Computers & Graphics (2022)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Structure-aware bottle cap art

Leonardo Sacht

ARTICLE INFO

Article history:

Keywords: Bottle cap art, Structural similarity, Image discretization

ABSTRACT

We propose a new image processing problem that consists in approximating an input image with a set of plastic bottle caps. This problem is motivated by the appreciation caused by low resolution imaging, combined with the goal of finding a new destination for plastic caps that would be discarded and possibly damage the environment. Our solution consists in formulating an energy that measures how well a bottle cap art reproduces structures and features from the reference image and maximizing it with a simulated annealing strategy. Examples in the paper show that our method produces high quality results in this very low spatial and color resolution setting.

1. Introduction

Low resolution images such as pixel art are highly appreciated by the large public. Examples include outdoor sculptures such as the Digital Orca at the Vancouver Convention Centre, indoor pixel art containing famous video game and movie characters, and stylized icons displayed on computer and cell phone screens. Although their use emerged in the context of low technological resources some decades ago, low resolution images are now seen as a way to represent the most important information of a scene in a minimalistic and pleasant way.

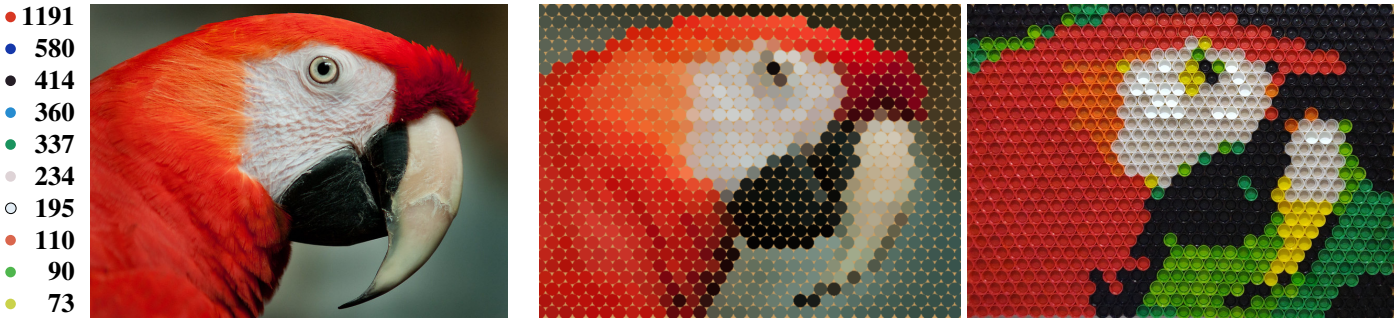
To the best of our knowledge, this paper is the first to investigate a new form of generating images with very low color and spatial resolution: it tackles the problem of arranging plastic bottle caps on a rectangular canvas to approximate a given input image. Instead of using resources that are harmful to the environment such as energy, inks, or pieces made-up for large mosaics, our image generation process is constrained to arrange plastic bottle caps that could otherwise be inappropriately discarded and cause environmental damage, such as the death of sea animals that ingest them [1, 2].

Despite existing initiatives to re-use bottle caps to make art, these usually demand skilled artists, hundreds of thousands of caps, and large spaces (on the scale of meters) to arrange them [3]. This paper proposes a fully automatic method to generate bottle cap arts such that physically assembling them is possible for inexperienced users. Moreover, most of the results in this paper have a scale of one meter and use caps in a number that can be gathered by a single person or a small group of people.

Figure 1 summarizes our method. The input for the method is an image, a set of bottle caps with specific colors and quantities (a), and the desired number of caps in the horizontal extent (32 in this example) which determines the size (101 cm wide, 73 cm high in this example) of the final bottle cap art (c). The first step maps the input image to the given spatial resolution (b), disregarding cap color constraints. This intermediate full palette discretized image is used as the reference to calculate the most similar bottle cap art that satisfies the given constraints. This similarity is measured by an adaptation of the Structural Similarity (SSIM) metric [4] and the optimal placing of the caps is obtained using the simulated annealing strategy [5]. After the optimization, the user is given a set of instructions to physically assemble the result (c), which contains only caps that were given as input. In other words, our method works with the resources that are available and does not require the user to collect or produce additional caps.

The contributions of our paper are the following:

- A new image processing problem: given an input image and a set of colored disks of fixed size (the bottle caps), arrange them on a rectangular region to approximate the input image.
- A solution for this problem: a metric to assess the similarity between the desired bottle cap art and a reference image and an optimization procedure to maximize this similarity.



(a) Given bottle caps and input image (b) Spatial discretization (c) Output bottle cap art

Fig. 1. The user provides an input image and a set of plastic bottle caps (a). Our method first calculates a full palette bottle cap image (b) to then optimize the placements of the caps and obtain a result (c) that is as similar as possible to (b), while using only the caps given in (a). Input is a cropped version of an image by Christopher Gabbard obtained from flickr.com under CC BY-SA 2.0.

2. Related work

The problem of depicting images with limited spatial or color resolution has a long history in computer graphics. This problem was originally approached in the context of hardware limitations and then evolved to exploring the representation benefits and artistic values of image abstraction [6, 7, 8].

The most familiar problem in this category is color quantization, which involves approximating a full-color image with a smaller palette. It was largely explored in the 1980s and 1990s [9, 10, 11] when computer displays and data storage were very limited. Most of those quantization techniques required the final result to have at most a given number of levels, without specifying what the final levels should be. Despite we do perform quantization by reducing the number of colors in the image, our problem is more limited since the final color levels are given. Additionally, color quantization techniques did not require decreasing spatial resolution, which is also an extra difficulty in our problem and prohibits the use of dithering.

Mostly motivated by (2D) printing limitations, the problem of depicting images with very few colors (in many cases, only black and white) led to dithering and other halftoning techniques [12, 13, 14, 15, 16, 17, 18, 19]. Although this problem is usually more limited than ours with respect to the final palette, these techniques used the full spatial resolution of the input image to add randomness to color values, an idea that is not possible in our context due to the much more limited spatial resolution our outputs must have.

ASCII art [20, 21] was primarily motivated by the limited possibilities of image generation in the 1980s. This technique displays black ASCII characters over a white background to approximate images. It is related to our problem, since it uses a small number of given shapes and a single color, while ours uses a single shape (disk) and a small number of colors.

More recently, color limitation was explored to generate non-photorealistic approximations of images [22, 23]. For example, Artistic Thresholding [22] gave users the possibility to tune parameters to obtain varying effects in black and white output images. Although the authors dealt with spatial limitations, such issues were not as severe as in our case.

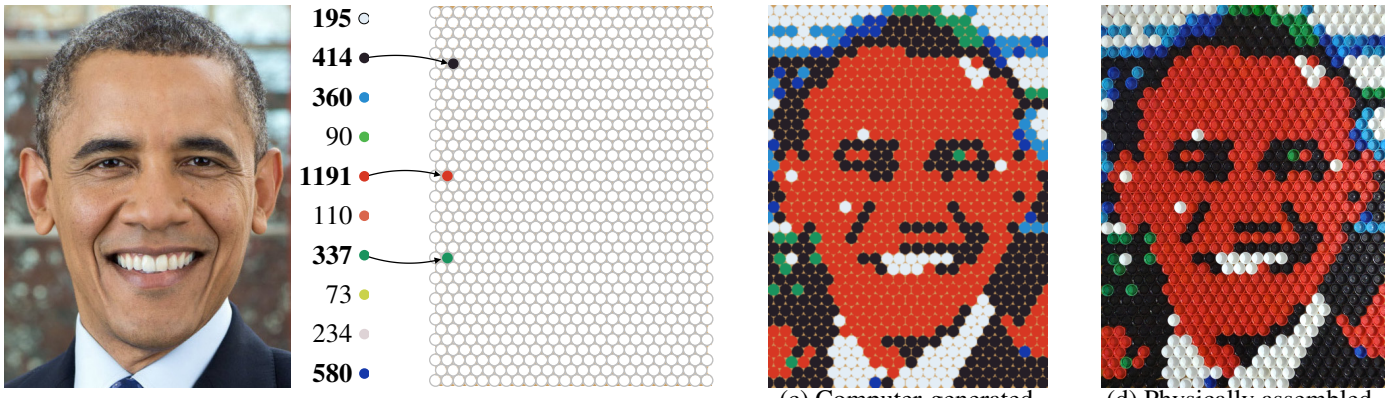
Pixel art generation [24, 25, 26] is a problem close to ours in the sense that it determines images with both very limited spatial and color resolutions. In contrast to our problem, only the number of output color levels is given, instead of the levels themselves, and in most cases, pixel arts have higher spatial resolutions. A more detailed discussion comparing our method to pixel art is provided in Section 6.1 and Figure 14. The method proposed in [27] physically assembles pixel art with LEGO bricks and considers the additional constraint of a given palette, but does not impose a limitation on the number of bricks of each color. In contrast, our method considers the limitation on the number of caps of each color. Image downscaling [28, 29] is another related problem, but it does not correspond with the color limitations that we have.

Bottle cap art calculation can be thought of as segmenting an image into regions to fill each resulting region with caps of a single color. Graphcut strategies [30, 31] could be considered in this context, but we found that it is very difficult in practice to tune their parameters to achieve good results for our problem. The user-assisted (more precise) option [32] is out of the scope of this research since we want bottle cap art calculation to be fully automatic. An intermediate strategy that we follow in this paper is to segment the input image into a larger number of segments that correspond to bottle cap regions using an adaptation of SLIC superpixels [33] (see Figure 1 (b) for an example and Section 4.1 for details). The resulting segmentation defines a discretized version of the input image that is used as reference to Structural Similarity (SSIM) metric optimization. SSIM [4] has been used for image downscaling [28] and halftoning [18] but, as previously emphasized, our problem involves a much more constrained setting.

Finally, some papers in computer graphics have proposed methodologies that took into consideration environmental impact. These include saving plastic in 3D printing [34, 35, 36] and energy in imaging and rendering [37, 38, 39]. The graphics problem we propose is different in the sense that it is based on environmental impact from its conception, instead of decreasing the impact caused by largely disseminated high resource-consuming technologies.

