OVERPAINT: Automatic Multi-Layer Stencil Generation without Bridges

Yuta Fukushima
The University of Tokyo
yuta04kaichi@g.ecc.u-tokyo.ac.jp

Anran Qi
The University of Tokyo
annranqi1024@g.ecc.u-tokyo.ac.jp

I-Chao Shen
The University of Tokyo
ichao.shen@ui.is.s.u-tokyo.ac.jp

Takeo Igarashi
The University of Tokyo
takeo@acm.org

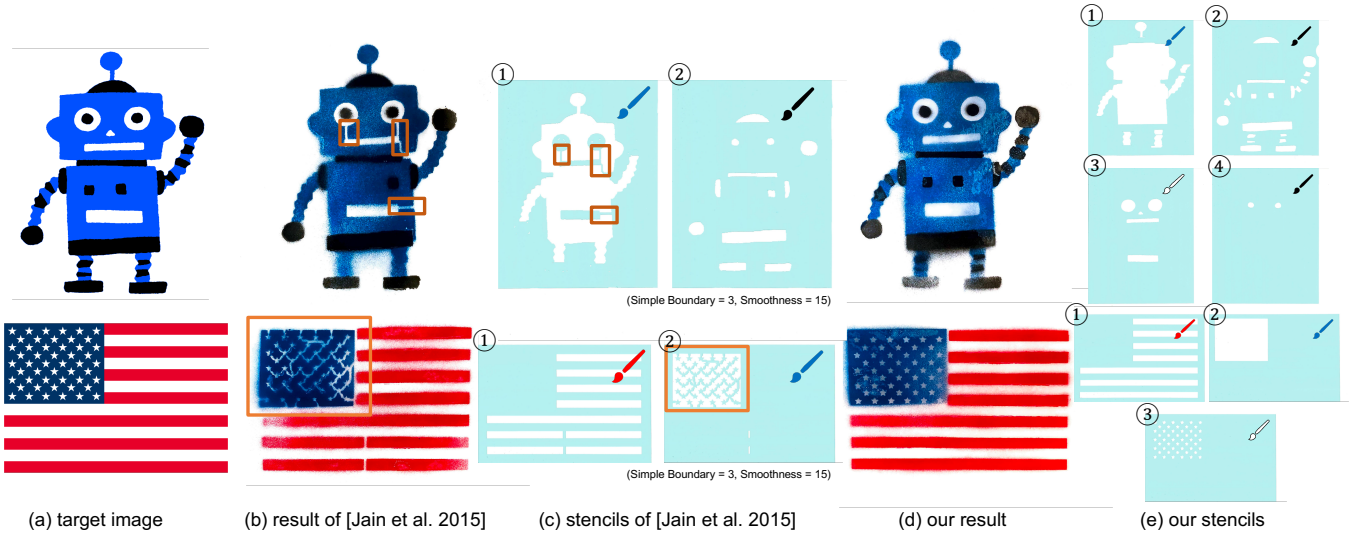

Figure 1: Given a target image (a), we show the painted stencil art and the stencil sheets of Jain *et al.* [Jain et al. 2015] ((b)-(c)) and our *OVERPAINT* method ((d)-(e)). The orange boxes ((b)(c)) highlight the artifacts introduced by the bridges. The brush color in the upper right and the number in the upper left ((c)(e)) indicate the painting color and the painting order of the stencil sheet.

ABSTRACT

Stencil art is a drawing technique for reproducing designs by repeatedly painting pigments through stencil sheets. Each stencil sheet is a connected surface maintained via thin connections called bridges. Existing computer-aided stencil art creation studies focus on choosing a set of bridges that causes the least amount of distortion while fully connecting all islands to the sheet. Unlike previous works, we propose a novel method called *OVERPAINT* that sidesteps the use of bridges by overpainting pigments. Our method automatically generates an ordered set of bridge-free stencil sheets from an input image. By painting sequentially, one can obtain a result resembling the appearance of the input image. We demonstrate our method on several artworks and a user study. The results show that our method enables the user to create visually precise and delicate stencil arts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SA '22 Technical Communications, December 6–9, 2022, Daegu, Republic of Korea
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9465-9/22/12…$15.00
https://doi.org/10.1145/3550340.3564217

CCS CONCEPTS

• **Computing methodologies** → **Computer graphics**.

