On Automatic Generation of Escher-like Metamorphosis

Shunsuke Nakamatsu[1] and Chiaki Sakama[2]

[1] Graduate School of Systems Engineering, Wakayama University, Wakayama, Japan
nakamatsu.shunsuke@g.wakayama-u.jp
[2] Department of Systems Engineering, Wakayama University, Wakayama, Japan
sakama@wakayama-u.ac.jp

Abstract. The Escherization problem seeks a tiling for an input plane figure S with a new tile figure T such that T is as close as possible to S. In this study, we conduct an automatic generation of Escher-like "Metamorphosis". More precisely, an input image is represented as a polygon and is divided into triangles, which are related to those of a tile image. We produce tile images using an affine transformation and introduce conditions for connecting those images. As a result of experiments, we succeed in partly simulating Escher's Metamorphosis and also produce new Metamorphoses using color images.

Keywords: Escherization problem · tessellation · tiling · Metamorphosis

1 Introduction

In recent years, systems that use artificial intelligence (AI) to automatically generate images and paintings, such as Midjourney and Stable Diffusion have been emerged.[3] The outputs of those systems are so sophisticated that it is often difficult to distinguish them from products by human artists. Currently, most of those systems generate images based on keywords or text input by a user, and it is still challenging to automatically generate works based on human creativity and imagination. An example of artwork that is difficult for computer-based automatic generation is optical illusions. The works of the Dutch graphic artist M. C. Escher [1] from the 20th century are a representative example of such illusions, where structures that cannot be reproduced in three dimensions are depicted in two dimensions. Escher is also known as an artist who pioneered tiling art, with his *Metamorphosis* being a representative work in which a figure in a tile continuously transforms to a different figure. *Tiling* refers to the operation of covering a plane without gaps or overlaps using a finite set of flat shapes (tiles), also known as plane filling or *tessellation*.

There have been several attempts to automatically generate tiling art using computers, and the problem of finding shapes that satisfy tiling while maintaining a similar shape to the input figure is called the *Escherization problem* [2]. Several studies propose different solutions to the problem [2, 4, 6–12], while few studies realize automatic generation of Escher-like Metamorphosis that accompanies dynamic and continuous changes of both tile shapes and images. It is worth noting that the state-of-the-art text to image AI does not generate Escher-style Metamorphosis using a simple prompt. Fig. 1

[3] https://www.midjourney.com; https://stablediffusionweb.com

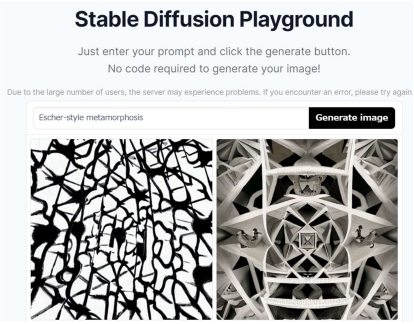


Fig. 1: Output of Stable Diffusion

shows an output of Stable Diffusion with the simple prompt 'Escher-style Metamorphosis', which is quite different from the intended output. As such, it is challenging to develop a system that can produce original Metamorphoses from input images.[4]

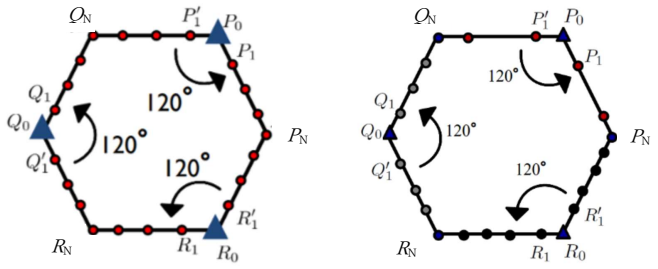
In this study, we develop an algorithm to automatically generate Escher-like Metamorphosis using tiling. The "Escher-like Metamorphosis" targeted in this study is defined as an image that gradually changes the tiling shape and pattern of one input image into those of another input image. The goal of this study is to automatically generate such images using an algorithm dedicated for Escher-style tiling problems. To achieve this, an input image is represented as a polygon and is divided into triangles that are related to those of a tile image. Tile images are generated using affine transformations, and constraints are introduced to connect the two tile images. We attempt to reproduce pattern changes in the original Metamorphosis by Escher, and also generate original Escher-like Metamorphosis using color images. The rest of this paper is organized as follows. Section 2 introduces the Escherization problem and related studies. Section 3 presents our approach to generating Escher-like Metamorphosis. Section 4 presents experimental results and analyses. Section 5 summarizes the paper.

