


Article

Stripe-Based Futurism-like Image—A New Type of Art and Its Application to Covert Communication

Shan-Chun Liu ¹, Da-Chun Wu ^{2,*}  and Wen-Hsiang Tsai ^{1,3}¹ Department of Computer Science, National Chiao Tung University, Hsinchu 300093, Taiwan² Department of Computer and Communication Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 824005, Taiwan³ Department of Information Communication, Asia University, Taichung 413305, Taiwan

* Correspondence: dcwu@nkust.edu.tw

Abstract: The generation of art images using computers has been studied recently. A new type of art image called a stripe-based Futurism-like image, and its application to covert communication, are proposed. Firstly, a source image is segmented into regions. The region corners are extracted by chain code analysis, and polygon approximation is conducted accordingly to simplify the region shapes. Then, each region is partitioned into stripes according to the region direction and the stripes are filled with the region's average color, or the color white alternatively, resulting in an image with the abstraction and dynamism of Futurism. Furthermore, a data hiding method is proposed to embed a secret message into the created art image by filling the stripes with the aforementioned colors in a message-dependent fashion, with the message bits being adjusted beforehand by a 2-to-3 mapping scheme to avoid filling a region's stripes with consecutive identical colors. Furthermore, hidden data security is enhanced by randomizing both the message-bit embedding order and the region processing sequence via the use of a secret key. By processing a database of 100 images, good experimental results of art image creation and secure message hiding have been obtained, showing the feasibility of the proposed methods for covert communication.

Keywords: computer art; data hiding; stripe-based Futurism-like image; covert communication



Citation: Liu, S.-C.; Wu, D.-C.; Tsai, W.-H. Stripe-Based Futurism-like Image—A New Type of Art and Its Application to Covert Communication. *Appl. Sci.* **2022**, *12*, 10422. <https://doi.org/10.3390/app122010422>

Academic Editors: Xin Zhong and Frank Y. Shih

Received: 14 August 2022

Accepted: 11 October 2022

Published: 15 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the history of Western art [1], different periods and thoughts lead to the creations of different art schools. In addition, many methods that use computer programs for the automatic creation of art images have been proposed in the past decades [2–15]. The aim of these methods is to make an artificially generated image look like a traditional artistic painting. For example, Hertzmann [2] created a computer art image that looks like a real oil painting with brush strokes, as shown in Figure 1. In Figure 2, three other types of computer art images are shown, where Figure 2a is a stained-glass-like image created by an algorithm presented by Mould [3], Figure 2b is a tile-mosaic-like image created by Hausner [4], and Figure 2c is a rectangle-based Neo-Plasticism-like image created by Liu et al. [5].

In this study, by using a computer, it was tried to create another type of painting art called *Futurism*, which is characterized by breaking an image down by painting in *stippled stripes and dots* to emphasize the dynamism of the subject matter [16]. Specifically, a new type of art image, called a *stripe-based Futurism-like image*, was generated automatically by a computer in this study. As illustrations, four examples of art images in the Futurism style are shown in Figure 3, and two examples of the stripe-based Futurism-like images automatically generated by the method proposed in this study are shown in Figure 4.

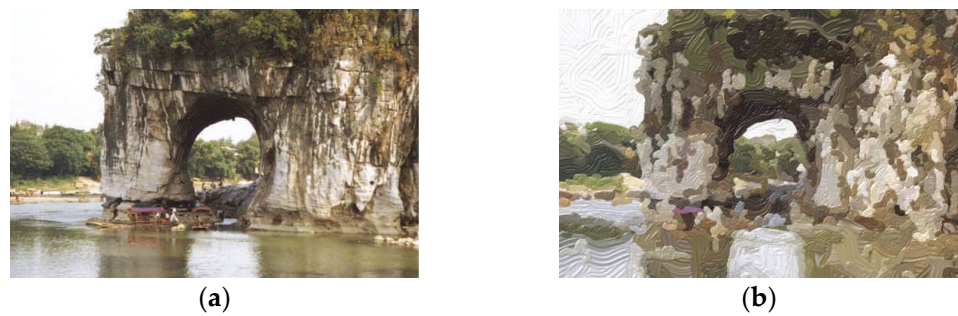


Figure 1. A computer-generated oil-painting art image created by Hertzmann [2]. (a) The original input image. (b) The created art image has an effect of oil painting with brush strokes.

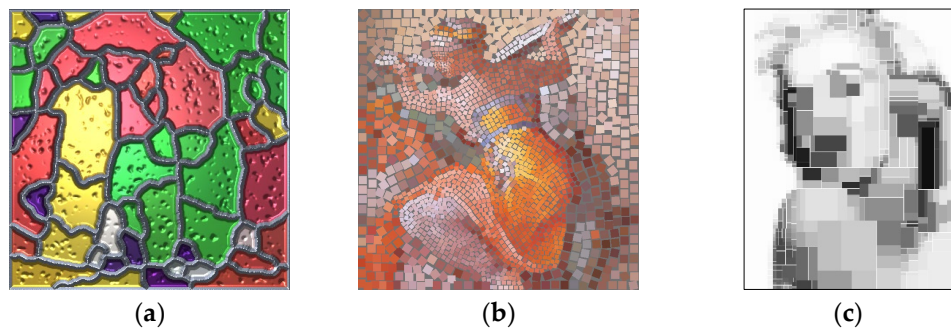


Figure 2. Three types of computer-generated art images. (a) A stained-glass-like image created by Mould [3]. (b) A tile-mosaic-like image created by Hausner [4]. (c) A rectangle-based Neo-Plasticism-like image created by Liu et al. [5].

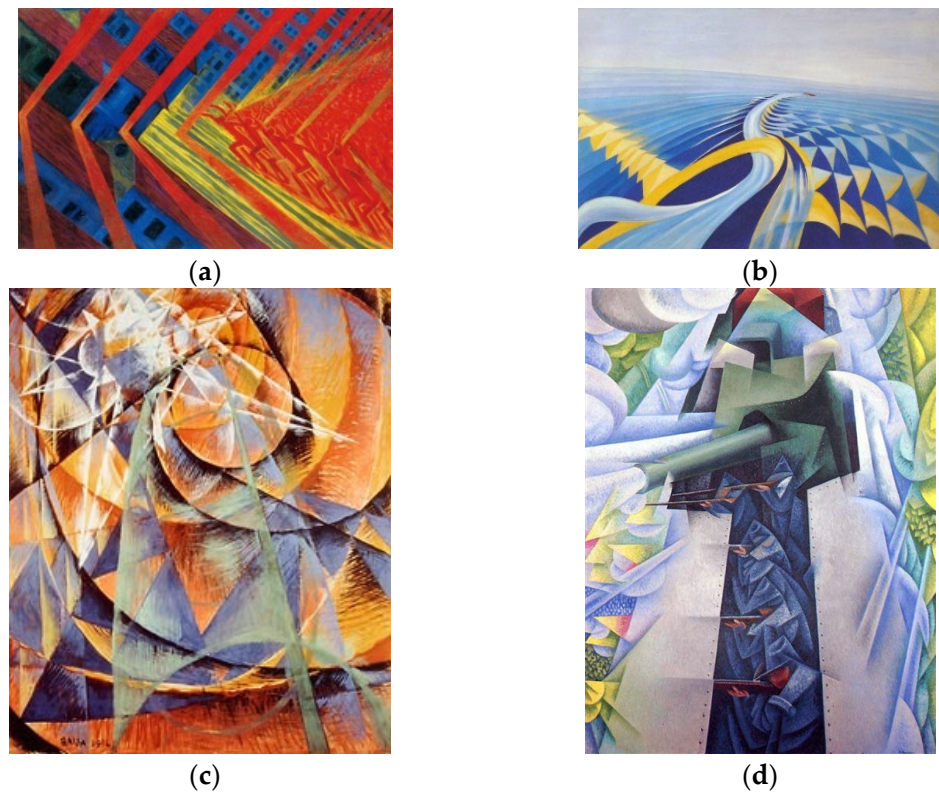


Figure 3. Four examples of Futurism art images. (a) “The Revolt,” by Luigi Rossolo, 1911. (b) “Speeding Motorboat,” by Benedetta Cappa, 1923. (c) “Mercury Passing Before the Sun,” by Giacomo Balla, 1914. (d) “Armored Train in Action”, by Gino Severini, 1915.

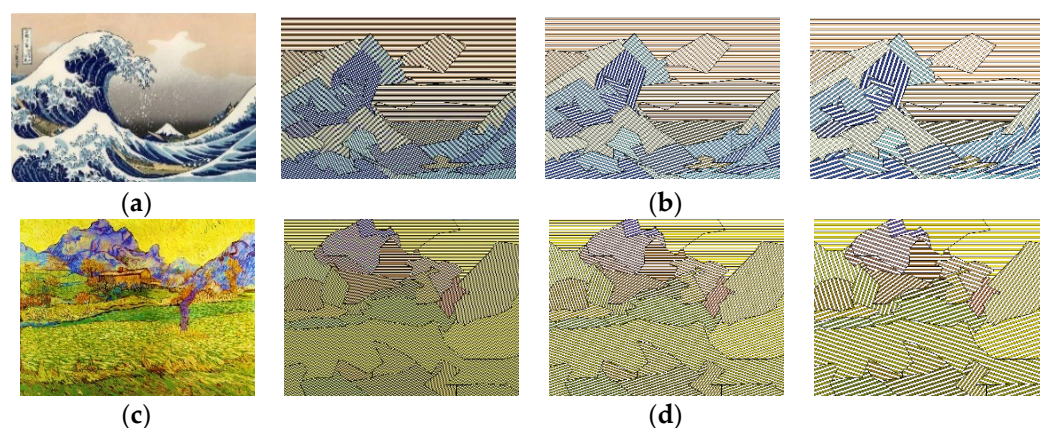


Figure 4. Stripe-based Futurism-like images generated in this study. (a) An original image, “The Great Wave off Kanagawa”, by Katsushika Hokusai, 1831. (b) Three stripe-based Futurism-like images with stripes of different widths generated in this study with Figure 4a as the input. (c) Another original image, “A Meadow in the Mountains”, by Vincent van Gogh, 1889. (d) Three stripe-based Futurism-like images with stripes of different widths generated in this study with Figure 4c as the input.

Covert communication is a way to embed messages imperceptibly into a cover medium. It not only hides a transmitted message but also conceals the existence of the message in the resulting stego-medium. The above-mentioned new type of computer art image, namely, the stripe-based Futurism-like image proposed in this study, can be created automatically as a cover image. *Making imperceptible the changes* in the stego-image which are caused by data embedding is a main goal in the field of data hiding or steganography [17–23]. It is known that the human visual system cannot distinguish subtle color differences in images. This weakness is often used to achieve the goal of imperceptibility of the data hiding technique in the resulting stego-image. In this regard, the optimal least-significant-bit (LSB) substitution method by Wang et al. [17], the pixel-value differencing (PVD) steganographic method by Wu and Tsai [18], and the histogram modification hiding method by Ni et al. [19], are three examples of data hiding techniques that can achieve this goal. More methods for data hiding and steganography can be found in Mandal et al. [24], which includes an extensive state-of-the-art paper review, analyses of recent relevant techniques, as well as detailed descriptions of popular steganography tools.

In this study, a new data hiding method is proposed to embed secret messages into stripe-based Futurism-like images created by the proposed art image generation method to achieve the goal of imperceptibility.

More specifically, the aim of the proposed image generation method is to pursue the sense of the *directional motion tendencies of objects through space* which is emphasized by the art school of Futurism. To create an image of this type, first the prominent regions in a given image are segmented by a scheme of region growing and merging [25,26]. Then, some regional features, such as corner point and region direction, are extracted by analyzing the chain codes of the boundaries of the segmented regions. By using these features, the direction of each region in the image is determined. Finally, each region is partitioned into stripes according to the extracted direction of the region; by using the stripes of different directions in each region, a type of art, called *stripe-based Futurism-like art image*, can be created with the regions in the image looking more vivid, satisfying the type of motion pursued by Futurist painters.

In addition, the proposed data hiding technique for embedding secret messages into created Futurism-like art images is conducted during the above-mentioned process of creating the stripes in the art image. Specifically, after partitioning each region into several stripes, a given secret message is hidden by coloring the stripes of the regions

with the region's average color, or the color white *randomly*. Moreover, an additional scheme is proposed to enhance the security of the hidden data by randomizing the order of embedding the bit data, as well as processing the regions and in the secret message. Being fascinated by the art displayed by the created image, a hacker hopefully will not notice the hidden message in the stego-image.

In the remainder of this paper, the proposed method for creating stripe-based Futurism-like art images is presented in Section 2. In Section 3, the proposed data hiding method, including a data embedding algorithm and a data extraction algorithm, are described. In Section 4, experimental results demonstrating the feasibility of the proposed methods, and comparisons of the techniques and results with those yielded by existing methods, are included. Discussions are given in Section 5, followed by concluding remarks in Section 6.

2. Creation of Stripe-Based Futurism-like Images

The Futurism art style aims to capture the dynamism and energy of the modern world by depicting the dynamic actions of speed and movement. To achieve this purpose, Futuristic painters developed various painting techniques based on divisionism, pointillism, cubism, etc. [16]. Besides adopting the techniques of blurring and repetition, artists of this school also use lines or stripes of force to convey the directional motion tendencies of objects. These spirits of Futurism result in paintings that appear to be fragmented and abstracted, stimulating the idea of creating the stripe-based Futurism-like art image developed in this study. In the remainder of this section, this idea is described at first in Section 2.1, and the details of the proposed method for the creation of stripe-based Futurism-like art images are described as an algorithm in Section 2.2.

2.1. Principle behind the Proposed Art Image Creation Method

Following the spirit of abstraction pursued by Futuristic painters, an input image is transformed automatically by the proposed method into a stripe-based Futurism-like art image as illustrated in Figure 5a by a process that includes five major steps: (1) segmentation of the image into regions; (2) polygon approximation of the regions; (3) computing the region directions and other parameters; (4) generating stripes in the regions according to the respective region directions; and (5) coloring the stripes, as illustrated by the flowchart shown in Figure 5b.

As a result, the image regions are filled with stripes of different directions in different colors. When observed as a whole, these regions look like an aesthetic clash that arouses the observer's vivid feeling of the original image content and shows the spirit of movement and dynamics of the Futurism art. Furthermore, different levels of abstraction can be created in the resulting image by setting the widths of the generated stripes to be of different values during the art image creation process. The created images are therefore called *stripe-based Futurism-like art images* as mentioned before.