3. Problem setup

We illustrate our problem statement in Figure 2 (a,b): given a set of bottle caps specified by their colors and quantities, the



(a) Input image and available caps (b) Bottle cap grid (c) Computer-generated bottle cap art (d) Physically assembled bottle cap art

Fig. 2. Our goal is to approximate the input image by placing a selection of the available caps (a) over a rectangular canvas. We choose to arrange the caps on a hexagonal-like lattice called *bottle cap grid* (b) since it has very small spaces between the caps through which the background is visible and makes the final result (d) simpler to assemble. The physical assembling is done following a result computed automatically by our method (c). Input image obtained from whitehouse.gov under Public Domain.

goal is to arrange them over a rectangular canvas to approximate a given input image as closely as possible.

Notice that, except for very simple cases, this is not an easy task for a human: the number of possible arrangements is combinatorially high and the very low spatial and color resolutions make it difficult to reproduce features from the image. For example, starting from an empty canvas, it is not clear that the man's face in Figure 2 (a) occupies an area of approximately 400 caps. A person could try to fill it with orange caps and notice along the way that they are not sufficient. Filling part of it with the 110 available orange caps and the rest with another color would lead to a non-existing seam in the final result. Giving up on the orange caps and starting it over with another color would lead to a waste of time. In addition, the reproduction of some features such as the cheeks and the chin of the man is very difficult for a human in practice. An automatic solution for this problem is then crucial to make it accessible to a large public.

A key choice in this problem is how to arrange the equal-sized caps over the rectangular canvas. We would like the union of the non-overlapping disks to cover the largest possible area of the canvas for the result to have as much as possible of the cap colors and as little as possible of the canvas color. This is an instance of the circle packing problem, for which the hexagonal-like lattice in Figure 2 (b) is optimal [40, 41]. This structure, called in our paper *bottle cap grid*, has the additional advantage of being easy for inexperienced users to assemble: after a careful assembling of the bottom-most row of caps, all the other rows are assembled by simply placing caps to be supported by the two adjacent caps on the previous row.

The resolution of the bottle cap grid is defined by the number of caps on the horizontal extent, which is provided by the user in our method. The number of caps on the vertical extent is automatically determined to obtain an aspect ratio as close as possible to the one of the input image. Given that the standard diameter of a plastic bottle cap is approximately 3.2 cm, these horizontal and vertical resolutions determine the size of the final result. For instance, the one in Figure 2 (d) has 24 caps on the horizontal extent and is 76 cm wide by 101 cm high.

We suggest the use of wasted cardboard as the rectangular canvas, over which the caps can be glued. Placing the caps with their bottom upward (Figure 2 (d)) has the advantages of hiding the beverage brands and being more easily glued to the cardboard.

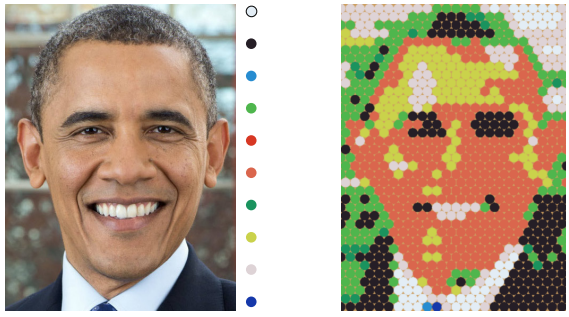
The RGB values of the caps and their quantities are provided by the user as input to our method. The assumption of constant-colored caps simplifies the problem but makes the results generated in the computer (Figure 2 (c)) slightly different from their physical versions (Figure 2 (d)). Different lighting conditions can also affect the perception of the physical results. Throughout this paper, we will use the computer-generated results for didactic purposes and the physically assembled ones for discussions about the final results.

4. Method

A naive approach to our problem would be to loop over the bottle cap grid and assign to each cap region the cap color that is the closest to the average color in the corresponding area of the input image. Even if we ignore the limited number of caps of each color and apply this approach to the image in Figure 3 (a), we obtain a result (b) which clearly fails to reproduce the features from the input image.

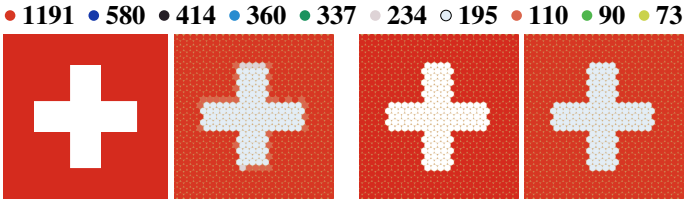
To avoid this problem, we could maximize the Structural Similarity (SSIM, [4]) between the full resolution input image and the desired bottle cap art. Although this solves the problem of missing features, it leads to another problem that is illustrated in Figure 4 (a): the resulting bottle cap art may have geometric distortions due to trying to reproduce statistical measures from the input image at a much coarser spatial resolution.

For these reasons, our approach consists in first calculating a feature-preserving discretization of the input image over the bottle cap grid (Section 4.1) and then running an optimization to maximize the similarity between the final result and this discretized image (Sections 4.3 through 4.5). The former is illustrated in Figure 4 (b, left) and the latter in Figure 4 (b, right). We adapt the SSIM index to our problem and augment it with



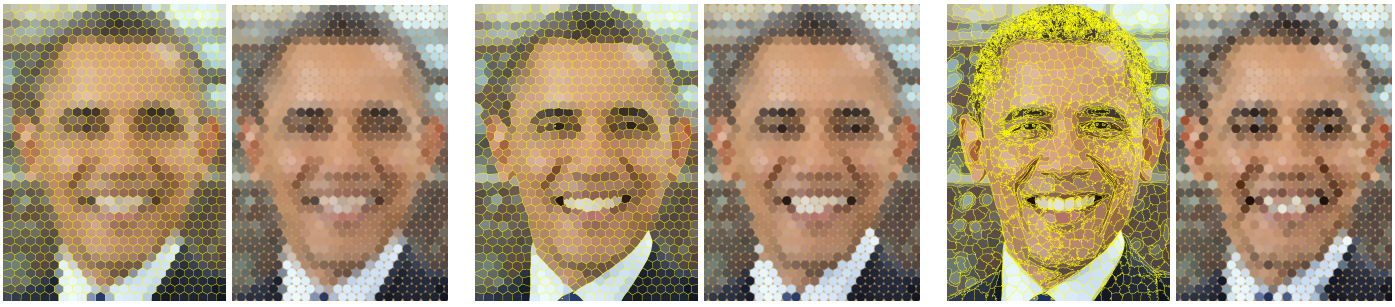
(a) Input image and caps (b) Naive result

Fig. 3. Result of applying a naive approach, assuming no limitation on the number of caps of each color.



(a) (b)

Fig. 4. Maximizing the Structural Similarity with respect to the full resolution input image (a, left) leads to results with geometric distortions (a, right). Our result (b, right) avoids this problem by using as reference for Structural Similarity maximization a discretized version of the input image (b, left). Input available under Public Domain.



(a) $\alpha = 500$ (too regular). (b) $\alpha = 50$ (our choice). (c) $\alpha = 5$ (too adaptive).

Fig. 5. Different choices of the compactness parameters α in the SLIC algorithm. $\alpha = 500$ (a) produces cells (left) that are similar to the bottle cap grid (right), but result in washed-out colors due to lack of adaptivity. On the other hand, $\alpha = 5$ (c) adheres very well to the boundaries of the input image (left) but produces high distortions when mapped to the bottle cap grid (right). We set the intermediate value $\alpha = 50$ (b) as the standard in our method due to its adaptivity and low distortion.

a penalty term to preserve regions that were constant in the reference (discretized) image. Details of these adaptations in the SSIM metric are provided in Section 4.2.

4.1. Spatial discretization

To avoid missing important features from the input image we must have all its pixels contributing to the result of this step. We then segment it in a way that each region of the segmentation corresponds to a bottle cap region and every pixel in the input image belongs to one region. This can be achieved by defining each cap center to be a centroid of a Voronoi diagram that partitions the input image: the pixels from the input image are assigned to the region defined by the (spatially) closest centroid. This results in a regular segmentation of the input image (Figure 5(a)) that is very close to the bottle cap grid. This regularity has a problem though: the regions often contain very different colors and averaging them to obtain the colors for each bottle cap leads to a loss of features. This happens in Figure 5 (a), for example, on the interface between the face of the man and the collar of his shirt.

To obtain a segmentation that better adapts to image boundaries, Achanta et al. [33] propose a method called *SLIC superpixels* that uses a color-space distance to cluster pixels. This method calculates centroids and pixel-to-region assignments that minimize

$$D = \sqrt{d_c^2 + \alpha^2 \left(\frac{d_s}{S}\right)^2}, \tag{1}$$

where d_c is a color distance, d_s is a position distance, S is a normalization factor, and α is a parameter that determines the relative importance between these two terms (please refer to [33] for details).

We show results for different choices of the "compactness" parameter α in Figure 5. For each result, we present the superpixel segmentation obtained minimizing (1) on the left (boundaries between superpixels are shown in yellow) and the corresponding bottle cap image on the right, obtained by assigning to each bottle cap region the average color of the pixels belonging to its corresponding converged region. As expected, a lower value for α leads to a more adaptive segmentation (c), but the corresponding bottle cap image has a lot of distortions, caused by the fact that final regions differ too much from the bottle cap grid. On the other hand, a very high α leads to a segmentation that is very similar to the bottle cap grid (a) but the result ends up having washed-out colors. We then set $\alpha = 50$ (b) for our results, which is a tradeoff between adaptivity and closeness to the bottle cap grid. For a discussion about the effect of α on the final results of our method (after the optimization described in Section 4.5 is applied), please see Section 6.2 and Figure 15.