2 Escherization Problem

Given a closed plane figure S, the problem of finding a new plane figure T that satisfies the following two conditions is called the *Escherization* [2]: (1) T is as close as possible to S; and (2) copies of T fit together to form a tiling of the plane. S is called an *input shape* or *goal shape*, and T is called a *tiling shape*. Several algorithms have been proposed for the problem. Kaplan *et al*. [2] propose an algorithm using simulated annealing to solve the problem for convex figures. It takes as input a goal shape and a set of *isohedral tiling*[5] to search for an optimal tiling. Koizumi *et al*. [4] capture figures as n-point polygons and evaluate the similarity of the shapes using the *Procrustes distance* which is scale and rotation invariant [5]. They formulate the Escherization problem as an eigenvalue problem and compute tiling for non-convex complicated figures. Imahori

[4] Recently, some prompts generating tessellation using Midjourney are reported on the Web.

[5] A tiling whose features are repeated regularly over a plane and which is constructed from only one tile fitted to itself in a number of different orientations is said to be *isohedral* [3].



(a) Each edge contains the same number of points.

(b) Each edge may contain different number of points.

Fig. 2: Constraints for IH7 ([4, 6])

et al. [6, 7] improve [4] by resolving the sensitivity of selecting points in the goal polygon, and propose an efficient algorithm for searching appropriate tile shapes. Ono *et al*. [8] use the genetic algorithm for producing possible tiling. Sugihara [9] introduces an algorithm that reproduces Escher's transmutation 'Sky and Water' that smoothly transforms a tile pattern into another one. Lin *et al*. [10] introduce a system to create a variety of Escher-like transmutations. Liu *et al*. [11] develop an interactive system for children to create Escher-like pattern evolution. Liu *et al*. [12] adapt Escher's dual shape tiling and perception effect to 3D manufacturing. These studies handle the Escher-style tiling problem in various ways, but no study challenges producing Escher-like Metamorphosis that accompanies dynamic and continuous changes of both tile shapes and images.

3 Automated Generation of Escher-like Metamorphosis

3.1 Mathematical Characterization for Tiling

We first overview the mathematical formulation of [4] on which our algorithm is based. First, an input shape and a tile shape are represented by polygons with n points on the boundary. Each n-point polygon is represented by a $2 \times n$ matrix in which the i-th column is the coordinates of the i-th vertex. Let W be a matrix representing an input shape, and U a matrix representing a tile shape. By definition, U and W change by the selection of the first vertex (called the *start point*). Then the goal is minimizing the Procrustes distance $d(U, W)$.[6] Next, constraint conditions are introduced for each type of the isohedral tiling. There are 93 types of tiling in isohedral tiling, from IH1 to IH93 [3]. For instance, the shape of IH7 is a hexagon and all the points on one edge must coincide with points on another edge by rotating the edge by 120° around the vertices P_0, Q_0, and R_0 that divide the hexagon into three equal parts (Fig. 2(a)).

The condition of tiling edges is represented by the following equations:

$$S(P'_i - P_0) = P_i - P_0, \quad S(Q'_i - Q_0) = Q_i - Q_0, \quad S(R'_i - R_0) = R_i - R_0 \ (i = 1, \ldots, N)$$

where $N(= n/6)$ is the number of points on the edge, S is the matrix representing 120° rotation, and P_i, P'_i, Q_i, Q'_i, R_i, R'_i are points on edges. In addition to the above

[6] $d(U, W)$ is defined as: $d^2(U, W) = 1 - \frac{\|UW^\top\|^2 + 2\det(UW^\top)}{\|U\|^2 \|W\|^2}$ where $\|X\| = \sqrt{\mathrm{tr}(X^\top X)}$.

