ICIC Express Letters
Volume **16**, Number **9**, September **2022**

 ISSN 1881-803X
pp. 965–972

GENERATING VARIABLE-SIZE PIXEL ART BASED ON ROW/COLUMN CLUSTERING

Zilin Xiao[1], Kohei Inoue[1,*], Kenji Hara[1], Naoki Ono[1] and Toru Hiraoka[2]

[1]Faculty of Design
Kyushu University
4-9-1 Shiobaru, Minami-ku, Fukuoka 815-8540, Japan
xiao.zilin.827@s.kyushu-u.ac.jp; { hara; ono }@design.kyushu-u.ac.jp
*Corresponding author: k-inoue@design.kyushu-u.ac.jp

[2]Faculty of Information Systems
University of Nagasaki
1-1-1 Manabino, Nagayo-cho, Nishi-Sonogi-gun, Nagasaki 851-2195, Japan
hiraoka@sun.ac.jp

Received February 2022; accepted April 2022

Abstract. *We propose a method for generating variable-size pixel art from a given image. The variable-size pixel art is a relaxed version of conventional pixel art composed of regular square patterns. By performing row/column clustering, we generate rectangles of various sizes on an image. Then, applying a post processing with an emboss filter, we add a relief effect to the generated rectangle pattern. Experimental results show an artistic effect generated by the proposed method which imitates an impression of a contemporary pixel art.*

Keywords: Variable-size pixel art, Row/column clustering, Rolling guidance filter, Emboss filter, Non-photorealistic rendering

1. **Introduction.** Pixel art is a form of digital art that originated in the dawn of computer games in the late 1970s, and the term 'pixel art' is first published in 1982 [1]. Figure 1 shows a binary pixel art from Space Invaders, a classic arcade game[1]. Recently, pixel art has been recognized as an art form, and has an impact on contemporary art. It may be a challenging and interesting task to represent an image content with a limited number of pixels and colors not only for professional artists but also for amateur creators. Yu presented a tutorial for creating a 96×96 pixel character sprite, where the basic steps are summarized as follows: sketch, color, shade and polish [2]. On the other hand, a number of automatic methods for image pixelization were also proposed by researchers. Fan proposed a private image pixelization method for sharing pixelized images with rigorous privacy guarantees [3]. Kopf et al. proposed a content-adaptive image downscaling method that adapts the shape of its downsampling kernel, yielding sharper and more detailed downscaled results [4]. Inglis and Kaplan proposed a pixelation algorithm for converting vector line art into pixel line art [5]. Kopf and Lischinski proposed an algorithm for generating a resolution-independent vector art from a pixel art [6]. Han et al. proposed an unsupervised deep learning method for pixelization to relieve the requirement for preparing paired pixel art training data [7]. These researches suggest that pixel art and image pixelization attract wide interest from people. Furthermore, a number of contemporary artists have been inspired by such digital art, and have created new expression techniques. Adam Lister produces interesting paintings that consist of geometric

DOI: 10.24507/icicel.16.09.965

[1]By Axel Pixel – Space Invaders (le jeu vidéo d'arcade original, 1978 Taito Corporation par Tomohiro Nishikado), CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=26855200.

FIGURE 1. Pixel art alien

interpretations of iconic imagery and pop culture references [8]. His work reminds us of the image of pixel art, because he divides a canvas into many segments with vertical and horizontal lines, and paints each segment in a nearly uniform color.

Inspired by his pixel-artistic artworks, we propose a method for generating variable-size pixel art from a given image in this paper. We make large and small rectangular segments on a given image by a clustering algorithm [9], which is based on the rolling guidance filter (RGF) proposed by Zhang et al. [10]. Different from [9] where synthetic 2D data are used for experiments, we show the results with real color image data in this paper. After image segmentation, we add a relief effect on it using an emboss filter. Experimental results show that the proposed method generates a geometric art form like Adam Lister's one.

The rest of this paper is organized as follows. Section 2 proposes the row/column clustering for making rectangular segments. Section 3 describes the post processing for making the relief effect. Section 4 shows the examples of generated variable-size pixel art. Finally, Section 5 concludes this paper.

