Pixelated Image Abstraction

Junru Wu

Texas A&M University
{sandboxmaster@tamu.edu}

Abstract. Pixel art is a form of digital art where images are edited on the pixel level and usually containing only limited color palette. In this paper, we set out to investigated an automatic method to abstract high resolution images into very low resolution outputs with reduced color palettes in the style of pixel art. A simple k-means based methods was introduced where we regard the color quantization of each pixel as a unsupervised clustering task. Next, nearest-neighbour interpolation was used to downsample the image into very low resolution. We demonstrate the effectiveness our methods both qualitatively and quantitatively. We show it produce comparable result compare with state-of-the-art methods while enjoying some speed up benefit brought by our simple framework.

1 Introduction

Pixel art has been a long-standing form of contemporary art and a common rendering technique in digital games and media. In recent years, the significance of pixel art has mostly been recognized as art communities as a rendering style in games. Classic pixel art games includes The Legend of Zelda, Pacman, and Space Invaders, also worth mentioning is a recently popluar pixel art game name Minecraft published in 2011, which has sold over 20 million copies worldwide.

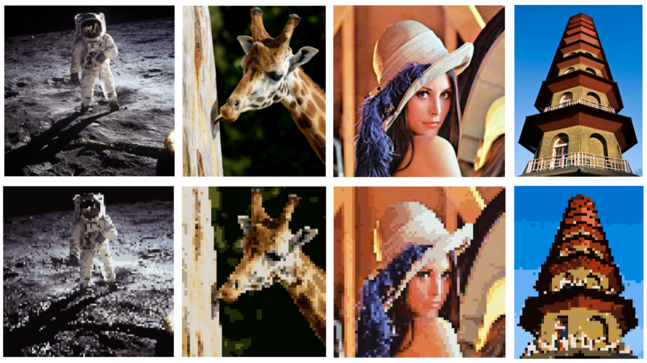

Fig. 1. Examples of generated Pixel Arts: The upper row show the original images, the lower row shows its generated pixel art

What makes pixel art both compelling and difficult are the limitations imposed on the medium, the final art form has to be composed with as few pixels and colors as possible. During creation, the pixel artist needs to carefully choose the set of colors and their placement in the image so that it best depicts the subject. It is generally completed by artists pixel-by-pixel, which is extremely time consuming and labor-intensive. However, there is few methods exist to automatically or semi-automatically create effective pixel art, which limits the amount of art style that people have accessible to.

2 Related Work

There has been automated and semi-automated methods proposed for some other art forms such as line drawing [1] and painting [2].

Methods of image abstraction such as those proposed by DeCarlo and Santella *et al.* [3] and Winnemoller *et al.* [4] not only abstract images, but do so while retaining the most salient features in a image. A similar method for pixel art creation would benefit the work process of existing artists and open the art style to a larger audience.

Timothy Gerstner *et al.* [5] introduce an iterative algorithm, where each iteration is a multi-step process that simultaneously segments the original image and solves for a limited sized palette. It then utilize a modified version of a segmentation algorithm proposed by Achanta *et al.* [6] and map each pixel in the output to a segment of the input image. Additonally, it use an adaptation of deterministic annealing by Kenneth Rose [7] to find the palette and its mapping to pixels in the output. The methods in [5] will be the main methods that to compare against in the paper.

3 Methodology

Our methods of pixelated image abstraction consist of two stage: Color Quantization and Image Downsampling.

3.1 Color Quantization

Generally a 32-bit color image would contains 20W-100W color palettes depending on its size and content. In the color quantization stage, we proposed to quantify the color palette from millions into dozens. We proposed to solve color quantization problem via feature extraction and unsupervised clustering. Specifically, we consider each pixel of the image as a sample and our goal to cluster millions of pixels into dozens of groups.

Feature Extraction In the feature extraction stage, for each pixel, we only consider three type of feature, the L, A, B Channel of each pixel, the 7x7 neighborhood pixels and Histogram of oriented gradients (HOG) descriptors of 7x7 neighborhood pixels.