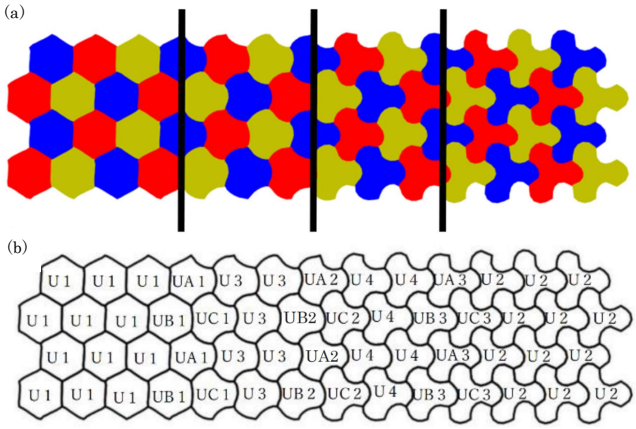


Fig. 3: Tiling for Metamorphosis

conditions, the condition that the centroid of the tile must coincide with the one of the input shape is introduced. Those conditions are represented as the linear equation $A\boldsymbol{u} = 0$ where $\boldsymbol{u}$ is a vector representing the coordinates of the tile shape and A is a coefficient matrix. The equation is further transformed to $\boldsymbol{u} = B\xi$ where B is a matrix consisting of the orthonormal basis of $Ker(A)$ (kernel of A) and ξ is a parameter vector. Finally, the problem of minimizing the Procrustes distance $d(U, W)$ is characterized by finding the maximum eigenvalue of $B^\top V B$, which is computed by $\frac{|B^\top W|^2}{|W|^2}$, where V is a matrix defined by vectors in W. In [4] the same number of points are assigned to every tiling edge. Imahori *et al.* [6] relax the condition and propose a local search-based method to find an appropriate assignment of points for an input polygon. It can place points non-uniformly on the boundary of an input figure to form a polygon, and different numbers of points can be assigned to tiling edges (Fig. 2(b)).

3.2 Tiling for Metamorphosis

To realize Escher-like Metamorphosis, we extend the algorithm of [4] and constraints introduced by [6]. We use a tile shape in a hexagon that is the IH7 type in isohedral tiling, since the tiling used by Escher in Metamorphosis I is IH7. So in what follows we describe constraints specific to IH7. Let n be the number of points on the boundary of an input shape. Then *reference points* are defined as $(P_1, P_2, P_3, P_4, P_5, P_6) = (x_1, x_2, x_3, x_4, x_5, x_6)$ satisfying the conditions that (i) $x_1 = 0$ and $x_6 < n$, and (ii) x_i $(2 \leq i \leq 5)$ is an even number such that $0 < x_i < x_6$. We explore possible arrangements of reference points that satisfy the constraint of IH7. For each such reference point, those that minimize the Procrustes distance are selected by changing the start point.

We use color images as well as figures for tiling.[7] Tile images are obtained by fitting the objects in the input image to a tile shape using an affine transformation. In [4] tiling

[7] The word 'figure' is used for a drawing and 'image' is used for a graphic, but the distinction is not strict.

is done using a single shape. In Escher-like Metamorphosis, on the other hand, tile shapes gradually change while tiling. Then we generate an intermediate tile between the two input shapes, and a boundary tile that connects the boundary between different tile shapes naturally. A pattern of Metamorphosis is generated in the following manner. First, the plane for an output is divided into blocks (Fig. 3(a)), and tiling is done using a single tile shape in the same block. The leftmost block represents the tiling of the first input shape, the rightmost block represents the tiling of the second input shape, and the in-between blocks are composed of intermediate shapes of tiling. The boundary of each block is represented by black lines in Fig. 3(a) where one (blue) tile is located at each odd row and two (red/yellow) tiles are located at each even row on the boundary. The first input figure is then morphed into the second input figure via intermediate figures. Fig. 3(b) illustrates the composition of different tile figures. In the figure, the first input figure is represented as U1, the second input figure as U2, the first intermediate figure as U3, and the second intermediate figure as U4. The boundary figures between U1 and U3 are UA1, UB1, UC1; the boundary figures between U3 and U4 are UA2, UB2, UC2; and the boundary figures between U4 and U2 are UA3, UB3, and UC3.

3.3 Boundary constraints

Koizumi *et al.* [4] introduce constraints on IH7 such that all the points on the edge overlap by rotating the figure by 120 degrees around the vertices. Under the constraint, however, a boundary line is not always straight. In Fig. 4(a), a boundary line (short white lines) moves to the left as it goes down. Such boundaries could be a cause for producing gaps or overlap in tiling. To make a boundary line straight, we introduce an additional condition. Consider the angle θ formed by $P_1P_4P_5$ on the top (blue) tile of Fig. 4(b). Since IH7 is the tiling method by the 120° rotation, set $\theta = 60°$ to make the boundary straight (Fig. 4(c)). Next consider the ratio between the length of P_1P_4 and the length of P_4P_5 (Fig. 4(d)). If the ratio is different between the left and right figures, points on the boundary shape will not fit perfectly and a gap could be produced. To avoid this, we set the ratio between $\overline{P_1P_4}$ and $\overline{P_4P_5}$ as a constant for all tile shapes. This is done by fixing $a : b$ in Fig. 4(e). The ratio is set to $a : b = 2 : 1$ to obtain a tile shape closer to a regular hexagon (Fig. 4(f)). The constraint is represented as the formula:

