [14]
Character image contour modification
Koizumi's analytical optimization method [4,5]
Tileable image generation
Tiles

Fig. 5. Structure of the proposed method and relationship with the previous studies.

in the vertices can drastically alter the shape of the output figure [10]. Additionally, the segments of the output figure sometimes intersect (self-intersection). To alleviate these problems, Imahori et al. introduced weights that prioritize part of the given figure [11]. Nagata et al. improved Koizumi's algorithm (which assigns the same number of vertices to all tiling edges) by arbitrarily assigning vertices to the tiling edges [12]. These methods have resolved some of the problems in Koizumi's method, enabling the generation of complex tiles.

The second approach, meta-heuristics, was pioneered by Kaplan who directly deformed a tile shape using Simulated Annealing [3] Ono et al. proposed a Genetic Algorithm (GA)-based method that also manipulates tile shapes [6, 7]. Both formulations guarantee that the produced figure satisfies the edge-matching constraint; however, due to the enormous search space of the Escherization problem, the tiles produced by these methods are less concavo-convex than those generated by analytical optimization.

Hisatomi et al. proposed a hierarchical optimization method named ELTHON, which combines the two approaches to alleviate their drawbacks, i.e., the need for repetition adjustment of the input figure and the self-intersection avoidance in the analytical optimization approach, and the insufficient search performance of the meta-heuristics approach [10, 13, 14]. By employing Koizumi's method in its lower-layer optimizer, ELTHON efficiently produces tiles satisfying the edge-matching constraint that are similar to the given goal shape. Meanwhile, the GA in the upper-layer optimizer finds the optimal set of vertices and inputs a suitable TP to the lower-layer optimizer. In addition, ELTHON adopts a bidirectional-mapping-based shape-similarity distance function, allowing the generation of tiles with complex shapes.

2.3 Convolutional Variational Autoencoder

Autoencoders (AEs) are types of Neural Networks (NNs) for data coding in unsupervised learning scenarios. A canonical AE learns feature representations of a given dataset [15], whereas a Variational Autoencoder (VAE), which is a deep generative model proposed by Kingma [16], learns the complex distributions underlying the given data. Similarly to AEs, VAEs involve encoder and decoder part AEs, but VAEs assume that the samples in the training set are generated from the distribution.

The encoder network of VAE aims to find the distribution of the dataset in the latent space. The distribution is typically modeled by a diagonal Gaussian, whose mean and variance are output from the encoder. The decoder network reconstructs the data points from the input points of the latent representation, which define the conditional data distribution.

A Conv-VAE is a VAE whose encoder and decoder include convolution and deconvolution layers, respectively [8]. Similarly to VAE, Conv-VAE forms a latent variable space in which the decoder can reconstruct continuously changing data points.

3. THE PROPOSED METHOD

3.1 Key ideas

This paper proposes a method that finds a tile-designable character pose from video images. Besides selecting character images suitable for tiling from video frames, the method generates character images with poses that are absent in the given image set. To solve these problems, the proposed method develops two key concepts:

Idea 1: **Learning the latent representation of character images:** Using Conv-VAE, the proposed method learns the latent representation of the target character images. Conv-VAE represents the features of the target character images by their lower-dimensional latent features, and continuously deforms the character by changing the values of latent variables.

Idea 2: **Optimization in the latent variable space:** To find the character poses suitable for tiling, the proposed method optimizes the variables in the latent space. Various poses are generated by a self-adaptive Differential Evolution (jDE) [17] algorithm that gradually converges while keeping the population diversity comparable to that of the state-of-the-art meta-heuristics. The decoded character-pose images are passed to ELTHON tiling design method, which manipulates their contour shapes into tileable shapes. The generated tiles are then evaluated if they maintain their original characteristic shapes and poses.

Fig. 5 shows the structure of the proposed method and the relationship between the proposed method and ELTHON [14]. The proposed method inputs the video images involving a target character to be transformed into a tile, whereas ELTHON inputs an image of the target character contour (or its silhouette). Therefore, the proposed method can be regarded as a hierarchical optimization architecture with three layers: an upper-layer for character-pose generation, a middle layer for character-contour modification, and a lower-layer for transformation into tiles.

Note that the proposed method only produces the silhouettes (or contours) of the target character. Internal character images are ignored because the most difficult problem of Escher-like tiling is designing the tile contour; once the contour is generated, its interior is easily filled in by human illustrators.