2. **Row/Column Clustering.** Let $F = [f_{ijk}] \in \mathbb{R}^{m\times n\times 3}$ be a three-dimensional array that denotes an RGB image with m rows and n columns, where $f_{ijk} \in [0, 1]$ is the (i, j, k)th element of F, f_{ij1}, f_{ij2} and f_{ij3} correspond to the R, G and B components of the (i, j)th pixel, respectively, and $\mathbb{R}$ denotes the set of real numbers. Then we make a set of rows as $\{R_i\}$ where $R_i = [f_{ijk}] \in \mathbb{R}^{n\times 3}$ is an $n \times 3$ matrix for $i = 1, 2, \ldots, m$, and divide it into several clusters by a rolling guidance filter (RGF)-based clustering algorithm [9] as follows: Let $\{U_i\}$ be another set of matrices, where each element $U_i \in \mathbb{R}^{n\times 3}$ corresponds to R_i for $i = 1, 2, \ldots, m$. Then U_i is iteratively updated by

$$U_i^{(t+1)} = \frac{\sum_{k=1}^{m} s\left(U_i^{(t)}, U_k^{(t)}\right) R_k}{\sum_{k=1}^{m} s\left(U_i^{(t)}, U_k^{(t)}\right)}, \tag{1}$$

where the superscript t denotes the number of iterations for $t = 0, 1, \ldots, T$ where T denotes the maximum number, and $s\left(U_i^{(t)}, U_k^{(t)}\right)$ denotes the similarity between $U_i^{(t)}$ and $U_k^{(t)}$ defined by

$$s\left(U_i^{(t)}, U_k^{(t)}\right) = \exp\left(-\frac{\left\|U_i^{(t)} - U_k^{(t)}\right\|_F^2}{2\sigma_r^2} - \frac{|i-k|^2}{2\sigma_s^2}\right), \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and σ_r and σ_s are positive parameters for controlling the extent of mutual influence between $U_i^{(t)}$ and $U_k^{(t)}$. We initialize U_i as $U_i^{(0)} = R_i$ for $i = 1, 2, \ldots, m$. If R_i and R_k belong to the same cluster, then U_i and U_k get closer to each other as the iterative process progresses. [9] describes an algorithm for clustering vector data. On the other hand, in this paper, we handle three-dimensional array data, that can be generalized to multi-dimensional arrays.

After that, we label each row with its cluster ID number by Algorithm 1, and obtain a set of labels $\{l_i^{\text{row}}\}$.

Algorithm 1 Labeling cluster ID numbers

Require: a set of matrices $\left\{U_i^{(T)}\right\}$, a threshold value $\theta > 0$
Ensure: a set of labels $\{l_i^{\text{row}}\}$
1: $c \leftarrow 1$
2: $l_1^{\text{row}} \leftarrow c$
3: **for** $i = 2, 3, \ldots, m$ **do**
4: $\quad D \leftarrow \sqrt{\left\|U_i^{(T)} - U_{i-1}^{(T)}\right\|_F^2 \Big/ m}$
5: $\quad$ **if** $D > \theta$ **then**
6: $\quad\quad c \leftarrow c + 1$
7: $\quad$ **end if**
8: $\quad l_i^{\text{row}} \leftarrow c$
9: **end for**
10: **return** $\{l_i^{\text{row}}\}$

Next, we make a set of columns as $\{C_j\}$ where $C_j = [f_{ijk}] \in \mathbb{R}^{m\times 3}$ is an $m \times 3$ matrix for $j = 1, 2, \ldots, n$, and divide it into several clusters in a similar way to the above row clustering for $\{R_i\}$, where let $\{V_j\}$ be another set of matrices corresponding to $\{C_j\}$, and then each V_j is iteratively updated by the similar equation to (1). Let $\left\{l_j^{\text{col}}\right\}$ be a set of labels assigned to columns $j = 1, 2, \ldots, n$ as a result of column clustering.