Two examples of such art images obtained in the experiments conducted in this study are shown in Figure 4, where the three images in Figures 4b and 4d are the created art images with varying stripe widths. As can be seen, as the width becomes larger and larger, the abstraction seen in the resulting image becomes higher and higher.

The above-mentioned five major art image creation steps, as illustrated in Figure 5b, are elaborated in the following sections, with the first step described in Section 2.1.1 as image segmentation into regions, the second step in Section 2.1.2 as polygon proximation for region shape adjustment, and the remaining three steps in Section 2.1.3 as region direction finding and stripe creation.

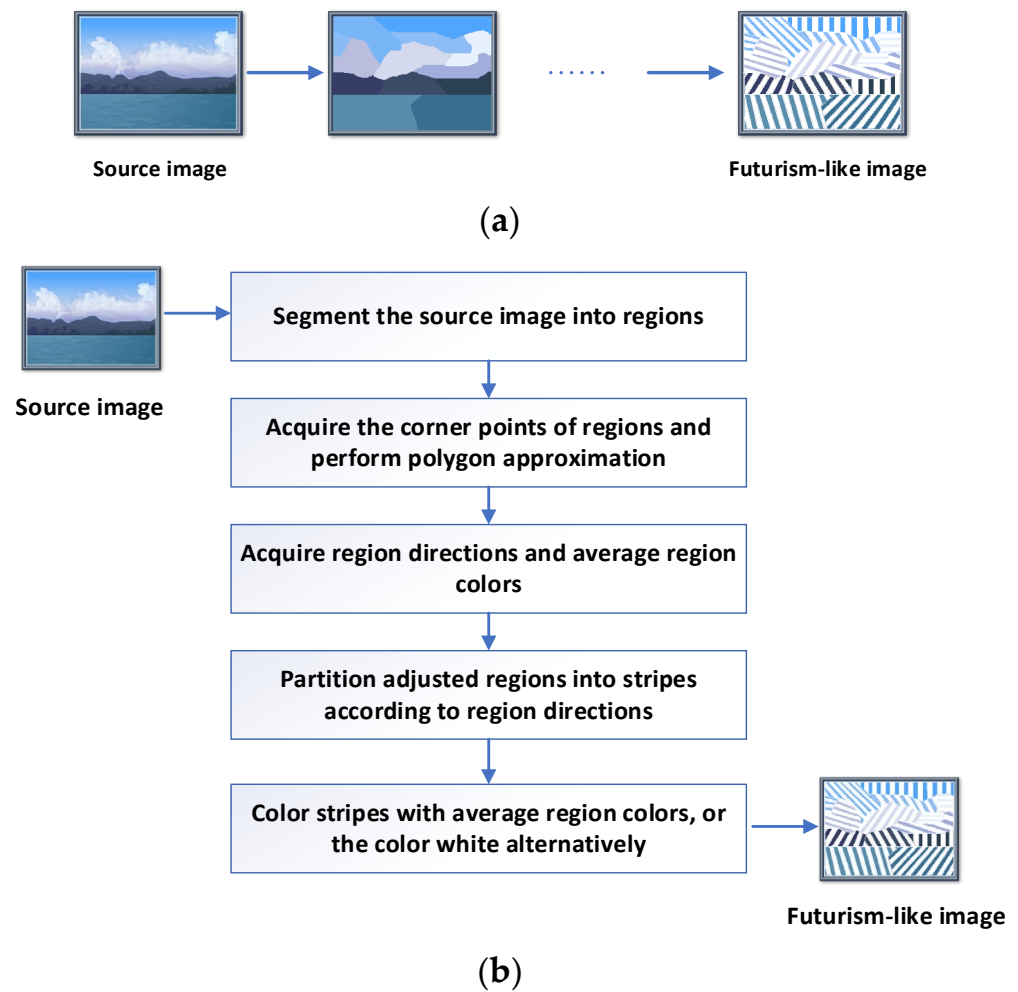


Figure 5. An illustration of the proposed method for stripe-based Futurism-like art image creation. (a) Basic idea. (b) A flowchart of the image creation process.

2.1.1. Image Segmentation into Regions

In the proposed art image creation process, first a source image is segmented into regions by region growing and merging techniques. This step starts from dividing the image into blocks and merging those neighboring blocks with similar colors. To save time for comparing the colors of image blocks in this step, color similarity was measured in this study in terms of the following 1-D color h , obtained from transforming the original 3-D RGB color (r, g, b) of the source image according to the following equation:

$$h(r, g, b) = b + 8 \times r + 64 \times g. \quad (1)$$

That is, two neighboring image blocks are regarded as similar in color if their average 1-D colors computed as above are close in value within a tolerant threshold. Note that in Equation (1) above, the way of coefficient assignment is based on the fact that while observing a color image, the human eye is the least sensitive to the blue color and the most sensitive to the green one, leading to an emphasis on the *intensity* of the image, as mentioned in [12].

Next, the erosion and dilation operations [27] are applied to the resulting blocks to eliminate isolated noise blocks, as illustrated by the example shown in Figure 6. Subsequently, the resulting smaller neighboring regions are merged into larger ones according to a region-area threshold value. Finally, the shapes of the resulting regions are then simplified by a polygon approximation scheme using chain codes as described in the next section.

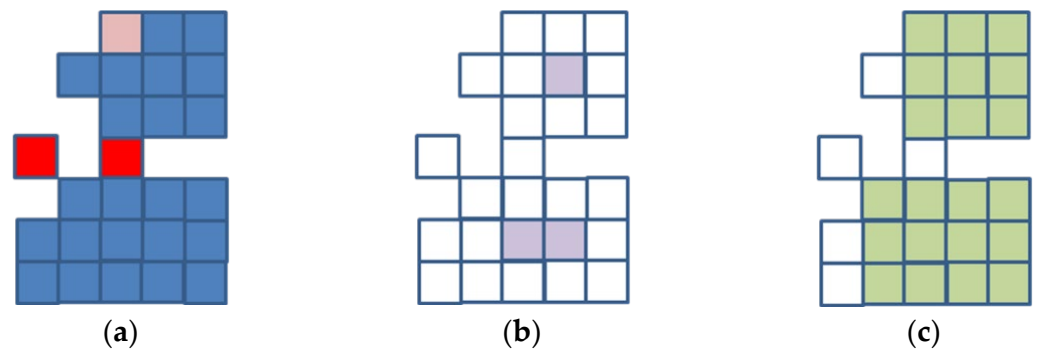


Figure 6. An example of deleting isolated points using chain codes. (a) The input. The pink grid is the start pixel of chain code tracing. (b) An erosion result of (a). (c) A dilation result of (b).

2.1.2. Polygon Approximation for Region Shape Adjustment

After the regions are obtained as described previously, a polygon approximation scheme is performed to make the region contours simpler to show more abstraction. The approximation is accomplished by finding the corner points along each region contour and connecting every two neighboring corner points by a line segment. For this aim, the chain codes as shown in Figure 7b were adopted in this study to describe region contours at first, which is a minorly modified version of the original Freeman codes [28] shown in Figure 7a. Additionally, for the convenience of computation, the upper-left point of the source image was taken to be the origin point (0, 0) for chain code tracing in this study instead of taking the lower-left point as in [28].



Figure 7. An illustration of chain codes. (a) The eight directions given by Freeman chain codes [28]. (b) The new eight directions used in this study.

In detail, the chain codes of the contour of each region were extracted in this study to represent the region shape. Furthermore, an expression (D_i, L_i) was utilized to efficiently denote a segment of chain codes which contains a maximal subsequence of identical codes, called *coherent chain codes*, where D_i is the value of the direction of the coherent chain codes in the i th segment and L_i is the number of such codes. For example, a given chain code sequence {1100000222235555550777} can be represented as a sequence of *coherent chain code segments* $\{(1, 2), (0, 5), (2, 4), (3, 1), (5, 7), (0, 1), (7, 3)\}$.

With the coherent chain codes defined as above, the chain codes of each region can be analyzed to obtain the corner points of the region as carried out in this study. For this purpose, an even simpler shape representation scheme proposed by Sanchez-Cruz [29], in which only three symbols $C = \{0, 1, 2\}$ are used, was adopted in this study further to represent the region shape to save storage more efficiently. The first symbol, 0, means that the direction is the same as the previous segment, called the *reference segment*; the second, 1, indicates a *direction change* upward with regard to the reference segment; and the last, 2, indicates a *backward direction change* with respect to the reference segment. To facilitate detecting the corner points in a shape contour via chain codes, a different set of symbols $C = \{-1, 0, +1\}$ was used in this study for shape representation, with the

three symbols -1 , 0 , and $+1$ corresponding to the three symbols 2 , 0 , and 1 in the set $C = \{0, 1, 2\}$ mentioned above, i.e., -1 means a backward direction change, $+1$ means a forward direction change, and 0 means no change in direction. The symbols -1 , 0 , and $+1$ will be called *direction-change codes* (abbreviated as *DCC*) subsequently. Finally, the DCCs are computed for the X- and Y-coordinate directions (i.e., for the horizontal and vertical directions), respectively, and they are called *horizontal DCC* and *vertical DCC*, respectively.

For the chain codes shown in Figure 7b, the corresponding horizontal and vertical DCCs for the X- and Y-coordinate directions are shown in Table 1. For example, when the direction of the chain code is 0, as specified in the leftmost column of the first non-title row, it means that the tracing of the shape contour goes from one contour point horizontally to another point on the right side, so that the X coordinate of the second contour point increases, meaning that the change of X coordinates is positive and so the horizontal DCC is $+1$. Additionally, since there is no change of Y coordinates in this case, the vertical DCC is 0 . The other rows can be interpreted similarly.

Table 1. Directions of chain codes and corresponding horizontal and vertical DCCs in the X- and Y-coordinate directions (DCC means direction-change code).

Direction	Horizontal DCC (for the X Coordinate)	Vertical DCC (for the Y Coordinate)
0	$+1$	0
1	$+1$	$+1$
2	0	$+1$
3	-1	$+1$
4	-1	0
5	-1	-1
6	0	-1
7	$+1$	-1

As an illustrative example, which is shown in Figure 8, for a sequence of chain codes “012222335567676” of the contour of a sample region in a source image, as shown in Figure 8a, if the pink contour point nearest to the origin of the image is taken to be the start point of chain-code tracing, the corresponding sequence of coherent chain-code segments are $\{(0, 1), (1, 1), (2, 4), (3, 2), (5, 2), (6, 1), (7, 1), (6, 1), (7, 1), (6, 1)\}$. Additionally, the sequences of the horizontal and vertical DCCs can, respectively, be calculated to be:

$$S_h = +1, +1, 0, 0, 0, 0, -1, -1, -1, -1, 0, +1, 0, +1, 0;$$

$$S_v = 0, +1, +1, +1, +1, +1, +1, +1, -1, -1, -1, -1, +1, -1, -1, -1,$$

as illustrated in Figure 8b,c, respectively, where the symbols $+$ and $-$ are used to represent the symbols $+1$ and -1 , respectively.

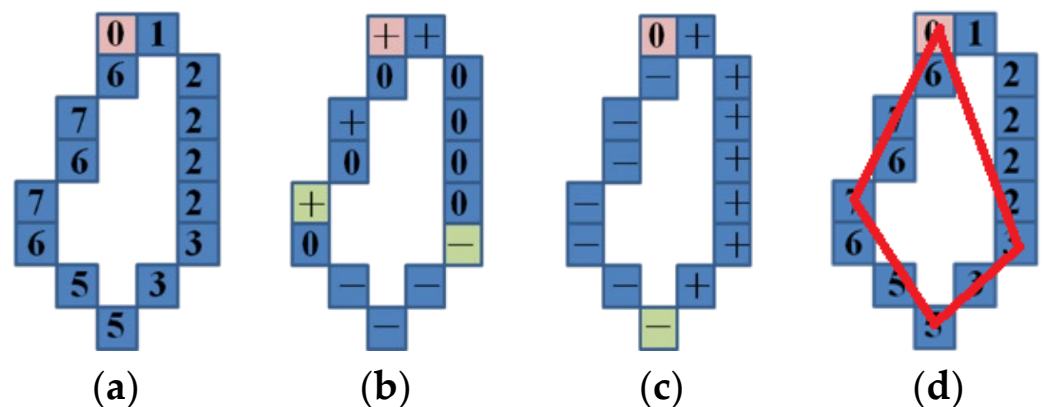


Figure 8. An example of finding corner points. (a) An example of chain codes. (b) The direction changes in X coordinate. (c) The direction changes in Y coordinate. (d) The result of connecting all corner points.

With the horizontal and vertical DCCs defined as above, it is difficult to see that a direction change in a chain code sequence will appear when the horizontal or vertical DCC changes from +1 to −1, or from −1 to +1, during contour-chain-code tracing. Accordingly, corner points in the region shape contour may be detected at the positions of the contour points *where changes of the horizontal or vertical DCCs occur*, as can be deduced. In this way, corner points along the contour of each region can be found, and by connecting these corner points by line segments, the polygon approximation of the contour shape is completed.

Continuing the illustration example of Figure 8, and following the above contour corner-finding rule, one can see from Figure 8b that two changes of the horizontal DCCs appear at the two corners, shown by the two green blocks in Figure 8b. Similarly, one can see that one change of the vertical DCCs appears at the one corner shown by the green block in Figure 8c. As the start point is also a corner point, there are four corner points in total along the shape contour, and all of these corner points can be connected in an orderly fashion to obtain a new polygonal shape, as shown in Figure 8d.

Two examples of the results of the steps of image segmentation into regions and polygon approximation for region shape adjustment are shown in Figure 9.