Comment #1

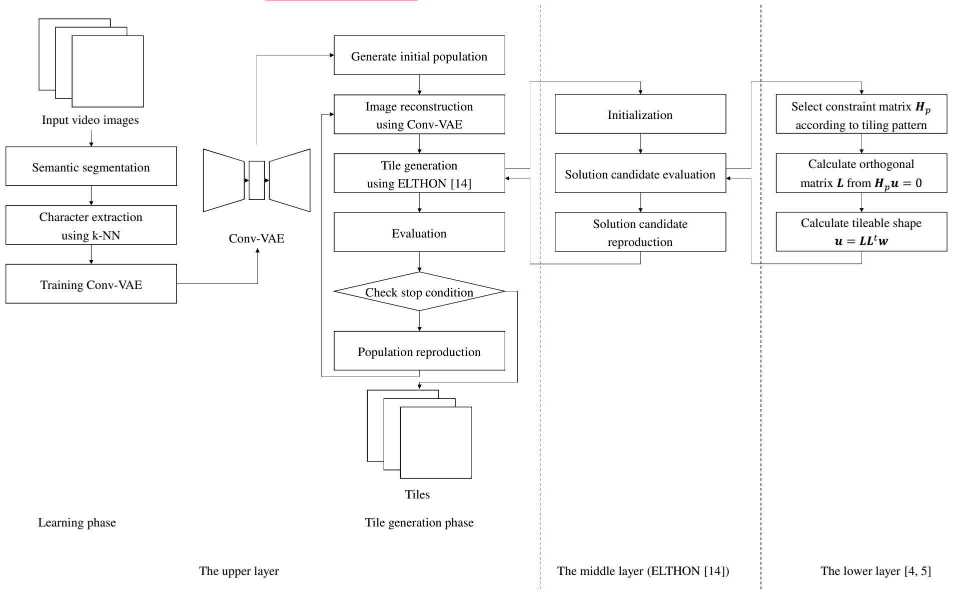

Input video images
Semantic segmentation
Character extraction using k-NN
Training Conv-VAE
Conv-VAE
Generate initial population
Image reconstruction using Conv-VAE
Tile generation using ELTHON [14]
Evaluation
Check stop condition
Population reproduction
Tiles
Initialization
Solution candidate evaluation
Solution candidate reproduction
Select constraint matrix H_p according to tiling pattern
Calculate orthogonal matrix L from $H_p u = 0$
Calculate tileable shape $u = LL^t w$
Learning phase
Tile generation phase
The upper layer
The middle layer (ELTHON [14])
The lower layer [4, 5]

Fig. 6. Process flow of the proposed method.

3.2 Process flow of the proposed method

The proposed method consists of two phases: learning and tile generation, as shown in Fig. 6. In the learning phase, the proposed method trains Conv-VAE on silhouette images of the target character extracted from the given videos. To obtain tileable shapes of the target character, the proposed method ignores the interior of the target character region as mentioned above. After training Conv-VAE, the proposed method tries to generate character images with poses appropriate for tiling. Comment #1 The middle and lower layers, which correspond to ELTHON [14], generate the tileable shapes in the third step of the generation phase.

At the beginning of the learning phase, the proposed method applies a semantic soft segmentation method [18] to video frames. The segments of the target character in T_f frames are then manually labeled, and the segments of that character in the remaining video frames are automatically extracted by a simple k-NN classifier that uses the HSV color histogram as features. The frames involving the extracted character segments are then translated into black and white silhouette images, and the character segments are trimmed and resized into $P_t \times P_t$ pixels, forming the training dataset for Conv-VAE. Some training images obtained by the above processes are shown in Fig. 7. Next, the proposed method trains the Conv-VAE on the character silhouette images. Because Conv-VAE can be trained without supervised information, the only manual work in the training process is labeling the character segments in the T_f video frames.

In the generation phase, the proposed method seeks the appropriate values of the latent variables that produce the target character-poses suitable for tiling. As shown in Fig. 6, the generation phase is based on a meta-heuristics algorithm, i.e.,

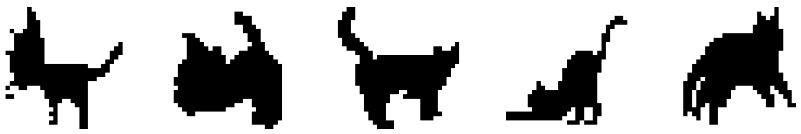
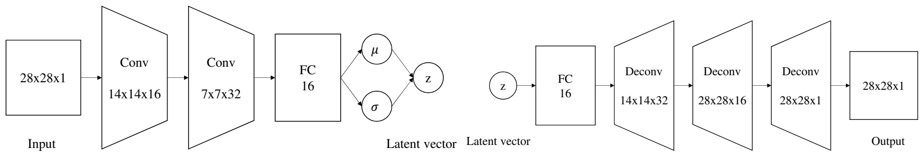


Fig. 7. Training image examples for Conv-VAE generated in the learning phase.



(a) The encoder network. (b) The decoder network.

Fig. 8. Network structure of Conv-VAE the proposed method uses.

repetition of solution candidate generation and evaluation. Because the optimization variables are latent variables learned by Conv-VAE, the character images are decoded by the decoder network. In the proposed method, the generated character images are converted into tiles by ELTHON, but any previous method that solves the Escherization problem is applicable. We employed ELTHON because it deforms the character contour, enabling the transformation into a tile with similarity to the given shape. After evaluating the obtained tiles, new solution candidates are generated. The above processes are repeated until the stop condition is satisfied.

3.3 Network structure and training of Conv-VAE

Fig. 8 shows the network structure of the Conv-VAE employed by proposed method. The encoder network contains two convolutional layers followed by a fully connected layer. All layers use a rectified linear activation function. The fully connected layer outputs the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$, which are constructed into a latent vector $\boldsymbol{z}$ for input datapoint $\boldsymbol{x}$ as follows:

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{1}$$

where $\odot$ denotes an element-wise product, and Comment #2 $\mathcal{N}$ is a Gaussian distribution. The structure of the decoder network opposes the encoder structure, i.e., a fully connected layer followed by three deconvolution layers. The above encoder-decoder network reduces the number of feature dimensions. The number of dimensions of the latent space is denoted as D_l.

Conv-VAE in the proposed method is trained by the standard VAE training approach [16]. The parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ of the encoder and the decoder, respectively, are optimized by maximizing the marginal log-likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. That is, the training minimizes Comment #2 the loss function $L(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$:

$$L(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -D_{\mathrm{KL}} \left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p_{\boldsymbol{\theta}}(\boldsymbol{z}) \right) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} [\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] \tag{2}$$

Comment #2 where D_{KL} and $\mathbb{E}_{q_{\boldsymbol{\phi}}}$ denote the Kullback-Leibler (KL) divergence and expectation, respectively. The probabilistic encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ is assumed as a multivariate Gaussian with the following diagonal covariance structure:

$$\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \log \mathcal{N} \left(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \boldsymbol{I} \right) \tag{3}$$

In the second version of the estimator proposed in [16], the KL divergence can be computed and differentiated without estimation as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) \simeq \frac{1}{2}\sum_{d=1}^{D_l}\left(1+\log((\sigma_d)^2)-\mu_d^2-\sigma_d^2\right)+\frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}^{(l)}) \tag{4}$$

where μ_d and σ_d are the d-th elements of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, respectively, and $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are simultaneously optimized using the gradient of eq. (4).