$$S(\boldsymbol{p}_5 - \boldsymbol{p}_4) = \frac{1}{2}(\boldsymbol{p}_1 + \boldsymbol{p}_4)$$

where $\boldsymbol{p}_1$, $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$ are vectors representing P_1, P_4 and P_5, respectively; and S is a matrix for 60° rotation. $\boldsymbol{p}_5 - \boldsymbol{p}_4$ represents a vector $\overrightarrow{P_4P_5}$. The formula represents that rotating the vector $\overrightarrow{P_4P_5}$ in 60° coincides the midpoint of P_1 and P_4.

3.4 Intermediate figures

The shape of an intermediate figure between two input figures is computed as follows. First, weighting the two-dimensional matrices representing each figure by taking into account the proportion of the figures, then adding them and dividing by the sum of the weights. Fig. 5 shows an example of producing an intermediate figure (c) from two

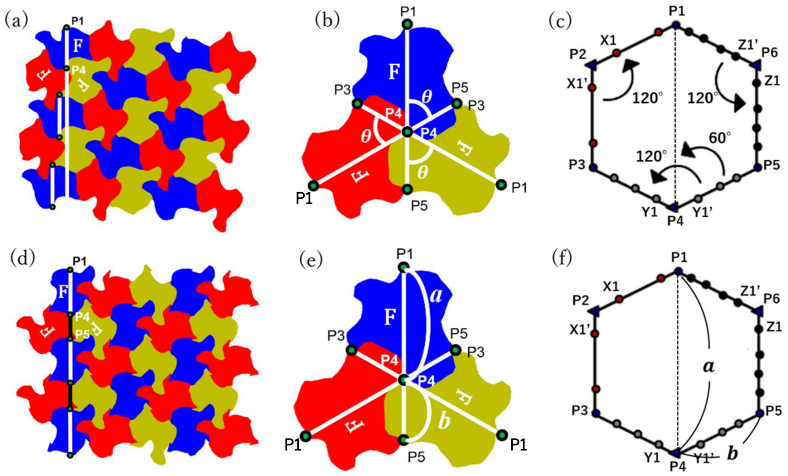


Fig. 4: Boundary constraints

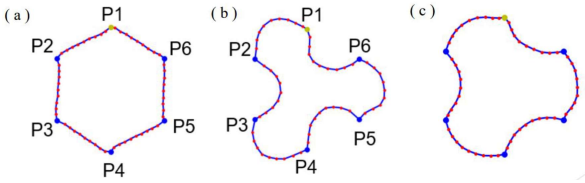


Fig. 5: Intermediate figure

input figures (a) and (b). In this study, two intermediate figures U3 and U4 (Fig. 3) are considered, and the weight ratio of input figures 1 and 2 is set to 3:1 for U3 and 1:3 for U4. Before adding the coordinate values, the size, the orientation, and the number of points on each edge of the two figures are aligned. In Fig. 5, the size means the length of P_1P_4, and the orientation means the angle of $\overrightarrow{P_4P_1}$. Compare the number of points on corresponding edges in two figures, and adjust the smaller number to match the lager number using spline interpolation. When the number of points on an edge is changed, the number of points on the corresponding edge in the input figure is also changed to maintain the correspondence between the tile figure and the input figure.

3.5 Boundary figures

Three types of boundary figures UA, UB, and UC are considered as shown in Fig. 6(a). UA has the point P1 on the top, and UB (resp. UC) has the point P1 on the left (resp. right). Fig. 6(d) shows UA, which is obtained by joining the left half of the tile (b) and the right half of the tile (c). To get the boundary figures UB and UC, first rotate the figure (b) 120 degrees in counter-clockwise, and (c) in clockwise to locate P1 in the corresponding place as in Fig. 6(e) and (f). Then, connect the lines between P3 and P4 in Fig. 6(e) and P4 and P5 in Fig. 6(f), replace them with the intermediate lines