4.2. Structure-aware metric

In the previous section, we calculated an approximation of the input image over the bottle cap grid (Figure 5 (b), right), but without considering the color limitations of the problem. In this section, we are going to adapt the SSIM metric to work with two

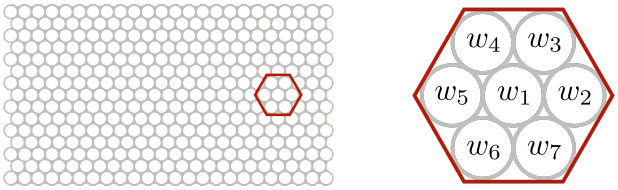


Fig. 6. (a) bottle cap grid with a window highlighted in red. (b) window and weights.

bottle cap images: the full palette one from the previous section (represented by $\mathbf{X}$) and a bottle cap image that satisfies the constraints of the problem (represented by $\mathbf{Y}$). Value $\mathbf{x}_i$ (resp. $\mathbf{y}_i$) is the color of the i-th bottle cap region in $\mathbf{X}$ (resp. $\mathbf{Y}$). For simplicity, we suppose that $\mathbf{x}_i$ and $\mathbf{y}_i$ are real values that belong to $[0, 1]$ and the full-color version of the metric that we optimize for will be just the average of the metric values calculated over the R, G, and B channels separately.

The choice of using SSIM is based on the fact that the approximation obtained in the previous section preserves features from the input image and so does SSIM optimization. By comparing statistical measures such as color mean and deviation in sliding windows of the images, this metric will be able to capture the structure of the scene, which is highly desirable for our very limited problem. Its values range on the $[0, 1]$ interval and $\text{MSSIM}(\mathbf{X}, \mathbf{Y}) \approx 1$ means $\mathbf{X}$ and $\mathbf{Y}$ are very similar and $\text{MSSIM}(\mathbf{X}, \mathbf{Y}) \approx 0$ means they are very dissimilar, i.e., the higher this value, the better. More precisely, the Mean Structural Similarity (MSSIM, originally proposed in [4]) is given by

$$\text{MSSIM}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^{M} \text{SSIM}(\mathbf{X}_j, \mathbf{Y}_j), \tag{2}$$

where $\mathbf{X}_j$ is a neighborhood of the j-th cap in $\mathbf{X}$, $\mathbf{Y}_j$ is a neighborhood of the j-th cap in $\mathbf{Y}$, and M is the number of windows over which the (local) SSIM value is calculated. This local value is given by

$$\text{SSIM}(\mathbf{X}_j, \mathbf{Y}_j) = \frac{(2\mu_x\mu_y + \gamma_1)(2\sigma_{xy} + \gamma_2)}{(\mu_x^2 + \mu_y^2 + \gamma_1)(\sigma_x^2 + \sigma_y^2 + \gamma_2)}, \tag{3}$$

where

$$\mu_x = \sum_{i=1}^{N} w_i \mathbf{x}_i \tag{4}$$

is the weighted mean color value in the neighborhood $\mathbf{X}_j$ (μ_y is defined analogously),

$$\sigma_x = \left(\sum_{i=1}^{N} w_i (\mathbf{x}_i - \mu_x) \right)^{\frac{1}{2}}, \tag{5}$$

is the weighted color deviation within $\mathbf{X}_j$ (σ_y is defined analogously) and

$$\sigma_{xy} = \sum_{i=1}^{N} w_i (\mathbf{x}_i - \mu_x)(\mathbf{y}_i - \mu_y) \tag{6}$$

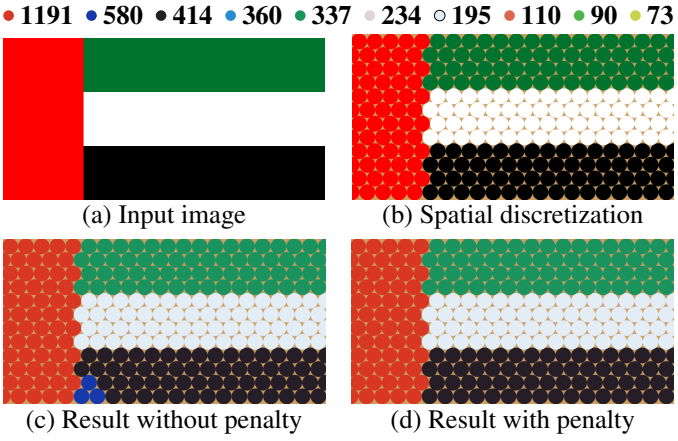


Fig. 7. Maximizing SSIM with respect to the discretized image (b) may produce wrong colors (c) due to the limited palette (top). Adding a penalty term to avoid variations in areas that are constant in (b) leads to a result (d) without this problem. Input image (a) obtained from wikipedia.org under Public Domain.

is the color correlation between the two neighborhoods. The constants $\gamma_1 = (K_1 L)^2$ and $\gamma_2 = (K_2 L)^2$ are set as in the original paper [4] for stability reasons: $K_1 = 0.01$, $K_2 = 0.03$ and $L = 1$.

All the ingredients to compute MSSIM (2) between two bottle cap images described so far are identical to the original MSSIM formulation [4]. The only change we make is in the weights that define the statistical measures in (4, 5, 6). The originally proposed 11×11 *pixel* window would be too small for our problem since its area would cover less than one bottle cap region and then would not capture variations in the bottle cap images. For example, the diameter of each cap in Figure 2 is approximately 18 pixels. On the other hand, 11×11 *cap* windows would cover almost half of the image and the metric would miss its locality.

We opt for windows centered at bottle caps and containing all caps in a 1-ring from the central one. Figure 6 illustrates a typical window centered at a cap that is not on the boundary of the bottle cap grid (boundary windows have fewer caps). These neighborhoods can capture statistical measures around a bottle cap since 7 caps cover a considerable portion of the bottle images. They are used for both the reference image $\mathbf{X}$, which is discretized over the bottle cap grid, and the bottle cap image $\mathbf{Y}$.

The weight of a cap C_i around a central cap C is given by

$$w_i = \frac{1}{S} e^{-\frac{d(C_i, C)}{2r^2}}, \tag{7}$$

where $d(C_i, C)$ is the distance in pixels from the center of cap C_i to the center of cap C, r is the radius of the caps in pixels and S is a normalization factor to have the sum of the weights equal to 1. For example, the application of this formula to the example in Figure 2 leads to $w_1 \approx 0.55$, $w_2 \approx 0.075$, $w_3 \approx 0.075$, $w_4 \approx 0.075$, $w_5 \approx 0.075$, $w_6 \approx 0.075$, $w_7 \approx 0.075$.

The optimization of MSSIM (2) subject to the limited palette we have at our disposal may lead to unexpected results such as the one in Figure 7 (c): an interface between two constant regions contains a third color that appears with the goal of approximating mean and standard deviation present in the full palette spatial discretization (b). This happens because the exact colors

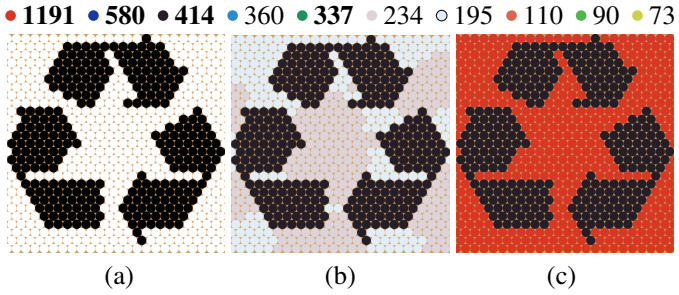


Fig. 8. Given the reference discretized image (a), the maximization of our energy using all available colors may lead to artificial contours in the result (b). Maximization using only an automatically selected subset of colors (the ones in bold at the top) leads to a result (c) without this problem. Input image obtained from publicdomainvectors.org under Public Domain.

from the constant regions in (b) are not available.

To avoid this problem, we augment the metric with a term that encourages constant (or nearly constant) regions in the spatially discretized image (Figure 7 (b)) to be constant in the final result. This term is calculated in the following manner:

1. Run K-means on the spatially discretized image (Figure 7 (b)) to identify the K colors that best represent all the colors in this image, where K is the number of different cap colors available in our problem.

2. For each of the K representative colors, select the caps in the spatially discretized image (Figure 7 (b)) with colors such that their distance to the representative color is smaller than 10% of the diameter of the $[0,1] \times [0,1] \times [0,1]$ RGB cube. These caps will be encouraged through a penalty term (8) to be constant in the output bottle cap image.

3. For each optimization intermediate state $\mathbf{Y}$, calculate the standard deviation σ of colors within each of the K groups and assign to every cap j in each group a penalty term given by

$$\text{Pen}(j) = (1 - \sigma)^2. \tag{8}$$

The final energy that we optimize for all results in this paper is a modification of (2) that combines SSIM with this penalty term:

$$E(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^{M} \text{SSIM}(\mathbf{X}_j, \mathbf{Y}_j) \cdot \text{Pen}(j). \tag{9}$$

This energy penalizes regions that are nearly constant in the reference full palette bottle cap image $\mathbf{X}$ but have different colors in the intermediate optimization state $\mathbf{Y}$, since in this case $\sigma > 0$ and $\text{Pen}(j) < 1$, making maximization of E to prefer states with $\sigma = 0$ and $\text{Pen}(j) = 1$ (within the nearly constant regions). Figure 7 (d) shows the result of maximizing this energy.

4.3. Constraints

Given the objective function to be maximized (9), we now turn our attention to the constraints of the problem, which are determined by the caps we have at our disposal. The optimization cannot demand additional colors or caps in a quantity that exceeds the available caps. In other words, the available caps must be treated as hard constraints in our problem.