3.4 Optimization in the latent space

The proposed method creates character images with poses that are suitable for tiling by optimization in the latent space. The obtained images are not necessarily included in the given video images. The variables to be optimized are D_l-dimensional latent variables $\boldsymbol{z}$, and the objective function F_{lo} to be minimized is the weighted sum of the shape-similarity distances between the character shape $\boldsymbol{W}(\boldsymbol{z})$ generated by the Conv-VAE decoder and the transformed tile shape $\boldsymbol{U}(\boldsymbol{z})$ and the shape complexity of $\boldsymbol{U}(\boldsymbol{z})$:

$$F_{lo}(\boldsymbol{z}) = \gamma F_{BM}\left(\boldsymbol{U}(\boldsymbol{z}), \boldsymbol{W}(\boldsymbol{z})\right) + (1-\gamma)F_C\left(\boldsymbol{U}(\boldsymbol{z})\right) + \rho\left(\boldsymbol{w}(\boldsymbol{z})\right). \tag{5}$$

where $\boldsymbol{U}(\boldsymbol{z})$ denotes a tile obtained from the latent variables $\boldsymbol{z}$, F_{BM} is the bidirectional-mapping-based shape-similarity distance function applied in ELTHON [14], and $\boxed{\text{Comment \#2}}$ γ is a weight parameter $(0 < \gamma < 1)$. The function $F_C(\boldsymbol{U}(\boldsymbol{z}))$assesses the shape complexity as follows:

$$F_C(\boldsymbol{U}(\boldsymbol{z})) = \frac{l(\boldsymbol{U}(\boldsymbol{z}))}{2\sqrt{S(\boldsymbol{U}(\boldsymbol{z}))\pi}} \tag{6}$$

where $l(\boldsymbol{U}(\boldsymbol{z}))$ and $S(\boldsymbol{U}(\boldsymbol{z}))$ are the outline length and area of $\boldsymbol{U}(\boldsymbol{z})$, respectively. $F_C(\boldsymbol{U}(\boldsymbol{z}))$ computes the length ratio of the circumference of $\boldsymbol{U}(\boldsymbol{z})$ to the circumference of a circle having the same area as $\boldsymbol{U}(\boldsymbol{z})$. This function avoids the generation of tiles with excessively simple shapes. $\rho(\boldsymbol{W}(\boldsymbol{z}))$ is a penalty function that returns penalty values when $\boldsymbol{W}(\boldsymbol{z})$ violates the following tow constraints: "$\boldsymbol{W}$ includes no large blurred areas," and "the number of contiguous regions in $\boldsymbol{W}$ is unity." Accordingly, $\rho(\boldsymbol{W}(\boldsymbol{z}))$ is calculated as

$$\rho(\boldsymbol{W}(\boldsymbol{z})) = \rho_{blur}(\boldsymbol{W}(\boldsymbol{z})) + \rho_{cont}(\boldsymbol{W}(\boldsymbol{z})) \tag{7}$$

$$\rho_{blur}(\boldsymbol{W}(\boldsymbol{z})) = \begin{cases} V_{blur} & \text{if } var(\mathcal{L}(\boldsymbol{W}(\boldsymbol{z}))) < R_{blur} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\rho_{cont}(\boldsymbol{W}(\boldsymbol{z})) = \begin{cases} V_{cont} & \text{if } N_{area}(\boldsymbol{W}(\boldsymbol{z})) > 1 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $var(\mathcal{L}(\boldsymbol{W}(\boldsymbol{z})))$ denotes the variance of the Laplacian of $\boldsymbol{W}(\boldsymbol{z})$ [19, 20], and $N_{area}(\boldsymbol{W}(\boldsymbol{z}))$ denotes the number of contiguous regions in $\boldsymbol{W}(\boldsymbol{z})$. $\boxed{\text{Comment \#2}}$ V_{blur} and V_{cont} are penalty values corresponding to the first and second constraints, respectively, and R_{blur} is a threshold.

The proposed method employs jDE [17] that includes a simple self-adaptation mechanism, whereby each solution candidate has its own control parameters (scale factor SF_i and crossover rate CR_i) rather than unified parameters that are shared by the whole population. With the probability of R_{SF} and R_{CR} in each generation, jDE randomly changes the values of the control parameters within specified ranges. Although the convergence speed of jDE is slower than other self-adaptive meta-heuristics such as JADE [22], SHADE [23, 24], and the covariance matrix adaptation evolutionary strategy [25], wide upper and lower limits of parameters SF_i and CR_i allows good convergence properties of jDE by maintaining population diversity (facilitating escape from local optima) [21].

Fig. 9 shows the pseudo-code of the proposed optimization method in the latent space using jDE. In the first step, all solution candidates $\boldsymbol{z}_i^{(0)} = \left(z_{i,1}^{(0)}, z_{i,2}^{(0)}, \dots z_{i,D_l}^{(0)}\right)$ $(i \in \{1, \dots, N_p\})$ are initialized to uniform random values. The scale factors $SF_i^{(0)}$ and crossover rates $CR_i^{(0)}$ for each candidate i are initialized to random numbers within the rages $[SF_L, SF_U]$ and $[0, 1]$, respectively. A set of all solution candidates in generation g comprises the population $\boldsymbol{P}^{(g)}$ of that generation. After generating initial population $\boldsymbol{P}^{(0)}$, all candidates are evaluated by calculating $F_{lo}(\boldsymbol{z}_i^{(0)})$ for $i \in \{1, \dots, N_p\}$. In this calculation, $\boldsymbol{z}_i^{(0)}$ is input to the decoder network of Conv-VAE to obtain $\boldsymbol{W}(\boldsymbol{z}_i^{(0)})$, then ELTHON is applied to $\boldsymbol{W}(\boldsymbol{z}_i^{(0)})$, eq. (5) is calculated using the obtained tile $\boldsymbol{U}(\boldsymbol{z}_i^{(0)})$.