HOG descriptors is well-known for its invariance to geometric and photometric transformations and is widely used in object detection task. In the HOG descriptors, we choose the number of cells per block to be 1, the number of pixels per cell to be 7x7, and the number of bins per cell histogram to be 9, which result in the final our HOG descriptors to be 9 dimension.

Those three type of feature is then concatenated into a vector of 61 dimension. The feature extraction pipline is shown in Figure 2. We futher experiment the effect using different feature set as shown in Fig 3 and we found that a combination of those three type of features performs the best.

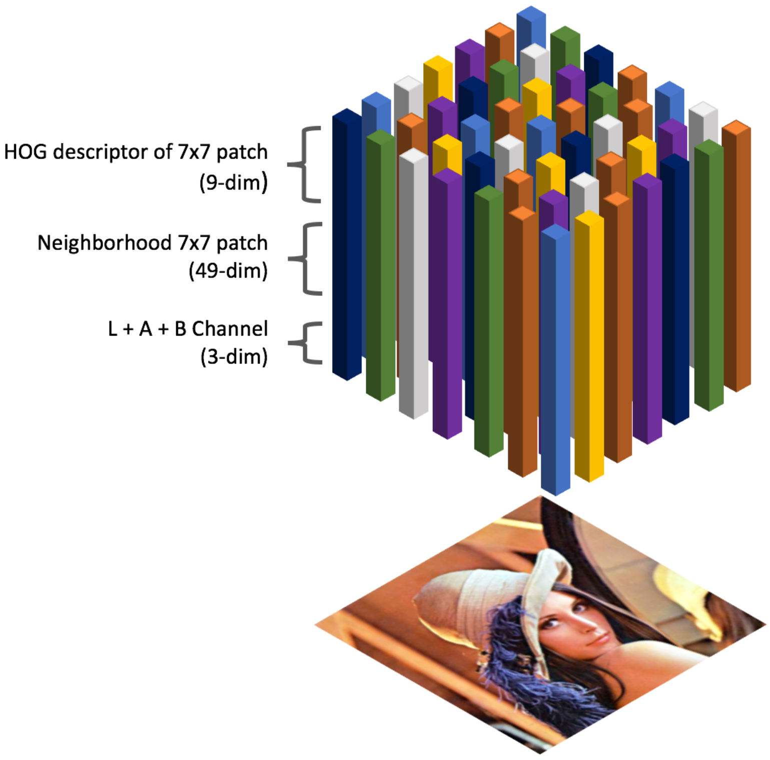

Fig. 2. Feature Extraction Pipline of our methods

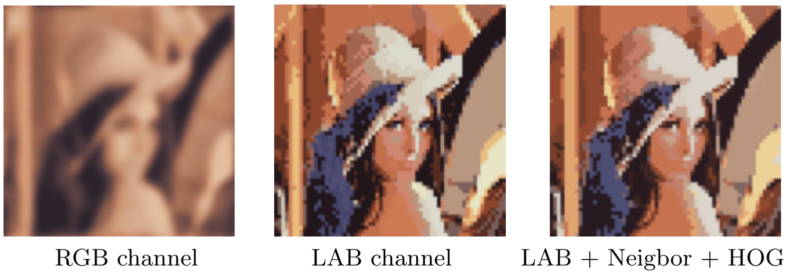

Fig. 3. Comparision of Result using different feature set (64×64, 16 colors, nearest-neighbor downsampling)

K-means Clustering After feature extraction, each pixel is represent by a feature vector, we can then using their corresponding feature vector to do clustering. We proposed to implement clustering via k-means. The k-means clustering algorithm is used to partition a given set of observations into a predefined amount of k clusters. In our case, k equal to the target color palette size of the color quantization stage, usually only dozens.