For all results in this paper, we used the following set of caps, which consists of all caps that we were able to collect for this work:

1191 580 414 360 337 234 195 110 90 73

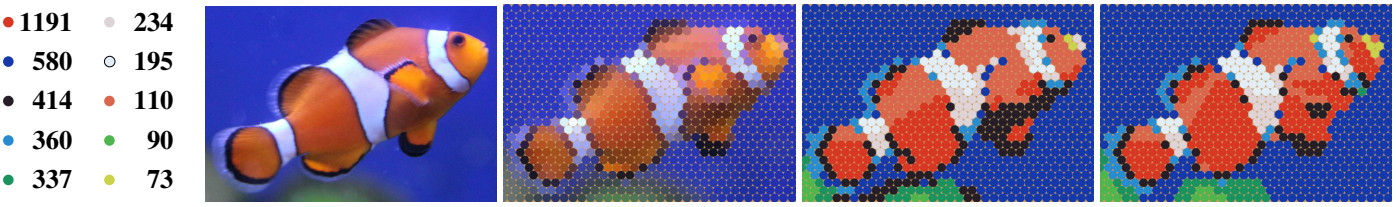
The color of each disk is an approximation for each cap color and the number close to it corresponds to the number of available caps of that color. This information is passed to the algorithm as a matrix where the three first columns are the R, G, and B coordinates of the colors and the fourth column describes the quantity of each cap. In our case, this matrix is given by:

$$\begin{bmatrix} 0.85 & 0.22 & 0.14 & 1191 \\ 0.08 & 0.22 & 0.67 & 580 \\ 0.15 & 0.12 & 0.15 & 414 \\ 0.15 & 0.55 & 0.82 & 360 \\ 0.10 & 0.58 & 0.37 & 337 \\ 0.87 & 0.83 & 0.84 & 234 \\ 0.89 & 0.93 & 0.96 & 195 \\ 0.85 & 0.40 & 0.30 & 110 \\ 0.31 & 0.71 & 0.30 & 90 \\ 0.79 & 0.82 & 0.29 & 73 \end{bmatrix} \tag{10}$$

The limitation on the number of caps of each color may be a problem in cases such as the one illustrated in Figure 8 (b). The white background of the image (a) occupies a region larger than any of the available bright colors can cover, leading to artificial contours when optimizing energy E (Eq. 9) using all available caps, even with the penalty term (Eq. 8) in the energy. This occurs because the penalty term eliminates spurious variations only when it has at its disposal the necessary caps in sufficient quantity, such as in Figure 7 (d). For cases such as Figure 8 (b), maximization of E (Eq. 9) tries to fill the white background with colors similar to the ones in the reference image (Figure 8 (a)) but fails to do so because none of these colors has caps in sufficient quantity. Removing or increasing the penalty term in the energy formulation leads to a result that is qualitatively similar to the one in Figure 8 (b) and does not solve the problem.

We then propose a methodology to discard cap colors that would likely be used in the result but are not available in sufficient quantity. The details of this process are described in Appendix A and the colors that survive are the ones that are actually used in our optimization, i.e., some rows in (10) are removed as a preprocess.

Figure 8 (c) shows the result of our optimization using only the colors whose numbers are highlighted in bold at the top of the figure. Notice that the artificial contours in (b) are no longer present. Among the caps that survived the discarding process (red, dark blue, black, and green), the red caps are the ones that more closely approximate the white background and reproduce color differences to the black arrows, and thus were chosen by the maximization of Eq. (9). We adopt the boldface convention for cap colors that are actually used in the optimization in all the results in this paper.



(a) Bottle caps (b) Input image (c) Spatial discretization (d) Initial solution (e) Optimization result

Fig. 9. Given a set of bottle caps (a) and an input image (b), our method first calculates a spatial discretization of the image over the bottle cap grid (c), which is taken as reference for our optimization. An initial solution (d) is quickly calculated and then improved by the optimization, leading to our result (e). Input is a cropped version of an image by Mike Sayre obtained from flickr.com under CC BY 2.0.

4.4. Initial solution

The method we are going to use in Section 4.5 to maximize the energy (9) demands an initial guess. The computation of this initial solution should be fast to let most of the computational time for the actual optimization and, more importantly, the initial solution must satisfy the constraints of the problem to start the optimization from a feasible state.

Using the example in Figure 9 as reference, we compute the initial solution applying the following heuristic: we loop over all caps in the spatially discretized image (c) in random order and determine the available cap that is the closest (in RGB coordinates) to the color of that cap in (c).

This simple strategy leads to the initial solution presented in (d), which misses some features of the fish such as the lower part of its body, but is much closer to the optimal solution than, for example, a random initial solution. Letting $\mathbf{X}$ be the reference discretized image (Figure 9, c), $\mathbf{Y}_0$ the initial solution (d), $\mathbf{Y}_{op}$ the solution (e) generated by the optimization to be described in Section 4.5 and $\mathbf{Y}_{ra}$ some random feasible solution, the energy values (9) are given by

$$E(\mathbf{X}, \mathbf{Y}_0) \approx 0.72,\ E(\mathbf{X}, \mathbf{Y}_{op}) \approx 0.75,\ E(\mathbf{X}, \mathbf{Y}_{ra}) \approx 0. \quad (11)$$

Recalling that $E(\mathbf{X}, \mathbf{Y}) \in [0, 1]$ for all possible $\mathbf{Y}$ and that the higher this value the more similar the images are, we conclude that this heuristic is successful at getting a good initial solution.

Notice that our initial solution strategy adopts randomness only to decide which cap index to assign the closest color first and not to decide the color itself. When the closest colors are available in sufficient quantity such as in Figure 9 (d), this is equivalent to assigning to all cap regions their closest colors. In cases such as the one in Figure 11 (c), our strategy does introduce some randomness to the initial solution, but our optimization successfully restores homogeneous regions (Figure 11, d).

4.5. Optimization

The maximization of energy E (9) is a difficult discrete optimization problem: even though we work at a low resolution, the number of possible cap arrangements is combinatorially high, making it impossible to try all of them using a brute force approach.

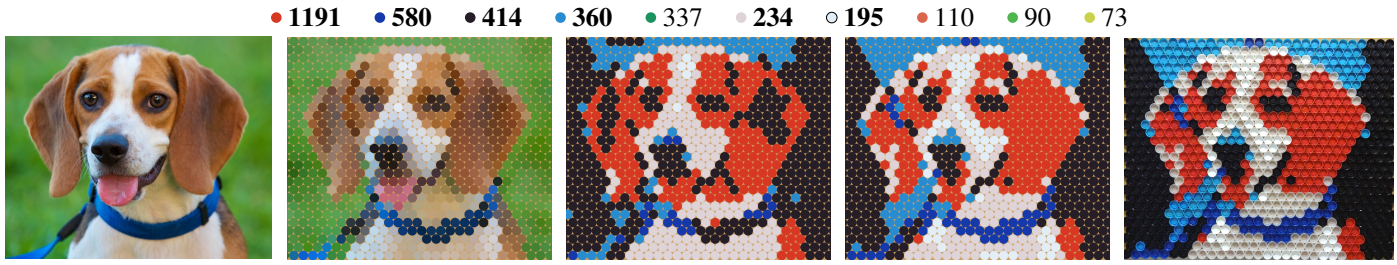
To be able to solve our problem in acceptable time, we have adopted the simulated annealing strategy [5]. To prevent this section from getting too long, we present here the general ideas of how we have applied this strategy to our problem and leave the details for Appendix B.

Starting from the initial solution (Section 4.4), we update some of the cap colors and accept the new state according to the following criteria: if it increases the energy value (i.e, if it is better), the new solution is accepted. If it decreases the energy value (i.e., it is worse), it may be accepted depending on a probability that decreases as the number of iterations increases. This acceptance of some lower value energy states is important to prevent the optimization of getting stuck during the initial states.

The cap color update is performed in the following manner: we randomly pick a cap and update this cap and some of its 1-neighbors (Figure 6, b) for all to have a single color among the available ones. In the first round of iterations, we randomly select 7 cap indices in each neighborhood and then exclude repetitions at every iteration. This process is repeated until the simulated annealing strategy converges and the resulting state corresponds to an energy maximization constrained to updating larger neighborhoods. From this state, we restart the simulated annealing procedure with the second round of iterations where 6 neighbors are randomly selected for every update. We repeat this process with 5 additional rounds where the last one consists of updating only 1 of the caps in each neighborhood at every iteration. This multi-stage strategy is important to improve the energy value faster at the initial stages and then reproduce the image (finer) details in the final stages. An example with some intermediate results of this multi-stage can be found in Figure B.19.

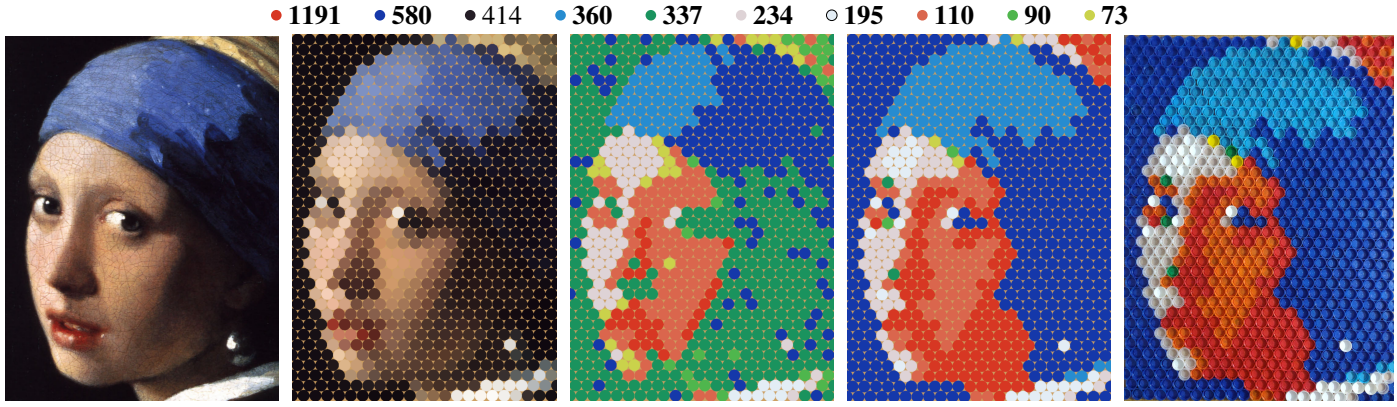
The color used as a candidate to update the color of the selected caps is randomly chosen according to a probability that is higher if more caps of that color are available. This probability is defined as the ratio of the number of available caps of that color by the total number of available caps. This approach guarantees that colors with no more caps available will not be used as update candidates and that the boundary of the feasible set will be avoided. We recall that the color update is accepted only if it satisfies a simulated annealing criterium, meaning that the most frequent colors are more often set as candidates but not necessarily accepted in the results.

A result of our optimization can be seen in Figure 9 (e), which is more similar to the reference image (c) than the initial solution (d). All the details of this process and pseudocode for it can be found in Appendix B. Timings for the optimization are provided in Table 1.