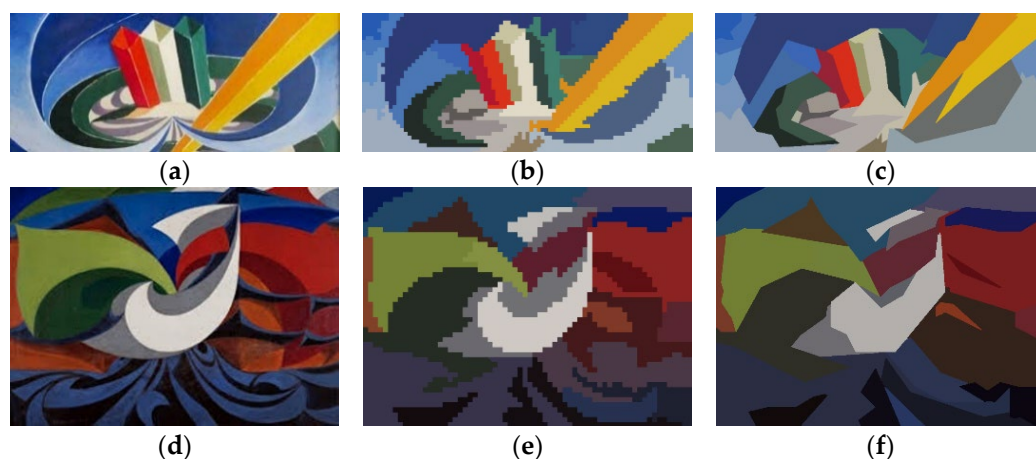


Figure 9. Two examples of image segmentation and region shape adjustment results. (a) An original image, “Canto patriottico-Piazza Siena”, by Giacomo Balla, 1915. (b) The result of image segmentation by region growing and merging of (a). (c) The result of region shape adjustment by polygon approximation of (b). (d) Another original image, “Forme grido Viva l’Italia,” by Giacomo Balla, 1915. (e) The result of image segmentation by region growing and merging of (d). (f) The result of shape adjustment by polygon approximation of each region of (e).

2.1.3. Region Direction Finding and Stripe Creation

Finally, the corner points can be utilized to find the direction of each region. Specifically, each pair of corner points constitute a vector which has a length value and an angle value. These vectors were classified into 10 groups in this study, each group covering 1/10 of 180 degrees. The sum of the lengths of the vectors in each group is calculated separately. The direction of the group that has the maximum sum of length values is taken as the desired direction of the region. According to the angle value of the direction of the region so found, the region is partitioned by lines into multiple stripes with a pre-selected width. Finally, by coloring the stripes with the average color of the region, or the color white alternatively, a stripe-based Futurism-like art image is created from the source image. Additionally, by varying the pre-selected width of the stripes, art images of different levels of abstraction of the original image can be created.

Two examples of the results of the above region direction finding and stripe-creation process with the images of Figure 9c,f as inputs are shown in Figure 10, where Figure 10a includes three abstraction levels of Figure 9a with stripe widths of 3, 6, and 9 pixels, respectively, and Figure 10b includes those of Figure 9d. It should also be mentioned that, due to the different resolutions of the displayed images, in certain cases undesired

moiré patterns will appear in the images, which actually do not appear in the original experimental results yielded in this study.

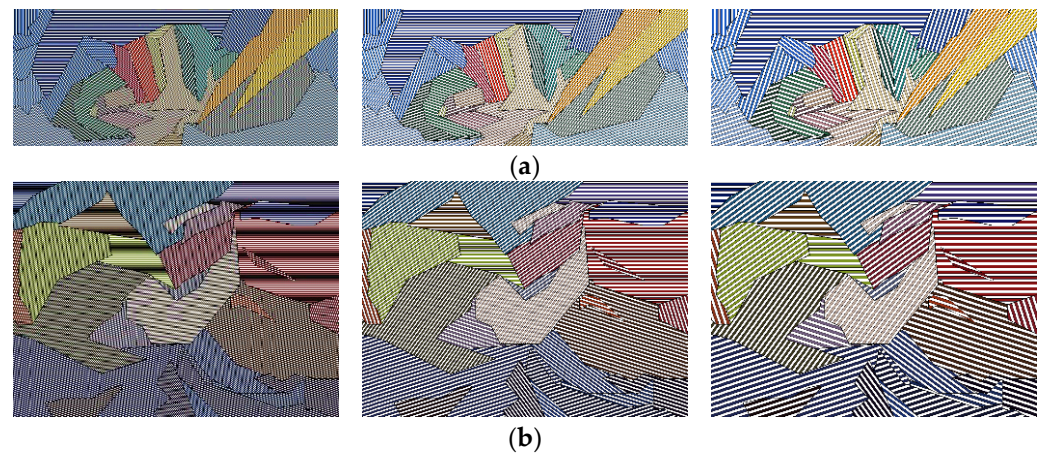


Figure 10. Two examples of the results of region direction finding and stripe creation. (a) Three results of different abstractions of the image by Giacomo Balla shown in Figure 9a. (b) Three results of different abstractions of the image by Giacomo Balla shown in Figure 9d.

2.2. Proposed Art Image Creation Process

The details of the process described in the previous section for creating a stripe-based Futurism-like art image from a given image are described as Algorithm 1 in the following.

Algorithm 1. Creation of a stripe-based Futurism-like art image.

Input: a source image S , the threshold T_m for controlling merging of image blocks with similar colors, the threshold A_m for controlling merging of neighboring image regions close by, and the value W_s for setting the stripe width.

Output: a stripe-based Futurism-like art image F .

Steps.

Stage 1—Finding the region direction.

Step 1. Compute the 1-D h -color of the RGB color values of each pixel in source image S according to Equation (1).

Step 2. Perform the following region growing and merging steps to segment S into regions R_1, R_2, \dots, R_n in S .

2.1 Divide S into 8×8 blocks.

2.2 Calculate the average h -color of each block.

2.3 Merge every two neighboring blocks into one if the difference between their average h -colors is smaller than the image-block merging threshold T_m .

2.4 Apply the erosion operation and then the dilation operation to the resulting blocks to delete isolated points as illustrated by Figure 6b,c.

2.5 Merge every two resulting regions into one if the Euclidean distance between them is smaller than the image-region merging threshold A_m .

Step 3. For each resulting region, set the point in its boundary nearest to the origin point in the upper-left corner of image S as its *start point* that will be used for extracting the chain codes of the region in the Step 5.

Step 4. Raster scan the entire source image S to access the start points of all the regions, rearrange the regions according to the raster-scan order of accessing the start points, and name the regions as R_1, R_2, \dots, R_n and their respective start points as $(S_{x1}, S_{y1}), (S_{x2}, S_{y2}), \dots, (S_{xn}, S_{yn})$.

-
- Step 5. Perform the following steps to find the coherent chain codes of each region $R_i, i = 1, 2, \dots, n$.
- 5.1 Trace the boundary of R_i to obtain the sequences S_1, S_2, \dots, S_k of the chain codes of R_i by the directions illustrated in Figure 7b until the tracing point reaches the start point (S_{xi}, S_{yi}) .
 - 5.2 Transform the sequence S_1, S_2, \dots, S_k into a sequence of coherent chain code segments $E_i = \{(D_1, L_1), (D_2, L_2), \dots, (D_m, L_m)\}$.
- Step 6. Perform the following steps to conduct polygon approximation using the sequence of coherent chain code segments of each region $R_i, i = 1, 2, \dots, n$.
- 6.1 Analyze the values of the chain codes of the sequence of coherent chain code segments $E_i = \{(D_1, L_1), (D_2, L_2), \dots, (D_m, L_m)\}$ of R_i to obtain the DCC (direction-change code) sequences X_1, X_2, \dots, X_m and Y_1, Y_2, \dots, Y_m of the X- and Y-coordinate directions, respectively, according to Table 1.
 - 6.2 Acquire the corner points in R_i by the following operations.
 - (a) Create a corner point sequence $P_i = \{(P_{xi1}, P_{yi1}), (P_{xi2}, P_{yi2}), \dots, (P_{xit}, P_{yit})\}$ to record to the positions of the corner points of R_i to be found.
 - (b) Assume that the start point (S_{xi}, S_{yi}) of R_i is the first corner point, i.e., the position (P_{xi1}, P_{yi1}) is (S_{xi}, S_{yi}) .
 - (c) Set a segment, for example the c -th segment, in E_i as a corner segment when one of the following conditions is met.
 - (c1) The length L_c in the X-coordinate direction of the chain code segment E_i is larger than a half of the region width or when the length L_c in the Y-coordinate direction of the chain code segment E_i is larger than a half of the region height.
 - (c2) The chain code value of the c -th segment, in the DCC sequence X_1, X_2, \dots, X_m or Y_1, Y_2, \dots, Y_m changes from -1 to $+1$ or from $+1$ to -1 .
 - (d) Obtain the position (P_{xij}, P_{yij}) of a corner point, say the j -th corner point, in the corner point sequence P_i from the c -th segment in E_i by the following method, assuming that each of the functions F_x and F_y computes, respectively, the changes of X and Y coordinates in direction according to Table 1, and that the previous corner segment is the b -th segment in E_i :

$$P_{xij} = P_{xi(j-1)} + \sum_{a=b+1}^c (F_x(D_a) \times L_a)$$

$$P_{yij} = P_{yi(j-1)} + \sum_{a=b+1}^c (F_y(D_a) \times L_a)$$
 - (e) Delete a corner point of two neighboring ones if the distance between the two is smaller than a quarter of the shorter of the region width and height.
 - 6.3 Connect the resulting corner points of each region R_i in order as the new edges of region R_i .
 - 6.4 Obtain new regions, called adjusted regions, from the new edges and denote them as R_1', R_2', \dots, R_k' by filling pixels into their interiors and deleting the original pixels, which belong to the original regions but outside the adjusted regions.
- Step 7. Merge the remaining regions, which are residues after the adjustment of the last step, into the nearest adjacent regions of R_1', R_2', \dots, R_k' according to the measure of distance between the average h -colors of the remaining regions and the adjusted regions.
- Step 8. Re-compute the chain codes and the corner points of R_1', R_2', \dots, R_k' by repeating Steps 3 through 6.
- Step 9. Perform the following steps to obtain the region direction of each adjusted region $R_i', (i = 1$ through $k)$ using their resulting corner points.
- 9.1 Acquire an edge from region R_i' with its length and angle to form an edge vector.
 - 9.2 Classify the edge vectors of R_i' into ten groups according to the angles of the edge vectors, each group covering the range of 18 degrees in orientation.
 - 9.3 Calculate the sum of the lengths of the vectors in each group of R_i' separately.
 - 9.4 Take the direction of the group with the maximum sum value as the desired direction of the region R_i' .
-

Stage 2—Generating the desired Futurism-like art image.

- Step 10. Partition each adjusted region R_i' , ($i = 1$ through k) into stripes according to the region direction of R_i' and the parameter of the stripe width W_s .
- Step 11. Compute the average region color C_1', C_2', \dots, C_k' of each adjusted region R_i' ($i = 1$ through k).
- Step 12. Draw the partitioning line between every two stripes with the color black.
- Step 13. Fill the stripes in each R_i' ($i = 1$ through k) with the average region color C_i' , or the color white alternatively.
- Step 14. Assign the color black to all the pixels of the boundary of each region R_i' described by its chain-code sequence.
- Step 15. Take the final image S as the desired stripe-based Futurism-like art image F .
-

In the above algorithm of stripe-based Futurism-like image creation, the focus is on the processes of polygon approximation for region shape adjustment and finding the region direction by use of the chain codes for stripe generation. By connecting the corner points yielded by analyzing the chain codes, a simpler shape of each region in the input image can be obtained. The effect of polygon approximation also shows the geometric style of the school of Futurism, as can be seen from the examples of experimental results shown in Figures 4 and 10. Furthermore, each image region is partitioned by its direction into stripes to pursue the motion style of Futurism. More experimental results will be given in Section 4.

3. Data Embedding and Extraction via Art Images

In this section, the idea of the proposed data hiding technique for covert communication via stripe-based Futurism-like art images is explained in Section 3.1. In Section 3.2, the algorithm of the proposed data hiding technique is describing in detail. Then, the algorithm of the proposed secret message extraction process is presented in Section 3.3. Finally, some security considerations of the proposed techniques are described in Section 3.4.

3.1. Idea of the Proposed Data Hiding Method

As described in Algorithm 1, to create a stripe-based Futurism-like art image, the stripes of each region in the image are filled with the region's average color, or the color white alternatively. However, like some Futurists' painting style of leaving some white spaces in their paintings, occasionally it is also possible to paint certain regions successively with non-white colors. This gives a hint for hiding messages in the proposed art image, i.e., taking the colors of the stripes in the art image to represent the message bits, for example, with a white stripe representing a bit of 0 and a non-white stripe representing a bit of 1. This idea was implemented in this study as the core steps of the proposed data hiding method. Specifically, the colors of the stripes were adopted to embed the bit sequence of a message, and the resulting art image, called *stego-image*, looks even more dynamic in appearance. People will tend to feel the stego-image as an art painting, being unaware of the existence of the embedded message. It is in this way that the goal of data hiding is achieved via the proposed stripe-based Futurism-like art image.

Two examples of the results of data hiding in stripe-based Futurism-like art images are shown in Figure 11, whose versions with no data embedded are shown in Figure 4. As can be seen from any of the six images shown in Figure 11b,d, the stripes in the image regions are not filled with two colors in a fixed alternative fashion, but in a rather random way.

More descriptions about the techniques used in the proposed data hiding process are included in the following sections.

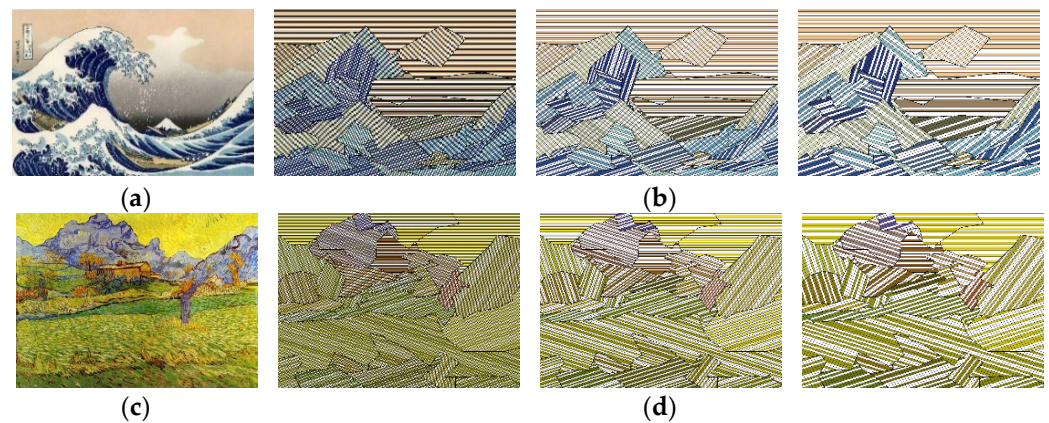


Figure 11. Results of data hiding in stripe-based Futurism-like images. (a) The original image, “The Great Wave off Kanagawa,” by Katsushika Hokusai, 1831. (b) Three stripe-based Futurism-like images with message data embedded with Figure 4a as the input. (c) The second original image, “A Meadow in the Mountains,” by Vincent van Gogh, 1889. (d) Three stripe-based Futurism-like images with message data embedded with Figure 4c as the input.