KEYWORDS

computer-aided art, stencil generation

ACM Reference Format:
Yuta Fukushima, Anran Qi, I-Chao Shen, and Takeo Igarashi. 2022. OVERPAINT: Automatic Multi-Layer Stencil Generation without Bridges. In *SIGGRAPH Asia 2022 Technical Communications (SA '22 Technical Communications), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3550340.3564217

1 INTRODUCTION

Stencil art is an artistic technique to paint a picture with stencils, which are sheets with cut-out areas, called holes. During the creation process, artists paint pigments through the holes on the stencil

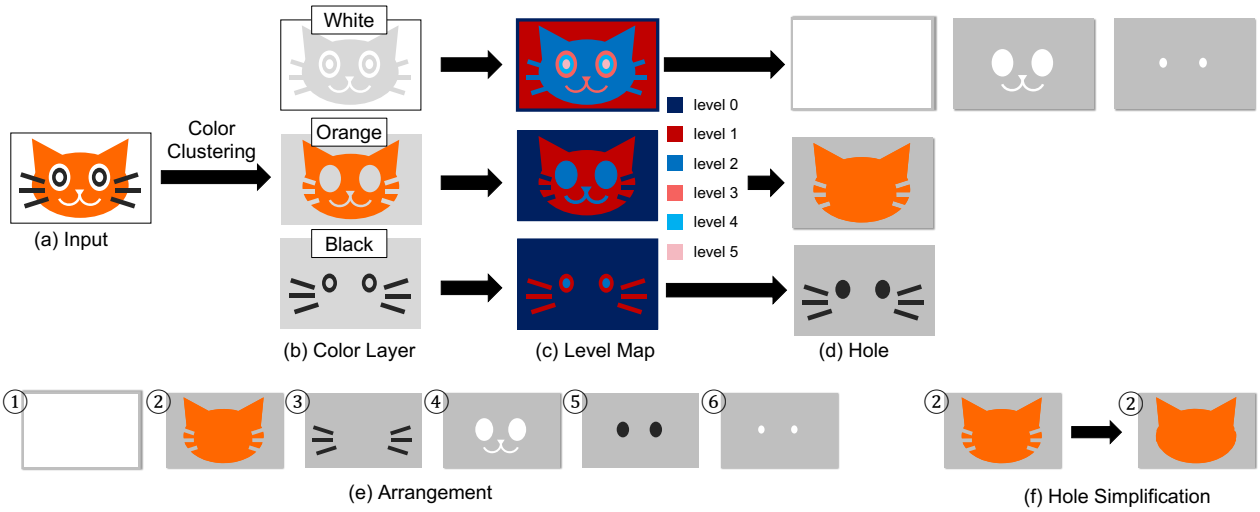

Figure 2: Given an input image (a), we decompose the image by color to create color layers (b). Next, for each layer, we assign levels to its components (c). Based on the levels, we generate a set of holes to be punched in stencils (d) and arrange them (e). Finally, we dilate the holes to simplify their outlines for easy cutting (f).

on any surface, such as walls, wood, and fabric. The history of stencil art dates back over 37 thousand years, as evident in Neanderthal cave art found in Spain and it is still a popular art nowadays.

The most challenging part of creating stencil art is to design stencil sheets. Firstly, the sheets should be able to achieve the appearance of the input image after painting. Secondly, each stencil sheet should be an entirely connected surface in order to be reproducible. Isolated components, called islands, need to be connected to the sheet via thin plates, called bridges. However, it is not only tedious to plan for the placement of any necessary bridges in the artwork but also brings a bridge width dilemma in the designing process: shall we choose a wider bridge or a narrower one? A wide bridge is stable to connect the islands with the stencil sheet but disrupts the design and it degrades the aesthetic quality of the created stencil art. A narrow bridge is aesthetically pleasing but is less likely to maintain the physical stability of the stencil. Meanwhile, it could be time-consuming to determine the placement and width of bridges one by one for sophisticated designs.

Computer-aided stencil art creation isolates the stencil sheet design from the artistic design process and solves it computationally. This enables artists to focus on expressing their ideas. The existing studies propose various methods for choosing a set of bridges that causes the least amount of distortion while still fully connecting all islands to the sheet for monochromatic [Bronson et al. 2008; Igarashi and Igarashi 2010] and multicolored [Jain et al. 2015] input images. Different from the previous studies, we present *OVERPAINT*, a novel method that sidesteps the use of bridges by overpainting pigments. Our method automatically generates an ordered set of bridge-free stencil sheets from a multicolored image. By painting sequentially, the stencil sheets could faithfully recover the appearance of the input image. In detail, the input image is firstly decomposed into a set of color layers. Then we build a level map by adopting a modified flood-fill algorithm for each color layer. Finally, a set of bridge-free stencil sheets is generated by leveraging the inclusion relations among holes calculated from the level map. Example stencil arts created by our method are shown in Figure 1(d), and our results do not contain bridges compared to [Jain et al. 2015].