(a) Input image (b) Spatial discretization (c) Initial solution (d) Comupter-generated bottle cap art (e) Physically assembled bottle cap art

Fig. 10. Example of all steps of our method, from the input image (a) and caps (top) to the physically assembled result (e). Input is a cropped version of an image by Ted Murphy obtained from flickr.com under CC BY 2.0.



(a) Input image (b) Spatial discretization (c) Initial solution (d) Comupter-generated bottle cap art (e) Physically assembled bottle cap art

Fig. 11. Result for which the initial solution (c) is far from the optimal solution (d), illustrating that our optimization successfully recovers features. The input image is a crop from the painting *Girl with a Peral Earring* by Johannes Vermeer, available in the public domain.

Result	Secs. 4.1 to 4.4	Section 4.5	Assembling
Figure 1	0.33 secs	192.97 secs	1h 45 min
Figure 2	0.31 secs	133.21 secs	1h 43 min
Figure 10	0.72 secs	129.51 secs	1h 43 min
Figure 11	0.72 secs	222.87 secs	1h 27 min
Figure 12	0.77 secs	18.58 secs	35 min
Figure 13	0.67 secs	83.77 secs	37 min

Table 1. Timings of our method.

5. Implementation and performance

We implemented our method as a serial Matlab program. The inputs are an image in JPEG or PNG format, a list of RGB colors, and cap quantities in the same format as (10), and the number of caps on the horizontal extent of the bottle cap grid (Figure 2, (b)). The output is a computer-generated bottle cap art (Figure 2, (c)).

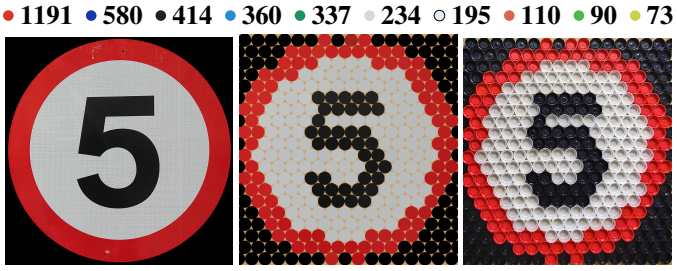
The spatial discretization (Section 4.1) is obtained running the `superpixels` Matlab built-in function, with a small change to make the coordinates of the centroids match the bottle cap grid centers. This function is written in C++ and run on Matlab as a MEX file. The metric calculation (Section 4.2) is coded as a Matlab function that re-uses the $\mathrm{SSIM}(\mathbf{X}_j, \mathbf{Y}_j)$ values between neighborhoods that were not affected by the state update to calculate (9) faster. Additionally, we use Matlab's `kmeans` function to compute the groups of caps that should be constant in the result to define the penalty term $\mathrm{Pen}(j)$ in (9). The input cap colors preprocessing (Section 4.3) and initial solution calculation (Section 4.4) are programmed as simple loops in Matlab and the simulated annealing process (Section 4.5 and Appendix B) is programmed as a translation of Algorithm 1 to Matlab.

We report in Table 1 the timings of our code for some examples recorded on a MacBook Pro Intel Core i5 2.3 Ghz computer with 8 GB memory. We also present in the last column the time taken to physically assemble the results. We can see that the most time-consuming stage is the physical assembling of the results. Our experience tells us that users get faster with practice, but we do not expect them to assemble final results at the scale of one meter in less than one hour.

6. Results

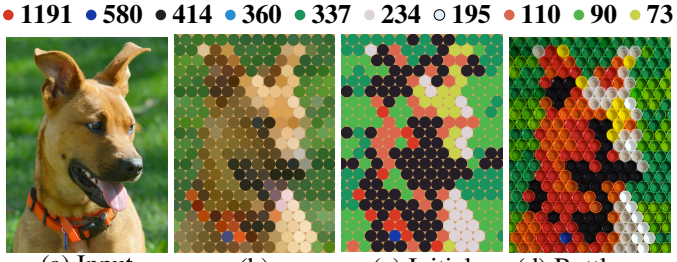
Throughout this paper, we have already presented many results with the goal of illustrating specific steps of our method. We now turn our attention to presenting results with all steps that generated them. Timings for these results are reported in Table 1.

Figure 10 presents an input image in (a) and the available caps at the top. The spatial discretization (b) reproduces well the features from (a) since it is adaptive. The initial solution (c) for the optimization fails to reproduce some details such as the



(a) Input image (b) Discretization (c) Bottle cap art

Fig. 12. An example for which the initial solution is identical to the result of the optimization (c). Input image by zaphad1 obtained from flickr.com under CC BY 2.0



(a) Input image (b) Discretization (c) Initial solution (d) Bottle cap art

Fig. 13. Result with a very low resolution. Input is a cropped version of an image by Eric Sonstroem obtained from flickr.com under CC BY 2.0.

ears of the dog and the leash. The result of our optimization (d) successfully recovers these features and is used to guide the physical assembling of the final bottle cap art (e), which is 92 cm wide by 73 cm high.

We present in Figure 11 a case with a bad initial solution (c). There were not enough black caps to reproduce the dark regions in (b) and these caps were eliminated by the process described in Section 4.3 and Appendix A. The method to determine the initial solution (Section 4.4) then got confused on how to fill these regions with green and blue caps. Nonetheless, the optimization (Section 4.5) successfully recovered the homogeneity of the dark regions and details such as the eye and the neck of the girl, as shown in (d). The physically assembled result (e) is 73 cm wide by 98 cm high.

On the other hand, Figure 12 presents a result (c) for which the initial solution is identical to the result of the optimization. The simple heuristic to determine the initial solution (Section 4.4) was capable of reproducing the features present in (b) for this case. The physically assembled result (c) is 54 cm wide by 55 cm high.

We also present in Figure 13 an example with a very low resolution of only 14 caps horizontally. Despite not being able to reproduce all details from the input image (a), our method produces a result (d) that clearly presents the silhouette of the dog with some of its details and again is a clear improvement over the initial solution (c). The result (d) is 45 cm wide by 56 cm high.

6.1. *Comparisons*

The novelty of the problem we are approaching makes it difficult to establish comparisons to previous methods. Naive approaches such as the one illustrated in Figure 3 and the initial solution for our optimization (Section 4.4) have clearly produced results worse than the ones produced by our method.

(a) Caps (b) Input image (c) Discretization (d) Bottle cap art

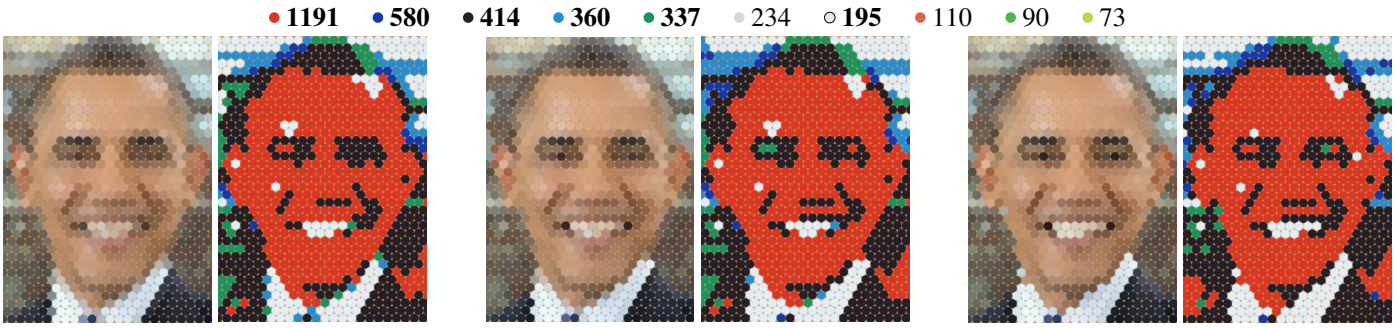
Fig. 14. Input image used by two pixel art papers [24, 25]. Despite the different settings we work with, our method successfully reproduces the features these methods do. Input image by William Warby obtained from flickr.com under CC BY 2.0.

To establish some comparison to previous methods, we present in Figure 14 an image (a) that was used as input in two pixel art papers: it appears in Figure 13 of [24] and in the supplementary materials of [25]. If we run our method with settings similar to theirs (number of bottle caps in the grid equal to the number of pixels used by them, a 16-color palette which, in our case, was determined running K-means over (b), and a very high number of caps of each color to ensure that this is not a constraint in practice), our result (c) reproduces well features such as the roofs and windows of the pagoda and has a quality similar to theirs, with the advantage of less staircased diagonal edges due to the different structure of the bottle cap grid in comparison to the pixel grid. Physically assembling the result in Figure 14 (c) would lead to a 1.54 m wide and 2.05 m high result, which is out of the scope of this paper. The point we make here is that if we were afforded to have settings similar to the ones used by these two methods, we would obtain similar quality results.

6.2. *Tests with other settings*

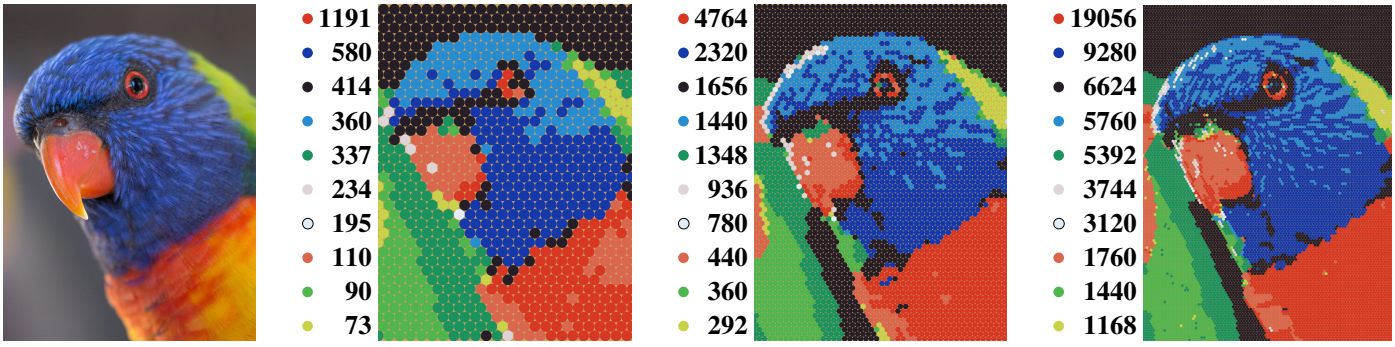
This section is devoted to studying how the variation of some settings in our method affects its results.