The main loop of jDE (lines 4 through 10 in Fig. 9) iterates the parameter update, population reproduction, and evaluation. The parameters SF_i and CR_i of solution candidate i are updated as follows:

$$SF_i^{(g+1)} = \begin{cases} SF_L + SF_U \cdot r_1 & \text{if } r_2 < R_{SF} \\ SF_i^{(g)} & \text{otherwise} \end{cases} \tag{10}$$

$$CR_i^{(g+1)} = \begin{cases} r_3 & \text{if } r_4 < R_{CR} \\ CR_i^{(g)} & \text{otherwise} \end{cases} \tag{11}$$

where r_j are uniform random values in the range $[0, 1]$, and R_{SF} and R_{CR} are probabilities of changing SF_i and CR_i, respectively. SF_L and SF_U determine the ranges of the scale factor.

After updating $SF_i^{(g)}$ and $CR_i^{(g)}$, jDE reproduces the solution candidates using the mutation and crossover operators, which are applied to all solution candidates. The mutation produces a mutant vector $\boldsymbol{v}_i^{(g)}$ depending on the DE strategies, such as “DE/rand/1” and “DE/best/2”. The proposed method employs the most popular “DE/rand/1” mutation strategy, which produces $\boldsymbol{\nu}_i^{(g)}$ by the following equation:

$$\boldsymbol{\nu}_i^{(g)} = \boldsymbol{z}_{r_5}^{(g)} SF_i(\boldsymbol{z}_{r_6}^{(g)} - \boldsymbol{z}_{r_7}^{(g)}) \tag{12}$$

where $r_5 \neq r_6 \neq r_7$. Subsequently, the trial vectors $\boldsymbol{\tau}_i^{(g)}$ are generated through the crossover operator. In general DE, two popular crossover methods are mainly used: binomial and exponential crossover, and this study employs binomial crossover as follows:

$$\tau_{i,d}^{(g)} = \begin{cases} \nu_{i,d}^{(g)} & \text{if } r_{8,d} \leq CR_i \text{ or } d = d_{rand} \\ z_{i,d}^{(g)} & \text{otherwise} \end{cases} \tag{13}$$

where $\tau_{i,d}$, $\nu_{i,d}$, $z_{i,d}$ are D_l-th elements of target vector $\boldsymbol{\tau}_i$, mutant vector $\boldsymbol{\nu}_i$, and i-th solution candidate $\boldsymbol{z}_i$, respectively (i.e., $\boldsymbol{z}_i = \{z_{i,1}, z_{i,2}, \dots, z_{i,D_l}\}$), and $r_{8,d}$

Algorithm 1 Optimization in the latent space using jDE

```
1: $g \leftarrow 0$
2: Generate an initial population $\boldsymbol{P}^{(g)}$ consisting of individuals $\boldsymbol{z}_1^{(g)}, \ldots, \boldsymbol{z}_{N_P}^{(g)}$.
3: Evaluate all solution candidates $\boldsymbol{z}_i^{(g)}$ $(i \in \{1, \ldots, N_P\})$ in population $\boldsymbol{P}^{(g)}$.
4: while $g$ reaches the limit $g_{limit}$ do
5:     for $i = 1$ to $N_P$ do
6:         Update $\boldsymbol{z}_i^{(g)}$'s control parameters: $SF_i$ and $CR_i$.
7:         Apply mutation and crossover operators for $\boldsymbol{z}_i^{(g)}$.
8:         Evaluate all trial vectors $\boldsymbol{\tau}_i^{(g)}$ $(i \in \{1, \ldots, N_P\})$.
9:         Add the better one from either $\boldsymbol{z}_i^{(g)}$ or $\boldsymbol{\tau}_i^{(g)}$ to $\boldsymbol{P}^{(g+1)}$.
10:    end for
11:    $g \leftarrow g + 1$
12: end while
```

Fig. 9. Algorithm of the Self-Adaptive DE.

is a random number ranging from 0 to 1. The combined strategy used in the proposed method is denoted as "DE/rand/1/bin".

Finally, the trial vectors $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{N_P}$ are evaluated Comment #5 by the fitness function eq.(5), and their fitness values are compared with those of solution candidates $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N_P}$. The solution candidate $\boldsymbol{z}_i^{(g)}$ is replaced with the trial vector $\boldsymbol{\tau}_i^{(g)}$ $(\boldsymbol{z}_i^{(g)} \leftarrow \boldsymbol{\tau}_i^{(g)})$ if $F_{lo}(\boldsymbol{\tau}_i^{(g)}) < F_{lo}(\boldsymbol{z}_i^{(g)})$. The main loop is iterated until the stop condition is satisfied.

3.5 Tile generation using ELTHON

In the generation phase, the proposed method transforms the character image $\boldsymbol{W}(\boldsymbol{z})$ decoded by the decoder network into a tile $\boldsymbol{U}(\boldsymbol{z})$ using ELTHON [14], which has a hierarchical optimization architecture. As shown in Fig. 5, the upper and lower layers of ELTHON correspond to the middle and lower layers of the proposed method, respectively.