3. **Generating Variable-Size Pixel Art.** Assume that we have obtained two sets of cluster labels $\left\{l_i^{\text{row}}\right\}$ and $\left\{l_j^{\text{col}}\right\}$ by the above row/column clustering procedure, and let M and N be the numbers of clusters in row and column data, respectively. Then we can segment the image plane of F into $M \times N$ rectangles using cluster labels $l_i^{\text{row}} \in \{1, 2, \ldots, M\}$ and $l_j^{\text{col}} \in \{1, 2, \ldots, N\}$. For each rectangle with a pair of labels $\left(l_i^{\text{row}}, l_j^{\text{col}}\right)$, we compute a mean color as

$$\bar{f}_{l_i^{\text{row}}, l_j^{\text{col}}, :} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l_i=1}^{M}\sum_{l_j=1}^{N}\delta_{l_i,l_i^{\text{row}}}\delta_{l_j,l_j^{\text{col}}}f_{i,j,:}}{\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l_i=1}^{M}\sum_{l_j=1}^{N}\delta_{l_i,l_i^{\text{row}}}\delta_{l_j,l_j^{\text{col}}}}, \tag{3}$$

where $\delta_{l_i,l_i^{\text{row}}}$ denotes the Kronecker delta defined that $\delta_{l_i,l_i^{\text{row}}} = 1$ if $l_i = l_i^{\text{row}}$, and 0 otherwise, and $f_{i,j,:}$ means an RGB color vector $[f_{i,j,1}, f_{i,j,2}, f_{i,j,3}]$. Then we have a segmented image $S = [s_{i,j,:}]$ for $s_{i,j,:} = \bar{f}_{l_i^{\text{row}}, l_j^{\text{col}}, :}$.

Next, we convert the color space from RGB to YIQ as

$$\begin{bmatrix} y_{ij} \\ i_{ij} \\ q_{ij} \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} s_{i,j,1} \\ s_{i,j,2} \\ s_{i,j,3} \end{bmatrix}, \tag{4}$$

from which we have a grayscale image $Y = [y_{ij}]$. Then we apply an emboss filter to Y to have a relief effect, i.e., the embossed result $\tilde{Y} = [\tilde{y}_{ij}]$ is given by

$$\tilde{y}_{ij} = \sum_{k=-1}^{1}\sum_{l=-1}^{1} w_{k+2,l+2} y_{i+k,j+l}, \tag{5}$$

where $w_{k+2,l+2}$ denotes the coefficients for emboss filtering given by

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}. \tag{6}$$

Then we normalize the values of $\tilde{Y}$ by

$$z_{ij} = \frac{\tilde{y}_{ij}}{\max\{|\tilde{y}_{ij}|\}}, \tag{7}$$

from which we have a normalized embossed result $Z = [z_{ij}]$. For generating a three-dimensional appearance on the image, we add Z to Y as follows:

$$Y' = Y + \alpha Z, \tag{8}$$

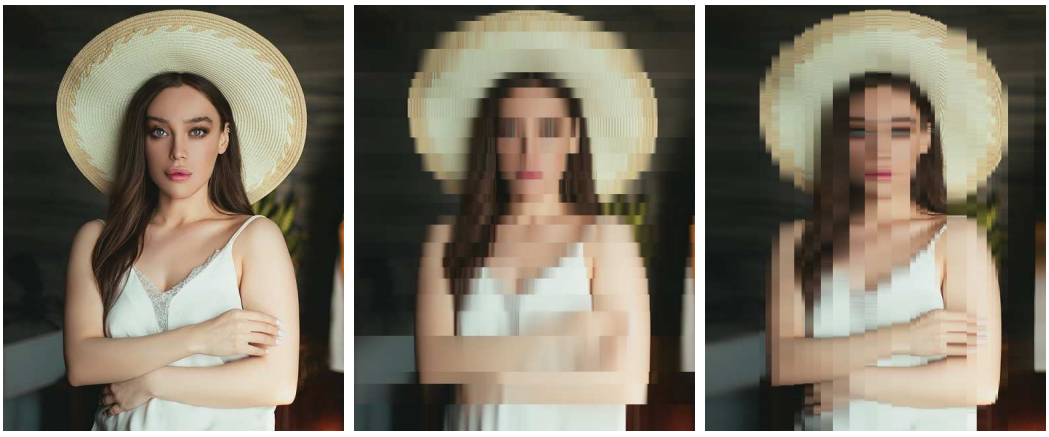
where α is a parameter for controlling the intensity of the emboss effect. Finally, we replace Y with $Y' = \left[y'_{ij}\right]$, and get back to the original RGB color space as