3.1.1. Adjustment of Message Bit Data before Embedding

As mentioned previously, to achieve covert communication, the stripes in each image region are colored to embed message data. In detail, instead of coloring the stripes in a fixed alternative order, the stripe colors controlled by the embedded message bits. It should be mentioned that in some extreme cases of the above-mentioned data hiding method, the stripes in a region or in a large part of the region may all be filled with an identical color (with the average region color, or the color white only) due to the embedding of a sequence of identical bit values (all 0s or all 1s), resulting in an art image that might be seen as strange by people. In order to avoid this situation, the secret message sequence is adjusted in advance by transforming every two bits into three before the message bits are embedded into the art image. As a result, the stego-image will not become the extreme case mentioned. The transformation is accomplished by the use of a table, as shown in Table 2, called *2-to-3 mapping table* subsequently.

Table 2. A mapping table that transforms two bits of the original message data into three bits.

Two Original Bits	Three New Bits
00	101
01	100
10	110
11	010
Ending pattern	011

For example, if a segment of a message to be hidden appears to be a sequence of eight 0s, then originally the corresponding stripe coloring result will be eight consecutive stripes all filled an identical color, namely, the color white, which integrally appears to be a larger white stripe and will possibly arouse the observer’s notice. Now, according to Table 2, the eight 0s will be mapped to be 101101101101 (i.e., every two 0s are mapped to be 101) which has segments of no more than two 1s with 0 inserted in between. Therefore, no large stripe of a single color will then appear.

3.1.2. Considerations of Embedded Data Security

Furthermore, to increase the security of the embedded data against secret hacking resulting from the proposed data hiding method, the order of listing the three-bit items in the second column in Table 2, which are mapped to the two-bit items in the first column, are not fixed but randomized by a secret control key, as implemented in this study. That is, for each different secret control key, the mappings in the table will also be different; Table 2 is

just a possible mapping set among many randomized ones. For example, another possible mapping set may be (00 \Rightarrow 100, 01 \Rightarrow 110, 10 \Rightarrow 010, 11 \Rightarrow 011, and ending pattern \Rightarrow 101) which is just a circular shift of the three-bit segments in the second column.

On the other hand, in order to recover the hidden secret message from a stego-image, the start point of each region has to be recorded as the beginning code of the chain-code sequence extracted from each region contour. Additionally, the next boundary points should be recorded in order, so as to avoid getting the wrong chain code sequence in the message extraction process. Finally, to facilitate judging the end of the data extraction process, a specific *end pattern* of three bits (as shown in the last row in Table 2) are inserted to the end of the message bit sequence before the data embedding process is started.

3.1.3. Major Steps of the Proposed Data Embedding Process

A flowchart of the proposed data embedding process is shown in Figure 12. The process essentially is based on the output of the stripe-based Futurism-like art image generation process described by Algorithm 1 and depicted in Figure 5.

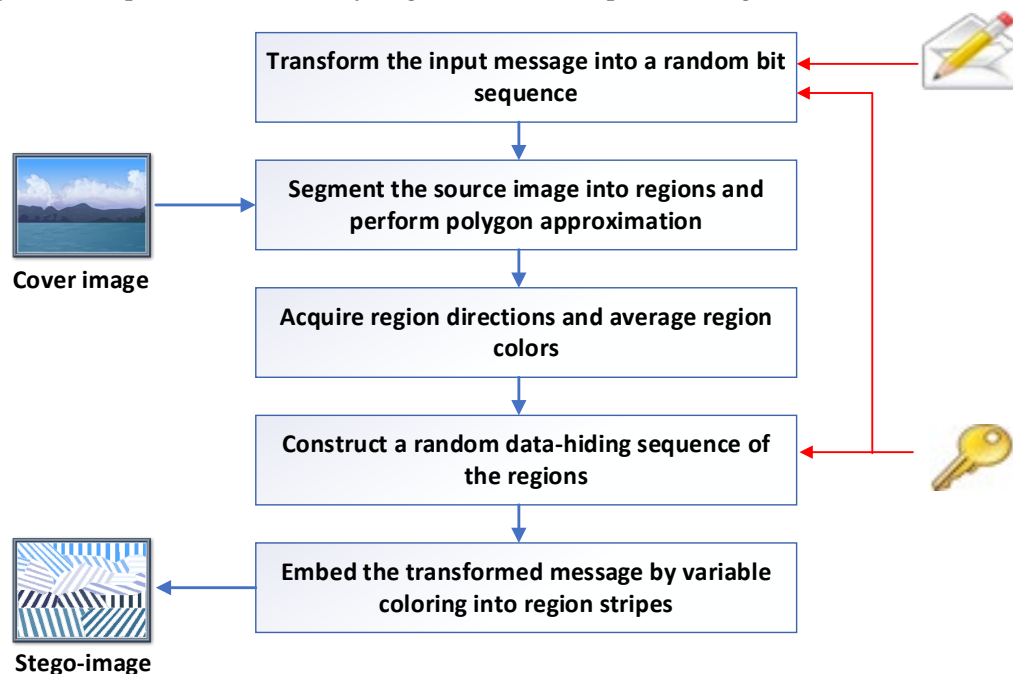


Figure 12. A flowchart of the proposed data embedding process.

Firstly, the input secret message is transformed into a random digit sequence using a secret key. Additionally, Algorithm 1 is applied to segment the input source image into regions which are adjusted next by polygon approximation. The direction and the average color of each adjusted region are obtained in the meantime. Then, a random-ordered data hiding sequence of the adjusted regions is constructed with a secret control key, which is followed finally to embed the transformed message in the form of a random digit sequence into the intact stripes of the regions by variable coloring of the stripes in each region. A stego-image is so generated which can be used for covert communication.

3.2. Proposed Data Hiding Process

A detailed algorithm describing the aforementioned details of the proposed data embedding process is presented in the following section. The algorithm is essentially based on the output data of Algorithm 1 with data embedding steps included additionally. However, different from Algorithm 1, which uses all RGB colors freely to create stripe-based Futurism-like art images, some special colors, namely, the black and nearly black

colors (0, 0, 0), (0, 1, 0), (0, 0, 1), (1, 0, 0) and the color white (255, 255, 255), are used to accomplish the tasks of drawing the inter-stripe partition lines, locating the start points and the next points of the region boundaries, and painting the inner part of the stripes. Such tasks are indispensable for the purpose of extracting the right information or hidden data successfully in the message data extraction process described in the next section. Therefore, the colors in the regions of the source image should be adjusted slightly in advance to avoid using these special colors in the data embedding process described in the Algorithm 2. Specifically, if the RGB color values (C_r, C_g, C_b) in the source image are found to be one of the above special colors, then the values 0 and 255 in them are reset to be 1 and 254, respectively. To the user, the slight color alternations in the image cause nearly no visual effect because the human vision system cannot detect such extremely little color differences.

Furthermore, when the message-bit sequence is not long enough to be hidden in all the image regions, then as described in the Algorithm 2, in those regions that are not used for data embedding, stripes are generated as well and filled with the average region color ($C_{rj}', C_{gj}', C_{bj}'$), or the color white (255, 255, 255) randomly, according to a table like Table 2, which is controlled by a secret key K . This scheme yields a stego-image with randomly colored region stripes, showing a uniform appearance of shape and color randomness in the stego-image which increases hidden data security and hopefully arouses no suspicion in observers or even hackers.

Algorithm 2. Embedding a secret message into a stripe-based Futurism-like art image.

Input: a cover stripe-based Futurism-like art image S , a secret key K , a secret message M , and a parameter value of the stripe width W_s .

Output: a stego-image I into which M is embedded.

Steps.

Stage 1—Randomizing the message bits and adjusting the colors of the source image.

- Step 1. Transform the secret message M , which has eight bits for each character, into a bit sequence M' , and randomize the bits in M' by the secret key K .
- Step 2. Create a 2-to-3 mapping table T like that shown in Table 2, with the listing order of the 3-bit items in the second column of the table being randomized by the secret key K as well.
- Step 3. Transform every two bits of M' into three bits according to the created mapping table T , resulting in a new sequence M'' .
- Step 4. Append the ending pattern in table T to the end of M'' to form a new bit sequence with a length of $(M' \times 3)/2 + 3$.
- Step 5. Divide M'' into a series of 3-bit segments m_1, m_2, \dots, m_n .
- Step 6. If the RGB color (C_r, C_g , and C_b) of any pixel in the cover image S is one of the black and nearly black colors (0, 0, 0), (0, 1, 0), (0, 0, 1), (1, 0, 0), or the color white (255, 255, 255), then reset the values 0 and 255 in them to be 1 and 254, respectively.

Stage 2—Constructing a data hiding sequence of the intact regions.

- Step 7. Trace the boundaries of the regions in the input cover image S to obtain the adjusted regions R_1, R_2, \dots, R_k in S , and the chain-code sequence $\{(P_{xij}, P_{yij}), j = 1, 2, \dots, n_i\}$ of each R_i ($i = 1$ through k), as well as the information of some parameters of the regions, namely, the start points $(S_{x1}, S_{y1}), (S_{x2}, S_{y2}), \dots, (S_{xk}, S_{yk})$, the region directions D_1, D_2, \dots, D_k , and the average RGB color values $(C_{r1}, C_{g1}, C_{b1}), (C_{r2}, C_{g2}, C_{b2}), \dots, (C_{rk}, C_{gk}, C_{bk})$ of R_1, R_2, \dots, R_k , respectively.
 - Step 8. Rearrange randomly the *processing order* of the regions by the secret key K , resulting in a new coloring sequence $C_s = \{R_1', R_2', \dots, R_k'\}$; and change accordingly the corresponding orders of the start points, the region directions, and the average RGB color values, resulting in the new orders of $(S_{x1}', S_{y1}'), (S_{x2}', S_{y2}'), \dots, (S_{xk}', S_{yk}'), D_1', D_2', \dots, D_k'$, and $(C_{r1}', C_{g1}', C_{b1}'), (C_{r2}', C_{g2}', C_{b2}'), \dots, (C_{rk}', C_{gk}', C_{bk}')$, respectively.
-

Stage 3—Embedding the transformed message by coloring the intact stripes in the regions.

- Step 9. Partition each region R_i' ($i = 1$ through k) into stripes by the following steps until the bit sequence M'' is exhausted, or until all regions are fully embedded but the bit sequence M'' is not exhausted yet; for the latter case, show a message to tell the user the shortage of space for data embedding and exit.
- 9.1. Beginning from the start point (S_{xi}', S_{yi}') , partition the region R_i' into stripes parallel to the region direction D_i' in order according to the stripe width W_s .
 - 9.2. Draw the partition line between every two stripes with the nearly black color $(0, 1, 0)$.
 - 9.3. Fill the stripes in order with colors according to the secret message sequence M'' in the following way:
 - (a) fill an uncolored stripe with the color white $(255, 255, 255)$ if the next unprocessed bit of M'' is 0; or
 - (b) fill an uncolored stripe with the average region color $(C_{ri}', C_{gi}', C_{bi}')$ if the next unprocessed bit of M'' is 1.
- Step 10. Partition each intact region R_j' , which has not been used for message bit embedding so far, into stripes, draw partition lines between every two stripes with the nearly black color $(0, 1, 0)$, and fill the stripes with the average region color $(C_{rj}', C_{gj}', C_{bj}')$, or the color white $(255, 255, 255)$ randomly controlled by the secret key K .
- Step 11. Trace the pixels of the boundary of each region R_i' using its chain codes $\{(P_{xij}', P_{yij}')\}$, $j = 1, 2, \dots, n_i'\}$ and paint each traced pixel with the color black $(0, 0, 0)$.
- Step 12. Hide the start point (S_{xi}', S_{yi}') of each region R_i' , which is also the first point of the chain-code boundary points, (P_{xi1}', P_{yi1}') , as well as the second chain-code boundary points (P_{xi2}', P_{yi2}') , by coloring them with the nearly black colors $(0, 0, 1)$ and $(1, 0, 0)$, respectively.
- Step 13. Take the content of the final S as the desired stego-image I .
-

3.3. Proposed Secret Message Extraction Process

Before the data extraction process, the start point and the next boundary point of each region have to be found, which are then used to fetch the first direction of the chain code. After obtaining the information, all the chain codes of the region can be extracted, by which the entire region can be recovered. Then, the chain codes of the region can be used to reconstruct the process of partitioning the stripes and acquiring the secret message. In addition, the mapping order in the mapping table has to be constructed by utilizing the secret key. Based on the resulting mapping table, the secret message embedded in the stego-image can be retrieved accordingly. The detailed process of the secret data extraction process is given as Algorithm 3 in the following.

Algorithm 3. Extracting a secret message from a stego-image.

Input: a stego-image S and a secret key K identical to that used in Algorithm 2.

Output: the secret message M supposedly embedded in S .

Steps.

Stage 1—Reconstructing the data hiding sequence while retrieving the chain codes.

- Step 1. Conduct a raster scan of the entire stego-image S to obtain the first points of the boundary points, (P_{x11}, P_{y11}) , (P_{x21}, P_{y21}) , \dots , (P_{xn1}, P_{yn1}) , and the second boundary points (P_{x12}, P_{y12}) , (P_{x22}, P_{y22}) , \dots , (P_{xn2}, P_{yn2}) of regions R_1, R_2, \dots, R_n in S by the following steps:
- 1.1. Mark each pixel found with color $(0, 0, 1)$ as the first point (P_{xi1}, P_{yi1}) of the boundary points of a region R_i in S where (P_{xi1}, P_{yi1}) is also called the start point of the boundary denoted as (S_{xi}, S_{yi}) ;
 - 1.2. With the first point (P_{xi1}, P_{yi1}) found, scan the neighboring eight pixels in the clockwise direction according to Figure 7b to find a pixel with color $(1, 0, 0)$, and mark the pixel as the second boundary point (P_{xi2}, P_{yi2}) of R_i .
-

-
- Step 2. Mark a pixel with the color black (0, 0, 0) as a region boundary pixel, and a pixel with the near-like color (0, 1, 0) as a pixel of an inter-stripe partition line.
- Step 3. Retrieve the processing order of the regions by using the secret key K and denote the resulting new order of the regions as R_1', R_2', \dots, R_n' .
- Step 4. Acquire the chain-code sequence of each region R_i' via the start points (S_{xi}', S_{yi}') , which is also the first point of the chain-code boundary points (P_{xi1}', P_{yi1}') , and the second boundary point (P_{xi2}', P_{yi2}') by performing Step 5 of Algorithm 1.
- Step 5. Perform Step 9 of Algorithm 1 to obtain the region directions D_1', D_2', \dots, D_n' of the regions R_1' through R_n' .
- Step 6. Obtain the stripe width W_s by checking the distance of two neighboring partition lines of any stripe in any region.