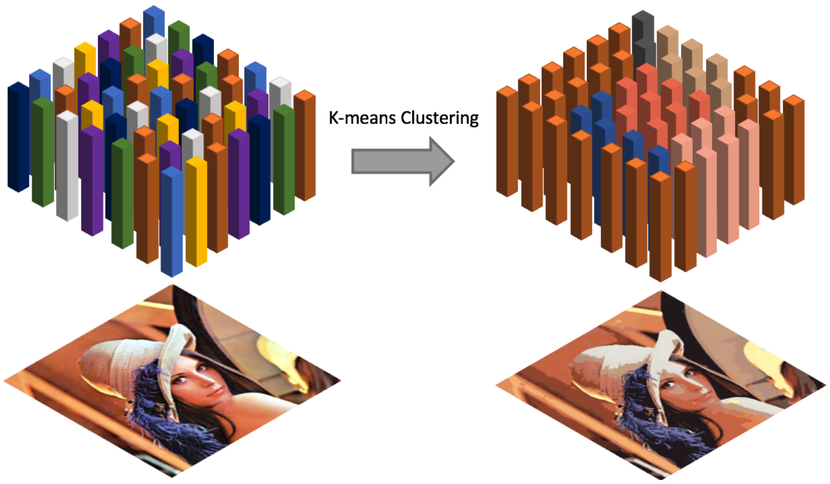

Fig. 4. An illustration of k-means clustering in color quantization

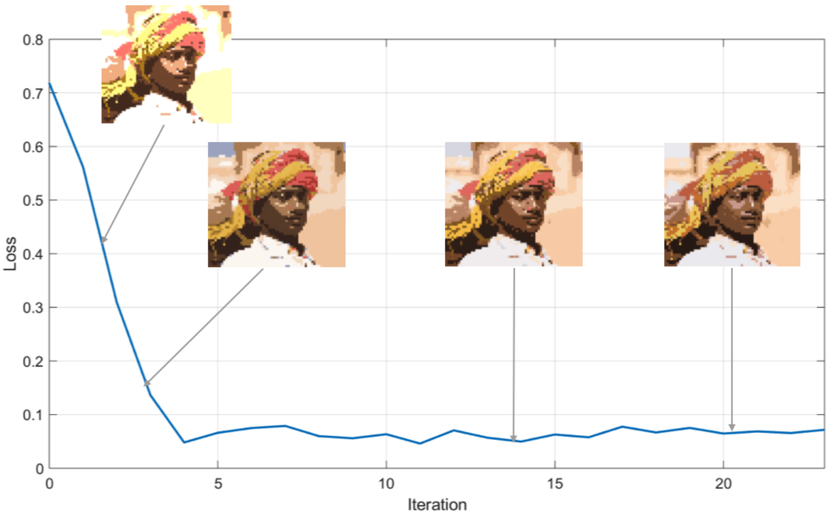

Fig. 5. The loss function of k-means clustering vs quantization result. We can observe that as the loss decreasing, the quantization result becomes better

The k-means algorithm starts with a random set of k center-points (μ). During each update step, all observations x are assigned to their nearest center-point (as shown in equation 1).

$$S_i^{(t)} = \{x_p : \left\|x_p - \mu_i^{(t)}\right\|^2 \leq \left\|x_p - \mu_j^{(t)}\right\|^2 \; \forall j, 1 \leq j \leq k\} \tag{1}$$

Afterwards, the center-points are repositioned by calculating the mean of the assigned observations to the respective center-points.

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{2}$$

The update process performs repeatedly until all observations remain at the assigned center-points and therefore the center-points would not be updated anymore.

After k-means clustering, the feature vector of each pixel becomes the centroid of its belonging cluster, we then truncated the feature vector by taking its first 3 dimension to recover the L, A, B channel of the image. An illustration of k-means clustering in color quantization is shown in Figure 4 and its corresponding loss function in training is shown in Figure 5.

Total Variation Denosing After the pixel-level clustering, we found there might be some nosiy pixel (outliers) in the image. We proposed to tackle it though total variation denosing. The task of total variation denosing is define as following. Given a 2D digital signal y, such as images. The total variation is define as:

$$V(y) = \sum_{ij} \sqrt{|y_{i+1,j} - y_{i,j}|^2 + |y_{i,j+1} - y_{i,j}|^2} \tag{3}$$

The standard total variation denoising problem is of the form:

$$\min_y E(x, y) + \lambda V(y) \tag{4}$$

where E is the 2D L2 norm. Solving this denoising is non-trivial and the algorithm i used to solves this is known as the primal dual method. We compare the result of with and without Total Variation Denosing and show that the denosing process would remove nosiy pixels.