$$\begin{bmatrix} s'_{i,j,1} \\ s'_{i,j,2} \\ s'_{i,j,3} \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} y'_{ij} \\ i_{ij} \\ q_{ij} \end{bmatrix} \tag{9}$$

to have a modified RGB image $S' = \left[s'_{ijk}\right]$. Additionally, to avoid a gamut problem, we make a modification on S' to have the final result $S'' = \left[s''_{ijk}\right]$ by clipping the values as

$$s''_{ijk} = \min\left(\max\left(0, s'_{ijk}\right), 1\right) \in [0, 1]. \tag{10}$$

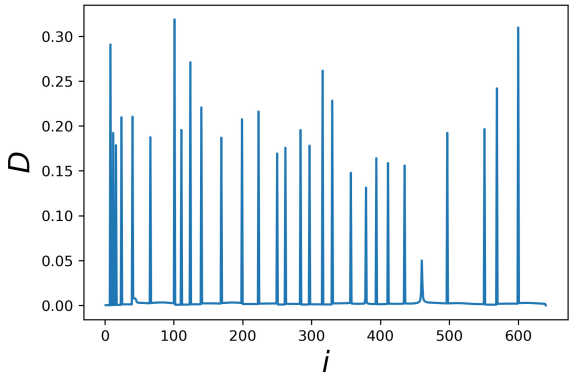
4. **Experimental Results.** In this section, we show some examples of variable-size pixel art generated by the above proposed method. Figure 2 shows a result of row/column clustering by the above RGF-based algorithm, where we set that $\sigma_r^2 = 3ns_r^2$ for row clustering and $\sigma_r^2 = 3ms_r^2$ for column clustering with $s_r = 1/25$, and $\sigma_s = 20$ for both clustering. The number of iterations T for computing (1) iteratively is 30, which is an enough number for our requirement. Figures 2(b) and 2(c) show the results of row and column clustering, where $U_i^{(T)}$ and $V_j^{(T)}$ are used instead of R_i and C_j, respectively.



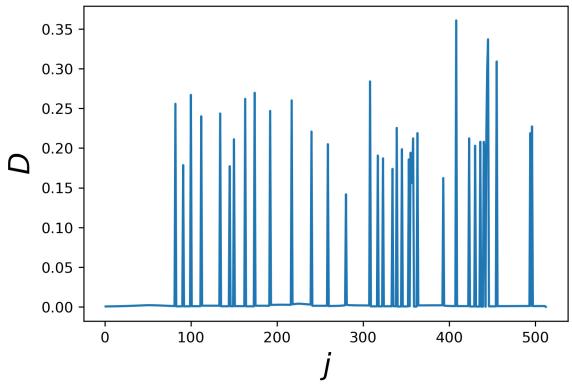
(a) Original (b) Row clustering (c) Column clustering

FIGURE 2. Row/Column clustering: (a) Original image with 640×512 pixels, (b) rows clustered into 29 clusters, and (c) columns clustered into 40 clusters

After row/column clustering, we assign cluster IDs to each row and column by Algorithm 1. The distances D in Algorithm 1 between adjacent rows or columns are shown in Figures 3(a) and 3(b), respectively, where the vertical and horizontal axes denote D and row number i or column number j. These figures show that the distances between clusters are relatively larger than that between rows or columns within a cluster. From this observation, we set that $\theta = 0.05$. Then Algorithm 1 outputs the sets of labels $\{l_i^{\text{row}}\}$ and $\left\{l_j^{\text{col}}\right\}$, with which we compute a mean color for each rectangle by (3), and obtain a segmented image shown in Figure 4(a), where each rectangle is painted in the mean color.