The upper-layer of ELTHON receives $\boldsymbol{W}$ and downsamples it to $\tilde{\boldsymbol{W}}$. After downsampling, the tile generated by the lower-layer optimizer always satisfies the edge-matching constraints and is similar to the given $\boldsymbol{W}$. Exploiting the GA in its upper-layer optimizer, ELTHON computes the bidirectional-mapping-based shape-similarity distance function, which is not differentiable but provides a good image-similarity metric for the Escherization problem. Besides transforming $\boldsymbol{W}$, the upper-layer optimizer of ELTHON selects a suitable TP for $\boldsymbol{W}$ among its 29 TPs [14]. By representing the deformation of $\boldsymbol{W}$ and TP selection as chromosomes in the GA, the simultaneous solution of both problems becomes tractable. Unlike Koizumi's method [4, 5] and the previous meta-heuristics approaches [6, 7], the output tile of ELTHON rarely includes self-intersections.

4. EVALUATION

4.1 Experimental setup

The effectiveness of the proposed method was experimentally verified in tile generation from videos. For this experiment, we created a dataset including 69 videos of kittens (2,673 frames per video on average). To create this dataset, we manually labeled 150 frames for segmentation (i.e., $T_f = 2$ or 3, depending on the video length).

The Conv-VAE-related parameters in the proposed method were configured as follows. The input image size P_t was set to 28 pixels, and the reduced number of dimensions of the latent space D_l was set from 6 to 10. The batch size and number of epochs were set to 128 and 100, respectively. For optimization in the latent space, the jDE control parameters F_L, F_U, R_{SF}, and R_{CR} were set to 0.1, 0.9, 0.1, and 0.1, respectively. $\boxed{\text{Comment \#3}}$ The population size and iteration limit g_{limit} in the jDE were set to 10 and 50, respectively. $\boxed{\text{Comment \#2}}$ The remaining parameters V_{blur}, V_{cont}, R_{blur}, and γ were set to 10^3, 10^3, 3.0×10^3, and 0.01, respectively.

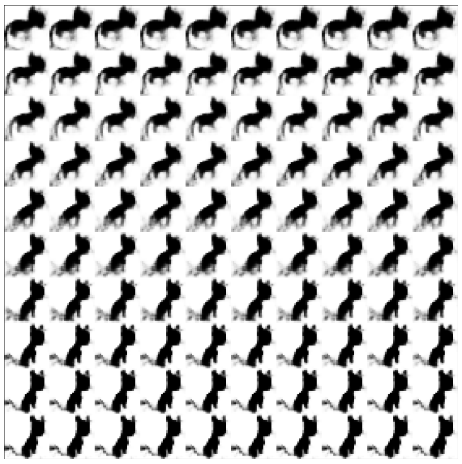
As the ELTHON model [14], we applied a generation alternation model with uniform crossovers, bit flip mutations, and a Minimal Generation Gap [26]. The maximum function evaluation count (stop condition), population size, crossover rate, mutation rate, and number of trials for each solution candidate were set to 25,000, 50, 0.4, 0.03, and 1 respectively.

4.2 Experimental results

Fig. 10 shows examples of the visualized latent space of the trained Conv-VAE. A case of $D_l = 7$ shown in Fig. 10(a), z_1 and z_6 were changed within the ranges of $[-4.0, 0.0]$ and $[-4.0, 4.0]$, respectively, while z_2, z_3, z_4, z_5 and z_7 were fixed at 2.0, 0.0, 0.0, 0.0, and 0.0, respectively. As the vertical axis gradually increased, the kitten was deformed into a sitting pose. Similarly, a case of $D_l = 10$ shown in Fig. 10(b), z_3 and z_5 were changed within the ranges of $[-4.0, 4.0]$ and $[-4.0, 0.0]$, respectively, while z_1, z_2, z_4, z_6, z_7, z_8, z_9 and z_{10} were fixed at 0.0, 1.0, 2.0, 1.0, 2.0, 1.0, 1.0 and 2.0, respectively. As the horizontal axis increased, the kitten looking left gradually turned to the right.

Figs. 11 through 15 show the target character images with poses suitable for tiling, which were generated by optimizing the latent space using jDE, and the tiles generated by ELTHON, along with the values of F_{lo}. The figures show the results for the number of dimensions of the latent space was changed from six to ten, respectively. Benefiting from the multi-point search, the proposed method explored a variety of character poses suitable for tiling. The most similar training image for each output is also shown in each figure. Some outputs, such as the third and the fourth outputs in Fig. 11, the first output in Fig. 13, and the third and the fifth outputs in Fig. 15, were dissimilar to the training images, e.g., the positions of its tail or feet look different. indicating that Conv-VAE successfully created intermediate poses among the training images, and that jDE optimization in the latent space successfully found the interpolated poses that were suitable for tiling.

Fig. 16 shows the tiling results of the experimentally obtained tiles. The internal images were drawn by a human illustrator. Clearly, the proposed method


(a) Example 1 when $D_l = 7$ (the horizontal and vertical axes correspond to $-4.0 \leq z_1 \leq 0.0$ and $-4.0 \leq z_6 \leq 4.0$, respectively. z_2, z_3, z_4, z_5 and z_7 were fixed to 2.0, 0.0, 0.0, 0.0 and 0.0, respectively).

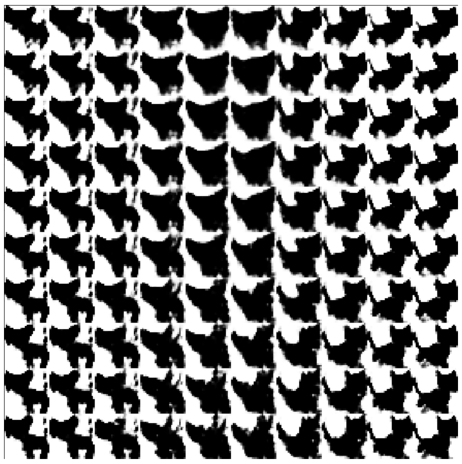

(b) Example 2 when $D_l = 10$ (the horizontal and vertical axes correspond to $-4.0 \leq z_3 \leq 4.0$ and $-4.0 \leq z_5 \leq 0.0$, respectively. z_1, z_2, z_4, z_6, z_7, z_8, z_9 and z_{10} were fixed to 0.0, 1.0, 2.0, 1.0, 2.0, 1.0, 1.0 and 2.0, respectively).