Stage 2—Extracting the embedded secret message M .

- Step 7. Create an empty bit sequence Q initially.
- Step 8. Create a bit mapping table as shown in Table 2 with the mappings decided by the secret key K .
- Step 9. Extract the secret message bits from each region R_i' ($i = 1$ through n) by reversing the process of partitioning stripes in the following steps.
- 9.1. Use the region direction D_i' and the start point (S_{xi}', S_{yi}') to partition the region R_i' into stripes according to the stripe width W_s in order.
 - 9.2. Follow the stripe-partitioning order to obtain the color of each stripe, and extract a secret message bit in the following way and put it into Q : if the stripe color is the color white, set the secret value to be 0; otherwise, set the secret value to be 1.
 - 9.3. Transform every three extracted bits of Q into two according to the bit-mapping table generated in Step 8.
 - 9.4. Repeat Steps 9.2 to 9.3 until the three extracted bits are equal to the end pattern in the table.

Stage 3—Postprocessing the extracted message bits to be the desired secret message.

- Step 10. Use the secret key K to reorder the bits of Q , resulting in a new sequence Q' .
- Step 11. Transform every eight values of Q' into characters and take the result as the desired secret message M .
-

3.4. Security Considerations

As described in the previously presented data embedding and extraction algorithms implementing the proposed methods, based on the use of a secret key, the security of the embedded secret message is enhanced in three ways:

- (1) the randomization of the sequence of the secret message bits;
- (2) the randomization of the order of coloring the regions in the art image; and
- (3) the randomization of the 3-bit items in the 2-to-3 mapping table (an example is Table 2).

Accordingly, without the key it is very difficult for a malicious user to retrieve the correct secret message correctly. That is, the secret message can be strongly protected using the proposed data hiding method. Furthermore, with the steganographic effect of the dynamism and motion appearance of the stripe-based Futurism-like art image, people will tend to feel the stego-image as an art painting, and become unaware of the existence of the embedded secret message.

4. Experimental Results and Comparisons

Some experimental results using the proposed algorithms are presented in Section 4.1. In Section 4.2, a comparison of the art images yielded by the proposed art image creation algorithm with those yielded by three existing studies of art image creation and a comparison of the proposed data hiding method with four existing studies of data hiding via art images are described.

4.1. Experimental Results

Some experiments of applying Algorithm 1 to create stripe-based Futurism-like images have been conducted in this study. The threshold of stripe widths was set to decide the width of each stripe in each region segmented out of a source image. Because the feeling of artistic beauty is different to everyone, users were allowed to select the width of stripes using a program implementing the proposed method proposed in this study. Three choices were provided, which were narrow, middle-width, and wide stripes, and were 3, 6, and 9 pixels in width, respectively. Different choices resulted in different visual effects, and the user could choose their favorite type. Eight images were used in the experiments as the source images, as shown in Figures 4a,c, 9a,d, 13a, 14a, 15a and 16a. The created stripe-based Futurism-like art images with Figures 4a,c, and 9a,d as the source images are shown in Figures 4b,d and 10a,b, respectively, while those with Figures 13a, 14a, 15a and 16a as the source images are shown in Figures 13d, 14d, 15d and 16d, respectively. In these art images created in the experiments of this study, the abstract style of Futurism can be seen clearly from the created simple-shaped regions and motion-like stripes in the images.

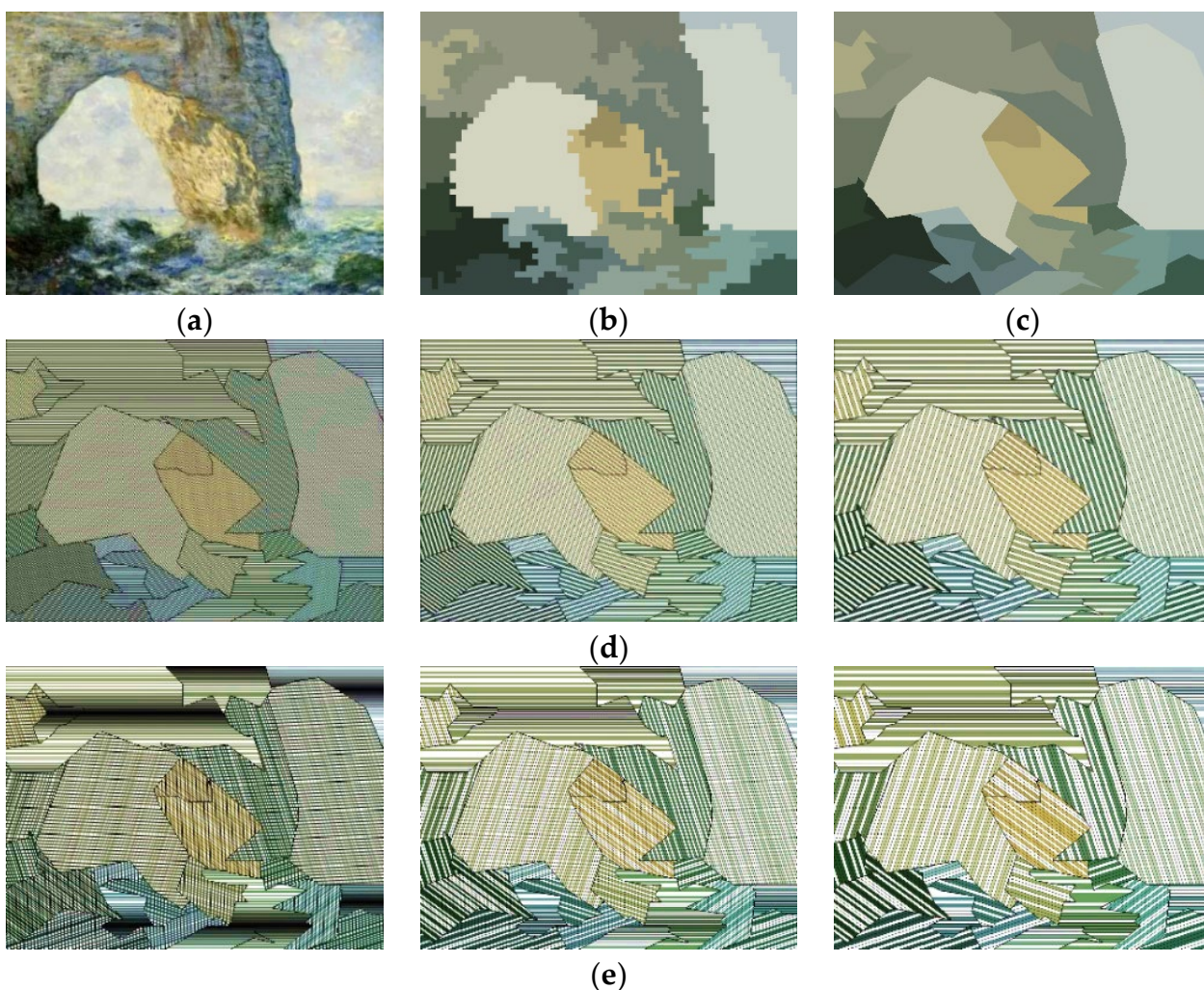


Figure 13. Experimental result 1. (a) An original image, “The Manneport, Rock Arch West of Etretat,” by Claude Monet, 1883. (b) The result of image segmentation of (a). (c) The result of polygon approximation of (b). (d) Three Futurism-like images created from (a) with the stripe widths of 3, 6, and 9 pixels, respectively. (e) Three stego-images created from (a) by embedding the message “One is never too old to learn.” with the secret key “Monet” and the stripe widths of 3, 6, and 9 pixels, respectively.



Figure 14. Experimental result 2. (a) An original image, “Cypresses,” by Vincent van Gogh, 1887. (b) The result of image segmentation of (a). (c) The result of polygon approximation of (b). (d) Three Futurism-like images created from (a) with the stripe widths of 3, 6, and 9 pixels, respectively. (e) Three stego-images created from (a) by embedding the message “Industry is the parent of success.” with the secret key “GOGH” and the stripe widths of 3, 6, and 9 pixels, respectively.

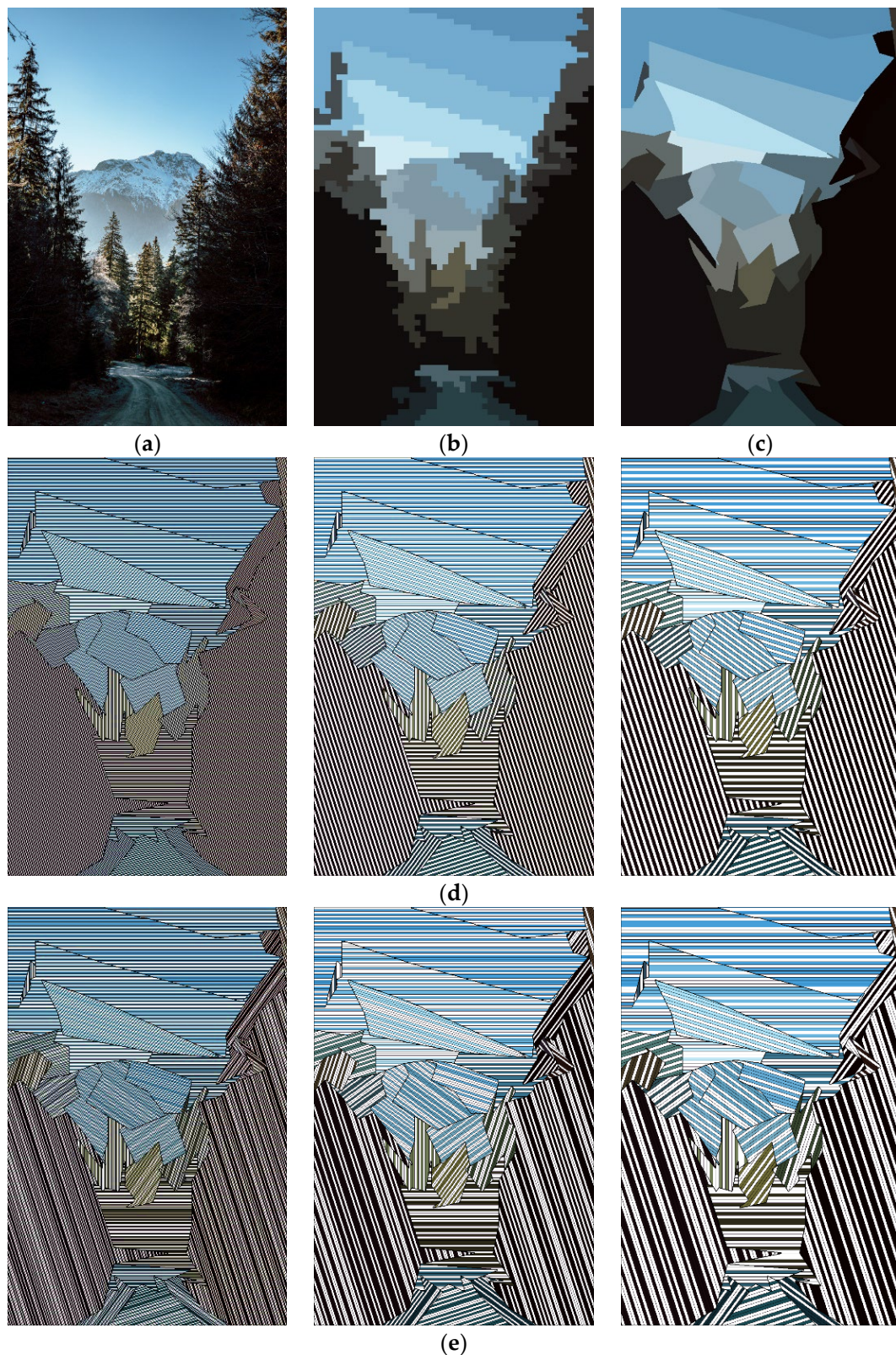


Figure 15. Experimental result 3. (a) An original image by Andrei Tanase [30]. (b) The result of image segmentation of (a). (c) The result of polygon approximation of each region of (b). (d) Three Futurism-like images created by (a) with the stripe widths of 3, 6, and 9 pixels, respectively. (e) Three stego-images created from (a) by embedding the message “Actions speak louder than words.” with the secret key “Action” and the stripe widths of 3, 6, and 9 pixels, respectively.