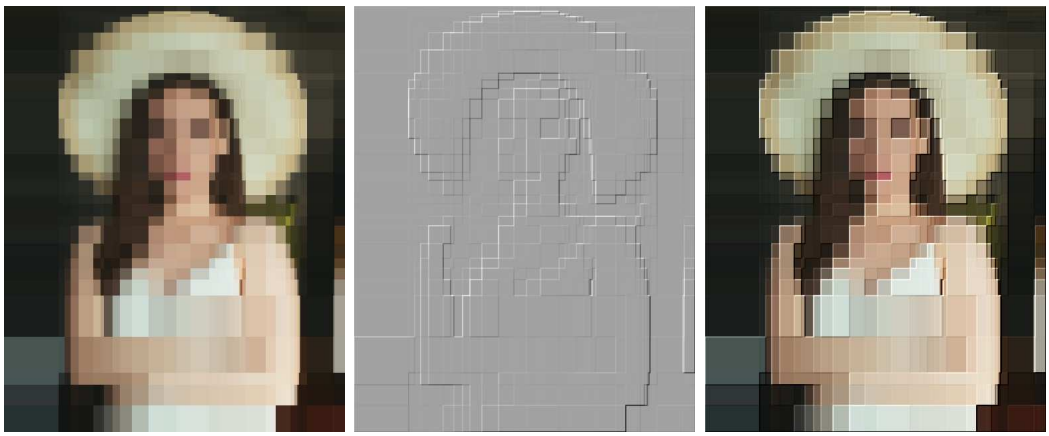


(a) Row clustering



(b) Column clustering

FIGURE 3. Distance between adjacent rows or columns vs the indices of rows (a) or columns (b)



(a) Segmented image (b) Emboss-filtered (c) Relief effect

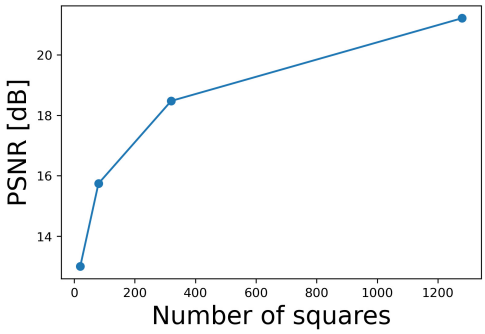
FIGURE 4. Generating variable-size pixel art: (a) Segmented image with variable-size rectangles painted in mean colors, (b) emboss-filtered result for Y component of the segmented image, where the values are linearly transformed for imaging, and (c) final result with relief effect

In our experiments, we determined the values of parameters manually to obtain similar results to Adam Lister's artworks.

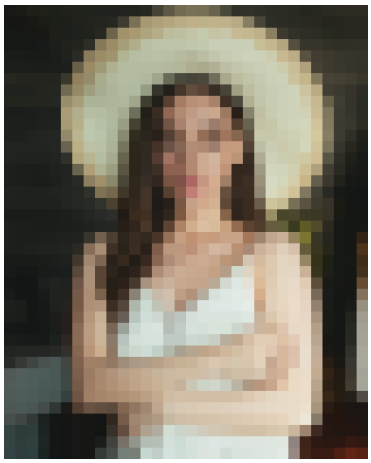
Next, we convert the color space of the segmented image from RGB to YIQ by (4), and apply an emboss filter given by (5) and (6) to the Y component, the result of which

is shown in Figure 4(b). Then the output of emboss filter is normalized as (7), which is scaled by $\alpha = 1.5$ and added to the original Y component to have the modified Y component. Finally, we go back to the original RGB color space as (9), and avoid a gamut problem by (10). Consequently, we obtain a variable-size pixel art with relief effect as shown in Figure 4(c).

We compared the obtained segmented image in Figure 4(a) with a conventional image composed of regular squared pixels. Figure 5(a) shows a graph of peak signal-to-noise ratio (PSNR) between the regular squared pixel art and the original image in Figure 2(a), where the vertical and horizontal axes denote PSNR and the number of squares, respectively. This graph shows the monotonic increase in PSNR according to the improvement of the image resolution. For example, an obtained pixel art with 1280 squares is shown in Figure 5(b), which is compared with our result in Figure 4(a) as summarized in Table 1, where our result (denoted by 'Variable size') achieves higher value of PSNR, which means that our result is closer to the original image than the conventional result in Figure 5(b), with smaller number of segments than the regular square pixel art.