We present in Figure 15 the effect of changing the compactness parameter α (Section 4.1) on both the spatial discretization (left) and the final result (right). Recall that a higher value for this parameter leads to a smoother spatial discretization. For example, setting $\alpha = 500$ (a) leads to a spatial discretization (left) with smooth color variations between the face of the man and the collar of his shirt. The optimization tries to reproduce these variations with a very limited palette but obtains green and blue caps between these regions in the final result. A similar problem can be found in the boundary between the shirt and the suit and between the man's mouth and smile. Decreasing α to 100 (b) alleviates this problem but does not solve it. Our choice $\alpha = 50$ (left) leads to a spatial discretization with sharper transitions that can be reproduced in the final result with the available colors. Decreasing α further would lead to a noisy and distorted spatial discretization (Figure 5, c) that would be taken as reference for the optimization.



(a) $\alpha = 500$ (b) $\alpha = 100$ (c) $\alpha = 50$

Fig. 15. Different choices of compactness parameter α affect the spatial discretization (left) and the final result (right). Higher values lead to a discretization with smoother boundaries that cannot be reproduced with the available colors (for example, the boundary between the man's face and the collar of the shirt). Our choice $\alpha = 50$ (c) leads to sharper edges that are better reproduced in the final result.



(a) Input image (b) 1 meter by 0.75 meters (c) 2 meters by 1.5 meters (d) 4 meters by 3 meters

Fig. 16. Results with different resolutions and the number of available caps varying proportionally. Input is a cropped version of an image by Flickr user troposa1 available under CC BY 2.0.

The low resolution we have adopted for most of the results can be a problem when the input image has high frequency details. This can be seen in Figure 16 (b), where the result has 24 caps on the horizontal extent and, if physically assembled, would be approximately 1 m high by 0.75 m wide. Doubling the dimensions and multiplying by 4 the number of available caps leads to a result (c) with some features such as the eye and the beak of the bird better reproduced. If we double again the dimensions to obtain a bottle cap art with approximately 4 by 3 meters, higher frequencies such as feathers around the eyes are present in the final result. Figure 16 shows that our method is able to reproduce more details from the input image if more resources are available.

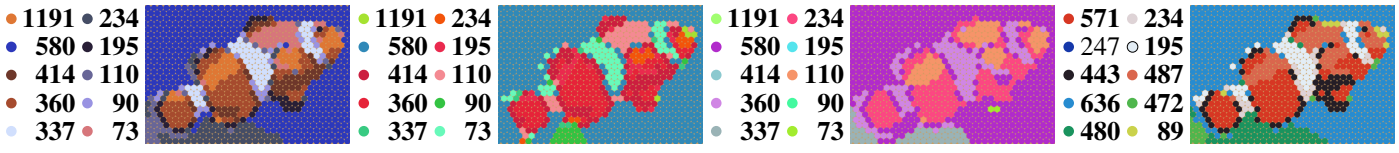
We also present in Figure 17 an experiment where use different cap colors, instead of our standard set of caps (Figure 9, a). A set of colors that are representative of the input image leads to a result (Figure 17, a) that is very similar to the reference image (Figure 9, c). Setting two sets of random colors (Figure 17, b, c) leads to results with less quality but still revealing the content of the image and preserving most of its features. Denoting by $\mathbf{X}$ the reference image (Figure 9, c), $\mathbf{Y}_{st}$ the result with our standard set of caps (Figure 9, e), $\mathbf{Y}_a$ the result in Figure 17 (a), $\mathbf{Y}_b$ the result in Figure 17 (b) and $\mathbf{Y}_c$ the result in Figure 17 (c) the energy values (Eq. 9) are given by $E(\mathbf{X}, \mathbf{Y}_a) \approx 0.86$, $E(\mathbf{X}, \mathbf{Y}_{st}) \approx 0.75$, $E(\mathbf{X}, \mathbf{Y}_b) \approx 0.65$ and $E(\mathbf{X}, \mathbf{Y}_c) \approx 0.62$, which shows that the metric E measures well the similarity between the images.

We observe that the background in Figure 17 (c) is filled by purple caps and the middle part of the body of the fish by caps of a similar color, while the background color in the reference image (Figure 9, c) is dark blue and the middle part of the fish is white. As a consequence, the result in Figure 17 (c) has less contrast than the reference image. This is explained by the terms $\frac{(2\mu_x\mu_y+\gamma_1)}{(\mu_x^2+\mu_y^2+\gamma_1)}$ and $\frac{(2\sigma_{xy}+\gamma_2)}{(\sigma_x^2+\sigma_y^2+\gamma_2)}$ in the SSIM mertic (3): while the former prioritizes color fidelity, the latter focuses on contrast and structures. Adjusting the constants γ_1 and γ_2 could change the balance between these terms but prioritizing contrast too much could lead to results with colors that are too different from the reference image, even when some of these colors are available in sufficient quantity.

Figure 17 (d) shows an additional result generated using our standard set of cap colors but with different quantities. The new quantities are random numbers between 1 and 700 such that their sum is equal to our standard total number of caps (3854). The main difference to the result using our standard setup (Figure 9, e) is the background color since there were not sufficient dark blue caps to fill the background area. Nonetheless, the new result $\mathbf{Y}_d$ in Figure 17 (d) presents features similar to the ones in Figure 9 (e) and has energy value $E(\mathbf{X}, \mathbf{Y}_d) \approx 0.71$, which shows that its quality is superior to the results in Figure 17 (b,c).

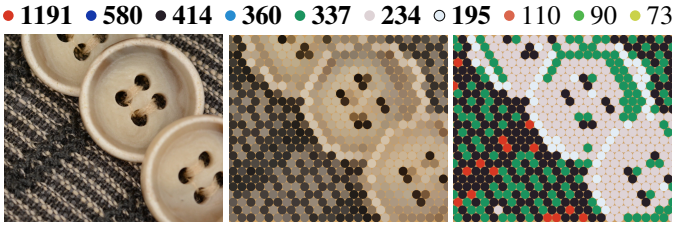
6.3. Limitations

The main limitations of our method are related to the very low color and spatial resolution we work with. Figure 18 presents the result of applying our method to an input with



(a) (b) (c) (d)

Fig. 17. Result of applying our method to the input image in Figure 9 using different sets of available bottle caps.



(a) Input image (b) Discretization (c) Bottle cap art

Fig. 18. Applying our method to an input image with high frequencies (a) leads to aliasing in both the discretization over the bottle cap grid (b) and our result (c), which also has spurious color variations caused by the very limited color resolution. Input is a cropped version of an image by Clint Budd obtained from flickr.com under CC BY 2.0.

high spatial frequencies (a). Aliasing produced at the first step of our method (b) is carried over all steps and leads to a result (c) where the orientation of the lines in high frequency areas is wrong. Preprocessing the input image with antialiasing techniques [42, 43, 44] could alleviate this problem, but would considerably modify the input to deal with the very low spatial resolution.

Figure 18 also illustrates a problem related to the very low color resolution imposed by the caps we have at our disposal: smooth color transitions in the input may lead to spurious colors in the results, such as the green caps in the buttons and the red caps between green and black caps in the high frequency areas. This may also lead to isolated outliers such as the white caps in the face of the man in Figure 2 (c,d), caused by highlights in the input image that were not properly reproduced in the final result. Notice that postprocessing the result to remove outliers would not be a proper solution, since this would remove important features of the size of one cap, for example, the eye of the fish in Figure 9 (e) and the pearl earring in Figure 11 (d,e)).

7. Conclusion

In this paper, the problem of approximating an input image with given plastic bottle caps has been proposed and a solution involving spatial discretization followed by a structure-aware optimization has been presented. Despite the very limited spatial and color resolution of the problem, our method is capable of producing high quality results for a variety of input images.

As a possible future work, we consider generating bottle cap arts for collections of images, instead of single images as in this paper. Energy (9) would allow ranking which image in the collection is best approximated by the given set of caps. Looping over all input images and applying our method would be prohibitive due to the time taken by our optimization and we are considering ways such as a multigrid strategy to speed it up.

Other interesting directions for future work include considering assembling caps over irregular grids (including grids with multiple layers of caps) and allowing caps of different sizes. This would make it possible to obtain results with sharper edges, more varied artistic effects, and occluded canvas. A possible solution in this context would be to break the problem into two parts: the first would calculate an optimal grid for a given input image and the second would determine the colors of the caps to cover this optimal grid. The first part would have to consider the practical feasibility and stability of the physically assembled bottle cap art and the second could use an optimization framework similar to the one we presented in this work.

This paper has also shown that it is possible to propose and approach valuable computer graphics problems using material that would be discarded and potentially damage the environment. We hope that it serves as an encouragement for the community to consider more of these problems in the close future.

Acknowledgments

The author thanks Joao Paulo N. Barbosa, Marianna R. Vago, and the UFSC community for gathering caps for this project, Joao Paulo N. Barbosa for the initial discussions, Bryce van Ross for proofreading part of the paper, the anonymous reviewers for their suggestions, and Flickr users for making available their images.

References

[1] Boonstra, M, van Hest, F. The findings of the first survey into plastic bottle cap pollution on beaches in the netherlands. Tech. Rep.; The North Sea Foundation; 2017.

[2] noordzee TV, . The bottle cap report - north sea foundation. 2017. URL: https://youtu.be/JmeA7UGNNEU.

[3] Independent, TS. 'beauty through toxicity' — sampson independent. 2017. URL: https://www.clintonnc.com/news/23475/beauty-through-toxicity.

[4] Wang, Z, Bovik, AC, Sheikh, HR, Simoncelli, EP. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 2004;13(4):600–612. doi:10.1109/TIP.2003.819861.

[5] Press, WH, Teukolsky, SA, Vetterling, WT, Flannery, BP. Numerical Recipes in C. Second ed.; Cambridge, USA: Cambridge University Press; 1992.

[6] Haeberli, P. Paint by numbers: Abstract image representations. SIGGRAPH Comput Graph 1990;24(4):207–214. URL: https://doi.org/10.1145/97880.97902.