Fig. 10. Examples of the visualization results of the trained latent space.

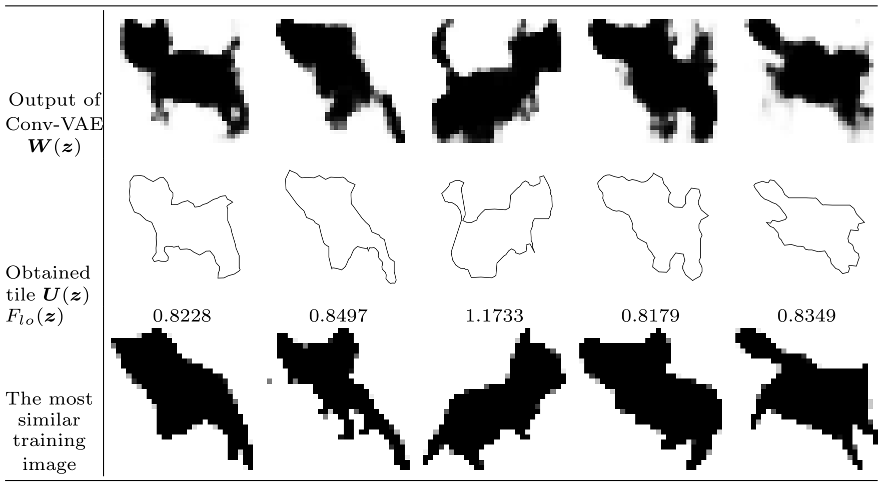

Output of Conv-VAE $\boldsymbol{W}(\boldsymbol{z})$
Obtained tile $\boldsymbol{U}(\boldsymbol{z})$ $F_{lo}(\boldsymbol{z})$
0.8228
0.8497
1.1733
0.8179
0.8349
The most similar training image

Fig. 11. Generated character images and obtained tiles when $D_l = 6$.

generated tileable figures of kittens in various poses.

5. CONCLUSION

This paper proposes a method that generates tileable shapes of characters shown in videos. After learning the feature space using Conv-VAE, the method finds character-poses suitable for tiling, and generates new poses that are not

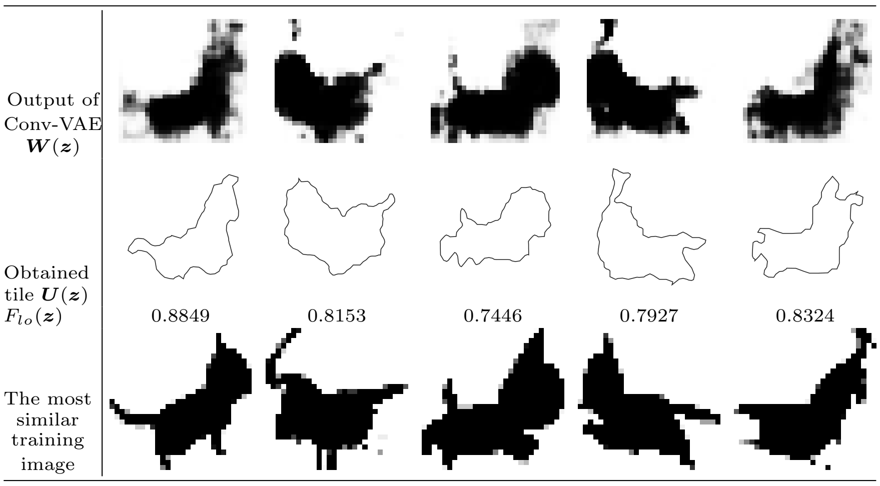

Output of Conv-VAE $\boldsymbol{W}(\boldsymbol{z})$
Obtained tile $\boldsymbol{U}(\boldsymbol{z})$ $F_{lo}(\boldsymbol{z})$
0.8849
0.8153
0.7446
0.7927
0.8324
The most similar training image

Fig. 12. Generated character images and obtained tiles when $D_l = 7$.

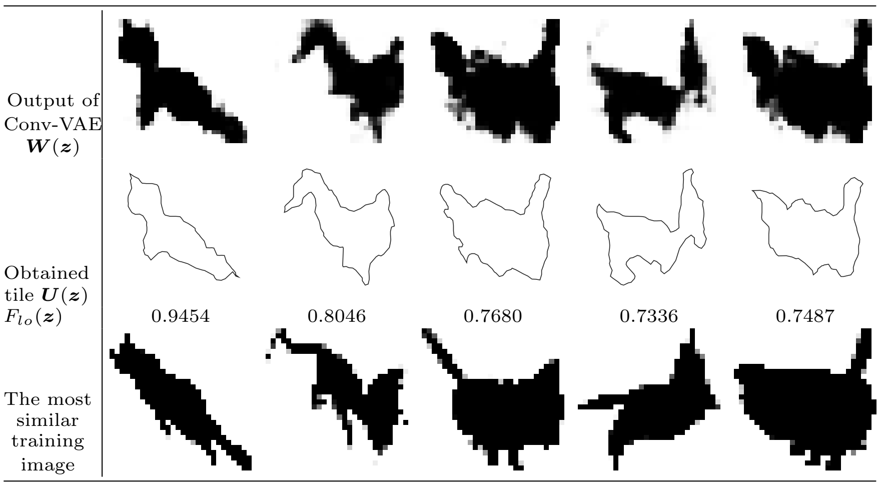

Output of Conv-VAE $\boldsymbol{W}(\boldsymbol{z})$
Obtained tile $\boldsymbol{U}(\boldsymbol{z})$ $F_{lo}(\boldsymbol{z})$
0.9454
0.8046
0.7680
0.7336
0.7487
The most similar training image

Fig. 13. Generated character images and obtained tiles when $D_l = 8$.

included in the videos. Experimental results showed that the proposed method extracted character images with various poses suitable for tiling, some of which were not included in the training dataset.