(a) PSNR vs No. of squares



(b) Regular square pixel art with 1280 segments

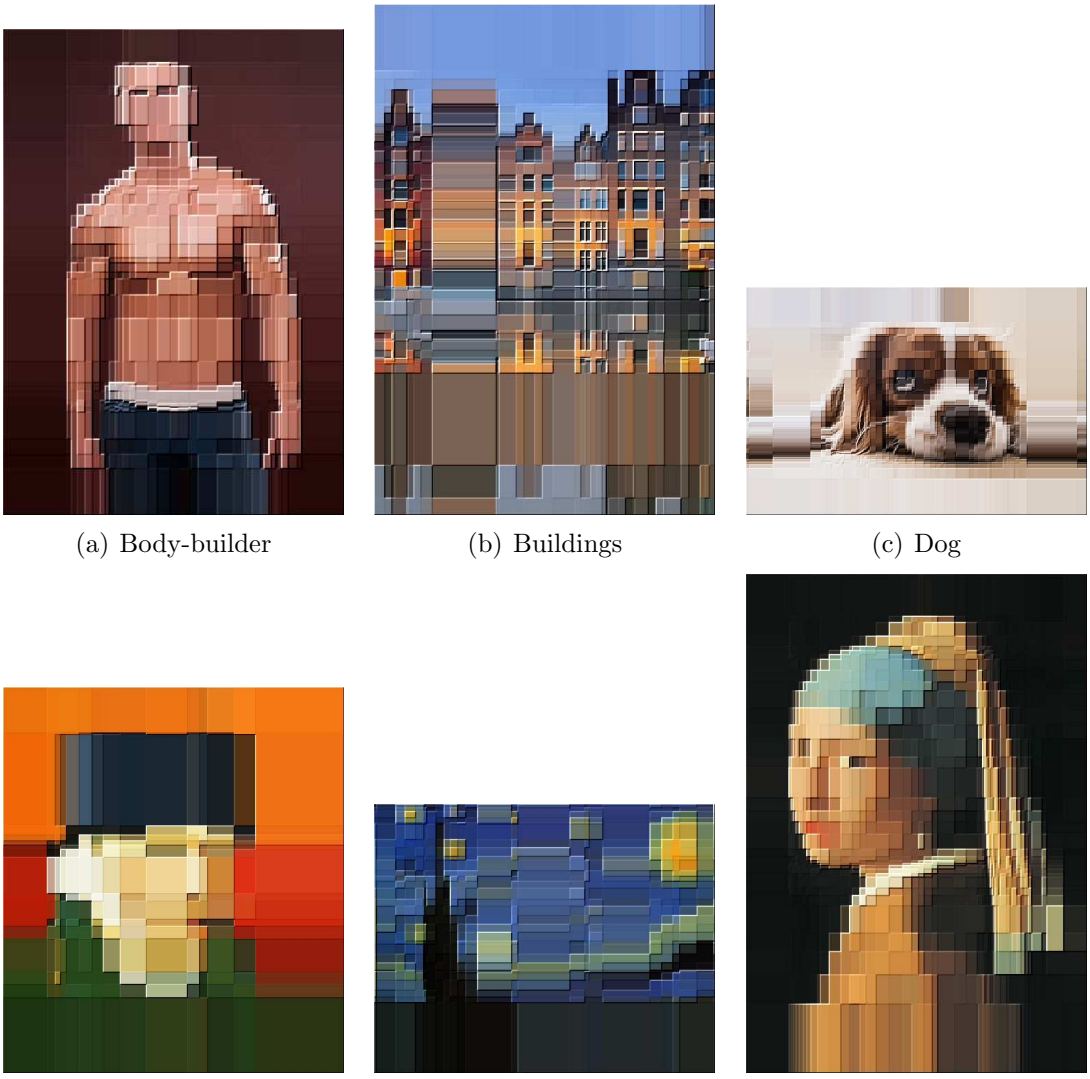
FIGURE 5. Quantitative and visual qualities of regular square pixel art: (a) Peak signal-to-noise ratio (PSNR) monotonically increases with the number of squares, and (b) the image with 1280 squares captures the outline of the original image in Figure 2(a)