[7] DeCarlo, D, Santella, A. Stylization and abstraction of photographs. ACM Trans Graph 2002;21(3):769–776. URL: https://doi.org/10.1145/566654.566650.

[8] Collomosse, J, Kyprianidis, JE. Artistic Stylisation of Images and Video. In: Martin, R, Torres, JC, editors. Eurographics 2011 - Tutorials. The Eurographics Association; 2011,doi:10.2312/EG2011/tutorials/t3.

[9] Heckbert, P. Color image quantization for frame buffer display. SIGGRAPH Comput Graph 1982;16(3):297–307. URL: https://doi.org/10.1145/965145.801294.

[10] Gervautz, M, Purgathofer, W. A simple method for color quantization: Octree quantization. In: Magnenat-Thalmann, N, Thalmann, D, editors. New Trends in Computer Graphics. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-83492-9; 1988, p. 219–231.

[11] Xiang, Z. Color image quantization by minimizing the maximum intercluster distance. ACM Trans Graph 1997;16(3):260–276. URL: https://doi.org/10.1145/256157.256159.

[12] Floyd, RW, Steinberg, L. An Adaptive Algorithm for Spatial Greyscale. Proceedings of the Society for Information Display 1976;17(2):75–77.

[13] Knuth, DE. Digital halftones by dot diffusion. ACM Trans Graph 1987;6(4):245–273. URL: https://doi.org/10.1145/35039.35040.

[14] Velho, L, Gomes, JdM. Digital halftoning with space filling curves. In: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '91; New York, NY, USA: Association for Computing Machinery. ISBN 0897914368; 1991, p. 81–90. URL: https://doi.org/10.1145/122718.122727.

[15] Ostromoukhov, V, Hersch, RD. Multi-color and artistic dithering. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '99; USA: ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605; 1999, p. 425–432. URL: https://doi.org/10.1145/311535.311605.

[16] Ostromoukhov, V. A simple and efficient error-diffusion algorithm. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '01; New York, NY, USA: Association for Computing Machinery. ISBN 158113374X; 2001, p. 567–572. URL: https://doi.org/10.1145/383259.383326.

[17] Jodoin, PM, Ostromoukhov, V. Halftoning over a hexagonal grid. In: Eschbach, R, Marcu, GG, editors. Color Imaging VIII: Processing, Hardcopy, and Applications; vol. 5008. International Society for Optics and Photonics; SPIE; 2003, p. 443 – 454. URL: https://doi.org/10.1117/12.473230.

[18] Pang, WM, Qu, Y, Wong, TT, Cohen-Or, D, Heng, PA. Structure-aware halftoning. In: ACM SIGGRAPH 2008 Papers. SIGGRAPH '08; New York, NY, USA: Association for Computing Machinery. ISBN 9781450301121; 2008,URL: https://doi.org/10.1145/1399504.1360688.

[19] Li, H, Mould, D. Contrast-aware halftoning. Computer Graphics Forum 2010;29(2):273–280. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01596.x.

[20] Xu, X, Zhang, L, Wong, TT. Structure-based ascii art. ACM Transactions on Graphics (SIGGRAPH 2010 issue) 2010;29(4):52:1–52:9.

[21] O'Grady, PD, Rickard, ST. Automatic ascii art conversion of binary images using non-negative constraints. In: IET Irish Signals and Systems Conference (ISSC 2008). 2008, p. 186–191. doi:10.1049/cp:20080660.

[22] Xu, J, Kaplan, CS. Artistic thresholding. In: Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering. NPAR '08; New York, NY, USA: Association for Computing Machinery. ISBN 9781605581507; 2008, p. 39–47. URL: https://doi.org/10.1145/1377980.1377990.

[23] Lai, YK, Rosin, PL. Non-photorealistic Rendering with Reduced Colour Palettes. London: Springer London; 2013, p. 211–236.

[24] Gerstner, T, DeCarlo, D, Alexa, M, Finkelstein, A, Gingold, Y, Nealen, A. Pixelated image abstraction with integrated user constraints. Computers & Graphics 2013;37(5):333–347. URL: http://www.sciencedirect.com/science/article/pii/S0097849313000046. doi:http://dx.doi.org/10.1016/j.cag.2012.12.007.

[25] Kopf, J, Shamir, A, Peers, P. Content-adaptive image downscaling. ACM Trans Graph 2013;32(6). URL: https://doi.org/10.1145/2508363.2508370.

[26] Han, C, Wen, Q, He, S, Zhu, Q, Tan, Y, Han, G, et al. Deep unsupervised pixelization. ACM Transactions on Graphics (TOG) 2018;37(6):1–11.

[27] Kuo, MH, Lin, YE, Chu, HK, Lee, RR, Yang, YL. Pixel2brick: Constructing brick sculptures from pixel art. Comput Graph Forum 2015;34(7):339–348. URL: https://doi.org/10.1111/cgf.12772.

[28] Öztireli, AC, Gross, M. Perceptually based downscaling of images. ACM Trans Graph 2015;34(4). doi:10.1145/2766891.

[29] Weber, N, Waechter, M, Amend, SC, Guthe, S, Goesele, M. Rapid, detail-preserving image downscaling. ACM Trans Graph 2016;35(6). URL: https://doi.org/10.1145/2980179.2980239.

[30] Wu, Z, Leahy, R. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. IEEE Trans Pattern Anal Mach Intell 1993;15(11):1101–1113. URL: https://doi.org/10.1109/34.244673.

[31] Shi, J, Malik, J. Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 2000;22(8):888–905. URL: https://doi.org/10.1109/34.868688.

[32] Rother, C, Kolmogorov, V, Blake, A. "grabcut": Interactive foreground extraction using iterated graph cuts. ACM Trans Graph 2004;23(3):309–314. URL: https://doi.org/10.1145/1015706.1015720.

[33] Achanta, R, Shaji, A, Smith, K, Lucchi, A, Fua, P, Süsstrunk, S. Slic superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence 2012;34(11):2274–2282. doi:10.1109/TPAMI.2012.120.

[34] Vanek, J, Garcia Galicia, J, Benes, B. Clever support: Efficient support structure generation for digital fabrication. Computer Graphics Forum 2014;33. doi:10.1111/cgf.12437.

[35] Karasik, E, Fattal, R, Werman, M. Object partitioning for support-free 3d-printing. Computer Graphics Forum 2019;38:305–316. doi:10.1111/cgf.13639.

[36] Wall, LW, Jacobson, A, Vogel, D, Schneider, O. Scrappy: Using scrap material as infill to make fabrication more sustainable. In: CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. 2021,.

[37] Wang, R, Yu, B, Marco, J, Hu, T, Gutierrez, D, Bao, H. Real-time rendering on a power budget. ACM Trans Graph 2016;35(4). URL: https://doi.org/10.1145/2897824.2925889.

[38] Chuang, J, Weiskopf, D, Möller, T. Energy aware color sets. Computer Graphics Forum 2009;28(2):203–211. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01359.x.

[39] Chen, H, Wang, J, Chen, W, Qu, H, Chen, W. An image-space energy-saving visualization scheme for oled displays. Computers & Graphics 2014;38:61 – 68. URL: http://www.sciencedirect.com/science/article/pii/S0097849313001611. doi:https://doi.org/10.1016/j.cag.2013.10.020.

[40] Thue, A. Über die dichteste Zusammenstellung von kongruenten Kreisen in einer Ebene. J. Dybwad; 1910.

[41] Chang, HC, Wang, LC. A simple proof of thue's theorem on circle packing. 2010. arXiv:1009.4322.

[42] Shannon, CE. Communication in the presence of noise. Proc Institute of Radio Engineers 1949;37(1):10–21.

[43] Gastal, ESL, Oliveira, MM. Spectral remapping for image downscaling. ACM Trans Graph 2017;36(4). URL: https://doi.org/10.1145/3072959.3073670.

[44] Germano, RL, Oliveira, MM, Gastal, ESL. Real-time frequency adjustment of images and videos. Computer Graphics Forum 2021;40(2):23–37. URL: https://www.inf.ufrgs.br/~eslgastal/RealTimeFrequencyAdjustment/. doi:https://doi.org/10.1111/cgf.142612.

Appendix A. Caps preprocessing

The maximization of MSSIM (2) subject to the limited number of caps can lead to results such as in Figure 8 (b) since the term $\frac{(2\mu_x\mu_y+\gamma_1)}{(\mu_x^2+\mu_y^2+\gamma_1)}$ in (3) prioritizes color fidelity over contrast and structures, which are present in the second term $\frac{(2\sigma_{xy}+\gamma_2)}{(\sigma_x^2+\sigma_y^2+\gamma_2)}$. In this very constrained setting, the optimization may opt for color fidelity even if this introduces variations that are inexistent in the reference image (Figure 8 (a)). Adjusting the constants γ_1 and γ_2 to prioritize variances instead of mean colors is not a solution since it may lead to the use of wrong colors when the right ones are available. An alternative would be to increase the

penalty term in (9), but this would make the method rely too much on segmentation, which demands user interactions to be done precisely and is out of the scope of this paper since our goal is to propose a fully automatic method to compute bottle cap arts.

We propose a preprocessing to discard caps that would likely be used to increase color fidelity but are not available in sufficient quantity to fill large regions. The maximization of (9) is then performed using only cap colors that are available in sufficient quantity. This may lead to results with different colors for large areas (Figure 8 (c)) but with features preserved due to the maximization of SSIM.

Let C_j be the RGB coordinates of the available cap colors and n_j their respective quantities. For all results in this paper (in particular the one in Figure 8) we have $j = 1, \ldots, 10$ and, for example, the white cap color is given by $C_7 = (0.89, 0.93, 0.96)$ and $n_7 = 195$.

We define N_j, the number of caps in the spatially discretized image $\mathbf{X}$ (Figure 8 (a)) that have colors closer to C_j than to any other available cap color C_i, as

$$N_j = \#\,\{\text{ cap colors } C \text{ in } \mathbf{X} \mid d(C, C_j) < 0.96 \cdot d(C, C_i), \text{ for all } i \neq j\,\}. \tag{A.1}$$

The factor 0.96 serves to determine colors that have distances similar to more than one color in the available palette and do not contribute to any specific N_j.