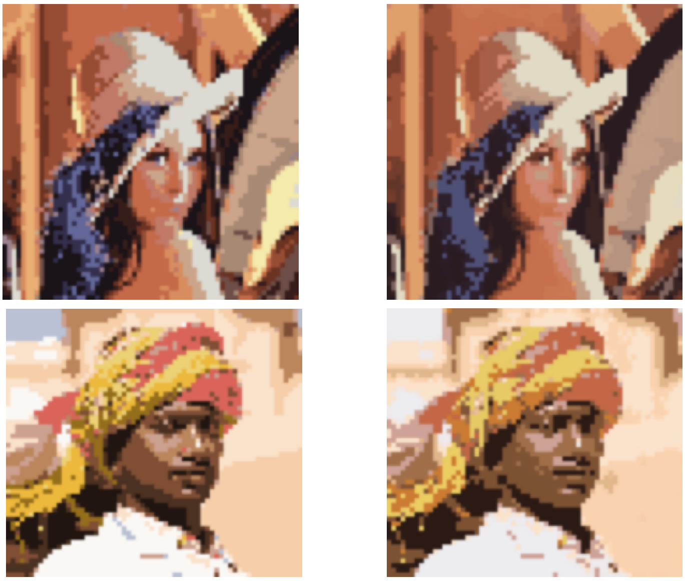

Fig. 6. Comparison of results, with and without total variation denosing (64×64, 16 colors, nearest-neighbor downsampling)

3.2 Image Downsampling

We compare different interpolation methods in image downsampling as shown in Figure 7, We can see that the bicubic and bilinear methods tend to generate blurry images. The nearest-neighbor methods is the best fit for pixelated image abstraction since i can still produce sharp images in very low resolution of 64×64.

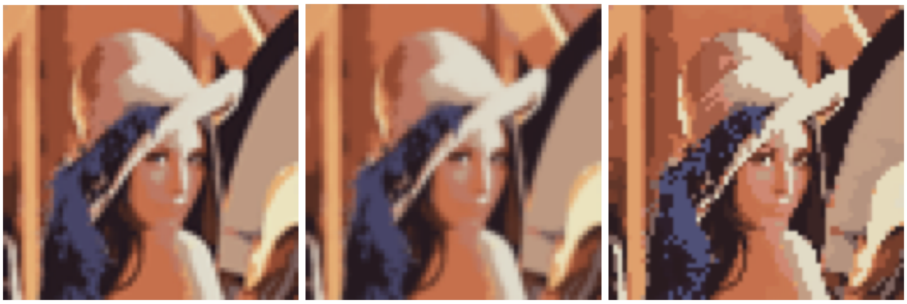

Fig. 7. Comparison of results using different interpolation methods in image downsampling, we can see that nearest-neighbor methods best fit the task od pixelated image abstraction (64×64, 16 colors)

3.3 Testing Images

Due to the lake of suitable dataset for pixelated image abstraction and for sake of fair comparison, our dataset only consist of 10 image example images from [5] in which the author used as the basline of generated pixel arts.

4 Qualitative Analysis

We compare our methods against the methods in [5]. Some qualitative comparision is shown in Figure 8. We can see that our result is comparable with that in [5]. Hence, our methods is much easier to implement due to its simple pipline.

5 Quantitative Analysis

Running Time Through our experiment on a i7-7700 machine, we found that our methods is 3-4x faster (˜5s vs ˜18s) than [5] when measuring the performmance a 500×500 image converting into a pixel art of 64×64, 16 colors.