2 RELATED WORK

Several monochromatic stencil plate design systems have been proposed to tackle a variety of input modalities. Bronson *et al.* [Bronson et al. 2008] propose an approach to generate stencils from 3D meshes or images. The system allows the user to adjust the desired view, lighting conditions, line thickness, and bridge preferences of the input geometry. The stencil creation algorithm uses multiple metrics to minimize the distortion of the abstracted image caused by stabilizing bridges and find the best fit bridge connections. Igarashi *et al.* [Igarashi and Igarashi 2010] present a system that takes freeform strokes as input and enables the user to directly edit stencil using a combination of brush and fill strokes on a canvas screen. The system detects the isolated island and automatically connects it to the other part of the stencil with bridges. These methods use bridges to maintain the stencil's connectivity. In contrast, our method takes the colorful image as input and automatically generates stencil sheets without bridges by computing an optimal ordering of the stencil sheets. Jain *et al.* [Jain et al. 2015] is the most similar to us. It considers both building bridges and removing islands to generate a multi-layer stencil by minimizing an energy function of image segmentation.

3 METHODS

The proposed *OVERPAINT* method first decomposes the input image into different color layers. Given those color layers, our method constructs a level map by adopting a modified Flood-fill algorithm for each color layer. Finally, we generate an ordered set of bridge-free stencils by leveraging the enclosure relations of different holes computed from the level map. Figure 2 illustrates the overview of the *OVERPAINT* method.

3.1 Color Decomposition

We adopt a modified K-Means clustering [Zhang et al. 2017] to decompose the input image I into k color layers. The value of k is a user-defined hyperparameter. Following Zhang *et al.* [Zhang et al. 2017], we compute a color palette which contains the most representative k colors in the input image and cluster pixels into those colors. Then for each color in the palette, we generate a color layer whose size is the same as the input image. The pixel value of position (i, j) is assigned to the current palette color if it belongs to the current color's cluster; background color otherwise. We set the color of the border pixels into background color in order to physically frame each stencil sheet.

3.2 Level Map Construction

We then compute a level map for each color layer (Figure 2 (c)). The level map represents the enclosure relationship among connected regions in each color layer. For each color layer, we first perform a Flood-fill algorithm in up, down, left, and right directions so that contiguous pixels of the same color are assigned to the same region. After this, the level map is built by recursively identifying the outermost regions. We define the outermost region as the root node of the tree. From the root node, recursively make the region a child node if it is adjacent to the current node. Therefore, the level of a region is the same as the depth of the corresponding node in the tree. After constructing the level map, we classify each level into island level or hole level based on their color, i.e.,

$$l_n = \begin{cases} \text{hole level} & l_n\text{'s color} = \text{palette color} \\ \text{island level} & l_n\text{'s color} \neq \text{palette color} \end{cases} \quad (1)$$

As we could observe, the odd levels are recognised as the hole levels (shades of red color in Figure 2 (c)), even levels as island levels (shades of blue color in Figure 2 (c)).

3.3 Stencil Generation

Instead of building bridges to connect the isolated islands, we design a set of stencil layers and compute an optimal order for them. So that by painting sequentially, these stencil sheets could faithfully recover the appearance of the input image.

To fulfill the connectedness constraint for each stencil layer, we propose to make the hole include both interior hole levels and island levels. More specially, for the color s's level map, let $\{A^s_n\}$ be the set of regions in the l_n level and $\{H^s\}$ be the set of holes in the output stencils (Figure 2 (d)). Then, for each hole level i in the color s's level map, we define

$$H^s_i = (A^s_{B^s_i} \cup A^s_{B^s_i+1} \cup \ldots \cup A^s_{m_s}) \quad (2)$$

where m_s is the maximum level in the color s's level map and B^s is the set of hole levels, i.e., odd levels, in the color s's level map.

After obtaining $\{H^s | s = 1, \ldots, K\}$, we assign ordering numbers to these holes. The arrangement of holes depends on enclosure relations related to other holes. For instance, hole A inside hole B has to be painted after hole B, so that pigments through the hole A will not be covered by that of hole B. We arrange and cut the holes via greedy scheduling in a user-defined color order (white-orange-black in Figure 2) and ignore the hole whose area is not greater than 0.01 percent of that of the stencil sheet so as not to explode the number of stencils. Note, we might need to split the hole into different sheets in the cutting process if it contains multiple enclosure relationships with other holes (Figure 2 (e) ③⑤).