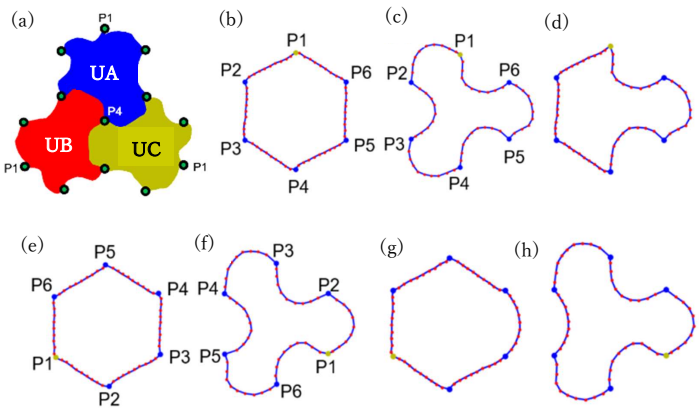


Fig. 6: Boundary figures

obtained by adding the coordinates of the two lines and dividing the sum by 2. The resulting boundary figures UB and UC are shown in Fig. 6(g) and (h), respectively.

3.6 Generation of tile images

Next we present a method for fitting the objects in the input image to the shape of tiles. An input image is represented as a polygon using contour points that are extracted when computing tile images from input images. To transform an input shape to another shape, the polygon of the object in an input image is divided into triangles. Applying the same division to tile images, triangles of an input object are related to those of a tile image. Then performing an affine transformation between the corresponding triangles, we can convert triangles of an input object into those of a tile image. Applying this transformation to all triangles, an input image is transformed to a tile image. We use two methods for triangle division: *centroid triangulation* and *Delaunay triangulation*. Centroid triangulation divides a polygon into triangles by connecting the centroid of a polygon and two adjacent points on the boundary. Delaunay triangulation divides a polygon in a way that the circumcircle of each triangle does not contain any other points and the resulting triangles are as close to equilateral triangles as possible.

Algorithm 1 sketches a procedure for generating Metamorphosis.

4 Experimental Results

4.1 Comparing the effects of triangulation

We use the Jupyter Notebook, a web-based interactive computing platform, for implementing the system. The triangle library is used for Delaunay triangulation, and OpenCV libraries 'cv2.getAffineTransform' and 'cv2.warpAffine' are used for affine transformation. First, we examine the results of centroid triangulation. In Fig. 7, two different types of input images are shown. We use a lemon as an example of a simple figure, and a doll as an example of a complex figure. Fig. 7(a) and (d) represent

Algorithm 1 Automatic Generation of Metamorphosis

Input: Input Images I_1 and I_2; number n of points
Output: Metamorphosis-style image MI

1: Extract the contours of I_1 and I_2 using n points, and represent them by matrices W_1 and W_2, respectively.
2: Find the kernel $KerA$ of A where A is a coefficient matrix representing constraints of IH7.
3: Compute the orthogonal bases B of $KerA$.
4: **for** $i = 1, 2$ **do**
5: **for** all the arrangement of reference points in W_i **do**
6: **for** all the arrangement of the start point in W_i **do**
7: Update $max(\frac{|B^\top W_i|^2}{|W_i|^2}) \rightarrow val_i$
8: **end for**
9: **end for**
10: Using W_i that constitutes val_i, put $U_i := BB^\top W_i$.
11: **end for**
12: Compute intermediate figures: $U_3 := (3 * U_1 + U_2)/4$ and $U_4 := (U_1 + 3 * U_2)/4$.
13: Compute boundary figures UA_j, UB_j, UC_j $(j = 1, 2, 3)$.
14: Transform an input image to a tile image using triangulation and affine transformation.
15: Embed those images into intermediate figures and boundary figures.
16: Construct MI using input images, intermediate images, and boundary images.

the results of centroid triangulation of input images. Fig. 7(b) and (e) represent tile images obtained by the input images. Comparing the results, centroid triangulation is done without overlapping in (b), which results in an almost successful output (c). By contrast, centroid triangulation is overlapping in (e), then the object protrudes beyond the contour in the output in (f).

Next, we examine the results of Delaunay triangulation in Fig. 8. When the number of vertices increases in a lemon, parts of the vertices disappear on the edge (blue areas) of (a) and (b). As a result, the output tile image has points that protrude beyond the contour as in (c). By contrast, in a doll non-overlapping divisions are made as in (d) and (e), which results in an almost successful output (f). In this way, two triangulation methods are effectively used depending on an input shape and its number of vertices.