Users Study To quantitative compare the result of our methods with [5], we invited 8 students in our lab to do a simple user study. The user study is in form of a blinded test, i.e. given two pixel-art images and its corresponding original image, subject is asked two questions:

(a) Which image can better represent the origninal one?
(b) Which image is better quality pixel-art?

A total of 8 students and 10 images is used and there are 80 votings for each question.

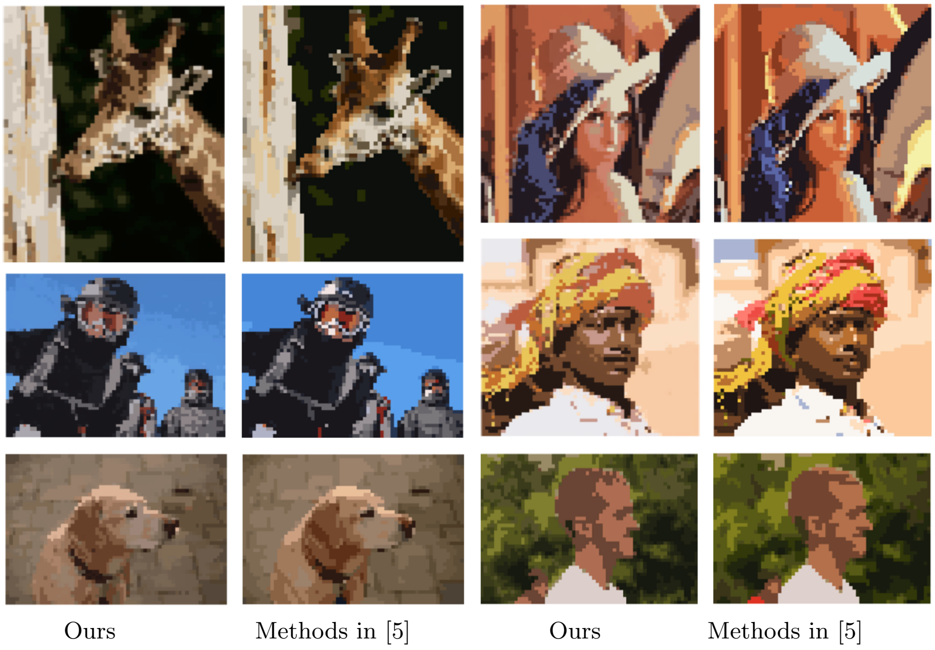

Fig. 8. Comparison of results with [5]

The voting result is shown in Table 1, we can see that ours methods looks closer to the original image (keep more semantic information) while still lacking some style of pixel art as compare to [5].

Question	Represent the origninal one?	Better quality pixel-art?
Ours	**46**	31
Methods in [5]	34	**49**

Table 1. The result of user study

References

1. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., Santella, A.: Suggestive contours for conveying shape. In: ACM SIGGRAPH 2003 Papers. SIGGRAPH '03, New York, NY, USA, ACM (2003) 848–855
2. Gooch, B., Coombe, G., Shirley, P.: Artistic vision: Painterly rendering using computer vision techniques. In: Proceedings of the 2Nd International Symposium on Non-photorealistic Animation and Rendering. NPAR '02, New York, NY, USA, ACM (2002) 83–ff
3. DeCarlo, D., Santella, A.: Stylization and abstraction of photographs. In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '02, New York, NY, USA, ACM (2002) 769–776
4. Winnemöller, H., Olsen, S.C., Gooch, B.: Real-time video abstraction. In: ACM SIGGRAPH 2006 Papers. SIGGRAPH '06, New York, NY, USA, ACM (2006) 1221–1226
5. Gerstner, T., DeCarlo, D., Alexa, M., Finkelstein, A., Gingold, Y., Nealen, A.: Pixelated image abstraction. In: Proceedings of the Symposium on Non-Photorealistic Animation and Rendering. NPAR '12, Goslar Germany, Germany, Eurographics Association (2012) 29–36
6. Achanta, R., Smith, K., Lucchi, A., Fua, P., Ssstrunk, S.: Slic superpixels
7. Rose, K.: Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In: Proceedings of the IEEE. (1998) 2210–2239