TABLE 1. Peak signal-to-noise ratio (PSNR): The proposed variable-size pixel art (Figure 4(a)) achieves higher PSNR than that of regular square pixel art (Figure 5(b)) with smaller number of segments

	No. of segments	PSNR [dB]
Regular square	1280	21.21
Variable size	**1160**	**22.20**

Figure 6 exhibits six results generated from photographs (6(a)-6(c)) and paintings (6(d)-6(f)). In these examples, we can see that the image contents are roughly expressed by a variety of rectangles with relief effect, which yield a geometric composition similar to Adam Lister's artworks.

5. **Conclusions.** In this paper, we proposed a method for generating variable-size pixel art from a given image. The proposed method uses a clustering algorithm based on rolling guidance filter for making rectangular segments. The post processing with an emboss filter



(a) Body-builder (b) Buildings (c) Dog

(d) Self-portrait with bandaged ear and pipe (e) The starry night (f) Girl with a pearl earring

FIGURE 6. Variable-size pixel art: (a) 576×404 pixels, $s_r = 1/30$, $\sigma_s = 20$, (b) 720×480 pixels, $s_r = 1/20$, $\sigma_s = 20$, (c) 512×768 pixels, $s_r = 1/20$, $\sigma_s = 20$, (d) 723×640 pixels, $s_r = 1/16$, $\sigma_s = 20$, (e) 432×550 pixels, $s_r = 1/26$, $\sigma_s = 10$, and (f) 731×500 pixels, $s_r = 1/30$, $\sigma_s = 20$

adds a relief effect to the segmented image. The obtained results imitate Adam Lister's art form.

Future work will include the development of other post processing algorithms for various artistic expressions.

Acknowledgment. This work was supported by JSPS KAKENHI Grant Numbers JP19 K12664 and JP21K11964.

REFERENCES

[1] A. Goldberg and R. Flegal, ACM president's letter: Pixel art, *Communications of the ACM*, vol.25, no.12, pp.861-862, DOI: 10.1145/358728.358731, 1982.

[2] D. Yu, *Pixel Art Tutorial*, https://www.derekyu.com/makegames/pixelart.html, Accessed on 21-April-2022.

[3] L. Fan, Image pixelization with differential privacy, in *Data and Applications Security and Privacy XXXII. DBSec 2018. Lecture Notes in Computer Science*, F. Kerschbaum and S. Paraboschi (eds.), DOI: 10.1007/978-3-319-95729-6_10, 2018.

[4] J. Kopf, A. Shamir and P. Peers, Content-adaptive image downscaling, *ACM Trans. Graphics*, vol.32, no.6, DOI: 10.1145/2508363.2508370, 2013.
[5] T. C. Inglis and C. S. Kaplan, Pixelating vector line art, *International Symposium on Non-Photorealistic Animation and Rendering*, DOI: 10.2312/PE/NPAR/NPAR12/021-028, 2012.
[6] J. Kopf and D. Lischinski, Depixelizing pixel art, *ACM Trans. Graphics (Proc. of SIGGRAPH 2011)*, vol.30, no.4, pp.99:1-99:8, DOI: 10.1145/2010324.1964994, 2011.
[7] C. Han, Q. Wen, S. He, Q. Zhu, Y. Tan, G. Han and T.-T. Wong, Deep unsupervised pixelization, *ACM Trans. Graphics (SIGGRAPH Asia 2018 Issue)*, vol.37, no.6, pp.243:1-243:11, DOI: 10.1145/3272127.3275082, 2018.
[8] *Adam Lister Gallery*, https://adam-lister-gallery.myshopify.com/, Accessed on 21-April-2022.
[9] T. Hattori, K. Inoue and K. Hara, Rolling guidance filter as a clustering algorithm, *IEICE Trans. Inf. & Syst.*, vol.E104-D, no.10, pp.1576-1579, 2021.
[10] Q. Zhang, X. Shen, L. Xu and J. Jia, Rolling guidance filter, *Proc. of the 13th ECCV: European Conference on Computer Vision*, 2014.