4.2 Generating Metamorphosis

Fig. 9 shows some Metamorphoses generated by the system. Fig. 9(a) is generated from input images of a hexagon and a human figure using Delaunay triangulation with 120 vertices. Fig. 9(b) is generated from input images of a lemon and a banana using Delaunay triangulation with 120 vertices. Comparing the two images, (a) shows natural transmutation of two input images without noticeable gaps or overlapping, while (b) has some unnatural sharp edges or unnecessary gaps. Fig. 9(c) is generated from input images of blueberries and bananas using centroid triangulation with 36 vertices. In this example, the two input images have very different colors, which results in unnatural color change in the output image. Since the color and pattern of tile images are ad-

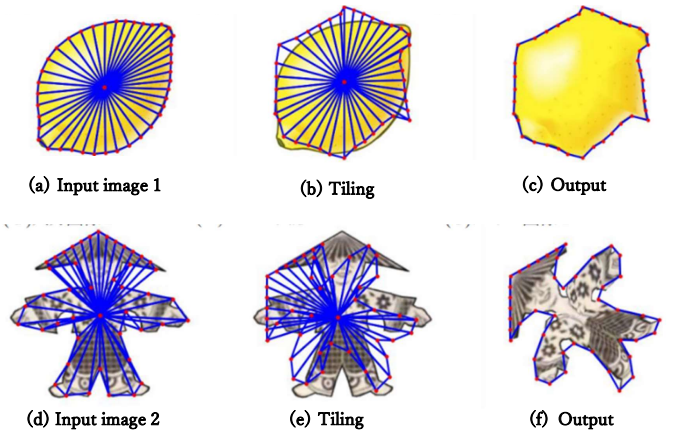


Fig. 7: Examples of centroid triangulation

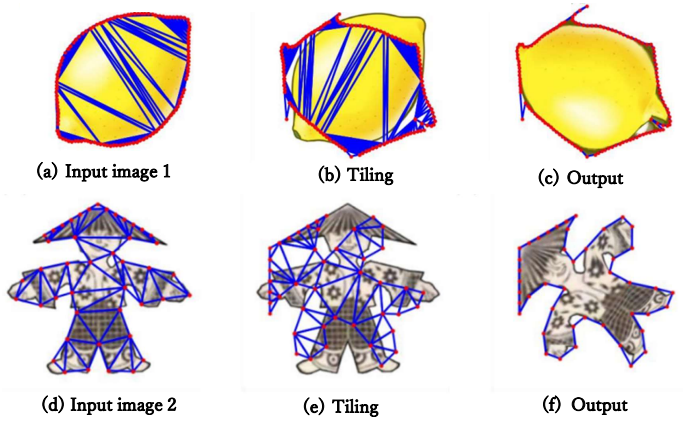
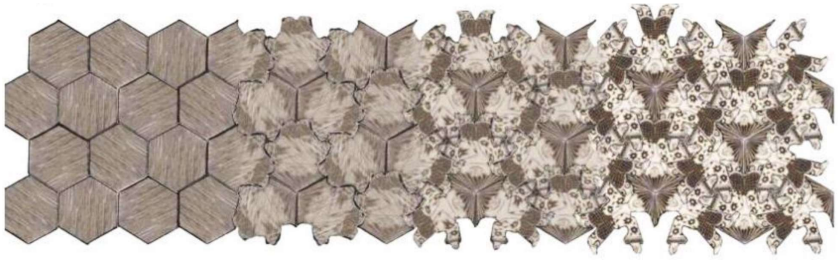


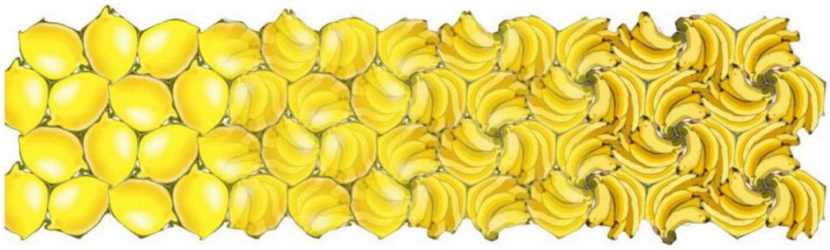
Fig. 8: Examples of Delaunay triangulation

justed at their boundary, such unnatural color changes occur at the boundary even if the number of intermediate shapes is increased.