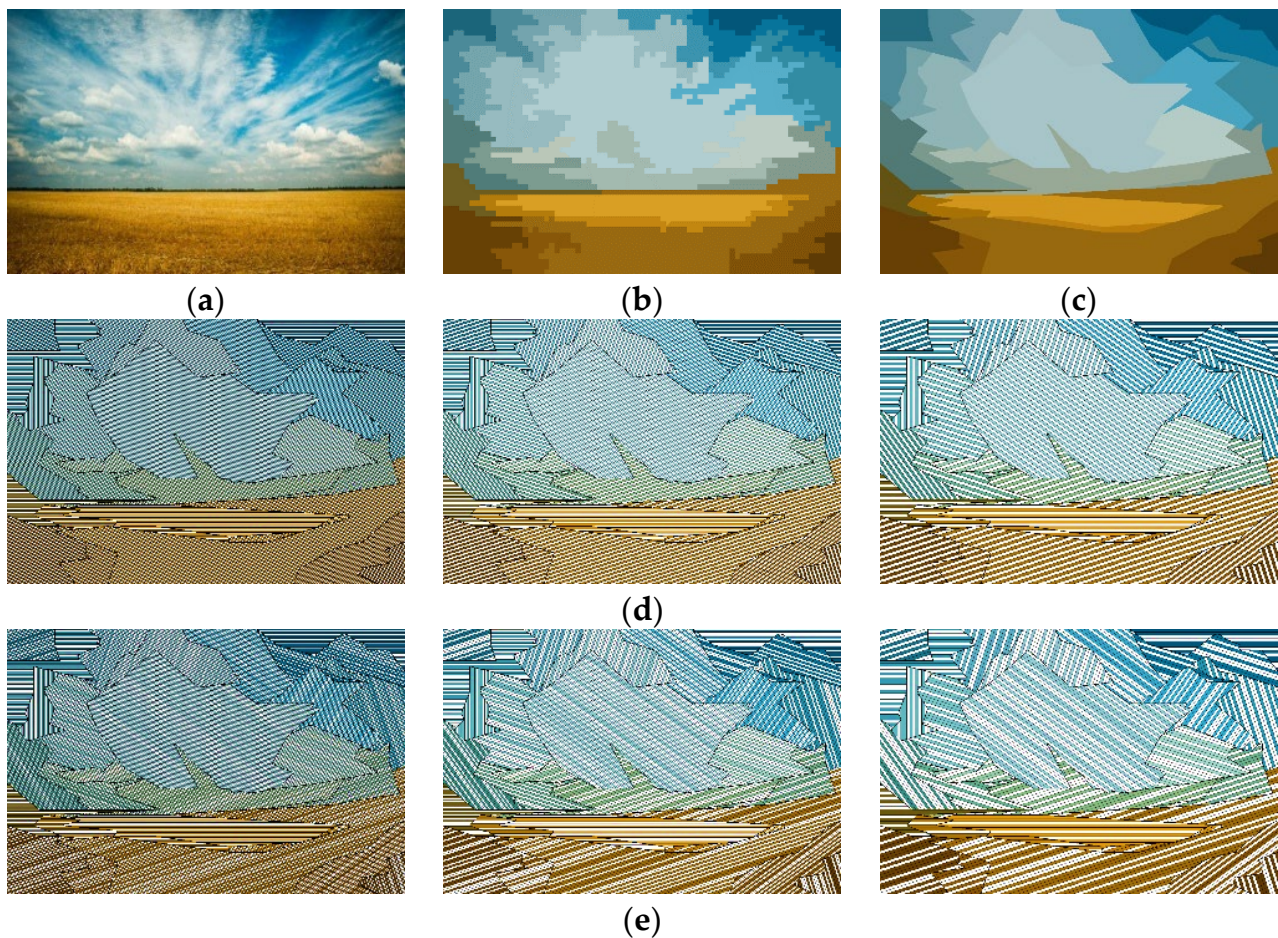


Figure 16. Experimental result 4. (a) An original image by Oleksandr Pidvalnyi [31]. (b) The result of image segmentation of (a). (c) The result of polygon approximation of (b). (d) Three Futurism-like images created from (a) with the stripe widths of 3, 6, and 9 pixels, respectively. (e) Three stego-images created from (a) by embedding the message “Make hay while the sun shines.” with the secret key “Positive” and the stripe widths of 3, 6, and 9 pixels, respectively.

Next, the embedding capacities of the eight images to which Algorithm 2 was applied with different stripe widths are shown in Table 3, where the number of embedded bits of secret messages in an image can be calculated to be:

$$\text{No. of embedded secret bits} = \lfloor (\text{No. of stripes} - 3) / 3 \rfloor \times 2 \quad (2)$$

where $\lfloor \cdot \rfloor$ is the floor function. Note that the RGB color of each stripe offers the embedding capacity of one bit. Note that the term “ -3 ” in Equation (2) comes from the fact that three of the stripes are used for embedding the ending pattern of 3 bits which are not secret message bits. Obviously, the widths of the stripes will affect the embedding capacity of the created art image. The smaller the stripe widths, the more stripes are created, and so the more embedded message bits are created, and vice versa.

Four experimental results of applying Algorithm 2 for data embedding are shown in Figures 13–16e, which were created from the source images shown in Figures 13–16a by Algorithm 1, respectively, with the stripe widths of 3, 6, and 9 pixels. Specifically, these experimental results were created in the following way.

- (1) With Figure 13a as the source image and the secret key “Monet,” three stego-images were created, into which the secret message “One is never too old to learn.” was embedded, as shown in Figure 13e.
- (2) By using Figure 14a as the source image and the secret key “GOGH,” three stego-images were created, into which the secret message “Industry is the parent of success.” was embedded, as shown in Figure 14e.
- (3) By using Figure 15a as the source image and the secret key “Action,” three stego-images were created, into which the secret message “Actions speak louder than words.” was embedded, as shown in Figure 15e.
- (4) By using Figure 16a as the source image and the secret key “Positive,” three stego-images were created, into which the secret message “Make hay while the sun shines.” was embedded, as shown in Figure 16e.

Figures 17–20 show the results of the extracted secret messages from two stego-images. As shown in Figures 17 and 19, by using the right key, the embedded secret messages were extracted successfully. On the contrary, if an incorrect secret key is used in the process of secret message extraction, the extracted secret message will be incorrect, as illustrated by Figures 18 and 20.

Table 3. Embedding capacities (in the unit of bit) of eight cover images with different stripe widths.

Title of Cover Image	Image Size	3 Pixels in Width		6 Pixels in Width		9 Pixels in Width	
		No. of Stripes	No. of Secret Bits	No. of Stripes	No. of Secret Bits	No. of Stripes	No. of Secret Bits
The Great Wave off Kanagawa (Figure 4a)	928 × 640	1535	1020	780	518	533	353
A Meadow in the Mountains (Figure 4c)	1024 × 768	1781	1184	907	602	610	404
Canto patriottico-Piazza Siena (Figure 9a)	1392 × 580	1884	1254	958	636	647	429
Forme grido Viva l’Italia (Figure 9d)	1056 × 764	1823	1212	925	614	626	415
The Manneport, Rock Arch West of Etretat (Figure 13a)	1024 × 768	1581	1052	803	533	541	358
Cypresses (Figure 14a)	768 × 1009	1592	1058	810	538	549	364
Untitled picture (taken by Andrei Tanase [30]) (Figure 15a)	800 × 1200	1796	1194	912	606	620	411
Untitled picture (taken by Oleksandr Pidvalnyi [31]) (Figure 16a)	1200 × 798	2028	1350	1027	682	696	462

Additionally, 100 images were obtained from the image database of the website Pexels [32] for an experiment of creating stripe-based Futurism-like art images, embedding as many message bits into them as possible, and then computing the resulting three parameters of the embedding capacity (bits), the embedding rate (bits per pixel or bpp) for stripes with the width of 6 pixels, as well as the K–L divergence value for each stego-image. The results are shown in Table 4. It is noted that the longer side of each obtained original image from the database was resized to be of the length of 1024 pixels before being processed by the proposed art-image-creating algorithm (Algorithm 1). As can be seen from Table 4, the message embeddable into an art image for covert communication on average includes 783 bits, which is equivalent to $(783 - 3) \div 3 \times 2 \div 8 = 65$ characters.

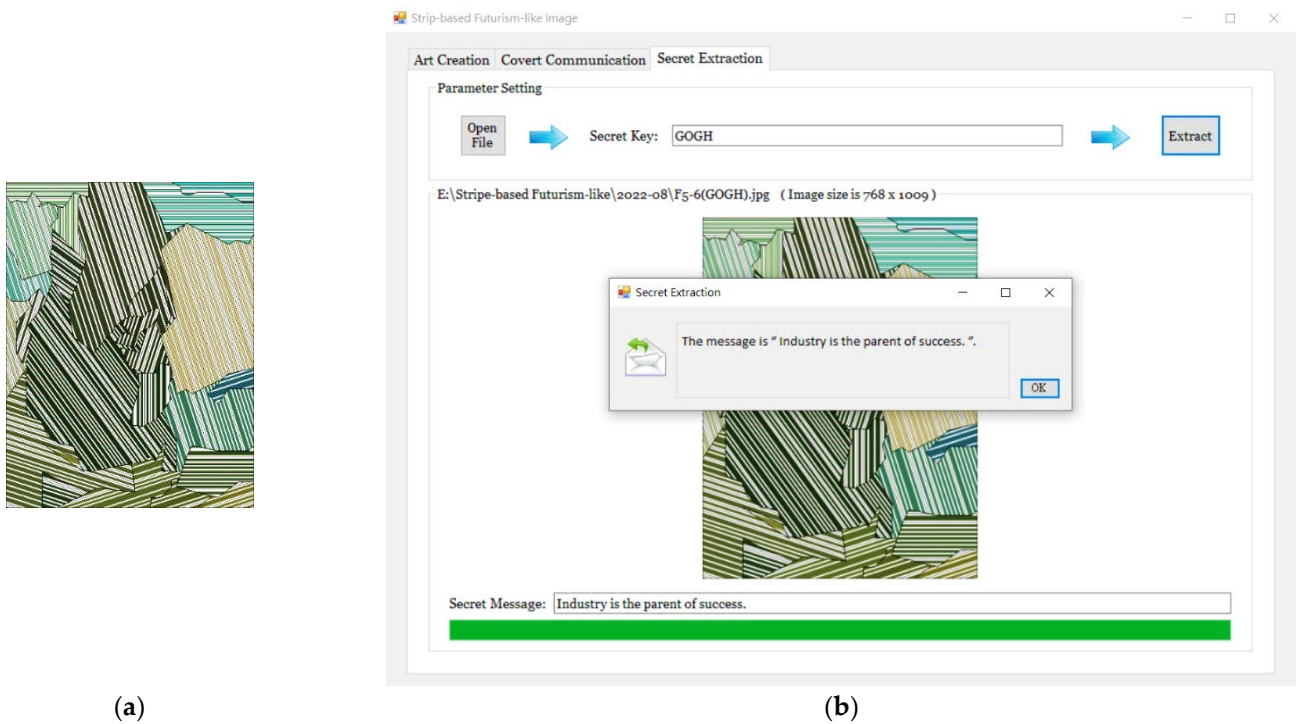


Figure 17. An illustration of extracting the secret message from the stego-image using the right secret key. (a) The stego-image with the stripe widths of 6 pixels shown in Figure 14e. (b) The correct result “Industry is the parent of success.” of hidden data extraction using the secret key “GOGH”.

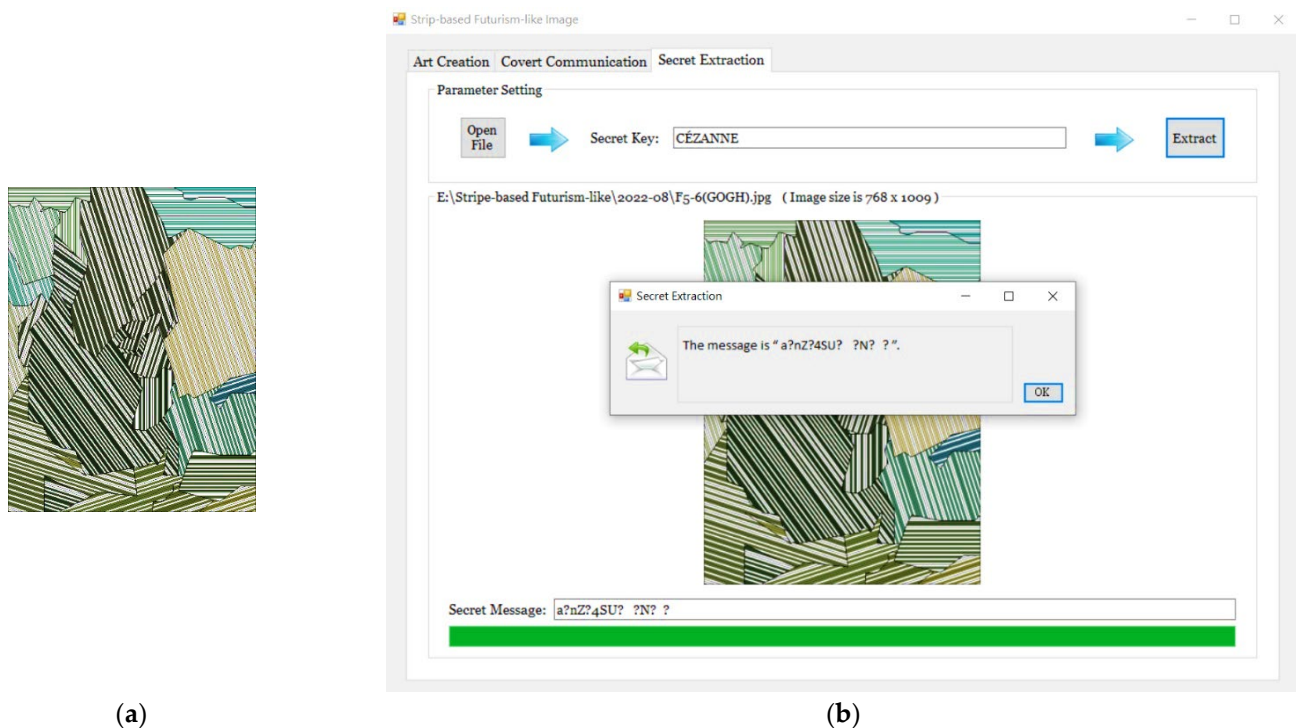
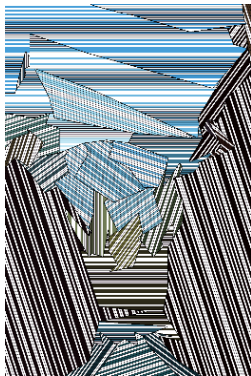
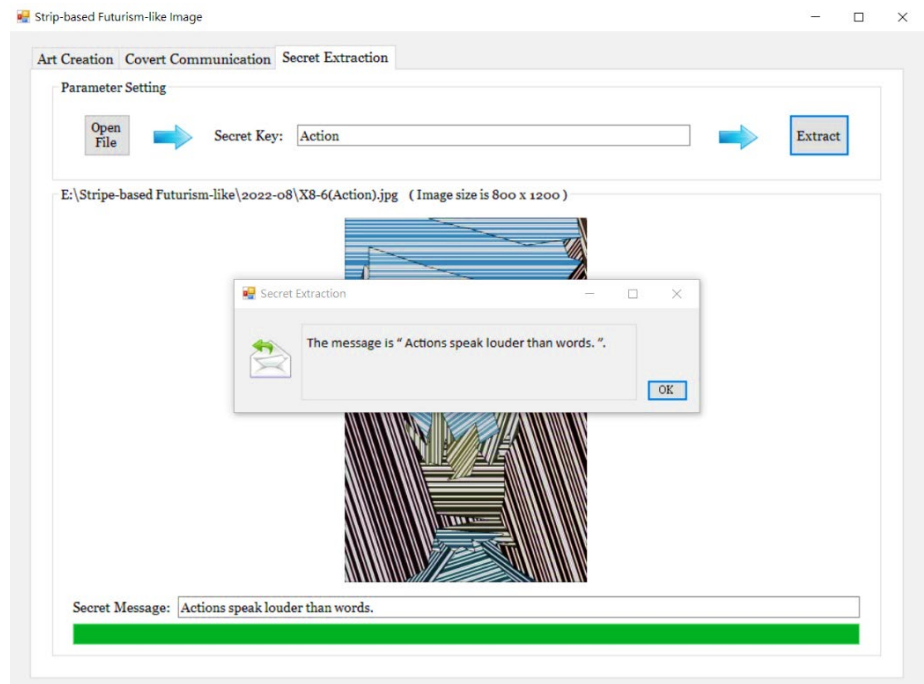


Figure 18. An illustration of extracting an erroneous secret message from the stego-image using an incorrect secret key “CéZANNE”. (a) The stego-image with the stripe width of 6 pixels shown in Figure 14e. (b) The erroneous hidden data extraction result.