In future, we plan to employ VQ-VAE-2 [27] to produce higher-resolution output images because utilizing other meta-heuristics allows optimization in its discrete latent space.

REFERENCES

1. C. S. Kaplan, "Introductory tiling theory for computer graphics," *Synthesis Lectures on Computer Graphics and Animation*, Vol. 4, no. 1, 2009, pp. 1–113.

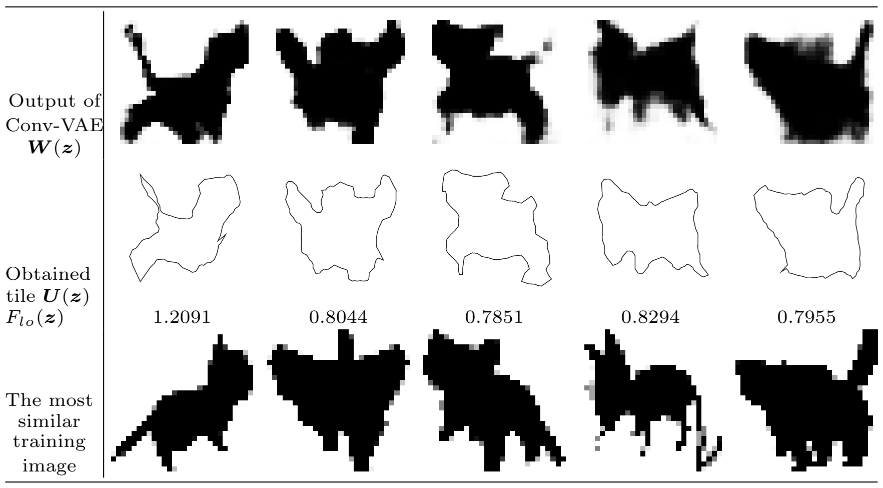

Output of Conv-VAE $\boldsymbol{W}(\boldsymbol{z})$
Obtained tile $\boldsymbol{U}(\boldsymbol{z})$
$F_{lo}(\boldsymbol{z})$
1.2091
0.8044
0.7851
0.8294
0.7955
The most similar training image

Fig. 14. Generated character images and obtained tiles when $D_l = 9$.

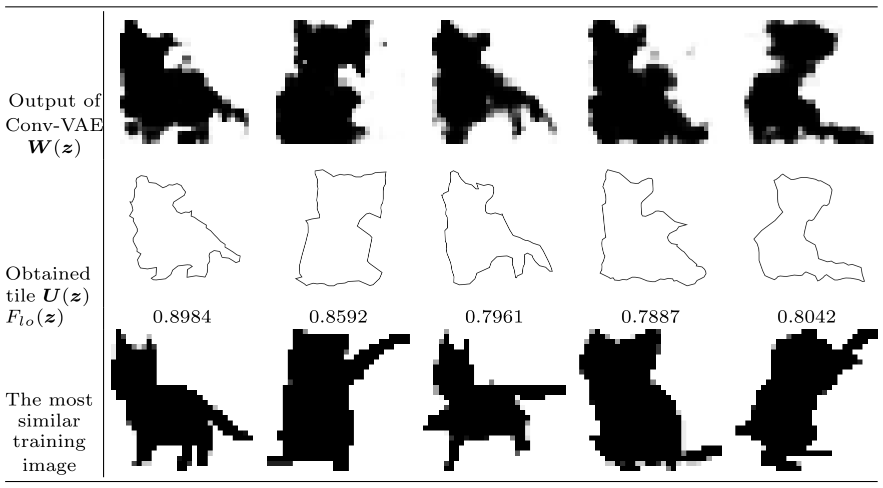
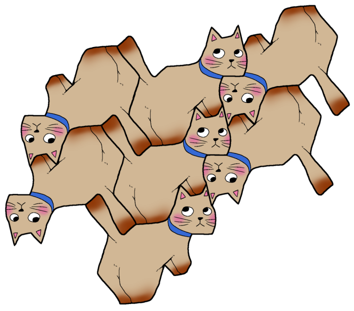
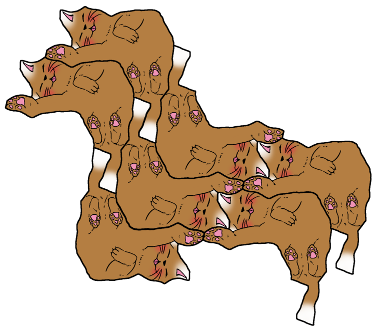
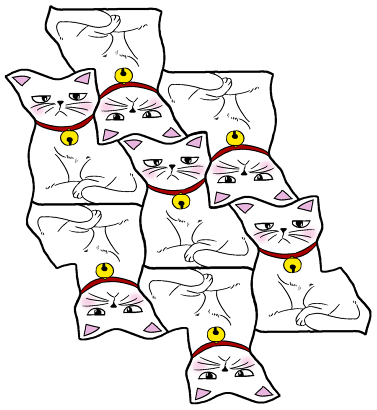

Output of Conv-VAE $\boldsymbol{W}(\boldsymbol{z})$
Obtained tile $\boldsymbol{U}(\boldsymbol{z})$
$F_{lo}(\boldsymbol{z})$
0.8984
0.8592
0.7961
0.7887
0.8042
The most similar training image

Fig. 15. Generated character images and obtained tiles when $D_l = 10$.