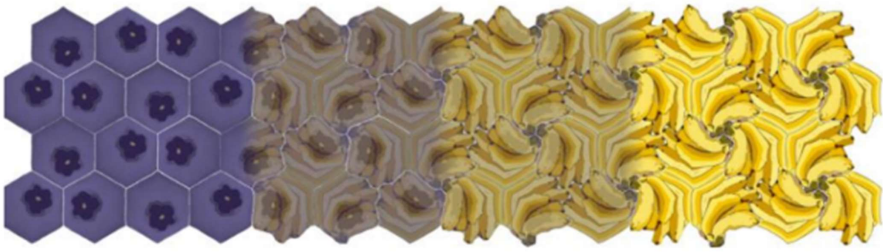
Fig. 10 shows our original Metamorphoses: (a) banana and yellow car, (b) blueberry and blue car, and (c) peony and rose. In Fig. 11 we use objects appearing in the art by Escher as input images: (a) hexagon and lizard, (b) hexagon and bird, and (c) hexagon and fish. All these figures use Delaunay triangulation with 36 vertices. In contrast to Fig. 9(c), two input images have similar colors then the one smoothly transforms to the other. When we use simple input images such as blueberries or hexagons, changes in pattern or color are clearly visible within the object. On the other hand, when complex input images such as bananas, cars, or flowers are used, changes in pattern or color are less noticeable, resulting in more natural transmutation. In particular, with input images that have similar patterns like in Fig. 11(c), the boundary of the change is hardly noticeable.



(a) Hexagon and Human



(b) Lemon and Banana

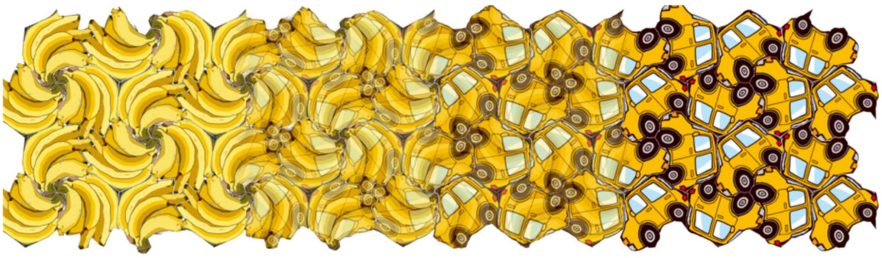


(c) Blueberry and Banana

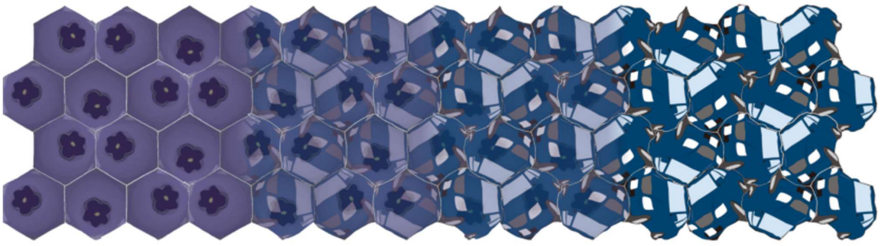
Fig. 9: Examples of successful and unsuccessful Metamorphosis

5 Conclusion

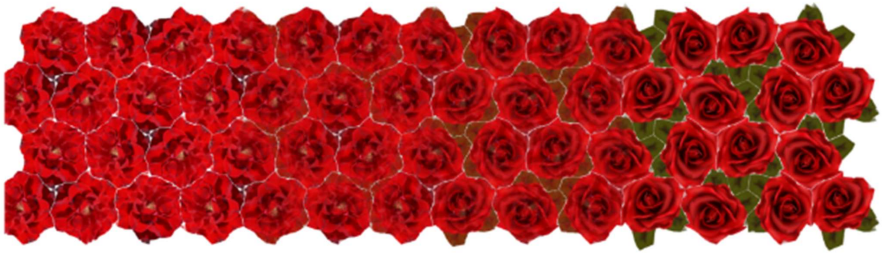
In this study, we built a system that automatically generates morphing images capturing the characteristics of Metamorphosis by applying the Escher-style tiling problem and continuously changing the shape and pattern of tiling. Experimental results show that the system successfully reproduces part of Escher's Metamorphosis and produces original Metamorphosis. The result would contributes to applications in tiling and graphic design. In the current system, the quality of an output image depends on the number of points on the contour and the selection of triangulation. In this regard, machine learning techniques could be used for exploring appropriate settings of these parameters. There is also room for improvement by using more versatile partitioning and modifying constraints so that the result is independent of the number of points. These improvements remain as future challenges.