(a)

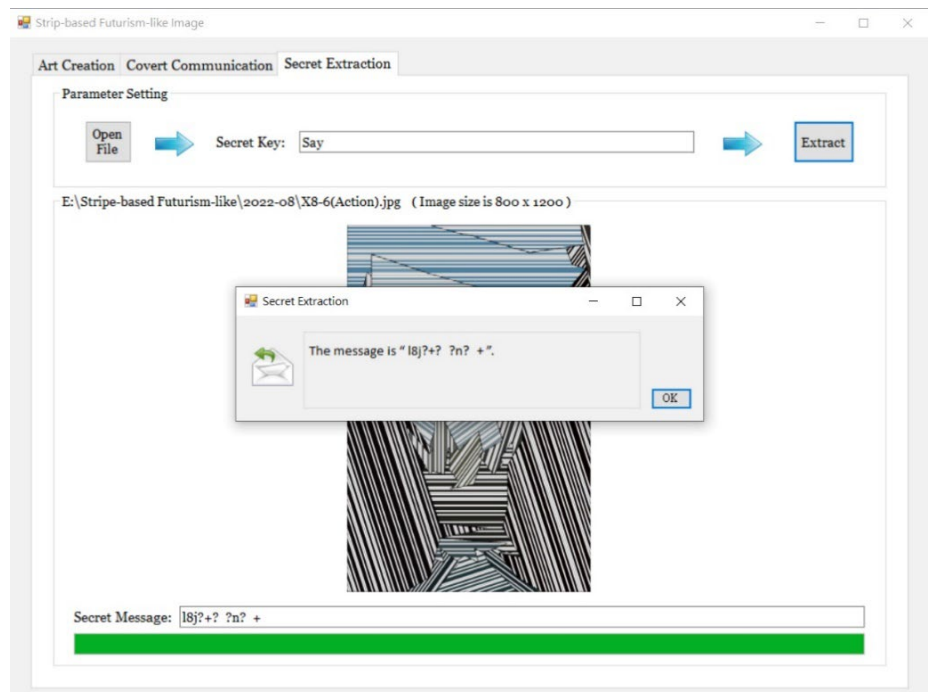


(b)

Figure 19. A second illustration of extracting the secret message from the stego-image using the right secret key. (a) The stego-image with the stripe widths of 6 pixels shown in Figure 15e. (b) The correct result “Actions speak louder than words.” of hidden data extraction using the secret key “Action”.








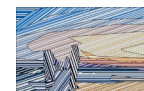

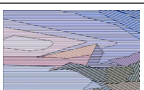


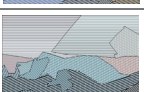


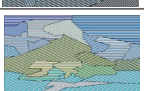


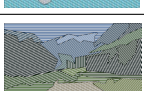
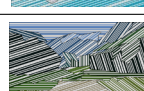

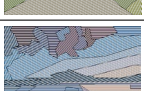
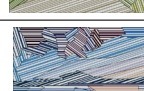

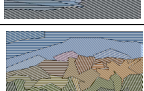


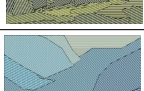




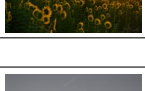
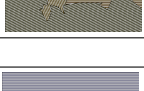
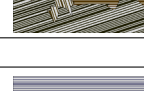

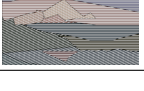

(a)



(b)

Figure 20. An illustration of extracting an erroneous secret message from the stego-image using an incorrect secret key “Say”. (a) The stego-image with the stripe width of 6 pixels shown in Figure 15e. (b) The erroneous hidden data extraction result.

Table 4. Some statistics of 100 tested images where created art images have stripes with 6-pixel widths.

Image Title	Original Image	Art Image (Cover Image)	Stego-Image	Image Size (Pixel)	Total Pixels in Image	Embedding Capacity (Bits)	Embedding Rate (bpp)	K–L Divergence Value
A01				1024 × 682	698,368	673	0.0010	0.00293
A02				1024 × 683	699,392	718	0.0010	0.00451
A03				1024 × 703	719,872	672	0.0009	0.00415
A04				1024 × 683	699,392	528	0.0008	0.00121
A05				1024 × 684	700,416	626	0.0009	0.00183
A06				1024 × 678	694,272	770	0.0011	0.00284
A07				1024 × 683	699,392	711	0.0010	0.00253
A08				1024 × 683	699,392	1079	0.0015	0.00291
A09				1024 × 683	699,392	535	0.0008	0.00184
A10				1024 × 683	699,392	495	0.0007	0.00202
A50				1024 × 681	697,344	389	0.0006	0.00055
A100				1024 × 646	661,504	787	0.0012	0.00250
Summary	Maximum			1024 × 837	857,088	1354	0.0016	0.00475
	Minimum			1024 × 482	493,568	266	0.0005	0.00024
	Average				700,498	783	0.0011	0.00252

Furthermore, a concern in the field of data hiding or steganography is that a stego-image yielded from an input cover image should be as similar to the cover image as possible. To measure such image similarity value, with the two images being both the art image created in this study and the corresponding stego-image, the Kullback–Leibler (K–L) divergence measure [24,33], which is essentially a relative entropy, is considered to be appropriate and so was adopted for use in this study, as described in the following:

$$D_{KL}(S||C) = \sum_x S(x) \ln\left(\frac{S(x)}{C(x)}\right) \tag{3}$$

where $C(x)$ and $S(x)$ are the probability functions representing the pixel-value distributions of the cover image I_C and the stego-image I_S , respectively, with the pixel values of the two images being represented in this study by a *quantity-reduced version* h' of the 1-D color h (with values ranging from 0 to 18,615) described by Equation (1). That is, the scale of the value h was reduced to be h' of just 256 levels (with values ranging from 0 to 255) for the purpose of simplifying the computation involved in Equation (3). It is noted that the larger the similarity between the two images $C(x)$ and $S(x)$, the smaller the K–L divergence value computed according to Equation (3).

The computed K–L divergence values computed for the 100 art images and the corresponding stego-images mentioned previously (in Columns 3 and 4 in Table 4) are listed in Table 4 as well (in Column 9), from which it can be seen that all such K–L divergence values are very small, with an average of 0.00252, meaning that the stego-images were quite similar to the respective art images. This means in turn that each stego-image looks similar to the corresponding cover image *from far-distance observations*.

As an example, in Figure 21 we show the histograms of the cover image and the corresponding stego-image created from an image (the first image shown in Table 4) obtained from the database of the website Pexels [32], where the histogram of the cover image is shown as red-colored bars, that of the corresponding stego-image as blue bars, and the difference between the two histograms as a gray line chart. Additionally, shown in the figure are all of the non-zero bins (numbered 0, 36, 84, . . . , 255) of the quantity-reduced color value h' and the numbers of pixels falling in each bin.

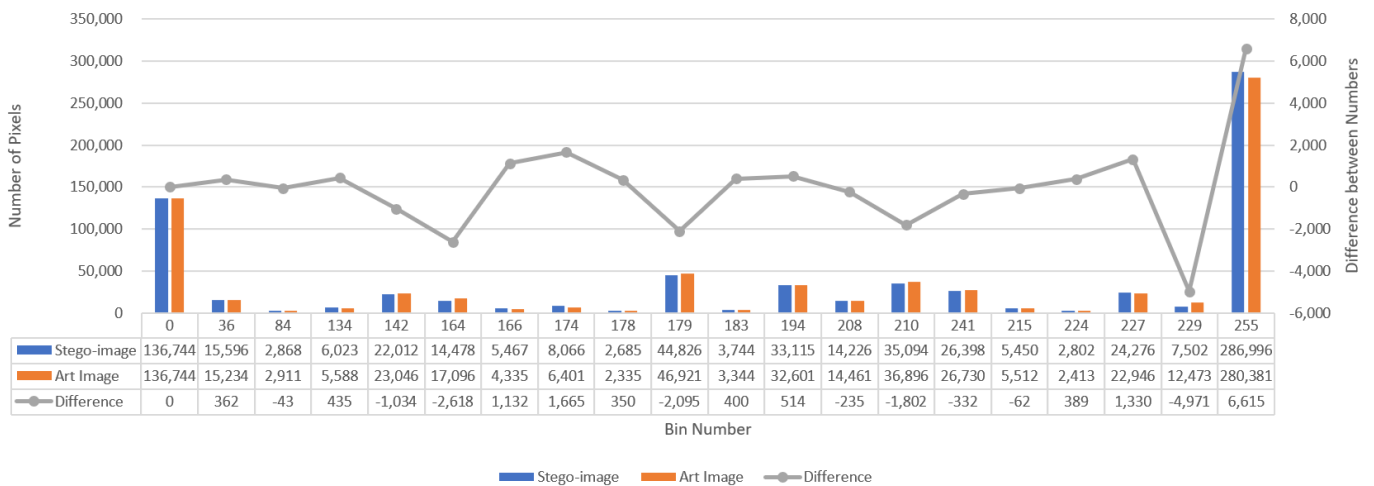


Figure 21. The histograms of the cover art image (shown as red-colored bars) and the corresponding stego-image (shown as blue bars) created from an image (the first image in Table 4), as well as their difference (shown as the gray line chart).

4.2. Comparisons with Existing Methods

As a comparison of the effect of the proposed art image generation algorithm (Algorithm 1) with those yielded by some existing methods [3–5], the three images shown in Figure 2 were taken as inputs to the proposed algorithm to yield three stripe-based Futurism-like art images, respectively, as shown in Figure 22, which can be contrasted with those shown in Figure 2, showing interestingly different art styles generated by different computer art algorithms for the same images.

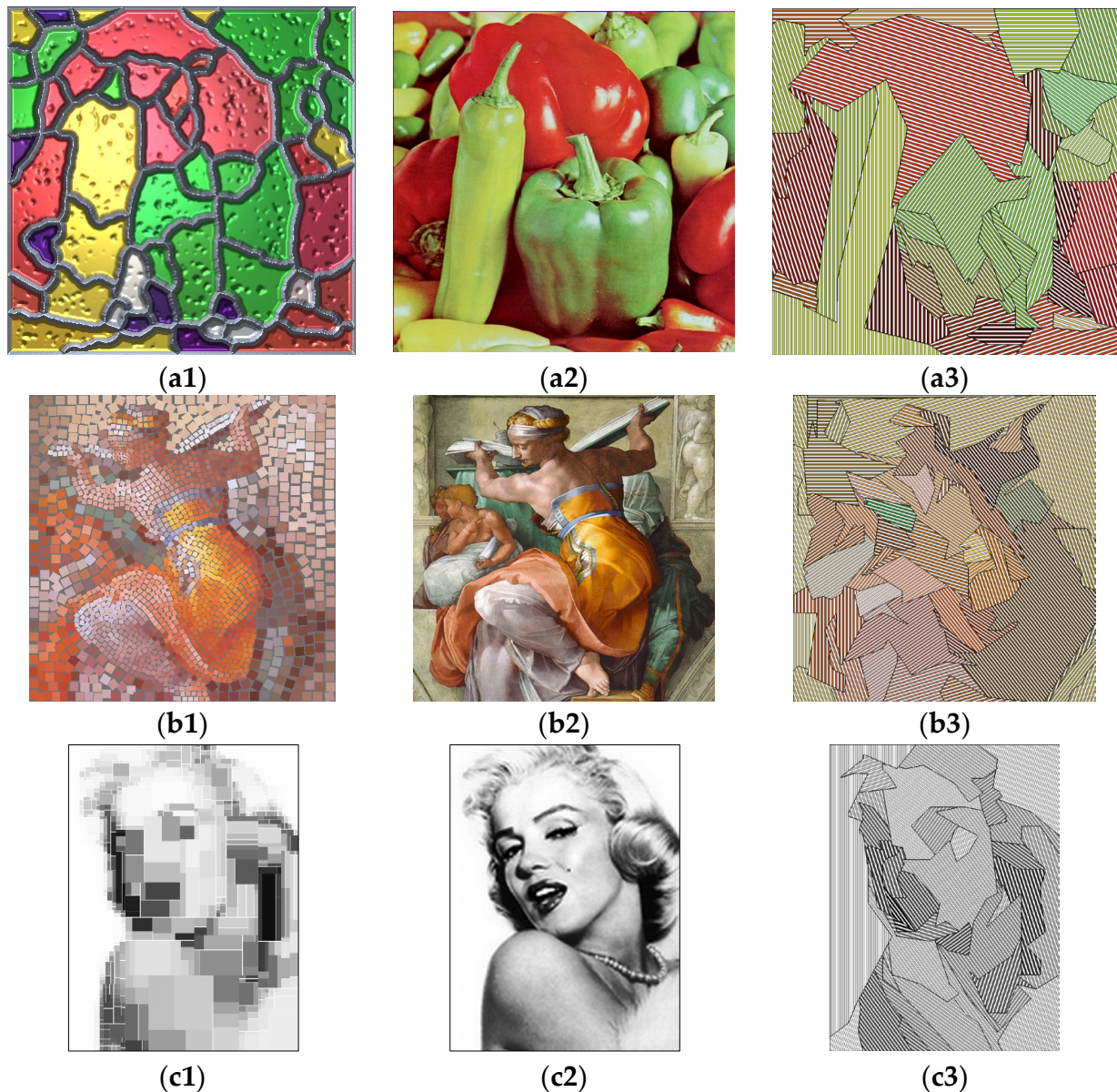







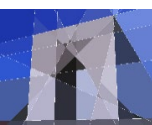
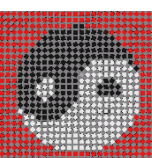
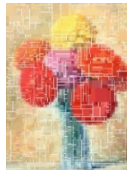


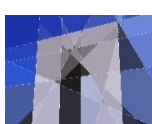
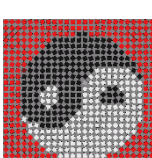




Figure 22. Stripe-based Futurism-like images generated in this study using the images in Figure 2 as inputs. (a1–a3) A stained-glass-like image created by Mould [3], the corresponding original image, and the stripe-based Futurism-like image generated by the proposed algorithm (Algorithm 1). (b1–b3) A tile-mosaic-like image created by Hausner [4], the corresponding original image, and the stripe-based Futurism-like image generated by the proposed algorithm (Algorithm 1). (c1–c3) A rectangle-based Neo-Plasticism-like image created by Liu et al. [5], the corresponding original image, and the stripe-based Futurism-like image generated by the proposed algorithm (Algorithm 1).