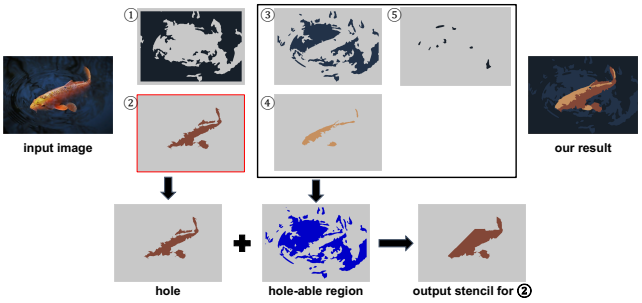

Figure 3: Illustration of the hole-able region and hole simplification.

3.4 Hole Simplification

Up till this point, we can generate stencil sheets and create stencil art with them. However, we argue that besides being devoted to the design, a stencil should have a simple geometry in practice, i.e. simple boundaries, as the cutting cost is proportional to the complexity of the boundary. Thus, we simplify the contour that will be covered by other sheets via a simplification operation. Since the pigments painted later will overlay on the expanded contour, this operation will not change the final appearance and makes the stencil physically stable and easier to cut.

For simplification, we first look for *hole-able regions*, which will be overpainted later and thus converting it into holes will not change the appearance of the final product. For example, given two stencil sheets S_a and S_b, suppose that they are put on a canvas in the order of S_a, S_b. In this case, we can punch a region in S_a which is a hole in S_b but not in S_a. Now, we define this region as a hole-able region. To achieve the hole boundary simplification, we apply a 3×3 filter to every hole-able pixel and merge it into the hole only if there is greater than or equal to four hole pixels. As shown in Figure 3, by using this filter, hole borders will converge on straight lines that stretch to one of the eight directions: cardinal and ordinal directions. This enables users to cut the sheet easily and quickly.

4 RESULTS

Our *OVERPAINT* method creates faithful and physically valid stencils. We show the real-world stencil arts created with our method (Figure 4) and conduct a user study to demonstrate the validity of our method. We use polypropylene plates as stencil sheets and cut them by a programmable cutting machine[1]. We paint the sheets with untransparent pigments such as stencil sprays (for results in Figure 1) and acrylic paints (for results in Figure 4). In practice, we dry the stencil canvas after each painting to avoid the mixture of pigments.

4.1 User study

We conducted a user study to compare the visual quality and usability (easiness of drawing) between our method and Jain *et al.* [Jain