(a) Banana and Yellow Car



(b) Blueberry and Blue Car

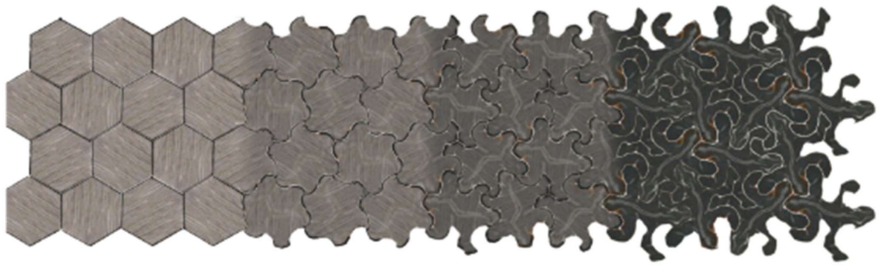


(c) Peony and Rose

Fig. 10: Original Metamorphoses

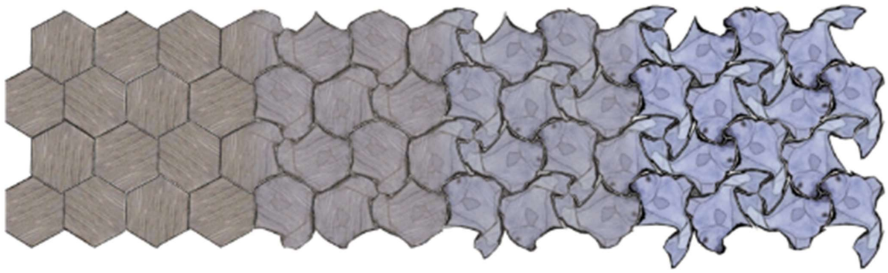
References

1. M. C. Escher – The official website, (accessed June 2024), https://mcescher.com
2. C. S. Kaplan and D. H. Salesin. Escherization. In: Proc. 27th Annual Conf. Computer Graphics and Interactive Techniques, pp. 499–510 (2000)
3. Tom McLean: Isohedral Tilings, jaapsch.net, (accessed June 2024), https://www.jaapsch.net/tilings/mclean/index.html
4. H. Koizumi and K. Sugihara. Maximum eigenvalue problem for Escherization. Graphs and Combinatorics 27(3):431–439 (2011)
5. M. Werman and D. Weinshall. Similarity and affine invariant distance between 2D point sets. IEEE Trans. Pattern Analysis and Machine Intelligence 17:810–814 (1995)
6. S. Imahori and S. Sakai. A local-search based algorithm for the Escherization problem. In: Proc. IEEE Int'l Conf. Industrial Engineering and Engineering Management, pp. 151–155 (2012)



(a) Hexagon and Lizard

(b) Hexagon and Bird



(c) Hexagon and Fish

Fig. 11: Metamorphoses using objects in Escher's art

7. Y. Nagata and S. Imahori. An efficient exhaustive search algorithm for the Escherization problem. Algorithmica 82(3):2502–2534 (2020)
8. S. Ono, M. Kisanuki, H. Machii, K. Mizuno. Figure pattern creation support for Escher-like tiling by interactive genetic algorithms. In: Proc. 18th Asia Pacific Symp. Intelligent and Evolutionary Systems, vol.1, pp. 421–432 (2015)
9. K. Sugihara. Computer-aided generation of Escher-like Sky and Water tiling patterns. Journal of Mathematics and Arts 3(4):195–207 (2009)
10. S.-S. Lin S, C. C. Morace, C.-H. Lin, L.-F. Hsu, T. Y. Lee. Generation of Escher arts with dual perception. IEEE Trans. Visualization and Computer Graphics 24(2):1103–1113 (2018)
11. X. Liu, X. Li, C. Zhang, C. Tang, X. Zhang, C. Zheng, Y. Tao, G. Wang, W. Xu, C. Yao, F. Ying. The artwork generating system of Escher-like positive and negative pattern evolution. In: Proc. ACM SIGGRAPH, poster (2017)
12. X. Liu, L. Lu, A. Sharf, X. Yan, D. Lischinski, C. Tu. Fabricable dihedral Escher tessellations. Computer-Aided Design 127 (2020) 102853.