All available caps of color C_j are discarded if $n_j < N_j$, i.e., the number of caps of this color is not sufficient to fill areas that would likely be filled by this color. For instance, in Figure 8, we have $N_7 = 376$ and $n_7 = 195$ and thus the white caps are discarded. After all necessary caps are discarded by this process, we re-apply it to the remaining caps since the same problem can happen to them. For the example we are considering (Figure 8) a second run leads to discarding the caps of beige color ($n_6 = 234$), since $N_6 = 376$ for the second run. The process is re-applied until no more caps are discarded by it.

In the unlikely cases when all caps are discarded, we decrease the factor 0.96 in (A.1) and run this process from the very beginning with all colors. If the problem persists, we re-apply this strategy as many times as needed, printing warnings and suggesting the user to decrease the final bottle cap art resolution since these cases happen when there are much fewer caps than necessary to cover large areas.

Appendix B. Optimization details

The intent of this appendix is to present additional details of the optimization discussed in Section 4.5. We are going to base our presentation on the pseudocode in Algorithm 1. The inputs for the optimization are a spatially discretized (full palette) image $\mathbf{X}$ generated as described in Section 4.1, an initial solution $\mathbf{Y}_0$ (Section 4.4) and a matrix A_0 that results from applying the preprocessing in Appendix A to a matrix in the format described in Section 4.3 (e.g., the one in Eq. (10)). The output is a bottle cap art $\mathbf{Y}$ that satisfies the constraints of the problem. Despite we cannot ensure its optimality due to the lack of guarantees of the simulated annealing strategy and local minima of the energy (9), $\mathbf{Y}$ is most of the times a qualitative and quantitative improvement over the initial solution $\mathbf{Y}_0$ (it is not for cases such as the one presented in Figure 12).

The simulated annealing test to accept a new bottle cap art is given in line 18 of Algorithm 1. If the energy value E_{new} (line 15) for the state $\mathbf{Y}_{new}$ (line 14) is greater than the current energy value E, then $\Delta E > 0$ (line 16), $\frac{\Delta E}{t} > 0$, $exp\left(min\left(0, \frac{\Delta E}{t}\right)\right) = 1$ and the state is accepted, since $r < 1$ (line 17). Recall that we are *maximizing* the energy and thus increasing the energy value is the main goal. Otherwise ($\Delta E < 0$), the condition in line 18 will have a higher probability to be satisfied the closer ΔE is to 0 and the higher the temperature t is. The temperature update in line 33 makes it more difficult for states with $\Delta E < 0$ to be accepted as the number of iterations increases.

To make the optimization faster, we start using the parameter $N = 7$ (line 10) to update a group of caps in a neighborhood, instead of single cap updates. When the number of unsuccessful attempts using this value of N exceeds the maximum M (line 26) we decrease it by 1 (line 27), reset the temperature for simulated annealing (line 29) and triple the maximum number of attempts (line 30). This last update serves to give the algorithm more chances to reproduce features at finer scales. Figure B.19 presents intermediate states obtained after reaching the maximum number of attempts for $N = 5, 4, 3, 2$ and 1 ($N = 7$ and 6 are omitted due to their similarity to the initial solution). Notice that the result ($N = 1$) in Figure B.19 is not identical to the result presented in Figure 2 (c), which was obtained with the same settings. This illustrates the fact that our method does not always return the same results, due to the random updates in Algorithm 1 (e.g., lines 12, 13 and 17) and the local minima of the energy. Nonetheless, these two results have only a few different caps, and both reproduce the main features from the reference image.

To conclude this appendix, we explain below all the functions that are called during the execution of Algorithm 1:

- AvailableCaps (lines 2 and 20) returns a matrix in the same format as A_0, but with cap quantities that correspond to the ones in A_0 minus the ones used in the state $\mathbf{Y}$.

- Energy (lines 3 and 15) evaluates Eq. (9). It takes as inputs the reference image $\mathbf{X}$, a state $\mathbf{Y}_{new}$, precomputed local SSIM values S (Eq. (3)), and the indices U of the caps that changed with respect to the previous (accepted) state $\mathbf{Y}$. The two last arguments of this function have the purpose of speeding it up since only caps around the caps in U have their local SSIM values affected by the update. If $\mathbf{Y}_{new}$ is accepted, then the additional output S_{new} is used in the next energy evaluations.

- CapsToUpdate (line 12) randomly picks a cap index to be the center of a neighborhood (Figure 6). It then runs N draws to obtain indices in the neighborhood and excludes repeitions, i.e., the set of output indices U has at most N elements.

- RandomColor (line 13) determines the cap color to be

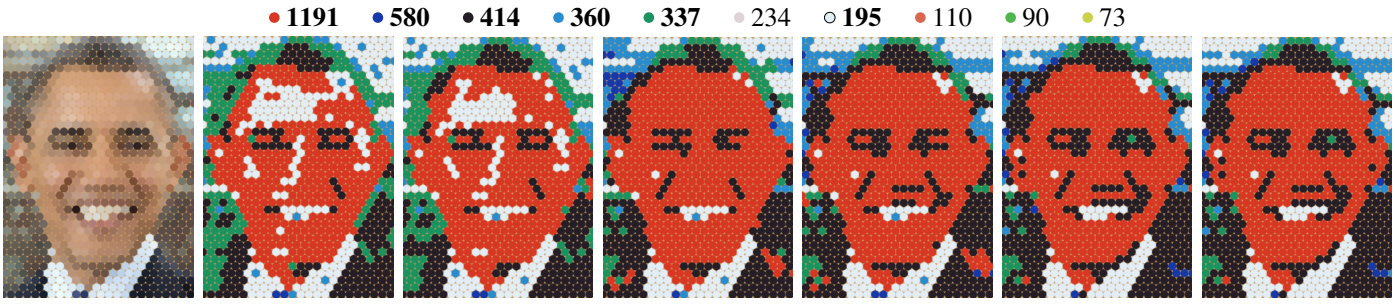


Fig. B.19. Reference image, initial solution, and intermediate stages of the optimization. The smaller the number of caps N that are updated at every iteration, the more details from the reference image are reproduced. The state with $N = 1$ is the final result of the optimization.

used for all caps in U in the following manner: let N_k be the number of available caps of the k-th color in A, minus the number of elements in U (if this calculation leads to a negative number, we impose $N_k = 0$). The function then returns the k-th color according to the probability $P_k = \frac{N_k}{\sum N_k}$.

• UpdateBottleCapArt (line 14) updates the colors in state $\mathbf{Y}$ to be C for all cap indices in U. In the unlikely cases when the output $\mathbf{Y}_{new}$ is equal to the input $\mathbf{Y}$, functions CapsToUpdate and RandomColor are re-run until an actual new state is obtained. This verification is important to base the choice of the maximum number of unsuccessful attempts on actual changes in the running state.

Algorithm 1: Bottle cap art simulated annealing

Inputs :
- $\mathbf{X}$ Spatially discretized image
- $\mathbf{Y}_0$ Initial solution
- A_0 Matrix with available cap colors and quantities

Outputs:
- $\mathbf{Y}$ Bottle cap art

1 $\mathbf{Y} \leftarrow \mathbf{Y}_0$ // *Bottle cap art initialization*
2 $A \leftarrow \text{AvailableCaps}(A_0, \mathbf{Y})$ // *Update available caps*
3 $\{E, S\} \leftarrow \text{Energy}(\mathbf{X}, \mathbf{Y}, [], \{1, \ldots, \#\mathbf{Y}\})$ // *Initial energy value*
4 $t_0 \leftarrow 2\text{e}{-4}$ // *Initial temperature (simulated annealing parameter)*
5 $t \leftarrow t_0$ // *Temperature initialization*
6 $\rho \leftarrow 0.8$ // *Factor to decrease temperature during annealing*
7 $iter_u \leftarrow 0$ // *Initialization of number of unsuccessful attempts*
8 $M \leftarrow 20$ // *Maximum number of unsuccessful attempts*
9 $iter \leftarrow 1$ // *Initialization of total number of iterations*
10 $N \leftarrow 7$ // *Initial cap neighborhood size*
11 **while** $N > 0$ **do**
12 $U \leftarrow \text{CapsToUpdate}(N, \#\mathbf{Y})$ // *Caps to have a new color*
13 $C \leftarrow \text{RandomColor}(A, U)$ // *Color to use for all caps in U*
14 $\mathbf{Y}_{new} \leftarrow \text{UpdateBottleCapArt}(\mathbf{Y}, U, C)$ // *New state*
15 $\{E_{new}, S_{new}\} \leftarrow \text{Energy}(\mathbf{X}, \mathbf{Y}_{new}, S, U)$ // *New energy*
16 $\Delta E = E_{new} - E$ // *Energy difference*
17 $r \leftarrow \text{Rand}(0, 1)$ // *Random number uniform in* $(0, 1)$
18 **if** $r < exp\left(min\left(0, \frac{\Delta E}{t}\right)\right)$ **then** // *Accept update*
19 $\mathbf{Y} \leftarrow \mathbf{Y}_{new}$ // *Update bottle cap art*
20 $A \leftarrow \text{AvailableCaps}(A_0, \mathbf{Y})$ // *Update available caps*
21 $S \leftarrow S_{new}$ // *Update local SSIM values*
22 $E \leftarrow E_{new}$ // *Update energy value*
23 $iter_u \leftarrow 0$ // *Reset number of unsuccessful attempts*
24 **else** // *Reject update*
25 $iter_u \leftarrow iter_u + 1$ // *Update unsuccessful attempts*
26 **if** $iter_u > M$ **then** // *Exceeded max number of attempts*
27 $N \leftarrow N - 1$ // *Decrease neighborhood size*
28 $iter_u \leftarrow 0$ // *Reset number of unsuccessful attempts*
29 $t \leftarrow t_0$ // *Reset temperature*
30 $M \leftarrow 3 * M$ // *Triple maximum number of attempts*
31 $iter \leftarrow iter + 1$ // *Update total number of iterations*
32 **if** $mod(iter, 500) = 0$ **then** // *At every 500 itearions*
33 $t \leftarrow \rho * t$ // *Cool temperature down*