[1] Silhouette Cameo 4

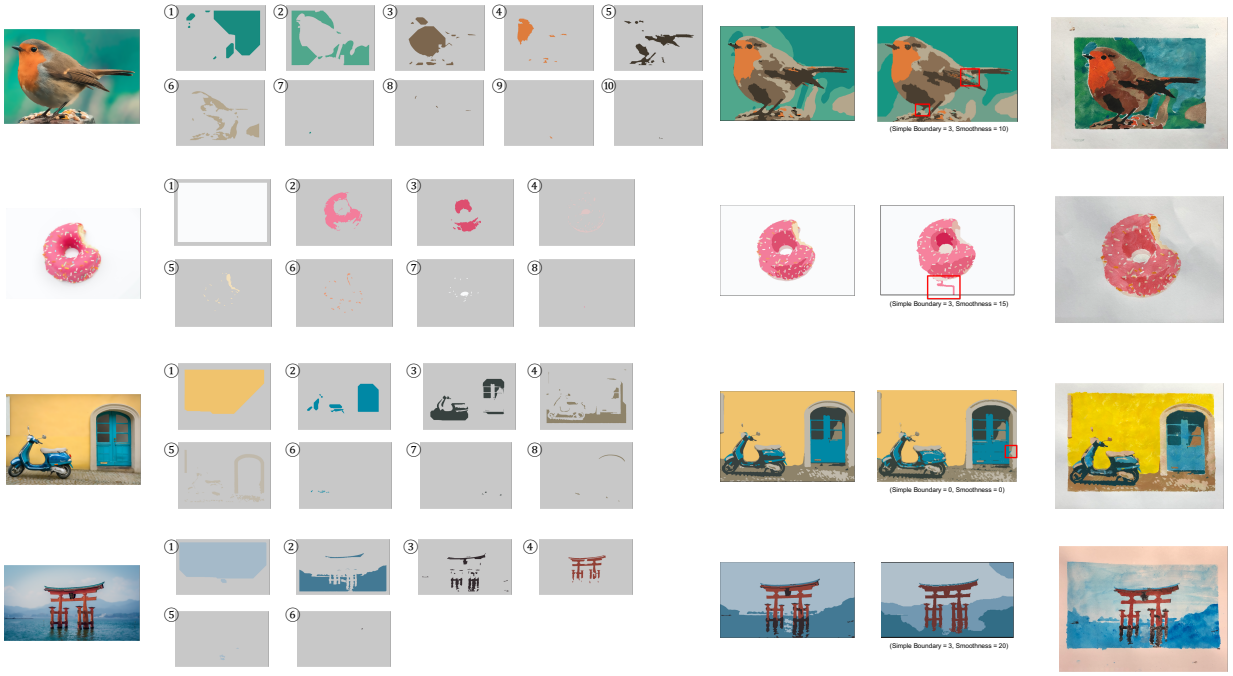

Figure 4: Given an input image (a), we show the stencil sequence generated by our *OVERPAINT* method (b), the image rendered using the stencil sequence by *OVERPAINT* (c) and Jain *et al.* [Jain et al. 2015] (d) and the stencil art painted physically (e). The red boxes in (d) indicate the artifacts caused by bridges.

et al. 2015]. We recruited three participants. Firstly, we asked each participant to choose a motif image as input. Secondly, we generated two different sequences of stencils from the input image using our method and Jain *et al.* [Jain et al. 2015]. In our method, we fix the number of the color layer to three during color decomposition. Then, each participant painted two pictures with the generated stencils and color sprays. After painting the pictures, we asked them to rate both methods in terms of visual quality and usability. We show the artworks the participants create in the study in Figure 1.

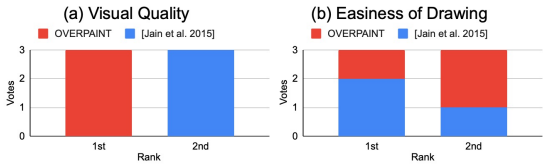

Figure 5: The user ratings of visual quality and easiness of drawing.

As shown in Figure 5 (a), *OVERPAINT* method is favored by all participants in terms of visual quality. This result shows that the *OVERPAINT* method can generate faithful stencil arts with detailed local features of an original image. Figure 5 (b) suggests that more participants feel the stencils generated by [Jain et al. 2015] are easier to draw than the stencils generated by the *OVERPAINT* method. This is mainly because our method generates more stencil sheets in return for better visual quality and physical stability.

5 CONCLUSION AND FUTURE WORK

We propose a novel method, *OVERPAINT*, which automatically generates a series of multicolored stencils without bridges from any input image. Our method assists people to create visually more precise and delicate stencil arts than the previous method.

Our method occasionally misses some semantic features such as eye highlights when dividing the input image into color layers by k-means. Existing studies [Gerstner et al. 2012] handle this problem by additional user annotation modeled as an *importance map*. In the future, we would like to enable user annotation during our color decomposition process. On the other hand, our method generates more stencil sheets compared to Jain *et al.* [Jain et al. 2015]; thus, it takes more time to paint the stencil art using our result. We plan to investigate the possibilities of designing aesthetically pleasing bridges to connect islands.

REFERENCES

Jonathan Bronson, Penny Rheingans, and Marc Olano. 2008. Semi-automatic stencil creation through error minimization. In *Proceedings of the 6th NPAR*. 31–37.
Timothy Gerstner, Doug DeCarlo, Marc Alexa, Adam Finkelstein, Yotam I Gingold, and Andrew Nealen. 2012. Pixelated image abstraction. In *NPAR@ Expressive*. 29–36.
Yuki Igarashi and Takeo Igarashi. 2010. Holly: A drawing editor for designing stencils. *IEEE CG&A* 30, 4 (2010), 8–14.
Arjun Jain, Chao Chen, Thorsten Thormählen, Dimitris Metaxas, and Hans-Peter Seidel. 2015. Multi-layer stencil creation from images. *Comput.Gr.* 48 (2015), 11–22.
Qing Zhang, Chunxia Xiao, Hanqiu Sun, and Feng Tang. 2017. Palette-based image recoloring using color decomposition optimization. *IEEE TIP* 26, 4 (2017), 1952–1964.