So far, there are only a few studies on data hiding via art images, such as the stained-glass image investigated by Hung et al. [14], the line-based Cubism-like image explored by Liu and Tsai [9], the tile mosaic image proposed by Hung et al. [15], and the rectangle-based Neo-Plasticism-like image studied by Liu et al. [5]. The techniques used by these investigations and the results yielded by them are compared in this study with those of the proposed method (implemented by Algorithm 2) *in a qualitative way*, as shown in Table 5 below.

Table 5. A comparison of the proposed method with four art image generation methods with data hiding capabilities.

Method Item	Hung et al. [14]	Liu and Tsai [9]	Hung et al. [15]	Liu et al. [5]	Proposed Method
Type of art image	Stained-glass image	Line-based Cubism-like image	Tile mosaic image	Rectangle-based Neo-Plasticism-like image	Stripe-based Futurism-like Image
Hidden object type (an example)	A secret message ('Damou, one thing I did not dare to mention to you till today is: I love you.') (in Chinese)	A secret message ('Hi, I am Helen. Nice to meet you!')	 A watermark image	A secret message ('Life is not an exact science, it is an art.')	A secret message ('Industry is the parent of success.')
Original image					
Cover art image					
Stego-image (with the above hidden object)					
Data hiding technique	Changing the number of tree nodes in the image	Changing the region colors in the image while keeping the average colors unchanged.	Changing the orientations of the tiles in the image	Substituting the LSBs of each region's colors or generating additional partition lines in the region	Filling the stripes in the region with the region's average color, or the color white in a message-dependent fashion
Application	Information protection	Lossless data hiding	Copyright protection	Covert communication	Covert communication

- (1) Hung et al. [14] created a type of stained-glass art image, and embedded a message into a cover art image for information protection by changing the number of nodes in the tree structure obtained from scanning the art image pixels;
- (2) Liu and Tsai [9] constructed a type of line-based Cubism-like image and embedded a message into a cover art image for the purpose of lossless data hiding by changing the region colors in the art image while keeping the average colors unchanged;

- (3) Hung et al. [15] created a type of tile mosaic art image and embedded a watermark image into a cover art image to conduct watermarking for the purpose of copyright protection by changing the orientations of the tiles in the art image; and
- (4) Liu et al. [5] constructed a type of rectangle-based Neo-Plasticism-like art image and embedded a message into a cover art image by substituting the LSBs of each region's colors or generating additional partition lines in the region for the purpose of covert communication.

Note that the art image creation and data hiding techniques used by the four methods and the one proposed in this study are all different because the line and region structures in the types of art image used in the techniques are different. Therefore, the qualities of the art images created by these methods, which are used as cover images, as well as the bit-embedding capabilities yielded by them, are also all different and variable.

5. Discussion

Stripe width is an important parameter for generating stripe-based Futurism-like images. The width of the stripe will affect the appearance of the image. The wider the stripes are created in the regions, the stronger the dynamic effect appearing in the image, and vice versa. Furthermore, the width of the stripe will also affect the human visual perception of the region color. Because the boundary line of the stripe is black, if the stripe width is smaller, there will be more stripes and more boundary lines in the region. However, the overall visual appearance of the regions will be darker. On the contrary, if the stripe width is larger, the created stripes will be relatively fewer, so that the overall visual appearance of the created art image will be lighter.

In addition, when using Algorithm 2 to embed message information into the generated art image, if the stripe width is set too large to yield enough stripes for embedding the secret message completely, then the stripe width can be reduced to increase the embedding capacity, making the proposed data hiding method more adaptable for covert communication applications. Additionally, when the bits of the message are not so many to be embedded into all the stripes in a stripe-based Futurism-like art image, those stripes in which no message bits are embedded are painted in white or with the region's average colors as well in a random fashion in this study, so that no difference of coloring styles in the regions can be detected. This increases the security of the embedded message data against possible hackers' attacks.

In the proposed method, a secret key is used for data security control. However, according to Paul et al. [34], it is also appropriate in this study to use *the number of the regions or the total number of corner points* in an art image as the key for data security control. In this way, there is no need to transfer an extra secret key from the sender to the receiver during the covert communication process.

In the future, to test the resistance of the proposed data hiding method against possible hidden data destruction conducted by hackers, the techniques of steganogram removal using multidirectional diffusion [35], multibit sterilization using pixel eccentricity [36], as well as deep residual networks for steganalysis of digital images [37] may be tried.

6. Conclusions

A new computer art, called stripe-based Futurism-like image, has been proposed. To generate such a type of image automatically, the regions in a source image are first transformed into polygonal shapes by connecting the region corners yielded by chain-code analysis. Next, the region directions are obtained by analyzing the directions of the region edges. Then, by partitioning each region into stripes in accordance with its direction and assigning alternatively the average region colors and the white colors to the stripes, an art image is created, which is an abstract geometric form of the source image showing a dynamic motion effect of Futurism. In addition, a data hiding method which assigns random colors to the region stripes according to the bits of a secret message has been proposed. To avoid assigning successive identical colors to the stripes, a 2-to-3 mapping

scheme is used to transform the message bits into a sequence with identical-bit segments no longer than three bits. Hidden data security is enhanced by the use of a secret key to randomize the message bit sequence and the processing order of the regions, as well as the 2-to-3 mapping items. The artistic content of the resulting Futurism-like image will attract an observer's attention, reducing his/her suspicion about the existence of the hidden message; therefore, the proposed methods are useful for covert communication. Good experimental results of processing a database of 100 images have been yielded, which prove the feasibility of the proposed methods.

Author Contributions: Conceptualization, S.-C.L. and W.-H.T.; methodology, S.-C.L. and W.-H.T.; software, S.-C.L. and Wu; validation, S.-C.L. and D.-C.W.; formal analysis, S.-C.L., D.-C.W. and W.-H.T.; investigation, S.-C.L. and D.-C.W.; resources, W.-H.T.; data curation, S.-C.L. and D.-C.W.; writing—original draft preparation, S.-C.L.; writing—review and editing, D.-C.W. and W.-H.T.; visualization, D.-C.W. and W.-H.T.; supervision, W.-H.T.; project administration, D.-C.W. and W.-H.T.; funding acquisition, W.-H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported partially by the Ministry of Science and Technology, Taiwan under Grant No. NSC 99-2631-H-009-001 and No. NSC 100-2631-H-009-001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Wen-Hsiang Tsai is the PI and Da-Chun Wu is the Co-PI of the two projects listed in the Funding section.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Adams, L.S. *A History of Western Art*, 5th ed.; McGraw-Hill Education: New York, NY, USA, 2010.
2. Hertzmann, A. A Survey of Stroke-based Rendering. *IEEE Comput. Graph. Appl.* **2003**, *23*, 70–81. [[CrossRef](#)]
3. Mould, D. A Stained Glass Image Filter. In Proceedings of the 14th Eurographics Workshop on Rendering, Leuven, Belgium, 25–27 June 2003; pp. 20–25.
4. Hausner, A. Simulating Decorative Mosaics. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 12–27 August 2001; pp. 573–580.
5. Liu, S.C.; Wu, D.C.; Tsai, W.H. Rectangle-based Neo-Plasticism-like Image—A New Type of Art Image and Its Application to Covert Communication. *J. Inf. Sci. Eng.* **2021**, *37*, 441–468.
6. Matsumura, N.; Tokura, H.; Kuroda, Y.; Ito, Y.; Nakano, K. Tile Art Image Generation Using Conditional Generative Adversarial Networks. In Proceedings of the 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW), Takayama, Japan, 27–30 November 2018; pp. 209–215.
7. Fahim, M.A.N.I.; Hossain, S. A Simple Way to Create Pointillistic Art from Natural Images. In Proceedings of the 2017 3rd IEEE International Conference on Cybernetics (CYBCONF), Exeter, UK, 21–23 June 2017; pp. 1–5.
8. Avila, L.; Bailey, M. Art in the Digital Age. *IEEE Comput. Graph. Appl.* **2016**, *36*, 6–7.
9. Liu, S.C.; Tsai, W.H. Line-based Cubism-like Image—A New Type of Art Image and Its Application to Lossless Data Hiding. *IEEE Trans. Inf. Forensic Secur.* **2012**, *7*, 1448–1458.
10. Salisbury, M.P.; Wong, M.T.; Hughes, J.F.; Salesin, D.H. Orientable Textures for Image-based Pen-and-ink Illustration. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 3–8 August 1997; pp. 401–406.
11. Hertzmann, A. Fast Paint Texture. In Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering, Annecy, France, 3–5 June 2002; pp. 91–96.
12. Lai, I.J.; Tsai, W.H. Secret-fragment-visible Mosaic Image—A New Computer Art and Its Application to Information Hiding. *IEEE Trans. Inf. Forensic Secur.* **2011**, *6*, 936–945.
13. Lee, Y.L.; Tsai, W.H. A New Secure Image Transmission Technique via Secret-fragment-visible Mosaic Images by Nearly Reversible Color Transformations. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *24*, 695–703.
14. Hung, S.C.; Wu, D.C.; Tsai, W.H. Data Hiding in Computer-Generated Stained Glass Images and Its Applications to Information Protection. *IEICE Trans. Inf. Syst.* **2020**, *E103-D*, 850–865. [[CrossRef](#)]
15. Hung, S.C.; Liu, T.Y.; Tsai, W.H. A New Approach to Automatic Generation of Tile Mosaic Images for Data Hiding Applications. In Proceedings of the 2005 Conference on Digital Contents Management & Applications, Kaohsiung, Taiwan, 18 June 2005; pp. 11–20.

16. Guide to Futurism: History and Characteristics of Futurism. Available online: <https://www.masterclass.com/articles/futurism-art-guide#what-is-futurism> (accessed on 25 July 2022).
17. Wang, R.Z.; Lin, C.F.; Lin, J.C. Image Hiding by Optimal LSB Substitution and Genetic Algorithm. *Pattern Recognit.* **2001**, *34*, 671–683. [CrossRef]
18. Wu, D.C.; Tsai, W.H. A Steganographic Method for Images by Pixel-value Differencing. *Pattern Recognit. Lett.* **2003**, *24*, 1613–1626. [CrossRef]
19. Ni, Z.C.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible Data Hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
20. Zhong, X.; Huang, P.C.; Mastorakis, S.; Shi, F.Y. An Automated and Robust Image Watermarking Scheme Based on Deep Neural Networks. *IEEE Trans. Multimed.* **2020**, *23*, 1951–1961. [CrossRef]
21. Shih, F.Y.; Zhong, X.; Chang, I.; Satoh, S. An Adjustable-purpose Image Watermarking Technique by Particle Swarm Optimization. *Multimed. Tools Appl.* **2018**, *77*, 1623–1642. [CrossRef]
22. Wu, H.Y.; Chen, L.H.; Ching, Y.T. Block-Based Steganography Method Using Optimal Selection to Reach High Efficiency and Capacity for Palette Images. *Appl. Sci.* **2020**, *10*, 7820.
23. Järpe, E.; Weckstén, M. Velody 2—Resilient High-Capacity MIDI Steganography for Organ and Harpsichord Music. *Appl. Sci.* **2021**, *11*, 39. [CrossRef]
24. Mandal, P.C.; Mukherjee, I.; Paul, G.; Chatterji, B.N. Digital Image Steganography: A Literature Survey. *Inf. Sci.* **2022**, *609*, 1451–1488. [CrossRef]
25. Hojjatoleslami, S.A.; Kittler, J. Region Growing: A New Approach. *IEEE Trans. Image Process.* **1998**, *7*, 1079–1084. [CrossRef]
26. Chang, Y.L.; Li, X. Adaptive Image Region-growing. *IEEE Trans. Image Process.* **1994**, *3*, 868–872. [CrossRef]
27. Haralick, R.M.; Sternberg, S.R.; Zhuang, X. Image Analysis Using Mathematical Morphology. *IEEE Trans. Pattern Anal. Mach. Intell.* **1987**, *9*, 532–550. [CrossRef]
28. Freeman, H. On the Encoding of Arbitrary Geometric Configuration. *IRE Trans. Electron. Comput.* **1961**, *10*, 260–268. [CrossRef]
29. Sanchez-Cruz, H. A Proposal Method for Corner Detection with An Orthogonal Three-Direction Chain Code. In *Lecture Notes in Computer Science*; (ACIVS 2006); Springer: Berlin, Germany, 2006; Volume 4179, pp. 161–172.
30. Tanase, A. Pexels. Available online: <https://www.pexels.com/photo/trees-near-a-mountain-6558452> (accessed on 5 August 2022).
31. Pidvalnyi, O. Pexels. Available online: <https://www.pexels.com/photo/photo-of-grass-field-1227513> (accessed on 5 August 2022).
32. Pexels. Available online: <https://www.pexels.com> (accessed on 5 August 2022).
33. Cachin, C. An Information-theoretic Model for Steganography. *Inf. Comput.* **2004**, *192*, 41–56. [CrossRef]
34. Paul, G.; Davidson, I.; Mukherjee, I.; Ravi, S.S. Keyless Dynamic Optimal Multi-bit Image Steganography Using Energetic Pixels. *Multimed. Tools Appl.* **2017**, *76*, 7445–7471. [CrossRef]
35. Geetha, S.; Subburam, S.; Selvakumar, S.; Kadry, S.; Damasevicius, R. Steganogram Removal Using Multidirectional Diffusion in Fourier Domain While Preserving Perceptual Image Quality. *Pattern Recognit. Lett.* **2021**, *147*, 197–205. [CrossRef]
36. Mukherjee, I.; Paul, G.; Jawahar, J.A. Defeating Steganography with Multibit Sterilization Using Pixel Eccentricity. *IPSI BgD Internet Res. Soc.* **2015**, *11*, 25–34.
37. Boroumand, M.; Chen, M.; Fridrich, J. Deep Residual Network for Steganalysis of Digital Images. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1181–1193. [CrossRef]