2. A. Hisatomi, H. Koba, K. Mizuno, and S. Ono, "Application of escher-like tiling design to confectionery shape design," *Int'l Conf. Technologies and Applications of Artificial Intelligence (TAAI).* , 2019, pp. 1–6.
3. C. S. Kaplan and D. H. Salesin, "Escherization," *Annual Conf. Computer Graphics and Interactive Techniques*, (SIGGRAPH) 2000, p. 499-510.
4. H. Koizumi and K. Sugihara, "Computer-aided design of escher-like tiling," *NICOGRAPH Paper Contest*, 2009.
5. H. Koizumi and K. Sugihara, "Maximum eigenvalue problem for escherization," *Graphs and Combinatorics*, Vol. 27, no. 3, 2011, 431.
6. S. Ono, M. Kisanuki, H. Machii, and K. Mizuno, "Figure pattern creation support for escher-like tiling by interacitive genetic algorithms," Asia Pacific Symp. Intelligent and Evolutionary Systems, Vol. 1, 2014, pp. 421–432.


(a) The third tile of Fig 11


(b) The fourth tile of Fig 12


(c) The first tile of Fig 14

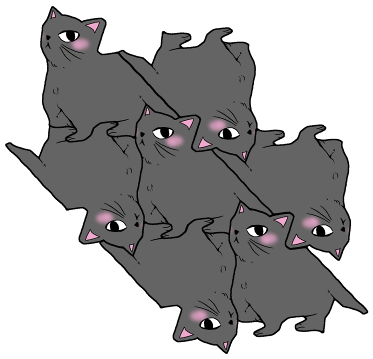

(d) The first tile of Fig 15

Fig. 16. Example tiled arts of the obtained tiles.

7. S. Ono, M. Kisanuki, H. Machii, and K. Mizuno, "Creation support for escher-like tiling patterns by interactive genetic algorithms," *SIGGRAPH Asia*, 2014.
8. A. D. Jason Rock, Threerasit Issararin and D. Forsyh, "Authoring image decompositions with generative models," *Computing Research Repository*, 2016.
9. B. Grunbaum and G. C. Shephard., *Tilings and Patterns.* New York: W. H. Freeman, 1987.
10. A. Hisatomi, H. Koba, K. Mizuno, and S. Ono, "Escher-like tiling design using estimation of distribution algorithm," *Int'l Symp. Artificial Life and Robotics (AROB).* , 2019, pp. 262–265.
11. S. Imahori, S. Kawade, and Y. Yamakata, *Escher-like Tilings with Weights.* Cham: Springer Int'l Publishing, 2016, pp. 132–142.
12. Y. Nagata and S. Imahori, "An efficient exhaustive search algorithm for the escherization problem," *Algorithmica*, 2020, pp. 1–33.
13. A. Hisatomi, H. Koba, M. Kamizono, K. Mizuno, and S. Ono, "Escher-like tiling design using hierarchical optimization," *Genetic and Evolutionary Computation Conf. Companion*, 2017, pp. 89–90.
14. A. Hisatomi, H. Koba, K. Mizuno, and S. Ono, "Elthon: An escher-like tile design method using hierarchical optimization," *Applied Soft Computing*,

(under review). [Opened only for reviewers: https://www.ibe.kagoshima-u.ac.jp/~ono/Tmp/Hisatomi2020_ASoC_under-review.pdf]

15. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, Vol. 313, no. 5786, 2006, pp. 504–507.
16. M. W. Diederik P. Kingma, "Auto-encoding variational bayes," *ICLR*, 2014.
17. J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *Trans. Evolutionary Computation*, Vol. 10, no. 6, 2006, pp. 646–657.
18. A. Yagiz, O. Tae-Hyun, P. Sylvain, P. Marc, and M. Wojciech, "Semantic soft segmentation," *SIGGRAPH*, 2018.
19. J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," *Int'l Conf. Pattern Recognition (ICPR)*, Vol. 3. , 2000, pp. 314–317.
20. S. Pertuz, D. Puig, and M. A. Garcia, "Analysis of focus measure operators for shape-from-focus," *Pattern Recognition*, Vol. 46, no. 5, 2013, pp. 1415 – 1432.
21. S. Eguchi, Y. Matsugano, H. Sakaguchi, S. Ono, H. Fukuda, R. Furukawa, and H. Kawasaki, "A comparative study on self-adaptive differential evolution algorithms for test functions and a real-world problem," *Int'l Conf. Learning and Intelligent Optimization.* , 2015, pp. 131–136.
22. J. Zhang and A. Sanderson, "Jade: Adaptive differential evolution with optional external archive," *Trans. Evolutionary Computation*, Vol. 13, no. 5, 2009, pp. 945–958.
23. R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," *Congress on Evolutionary Computation (CEC)*, 2013, pp. 71–78.
24. R. Tanabe and A. Fukunaga, "Evaluating the performance of shade on cec 2013 benchmark problems," *Congress on Evolutionary Computation (CEC)*, 2013, pp. 1952–1959.
25. N. Hansen, "The cma evolution strategy: a comparing review," *Towards a new evolutionary computation.* Springer, 2006, pp. 75–102.
26. H. Satoh, "Minimal generation gap model for gas considering both exploration and exploitation," *Int'l Conf. Fuzzy Logic, Neural Nets and Soft Computing*, 1996, pp. 494–497.
27. A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *Advances in Neural Information Processing Systems*, 2019, pp. 14 866–14 876.